A/B Testing - A Search Based Approach

Hannes Rabo
[hrabo@kth.se]

Philippa Örnell
[pornell@kth.se]

***Abstract*—A/B testing is an increasingly popular way of creating an environment of continuous experimentation. It can provide a lot of insight into user behaviour and preference, however, it comes with a lot of challenges. By reformulating A/B testing as an optimization problem, search based software engineering practices such as genetic algorithms can be applied, and to some extent help in the creation of tests. This new area of automated tests shows promise for the future, while still having some unsolved issues such as creation of complex combinations of features and dependency detection among them.**

I. Introduction

Continuous experimentation together with a range of current automation tools can be used to create automated testing environments with both high business effectiveness as well as highly efficient use of development resources. This type of technology is however not available to all companies as it has certain requirements that can delay adoption among both small and larger actors. With recent research around ways experiments could be automatically constructed and scheduled, it may become much more accessible and applicable to a wider range of use cases and companies.

Three common release methods for testing software that are used today are *canary releases*, *dark launches* and *A/B-testing*, all of which are illustrated in figure 1. These techniques have slightly different goals and should be used appropriately. Canary releases are smaller releases to a subset of the user group, while dark launches exposes a running services implementation to user data in parallel to a working solution to make sure that the new implementation produces the correct output compared to known working version. Finally, A/B tests compare two versions of a software product to find which one better aligns with user preferences [8]. To narrow the scope of this paper, our primary focus is to explore current research within continuous experimentation with A/B testing as those experiments are more complex in their practical implementation.

The practice of continuous delivery and deployment is one of the enabling factors for continuous experimentation, which is why those trends have become popular at around the same time [8]. Continuous experimentation practices, such as A/B testing, give companies the opportunity to test new features in the production environment in order to get fast insights into product performance [7]. This way of live testing fosters more data-driven decisions and can ultimately bring a lot of business value. As an example, Microsoft explains that by running controlled experiments at Bing, they got improvements in a lot in both usability, performance as well as revenue [5]. Netflix also reports that they thoroughly use A/B testing. All product changes goes through rigorous A/B testing before becoming a part of the default user experience [2].

* Paper in course *DD2482 Automated Software Testing and DevOps*, KTH - Royal Institute of Technology, April 2019

II. A/B Testing

In the following section, we provide background information on what A/B testing is, the use of it and some problems that arise in the context of conducting continuous experimentation.

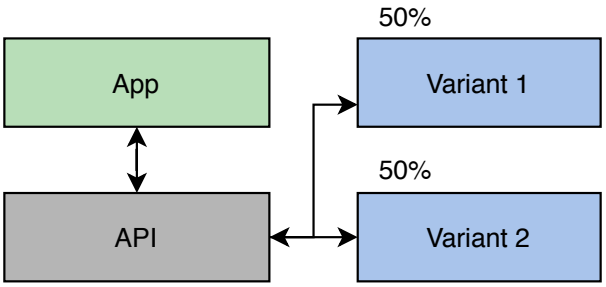
In A/B testing, also called a controlled experiment, different variants of an application are run in parallel and evaluated against each other. Users are randomly assigned to be exposed to one of the variants and their interactions are observed. The different versions are then compared and the variant that maximizes any metric of interest are chosen as the new version for the software [9]. The metrics of interest could for example be conversion rate or number of clicks. The goal of A/B testing is thus to evaluate the performance of the different versions and determine if the results of the experiment are great enough to be statistically significant and motivate a change or addition of a new feature [8].

In general, all testing techniques from figure 1 can be classified based on two different goals, either regression-driven experiments, or business-driven experiments. Regression-driven experiments tries to find functional defects such as algorithmic or performance issues, while business-driven experiments tries to answer the question regarding which options are better at delivering business value. A/B-testing is optimally applied to the later case while the first two are more suited for regression-driven tests [8].

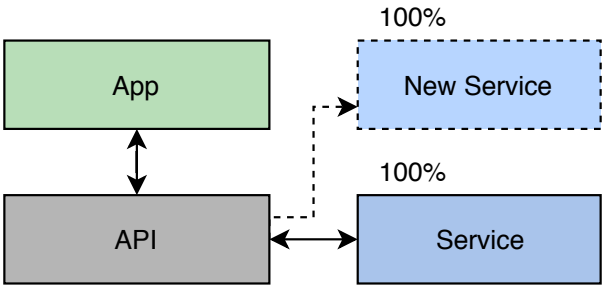
A. Challenges

A/B testing has a lot of benefits, but we need to be aware of the shortcomings of it to select and use it with optimal performance. It would not be possible to provide a complete list, however, some of the challenges mentioned in literature are summarized in table I and a short introduction is provided below.

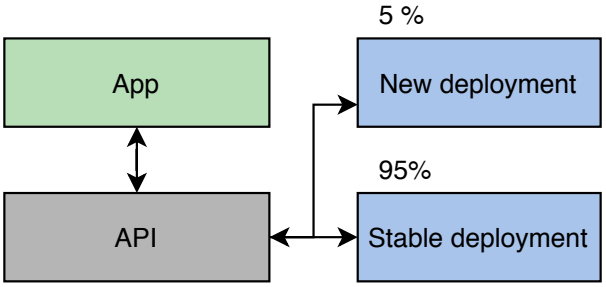
In the recent research paper *"We're doing it live: A multi-method empirical study on continuous experimentation"* [8]



(a) A/B testing - comparing different parallel versions.



(b) Dark Launching - copying live traffic to a service that never responds to the users.



(c) Canary Release - releasing to a small user group to test that everything works as expected.

Fig. 1: Different release strategies

the authors explore the actual state of practice in continuous experimentation. In this empirical study they interviewed developers and release engineers to find out what is needed for organizations to leverage continuous experimentation. The authors report on the main obstacles and challenges related to running business-driven experiments. One thing that they found to hinders, is that not all companies actually have a software architecture that support running and comparing two or more versions in parallel. For smaller businesses it was also found that the small user base made it impossible to create statistically sound tests with limited experience. A lack of expertise to analyze the results were generally not perceived as a large problem, but still, 15% of the respondents mentioned this as a cause for not conducting A/B testing. 33 % of the respondents also mentioned that the return on investment (ROI) would be too low and that it would not be worth the effort needed.

The way tests are evaluated is increasingly important with higher frequencies of experiments as that requires more data to be collected. Without leveraging all user data to the maximum, smaller companies would not have enough user data to perform continuous experiment with all relevant features. However, Kharitonov et. al. found that by careful construction of metrics, it was possible to achieve fixed level of confidence with less than a fifth of the data compared to what was required for simpler click based metrics [4]. Finding such metrics that fits the specific case can on the other hand be hard for companies with limited resources.

It should also be noted that many of these measurements are very sensitive to small changes in software behaviour and subsequently also external factors. Making sure that real statistics and sufficient experiment isolation is used is therefore of high importance in the case of A/B testing [6].

B. The Future of A/B Testing

Fabijan et al. points out that controlled experimentation is becoming the norm in advanced software companies, and for *"reliably evaluating ideas with customers in order to correctly prioritize product development activities."* [1]. Fabijan further explains that many companies within software development are moving in the direction of becoming more data-driven and are aware of the competitive edge that A/B testing brings, but that it exists difficulties to actually adopt the methodology and use it efficiently. The companies have the data but the transformation into data-driven organizations still stays low [1].

Tamburelli and Margara has the same viewpoints, and argues that even if A/B testing is widely adopted by the industry, it is still considered a *"complex and crafted activity rather than a well-established software engineering practice."* [9]. They want to address this issue by automating the A/B tests. They are not the only ones thinking this way and current research investigating automating A/B tests has started to emerge, although the field is still in its infancy. We will present some of this research in the next section.

III. Achieving Automated A/B Tests

The challenge of creating a well functioning automated test system is twofold. Firstly, all the basic requirement and preconditions for implementing A/B tests in general has to be fulfilled (as shown in section II-A, table I) and secondly, algorithmic versions of the previously manual process has to be constructed without loosing reliability. Based on current research, this section presents the potential solution which tries to solve the second problem, automation, by using genetic algorithms that optimize metrics within constraints related to the specific software and dependencies between them.

A. The Search Based Approach

Tamburelli and Margara holds the position that A/B testing suffers from several limitations. They argue that the process

TABLE I: Challenges and Important Aspects to Consider

Challenge	Description
Architecture	With a traditional monolithic architecture it is very complex to independently deploy features and services which can obstruct the workflow. An older architecture might not even support experiments [8].
User base	If the user base is too small, results from experiments become unreliable and the running time required blocks new tests [6]
Value addition	It is not always clear how much business value experiments add which can make it harder to motivate the decision [6].
Good metrics	The way user engagement and interaction is measured has a very large impact on the amount of data that is required [4].
Speed	Speed and responsiveness are major factors for user experience. A testing infrastructure may not add any significant delays when using the software [6].
Scheduling experiments	How experiments are scheduled has a large impact on time to completion and reliability of results [7].
Service Quality	It is important that tests do not create unusable interfaces or a bad user experience [10].

of A/B testing, i.e to conceive, run and analyze the results of the tests is not a trivial task to do. It is a error prone and costly manual activity [9]. They formulate automated A/B testing as an search-based software engineering problem, targeting the problem with the usage of genetic algorithms. By formulating testing as an optimization problem, it enables the use of automated search algorithms for finding optimal solutions. They built a initial prototype tool that lets the developers specify the features that are relevant for testing with Java annotations (as shown in listing 1). The framework can then handle the task of generating, executing, and evaluating variants with the help of a genetic algorithm. Initial results from experiments indicates that the genetic algorithm was capable converging towards the optimal solution in relatively few steps for a somewhat homogeneous user group. However, making sure that the algorithm converges towards the best solution for user bases with clearly defined sub-groups with different preferences, is still a problem that has to be further explored [9].

Listing 1: Java annotation for feature testing

```
@IntegerFeature(range="12:1:18")
int textSize;
@StringFeature(
  values={"CheckOut","Buy","BuyNow!"}
)
String buttonText;
```

Another way to automate the process of running experiments is proposed by Schermann and Leitner in the paper *"Search-Based Scheduling of Experiments in Continuous Deployment"* [7]. They have a different focus from the research conducted by Tamburelli, and works on the scheduling and execution of continuous experiments while taking various experiment dependencies and constraints into account. They investigate the possibility of using search-based scheduling to make it easier to execute experiments in parallel and at the same time avoid overlapping experiments which can cause reliability problems as previously discussed. They formulate the scheduling as an optimization problem and investigate different search-based approaches such as genetic algorithms and local search to solve it. The fitness of the solution for such a task depends on what aspects are considered most important, either how well optimized the solution is for producing reliable results, or how quick all scheduled experiments can be executed. Schermann and Leitner opted to use a weighted fitness score based on total experiment time, delay before start and how well experiments converged towards group preferences. In their experiments they found that a newly proposed search based genetic algorithm outperformed multiple alternative constraint solvers, giving a weighted fitness score up to 19% higher. The authors envisions that this type of scheduling could be a part of a release or deployment pipeline in the future when some more progress within the research area has been made [7].

B. A/B Pipelines in Industry

In the current state of research within academia, there is no complete solution to handling the automation of scheduling and experiment design. Instead, most software heavy companies are building their own pipeline which integrates launch of experiments, collection of user data and to some extent data processing [1–3, 10, 11]. To a large degree, experiments has to be classified and scheduled appropriately by a developer to

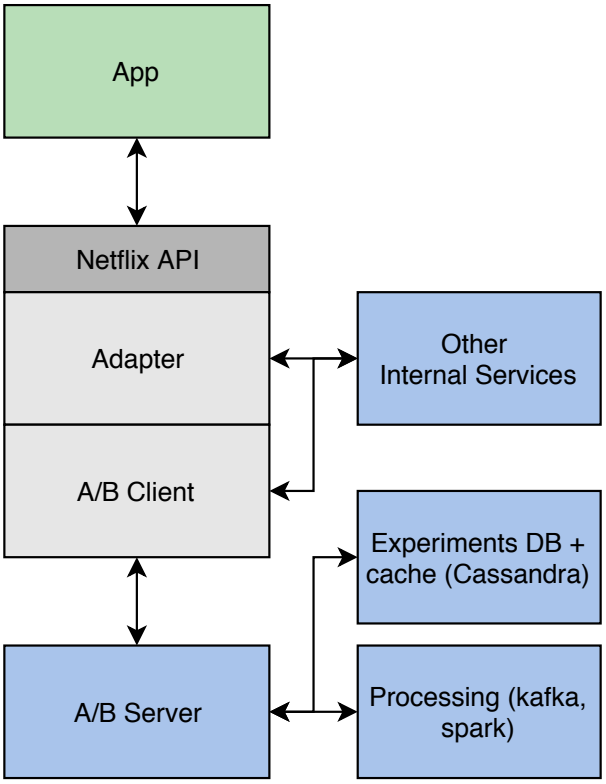


Fig. 2: Overview of the test architecture used by Netflix [11].

maintain usability and experiment independence which could not be performed by an automated tool today [9, 10].

To demonstrate this with an example, Netflix internally uses a microservice architecture which reflects in the design of their infrastructure for testing as shown in figure 2. In their case, requests are handled by their API, which in turn, connects to a test service that handles both scheduling, code and monitoring of user behaviours. When the tests and code for the current client has been established, subsequent requests to other internal services can provide the extra data required for any test [11]. This type of pipeline would be applicable for a completely automated system as well as that system would primarily create the content in the experiments database and analyze results.

IV. Conclusion

Running controlled experiments can provide companies with useful insights and ultimately higher profits, which has created a large interest into adopting testing pipelines into production among many tech-companies. However, there are still some challenges that has to be solved before a completely automated solution can be created.

With current technology, it is possible to create a very successful pipeline for testing code before deployment. We have seen examples and reports of this from several large companies. On the other hand, the challenges of creating good tests as well as automated scheduling without creating dependencies are still largely unsolved even though genetic algorithms are starting to show promising results.

Most notably, it is complex to automatically construct test of non-primitives (as opposed to primitives such as strings and integers) as developers are required to build software that can automatically switch between multiple complex constructs with possible dependencies. This argument is valid for scheduling of experiments as well, as dependencies between experiments has to be detected to prevent results from interfering with each other.

At the same time as tests are running, the quality of service also has to be maintained which can be complex to achieve without strict and manually constructed constraints. It is not acceptable to degrade the user experience significantly that would be required for a optimization algorithm to explore all paths the service interface could be altered in. This introduce a new source of errors and in some cases, it might be complex to create well defined boundaries.

Overall, this field is a very active research area and new technologies are actively being deployed. It is a high chance that more automated solutions will become available to users within a not so distant future. Such a change would have a huge impact on specifically the smaller actors on the market, that not always have access to all skills and manpower needed to use the power of continuous experimentation in their daily business today.

References

[1] Fabijan, Aleksander et al. “The evolution of continuous experimentation in software product development: from data to a data-driven organization at scale”. In: *Proceedings of the 39th International Conference on Software Engineering*. IEEE Press. 2017, pp. 770–780.

[2] Govind, Nirmal. *A/B Testing and Beyond: Improving the Netflix Streaming Experience with Experimentation and Data*. Jan. 2017. URL: https://medium.com/netflix-techblog/a-b-testing-and-beyond-improving-the-netflix-streaming-experience-with-experimentation-and-data-5b0ae9295bdf (visited on 04/26/2019).

[3] Kevic, Katja et al. “Characterizing experimentation in continuous deployment: A case study on bing”. In: *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track, ICSE-SEIP 2017* (2017), pp. 123–132. DOI: 10.1109/ICSE-SEIP.2017.19.

[4] Kharitonov, Eugene, Drutsa, Alexey, and Serdyukov, Pavel. “Learning Sensitive Combinations of A/B Test Metrics”. In: *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. 2017, pp. 651–659. ISBN: 9781450346757. DOI: 10.1145/3018661.3018708.

[5] Kohavi, Ron et al. “Online controlled experiments at large scale”. In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 1168–1176.

[6] Kohavi, Ron et al. “Seven rules of thumb for web site experimenters”. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014), pp. 1857–1866. DOI: 10.1145/2623330.2623341.

[7] Schermann, Gerald and Leitner, Philipp. “Search-based scheduling of experiments in continuous deployment”. In: *Proceedings - 2018 IEEE International Conference on Software Maintenance and Evolution, ICSME 2018* (2018), pp. 485–495. DOI: 10.1109/ICSME.2018.00059.

[8] Schermann, Gerald et al. “We’re doing it live: A multi-method empirical study on continuous experimentation”. In: *Information and Software Technology* 99.March 2017 (2018), pp. 41–57. ISSN: 09505849. DOI: 10.1016/j.infsof.2018.02.010.

[9] Tamburrelli, Giordano and Margara, Alessandro. *Towards Automated A/B Testing*. Tech. rep. Faculty of Informatics. University of Lugano, Switzerland, 2014, pp. 184–198.

[10] Tang, Diane et al. “Overlapping experiment infrastructure”. In: *ACM* (2010), p. 17. DOI: 10.1145/1835804.1835810.

[11] Urban, Steve, Sreenivasan, Rangarajan, and Kannan, Vineet. *It’s All A/Bout Testing: The Netflix Experimentation Platform*. Apr. 2016. URL: https://medium.com/netflix-techblog/its-all-a-bout-testing-the-netflix-experimentation-platform-4e1ca458c15 (visited on 04/26/2019).