2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)

Three Key Checklists and Remedies for Trustworthy Analysis of Online Controlled Experiments at Scale

Aleksander Fabijan
Malmö University
Malmö, Sweden
aleksander.fabijan
@mau.se

Pavel Dmitriev
Outreach.io
Seattle WA, USA
pavel.dmitriev
@outreach.io

Helena Holmström Olsson
Malmö University
Malmö, Sweden
helena.holmstrom.olsson
@mau.se

Jan Bosch
Chalmers University of Tech.
Göteborg, Sweden
jan.bosch
@chalmers.se

Lukas Vermeer
Booking.com
Amsterdam, Netherlands
lukas.vermeer
@booking.com

Dylan Lewis
Intuit
San Diego, USA
dylan_lewis
@intuit.com

***Abstract*— Online Controlled Experiments (OCEs) are transforming the decision-making process of data-driven companies into an experimental laboratory. Despite their great power in identifying what customers actually value, experimentation is very sensitive to data loss, skipped checks, wrong designs, and many other 'hiccups' in the analysis process. For this purpose, experiment analysis has traditionally been done by experienced data analysts and scientists that closely monitored experiments throughout their lifecycle. Depending solely on scarce experts, however, is neither scalable nor bulletproof. To democratize experimentation, analysis should be streamlined and meticulously performed by engineers, managers, or others responsible for the development of a product. In this paper, based on synthesized experience of companies that run thousands of OCEs per year, we examined how experts inspect online experiments. We reveal that most of the experiment analysis happens before OCEs are even started, and we summarize the key analysis steps in three checklists. The value of the checklists is threefold. First, they can increase the accuracy of experiment set-up and decision-making process. Second, checklists can enable novice data scientists and software engineers to become more autonomous in setting-up and analyzing experiments. Finally, they can serve as a base to develop trustworthy platforms and tools for OCE set-up and analysis.**

Keywords*— *'Online Controlled Experiments', 'A/B testing', 'Experiment Checklists'

I. Introduction

"The road to hell is paved with good intentions and littered with sloppy analysis." – Anonymous

Online Controlled Experiments (OCEs) are becoming a standard operating procedure in data-driven software companies[1]–[4]. When executed and analyzed correctly, OCEs deliver many benefits. For example, experiments increase the quality of the product, enable feature teams to learn what changes and features are bad for the users of the product, and align the organization around a unifying goal [4], [5].

To learn about the impact of an experiment, OCEs need to be accurately analyzed. This may not seem difficult at first, however, analyzing experiments at large scale is challenging [6], [7]. To arrive at an informed decision whether to ship or not to ship a new feature, many steps need to be meticulously executed in the correct order by an experienced analyst. For example, before an experiment is started, analysts need to determine the design of the experiment (e.g. A/B or multivariate), the audience that will be exposed to the experiment (e.g. US market or Mobile users), the desired sample size, etc. During the experiment, analysts will examine specific metrics for business harm, and after the experiment is completed a stringent process of arriving to a ship/no-ship decision will be applied. The critical insights needed for an informed business decision may be buried and can only be discovered through a deep-dive analysis that spans through multiple heterogeneous segments of users [8]. One way to mitigate analysis challenges is to develop tools and procedures that support companies in the analysis process [8]–[10]. However, a fundamental understanding of what to examine and in what order is critical even for mature companies that apply automated analysis and examination tools [11], [12].

The importance of accurate experiment analysis is even greater for companies and feature teams that are new to experimentation. As more and more companies adopt the scientific method in their product development [1]–[3], the likelihood of analytical ineptitude – the lack of skill to correctly apply stringent experiment analysis – is high. Data scientists [13] with skills and experience to execute and analyze experiments are scarce. Although data scientists are already one of the hardest profession to hire [14], many of them specialize in Machine Learning and A.I. where experimentation skills are secondary to their other tasks. And since some of the steps in experiment analysis can be automated through tooling, it may be tempting to do that instead. However, great care needs to be taken to transparently present experiment results to analysts using those tools [11]. For example, some experimentation

978-1-7281-1760-7/19/$31.00 ©2019 IEEE
DOI 10.1109/ICSE-SEIP.2019.00009

platforms reveal only the outcomes. Other information such as the underlying assumptions about the representativeness of the sample in an experiment are left implicit even though experiment analysts should have known to what extent they were satisfied. Failing to recognize a violated assumption can (confidently) steer the product team into making wrong conclusions and introducing harm to business [11]. Bad data can be actively worse than no data, making practitioners blinded with pseudoscience.

"To call in the statistician after the experiment is done may be no more than asking him to perform a post-mortem examination: he may be able to say what the experiment died of." -- Ronald Fisher

In this paper, we present the most critical steps of experiment analysis at Microsoft, and at a number of other large-scale companies that run OCEs. The research question that we aim to answer is ***"How can Online Controlled Experiments be reliably analyzed?"***. Our solution to the aforementioned challenges is ingeniously simple, yet tremendously powerful. To guarantee high level of correctness in experiment analysis, we provide product development practitioners involved in the experimentation process with the same mechanism that pilots, doctors, and even market investors rely on a daily basis – a checklist.

The answer to our research question and our main contribution is a set of inductively derived checklists that can be used to analyze experiments throughout their lifecycle. Our checklists are a synthesis of experiment analysis insights from several software companies. In particular, we provide a checklist for experiment analysis in the design stage, for experiment analysis in the execution stage, and a checklist for experiment analysis when experiments are completed. In addition, we provide a list of common symptoms that make the experiment execution process unhealthy, contributing to invalid experiment analysis.

The value of our contribution is threefold. First, checklists can increase the accuracy of experiment set-up and decision making process. Second, they can enable non-experienced data-scientists, software engineers, and product managers to become more autonomous in analyzing experiments. Finally, checklist can be automated and our results can serve as a base for building more reliable tooling. Ultimately this should lead to a greater adoption of OCEs and increase the trustworthiness of OCEs.

The remainder of this paper is organized as follows. In Section II, we present the relevant background and prior works. In section III, we briefly discuss our method which we used to derive our main contribution presented in section IV. We discuss our work in Section V and conclude the paper in Section VI.

II. Background

A. Experimentation in Software Product Development

In software development, the term "experimentation" can be used to describe different techniques for exploring the value of the changes introduced to a product [15]. For example, experimentation could refer to iterations with prototypes in the startup domain [16], [17], canary flying [18] of software features (exposing a small percentage of users to a new feature or a change), gradual rollout [18] (deploying a change to one customer groups and expanding to the next one), dark launches [19] (releasing new features disabled and testing them in production), and controlled experimentation [20] - releasing multiple variants of the product and evaluate the differences between them through statistical tests. In this paper, and when we discuss OCEs, we refer to the latter – the scientifically proven technique of randomized clinical trials [21] in an online setting, which we briefly introduce next.

B. Online Controlled Experiments

The theory of controlled experiments dates back to Sir Ronald A. Fisher's experiments at the Rothamsted Agricultural Experimental Station in England during the 1920s [20]. In the simplest controlled experiment, two comparable groups are created by randomly assigning experiment participants to either of them; the control and the treatment. The only difference between the two groups is a change X. For example, if the two variants are software products, they might have different design solutions or communicate with a different server. If the experiment were designed and executed correctly, the only thing consistently different between the two groups of participants is the change X. External factors such as seasonality, impact of other product changes, competitor moves, etc. are distributed evenly between control and treatment. Hence any difference in metrics between the two groups must be due to the change X (or random chance, that is rejected as being unlikely using statistical testing). This design establishes a causal relationship between the change X made to the product and changes in user behavior, measured through metrics.

C. Experiment Lifecycle

In our previous research [22], we introduced the Experiment Lifecycle – the three main stages of every Online Controlled Experiment. We visualize it on Figure 1 below.

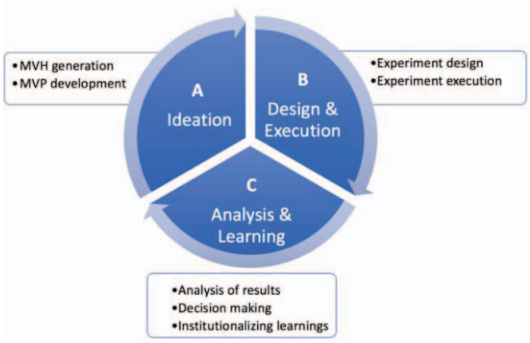

Figure 1. The Experiment Lifecycle [22].

Based on the experience of analyzing hundreds of online controlled experiments at Microsoft, we identified that every experiment first enters the *A - Ideation phase*. In this phase, practitioners propose changes to the product to evaluate their impact. The second phase is the *B- Design & Execution phase* where experiment ideas are transformed into experiments. Here, for example, the power that will be needed to detect the expect change is calculated, along with many other checks that are increasing the likelihood of a valid experiment. Finally, in the *C - Analysis & Learning phase* experimenters examine the outcome of an experiment after it is stopped.

In this paper, we focus on the latter two phases by inducing checklists that apply there. While the ideation process of selecting the idea to experiment with, out of many possible ideas, is an important aspect and that experimentation helps inform, the scope of this paper focuses more narrowly - on the process for analyzing a concrete experiment.

D. Trustworthy Growth of Experimentation

The analysis of an isolated experiment as described above is relatively simple. Correctly analyzing many experiments in a production environment, however, is challenging. Prior work both from us [10], [23] and from other researchers [24]–[26] and companies such as Booking.com [11], [27], LinkedIn [28], Google [29] and Facebook [19] stress that the growth of experimentation is conditional on the correct execution of the scientific method. For example, in the Experimentation Growth model (EG model) presented in [30], we highlighted that experimentation typically starts within an isolated feature team, and quickly grows to other teams, departments and products within an organization. The evolution is a journey that spans over several stages of experimentation maturity. One of the dimensions of the Experimentation Growth model that fuels this growth is the Feature team self-sufficiency – the extent to which individual feature teams manage experimentation set-up and analysis tasks. In the initial 'Crawl' stage of the EG model, this level is low as the teams rely on diligent experimentation scientist. In contrast, in the final 'Fly' stage, most of the experiments should be set-up and analyzed without experienced data-scientist involvement.

The research question in this paper is related to the velocity of advancing along this dimension, thereby accelerating trustworthy growth of experimentation in software companies. And our answer is to employ checklists.

E. Checklists

Checklists are an essential tool to prevent humans from overlooking critical items. They are used by researchers and practitioners of various experience levels and on all types of positions and industries. For example, the effectiveness of checklists has been reported both in qualitative research [31], as well as in quantitative studies such as Randomized Clinical Trials [32]. Furthermore, checklists help pilots dozens of time a day to guarantee a safe flight, and doctors in an operating room to overcome common complications.

The reason why checklists are so effective is that they solve the two major challenges of human beings. These are human fallibility and tendency to skip steps [33]. In online controlled experiment analysis, a forgotten step or a wrong interpretation of the data can have devastating consequences for the business. For example, a failed data quality check can make the results of an experiment invalid. Overlooking the failed test could lead the team into believing that the results can be trusted and pivoting the product into the wrong direction. Therefore, by inducing a set of checklists through case study research described in the next section, we hope to increase the reliability and comprehensiveness of experiment analysis.

III. Research Method

In this section, we describe our research approach, present the validity concerns, and state our mitigations for improving the generalizability of this research.

A. Research Approach

The research presented in this paper is a result of ongoing work with nine case companies that all run hundreds or thousands of online controlled experiments every year. We organized our work in three main phases, for which we present our data-collection and data-analysis approaches as suggested by Runeson and Höst [34] below.

Phase1: Our first step was a case study at Microsoft Corporation, which we conducted between April 2016 and August 2018. Our main data collection in this phase were (1) *experiment analyses observations* and (2) *documentation analysis.* The experiment analyses observations were both opportunity-based, as well as structured studies. In particular, the first author of this paper observed and recorded (screen capture and audio recording) 22 practitioners that work at the Analysis & Experimentation team while they examined real online controlled experiments (on average 5 per participant). This was a study that lasted 30minutes on average with every participant, with a goal to identify how analysts examine experiments. Practitioners working at Analysis and Experimentation (A&E) team are domain experts and work with experiments on a daily basis. To record as many insights as possible, the participants were asked to speak out loud during the study. The first author of this paper took notes during experiment examinations and used them while examining the recordings to elicit the key steps and items that individuals examined in every experiment. In addition, the first two authors worked at the case company and participated in hundreds of experiment review meetings where practitioners from product teams were sharing their learnings, pitfalls, and OCE analyses.

Phase2: To detail, increase the correctness, and improve the generalizability of the findings induced in phase 1, the first two authors of this paper (1) designed a questionnaire (available at [35]) that asks the survey participants to reveal the key steps in their experimental analysis, and (2) jointly reviewed related literature referenced in this paper for experiment analysis steps and explanations of them. In the questionnaire, we asked 8 open-ended questions-see Appendix 1. We shared the questionnaire with contacts (experimentation experts / experimentation directors) that work at the following case companies: Airbnb, Snap, Skyscanner, Outreach.io, Microsoft, Intuit, Netflix, and Booking.com. We received a response from each of the aforementioned companies through email. Based on the information that our contacts provided in their responses, we improved the initial checklists from phase 1.

Phase3: In the final phase of this research, we iterated our working checklists with contacts at the case companies that participated in the first two stages of this research. The data collection activity in this phase were emails and LinkedIn messages. The focus of this phase was on validating the completeness of checklists (that no steps are missing), and their order (that the steps appear in the right order). We updated our report accordingly based on the input that we received from the experimentation experts.

B. Threats to Validity

In this section, we briefly discuss our efforts in mitigating the validity concerns that are applicable for this type of qualitative research [34]. In particular construct validity, external validity, and reliability.

Construct validity. The authors of this paper as well as the participants of the study are all well-familiar with online controlled experimentation and each other. For example, practitioners that participated in this study frequently collaborate with each other. Despite this, and when reaching out to practitioners in phase 2, we used a template in which we explained the study objective and terms used in our questionnaire before revealing the questions. Also, several practitioners asked for clarifying questions during phase 2 (for example, regarding the granularity of the answers that we seek). In such scenarios, we replied with a predefined template that contained a few example answers from phase 1. This increased the mutual understanding of the phenomena under investigation and partially mitigates construct validity threats.

External validity. The first two authors selected the case companies in an opportunity-based fashion. For example, the practitioners that responded to our questionnaire and later validated the findings are our acquaintances. Therefore, this research risks on being biased by this selection, as well as self-reporting bias that the practitioners might have while answering our questionnaire. Despite this, the main contribution of this work has been derived based on the experience of several, experienced large-scale online companies, each running hundreds or thousands of experiments every year. Also, the domains of the participating case companies are broad (e.g. contain search-engine companies, accommodation providers, streaming services, etc.). Considering this, we believe that any company striving for a standardized and trustworthy experiment analysis process can benefit from our findings.

Reliability. To mitigate reliability threats, we employed constant-comparison. In particular, each interpretation and finding was compared with existing findings from previous questionnaire response, and with the analysis findings that emerged in the preceding phase. We used all of the data that we had available (comprehensive data use) for the analysis, as well as used multiple and independent analysts throughout the analysis process (analyst triangulation).

IV. Experiment Analysis Checklists

In this section, we present the key analysis steps of an experiment, and the checklists that we derived for each of them. In particular, we first provide two checklists for the Experiment Design & Execution phase of the Experiment Lifecycle illustrated on Figure 1. Second, we provide a checklist for analyzing experiments when they are completed – Experiment Debrief Analysis. Third, we provide a list of critical symptoms of unhealthy experiment execution and tips to mitigate them. Finally, we discuss how to implement the checklists in an organization. By stringently applying these checklists, companies can reduce *Experiment Harm* – the wasted time, wrong learnings, and possible wrong decisions. As discussed before, mistakes can result in very wrong decisions, not just delays but also in active harm to users and business.

A. Experiment Design Analysis

Based on our findings, there are ten items that need to be checked before an experiment can be started. We summarize them in Checklist 1 and discuss each of the items next.

EXPERIMENT DESIGN ANALYSIS

- ☑ Experiment hypothesis is defined and falsifiable.
- ☑ Experiment design to test the hypothesis is decided.
- ☑ Metrics and their expected movement are defined.
- ☑ Required telemetry data can be collected.
- ☑ The risk associated with testing the idea is managed.
- ☑ The minimum size effect and OCE duration are set.
- ☑ Overlap with related experiments is handled.
- ☑ Criteria for alerting and shutdown are configured.
- ☑ Experiment owners for contact are known.
- ☑ Randomization quality is sufficient.

Checklist 1. Experiment Design Analysis.

☑ **Experiment hypothesis is defined and falsifiable.** Experiment analysis begins with the transformation of an idea for testing into an experiment. The first step in this process is the impact analysis. Every change for testing should be introduced with a description of what the change that will be evaluated is (e.g. a change to ranking algorithm), who will see the change (e.g. users searching for accommodations in Europe), what the expected impact is (increase in booking rates), and how this impact is connected to the top-level business goals (increase in revenue). Most importantly, an impact analysis should contain a line of reasoning – belief - that explains why a change is expected to have an impact. Common questions that help in evaluating this item are "Why are we improving this?", "Why is existing state not good enough?", "Why will the change make a difference?", "Will it help the users, the business, or both?"

Every experiment requires a hypothesis that combines the change in an experiment with its impact and reasoning why the expectation is as it is. Although we have seen many different ways in which companies track hypothesis for experiments, the following template [36] is one of the most crisp ones: *"Based on [qualitative/quantitative] insight, we predict that [change X] will cause [impact Y]."* This template combines the expected impact Y, which should be defined as a change to a metric for a certain delta due to some change X, with the insight that drove the experimenter into designing an experiment. Combining all three into a single statement makes the hypothesis an educated guess about the expectations of the experiment.

The hypothesis needs to be defined in such a way that it can be possible, with a single or multiple of experiments, to falsify it – show that it was wrong. In the contest of experimentation, to falsify a hypothesis typically means ability to express the hypothesis as a metric and having sufficient telemetry to compute that metric. For example, most web sites do not have profiles containing gender information for all their visitors. For such a site, an example of a non-falsifiable hypothesis is: "this redesign of the site will make it more

appealing to female visitors". There is no way to disprove such hypothesis in the absence of accurate gender information. In contrast, a good falsifiable hypothesis could be "this redesign of the site will increase visits to the site". Visits can usually be accurately measured and, therefore, if the redesign shows a decrease in visits the hypothesis will be falsified.

☑ **Experiment design is decided.** There are a number of different experiment designs. One of the most common approaches is the single factor OCE (SF-OCE). In an SF-OCE, one change to the product is being applied in the treatment group. This is different from a MultiVariate OCE (MV-OCE) where multiple changes are applied to the treatment group at the same time. While analysis and interpretation of MV-OCEs is more challenging compared to single change experiments, they allow for multiple changes to be tested on the same audience at the same time, and detection of interaction between them can be handled. SF-OCEs are on the other hand more convenient both from a product engineering perspective (e.g. after a single change is developed the experiment can be started without the need to wait for other factors to be developed) as well as from the analysis perspective. An extensive discussion on selecting the most appropriate design is described in [37].

☑ **Metrics and their expected movement are defined.** It is typical for novice experimenters to care about only one metric when running the experiment – the specific user behavior the feature being tests is designed to induce. Prior research, however, discusses that to gain full understanding of experiment results, in addition to measuring the direct impact of the feature (feature metrics), three other important groups of metrics need to be computed [10], [30], [38]. Data quality metrics reveal issues with the data collection such as rate of data loss or cookie churn. Guardrail metrics reveal movements in measures that represent important business constraints and which should not be impacted by experiments outside a defined band. Finally, success metrics reveal whether the feature being tested helps move the product closer to achieving long-term company goals – these metrics are key to evaluating the overall success of the experiment and are usually based on higher-level user behavior patterns not having to do with any single feature. An example of a success metric could be a number of days the user visited the site during the experiment. The following paper contains guidelines in designing the aforementioned metrics [39]. To aid in interpretation of results and to help avoid pitfalls, our advice is to present these metrics either as a balanced scorecard [9] or through intuitive visualizations such as the one proposed in [8].

☑ **Telemetry data can be collected.** Data quality is of critical importance for OCE analysis. It all starts with the source of the data – product telemetry. In most software products telemetry logging was put in place for debugging or testing purposes [40]. The challenge with this type of logging, however, is that it does not necessarily reveal how the products are used. Hard to use data limits the number and types of metrics that experiment analysts can define, and the number of ad-hoc analyses they will do, all resulting in sub-optimal decisions and fewer learnings obtained from the experiments. Our recommendation is to create a centralized catalog of log events and implement those events in product as described in [41]. In particular, events such as clicks, swipes, durations, dwell times, feature usage, exceptions, or anything else that your product can measure [42] should be captured and reliably stored for later use in metric computation.

Some experiments will also require triggered telemetry for successful analysis. Data collected from web traffic on a large site or a product used by many users typically has tremendous variability, thereby making it difficult to run an experiment with sufficient power to detect effects on smaller features. Consider for example a change that impacts only the check-out page – say a new payment option. Many users that visit web stores never arrive to the checkout page. From the analysis perspective, the data generated by these users can be considered noise. For that reason, triggered telemetry - log signals that indicate the visibility or potential visibility if in control of the new feature – needs to be implemented for features that are active or visible under special conditions.

☑ **Feature risk is managed.** Experimentation exposes new features and ideas to thousands of users. There three concerns with this – technical debt, visibility, and user impact. First, while we recommend testing the new code changes for correctness, certain things such as scalability may not necessarily need to be optimal. For example, since only a limited number of users will be exposed to a new feature in an experiment, technical debt is acceptable. However, it should be acknowledged and taken care of in a later release. Second, since the experiment exposes real users, competitors may get an early insight into the direction of the product development. In such scenarios, longer experiments with smaller audiences may be necessary. Finally, even though the experiment will run on a limited number of users, a possible very bad experience may make them churn – making a big loss.

☑ **Effect size and experiment duration are set**. Experiment analysis should contain a minimum effect size - the minimum Δ% in the metrics of interest that we care about detecting due to the new feature or change in the code, as well as the duration of the experiment. This is a trickier problem than it seems. On the one hand, the running period needs to be long enough so that experimenters can collect the data they need. On the other hand, it cannot be too long since knowing the result early is of interest for product development. In addition, as mentioned above, multiple metrics of interest are usually involved. In general, detecting smaller changes will typically require longer experiments (as more users will use each of the experiment variants), and vice-versa. Also, note that minimum effect size Δ% differs from the expected effect size. The latter is difficult to predict as it may substantially differ from experiment to experiment. Minimum effect size on the other hand is typically more consistent across experiments, determined by business requirements and/or the number of active users of the product.

Based on the minimum effect size, as well as the standard deviation which could be estimated based on historical data, experiment duration can be determined using a standard statistical formula [43]. Assuming the number of users is sufficient, a good practice is to always run experiments at least

for a week or two in order to cover weekdays, weekends, possible holidays, different traffic sources, and so on, as well as to consider the nature of the product. Other factors like apriori expectation of novelty or primacy effects in features, or warm up period needed for machine learning models may also be a factor for determining the duration of an experiment.

☑ **Overlap with related experiments is handled.** As companies grow their experimentation capabilities, several feature teams experiment at the same time. An overlapping single-factor design will therefore grow in importance. This increases the possibility of an overlap – a situation in which two or more experiments will interact and cause issues. An overlap can be handled with stringent coordination upfront, or through overlapping infrastructure described in [10], [29]. However, due to unknown and often unexpected nature of interactions among overlapping experiments, this problem cannot be completely eliminated through pre-experiment coordination and testing. Analysis to detect interactions among experiments will also need to be performed during the execution stage.

☑ **Criteria for alerting and shutdown are configured**. Some ideas may cause a significant impact on the business by e.g. unintentionally degrading the quality of the service. Therefore, it is important that experimenters or experimentation platform itself are aware of the criteria for alerting or shutting the experiments down. In contrast to experiment analysis that happens after an experiment is completed (which requires a certain amount of data-points for statistical significance), large degradations can be detected on much smaller samples. What is important, however, is that the alerting and shutdown criteria take into account the magnitude of the change in addition to its statistical significance. For example, an alert configured to shut down a web experiment when 1ms delay has been detected may be too aggressive and cause many false positives. In contrast, a large increase in timeouts may be an indicator of a serious issue which, depending on the severity, should be handled.

☑ **Experiment owners are known.** Every experiment should have a group of experiment owners that are responsible for monitoring and operations, and that can be contacted in case of a need. For example, experiment owners may start and stop the experiment, as well as act on any alerts. We recommend to always have several owners for a single experiment in order to guarantee availability of at least one in situations that may require an urgent action by an operations engineer. Several owners also increase the likelihood of knowledge transfer and sharing of learnings that is critical for growing a culture of experimentation.

☑ **Randomization quality is sufficient**. While randomization may sound simple, correctly randomizing the users into experiment variants in the presence of poor instrumentation, data loss, various corner cases like null user ids and user id churn, etc. is a challenging problem. Randomization needs to be validated and shown to be working correctly via a series of A/A tests. In particular, the distribution of p-values should be close to uniform. There are also advanced methods to deal with randomization imbalance which may happen by chance, discussed in [10].

B. Experiment Execution Analysis

During experiment analysis, we identified three items that need to be checked for correct and effective experimentation.

EXPERIMENT EXECUTION ANALYSIS
☑ No serious data quality issues are present. ☑ Experiment is not causing excessive harm. ☑ Possibility of early stopping is evaluated.

Checklist 2. Experiment Execution Analysis.

☑ **No serious data quality issues are present.** Complete and sufficient data are critical for an accurate analysis of OCEs. That said, the first item that should be checked while experiment is being executed is that the instrumentation and data pipeline are performing as designed. Data loss is present whenever client-side telemetry is used to compute metrics. Excessive amount of loss as well as possibility of treatment impacting the rate of loss can both skew experiment results. Data quality metrics that measure the amount of loss need to be created as described in [44], and overall sensitivity of results to the amount of loss needs to be understood. In particular, the following questions should be answered to satisfy this check: are the defined telemetry being logged? Is there any bias in the assignment of a variation? What is the data-loss rate in the collection process? Are there any quality issues yielding untrustworthy results?

To answer these questions, data quality tests can be executed on the collected data. One such test is the simple Sample Ratio Mismatch (SRM) which utilizes the Chi-Squared Test to compare the ratio of the observed user counts in the variants against the configured ratio.

"If metrics are down dramatically, we need to pull back and debug. Anything around logging, assignment, and data collection that's going wrong? Or is it actually delivering a bad experience?"

-- Data Scientist

☑ **Experiment is not causing excessive harm.** Many experiments result in outcomes that are different from expected. Statistics from Microsoft Bing state that only about 1/3 of ideas that people experiment with do what they were aiming to do, one third has no significant impact whatsoever, and a third of ideas results in harmful outcomes [45]. For example, some OCEs may negatively impact product quality metrics and cause a degradation of service. This can result in harm for the user (e.g. by not being satisfied with the performance of the product while in use) and for the business (e.g. by users not returning to use the product as frequently as in the control variation without the change). For these reasons, key success metrics and guardrails for the product should be examined for large degradations while the experiment is in execution.

If degradations are discovered, the experiment should be stopped before the predetermined end-time. A good practice for technically demanding experiments that may be hard to roll-back instantly is to gradually increase the size of the treatment group and examine metrics for large degradations at different stages, for example, after a few hours, a day, a week, etc.

☑ **Possibility of early stopping is evaluated.** Understandably, despite having calculated the required experiment duration prior to the experiment start, experimenters are inclined to "peek" at the results and stop the experiment early, especially in situations where treatment seems to be doing much better than originally expected. Such early stopping can make product development leaner as decisions on the features can be made earlier. However, there are several pitfalls associated with early stopping [7], [46]. For instance, simply monitoring at p-values and stopping as soon as statistical significance is reached is wrong and should be discouraged. Therefore, a decision to stop early should be made with great care. To avoid complexities with determining when early stopping may be Ok, we recommend relying on experimentation platform recommendations rather than leaving the issues in the hands of experimenters. Advanced experimentation platforms may implement statistical tests which allow for early stopping, such as [47], and notify the user when it is fine to stop early.

C. *Experiment Debrief Analysis*

For experiment analysis after the OCE is completed, we identified nine critical items to check before experiment lifecycle is closed. We list them in Checklist 3 and discuss next.

EXPERIMENT DEBRIEF ANALYSIS
☑ Experiment has sufficient power.
☑ Data quality metrics are not negative.
☑ Metrics are examined for selecting the winner.
☑ Novelty effects are excluded.
☑ In-depth analysis has been performed.
☑ Skewed data is treated.
☑ Coordinated analysis of experiments is done.
☑ Validation of experiment outcome is conducted.
☑ Experiment learnings are institutionalized

Checklist 3. Experiment Debrief Analysis.

☑ **Experiment has sufficient power.** The hypothesis defined before the experiment was started required a certain number of users to reach the required statistical power. At the end of the experiment, analysts should check if the number of units in the experiment variants is equal to or greater to the pre-calculated number. An alternative is to calculate the minimum detectable effect with the actual number of users and notify the experiment analyst if it is larger than what was initially provided. Note that observed effect size should not be used in the power calculation [48].

☑ **Data quality metrics are not negative.** The only way that experiment results can be trusted is by first examining whether there are any data quality issues that could make the analysis process invalid. For this step, the same data quality analysis is recommended as for the analysis of the experiment during the experiment execution phase.

☑ **Metrics are examined for choosing a winning variant.** Following the metric structure setup during pre-experiment analysis, the winning variant can be chosen as follows. First, data quality metrics are examined to ensure the experiment results can be trusted. Second, feature metrics are checked to ensure the feature is functioning as expected. Third, success metrics are evaluated to determine whether the feature results in long-term benefit for users and business. Finally, guardrail metrics are checked to ensure there is no large negative short-term impact on business metrics.

☑ **Novelty effects are excluded.** Certain changes to experiments can cause changes in product usage that do not replicate over time. For example, a new feature in a product might increase the usage of the product at first, however, over time the usage regresses. Although the experiment results overall are in favor of the variant with the new feature, an analysis over time can help in determining whether the impact is sustained over time.

☑ **In-depth analysis has been performed.** The decision on which variant won and should be shipped is only one benefit of experimentation. Experiments should also be analyzed in depth in order for interesting insights to be discovered, that could be used to iterate on the feature or provide ideas for new features. For example, although an experiment is neutral overall, one segment of users may have very positively reacted on the new feature, while another segment reacted negatively. These types of heterogeneous movements are common in experiments at scale and should be detected. Guidance on how to achieve this is described in [8].

☑ **Skewed data is treated.** Another item that should be checked with in-depth analysis is whether there are any outliers skewing the data. Outliers may indicate data quality issues which should be understood, fixed or dealt with by, e.g., capping. They may also indicate presence of legitimate special users. Consider for example a scenario where one user purchases 100% of items more than the average user in the group (for example, a travel agent on an accommodation booking website). In this case we do not want to cap such users. Instead, we want to create several metrics that look separately at different sub-populations or behavior patterns of users. Guidance on this process is provided in [49].

☑ **Coordinated analysis of experiments is done.** The results of our survey showed that often experiments are not executed in isolation but are a part of a coordinated initiative to answer a higher-level business question broader than a single feature. For example, a manager of a sales team may be interested in whether purchasing a video recording software and training the sales reps to record personalized videos will improve performance. To answer this question, a series of experiments needs to be run evaluating the impact of including videos in e-mails across different sales scenarios and at different stages of the sales process (initial e-mail, follow-up, closer to the deal, etc.) Such experiments need to be analyzed in coordinated fashion to arrive to an answer to the broader question being asked. A similar situation often occurs if the feature being experimented with is a follow-up from or is based on the result of an earlier experiment. In either case, it is important to view the results in context of other similar or related experiments.

☑ **Validation of experiment outcome is conducted.** Experimentation has the incredibly powerful property of reproducibility. Experiments that only marginally impact the metrics should be executed again, preferably with higher power (for example, allocating a larger number of users to the experiment variants) to validate the results and learnings.

☑ **Experiment learnings are institutionalized and shared.** The most valuable outcome of every experiment should not be the status whether the change has an impact on the product or not. The highest value are the learnings that can be captured in a series of experiments. These can be used to design future experiments and steer *Intrapreneurship* [50]. We recommend that analysts capture experiment hypothesis and outcomes, final ship decisions, links to scorecards and clearly annotated information about the scorecard that was used to make the final decision. If the experiment has changed something visually (for example, it created a UI change), screenshot should be captured if they were not recorded before it was started. Based on our data analysis, we identified three approaches to institutionalizing experiment learnings. The first approach which is applicable in the early stages of experimentation growth [30] is to manually capture learnings in a reporting tool such as Word, Power Point or Excel template. The second approach which works well for tens to hundreds of experiments yearly is to use a ticketing tool where state management and searching across different experiments is effortless. At very large scale, experiment learning capturing should be as automated as possible and preferably well integrated in the experimentation platform [10].

D. Critical Symptoms and Remedies

Throughout experiment analysis process, practitioners may experience a number of symptoms - issues that can block them from continuing to analyze the experiment. In this section, we provide three symptoms based on the experience from our case companies – one for each of the analysis checklists - and remedies to resolve them. This is not a comprehensive list, but rather a set of common issues and solutions.

Symptom 1: Lack of experiment coordination. Deciding which experiments should start next, which should be stopped, etc. can be a challenging task. Is an experiment in the queue impacting the part of the product where other experiments are already ongoing? What will the union of two experiment treatments be? In a part of the product with many ideas, which one should be experimented with next? While in the early stages of experimentation growth coordinating such experiments may be simple, performing this in a state where multiple teams are running multiple experiments can be very challenging. We recommend that product development establishes a role of **Experiment Coordinator (EC).** One of the main responsibilities of EC should be the ownership of Experiment Review Meeting.

We put the entire experimentation team into the room and confirm the Product Manager (owner), the program manager, the analyst, the developers/engineers, and the QA that worked on the experiment are present. --Principal Product Manager of Experimentation

This meeting has several goals. First, new experiment ideas are reviewed, assigned to their respective categories, and prioritized within those categories. Second, progress of the currently running experiments is reviewed and they are checked to ensure none of them need to be shut down due to clearly bad results. Third, new experiments that are ready and can be started need to be scheduled to start in the upcoming week. Fourth, the results of the recently completed experiments are reviewed and discussed, and decisions are made about promoting some variants to default and turning off others.

Symptom 2: Consistently failing A/A tests. A/A tests that consistently show non-uniform distribution of p-values are a symptom of a serious issue in the experimentation system. Our advice is not to proceed with an A/B test if a series of A/A tests fails, but rather debug the experimentation system to discover the root cause. One cause of this symptom can be the ***technical differences in how control and treatment variants are applied, independent of the treatment***. For example, experiments that redirect only the treatment group to another page will have a short delay and consequently degrade performance metrics for that variant. We recommend examining how the variations are set-up and provide a controlled environment for all of them by, if needed, causing a redirection to all of the variations. Another reason for this symptom can be ***a biased estimate of the variance***. For example, if the randomization is performed per user and the metric is aggregated per another unit different than user (e.g. a page view), the computation of the variance should be done through a delta method or through bootstrapping. Next, A/A tests can fail due to ***unequal splits of variants using shared resources*** (e.g. cache). In such situations, the larger variant will have a greater impact on the shared resource, which can negatively impact the experience for users in the smaller group. We recommend running A/A tests with equal groups. Also, A/A tests can consistently fail also due to ***practitioner actions***. For example, when a certain variation is performing poorly, someone in the company might have restarted it in the effort to improve it. Our recommendation is to provide alerts to users when an action that will likely cause a failing A/A test is triggered. The alert should notify the experimenter about the consequences or advise them to stop the experiment first before updating an individual variation.

Symptom 3: Consistently falling to decide. When at the end of an experiment that moved a success metric in a statistically significant way there is no consensus among the stakeholders that the change was actually good, practitioners will experience decision paralysis. This is the inability to decide whether the experiment was good or bad for the users: some people might argue that it was and others that it was not, there are plausible explanations for either way, there are counter-examples in the data, and perhaps at the end the decision is made to go against the results of the test. One solution to this challenge is to ***invest into designing good success metrics***. Practical guidance on how to resolve this and how to design sensitive success metrics is described in [51]. A single metric makes the exact definition of success clear, possibly through explicit weighted combination of metrics that describe the core components of the product. Of course, another solution is to be ***explicit about the ship criteria*** while designing the experiment.

"We always advise people to write their shipping criteria in such a way that a colleague could make the decision for them if they go on an unexpected vacation."

--Senior Product Owner, Experimentation

E. Implementing the Checklists

To implement the checklists in your organization, we suggest the following four steps:

1. **Identify which of the checks are automated.** Typically, experimentation platforms and commercially available tooling will automate several steps described in our checklists. Identify the checks that your platform supports, and the checks that need to be done manually.

2. **Provide analysts with annotated checklists.** Annotate each of the checklist items with [Auto-Checked] or [Manual-Check] depending on your findings from step 1. Provide the checklists with these labels to the analysts along with the instructions on how to interpret the labels.

3. **Track and display checks' status for every OCE.** The items that are supported by the tooling should be visualized to the experimenters, informing that they have either passed or failed. Just like a regular unit test. For the manual checks, provide analysts an option to tick/enter information about the status of the check.

4. **Automate as many of the checks as possible.** While relying on the checklists improves the effectiveness of the work [33], automating as many checks as possible should be one of the goals of experimentation growth.

V. Conclusion

The scale at which companies are striving to experiment is fundamentally changing. Not only is experimentation rapidly expanding within the large online software companies like Booking.com, and Microsoft [1], but also companies in other domains are catching up [2]. Without guidelines and checklists for trustworthy analysis of experiments, however, the scientific method of running OCEs will not scale in a trustworthy manner. If experiments are not analyzed correctly, the learnings from them may steer product development teams into making wrong decision, causing harm rather than adding value.

In this paper, we presented the key checklists for trustworthy experiment analysis at large scale, and three symptoms of unhealthy experiment execution. For each of the symptoms, we provided a number of remedies.

Based on our findings, we can conclude that most of the experiment analysis actually happens before the experiment is even started. This complements and contrasts the related literature (see e.g. [8], [24], [49], [52]) where experiment analysis is a discussion for the final stage of the experiment lifecycle (e.g. when an experiment is completed).

Furthermore, the checklists in this paper should be seen as a starting point towards increasing the trustworthiness of experimentation in the software industry. However, similarly to checklists in other industries, they likely need to be adapted for the company that uses them [33]. Future research needs to be done in applying these checklists at companies that were not part of this study and examining to what extent they satisfy their purpose of increasing the trustworthiness of OCEs.

Acknowledgment

We would like to acknowledge all case companies and thank all participants for contributing to the work presented in this paper and improving this research.

References

[1] A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch, "Online Controlled Experimentation at Scale: An Empirical Survey on the Current State of A/B Testing," in *Proceedings of the 2018 44rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2018.

[2] E. Lindgren and J. Münch, "Software development as an experiment system: A qualitative survey on the state of the practice," in *Lecture Notes in Business Information Processing*, 2015, vol. 212, pp. 117–128.

[3] F. Auer and M. Felderer, "Current State of Continuous Experimentation: A Systematic Mapping Study," in *Proceedings of the 2018 44rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2018.

[4] R. Kohavi and S. Thomke, "The Surprising Power of Online Experiments," *Harvard Business Review*, no. October, 2017.

[5] A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch, "The Benefits of Controlled Experimentation at Scale," in *Proceedings of the 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2017, pp. 18–26.

[6] A. Deng, P. Zhang, S. Chen, D. W. Kim, and J. Lu, "Concise Summarization of Heterogeneous Treatment Effect Using Total Variation Regularized Regression," *Submiss.*, Oct. 2016.

[7] P. Dmitriev, S. Gupta, K. Dong Woo, and G. Vaz, "A Dirty Dozen: Twelve Common Metric Interpretation Pitfalls in Online Controlled Experiments," in *Proceedings of the 23rd ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '17*, 2017.

[8] A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch, "Effective Online Experiment Analysis at Large Scale," in *Proceedings of the 2018 44rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2018.

[9] R. S. Kaplan and D. P. Norton, "The Balanced Scorecard: Translating Strategy Into Action," *Harvard Business School Press*. pp. 1–311, 1996.

[10] S. Gupta, L. Ulanova, S. Bhardwaj, P. Dmitriev, P. Raff, and A. Fabijan, "The Anatomy of a Large-Scale Experimentation Platform," in *2018 IEEE International Conference on Software Architecture (ICSA)*, 2018, no. May, pp. 1–109.

[11] T. Kluck and L. Vermeer, "Leaky Abstraction In Online Experimentation Platforms: A Conceptual Framework To Categorize Common Challenges," Oct. 2017.

[12] D. I. Mattos, J. Bosch, and H. H. Olsson, "Challenges and Strategies for Undertaking Continuous Experimentation to Embedded Systems: Industry and Research Perspectives," in *19th International Conference on Agile Software Development, XP'18*, 2018, no. March, pp. 1–15.

[13] 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.

[14] S. Miller and D. Hughes, "The Quant Crunch: How the Demand For Data Science Skills is Disrupting the Job Market," *Burning Glass Technologies*, 2017.

[15] G. Schermann, J. J. Cito, and P. Leitner, "Continuous Experimentation: Challenges, Implementation Techniques, and Current Research," *IEEE Softw.*, vol. 35, no. 2, pp. 26–31, Mar.

2018.
[16] E. Ries, *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. 2011.
[17] S. Blank, "Why the lean start-up changes everything," *Harvard Business Review*, vol. 91, no. 5. John Wiley & Sons, p. 288, 2013.
[18] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. 2010.
[19] D. G. Feitelson, E. Frachtenberg, and K. L. Beck, "Development and deployment at facebook," *IEEE Internet Comput.*, vol. 17, no. 4, pp. 8–17, 2013.
[20] J. F. Box, "R.A. Fisher and the Design of Experiments, 1922–1926," *Am. Stat.*, vol. 34, no. 1, pp. 1–7, Feb. 1980.
[21] S. D. Simon, "Is the randomized clinical trial the gold standard of research?," *J. Androl.*, vol. 22, no. 6, pp. 938–943, Nov. 2001.
[22] A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch, "The Experiment Lifecycle," *Accept. to Appear IEEE Softw.*, 2018.
[23] A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch, "The Evolution of Continuous Experimentation in Software Product Development," in *Proceedings of the 39th International Conference on Software Engineering ICSE'17*, 2017.
[24] K. Kevic, B. Murphy, L. Williams, and J. Beckmann, "Characterizing Experimentation in Continuous Deployment: A Case Study on Bing," in *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, 2017, pp. 123–132.
[25] 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.
[26] R. Kohavi, A. Deng, B. Frasca, R. Longbotham, T. Walker, and Y. Xu, "Trustworthy online controlled experiments," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '12*, 2012, p. 786.
[27] R. L. Kaufman, J. Pitchforth, and L. Vermeer, "Democratizing online controlled experiments at Booking. com," *arXiv Prepr. arXiv1710.08217*, pp. 1–7, 2017.
[28] Y. Xu, N. Chen, A. Fernandez, O. Sinno, and A. Bhasin, "From Infrastructure to Culture," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15*, 2015, pp. 2227–2236.
[29] D. Tang, A. Agarwal, D. O. Brien, M. Meyer, 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] A. Fabijan, P. Dmitriev, C. McFarland, L. Vermeer, H. Holmström Olsson, and J. Bosch, "Experimentation growth: Evolving trustworthy A/B testing capabilities in online software companies," *J. Softw. Evol. Process*, p. e2113, Nov. 2018.
[31] R. Power and B. Williams, "Checklists for improving rigour in qualitative research," *BMJ*, vol. 323, no. 7311, pp. 514–514, Sep. 2001.
[32] D. Moher, A. R. Jadad, G. Nichol, M. Penman, P. Tugwell, and S. Walsh, "Assessing the quality of randomized controlled trials: An annotated bibliography of scales and checklists," *Control. Clin. Trials*, vol. 16, no. 1, pp. 62–73, 1995.
[33] A. Gawande, *Checklist manifesto, the (HB)*. Penguin Books India, 2010.
[34] 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.
[35] A. Fabijan and P. Dmitriev, "Experiment Analysis Questionaire," 2018. [Online]. Available: http://www.fabijan.info/papers/ICSE_ExP_Analysis_Questionnaire.pdf.
[36] "Hypothesis Kit for A/B testing." [Online]. Available: http://www.experimentationhub.com/hypothesis-kit.html.
[37] 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, no. 1, pp. 140–181, Feb. 2009.
[38] A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch, "The Evolution of Continuous Experimentation in Software Product Development: From Data to a Data-Driven Organization at Scale," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, 2017, pp. 770–780.
[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] 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.
[41] T. Barik, R. DeLine, S. Drucker, and D. Fisher, "The bones of the system," in *Proceedings of the 38th International Conference on Software Engineering Companion - ICSE '16*, 2016, pp. 92–101.
[42] D. W. Hubbard, *How to measure anything: Finding the value of intangibles in business*. John Wiley & Sons, 2014.
[43] R. B. Bausell and Y.-F. Li, *Power analysis for experimental research: a practical guide for the biological, medical and social sciences*. Cambridge University Press, 2002.
[44] J. Gupchup *et al.*, "Trustworthy Experimentation Under Telemetry Loss," in *to appear in: Proceedings of the 27th ACM International on Conference on Information and Knowledge Management - CIKM '18*, 2018.
[45] R. Kohavi, B. Frasca, T. Crook, R. Henne, and R. Longbotham, "Online experimentation at Microsoft," in *Workshop on Data Mining Case Studies and Practice*, 2009.
[46] T. Crook, B. Frasca, R. Kohavi, and R. Longbotham, "Seven pitfalls to avoid when running controlled experiments on the web," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*, 2009, p. 1105.
[47] A. Deng, J. Lu, and S. Chen, "Continuous Monitoring of A/B Tests without Pain: Optional Stopping in Bayesian Testing," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2016, pp. 243–252.
[48] J. M. Hoenig and D. M. Heisey, "The abuse of power: The pervasive fallacy of power calculations for data analysis," *Am. Stat.*, vol. 55, no. 1, pp. 19–24, 2001.
[49] R. Kohavi, A. Deng, R. Longbotham, and Y. Xu, "Seven rules of thumb for web site experimenters," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, 2014, pp. 1857–1866.
[50] K. V. Desouza, "Intrapreneurship - Managing ideas within your organization," *Technol. Forecast. Soc. Change*, vol. 91, pp. 352–353, 2014.
[51] P. Dmitriev and X. Wu, "Measuring Metrics," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management - CIKM '16*, 2016, pp. 429–437.
[52] A. Deng, J. Lu, and J. Litz, "Trustworthy Analysis of Online A/B Tests," *Proc. Tenth ACM Int. Conf. Web Search Data Min. - WSDM '17*, pp. 641–649, 2017.