Lean UX Research at Scale: A Case Study

Kelly Krout
Microsoft
Redmond (WA), USA
kkrout@microsoft.com

Juan Pablo Carrascal
Microsoft
Barcelona, Spain
jucarra@microsoft.com

Travis Lowdermilk
Microsoft
Redmond (WA), USA
travisl@microsoft.com

Figure 1: Developer Day events conducted in 2017, 2018, and 2019.

ABSTRACT

In recent years, as technology engineering production and release cycles have sped up, user experience (UX) research practices have correspondingly become more *Lean* to best match those cycles. On top of that, the increasing interest in incorporating user feedback into product development keeps pushing the limit of resources that UX research teams can allocate. Is this seemingly unstoppable appetite for user insights sustainable? How can UX researchers properly support their product teams' growing needs? We argue that, to be successful at scaling up Lean UX research, product teams need to become active actors in the process, while researchers need to use their expertise to provide guidance and training as required. As an example of this approach, we describe our experience organizing "Developer Day", a yearly event aimed at scaling up the Lean UX research practices of Microsoft's Developer Division. Developer Day has also extended the reach of the division's UX Research team for impacting decisions about the company's software developer tools and services.

CCS CONCEPTS

• **Human-centered computing** → **Usability testing**; **User studies**.

KEYWORDS

Lean UX, large scale, UX research, user research, usability

MuC'20, September 6–9, 2020, Magdeburg, Germany

ACM ISBN 978-1-4503-7540-5/20/09…$15.00
https://doi.org/10.1145/3404983.3405587

ACM Reference Format:
Kelly Krout, Juan Pablo Carrascal, and Travis Lowdermilk. 2020. Lean UX Research at Scale: A Case Study. In *Mensch und Computer 2020 (MuC'20), September 6–9, 2020, Magdeburg, Germany*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3404983.3405587

1 INTRODUCTION

We are in the Developer Division at Microsoft, a product engineering organization composed of numerous teams totaling nearly 2,000 employees. We make popular developer products including Visual Studio, Visual Studio Code, .NET, and Azure development tools. We have adopted a *Lean* [7] approach of continuous testing and iteration, and almost all product-related projects are driven by user learning. This approach is possible by a close partnership across disciplines, including Program Managers (PMs), Engineering, UX Design and UX Research. These roles are deeply embedded with each other, interacting daily, and sharing their unique expertise.

The UX Research team within the Developer Division has 15 employees, whose day to day duties include running user studies along with product teams and providing advice on research methodologies. With the ever growing interest in user insights and a PM to researcher ratio of 15:1, the team faces a big challenge when trying to meet the demand of all the product engineering happening across the organization.

On top of hands-on daily work, the UX Research team provides continuous coaching and training on research practices to designers, PMs and engineers. This includes formal training activities in the form of "bootcamps" for new employees, ad-hoc advice on specific research topics, as well as guidance for the organization and preparation for specific research events. This approach allows the UX research team to actively involve other roles in research, distributing the research load while leveraging multiple perspectives along the process.

Figure 2: Some of the methods used for the speed sessions: usability tests (left), interviews (center), participatory design (right).

1.1 Case Study: Developer Day

One event that has become a staple of large-scale collaboration for the Division is Developer Day, a yearly gathering of over 100 users and more than 250 product engineering employees. It features more than 600 Lean UX research sessions spanning interviews, usability tests, participatory designs, and concept value tests, all produced in one day, yielding a wealth of data in a short amount of time. Insights obtained during Developer Day help make decisions for the future of our products.

Developer Day was originally conceived and run by the UX Research team. After 3 years, it has evolved and grown, and it is now organized by a dedicated committee, currently led by PMs with guidance from the UX Research team. The event serves as a signature event that exemplifies our collaborative, Lean, and user learning culture. With this case study we want to share our experience planning, executing, and maximizing the impact of this event.

Developer Day Schedule

[Attendee name]

Activity	Team Member	Location	Time
Registration and Breakfast		Adams	9:00-10:00
Opening Talk		Rainier	10:00-10:30
Interview: Containers, Registry	Rohit Tatachar	Rainier 3	10:30-10:50
Interview: Release coordination, adopting new features, and agile planning	Alex Nichols	Rainier 3	10:55-11:15
Interview: Azure Monitoring, Metrics Explorer, and DevOps Practices	Henry Chen	Rainier 3	11:20-11:40
Interview: VS Code	Chris Dias	Rainier 3	11:45-12:05
Interview: .NET Core Web API & Microservices	Glenn Condron	Rainier 3	12:10-12:30
Interview: .NET developers exploring or trying to learn ML	Olia Gavrysh	Rainier 3	12:35-12:55
Flash Voting		Rainier	1:00-1:30
Lunch Social		Adams	1:30-2:30

Microsoft

Figure 3: Example of a schedule generated for a Developer Day attendee. The Location column refers to the room (there were several in this particular venue) and table number assigned to the attendee.

2 METHODOLOGY

2.1 Recruitment

Developer Day coincides with Microsoft Build[1], the company's premier annual developer conference, that is joined by an audience of more than 5,000 attendees (including software developers, architects, DevOps engineers, among other roles) from around the world. We send an invitation to all these attendees to participate in Developer Day. Conference attendees that express interest in participating are asked to complete a comprehensive survey with questions about their demographics, the technologies they use and the applications they build. Upwards of 1,000 Build attendees respond to the initial invitation to attend Developer Day each year. After a selection process we typically obtain a diverse sample of over 100 users representing about 30 countries, working in both large and small development teams from about 90 companies, and building apps for web, mobile, desktop, IoT, mixed reality, cloud services, and more.

[1]https://en.wikipedia.org/wiki/Build_(developer_conference)

2.2 Preparation

In the weeks before Developer Day, the UX Research team focuses on several preparation activities:

- Providing advice to product teams on what research methods to use and how to adapt them to the event's research format (as explained in next section). We consider a range of methodologies to choose from, including interviews, usability tests, participatory designs [6], concept value tests [4], and even eye tracking (see Figure 2). Methods are chosen on the basis of the topic that product teams want to investigate.
- Coaching product teams on how to execute research within the constrains of the speed session format (see below).
- Defining the user profiles to speak to, again, based on the topics to be investigated.
- Identifying matching users from the Build attendees that expressed interest in Developer Day and sending confirmations to those users.
- Generating schedules for all the participants of the event, both product teams and users (see Figure 3). The schedule includes the times of each speed session and breaks.

2.3 Research format: Speed Sessions

Every product team will have several "speed sessions" throughout the day. The speed session format consists of short (20 minutes) and focused UX research sessions involving a user and a product team. A product team consists of a PM and an Engineer, with the former typically running the session and the latter taking notes. UX Designers also participate, either taking the place of the Engineer or joining the PM and Engineer together.

Each user is assigned to a table, which will be visited by a product team to start a session. After the time of a session is up, the product team has a 5-minute break to move on to the next user table to which they are assigned. This results in each PM and Engineer pairing having the chance to speak to six users. While conducting six speed sessions can be demanding, it gives the product teams an opportunity to generate multiple touch points on a single topic. In each Developer Day event, we schedule over 600 speed sessions in which users meet with more than 250 product team members.

2.4 Post-Event Activities

After Developer Day is over, the product teams meet—in collaboration with the UX Research team—to debrief and start data analysis. Analysis is guided by the *hypotheses* (as explained later in the *Lessons Learned* section) that teams test as part of their investigation. An event like Developer Day, built upon face-to-face interactions with users [1], focuses on the collection of mostly qualitative data. Thus, frequently used analysis methods include grounded theory-based affinity analysis [9] by means of Post-it notes, whiteboarding [3], and concept-driven coding [2]. Team work is particularly important to guarantee objectivity of interpretation. Whenever quantitative data are also collected during the speed sessions (e.g. task completion times and success metrics from usability tests [11]), they are usually analyzed with relevant numeric methods. Given the small sample sizes—up to six data points from speed sessions—most of the time we use descriptive statistics. Although the information from these six data points can be rich, other data sources are often used to increase the validity of the results. These additional sources can be similar studies conducted outside Developer Day—helping to increase sample size—or quantitative data from telemetry, surveys, etc.—for the purpose of cross-validation. Data from these external sources are also frequently used to form hypotheses to test during the event. Further additional methods might be applied when the hypotheses being tested require them.

An important aspect of the Lean UX research practice across the Developer Division is the continuous sharing of results. We do not necessarily wait until all the data are collected, analyzed and aggregated in a single, big report. Instead, we share results and observations on a continuous fashion as relevant insights are discovered. We have observed that this promotes higher awareness of the research work across teams and a more agile learning process. By using storytelling techniques, we also aim for increasing empathy towards users. We elaborate on our approach to sharing results in the *Make Data Representative and Relatable* section.

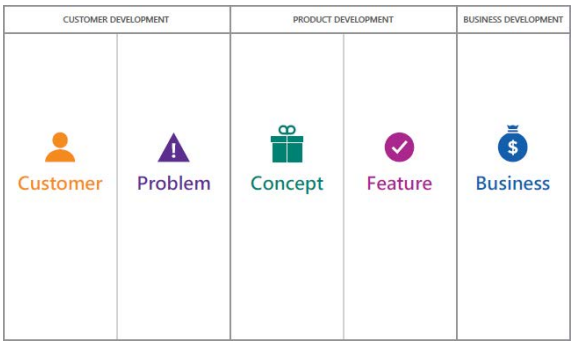

Figure 4: The Hypothesis Progression Framework (HPF) [8].

3 LESSONS LEARNED

We have learned a lot about scaling up Lean UX research after organizing Developer Day for 3 years in a row[2]. To ensure that similar events are engaging and valuable for both users and employees, it is necessary to invest time and effort to foster a user-focused culture within the organization. To that end, we have developed a set of strategies that we present next, illustrated with examples.

3.1 Use a Common Language

We needed a common language to allow us to communicate learnings at all levels of the organization. Additionally, we needed to minimize biases and remain objective, a difficult goal due to the close involvement with our products. To address these concerns, we introduced The Hypothesis Progression Framework (HPF) [8], a tool that helps product teams write useful hypotheses to organize their learning throughout a product's life cycle. Product teams identify assumptions and formulate them into hypotheses about users and their problems, product concepts and features, and business growth that they test in experiments and subsequently make sense of the data that they gather (Figure 4). Hypotheses also have become the *lingua franca* for keeping track and communicating the progress of research projects.

Guiding all research activities with the HPF has been instrumental to prepare for a large-scale event like Developer Day. Each PM and Engineer pair uses the HPF to formulate a set of relevant hypotheses to test during their sessions. The hypotheses guide the team to define the target user profile, the choice of methodology, and help to keep focus and take the most advantage of the speed sessions.

For instance, when observing telemetry from the Visual Studio (VS) IDE[3] for Windows, the VS team noticed that many users created new projects but stopped working on them shortly after. One of the team's assumptions was that the experience of starting a work session with VS—either by creating a new project or by opening an existing one—had room for improvement. To start the investigation, the VS team came up with several hypotheses of what the problems could be and how to solve them. Here are some

[2] A 2020 version of the event was planned but later cancelled due to the COVID-19 pandemic. We discuss this in the Limitations and Challenges section.
[3] Integrated Development Environment.

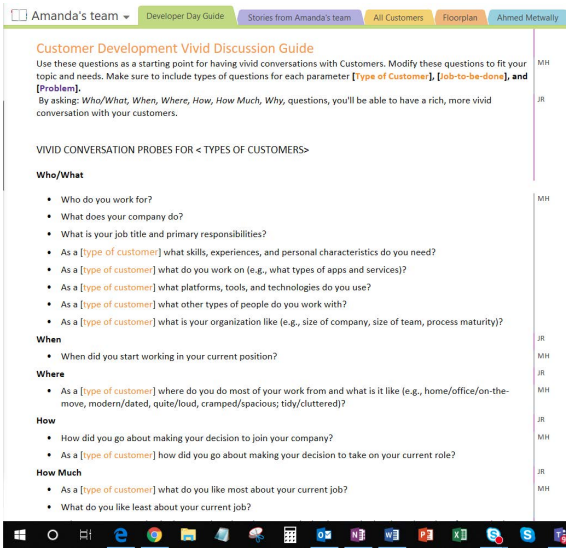

Figure 5: Notebook containing schedules, user profiles, hypotheses, discussion guides, and notes.

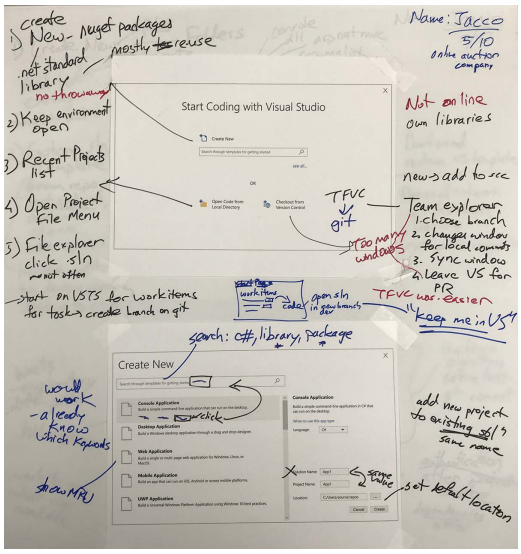

Figure 6: Paper prototype used during Developer Day to study the "Get to Code" screen and the New Project Dialog.

examples of the hypothesis they proposed and sought to validate (or invalidate):

- **[Problem Hypothesis]** We believe that Developers[4] *do not find adequate support in VS for finding and using the code scaffolding* they need when starting a new project.
- **[Problem Hypothesis]** We believe that Developers find it *difficult to open external code* (e.g. from a repository) in VS.
- **[Concept Hypothesis]** We believe that *a search-focused model for finding project templates* will make it easier for Developers to find the right template to start new VS projects.
- **[Concept Hypothesis]** We believe that *integrating an option for checking-out code from source control along with an "open local code" option in a single starting point* is a more intuitive workflow for Developers starting a work session with VS.

These hypotheses allowed the team to define the target profile, that is, users of VS for Windows[5]. Based on the hypotheses, the team also considered to talk to both newcomers as well as long time users to understand the different needs of both profiles, and to study the interaction with common code version control systems. The *Problem Hypotheses* suggested that an observational approach (likely a usability study) might be helpful to identify the bumps that users find along the way. Conversely, the *Concept Hypotheses* pointed towards a collaborative design activity involving users to develop the concepts towards an improved experience. Finally, the types of data usually collected with these methods (usability metrics and observations in the former, design artifacts and subjective opinions in the latter) and the hypotheses to validate, will help to choose what analysis methods to use.

3.2 Meet Teams Where They Are

We have realized that preparation is crucial for taking the most advantage of a full day of intense UX research activities. Prior to Developer Day, the UX Research team provides every PM and Engineer (and UX designers, when applicable) in the Developer Division with the training and tools they need for conducting Lean UX research based on the HPF. Training includes large sessions open to all employees and smaller meetings with individual product teams. UX Researchers use the product teams' hypotheses as a basis to provide advice specific to each research project, including strategies on how to connect with users. UX Researchers also help teams with preparing necessary materials (e.g discussion guides) and artifacts, such as the paper prototype and portable whiteboard depicted in Figure 6. Through these actions, the product teams go into their speed sessions with a plan for how to collect data about what they are trying to learn. By meeting them where they are, the UX Research team helps to ensure that product teams can be successful, no matter where they are in their product's lifecycle.

3.3 Leverage Lightweight Tools

To support a large-scale event like Developer Day, we have found it is necessary to use tools and frameworks that are flexible enough to meet each product teams' specific needs. Teams are less likely to engage with an activity like this if it adds the overhead of starting over with new complex tools. Whenever possible, we embrace tools already in use by product teams. For example, if a product team is using Slack or Microsoft Teams for their primary communication, we encourage them to leverage those platforms to share their findings.

Additionally, each product team gets its own notebook[6] going into Developer Day (Figure 5.). These notebooks include their individual schedules for the day, a profile for each user they will speak to, and a place to keep their hypotheses, discussion guides, and notes. Many product teams have been using this tool in their everyday work, thus when it comes time to use them for Developer

[4]Most of our users are software developers.
[5]We also develop the VS for Mac and VS Code IDEs, which are not relevant to these hypotheses.

[6]Most frequently we use Microsoft OneNote, which is a common work tool across the company

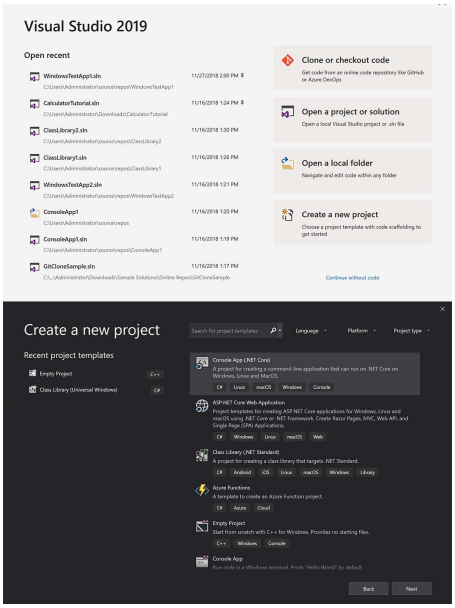

Figure 7: New designs of the "Get to Code" screen and New Project Dialog.

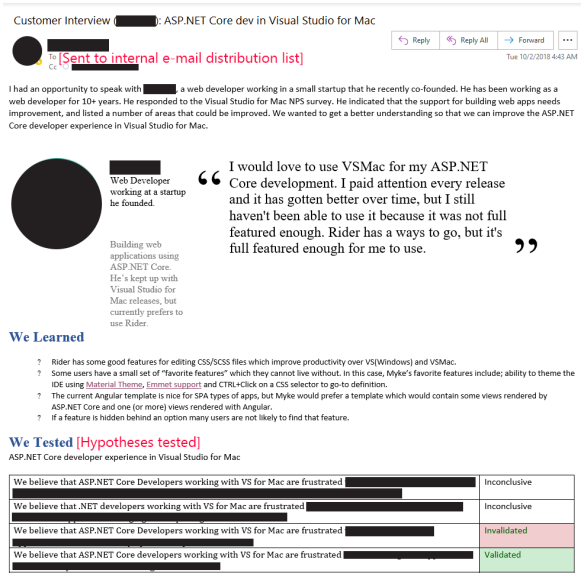

Figure 8: Example of a user story sent to our e-mail distribution list. Personal and confidential information has been redacted.

Day, they are ready and prepared. After the event, the notebooks become repositories of raw data for later reference and analysis.

Product teams also have the flexibility to use other tools that are best suited to collect their data. For instance, during the Developer Day speed sessions, the VS team used paper prototypes [13] to invite participants to write or draw the options they expected to find when VS first starts. Figure 6 shows a paper prototype annotated by one of the Developer Day participants. Besides input on these prototypes, the team also interviewed participants about their common behaviors when starting a coding session. After analysis, several common themes were identified and triangulated with other research (e.g. telemetry and lab studies) to increase confidence. The results were later conducive to the redesign of the Get to Code screen and the New Project Dialog in the 2019 version of Visual Studio (Figure 7, [10, 15]).

3.4 Make Data Representative and Relatable

Studies have shown that data alone is not enough to inspire people to take action [12, 14]. Study results must not only be accurate, but they also need to help teams empathize with their users. We have observed this first-hand: employees who attend lab study sessions state that they feel motivated to improve the tools they work on after observing users running into problems when using those tools. The same occurs to employees during Developer Day: listening to real-life stories directly from users in face-to-face conversation seems to elicit a feeling of empathy and inspiration. As expressed by a program manager who participated in Developer Day in 2018, *"as people [working] in technology, sometimes we don't understand that emotions have a direct correlation with the work that [users] are doing. The empathy part was something that we missed. If you get to those pain points or those levels of excitement, then you can really understand how you can improve somebody's work".*

We apply these observations when communicating results—either from Developer Day or from daily research activities—with the organization. One particular approach that we have found to create empathy is the use of *vivid stories* from actual users as opposed to fictitious personas built from a synthesis of several people. We include descriptions of profiles from actual users, highlighting their motivations, technology adoption patterns, cultural norms, and other critical user contexts. We use these profiles to tell stories that help bring research insights to life. This can be accomplished by applying various methods for eliciting empathy, such as including photos, videos, and quotes and using illustrations like journey maps, workflow diagrams, or storyboards [5]. Figure 8 is an anonymized example that includes a user story, a user photo and quotes. The UX Research team provides product teams with training and guides on how to tell these vivid stories about what they have learned from the users they have met. In terms of tools for sharing out, we use an e-mail distribution list where each interaction with a user is summarized and e-mailed to the entire organization. Summaries link back to the notebooks, in case a reader wants to know more details about the session. This system allows us to create a rich and searchable repository of all our user research results without having to manage and deploy a new tool. The example in Figure 8 shows a result summary from a study on the experience of VS for Mac users building Web applications with the ASP.NET platform[7].

4 IMPACT ON THE ORGANIZATION

Developer Day has a positive impact for the Developer Division at multiple levels. In this section we discuss research outcome metrics

[7] https://dotnet.microsoft.com/apps/aspnet

and time and cost savings as a result of research conducted during the event.

4.1 Learnings Volume

As mentioned earlier, the Developer Division uses an email distribution list to share learnings with the organization about their studies with users. In order to quantify the learning outcome of Developer Day, we analyzed the monthly volume of emails sent to the list, averaged across the 3 years the event has taken place. We found that in the month following Developer Day, the volume of emails increased by over 20% compared to the rest of the year.

4.2 Time and Cost Savings

Developer Day speed sessions are 20 minutes each and we do around 600 sessions. That yields an execution time of about 200 total hours across the employees involved. It is important to notice that a typical research session requires a time investment at the beginning and end for setting up, explaining goals and methods, debriefing, etc. Most of these steps are common to all speed sessions and they are completed only once during the event. We estimate this saves around 5 minutes per each 20 minute speed session, a 25% increase in data collection efficiency. Over the course of 600 sessions, this translates into 3000 minutes dedicated to actual research versus an introduction and setting research context. Therefore, the time savings for this type of approach can be substantial.

Additionally, around 100 different topics are investigated at Developer Day, which otherwise would take the form of 100 different studies. Obviously, considerable time goes into preparing for Developer Day, much more than any single other study. However, we save preparation time by attaching Developer Day to the Build conference—expediting recruitment—, and gathering participants and employees in a single place at the same time. The combined preparation of 100 studies altogether would likely take longer than that of Developer Day.

In financial terms, we are paying for an event that includes about 100 users producing 600 sessions across 100 different study topics. Without being able to disclose actual numbers, if we calculate the average cost for 100 studies with 6 participants each, the total would be more than 3 times that of Developer Day.

5 DISCUSSION

While Developer Day helps teams gain valuable user and product insights, the event has become much more than just collecting user data. More importantly, the event acts as a catalyst for creating user-centered culture growth. Many junior PMs and Engineer product team employees have direct conversations with users for the first time at Developer Day. The exposure serves as a launching pad for future user interactions. *"I found it useful to be able to interact directly with users and understand their perspective on features I interact with every day,"* said an employee that participated in 2018. The training and tools that employees receive from the UX Research team for Developer Day are applied to future sessions they have with users.

Additionally, Developer Day is a spark for subsequent continued or new research for product teams. Initial insights are gained from the speed sessions that need to be further validated or that generate new hypotheses to test. This following research can take the form of various methodologies.

Figure 9: Tweet from a Developer Day attendee.

It is important to note that Developer Day is also a social event that stimulates our user community. For instance, there are meal breaks along the day that employees and participants use to meet, to have informal conversations and to share experiences. Therefore, Developer Day not only benefits our employees: after an intense day of formal and informal interactions, participants leave feeling energized and hopeful about the future of Microsoft and our products. As put by a Developer Day attendee: *"[It is] a great opportunity to tell Microsoft exactly what I need; and I love it that Microsoft is becoming more transparent. So, I feel like you're sharing your world with me."* When discussing feedback he had given in 2018, a second-time attendee explained that *"seeing that suggestion become something baked into the product was like 'whoa!'."* The tweet in Figure 9 is another example.

We invite leaders from other organizations within Microsoft outside of the Developer Division to give them a chance to, not only participate in the event, but to learn more about our group's culture. In doing so, we can advance user-centered culture growth beyond our own organization and across the entire company. *"We took a lot of leaders from other divisions to come, observe and learn, because we feel this process can be further scaled out to more Microsoft teams,"* explains the CVP of the Developer Division.

5.1 Limitations and Challenges

An event like Developer Day represents a number of challenges and limitations. First, when recruiting such a large amount of users at once to match a variety of investigation areas, it is not unusual that some participants do not meet the teams' full profile criteria. This leads to unproductive speed sessions and lost time. More screening

vetting can be done beforehand to ensure better matches (e.g., brief follow up surveys or interviews), but this represents a trade-off with the amount of work added to the preparation phase.

Team members are as diverse as users, which sometimes leads to inconsistencies in the processes. For instance, we observed that some product team members are better than others at taking notes in real-time during sessions, specially those doing it for the first time. More description of how valuable notes can be for learning could increase motivation for better note taking. Typewriting training is another alternative that might improve the quality of the collected data. We have also considered recording the sessions on audio or video, but we discarded the idea because of the relative noisiness of the environment in which Developer Day is conducted, as well as due to privacy concerns. In any case, we embrace the differences in style and prefer to steer away from strict note-taking formats, adhering to the "meeting teams where they are" and "leveraging lightweight tools" principles, as explained in the Lessons Learned section.

Another big challenge we have faced is the work needed before the event. Among all the preparation activities, creating the master schedule has proven to be a major investment in time and effort. So far, we have used a spreadsheet consisting of over 100 rows (users) by over 100 columns (product teams), that need to be intersected at six points (speed sessions) with no overlaps. This yields a large sparse matrix which is difficult to maintain without errors, which has been a major headache for the UX Research team. We need to investigate whether existing event management software can aid with this task, or otherwise to develop a purpose-specific application in-house.

Finally, Microsoft has adapted a number of events such as conferences (including Build itself), workshops and large meetings to remote formats. Due to its nature, adapting Developer Day would present a number of particular challenges. We are redesigning Developer Day to make sure we can continue applying Lean UX research at scale in a remote scenario, while keeping one of the event's major value propositions—the close interaction between users and product teams.

6 CONCLUSION

Developer Day has been a successful way to practice Lean UX at scale. The event promotes the involvement of product teams in UX research, while researchers help such involvement with support and guidance. Through a speed session format of mixed methods, training and tools, and a highly collaborative approach across all organizational roles, we are together able to scale our UX research capacity an order of magnitude greater. This leads to an impact on strategy and design decisions for our products. More importantly, we are progressing user-centered culture growth that endures beyond the event and scales even further. There is much more that can be done to evolve a Developer Day type event. First, a wider range of research methodologies can be employed. Second, the audience can be expanded beyond a single group to other organizations across the company. We have begun doing this with other organizations inside Microsoft in the past Developer Day events, but it can be taken much further. Recently, we did a "Customer Day" that involved nearly 50 users and over 125 employees comprised from organizations across Microsoft resulting in nearly 150 speed sessions. Third, the event could be done more frequently than annually. We now include a "Customer Day" segment in a weeklong onboarding "bootcamp" to new employees of the Developer Division. These bootcamps happen quarterly each with an average of over 50 employees that conduct close to 50 speed sessions with nearly the same number of users. Last, with more employees and resources, the scale of the event can be increased even further.

ACKNOWLEDGMENTS

We thank the Developer Division leadership and employees, and most importantly, the event attendees, without whom none of this would be possible.

REFERENCES

[1] Giff Constable. 2014. Talking to humans. *Giff Constable* 1 (2014), 71.
[2] Graham R Gibbs. 2007. Thematic coding and categorizing. (2007).
[3] Colin M Gray. 2016. " It's More of a Mindset Than a Method" UX Practitioners' Conception of Design Methods. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 4044–4055.
[4] Bill Iuso. 1975. Concept testing: an appropriate approach. *Journal of Marketing research* 12, 2 (1975), 228–231.
[5] Jim Kalbach. 2016. *Mapping experiences: A complete guide to creating value through journeys, blueprints, and diagrams*. O'Reilly Media, Inc.
[6] Finn Kensing and Jeanette Blomberg. 1998. Participatory design: Issues and concerns. *Computer supported cooperative work (CSCW)* 7, 3-4 (1998), 167–185.
[7] Lassi A Liikkanen, Harri Kilpiö, Lauri Svan, and Miko Hiltunen. 2014. Lean UX: the next generation of user-centered agile development?. In *Proceedings of the 8th nordic conference on human-computer interaction: Fun, fast, foundational*. 1095–1100.
[8] Travis Lowdermilk and Jessica Rich. 2017. *The Customer-driven Playbook: Converting Customer Feedback Into Successful Products*. O'Reilly Media, Inc.
[9] Andrés Lucero. 2015. Using affinity diagrams to evaluate interactive prototypes. In *IFIP Conference on Human-Computer Interaction*. Springer, 231–248.
[10] Pratik Nadagouda. 2019. *Redesigning the New Project Dialog*. https://devblogs.microsoft.com/visualstudio/redesigning-the-new-project-dialog/
[11] Jakob Nielsen. 2001. *Usability Metrics*. https://www.nngroup.com/articles/usability-metrics/
[12] Keith Oatley. 1999. Why fiction may be twice as true as fact: Fiction as cognitive and emotional simulation. *Review of general psychology* 3, 2 (1999), 101–117.
[13] Marc Rettig. 1994. Prototyping for tiny fingers. *Commun. ACM* 37, 4 (1994), 21–27.
[14] Greg J Stephens, Lauren J Silbert, and Uri Hasson. 2010. Speaker–listener neural coupling underlies successful communication. *Proceedings of the National Academy of Sciences* 107, 32 (2010), 14425–14430.
[15] Cathy Sullivan. 2018. *Get to code: How we designed the new Visual Studio start window*. https://devblogs.microsoft.com/visualstudio/get-to-code-how-we-designed-the-new-visual-studio-start-window/