Spreadsheet use and programming experience

An exploratory survey

Sarkar, Advait; Borghouts, Judith W.; Iyer, Anusha; Khullar, Sneha; Canton, Christian; Hermans, Felienne; Gordon, Andrew D.; Williams, Jack

DOI
10.1145/3334480.3382807

Publication date
2020

Document Version
Final published version

Published in
CHI EA 2020 - Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems

Citation (APA)
Sarkar, A., Borghouts, J. W., Iyer, A., Khullar, S., Canton, C., Hermans, F., Gordon, A. D., & Williams, J. (2020). Spreadsheet use and programming experience: An exploratory survey. In *CHI EA 2020 - Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* Article 3382807 (Conference on Human Factors in Computing Systems - Proceedings). Association for Computing Machinery (ACM). https://doi.org/10.1145/3334480.3382807

Spreadsheet Use and Programming Experience: an Exploratory Survey

Advait Sarkar[1,4]
Judith W. Borghouts[1]
Anusha Iyer[1]
Sneha Khullar[2]
Christian Canton[2]
Felienne Hermans[3]
Andrew D. Gordon[1,5]
Jack Williams[1]

[1]Microsoft Research
Cambridge, United Kingdom
advait@microsoft.com
i.anusha@gmail.com
judith_borghouts@hotmail.com
adg@microsoft.com
t-jowil@microsoft.com

[2]Microsoft
Redmond WA, USA
snkhulla@microsoft.com
chcanton@microsoft.com

[3]Delft University of Technology
Delft, Netherlands
f.f.j.hermans@liacs.leidenuniv.nl

[4]University of Cambridge
Cambridge, United Kingdom

[5]University of Edinburgh
Edinburgh, United Kingdom

CHI '20 Extended Abstracts, April 25–30, 2020, Honolulu, HI, USA.

ACM ISBN 978-1-4503-6819-3/20/04.
http://dx.doi.org/10.1145/3334480.3382807

Abstract
We report results from a survey on spreadsheet use and experience with textual programming languages ($n = 49$). We find significant correlations between self-reported formula experience, programming experience, and overall spreadsheet experience. We discuss the implications of our findings for spreadsheet research and end-user programming research, more generally.

Author Keywords
spreadsheets; programming; expertise; experience; end-user programming; survey

CCS Concepts
•**Social and professional topics** → **Computing literacy;** Informal education; •**Applied computing** → **Spreadsheets;** •**Human-centered computing** → *Empirical studies in HCI;* •**General and reference** → Empirical studies;

Introduction
Spreadsheets are an empowering technology; users can store, manipulate, and analyse data for their own benefit. Yet there is a wide spectrum of spreadsheet expertise, which creates disparities between users. Recent analyses have shown that as little as ∼7% of spreadsheets contain formulas, suggesting that much spreadsheet use consists of little more than data storage and manipulation [2].

Previous work has shown how spreadsheet expertise is acquired in 'informal, opportunistic, and social' ways [28] – learnt primarily on the job, from informal resources such as online fora, and help from colleagues [23]. However, while previous work has focused on how the social environment can foster the acquisition of spreadsheet expertise, it has not explored how the rest of the users' technical ecosystem may play a role. The workplace as well as the education landscape has changed. It is increasingly common for people to acquire formal experience in computing and programming, whether it is through a computing curriculum at school,[1] after school code clubs,[2] or at university – where there is growing demand for programming courses in non-computing degrees [8], or at work, through personal efforts to become 'conversational' at programming [9].

In this paper, we aim to open up a relatively under-explored theme within the field of programming expertise, namely, the interplay of traditional programming expertise (in a textual programming language such as Python or Java), and end-user programming in spreadsheets. This contrasts with previous research in end-user programming expertise, that has viewed people through the narrow slice of their experience with a specific tool. In doing so we seek to better understand, and to be better prepared for, the impact of increased programming literacy on spreadsheet expertise. For instance, may we assume that an increase in formula expertise within the population will naturally follow from the increase in programming literacy?

Our research question is, therefore: *is programming experience related to spreadsheet experience?* We present the design and results of an exploratory survey to find out.

[1] https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study

[2] https://codeclub.org/en/

Background

Theories of programming expertise

There are multiple theories for the development of programming expertise. The leading theory holds that learning to program consists of a knowledge-restructuring process, leading to the development of hierarchical schemata – units of knowledge – that allow people to succeed at the activity of programming [10]. This theory derives from constructivist [6] and constructionist [24] perspectives. However, recent work has questioned the constructivist account of learning, suggesting that this view has been received and perpetuates due to incidental social circumstances, and that alternatives remain extremely under-explored, especially when compared with language or mathematics education [17].

While the mechanism of expertise acquisition is still debated, certain differences between expert and novice programmers are consistently observed. Expert programmers have highly organised knowledge that allows them to do better at recall tasks [33, 7]. Beginner programmers understand individual lines of code but not the relationships between them; experts see the more abstract, overall pattern of a program [20]. The effect of working memory and experience on programming skill is mediated through programming knowledge [3]. Experts possess sophisticated mental imagery representing software designs, which they can mentally manipulate and externalise as design and communication resources [25].

Spreadsheet expertise

What motivates people to acquire spreadsheet expertise?

Spreadsheet learning tends to be goal-driven rather than structured [22, 28], an approach that could lead to lower quality spreadsheets as users do not acquire principles of design. Spreadsheet users are usually focused on understanding their problem domain, rather than understanding

Question & answer text
How would you classify your spreadsheet experience? ***(single choice)***
(1) Little or no experience
(2) Some experience, but I'm still a beginner
(3) A lot of experience, but my use is basic
(4) A lot of experience, and I use some advanced features
(5) A lot of experience with many advanced features
What is the main purpose of spreadsheets you use? ***(multiple choice)***
- Maintaining lists (e.g. names and addresses)
- Tracking data (e.g. budgets, sales, inventories)
- Analysing data (e.g. financial, operational)
- Determining trends and making projections
- Other *(free text response)*

Table 1: Main questionnaire items in our final survey (part 1).

programming, and therefore while they might become more familiar with their domain due to spreadsheet experience, they might not necessarily become better programmers [15]. Learning out of necessity rather than curiosity has also been found for web designers [11].

How is spreadsheet expertise acquired?
Small groups of people within organisations have been found to be responsible for sharing files, establishing and perpetuating 'informally-defined norms of behaviour' [21]. While beginners learn spreadsheets mainly socially through colleagues, experts are more likely to further their knowledge using books, manuals and online resources, and in either case formal training is not common [19, 23, 28]. There is support for *constructivist* and *constructionist* accounts of spreadsheet expertise development; a think-aloud study of 10 participants [26] found that participants who self-explain while trying to learn how to use spreadsheets prove to be better problem solvers. A further study found that problem solving is a more effective method for acquiring spreadsheet skills than watching tutorials [18].

What differentiates spreadsheet experts from non-experts?
The flexibility of spreadsheets permits a variety of 'coping mechanisms' for users to deal with low expertise, without having to acquire additional expertise [28]. In some cases, these coping mechanisms can be characterised as 'bad practice' [19], which differentiates experts and non-experts; experts perform more planning and design activities when writing spreadsheets. Experts exhibit greater appreciation for the legibility of formulas to themselves and others, helping them avoid 'smells' [16]. Non-expert spreadsheet users sometimes collaborate with experts who can complete high-expertise tasks, therefore alleviating the need to learn, although this collaboration sometimes has an informal learning outcome for the non-expert [23].

Measuring expertise
Previous work has used several indicators as direct or indirect measures of programming experience. A comprehensive review of 161 papers that measure programming experience is given by Feigenspan et al. [14], which we will not repeat in detail. In summary, they found eight categories of measures (plus a ninth category of 'unreported'): years of programming experience, education level, self-estimation, unspecified questionnaire, size of programs written by the subject, unspecified pre-test, and skill estimation by the subject's supervisor/manager.

Ad-hoc programming tasks or tests are sometimes used [4]. Months or years of 'tenure' as participants on a project or on a platform such as GitHub are also sometimes used as proxies [32]. The measurement of *end-user* programming expertise specifically is under-explored, although one survey did show that Likert-scale questionnaire items could reliably be used to measure end-user computing skills [31].

In our exploratory study, we needed a concept that could be operationalised as a questionnaire item. We therefore choose to focus on self-reported experience, rather than developing mechanical tests of expertise, as previous work has shown that it is possible for respondents to self-report in a consistent manner [14, 30]. Self-reported experience measures do have the limitation that respondents may give levels different interpretations; respondents with lower experience may inadvertently place themselves on a higher experience level (a version of the Dunning-Kruger effect [13]), and respondents with higher experience may rate themselves lower due to greater self-awareness or modesty. Nonetheless, the provision of additional information to help respondents self-assess, as well as the opportunity to consult with the experimenters for clarification, can greatly mitigate these effects. For example, in some cases where

Question & answer text
Which best describes your expertise with spreadsheet formulas? *(single choice)*
(1) I don't know what they are
(2) I know what they are but don't use them
(3) I only use a few basic functions (such as SUM and AVERAGE) in my formulas. I don't know how to do more advanced things.
(4) I only use a few basic functions in my formulas. I know how to do more advanced things, but I rarely / never need to.
(5) I use a variety of different functions in my formulas
(6) I use a variety of functions. I have written my own functions using VBA, or installed add-ins that make new functions available.
Which best describes your programming experience in traditional programming languages, such as Java, Python, C, SQL, R, JavaScript, VBA, etc.? *(single choice)*
(1) I have never programmed
(2) I have learnt a little bit but never used it
(3) I know enough to use it for small infrequent tasks
(4) I am moderately experienced and write programs regularly
(5) I am highly experienced
(6) I program or have programmed in a professional capacity
(7) Other *(free text response)*

Table 2: Main questionnaire items in our final survey (part 2).

the sample is from a well-defined group (e.g., a university course cohort), response consistency on a self-reported experience question can be improved by asking respondents to rank their own position within the peer group [27]. An additional advantage of self-reporting is that it allows the opportunity to capture aspects of expertise that are hard to measure using tests (such as planning and design skills), but which make up a significant portion of expertise as manifested.

Survey

The survey was implemented using Microsoft Forms.[3] Participants were first presented with a brief description of the survey and signed a form of informed consent. Participants optionally provided the following demographic information: age range and occupation. Participants also noted the specific spreadsheet application(s) they normally used. We did not record the gender or location of participants.

To test the questionnaire we conducted a pilot survey with a convenience sample of 15 respondents. After they completed the survey, we discussed the appropriateness and effectiveness of the questionnaire items with our pilot participants. Our pilot prompted several adjustments to the survey design and implementation. The main questions used in our final questionnaire can be found in Tables 1 and 2. For brevity, we have excluded some questions such as optional demographics.

Participants

After revising the questionnaire, we recruited a fresh sample of 57 participants, located mostly in the UK and USA, through a combination of convenience and snowball sampling. This is unlikely to be representative of the global population. In our report, we therefore avoid drawing general inferences of population-level distributions, but instead focus on within-subjects correlations along different measures. The following general results serve purely to illustrate the constitution of our participants.

- **Age**: 56 respondents chose to provide their age range (1 declined). Of those that provided age range, the median and mode age range was 25-34.
- **Applications used**: all 57 participants reported regular use of Microsoft Excel, with 43 (75%) being exclusive Excel users. Twelve (21%) reported use of Google Sheets and Excel. One participant reported use of OpenOffice Calc and Excel, and another participant reported use of Apple Numbers and Excel.
- **Spreadsheet use cases**: previous work had revealed four categories of spreadsheet use [19], which we tracked using a questionnaire item. Analysing data (e.g., financial, operational) was the most common use case, regularly encountered by 43 respondents (75%). Nonetheless, all use cases were common amongst some respondents, with the least common use case ('Determining trends and making projections') still encountered by 10 respondents (18%).
- **Occupations**: respondents reported a wide range of occupations. They included academic research, business administration, marketing, sales, healthcare, education, and financial analysis.

Results

We assigned numeric codes to the responses for the questionnaire items regarding self-reported spreadsheet experience, formula experience, and programming experience. These were assigned integer codes from 1-5, 1-6,

[3] https://forms.office.com

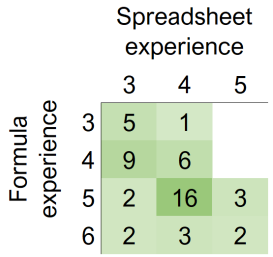

Figure 1: Counts of users at each level of formula and spreadsheet experience (higher is more experienced).

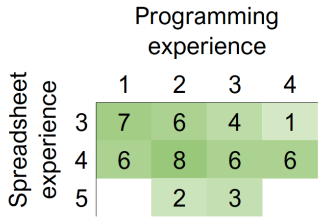

Figure 2: Counts of users at each level of spreadsheet and programming experience (higher is more experienced).

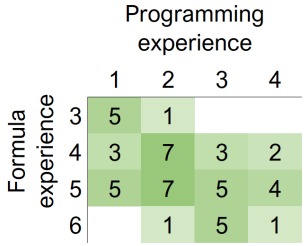

Figure 3: Counts of users at each level of formula and programming experience.

and 1-6 respectively, with 1 denoting the point of lowest self-reported experience. These codes establish an ordinal ranking of responses. Throughout our analysis, the correlations we report are Spearman's ρ (r_s). We apply a Bonferroni correction to our significance threshold α. For brevity, we henceforth use the word 'experience' to mean our ordinal scale measure of participants' self-reported experience. Certain groups were underrepresented in our sample. Programming experience levels 5 and 6, formula experience levels 1 and 2, and spreadsheet experience levels 1 and 2 all had 3 or fewer participants. We removed these groups from our correlation analysis, leaving us with a final sample of $n = 49$.

We find that spreadsheet experience and formula experience are positively correlated (Spearman's $r_s = 0.52$, $p = 1.3 \cdot 10^{-4}$). See Figure 1. This is the least surprising of our observed correlations, as much advanced spreadsheet use involves formulas.

We find that, although spreadsheet experience has a weak positive correlation coefficient with programming experience, this correlation is not statistically significant (Spearman's $r_s = 0.26, p = 0.07$). See Figure 2. Respondents with different levels of spreadsheet experience all showed a wide range of self-reported programming experience.

We find that formula experience and programming experience are positively correlated (Spearman's $r_s = 0.42$, $p = 2.6 \cdot 10^{-3}$). See Figure 3. This tells us that while we cannot assume that experienced spreadsheet users in general are more likely to have some experience of programming in a textual programming language, we *can* expect that spreadsheet users who are highly experienced at using *formulas* are more likely to also have experience of programming in a traditional language. This is discussed in greater detail in the next section.

We included a questionnaire item about spreadsheet use cases identified in previous work [5]. We did not observe any correlation between spreadsheet experience and the four use cases we track (determining trends, analysing data, tracking data, and maintaining lists), although the first two did skew slightly towards higher experience.

Discussion

Recall that we use 'experience' to mean our ordinal scale measure of participants' self-reported experience. We observed a correlation between spreadsheet and formula experience. This was expected; much of utility of spreadsheets is derived from formulas, which transform spreadsheets from passive datastores into active programs, and many features of spreadsheets are supported by formulas. For example, conditional formatting and data validations can both be specified in terms of formulas, and the name manager is only useful if names are referenced in formulas.

Furthermore, we observed that spreadsheet experience is not (significantly) correlated with programming experience. This can be attributed to the diversity of ways in which spreadsheet experience can manifest. Recall that we did not observe a relationship between use cases and spreadsheet experience. Nonetheless, certain use cases for spreadsheets (such as tracking data, and maintaining lists) require very little or no formula use. It is possible, therefore, to have high levels of experience using spreadsheets in this manner, without necessarily acquiring expertise in formula authoring (and by extension, in programming). Other phenomena, such as the delegation of formula authoring to others with more expertise [23], may also help in maintaining this diversity.

Our most interesting observation is that formula experience is correlated with programming experience. To our knowl-

edge, we are the first to observe this phenomenon. There are multiple potential explanations for this. One possibility is that there is a direct causal link between the acquisition of formula expertise and programming expertise. That is, if a spreadsheet user has pre-existing expertise in a traditional programming language, they are able to immediately translate that into higher expertise in formula authoring. Computing education research has shown that acquiring expertise in a second programming language is considerably easier than the first [29], due to the pre-existence of suitable mental models (e.g., notional machines [12]). Conversely, pre-existing expertise in formula authoring may help users acquire expertise in traditional programming languages.

Another possibility is that there is an underlying causal factor that prompts expertise acquisition in both formulas and traditional programming. Such an underlying factor may either be intrinsic or extrinsic. Aghaee et al. [1] have provided a personality-based account for intrinsic motivation in end-user programming. In particular, their research shows that certain personality profiles (which may be characterised as artistry, bricoleurism, and technophilia) are intrinsically predisposed to programming. Extrinsic factors may include problems in the user's work domain or personal life that can only be solved through the acquisition of programming skill. For instance, Sarkar and Gordon [28] found that skill acquisition in spreadsheets was often motivated by specific workplace problems.

These possibilities, i.e., mechanisms of influence between formula expertise, programming expertise, and underlying causal factors, are not mutually exclusive. Indeed, they are likely to interplay with each other. People may traverse a variety of paths through these, with different needs. For example, one person may learn traditional programming due to a workplace need, and then translate that into formula expertise. Another may learn spreadsheet formulas due to an intrinsic interest in spreadsheets, and then apply that knowledge to traditional programming. The nature of this interplay is likely to be highly nuanced and individualistic. In future work, it would be interesting to explore how this knowledge is acquired and transferred across different programming paradigms.

A limitation of our study was that we initially designed the survey as an interview recruitment tool. We wished to interview participants about how they learnt to use spreadsheets. We designed the survey to help us recruit participants with a range of spreadsheet and programming expertise. Thus we did not include items that would have enabled us to make causal inferences. Nonetheless, we found an interesting pattern that suggests a relationship between spreadsheet and programming experience. In future, we may obtain greater understanding of this relationship with results from our interviews.

Conclusion

Our paper explores the understudied relationships between experience in traditional programming and end-user programming, particularly in spreadsheets. Our study recognises that end-user programming systems and their users do not exist in isolation; there are relevant experiences from outside these systems that may strongly influence them, or be influenced by them. It is a simple premise that has not previously been shown through quantitative data. In our survey sample, we have found that experience in spreadsheet formula usage is correlated with experience in programming in a traditional language. We have discussed multiple possible relationships between experience in traditional programming and formula authoring. It would be worthwhile to explore this trend using a redesigned survey and a larger sample size.

REFERENCES

[1] Saeed Aghaee, Alan F Blackwell, David Stillwell, and Michal Kosinski. 2015. Personality and intrinsic motivational factors in end-user programming. In *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 29–36.

[2] Titus Barik, Kevin Lubick, Justin Smith, John Slankas, and Emerson Murphy-Hill. 2015. Fuse: a reproducible, extendable, internet-scale corpus of spreadsheets. In *Proceedings of the 12th Working Conference on Mining Software Repositories*. IEEE Press, 486–489.

[3] Gunnar Rye Bergersen and Jan-Eric Gustafsson. 2011. Programming skill, knowledge, and working memory among professional software developers from an investment theory perspective. *Journal of individual Differences* (2011).

[4] Gunnar R Bergersen, Dag IK Sjøberg, and Tore Dybå. 2014. Construction and validation of an instrument for measuring programming skill. *IEEE Transactions on Software Engineering* 40, 12 (2014), 1163–1184.

[5] Judith Borghouts, Andrew D. Gordon, Advait Sarkar, Kenton P. O'Hara, and Neil Toronto. 2019. Somewhere Around That Number: An Interview Study of How Spreadsheet Users Manage Uncertainty. *CoRR* abs/1905.13072 (2019). `http://arxiv.org/abs/1905.13072`

[6] Robert L Campbell, Norman R Brown, and Lia A DiBello. 1992. The programmer's burden: developing expertise in programming. In *The psychology of expertise*. Springer, 269–294.

[7] Michelene TH Chi. 2006. Laboratory methods for assessing experts' and novices' knowledge. *The Cambridge handbook of expertise and expert performance* (2006), 167–184.

[8] Parmit K Chilana, Celena Alcock, Shruti Dembla, Anson Ho, Ada Hurst, Brett Armstrong, and Philip J Guo. 2015. Perceptions of non-CS majors in intro programming: The rise of the conversational programmer. In *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 251–259.

[9] Parmit K Chilana, Rishabh Singh, and Philip J Guo. 2016. Understanding conversational programmers: A perspective from the software industry. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 1462–1472.

[10] Simon P Davies. 1994. Knowledge restructuring and the acquisition of programming expertise. *International Journal of Human-Computer Studies* 40, 4 (1994), 703–726.

[11] Brian Dorn and Mark Guzdial. 2010. Learning on the job: characterizing the programming knowledge and learning strategies of web designers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 703–712.

[12] Benedict Du Boulay. 1986. Some difficulties of learning to program. *Journal of Educational Computing Research* 2, 1 (1986), 57–73.

[13] David Dunning. 2011. The Dunning–Kruger effect: On being ignorant of one's own ignorance. In *Advances in experimental social psychology*. Vol. 44. Elsevier, 247–296.

[14] Janet Feigenspan, Christian Kästner, Jörg Liebig, Sven Apel, and Stefan Hanenberg. 2012. Measuring programming experience. In *2012 20th IEEE International Conference on Program Comprehension (ICPC)*. IEEE, 73–82.

[15] Thomas A Grossman. 2007. Spreadsheet engineering: A research framework. *arXiv preprint arXiv:0711.0538* (2007).

[16] Felienne Hermans, Martin Pinzger, and Arie van Deursen. 2015. Detecting and refactoring code smells in spreadsheet formulas. *Empirical Software Engineering* 20, 2 (2015), 549–575.

[17] Felienne Hermans and Marileen Smit. 2018. Explicit Direct Instruction in Programming Education. In *Proceedings of the 29th Annual Conference of the Psychology of Programming Interest Group (PPIG 2018)*. 86–93.

[18] Michael P Kerr and Stephen J Payne. 1994. Learning to use a spreadsheet by doing and by watching. *Interacting with Computers* 6, 1 (1994), 3–22.

[19] Barry R Lawson, Kenneth R Baker, Stephen G Powell, and Lynn Foster-Johnson. 2009. A comparison of spreadsheet users with different levels of experience. *Omega* 37, 3 (2009), 579–590.

[20] Raymond Lister, Beth Simon, Errol Thompson, Jacqueline L Whalley, and Christine Prasad. 2006. Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. *ACM SIGCSE Bulletin* 38, 3 (2006), 118–122.

[21] Wendy E Mackay. 1990. Patterns of sharing customizable software. In *Proceedings of the 1990 ACM conference on Computer-supported cooperative work*. ACM, 209–221.

[22] TJ McGill and MW Dixon. 2001. Spreadsheet knowledge: An exploratory study. (2001).

[23] Bonnie A Nardi and James R Miller. 1990. An ethnographic study of distributed problem solving in spreadsheet development. In *Proceedings of the 1990 ACM conference on Computer-supported cooperative work*. ACM, 197–208.

[24] Seymour Papert. 1980. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.

[25] Marian Petre. 2009. Insights from expert software design practice. In *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*. ACM, 233–242.

[26] Peter Reimann and C Neubert. 2000. The role of self-explanation in learning to use a spreadsheet through examples. *Journal of Computer Assisted Learning* 16, 4 (2000), 316–325.

[27] Advait Sarkar. 2015. The impact of syntax colouring on program comprehension. In *Proceedings of the 26th Annual Conference of the Psychology of Programming Interest Group (PPIG 2015)*. 49–58.

[28] Advait Sarkar and Andrew D Gordon. 2018. How do people learn to use spreadsheets? (Work in progress). In *Proceedings of the 29th Annual Conference of the Psychology of Programming Interest Group (PPIG 2018)*. 28–35.

[29] Jean Scholtz and Susan Wiedenbeck. 1990. Learning second and subsequent programming languages: A problem of transfer. *International Journal of Human-Computer Interaction* 2, 1 (1990), 51–72.

[30] Janet Siegmund, Christian Kästner, Jörg Liebig, Sven Apel, and Stefan Hanenberg. 2014. Measuring and modeling programming experience. *Empirical Software Engineering* 19, 5 (2014), 1299–1334.

[31] Gholamreza Torkzadeh and Jungwoo Lee. 2003. Measures of perceived end-user computing skills. *Information & Management* 40, 7 (2003), 607–615.

[32] Bogdan Vasilescu, Daryl Posnett, Baishakhi Ray, Mark GJ van den Brand, Alexander Serebrenik, Premkumar Devanbu, and Vladimir Filkov. 2015. Gender and tenure diversity in GitHub teams. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. ACM, 3789–3798.

[33] Susan Wiedenbeck. 2005. Factors affecting the success of non-majors in learning to program. In *Proceedings of the first international workshop on Computing education research*. ACM, 13–24.