Hellas, Arto; Zavgorodniaia, Albina; Sorva, Juha

Crowdsourcing in Computing Education Research: Case Amazon MTurk

Published in:
Koli Calling '20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research

DOI:
10.1145/3428029.3428062

Published: 19/11/2020

Document Version
Peer reviewed version

Please cite the original version:
Hellas, A., Zavgorodniaia, A., & Sorva, J. (2020). Crowdsourcing in Computing Education Research: Case Amazon MTurk. In *Koli Calling '20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research* (pp. 1-5). [32] ACM. https://doi.org/10.1145/3428029.3428062

Crowdsourcing in Computing Education Research: Case Amazon MTurk

Arto Hellas
arto.hellas@aalto.fi
Aalto University, Finland

Albina Zavgorodniaia
albina.zavgorodniaia@aalto.fi
Aalto University, Finland

Juha Sorva
juha.sorva@aalto.fi
Aalto University, Finland

ABSTRACT

Crowdsourcing platforms such as Amazon MTurk provide access to a human workforce that can be given tasks to complete online for a fee. In this article, we review studies in computing education research (CER) that rely on crowdsourcing; we also describe our own experiences of using Amazon MTurk for a CER study. We discuss challenges in recruiting workers with specific backgrounds—such as no programming experience—and considerations in filtering out unreliable research participants. Combining recommendations from the literature with the lessons that we learned whilst conducting our study, we synthesize advice for researchers in CER who are considering crowdsourcing. In our case study, we did not find widespread foul play by crowdsourced workers and, overall, our experiences and the literature suggest that crowdsourced CER is feasible. It is, however, uncertain to what extent crowdsourced data can produce answers that apply to specific educational contexts. More research is needed before definitive conclusions can be drawn about the validity and generalizability of crowdsourced CER.

CCS CONCEPTS

• **Social and professional topics** → Computing education.

KEYWORDS

crowdsourcing, computing education research, methodology, review, case study

ACM Reference Format:
Arto Hellas, Albina Zavgorodniaia, and Juha Sorva. 2020. Crowdsourcing in Computing Education Research: Case Amazon MTurk. In *Koli Calling '20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research (Koli Calling '20), November 19–22, 2020, Koli, Finland, Finland*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3428029.3428062

1 INTRODUCTION

Crowdsourcing refers to employing an outside workforce, a *crowd*, to perform a *task* [17]. A task can be virtually anything from a tiny survey to an immense effort such as creating Wikipedia. Crowdsourcing is particularly suitable for tasks that have a short time span and cannot be automated, and so, require human intelligence. The term can refer to asking volunteers to handle a task but also to recruiting paid workers [13].

Koli Calling '20, November 19–22, 2020, Koli, Finland, Finland

ACM ISBN 978-1-4503-8921-1/20/11…$15.00
https://doi.org/10.1145/3428029.3428062

Within computing education research (CER), some studies have enlisted students to construct and evaluate context-specific question repositories [9, 11, 24, 30]. Although some work in this vein [25, 31] describes itself as "crowdsourcing", we distinguish such "student-sourcing" from hiring external workers; we focus on the latter.

Hired workers have been used in CER for purposes ranging from participation in (preliminary) studies (e.g. [6, 33, 35]) to validations of earlier research findings (e.g. [20, 26, 36]).

The findings from using student-sourcing have been encouraging [10, 24], and the existing systems tend to employ a way to rate created content, which makes it possible to filter out poor content over time. Although there is evidence that students and paid workers perform similarly in certain conditions [3], crowdsourced data can be inaccurate or even completely invalid [8, 19]; it has been suggested that up to a third of data from a crowdsourcing platform could have serious quality issues [1]. Relatively little effort has been invested in analyzing the quality of crowdsourced data in CER.

In this article, we review studies in CER that have used fee-based crowdsourcing and describe our own methodological experiences from a CER study whose participants we recruited via Amazon Mechanical Turk. Our overall objective is to provide a methodological overview and a starting point for researchers who are considering crowdsourced data in computing education research.

2 AMAZON MECHANICAL TURK

Amazon Mechanical Turk (MTurk) "*is a crowdsourcing marketplace that makes it easier for individuals and businesses to outsource their processes and jobs to a distributed workforce who can perform these tasks virtually*" (from *https://www.mturk.com/*). On the platform, tasks are offered by *requesters* and completed by anonymous *workers*. A task's requester offers a fee for completing the task and monitors the quality of the workers' responses. The requester may reject responses that they deem unacceptable.

On the MTurk platform, offering a task starts by creating a new *project* by selecting a customizable template. MTurk's templates have a wide range of options that vary from simple surveys to sentiment analysis and video-categorization tasks. It is also possible to use a template with a link to an external system, thus enabling tasks to take place outside the MTurk platform itself. Once a template has been chosen, the requester fills in the task properties.

Task properties include: (1) a *task description* for candidate workers who browse through available tasks; (2) a *reward* for completing the task and the number of workers who can complete the task; and (3) *requirements*, i.e., criteria for prospective workers.

MTurk provides a selection of requirements for the task requester to choose from, such as location, age group, employment status, industry, and physical exercise habits. These are typically self-reported by the workers; although some requirements can be verified by MTurk (e.g., U.S. residents are required to provide valid taxpayer information), it is up to the requesters to choose whether

to trust the self-reported information. Importantly and unfortunately for CER, programming experience is not included in these self-reported criteria; researchers need to ask about it separately. In addition to requirements on personal background, MTurk-related requirements may be set, such as requiring workers to have previously completed at least a certain number of tasks or having reached a particular acceptance rate for past tasks.

The requester pays the fees to MTurk in advance. MTurk charges a minimum 20% commission, with an additional 20% if the number of workers is ten or more (as of 2020). Specific requirements for workers can increase the commission further.

The workers remain anonymous even when working with external systems. External systems are expected (but not obliged) to give a unique code to each worker, typically at the task's conclusion. The worker can enter the code into the MTurk platform to indicate task completion, which the requester can then verify.

3 CROWDSOURCING IN CER

In this section, we review crowdsourced research in CER (broadly construed). Based on the methodology and research foci of the studies, we have grouped them roughly in two categories: *surveys* and *system evaluations*. The two subsections below present these categories and are followed by a more general commentary on methodological and data-quality issues that arise from this review.

3.1 Surveys

Many studies on MTurk within and without CER take the form of surveys. MTurk provides templates and a custom XML-based language for constructing surveys. It has also been integrated with popular survey tools such as Qualtrix and SurveyMonkey.

Survey studies in CER include program comprehension studies where workers are expected to predict the output of given regular expressions [5] or programs [15], and perhaps also to assess the difficulty of the task and their confidence in their prediction [15].

A number of studies employed workers in order to identify preferences and consensus; for example, researchers have sought to identify preferences about the order of writing expressions such as `new byte[10 + length]` and `new byte[length + 10]` [4], about gradual typing semantics [34], and about specific features in a novel programming language [35].

Survey-like studies have also evaluated the quality of items in computer-constructed question banks [36] and categorized learning resources [7].

3.2 System Evaluations

In system evaluations, workers enter an external system, work with it for a while and then, often, answer a survey. The scopes of evaluations range from full environments and comparisons of environments down to specific features such as hint mechanisms.

In CER, crowdsourced evaluations have explored personified and non-personified feedback [20], labels and visual appearance of objectives [21], and multiple-choice questions with self-explanations in a novice programming environment [23]. Other studies have evaluated the instructional efficiency of visualizations for teaching algorithms [29], assessed approaches for teaching programming [22], and investigated the impact of working in a novice programming environment on adults' attitudes towards programming [6].

Several studies have evaluated specific system features that support program construction. These features include, for example, next-step hints and textual explanations [26, 27], hints generated from the inputs entered to programs [12], and refactoring support in a visual programming environment [33]. Learner-generated written annotations have been used for constructing tutorials shown to other learners [14].

3.3 Study Designs and Data Quality

The CER papers that we reviewed are studies that *use* crowdsourcing, not methodological papers *about* crowdsourcing. In these articles, researchers have not voiced major concerns about, or positioned themselves as being against, crowdsourcing for research purposes. Naturally, researchers who have put in significant effort into designing a study that relies on crowdsourcing are invested in the chosen research approach, which means there is an inherent potential bias in favor of the approach. This may in part explain the lack of negative commentary on crowdsourcing in these articles. As with much research, it is also unclear here whether the file-drawer effect—not writing up results that contradict researchers' expectations—has resulted in a publication bias that suppresses researchers' negative experiences with crowdsourcing. That being said, the reviewed studies do also report some "negative results," such as identifying that MTurk workers (or programmers in general) should likely not be used for finding a consensus for features of a new programming language [35], and observing that a particular feature of a system developed by the researchers is less efficient than other approaches [22].

Concerns about crowdsourced data quality have been highlighted in studies from other fields. Researchers have suggested that crowdsourced data can be inaccurate or even completely invalid [8, 19]. It has been recently suggested that up to a third of data from a crowdsourcing platform could have quality issues [1] and that many answers to "validity-indicator questions" (e.g., the number of children that the worker has) are obviously faulty (e.g., biologically unrealistic) [8], even when workers were required to have completed at least 50 tasks with a 85% acceptance rate. These concerns arise from studies in which researchers have, e.g., administered a validated personality measure surveys over time and have failed to replicate established findings. A particular concern is that the quality of the responses have reportedly decreased over the years, more recent data sets tending to be worse. [8] Similar concerns have been expressed earlier, too; for example, Kennedy et al. [19] observed an increase in fraudulent responses from particular countries. At the same time, these results may also depend on the task. In a comparison of undergraduate accounting students and MTurk workers, [3] noted that MTurk workers performed similarly to undergraduate accounting students on a task that did not require prior knowledge.

Within the CER studies that we have mentioned above, data-quality issues have largely not been discussed *per se*. Researchers have added requirements to workers, such as expecting a number of previously completed tasks with a high acceptance ratio. There also seems to be a tendency to trust self-evaluated reports of previous programming experience. In some cases, researchers have deliberately not mentioned the intended target group in MTurk's task description so as to eliminate self-selection bias; they have then

used only the responses of those workers who report themselves as novices (e.g. [21]).

In regards to whether MTurk was used as the sole source of data, or a complementary one, we saw a variety of approaches. Some of the studies were first conducted with local students, and then used MTurk workers for exploring the validity and generalizability of the findings (e.g. [26, 27, 36]). Others simply used MTurk workers as an additional population (e.g. [12, 34]). Some studies have focused solely on MTurk workers.

Most of the studies paid a fixed amount to each worker. Another approach was to employ workers' progress as a mechanism for assessing the success of an intervention as well as determining the size of the reward that workers received: Lee and Ko [20] compared personified and non-personified feedback by rewarding workers for each level that they completed in a novice programming environment; the researchers also used level completions as one measure of the efficacy of feedback.

3.4 Summary of Review

As our review demonstrates, crowdsourcing has been used in several ways in computing education research. However, compared to the overall volume of published research it is still relatively rare. The merits and drawbacks of crowdsourcing do not appear to be a major topic of discussion in the CER literature.

4 MTURK IN CER: A CASE STUDY

In this section, we describe experiences from a CER project, focusing on how we crowdsourced data from MTurk.

4.1 Background and Experiment Design

We started designing our experiment in early 2020. Our initial objective was to replicate a particular CER study [28]; we eventually ended up replicating another analysis [18] as well. In addition to the replication having value in itself, we had the secondary goal of learning about conducting CER via MTurk. The present article focuses on the latter goal and the methodological lessons we learned. The replications are discussed in more detail in [37, 38].

The structure of our experiment is shown in Figure 1. The participants first answered a demographic survey, then watched an instructional video on programming (randomly selected from three options), and finished by answering a post-test of program-tracing questions and a few additional survey items. A preceding screening survey was added later, as described in the next subsection.

The post-test included a "validity-indicator question" [8] (akin to a CAPTCHA) for assessing whether the worker had paid attention and was a human; this question asked the worker to think of an instrument and to write instrument's name and the name's first and last letter.

The experiment's overall duration was approximately 45 minutes, which includes 24 minutes of video. The content (i.e., survey, videos, post-test), were implemented on a website hosted at our institution.

Since the experiment was fairly long, we expected that some participants would skip parts and perhaps stop reading survey questions, answering at random. To identify such behavior, we added functionality for monitoring participants' behavior within our website. This allowed us to analyze whether the participants had answered the questions too fast, i.e., with such speed that they could not have read the questions that they were answering. This meant augmenting the survey, video, and the post-test with data collection facilities: in addition to the survey data itself, we collected metadata including the timestamps and targets of every click in the surveys. We programmatically removed the controls to pause or rewind the video; nevertheless, as a precaution, we logged and timestamped all video-watching events including pauses.

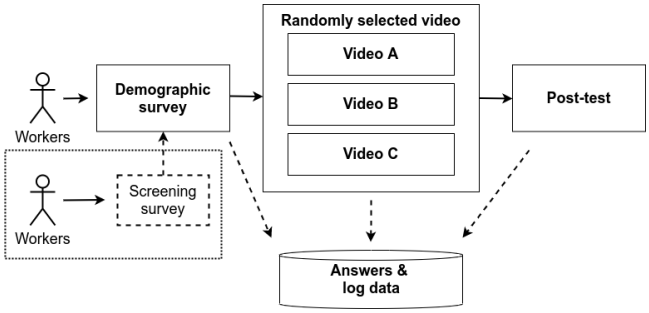

Figure 1: Experiment design. Participants answered a demographic survey, watched a video, and answered a post-test. During the study, a separate screening survey was added so as to invite only workers with no programming experience.

We had initially planned to collect responses from multiple sources: on-campus students, volunteers from certain online communities, and workers on MTurk. However, due to the COVID-19 pandemic, we did not approach on-campus students. In addition, when conducting pilot studies with unpaid volunteers from online communities, we found it difficult to gather substantial numbers of complete data sets given the experiment's relatively long duration. MTurk thus became the main focus for our study.

4.2 Data Collection on MTurk

As we explored MTurk and sought to identify appropriate options for recruiting beginner programmers, we used several tasks.

In the initial version of our MTurk task, we pruned out unreliable respondents and bots by requiring that most workers have at least 500 completed tasks with an acceptance rate of at least 99%.[1] To avoid (self-)selection bias, we did not disclose specific details of the experiment in the description, i.e., that it was about learning programming. The task had a reward of $7.5 per response, which amounts to approximately $10/h.

Since our objective was to find non-programmers, we chose not to use a pre-test of programming knowledge but asked participants to indicate their prior programming experience. Based on previous research using MTurk for computing education research, we expected that the majority of workers would have little to no programming experience. This expectation was proven wrong. Of the 123 workers who did our initial task, 61 had little to no programming experience (50%) and 47 had at least some experience (38%). The remaining workers (12%) were adjudged to have provided faulty answers or had other issues in their data, perhaps because of having disable JavaScript in their browser.

[1] During data collection, we adjusted these requirements somewhat as we learned about the platform and its use; the initial lowest requirement was 100 completed tasks with an acceptance rate of 98%

Since many of the initial workers had some programming experience, we needed to reach workers who had as little experience as possible, and MTurk did not offer previous programming experience as an requirement, we chose to create an additional screening survey (as suggested e.g. in [16]). The screening survey was a distinct, very short MTurk task. It was designed to identify potential participants for the primary task. Question topics in the screening survey included previous experiences in playing a musical instrument and computer programming.

We collected screening responses from approximately 1000 workers at $0.50 each. Of these workers, we invited some to participate in the experiment proper. More specifically, we invited everyone who had indicated that they have *no* programming experience, which was roughly a third of the screening-survey participants.

Finally, we created a new version of our primary task for those workers who had participated in the screening survey and had, in that survey, indicated no previous programming experience. This yielded data from 175 additional workers. When asked again at the beginning of the primary task, 145 of them (again) responded that they had little to no programming experience (83%), 18 (now) stated that they had at least some experience (10%), and the others provided faulty responses or had other issues with data (7%).

In the analyses reported in [37, 38], we focused only on those workers with little to no programming experience. That is, we ended up paying some workers whose data was of limited interest.[2]

4.3 Observations on Data Quality

Using metadata from the demographic survey, the video, and the post-test, a small proportion of participants could be identified as having provided faulty responses or having other issues in their data. This detection involved semi-automated analysis of log timestamps and manual analysis of the responses, including those entered to the validity-indicator question. Adding the screening survey decreased the proportion of faulty responses and responses with other issues (from 12% to 7%), and helped reach those with no previous programming experience (from 50% to 83%). We did notice, however, that a tenth of the workers had either gained some programming experience in the very short time span between the screening survey and the experiment or answered these two surveys inconsistently.

While previous studies have suggested that even one third of the data collected from MTurk may be faulty (e.g. [1]) and up to 60% of the responses to a validity-indicator question may be invalid (e.g. [8]), our observations are not quite as pessimistic. It is possible that this is in part due to our stricter requirements: compare our 99%-on-500-tasks requirement to the 85%-on-50-tasks discussed by Chmielewski and Kucker [8]. Amazon has also recently increased their efforts in detecting fraudulent activity on MTurk [2]. We went to some trouble to detect faulty responses and consider the proportion of faulty responses that we detected to be acceptable; further improvements to detection would still be welcome, of course.

We note anecdotally that many MTurk workers appear to have used a search engine when a question (after the instructional video) asked them to explain what a variable is in programming[3]. A large proportion of the responses were either direct or almost direct quotes from top search-engine hits. This illustrates that MTurk workers—like other people—search for answers in external sources.

5 DISCUSSION

In this article, we discussed how crowdsourcing platforms (mostly MTurk) have been used in computing education research, and described our experiences of collecting data from MTurk.

In our case study, we tried to reach workers with little to no previous programming experience. This proved to be more challenging than we expected but, in the end, appears to be achievable. We observed that many MTurk workers do have previous programming experience; since MTurk as a platform does not help in gauging that experience, it helps to create a separate screening task for identifying and inviting suitable workers.

Our case study on MTurk was successful in the sense that we did not discover major issues that can be clearly attributed to poor data quality, and so we have managed to conduct two meaningful analyses on our MTurk data set [37, 38]. These experiences, when combined with those from previous crowdsourced studies, suggest that platforms like MTurk are a plausible option for CER. In particular, MTurk provides access to cohorts broader than the typical student body at a university or school, which can be a boon to studies that seek to generalize from classroom and laboratory studies to larger populations. Crowdsourcing remains, however, an underinvestigated area with many potential issues.

Some 10% of the participants responded inconsistently to questions about programming experience in the screening survey vs. the demographic survey at the start of the experiment proper. This discrepancy suggests that data-quality issues exist but not necessarily at an alarming scale; further studies could investigate the reliability of answers to repeated questions on MTurk.

In addition to bots, cheating, and other data-quality concerns, crowdsourced studies on learning, such as ours, are inherently limited in that the study does not take place in a genuine educational context—formal or informal. This has implications for the learners' motivation and the generalizability of findings from MTurk to classroom contexts.

We outline the following suggestions for computing education researchers. They are in line with similar advice from other fields [16].

- Use MTurk requirements for requiring workers to have completed many tasks with a very high approval ratio. In our case, we (mostly) required that workers had completed 500 tasks with 99% approvals.
- When conducting research that requires that participants are novices, use a screening survey to identify a sub-population that matches the requirements. (A pre-test may also be appropriate, depending on the requirements.)
- When inviting workers to use an external system, track action timestamps to identify workers who may not have behaved honestly (e.g., responding to questions faster than it takes to read the questions).
- In surveys, include at least one "validity-indicator question" for sorting out nonsense and bot-generated data.
- When constructing assessment tasks, do not use questions for which responses can be easily found using a search engine.

[2] In the articles that report on the replication results [37, 38], we have marked 307 workers as the participant count. That differs slightly from the number of workers reported here due to the inclusion of nine participants from non-MTurk online communities.
[3] This additional question was not part of our analysis in [37, 38].

Our discussion so far has not considered the ethical aspects of crowdsourcing. A June 2018 article in *The Atlantic* pointed at a downside of crowdsourcing platforms, suggesting that they enable a work market where people are paid mere pennies [32]. This broader issue is entangled with the social security systems in the workers' countries (among other things), but we researchers must nevertheless ask the question of how to attend to this phenomenon when designing for crowdsourced research participation.

6 CONCLUSION

Our review of crowdsourcing in CER shows that this approach is being used by computing education researchers for a number of purposes. Crowdsourcing remains fairly uncommon, however, and the CER community has only begun to look into the opportunities that it presents, the caveats that are inherent to it, and the concerns that arise if studies are not designed with sufficient care. As crowdsourcing seems likely to increase in popularity in the future, CER as a field should be attentive to its strengths and weaknesses.

Together, the literature and the lessons from our case study suggest that there is potential in crowdsourced data. However, attracting a reliable crowd of learners that meet specific criteria—such as a particular level of programming experience—is a practical challenge and requires careful planning. Although we did not find evidence of widespread cheating or bot activity, more research is needed to explore these threats to validity. Moreover, while crowdsourcing provides access to a more diverse population than the typical classroom study, the transferability of crowdsourced findings to classroom contexts is a concern and requires further research.

REFERENCES

[1] Douglas J Ahler, Carolyn E Roush, and Gaurav Sood. 2019. The micro-task market for lemons: Data quality on Amazon's Mechanical Turk. In *Meeting of the Midwest Political Science Association*.

[2] Amazon Mechanical Turk Blog. 2020. Important updates on MTurk marketplace integrity, Worker identity and Requester tools to manage task quality. (2020). https://blog.mturk.com/

[3] Steve Buchheit, Derek W Dalton, Troy J Pollard, and Shane R Stinson. 2019. Crowdsourcing intelligent research participants: A student versus MTurk comparison. *Behavioral Research in Accounting* 31, 2 (2019), 93–106.

[4] Casey Casalnuovo, Prem Devanbu, and Emily Morgan. [n.d.]. Does Surprisal Predict Code Comprehension Difficulty? ([n. d.]).

[5] Carl Chapman, Peipei Wang, and Kathryn T Stolee. 2017. Exploring regular expression comprehension. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 405–416.

[6] Polina Charters, Michael J Lee, Amy J Ko, and Dastyni Loksa. 2014. Challenging stereotypes and changing attitudes: the effect of a brief programming encounter on adults' attitudes toward programming. In *Proceedings of the 45th ACM technical symposium on Computer science education*. 653–658.

[7] Yinlin Chen, Paul Logasa Bogen, Haowei Hsieh, Edward A Fox, and Lillian N Cassel. 2012. Categorization of computing education resources with utilization of crowdsourcing. In *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*. 121–124.

[8] Michael Chmielewski and Sarah C Kucker. 2020. An MTurk crisis? Shifts in data quality and the impact on study results. *Social Psychological and Personality Science* 11, 4 (2020), 464–473.

[9] Paul Denny, John Hamer, Andrew Luxton-Reilly, and Helen Purchase. 2008. PeerWise: students sharing their multiple choice questions. In *Proceedings of the fourth international workshop on computing education research*. 51–58.

[10] Paul Denny, Andrew Luxton-Reilly, and Beth Simon. 2009. Quality of student contributed questions using PeerWise. In *Proceedings of the Eleventh Australasian Conference on Computing Education-Volume 95*. 55–63.

[11] Paul Denny, Andrew Luxton-Reilly, Ewan Tempero, and Jacob Hendrickx. 2011. CodeWrite: supporting student-driven practice of Java. In *Proceedings of the 42nd ACM technical symposium on Computer science education*. 471–476.

[12] Madeline Endres, Georgios Sakkas, Benjamin Cosman, Ranjit Jhala, and Westley Weimer. 2019. InFix: automatically repairing novice program inputs. In *2019 34th IEEE/ACM Int. Conf. on Automated Software Engineering (ASE)*. IEEE, 399–410.

[13] Enrique Estellés-Arolas and Fernando González-Ladrón-De-Guevara. 2012. Towards an integrated crowdsourcing definition. *J. of Inf. Science* 38, 2 (2012).

[14] Mitchell Gordon and Philip J Guo. 2015. Codepourri: Creating visual coding tutorials using a volunteer crowd of learners. In *2015 IEEE symposium on visual languages and human-centric computing (VL/HCC)*. IEEE, 13–21.

[15] ME Hansen, A Lumsdaine, and RL Goldstone. 2013. An experiment on the cognitive complexity of code. In *Proceedings of the Thirty-Fifth Annual Conference of the Cognitive Science Society, Berlin, Germany*.

[16] David Hauser, Gabriele Paolacci, and Jesse J Chandler. 2018. Common concerns with MTurk as a participant pool: Evidence and solutions. (2018).

[17] Jeff Howe. 2006. The rise of crowdsourcing. *Wired magazine* 14, 6 (2006), 1–4.

[18] Dayu Jiang and Slava Kalyuga. 2020. Confirmatory factor analysis of cognitive load ratings supports a two-factor model. *Tutorials in Quantitative Methods for Psychology* 16 (2020), 216–225.

[19] Ryan Kennedy, Scott Clifford, Tyler Burleigh, Philip D Waggoner, Ryan Jewell, and Nicholas JG Winter. 2018. The shape of and solutions to the MTurk quality crisis. *Political Science Research and Methods* (2018), 1–16.

[20] Michael J Lee and Amy J Ko. 2011. Personifying programming tool feedback improves novice programmers' learning. In *Proceedings of the seventh international workshop on Computing education research*. 109–116.

[21] Michael J Lee and Amy J Ko. 2012. Investigating the role of purposeful goals on novices' engagement in a programming game. In *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 163–166.

[22] Michael J Lee and Amy J Ko. 2015. Comparing the effectiveness of online learning approaches on CS1 learning outcomes. In *Proceedings of the eleventh annual international conference on international computing education research*. 237–246.

[23] Michael J Lee, Amy J Ko, and Irwin Kwan. 2013. In-game assessments increase novice programmers' engagement and level completion speed. In *Proceedings of the ninth annual international ACM conference on International computing education research*. 153–160.

[24] Juho Leinonen, Nea Pirttinen, and Arto Hellas. 2020. Crowdsourcing Content Creation for SQL Practice. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. 349–355.

[25] Sarah KK Luger and Jeff Bowles. 2016. Comparative Methods and Analysis for Creating High-Quality Question Sets from Crowdsourced Data. In *The Twenty-Ninth International Flairs Conference*.

[26] Samiha Marwan, Joseph Jay Williams, and Thomas Price. 2019. An Evaluation of the Impact of Automated Programming Hints on Performance and Learning. In *Proceedings of the 2019 ACM Conf. on Int. Computing Education Research*. 61–70.

[27] Samiha Marwan, Nicholas Lytle, Joseph Jay Williams, and Thomas Price. 2019. The Impact of Adding Textual Explanations to Next-step Hints in a Novice Programming Environment. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. 520–526.

[28] Briana B Morrison. 2017. Dual modality code explanations for novices: Unexpected results. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*. 226–235.

[29] Matthew Mulvey. 2015. Effects of visualization on algorithm comprehension. (2015). MSc thesis.

[30] Nea Pirttinen, Vilma Kangas, Irene Nikkarinen, Henrik Nygren, Juho Leinonen, and Arto Hellas. 2018. Crowdsourcing programming assignments with CrowdSorcerer. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. 326–331.

[31] Nea Pirttinen, Vilma Kangas, Henrik Nygren, Juho Leinonen, and Arto Hellas. 2018. Analysis of Students' Peer Reviews to Crowdsourced Programming Assignments. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research*. 1–5.

[32] Alana Semuels. 2018. The internet is enabling a new kind of poorly paid hell. *The Atlantic* 23 (2018).

[33] Peeratham Techapalokul and Eli Tilevich. 2019. Code quality improvement for all: Automated refactoring for Scratch. In *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 117–125.

[34] Preston Tunnell Wilson, Ben Greenman, Justin Pombrio, and Shriram Krishnamurthi. 2018. The behavior of gradual types: a user study. *ACM SIGPLAN Notices* 53, 8 (2018), 1–12.

[35] Preston Tunnell Wilson, Justin Pombrio, and Shriram Krishnamurthi. 2017. Can we crowdsource language design?. In *Proc. of the 2017 ACM SIGPLAN Int. Symp. on New Ideas, New Paradigms, and Reflections on Programming and Software*. 1–17.

[36] Xu Wang, Srinivasa Teja Talluri, Carolyn Rose, and Kenneth Koedinger. 2019. UpGrade: Sourcing student open-ended solutions to create scalable learning opportunities. In *Proc. of the Sixth (2019) ACM Conf. on Learning@ Scale*. 1–10.

[37] Albina Zavgorodniaia, Rodrigo Duran, Arto Hellas, Otto Seppälä, and Juha Sorva. 2020. Measuring the cognitive load of learning to program: A replication study. In *United Kingdom & Ireland Computing Education Research conference (UKICER '20)*. ACM, 3–9.

[38] Albina Zavgorodniaia, Rodrigo Duran, Arto Hellas, Otto Seppälä, and Juha Sorva. 2020. Should explanations of program code use audio, text, or both? A replication study. In *Proceedings of the 20th Koli Calling International Conference on Computing Education Research (Koli Calling '20)*. ACM.