Towards Inclusive Software Engineering Through A/B Testing: A Case-Study at Windows

Irina Niculescu, Huibin Mary Hu, Christina Gee, Chewy Chong, Shivam Dubey, and Paul Luo Li
Microsoft
Redmond, WA
(irnicule, maryhu, christina.gee, chewy.chong, shivam.dubey, paul.li)@microsoft.com

***Abstract*—Engineering software to be inclusive of all those that might/could/should use the software is important. However, today, data used to engineer software can have inherent biases (e.g. gender identity) with inclusiveness concerns. While much attention has been given to this topic in the AI/ML space, in this paper, we examine another data-centric software engineering process, A/B testing, for which we have a dearth of understanding today. Using real-world data from the Windows out of box experience (OOBE) feature, we provide a case-study of how inclusiveness concerns can manifest in A/B testing, practical adjustments to A/B testing towards inclusive software engineering, and insights into ongoing challenges. We discuss implications for research and practice.**

***Index Terms*—Software engineering, data privacy, data analysis, data collection, software quality, software development management**

I. Introduction

Inclusiveness is an important topic in software engineering and data sciences today, with a key challenge being biases inherent in the data collected and used to engineer software [1]. These biases risk software being non-inclusive of all potential users, perpetuating societal inequalities of today into the future. For example, Google's speech recognition has been found to be more accurate for males than females [2]; Microsoft, IBM, and Face++ facial recognition systems have all been found to classify males more accurately than females [3]

Concurrently, A/B testing–a data-centric software engineering technique–is being used by many organizations to design and improve their software, e.g. Google, Facebook, Amazon, and Microsoft [4] [5] [6]. A/B testing is the gold standard for evaluating the causal impact of changes, empowering organizations to make data-driven decisions about their software. While much attention has been given to data-related inclusiveness concerns in the ML/AI space (e.g. [7] [8] [9]), relative little has been examined and reported about inclusiveness concerns for A/B testing.

In this paper, we address the gap in real-world knowledge about inclusive software engineering and A/B testing by contributing:

- A real-world case-study of how inclusiveness concerns can manifest in A/B testing
- Practical adjustments to A/B testing towards inclusive software engineering
- Insights into ongoing real-world challenges

We examine the Windows out of box experience (OOBE) functionality from the Windows 10 operating system, using actual designs and results from A/B tests to show how inclusiveness concerns can manifest. We then discuss adjustments to Windows experimentation to improve inclusiveness: analysis dashboards, statistical computations, and processes/methodologies. Finally, we discuss insights into ongoing challenges in this space. This knowledge may help practitioners improve their A/B testing methodology and may inspire researchers to develop improvements towards even more inclusive software engineering.

In the rest of this paper, we first discuss background and related work in Section II. Then, we explain our methodology in Section III. Next, in Section IV, we describe the Windows OOBE functionality and inclusiveness concerns in A/B tests. This is followed by adjustments to Windows experimentation in Section V. We discuss insights into ongoing challenges in Section VI. Finally, we conclude in Section VII.

II. Background and Related Work

We discuss background and related work in two areas: inclusiveness in software engineering and A/B testing.

A. Inclusiveness in Software Engineering

As data becomes increasingly integral to our society (e.g. in finance, in criminal justice, in employment, in admissions), so does the specter of inclusiveness, fairness, bias, and other equality issues arising, due to data perpetuating inequalities of the past and present [10]. This issue is especially concerning for software, since data is essential to many software products and software engineering processes.

Computer science research in numerous domains has examined how to detect, measure, and mitigate inclusiveness/fairness/bias concerns. These include NLP processing (e.g. gender biases in Google's Word2Vec embedding [11] and YouTube automated transcriptions [2]) and computer vision (e.g. gender and racial accuracy differences in Microsoft, IBM, and Face++ image processing software [3]). More generally, there is extensive work on the inclusiveness, fairness, and bias issues in ML/AI [7], [8], [9], including definitions of different kinds of fairness and approaches to adjust machine learning algorithms to achieve various definitions of fairness. A comprehensive review by Corbett-Davis and Goel is in [1]. However, relative little is known today about equality concerns

in the data-centric software engineering process of A/B testing, where (as we discuss below) organizations use test results to make feature decisions that may have inclusiveness/fairness implications.

B. A/B Testing

In its simplest form, an A/B test randomly assigns users one of two variants: control (A) or treatment (B). Usually control is the existing system and treatment is the software with a feature added/changed, say, feature X. User interactions with the two software variants are recorded and from that, metrics are computed and compared (e.g. offerings/services sign-up rates). If the A/B test was designed and executed correctly, the only difference between the two variants is feature X. External factors such as seasonality (i.e. time related effects like weekday/weekend or summer/winter differences), impact of other feature changes, moves by competition, etc. are distributed evenly between control and treatment, and therefore do not impact the results of the experiment. Hence, differences in metrics between the two groups (in aggregate) can be attributed to either feature X or noise (i.e. inherent/natural variation of any metric within a population). The noise option is ruled out using statistical tests (e.g. the t-test). The upshot is that a causal relationship between the change to the product and changes in user behavior is established through the A/B test. Experimentation has been shown to be especially useful when user reactions are uncertain for design changes or novel innovations [12], providing organizations an effective approach of making data-driven decisions to improve their software.

Fueled by the growing importance of A/B tests in the software industry, A/B testing is an active research area. Research has been focused on topics such as new statistical methods to improve metric sensitivity [13], [14], metric design and interpretation [15], [16], projections of results from a short-term A/B test to the long term [17], [18], benefits of experimentation at scale [12], [19], experiments in social networks [20], high-level architecture of A/B testing platforms [21], [22], as well as examples, pitfalls, rules of thumb and lessons learned from running controlled experiments in practical settings [23], [24].

Fairness has received attention, with recent work looking at novel metrics for assessing fairness in results of A/B tests [25] (pre-print) as well as approach to conduct A/B tests with differential private data [14]. Our work complements prior work by providing practical *methodology* adjustments that should work with (and enhance) any metric and any statistical testing approach, advancing inclusive software engineering practices.

III. Methodology and context

Our case study of inclusive software engineering through A/B testing covers work in the Windows organization at Microsoft Corporation in the USA between 2018 and 2019. The authors of this paper are or have been employed at Microsoft and are engineers, program managers, and data scientists who worked on the Windows out of box (OOBE) feature (the feature under examination) as well as the A/B testing platform. We briefly describe our empirical approach as prescribed by Runeson and Höst [26].

A. Data Collection

We leveraged two primary sources of data. First, we used materials from actual A/B tests, including analysis reports produced by engineering teams, which the authors had access to. These contained findings, conclusions, and subsequent engineering decisions. Second, we used documentation (e.g. notes, emails, presentations, and design docs), extracted from document depots (e.g. Sharepoint).

B. Data Analysis

Overall, we present factual findings from A/B testing of the feature and subsequent engineering adjustments to the A/B testing platform. We present actual data from experiments as well as implementation decisions and their implications. Our analyses and insights are based on the authors' knowledge and experience.

C. Threats to Validity

To reduce the risk to construct validity, we name features, components, and findings (where possible). We use industry standard terminology and provide explanations/examples to clarify ambiguous concepts and terms. To reduce threats to internal validity, this paper has been reviewed internally to ensure accuracy. Finally, though this paper is a case-study with external validity limitations, many of the topics and concerns are not tied directly to Windows and are applicable to other contexts and organizations.

IV. The Windows Out of Box Experience

We first describe the Windows out of box experience (OOBE) feature and the changes under consideration. Then, we discuss the initial A/B tests and the inclusiveness concerns discovered.

A. Blue and Light OOBE Designs

OOBE helps users that are new to their Windows devices to setup their Windows accounts and services. Historically, OOBE has utilized a Blue design, see Figure 1. In 2018, the feature team considered an alternative Light design, featuring additional graphics and a lighter color scheme, see Figure 2. Figure 3 is a close up comparison of the two designs for the initial first screen. The goal of the OOBE changes was to increase Windows services linking/sign-up, as measured by the proportion of users who link/sign-up for the Windows services.

B. Initial A/B Test Results

During the development process, the engineering team used an initial small-scale A/B test to evaluate the two designs using web-based mock-ups. The tests were statistically powered to detect roughly a 15% change in the link/sign-up rates, with each variant having roughly 150 paid participants. The participants were recruited with the criteria of having recently

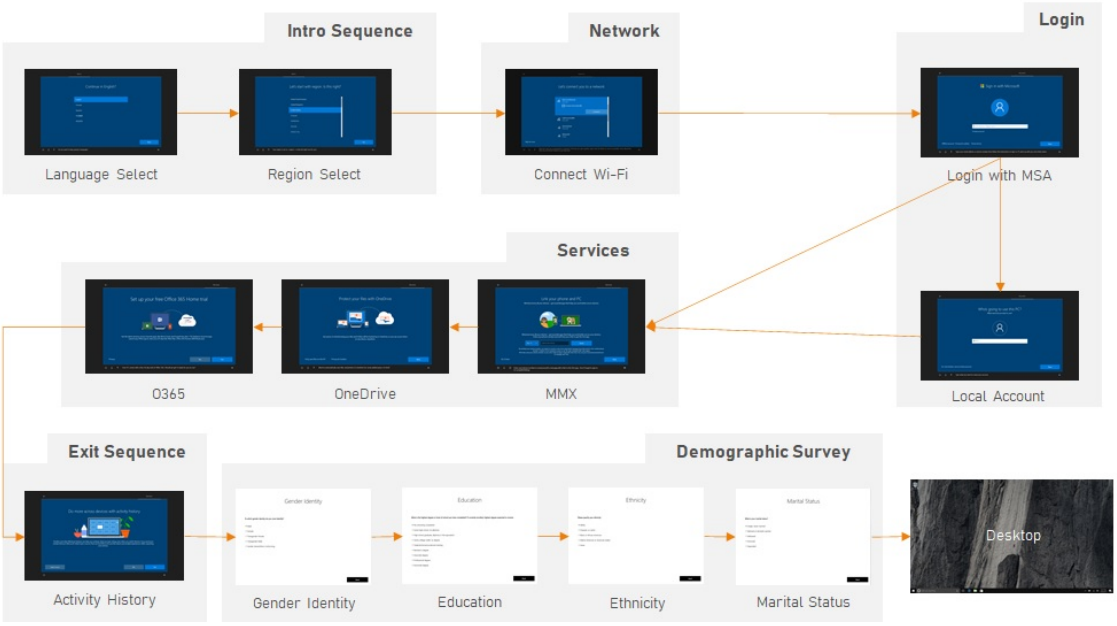

Fig. 1. The Blue OOBE design

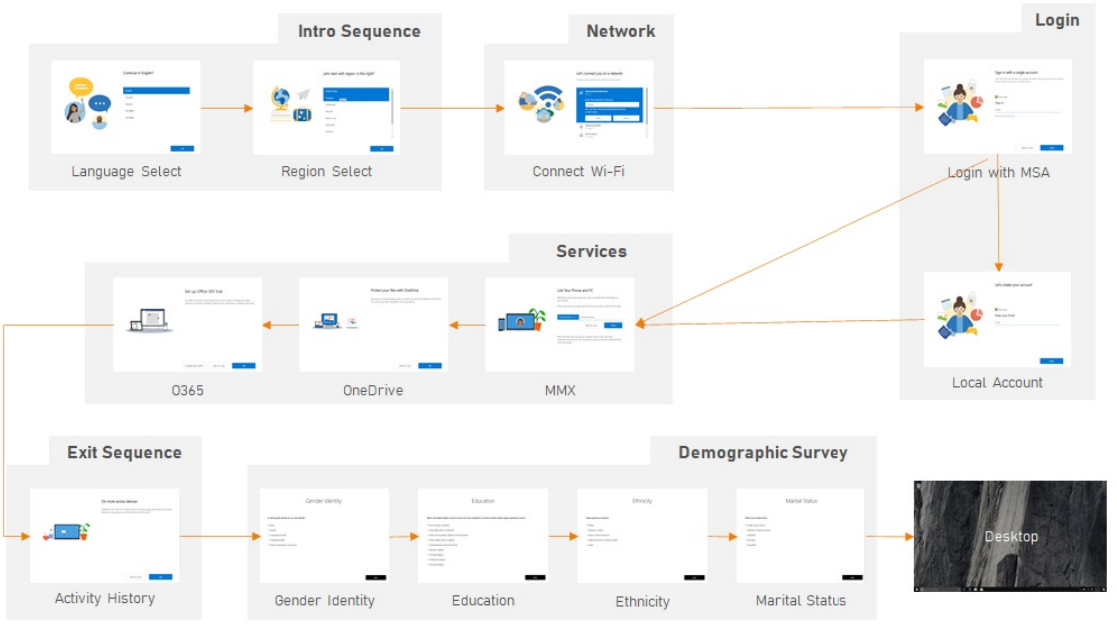

Fig. 2. The Light OOBE design

purchased a Windows PC. The A/B test consisted of going through the OOBE screens, followed by a common set of demographic questions (which was specific to the A/B test and not part of the OOBE feature), as shown in Figure 1 and Figure 2.

Initial analysis of the results, a summary of which is shown in Figure 4, indicated no statistically significant differences between the two designs. However, a closer examination of data along demographic slices indicated large differences along gender identity. The two biggest differences where with the OneDrive screen, shown in Figure 5, and the Office365 screen, shown in Figure 6 (exact percentages obfuscated). For OneDrive, those self-identified as Female had 18% *higher* opt-in for the Light theme, while those self-identified as Male had 21% *lower* opt-in. Similarly, for Office365, those self-identified as Female had 39% *higher* opt-in for the Light theme, with those self-identified as Male had 39% *lower* opt-in. However, since the A/B tests were not specifically powered to detect differences in the gender identity slice, the results were borderline statistically significant.

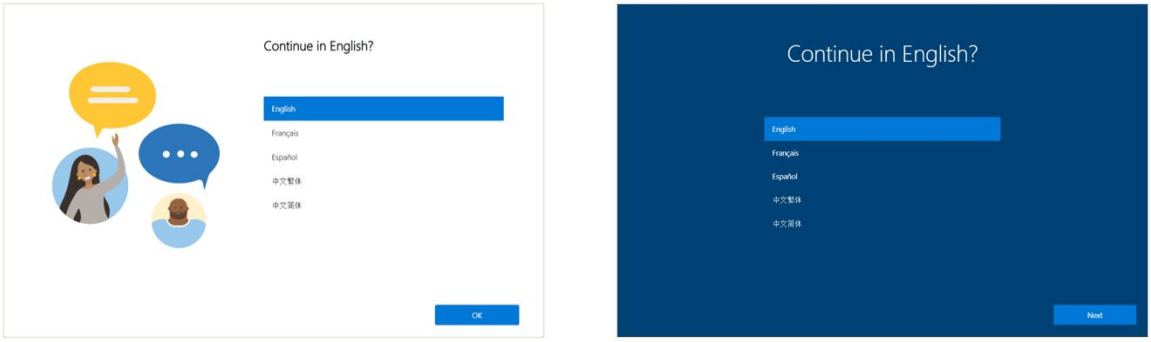

Fig. 3. Comparison of the Blue and White OOBE designs

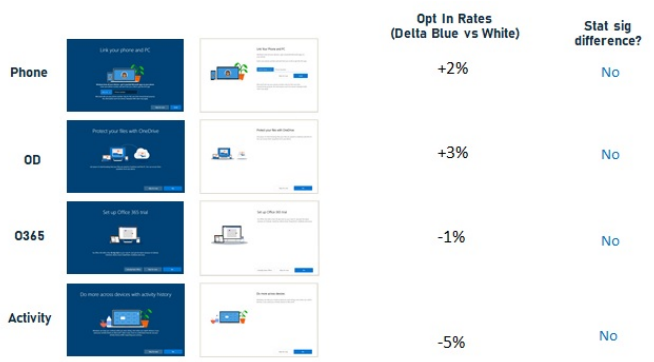

Fig. 4. Summary of initial A/B test

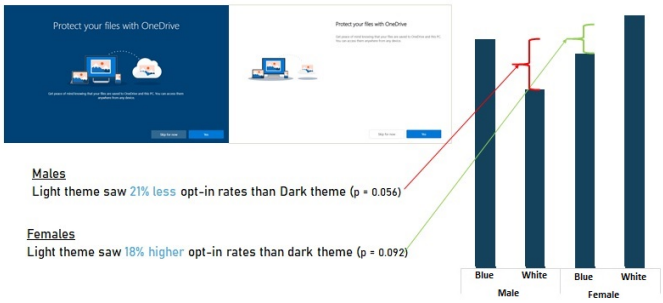

Fig. 5. Detailed results by gender identity for OneDrive

C. Inclusive Software Engineering Concerns

The A/B tests revealed several concerns about the process. First, this A/B test unintentionally recruited a balanced 50/50 split in gender identity, which enable the insights related to gender identity; however, in general, A/B tests are conducted with *random* sampling, which are unlikely to be balanced. For example, given the inherent under-representation of the gender identity of female in computing data sets [2], rather than no statistically significant difference, the overall results would likely have been the Blue design being better.

Second, the A/B test was not sufficiently powered to detect the changes in the sub-populations. Even though the differences all exceeded the initial 15% differences threshold, due to sub-populations being smaller, the results were only marginally statistically significant. In addition, other than the unintentional balance in gender identity, the demographics

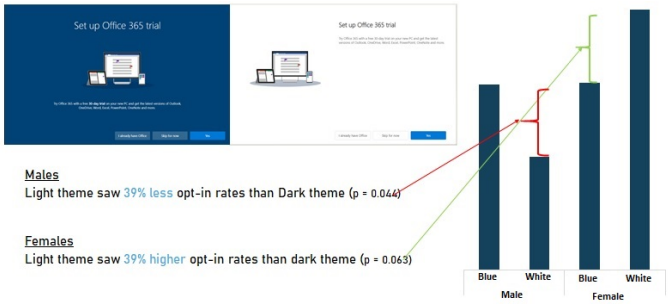

Fig. 6. Detailed results by gender identity for Office

were heavily imbalanced along other dimensions like education and ethnicity, unable to yield insights into those dimensions.

Finally, overall, the results indicated that explicit considerations and adjustments for inclusiveness are needed *a-priori*. Otherwise, inclusiveness concerns caused by inherent biases in real-world data might go undetected.

V. Adjustments to Windows Experimentation

In this section, we discuss initial steps that Windows has taken towards inclusiveness software engineering through adjustments to its A/B testing methodology. We first give background on the Windows Experimentation platform (WExp) and then describe adjustments to improve its ability to detect, quantify, and account for inclusiveness concerns.

While feature decisions (e.g. mitigation, modifications, ship/no-ship, etc.) can have many additional considerations (e.g. security, economics, costs) the fairness adjustments provides additional knowledge that enable organizations to make more informed decisions.

A. The Windows Experimentation Platform

The Windows Experimentation Platform (WExp) is an A/B testing platform built for Windows client-side *code* experiments and roll-outs. Other commercially available solutions and experimentation platforms in the literature are generally server-side systems (e.g. web-pages), where either control or treatment behavior is delivered to the user at the time when the experimental scenario is encountered (e.g. navigation to a web-page), and complete data are captured server-side. In contrast, for WExp, due to the need to quickly terminate A/B

tests and return devices to known safe conditions, the code for both control and treatments are delivered to devices, then a centralized cloud service manages the A/B assignments. This enables fast termination of A/B tests and prevents new/other devices from accidental exposure (i.e. devices are kept in the known safe condition, unless explicitly toggled via the cloud service). Due to possible offline scenarios, restarts, and telemetry delays, inbox components of WExp record and cache information about the assignments, exposures (i.e. actual executions of the code paths), and data about user/system behaviors. Finally, since information from other interacting services (e.g. Bing) may be important for evaluating the success of experiments, the system performs fuzzy joins from various data sources to produce the final analysis.

WExp leverages several existing Microsoft components/services, uses others in novel ways, and has new components. For example, delivery of code to devices leverages the Windows Update service [27]. Data is recorded and transmitted using Windows Telemetry services and adheres to user settings and restrictions [28]. Randomization and statistical comparison of data are performed using the ExP system [21]. For a more thorough technical description of the system, please refer to [22]. Finally, WExp has a set of policies and processes that help to ensure safety and best-practices. For example, all A/B tests intended for General users must first be run in internal and/or Windows Insider populations (a self-selected group of users who run pre-release versions of Windows to provide feedback [29]) to ensure safety and quality.

B. Adjustments for inclusiveness

Three adjustments were made to WExp towards inclusive software engineering: modifications to the pre-launch sample size computations, improvements to the analysis dashboards, and changes to guidance/education to consider possible inclusiveness concerns.

1) Pre-launch sample size computations: Best-practices today call for pre-release sample size calculations (within the time-frame of the A/B test) to determine the minimum samples needed to detect the predetermined amounts of change in key metrics of interest. The A/B test is then conducted on this minimal sample rather than the entire population. These best-practices help to minimize the possible down-sides of the A/B test. For Windows, this also helps to reduce costs associated with executing the A/B tests both for users (e.g. data collection and transmission costs) and for Windows (e.g. data processing and storage costs).

Based on the insights, the modification was to perform sample-size calculations for each metric for each sub-populations of interest and then take the maximum among the sample sizes required. There are two important reasons for this adjustment. The first, as discussed above, there are inherent sub-population imbalances which may lead to insufficient samples in the sub-population of interest to detect different (possibly negative) effects. For example, those identified as Female in the 2019 United States Census were 51.0% [30]; the proportion is *significantly* different (both statistically and in percentages) in the Windows Insider population. Therefore, to obtain the desired minimum sample size to detect movement of a metric for a given sub-population (e.g. the gender identity of female), a larger random sample would be needed compared to the sample needed to detect movement of the metric for the overall population. The second related reason is that metric characteristics (e.g. mean and standard deviation) can be different for each sub-population, which would also alter sample sizes needed to detect a desired amounts of metric movement.

As an example, adjustments to the standard sample-size computation for a two-sided equality of proportions (i.e. p-test) [31] is shown in Equations 1 and 2. Computations for Type I error rate α and power $1-\beta$ would be done for each of the k sub-populations of interest, with the proportions for control, p_{A_k}, for treatment (projected) p_{B_k} specific to each sub-population (since these can differ for each sub-population), this is Equation 1. The sample sizes n_k are then adjusted by the size of sub-population n_k relative to the overall population size, $prop_k$. The max is then taken over the sub-populations to obtain the desired sample size for the metric, this is in Equation 2. This is repeated for each metric of interest, with similar adjustments for other types of statistical tests (e.g. t-test). The max over all the metrics is the final sample size.

$$n_k = (p_{A_k}(1-p_{A_k}) + p_{B_k}(1-p_{B_k}))(\frac{z_{1-\alpha/2} + z_{1-\beta}}{p_{A_k} - p_{B_k}})^2 \quad (1)$$

$$n = \max_k(\frac{n_k}{prop_k}) \quad (2)$$

2) Analyses and dashboards: WExp analyses and dashboards were also modified to better enable inclusive software engineering. The data are labeled with sub-population information, and then analyses/dashboards can automatically surface notifications when statistically significant differences between sub-populations occur. Furthermore, a drill-down ability is provided to examine differences between sub-populations holistically across all metrics.

A hypothetical example for the gender identity dimension and the "Distinct Search Queries" metric is shown in Figure 7. In the example, the overall metric movement is not statistically significant with p-value=.70. This is due to a larger proportion of male in the random sample, for which there is no statistically significant difference, hiding the statistically significant difference in the smaller female sub-population. The dashboard visually indicates that there is a 'Segment of Interest' for the 'Gender_Seg' in the data; clicking on the line reveals the differences in the metric for the gender sub-populations. The information provided includes both the amount of difference in the metric and the sizes of the sub-populations. Many dimensions can be analyzed concurrently, for example, gender identity, regions of the world, device form factor, etc.

Another important analysis/dashboard is *all* metrics for given sub-populations. This is needed because, while examin-

Fig. 7. Sub-population movement detection UX

ing differences in a single metric is valuable, viewing all the metrics is important for triangulation. Best practice today calls for measuring behaviors using multiple metrics and multiple statistics to better holistically understand behaviors. This can include measuring multiple steps along a causal chain (e.g. a funnel). This can also include using multiple metrics (e.g. means, proportions greater than, and percentiles) to be more resilient to outliers. Thus, the analyses/dashboard also look at all the metrics sliced by sub-populations; a hypothetical example is in Figure 8. In the example, holistically examining looking link/sign-up rates across all screens provides better understanding of the differences between the designs. In this manner, all facets of impact to a sub-population can be assessed holistically.

3) Education and guidance: Finally, technical improvements and better statistics need to be complemented by cultural change in the engineers to utilize the information and to adjust their thinking/decisions. Therefore, Windows has also added several education and training documentation to help engineering teams be more aware of possible inclusiveness concerns. These entail considering sub-populations that may be relevant for the product change *a-priori*, then adjusting the A/B tests (as described above) to ensure that their impact can be assessed, and finally guidance to consider potential impacts on the sub-populations when making final product decisions.

Though not all features have inclusiveness issues, a shift in mindset to *consider* potential impacts for *all* features is important. For example, even a non-user-facing change in networking protocols can behave differently in countries with less-advanced networking infrastructure and adversely impact those sub-populations [22].

VI. Insights into ongoing challenges

While we have made a positive start, more work is needed. We provide insights into some of the on-going challenges that we hope to tackle, possibly through collaborations with researchers and other practitioners.

A. Targeted sampling

While it may be possible to conduct target sampling for specific sub-populations (e.g. to create representative sample), in practice we have found it to be impractical for two reasons. First, many dimensions are inter-related (e.g. higher proportions of the laptop form factor in the United States region); therefore, target sampling one dimension can lead to artificial imbalances in other dimensions. This creates problems in evaluating results, especially in assessing impact of changes for other sub-populations. Second, more importantly, some dimensions are not known at the beginning of the experiment. For example, the OOBE feature is often the *first* feature encountered by users, many dimension are not know at the time when users experience the feature (but would be known later). Therefore, to sample sufficient users of specific sub-populations into the experiment for a feature, a sufficiently large *random* sample is needed. Nonetheless, we realize that this approach has inefficiencies that can be improved.

B. Impacts of anonymity

One of the commonly discussed approaches to help protect vulnerable sub-populations is anonymity; by not providing information (e.g. demographics) organizations cannot use that information to bias their product and services [10]. However, we have found that anonymity also makes it difficult for responsible software organizations to adjust innovations to better suit those sub-populations. For example, in Section V-B2, without knowing the gender identity, the impact on female population would not be detectable in the A/B tests due to natural imbalanced in gender identity in the Windows Insiders population. In fact, all of our inclusive adjustments require knowing sub-population membership. While it is possible to create features that promote inclusiveness without A/B testing, it would then be difficult to understand whether the features had the intended impact and/or whether unforeseen negative impacts occurred. This might be considered as an example of a trade-off between fairness and privacy. Consequently, approaches to safely share information about sub-population

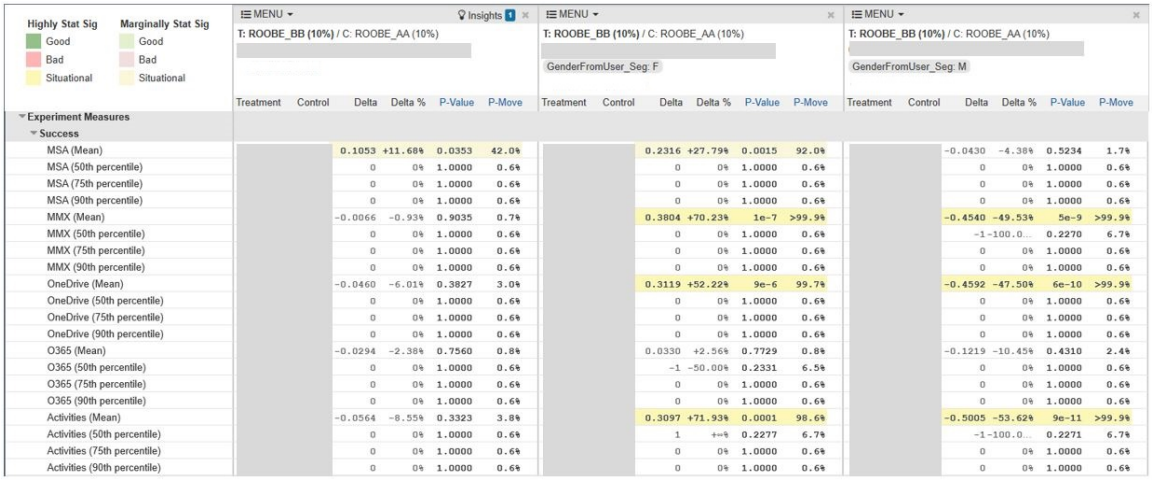

Fig. 8. Slicing by sub-population UX

membership, to promote trust in the information sharing, and/or to raise awareness about the potential benefits may be interesting areas of future research.

C. Unknown unknowns

We have open questions about whether observable data from interesting sub-populations have external validity concerns: whether those in the data are representative or biased. After all, with the goal of being inclusive, it is those that *cannot* be observed that may need the most help to be included. Therefore, whether the insights/conclusions for those that can be observed can be generalized to others of the sub-population (i.e. the unknown unknowns) is often a difficult question. Our general approach has been to make do with the data available, since making a decision with some data is better than no data. However, we recognize that this can (and should) be improved.

VII. Conclusion

Software should be inclusively engineered, taking into consideration all those that might/could/should use the software. However, data can have inherent biases (e.g. gender identity) that cause inclusiveness concerns, including for A/B tests, which are widely-used today by many organization to engineer their software. We have provided examples of how inclusiveness concerns can manifest as well as adjustments that practitioners can make towards inclusive software engineering. We have also provided insights into ongoing challenges that we face, which we hope can be overcome, possibly through collaborations/contributions of researchers. Together we can enable the engineering of better and more inclusive products for our ever increasingly software dependent society.

Acknowledgment

We thank all those that worked on the OOBE feature and on WExp for their contributions, especially Aaron Grady, whose sponsorship and support was invaluable.

References

[1] S. Corbett-Davies and S. Goel, "The Measure and Mismeasure of Fairness: A Critical Review of Fair Machine Learning," *arXiv preprint arXiv:1808.00023*, 2018. [Online]. Available: http://arxiv.org/abs/1808.00023

[2] R. Tatman, "Gender and Dialect Bias in YouTube's Automatic Captions," in *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, 2017, pp. 53–59.

[3] J. Buolamwini and T. Gebru, "Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification," in *Conference on Fairness, Accountability and Transparency*, 2018, pp. 77–91.

[4] D. Tang, A. Agarwal, D. O. Brien, and M. Meyer, "Overlapping Experiment Infrastructure : More , Better , Faster Experimentation," in *Proc SIGKDD '10*, 2010, pp. 17–26.

[5] G. Panger, "Reassessing the Facebook Experiment : Critical Thinking About the Validity of Big Data Research," *Information, Communication & Society*, vol. 19, no. 8, pp. 1108–1126, 2016.

[6] R. Kohavi, B. Frasca, T. Crook, R. Henne, and R. Longbotham, "Online Experimentation at Microsoft," *Data Mining Case Studies*, vol. 11, no. 2009, 2009.

[7] M. Hardt, E. Price, and N. Srebro, "Equality of Opportunity in Supervised Learning," in *Proc NIPS '16*, 2016, pp. 3315–3323.

[8] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork, "Learning Fair Representations," in *Proc ICML '13*, 2013, pp. 325–333.

[9] N. Kilbertus, M. Rojas-Carulla, G. Parascandolo, M. Hardt, D. Janzing, and B. Schölkopf, "Avoiding Discrimination through Causal Reasoning," in *Proc NIPS '17*, 2017, pp. 656–666.

[10] C. O'Neil, *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Broadway Books, 2016.

[11] T. Bolukbasi, K.-W. Chang, J. Y. Zou, V. Saligrama, and A. T. Kalai, "Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings," in *Proc NIPS '16*, 2016, pp. 4349–4357.

[12] R. Kohavi and S. Thomke, "The Surprising Power of Online Experiments," *Harvard Business Review*, vol. 95, no. 5, p. 74, 2017.

[13] A. Deng, Y. Xu, R. Kohavi, and T. Walker, "Improving the Sensitivity of Online Controlled Experiments by Utilizing Pre-experiment Data," in *Proc WSDM '13*, 2013, p. 123.

[14] B. Ding, H. Nori, P. Li, and J. Allen, "Comparing Population Means under Local Differential Privacy with Significance and Power," in *Proc AAAI '18*, 2018.

[15] P. Dmitriev, S. Gupta, K. Dong Woo, and G. Vaz, "A Dirty Dozen: Twelve Common Metric Interpretation Pitfalls in Online Controlled Experiments," *Proc KDD '17*, pp. 1427–1436, 2017.

[16] P. Dmitriev, B. Frasca, S. Gupta, R. Kohavi, and G. Vaz, "Pitfalls of Long-term Online Controlled Experiments," in *Proc Big Data '16*, 2016, pp. 1367–1376.

[17] H. Hohnhold, D. O'Brien, and D. Tang, "Focusing on the Long-term: It's Good for Users and Business," in *Proc KDD '15*, 2015, pp. 1849–1858.

[18] P. Dmitriev and X. Wu, "Measuring Metrics," *Proc CIKM '16*, pp. 429–437, 2016.

[19] A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch, "The Benefits of Controlled Experimentation at Scale," in *Proc SEAA'17*, 2017, pp. 18–26.

[20] Y. Xu, N. Chen, A. Fernandez, O. Sinno, and A. Bhasin, "From Infrastructure to Culture: A/B Testing Challenges in Large Scale Social Networks," in *Proc KDD'15*, 2015, pp. 2227–2236.

[21] S. Gupta, L. Ulanova, S. Bhardwaj, P. Dmitriev, P. Raff, and A. Fabijan, "The Anatomy of a Large-Scale Online Experimentation Platform," in *Proc ICSA '18*, 2018.

[22] P. L. Li, P. Dmitriev, H. M. Hu, X. Chai, Z. Dimov, B. Paddock, Y. Li, A. Kirshenbaum, I. Niculescu, and T. Thoresen, "Experimentation in the Operating System: The Windows Experimentation Platform," in *Proc ICSE '19*, 2019.

[23] R. Kohavi, A. Deng, R. Longbotham, and Y. Xu, "Seven Rules of Thumb for Web Site Experimenters," in *Proc KDD '14*. New York: ACM Press, 2014, pp. 1857–1866.

[24] R. Kohavi and R. Longbotham, "Online Experiments: Lessons Learned," Tech. Rep. 9, 2007.

[25] G. Saint-Jacques, A. Sepehri, N. Li, and I. Perisic, "Fairness Through Experimentation: Inequality in A/B Testing as an Approach to Responsible Design," *arXiv preprint arXiv:2002.05819*, 2020. [Online]. Available: http://arxiv.org/abs/2002.05819

[26] P. Runeson and M. Höst, "Guidelines for Conducting and Reporting Case Study Research in Software Engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, 2009.

[27] D. Halfin and B. Lich, "Build Deployment Rings for Windows 10 Updates," 2017. [Online]. Available: https://docs.microsoft.com/en-us/windows/deployment/update/waas-deployment-rings-windows-10-updates

[28] E. Bott, "Windows 10 Telemetry Secrets: Where, When, and Why Microsoft Collects Your Data," 2016. [Online]. Available: https://www.zdnet.com/article/windows-10-telemetry-secrets/

[29] T. Myerson, "An Update on What's Coming Next for Windows Insiders," 2017. [Online]. Available: https://insider.windows.com/en-us/articles/update-whats-coming-next-windows-insiders/

[30] U. S. C. Bureau, "ge and Sex Composition in the United States: 2019," Tech. Rep., 2019. [Online]. Available: https://www.census.gov/data/tables/2019/demo/age-and-sex/2019-age-sex-composition.html

[31] S.-C. Chow, H. Wang, and J. Shao, *Sample Size Calculations in Clinical Research*. CRC Press, 2007.