Weak Programmers Need Not Apply, LLMs Welcome! Survey Screening in the AI Era

Ita Ryan
School of Computer Science and Information Technology
University College Cork
Cork, Ireland
iryan@ucc.ie

Utz Roedig
School of Computer Science and Information Technology
University College Cork
Cork, Ireland
u.roedig@ucc.ie

Klaas-Jan Stol
Lero and School of Computer Science and Information Technology
University College Cork
Cork, Ireland
k.stol@ucc.ie

Abstract

Quantitative research on software development practice often depends on the results of anonymous online surveys. However, it is difficult to verify that the respondents to such surveys are genuine developers, especially if survey responses are paid for, incentivising respondents to game the system and attracting the attention of bots and professional survey farms. One solution is to include screening questions to assess respondents' coding skills, but this introduces the risk of unfairly excluding considered responses from developers who spent time and energy on the survey. We reviewed the responses of 86 developers who were excluded via screening from analysis of a large unpaid developer survey (n=1,048). We estimated that up to 86% of the developers who were excluded were genuine developers. The advent of large language models (LLMs) casts further doubt on the use of screening questions. Investigating LLM ability, we found that five sample LLMs could answer most widely-used screening questions. Powerful LLM-based survey-taking tools now exist. We researched current survey screening techniques and found that they are susceptible to fraud via LLM use, bots and survey farms. Recruitment strategy may be a better screening technique. We recommend that if survey respondents are compensated only known developers should be invited. Surveys that are widely distributed, e.g. on social media, should not compensate respondents. Instead, researchers should focus on good survey design and sustained, imaginative recruitment strategies.

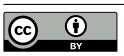
CCS Concepts

• **Social and professional topics → Professional topics**.

Keywords

Survey research, survey screening, software developers

ACM Reference Format:
Ita Ryan, Utz Roedig, and Klaas-Jan Stol. 2026. Weak Programmers Need Not Apply, LLMs Welcome! Survey Screening in the AI Era. In *2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE '26), April 12–18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3744916.3787789



ICSE '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2025-3/2026/04
https://doi.org/10.1145/3744916.3787789

1 Introduction

In previous work, we ran an anonymous online developer survey over the three months to January 31st 2022 as part of an investigation into secure coding practice and culture [36, 38, 40, 41]. The survey data is available on a file sharing site [43]. Recruitment for the Ryan survey was very successful, attracting 1,100 unpaid participants of whom 1,048 were regular coders and thus eligible for inclusion in the study. Following recommendations in the literature [9], we included two code screening questions that were designed to ensure that respondents were, in fact, genuine software developers. Two attention screening questions were also included. The intention, followed during screening, was to exclude respondents who got any of these four screening questions wrong.

However, observation of the data suggested that some excluded responses were valuable. Answering question 25 on who to contact in the event of a security flaw, excluded respondent P703 entered in free text: *'Other developers in a chat room of the company,'* a singular insight into company dynamics. Respondent P93 supplied a thoughtful free-text response when asked if they had heard of OWASP: *'Yes, I am aware of who they are and what they publish.'* Asked about training they had received, they entered *'Internal and contracted cloud security & secure coding practices.'* On legal obligations they again supplied a unique entry: *'(cannot disclose).'* They got both attention screening questions and some later security questions right, but failed on the second code screening question.

These two respondents were excluded from data analysis because they answered the second code screening question incorrectly and we followed the predetermined screening protocol to avoid bias. However, a common-sense view, based on their full contributions, was that they were probably genuine programmers. It was possible that there could be others like them amongst the 86 respondents who were screened out due to failing the code or attention screening questions (see Table 1 for question details). Indeed, the study from which the code questions were drawn found a 6% failure-rate for genuine programmers answering the second code-screening question [9]. The questions used C-like pseudocode that a SQL or Python developer might never encounter professionally. It was apparent in the Ryan data that some SQL and Python developers had got them wrong. We received the following message from a Python expert who was asked to distribute the security survey to their contacts: '*The code test at the beginning my stump some of my Python followers :-)*'. Not all programmers are familiar with C-like programming languages [53], or are good programmers, or will carefully check screening questions at the beginning of a survey. Arguably, excluding less-skilled or hasty programmers from the study

could bias study findings. The attention questions, meanwhile, were simply not understood by at least one attentive survey respondent, as documented in Ryan et al. [36] *'Ididint understand q56. Attention? No idea what you are wanting from this poor question.'*

We were in the fortunate position of having a very high number of responses. We could afford to 'lose' 86 responses from the analysis and still produce significant results, as can be seen from the papers on the study [36, 38, 40, 41]. However, losing up to 8% of survey respondents who could arguably be retained has the potential to reduce the comprehensiveness of survey results. This is particularly true for the 51 participants of this survey who failed the second code screening question, as they could have certain coding characteristics in common, including coding language, level of experience, or coding weakness. There is also the ethical question of whether it is fair to discard a thoughtful, considered survey response merely because the respondent's programming abilities are weak or they are unfamiliar with C-style programming idioms.

Furthermore, the screening questions might no longer achieve their intended purpose. Since they were tested [9], large language models (LLMs) have made coding puzzle solutions accessible to non-programmers [58]. In addition, there have been numerous documented incidents of bots [60], survey farms [19], and even survey-fraud-as-a-service actors [29] targeting surveys in order to earn compensation. This led us to ask three research questions:

RQ1: Do screening and attention questions run the risk of inadvertently excluding programmers unnecessarily?

RQ2: Can LLMs answer programmer screening questions?

RQ3: What screening techniques work best to check programmer ability and prevent survey fraud?

Section 2 discusses related work. We addressed RQ1 in Section 3. We examined the excluded Ryan survey participants in detail to see how many of them could plausibly have been retained for analysis. We explored whether screening and attention questions may skew survey results by removing weak or hasty programmers.

We considered RQ2 in Section 4. We submitted six widely-used screening questions [9] to five different LLMs. Five were answered correctly, and the sixth had three correct answers. We also tested a short node.js screening program from Reid et al. [33]. All LLMs got the output wrong, but online sites to run it are trivially discoverable.

We addressed RQ3 in Section 5. We reviewed relevant papers to find survey screening techniques. Reviewing the use of code screening questions in the LLM era, we considered other techniques to ensure that survey responses are genuine. We found that all current techniques are vulnerable to fraud and survey farms. We examined the effect of recruitment strategy, and advocate that, to deter bots and survey farms, either surveys should be highly targeted to known respondents or respondents should not be compensated.

Finally, Section 6 presents the limitations and threats to validity of the work. We discuss our finding that removing financial incentives is the best way to deal with survey fraud. This is followed by recruitment suggestions for unpaid surveys, and our conclusion.

2 Background and Related Work

2.1 Recruiting Participants for Field Studies

Rainer and Wohlin proposed a framework for defining participant relevance and ability to contribute in software engineering (SE) studies [32]. It does not address the unique characteristic of surveys that distinguishes them from other study types such as field studies; that the researcher usually does not encounter the respondent in person [51]. While field studies take place in 'natural' settings (i.e., the field where software development activities take place), surveys take place in 'neutral' settings; that is, survey studies could be done in any type of environment, as long as it is not distracting.

Patnaik et al. proposed creation of a database of developers known to be research-friendly [30]. Tahaei and Vaniea suggested that industry and open source should get involved in encouraging study participation, as they benefit from good research [52]. However, questions of bias could arise if all SE research depended on the same pool of respondents. Computer science (CS) students are an option but may be inexperienced [42]. Researchers often scrape GitHub for developer email addresses, but GitHub's acceptable use policies now prohibit sending unsolicited emails [17]. Perhaps this is because, as long ago as 2016, developers on GitHub reported receiving one or more research invitation emails a week, described as '*worse than spam*' [6]. Serafini et al. interviewed 30 company developers and found that adequate monetary compensation, good design, clear inclusion criteria communication and referrals from friends and colleagues all motivated study participation [46].

2.2 Participant Recruitment for Surveys

Developer surveys are often used to ascertain the characteristics of a developer population, or test a research hypothesis. Some recruiters recommend limiting distribution of survey invitations, because a broad invitation prevents researchers from keeping tight control of who is responding [54]. This strategy can introduce bias [23], and is in tension with the desire to obtain a large number of responses to the survey. Ghazi et al. [16] reviewed the literature and surveyed nine experts to find problems and solutions in SE research. Based on the interviews, they derived several guidelines for encouraging valid responses. They stated that none of their interview subjects believe researchers who claim to have done random or stratified sampling, because this is essentially impossible in SE research [4]. In most cases the target population cannot be defined, and even if random sampling were done the participants would be unmotivated. The experts recommended using convenience sampling, including with social media, and striving for heterogeneity, using demographic questions to characterise the sample.

Kokinda et al. discussed the importance of approaching participants in a respectful way, describing challenges in attracting research participants from sources such as Reddit, Twitter, and Discord [22]. Contrary to our experience, some survey participants swore, purposely added gibberish, and expressed disapproval or hatred of the researchers, or their institution. There were culture war-style entries, for example around responses to gender choices. Drumming up responses is difficult, and therefore researchers often pay organisations or platforms to provide survey respondents, or they pay respondents directly [5, 20]. Payment may lead to issues of respondent quality [3, 10, 11, 24, 33]. Payment levels for survey responses are usually low [3, 33], and those earning a professional developer's salary are unlikely to be motivated by them. Many platforms providing compensated participants do not screen the participants to verify their stated coding ability [3, 11, 33, 53].

2.2.1 Paid surveys. Some researchers do not discuss attempts (if any) to validate that paid survey respondents are genuine developers [14, 23, 56, 57]. Those who do validate often encounter problems. Kaur et al. compared six sources of respondents: four platforms offering paid respondents, and Google Play developers and CS students recruited by email and offered access to a gift card raffle. They found that self-reporting of skills and experience sometimes appears to be exaggerated and some sources are subject to fraud [20]. Ebert et al. tried to do a 22-person survey but found that none of the candidates sourced from a participant recruitment platform had the necessary knowledge to take part [11]. Their recommendations included screening developers and validating answers.

Danilova et al. used a professional recruitment service [10]. They found that 96 out of 129 respondents failed to pass programming tests. In follow-up work, Danilova et al. assessed a number of code screening questions for developer surveys, and advocated six [9]. These questions have been widely used by the SE community to validate participants in online surveys [11, 33, 35, 53]. Indeed, our Ryan survey used two of them. Reid et al. used all six to vet 260 pre-screened participants from a recruitment platform [33]. Having stated that they were node.js programmers, participants were asked an additional node.js question. Only 55 passed all seven questions. The researchers recommended running pre-screening and code screening questions when using paid recruitment platforms. Several of the Danilova questions can be time-boxed for participant screening, which is recommended by Russo [35].

Alami et al. discussed approaches to screening when recruiting from a commercial platform [3], observing that the Danilova questions may not work in the AI era. To screen survey responses for two different studies they used a gradual approach, only accepting 50 entries a day and asking several screening questions including an open-ended question. The valid response rate for both surveys was less than half of initial participants. Their screening recommendations included iterative assessment, targeted pre-screening, AI detection and qualitative evaluation of open-ended questions. Interestingly, in a third paid study they screened for *interview* participants with similar techniques, but added relatively difficult challenges such as writing short programs and describing work challenges. They told applicants that they would use AI detection to screen for AI responses. The percentage of respondents they considered valid in this study was much higher at 99.5%. Twelve respondents were then interviewed; they attended as scheduled and '*showed a high level of professionalism and experience*.' However, such difficult challenges are not appropriate for survey screening, since they are disproportionate to the expected effort, and payment if any [3, 9].

Paying directly-recruited developers can also attract fraud. Studying developer tool adoption, Witschey et al. offered the chance of a $15 Amazon gift card to the first 50 respondents from five tool mailing lists [59]. Their survey was completed in under five minutes by 313 respondents. When they had removed these responses, which they suspected had been automated, they were left with only 61 usable responses. In contrast, Liang et al. [23] added participants to a draw for four $100 gift certificates, recommended to boost participation [13], when studying developers' opinions on the use of AI coding assistants [23]. They emailed a survey link to 10,530 GitHub participants who had forked or starred related projects. They got 410 responses, approximately 4%. No validation was described. Open text responses, which often give the first hint of fraud, were analysed but no fraud was reported, possibly due to the targeted recruitment strategy.

2.2.2 Unpaid surveys. We observe that unpaid surveys do not seem to attract fraudulent responses. Escobar-Avila et al. studied CS online learning preferences via survey [14]. They publicised the survey in a Facebook CS student group and on Twitter. They received 205 responses, one of which indicated that the respondent was not a coder, giving a final sample of 204. Endres et al. studied cannabis use during software development [12]. They emailed GitHub contributors and University of Michigan students, and posted the survey on X. Having discarded 236 respondents who did not answer three core questions about programming and cannabis use, they did data validation on the remaining 809. Six participants were removed due to unusually fast responses or issues such as inconsistent age and years of programming experience, resulting in 803 respondents. Espinha et al. ran a survey which was open for 7 months and obtained 196 answers of which 2 were rejected due to irregularities in the data, leaving 194 [15]. This was a failure rate of only 1%.

2.3 Bots, LLMs, and Survey Farms

SE researchers have found evidence of bots and LLM use in survey responses [24]. We did not find discussion of '*survey farms*' in the SE literature, but the term is used in other research domains [58]. Kaylor-Tapscott et al. defined survey farms as *'locations where surveys are rapidly completed using fake IP addresses'* [21]. Guy et al. discussed survey farming that was apparent in two paid online survey studies of vulnerable populations [19]. In the first study, 1,433 screened participants were whittled down to 325 after eligibility checks. In the second, 2,059 became 142. Guy et al. noted that farming can be done by programmed bots of varying levels of sophistication, or by ineligible individuals who are not part of the target population but take a survey anyway for financial reasons, trying to mimic genuine respondents. Such individuals tend to report stereotypical behaviour in responses. Participants may take large numbers of surveys to earn money [25], and eligible respondents may try to make more money by taking a survey repeatedly.

Griffin et al. encountered a large number of bots when running a survey related to Covid-19 [18]. They had offered a $5 gift card for survey completion, accessed by completing a second survey. They paused recruitment after a day due to a disproportionately high number of respondents to the second survey. Investigations revealed 773 (61.8%) fraudulent responses, sophisticated enough to evade platform protections such as cookies and referrer link checks. They changed incentive structure, offering entry into a gift card raffle instead of guaranteed payment, without specifying the gift card amount. They also added honeypots (see section 5). They vetted based on time duration, removing responses submitted over unusually short or long periods. (The latter is not recommended [16].) Fraud rates of approximately 30% in subsequent rounds suggest that the raffle incentivised fraud as well as completion.

Another indicator that payment for survey responses attracts fraud comes from Ruby et al.'s online survey on gestational diabetes [34]. Between March 9 and October 13 2023 there were 73 responses, with 54 (74%) considered eligible. Aiming to obtain at least 200 responses, Ruby et al. introduced changes on October

11 2023, including '*a US $5 gift card incentive to increase survey engagement.*' Another 2,440 responses arrived between October 14 and 16. Observing data inconsistencies, the researchers closed the survey and introduced a detailed process for identifying fraudulent responses. It resulted in the discarding of all 2,440 paid responses.

Although the term is not yet in use in our discipline, SE researchers are no strangers to survey farming. Meem et al. studied automated program repair tools via an online survey [24], offering the chance to win a voucher. They were inundated with bots, and also found evidence of LLM-generated responses to open-ended questions. Discarding replies that appeared to be from bots, open-ended responses apparently generated by an LLM, and attention check failures, their 800 responses were reduced to 331, or 41%.

Recently, Westwood wrote an LLM-based tool '*capable of completing entire surveys with human-like responses*,' which solves puzzles and can emulate different demographics and political beliefs [58]. Westwood argues that '*foreign states...can easily use [such tools]...to bias public opinion measures.*' The LLM issue is pervasive.

3 Analysis of Excluded Respondents to the Ryan Survey (RQ1)

This section examines respondents excluded from analysis of the Ryan survey, described in previous work [36]. This survey is ideally suited to this analysis because there was a large number of respondents, there were several screening questions, and there were numerous open-ended questions which allowed respondents to enter free text. The result is a large trove of data [43] that allows the researcher to analyse respondents who failed screening by looking at all of their responses in a holistic way.

The Ryan survey was a large-scale survey of software developers to examine issues of secure-coding practice and culture [36]. It included five screening questions (see Table 1). The mandatory pre-screening Elimination Question asked respondents whether they coded regularly. If the answer was no, they were ineligible and could not complete the survey. The two mandatory code screening questions (CQ1 and CQ2 in Table 1) were intended to screen out non-developers. They were selected from a set of six evaluated and advocated by Danilova et al. [9]. They were slightly disguised by changing the input phrase from *'hello world'* to *'lorem ipsum.'* Fig. 1 shows the pseudocode used for these questions. Two *attention screening* questions were added to assess whether respondents were focused on their responses. One (AttQ1 in Figure 2) appeared early in the survey, the other near the end. They were designed to blend in with surrounding questions, so that anyone clicking blindly would not notice them.

In total, 138 participants failed the screening questions. The 52 participants who answered 'No' to the pre-screening Elimination Question at the beginning of the survey could not complete it. This left 1,048 potential study participants, of whom a further 86 failed one or more of the other four screening questions.

In the paper from which the two coding screening questions were sourced, Danilova et al. aimed to find questions that were answered correctly by programmers and incorrectly by non-programmers [9]. CQ1 (Danilova Q15) was answered correctly by 100% of programmers and 13% of non-programmers (See Table III in Danilova et al.). CQ2 (Danilova Q16) was answered correctly by 94% of programmers and 7% of non-programmers. If those statistics held in the Ryan survey's respondent population, and all 1,048 were genuine programmers, we would expect 63 programmers to get CQ2 wrong. In fact only 51 did. Table 2 shows that question failure rates in the Ryan survey for the two Danilova questions are in line with the failure rates of genuine programmers in Danilova's study, suggesting that excluded Ryan respondents were genuine programmers.

The survey was approved by an ethics review board, as reported in previous work [36].

Method for RQ1. In order to assess whether any of the 86 excluded participants could have been included we reviewed their answers. There is no standard procedure or tool for detection of fraudulent responses. It is usually done by researchers using the characteristics of the specific survey [34]. In this case, three investigators undertook a two-phased review process. For each participant, we examined the characteristics of their answers to all questions. Relevant aspects of their answers that we identified included:

- Their answers for each of the four screening questions;
- Their answers for two non-mandatory security questions;
- The development languages that they stated they use. Unfamiliarity with C-style languages could negatively affect respondents' ability to answer the coding questions. See 'not C-style developer' in Table 3.
- Their country. There is a higher possibility that non native English speakers could misunderstand a question.
- Whether they answered all, or most, of the survey questions, indicating a real effort. Skipping was permitted, which caused concern on automatic exclusion due to missed attention questions. See 'skipping not a fail' in Table 3.
- Whether they entered free-text responses, and whether those responses seemed convincing. Entering free-text responses, again, indicates that respondents made a real effort, going

```
main {
  print(func("lorem ipsum"))
}

String func(String s_in) {
        int x = len(s_in)
        String out = ""
        for (int i=x; i>=0; i--){
                out.append(s_in[i])
        }
        return out
}
```

Figure 1: Pseudocode used for the two code screening questions (CQ1 and CQ2), based on Danilova et al. [9].

This is an attention question. Please select a little.
- ◯ A great deal
- ◯ A lot
- ◯ A moderate amount
- ◯ A little
- ◯ None at all

Figure 2: The first attention screening question (AttQ1)

Table 1: Screening questions. Asterisked* questions were mandatory. After the Elimination Question 1048 respondents remained. 86 then failed one or more of the coding or attention questions and were removed from analysis. Some participants failed more than one question; totals sum to more than 86.

Question ID	Question	N failed
Elimination Question*	Do you write computer code frequently, either professionally or open source?	52
Coding Question 1 (CQ1)*	What is the parameter name of the method called 'func' in this pseudocode?	9
Coding Question 2 (CQ2)*	Please select the value that the 'func' method in the last question would return if run from main as above	51
Attention Question 1 (AttQ1)	This is an attention question. Please select a little.	15
Attention Question 2 (AttQ2)	Another attention question, please answer definitely not	34

Table 2: Comparing code screening failure percentages of Ryan et al. and Danilova et al. [9].

	Ryan et al.		Danilova et al.	
Q	N. failed	% failed	Fail % (prog.)	Fail % (non-prog.)
CQ1	9	0.9%	0%	87%
CQ2	51	4.9%	6%	93%

beyond the minimal effort of selecting answers in multiple-choice questions. See 'free text' in Table 3.

- Whether they answered '*I don't know*' too frequently, making their responses unusable. See 'I don't knows' in Table 3.

In the initial review phase, three investigators independently examined the 86 excluded participant records. Each researcher made a recommendation for each participant, having studied their answers to all questions in the survey. Recommendations were either 'Include,' 'Reject,' or 'Borderline.' Decision-making information was not shared until all three researchers had completed the work.

After the initial phase, seven cases were subject to a three-way tie. The three investigators all disagreed in their verdict for each one of them, with one 'Reject,' one 'Borderline,' and one 'Include' for each. The second review phase involved a joint discussion of each of these seven participants among the three investigators. Table 3 presents details of these seven participants.

Results for RQ1. The results of the initial review phase are shown on the left-hand side of Figure 3. The three investigators all decided on 'Include' for 39 participants, 45% of the excluded total. A majority of two of the three researchers wished to 'Include' an additional 32 participants, or 37% of the total. A majority agreed that eight participants, 9.3% of the excluded total, should be rejected. There were no consensus 'Borderline' or 'Reject' recommendations. Seven responses were subject to a three-way-tie and progressed to the second review phase.

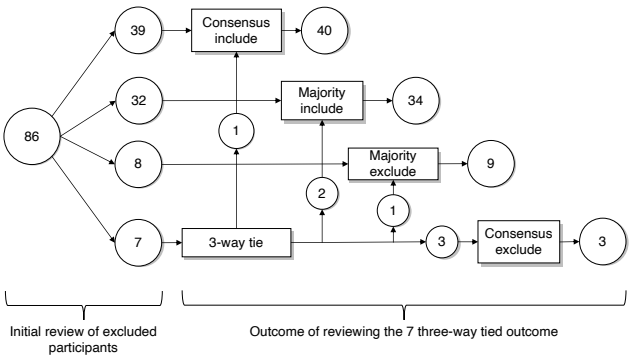


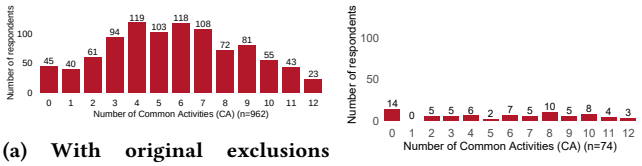
Figure 3: Outcome of the review process.

The second phase led to the inclusion of a further three (one by consensus, two by majority vote), and exclusion of the remaining four (three by consensus, one by majority vote). A spreadsheet of the analysis is available in supplementary material [39].

Based on our analysis, many or most of the 86 excluded participants could have been included. There was *consensus* among the three investigators that 40 of 86 excluded respondents (46.5%) could be retained. If we were to also retain respondents judged valid by a *majority* (two out of three), then 74 (86%) of respondents who were originally rejected would be retained. Exclusion is upheld for 12 (three by consensus, nine by majority), or 14% of those originally excluded. This would give a screening failure rate for the whole survey (based on the 1,048 responses) of 1.15%. Since this is around one percent, it could be argued that even retaining these 12 responses

Table 3: Post-discussion decisions on the seven excluded participants that all three researchers initially differed on.

ID	Failed	Doubts	Result	Reasons	Votes
69	AttQ2	AttQ2	Include	Free text, skipping not a fail	3/3
274	AttQ2	AttQ2, no replies after Q41	Reject	Early exit	3/3
321	AttQ1, AttQ2	Attention questions	Reject	Attention issue, probably English native	3/3
486	AttQ1, AttQ2	Attention questions, no replies Q27-Q61	Reject	Early exit	3/3
636	CQ1, CQ2	Coding questions, I don't knows	Include	Not C-style developer	2/3
881	CQ2, Att1, Att2	Attention questions, CQ2, I don't knows	Include	Not C-style developer, good free text	2/3
1047	CQ1, CQ2	Coding questions, I don't knows	Reject	I don't knows	2/3



(a) With original exclusions (n=962)

(b) Excluded sample (n=74)

Figure 4: (Ryan et al. Fig 2): Common Activities in participants' coding environments. *Left:* original Ryan et al. figure. *Right:* The 74 excluded then deemed valid.

would not have an adverse statistical effect on survey results: they would be absorbed by survey noise.

We wished to observe the effect of retaining excluded respondents on Ryan et al.'s conclusions, which were based on the plots in the paper [36]. We regenerated all plots but only excluded the minimum number of respondents (12), giving n=1,036. We included the 74 that were judged valid either by majority or by consensus. Appendix 1 of the online supplementary material shows all the Ryan et al. and new plots side by side [39]. We reproduce here in Fig. 4 two plots of common coding activities. The first is Fig. 2 in Ryan et al. (n=962), showing a normal distribution. The second shows coding activities for the excluded respondents that we now wish to include (n=74). We see that the distribution of the excluded sample is not normal. Pearson's Chi-squared test, applied to these two samples n=962 and n=74, gives a test statistic of 42.51 with $p < 0.001$, indicating that the two distributions differ significantly. However, a statistical comparison of the two samples n=962 and n=1036 using Pearson's Chi-squared gives a test statistic of 1.66, with a p-value of 0.1, indicating that the distribution in Ryan et al. and the new distribution with 74 excluded respondents added do not differ significantly.

Changes to the plots in Ryan et al. [36] caused by including the 74 additional respondents are minimal. For example, in Ryan et al.'s Figure 6, there is a decline of 1% in those who disagree or strongly disagree with the statement '*Would you agree that there is a security culture in your working environment?*' These changes do not affect our conclusions. This is further evidence that the excluded respondents were probably genuine developers, excluded unnecessarily. The very large Ryan survey sample allowed for useful conclusions without these excluded responses. However, most surveys in SE research are based on much smaller samples and could be adversely affected by unnecessary exclusion of respondents.

There appeared to be no or very little fraud in the survey results. We believe that this was because respondents were not financially compensated. This suggests that, if fraud were not a concern, code screening would be superfluous in a survey where good coding ability is not required. We were interested in the *secure coding environment* of *all* developers responding to the survey. We used the code screening questions to exclude non-programmers, *not* to exclude poor programmers. In fact, the secure coding environment of less able programmers could be particularly interesting, because those programmers need more secure coding support. Therefore the answer to RQ1 is yes, using tests of programming ability to screen for real coders *does* exclude some weaker coders and those using languages that differ significantly from the pseudocode (e.g. SQL vs. C-like code). Such tests should not be used where coders with a range of abilities, strengths, and languages are required.

Other survey studies may require that only competent or expert programmers are included. For such studies, using Danilova-style screening questions could be seen as a way to guarantee expertise. However, in this LLM era, the Danilova questions are vulnerable to fraud, as we discuss in Section 4.

Answer to RQ1: Using tests of programming ability to screen for genuine programmers excludes some weaker coders, and some coding in languages different to those used in coding questions.

4 Using LLMs to Pass Screening (RQ2)

Naiakshina et al. warned in 2018 against using survey questions for participant screening [27]. Their study took place in a laboratory setting where they observed that participants used the Google search engine to answer difficult questions; this cannot be monitored in surveys conducted online. As discussed, in 2021 Danilova et al. [9] tested screening questions that were *'designed to take as little time as possible so they can be used in free or cheap pre-screening surveys. They needed to be so easy as to not annoy or challenge people with programming skill so we do not falsely reject participants, but also hard enough so that people without programming skill cannot make an educated guess or google the answer quickly.'*

In this era of AI, LLMs could potentially be used to answer screening questions. They are easier to interrogate than Google and give more accurate results for this type of question. Assessing the barriers posed by the two Danilova [9] code screening questions in our survey, we decided to investigate the ability of current LLMs to answer the questions correctly.

Method for RQ2. We sought to simulate the likely actions of a survey respondent with no technical expertise submitting the questions to an LLM. We posited that they would choose a popular free LLM. We therefore did a Google search for '*best free LLM*' on July 8 2025. We tested the questions (see Table 1) with five free LLMs as provided by Google. We did not create accounts with the LLMs unless it was mandatory. To emulate the non-expert, we did not use prompt engineering. When asking CQ1, we copied and pasted the pseudocode and then the question into the LLM. When asking CQ2, we asked the LLM: '*Look at this pseudocode. What would it output?*', and then pasted in the pseudocode. We then investigated further, by checking the other Danilova questions and a short node.js program from Reid et al. [33]. Again, we pasted each question directly into the LLMs with no prompt engineering. The online supplementary material shows screenshots of all LLM queries [39].

Results for RQ2. Table 4 shows the results for CQ1 and CQ2. All five LLMs answered the first question correctly. The second question was a little more difficult, particularly for over-eager Pi which produced an answer of '*1, 2, 3*' after we submitted the question but before we pasted in the pseudocode. It then failed the actual question, though its answer, 'muspi murel' was wrong in an unexpected way. Also unexpected, Claude approached the question correctly but simply ignored the 'ip' in 'ipsum' and output 'mus murel.' Overall, three LLMs got CQ2 right, and the two wrong answers were sufficiently close to allow a user to guess correctly in

the survey. We conclude that a non-programmer survey respondent could use an LLM to solve these programming tasks.

To evaluate other screening questions that are in general use [11, 33], we submitted the other four questions from Danilova et al. [9] to the LLMs. See Table 5. All LLMs answered all four questions correctly. We then submitted a short node.js program used for screening by Reid et al. [33]. All five LLMs got the answer wrong. However, we were trivially able to obtain the correct answer by googling '*run node.js code online*' pasting the program into the top result site (onecompiler.com), and clicking 'Run'. We verified the result at top-four sites jdoodle.com and codeutility.io. This simple approach could be adopted by a non-programmer.

Alami et al. sought to deter use of LLMs for open-ended answers, warning respondents that AI would be used to detect LLM-generated text and code [3]. The warning might not be much of a deterrent. In a proof-of-concept experiment, we found that it was not difficult to fool AI into thinking that a piece of LLM-generated text was written by a human. Figure 5 shows how Claude's output can be made human-looking enough to dupe ChatGPT, simply by requesting Claude to make it 'super casual and human-sounding'. This experiment emulates a novice fraudster. More sophisticated fraudsters, bots, and survey farms can use tools like Humanize AI [1] and Undetectable AI Rewriter [2] for '*humanization*' [50]. Westwood has devised an LLM-based survey-response tool [58].

Answer to RQ2: Yes, most programmer survey screening questions can be answered correctly by LLMs. Results from short programs are the exception, but can easily be obtained online.

5 Evaluating Respondent Ability and Preventing Survey Fraud (RQ3)

Anonymous online surveys may attract fraudulent responses from individuals, bots and survey farms. Fraudulent individuals may face challenges such as lack of relevant knowledge. Sophisticated bots and farms, by contrast, can share correct answers between responses, and submit multiple responses quickly and cheaply to maximise returns [18, 19]. In this section, based on a review of recent survey literature, we describe the survey screening methods devised to deter fraud. We consider their impact on genuine survey respondents, and their efficacy against fraudulent responses.

Method for RQ3. To find instances of survey fraud, and survey screening techniques, we did an extensive Google Scholar search in the broader literature, with backward and forward searching for relevant work in the SE domain. SE papers were selected if they discussed survey screening or survey fraud. Some survey studies might not screen respondents or overtly address fraud; therefore we also included survey papers from ICSE 2025 to provide balance.

Results for RQ3. Table 7 contains the objective, numbers surveyed, screening techniques if any, and relevant observations or recommendations from review papers. Table 6 presents the screening techniques most frequently encountered in the literature. Assessments of the barrier each technique provides to participation by genuine respondents, to use of LLMs, and to bots and farms, are based on the literature. Each screening technique from the table

Table 4: LLMs correctly answered Ryan et al.'s screening questions, except Claude and Pi who failed at CQ2 (July 8 2025).

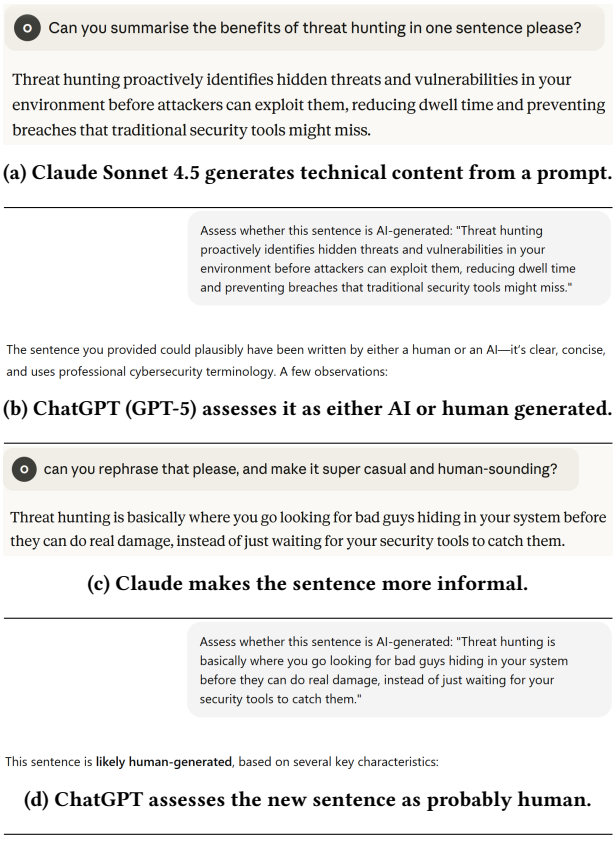
LLM	Version	CQ1	CQ2
ChatGPT	GPT-4	s_in	muspi merol
Claude	3.5 Sonnet	s_in	mus merol
Deepseek	R1	s_in	muspi merol
Gemini	2.5 Flash	s_in	muspi merol
Pi	Inflection 2.5	s_in	muspi murel

The correct answers are (CQ1) 's_in' and (CQ2) 'muspi merol.'

Table 5: LLM answers to screening questions DQ2-DQ4 and DQ6 from Danilova et al. [9], RK7 from Reid et al. [33].

		Danilova et al.				Reid et al.
LLM	Version	DQ2	DQ3	DQ4	DQ6	RK7
ChatGPT	GPT-5	✓	✓	✓	✓	0
Claude	Sonnet 4.5	✓	✓	✓	✓	190137
Deepseek	V3.2	✓	✓	✓	✓	2724520
Gemini	2.5 Pro	✓	✓	✓	✓	2727747
Pi	Inflection 2.5	✓	✓	✓	✓	2738

2727500 is the correct answer to RK7. Tests run on Nov 13 2025.



(a) Claude Sonnet 4.5 generates technical content from a prompt.

(b) ChatGPT (GPT-5) assesses it as either AI or human generated.

(c) Claude makes the sentence more informal.

(d) ChatGPT assesses the new sentence as probably human.

Figure 5: Demonstration showing how LLMs can be used to generate text that fools LLMs (5 Nov 2025).

Table 6: Screening techniques that can be used to screen out fraudulent or undesirable survey responses.

Purpose	Approach	Practical for respondents?	Human fraud proof?	LLM-proof?	Bot and farming-proof?
Assess technical skill	Pre-screen: Ask respondent	Yes	No	No	No
	Technical questions (TQs) [10, 11, 33]	Yes	Yes	No	No
	Time-boxed TQs [9, 35]	Yes	Yes	Helpful	No
	Writing code snippets [3]	No	Yes	No	No
	Video interviews [3]	No	Yes	Yes	Yes
Check attention	Ask respondent [45]	Yes	No	No	Helpful
	Ask the same thing twice [46]	Yes	Helpful	NA	Helpful
	Request specific response [7, 26, 33]	Yes	Helpful	NA	Helpful
Assess data quality	Time to complete answer [12, 19, 24]	Yes	Helpful	NA	Helpful
	Free-text open-ended questions [3, 24, 34]	Yes	Helpful	Helpful	Helpful
	Look for inconsistencies [12, 24, 28, 33]	Yes	Helpful	NA	Helpful
Detect bots and survey farms	Honeypots [18, 47]	Yes	NA	NA	Helpful
	Delayed compensation [19]	No	Helpful	Helpful	Helpful

appears in bold in the following text for quick reference. Survey platforms may provide screening techniques such as CAPTCHA and IP address checks [34]. These are not described here. They are not foolproof but are helpful.

Assessing Technical Skill. Numerous studies suggest pre-screening by **asking the respondent** whether they have a required skill. For example, Ryan et al. asked whether the respondent codes frequently [36]. Disadvantages of this approach are that programmers may overestimate their own skills [35, 37], and that it provides no protection against fraud. Its advantage is that it quickly confirms whether honest respondents are appropriate for the survey.

Technical questions such as those devised by Danilova et al. have already been discussed. The Ryan survey used them; see CQ1 and CQ2 in Table 1. They are vulnerable to fraud [3]; answers can quickly be found by LLMs (see Section 4). Their advantage is that the expertise of non-fraudulent respondents will be detectable. **Time-boxed** technical questions, also recommended by Danilova et al. [8, 9], are more difficult for individuals to game. Submitting them to LLMs may be too slow. However, the answers can be coded into bots, and disseminated in survey farms. Therefore time-boxed questions are also vulnerable to fraud.

More difficult tests of technical skill include **writing code snippets**. Alami et al. asked *interview* candidates to undertake short coding tasks [3]. Such tasks require time to complete. This overhead may be acceptable to candidates for paid interviews, who expect a higher financial reward later at interview stage. They are a disincentive to taking an online survey [3, 9].

Survey researchers do not usually have the opportunity to meet their participants, and tend to assume that once all screening tests have passed they are left with only valid respondents [3, 12]. However, that is not always the experience of field researchers. In non-SE work, Guy et al. [19] deferred payment to study participants while they used various techniques to detect whether respondents were genuine. At the end of their second study, they conducted in-depth **video interviews** with validated respondents. There were *'eight verified ineligible during videoconferencing; seven verified eligible during videoconferencing or through chain referral.'* (Chain referral is a technique used to identify members of vulnerable populations.) Meeting the respondent clearly has high verification value, although it is an impractical overhead for survey participants and is incompatible with anonymity. In an SE example, Ebert et al. applied screening tests to find relevant interview subjects on a recruitment platform. They interviewed eight of these subjects, and found that six interviews had to be disregarded and they were left with only two [11]. These results raise question-marks over the efficacy of screening tests that do not involve personal encounters.

Checking Attention. Researchers are usually anxious to ensure that survey respondents are paying attention to the survey, and not just clicking blindly. This leads to various types of attention checks. In research on college fraternities, Schalewski et al. tried simply **asking respondents** to agree with statements like *'I am reading this survey carefully,' 'I am telling the truth on the survey,'* and *'The answers I have given on the survey are true.'* [45]. Over 21% (n=125) of respondents failed at least one of these questions. These questions are easy for fraudsters, but could highlight respondents who are inattentive or answering at random, rendering results unreliable.

A more traditional approach is to **ask the same thing twice** in slightly different ways and check for consistency in responses. This may detect inattention but is just as likely to create it. Surveys should be engaging and entertaining. Asking the same question repeatedly with different wordings is likely to send people to sleep, is not respectful of their time, and can seem suspicious [46]. As well as inattention, these questions may detect fraud if not coded or provided for by bots or survey farms.

Attention questions in the **request specific response** style, such as those in the Ryan survey (see Figure 2 and Table 1), have had success in detecting survey fraud [33]. Meem et al. [24] included an attention question in this style, motivated by Murphy-Hill [26]. They removed 289 out of 800 survey responses which failed this single attention question. This is a proportion of 36%, suggesting bot or survey farm activity in survey responses. By contrast, in the Ryan survey, the first attention question was failed by only 1.4% of respondents, while the second was failed by only 3.2%. This discrepancy between the two studies may be based on compensation. Meem et al. had survey compensation in the form of a draw for a gift card, suggesting that the prospect of payment, however little, for participation in online surveys encourages fraudulent or inattentive responses.

Assess Data Quality. Researchers frequently assign a minimum **time to completion** for survey responses [19, 24], This can de-

Table 7: Studies that implemented screening. Shows study initial numbers, numbers that failed the three main screening types: basic screening, technical questions and attention questions, numbers that failed other tests (OT), and final eligible number. 'Notes, or description of Other tests (OT)' provides space for notes or may describe additional screening tests undertaken in the study. The 'Failed OT' column lists numbers of respondents who failed each additional screening test described, in order. Numbers do not always sum to initial number because in some studies participants failed multiple tests.

Paper	Payment	Source	Initial no.	Failed Screen	Failed Tech	Failed Att	Notes, or description of Other tests (OT)	Failed OT	Final No.	Final %
Alami et al. 2024 [3], Study 1 | Study 2	£0.50	Prolific	914 | 1,000	Unknown	Unknown	NA	Notes: Accepted limited numbers per day. Examined free-text comments for authenticity	Unknown	436 | 480	48% | 48%
Choudhuri et al. 2025 [7]	None stated	Internal survey tools in GitHub and Microsoft; team leads asked to distribute	343	20	1	29	OT: Excluded those who did not answer all close-ended questions	55	238	69%
Danilova et al. 2020 [10], Recruit 1 | Recruit 2	€43 (to platform) | €20 gift card	Qualtrics | DB, contacts, Xing	129 | 47	NA	96 | 4	5 | 2	OT: Quality issues, strange or inconsistent responses / Did not complete	4, 2	22 | 28	17% | 60%
Ebert et al. 2022 [11]	Amount unknown	Prolific	22	6	19	NA	OT: Must have submitted/reviewed GitHub PRs	22	0	0%
Endres et al. 2022 [12]	draw for five $100 awards	GitHub, uni CS emails, Twitter	1,045	236	NA	NA	OT: Time taken < '1.5 standard deviations below the median', Inconsistencies	6	803	77%
Escobar-Avila [14]	None stated	Facebook, Twitter, mailing list	205	NA	NA	NA	OT: Stated not a coder	1	204	100%
Liang et al. 2024 [23]	draw for four $100 gift certs	GitHub	410	NA	NA	NA	Notes: No tests or screening discussed. Open-ended questions present and analysed	NA	410	100%
Meem et al. 2024 [24]	draw for gift card	X, LinkedIn, contacts	800	68	NA	289	OT: Time taken, Excessive response selection (e.g. 22 ethnicities), AI responses, strange or inconsistent responses	125, 3, 5, 158	331	41%
Newman et al. 2025 [28]	Optional draw for $120	GitHub emails	502	NA	NA	NA	OT: Self-consistency filters and removing responses with hate speech	9	493	98%
Reid et al. 2022 [33]	£0.35	Prolific	680	223	151	24	OT: Nonsensical responses, is node.js programmer	5; 222	55	8%
Ruby et al. 2025 [34], Unpaid | Paid	unpaid | $5 gift card	Contacts, hospital & clinic posters, social media etc.	73 | 2,440	NA	NA	NA	OT: Ineligible or incomplete | Time taken, quality issues, strange or inconsistent responses, invalid email	19 | 2,440	54 | 0	74% | 0%
Sabouri et al. 2025 [44]	$15 Amazon gift card	Email, DM on social media such as LinkedIn	29	NA	NA	NA	Notes: Participants directly recruited. Open-ended questions included.	NA	29	100%
Shamim et al. 2025 [48]	None stated	10 respondents from 'Company-A'; 10 from OSS projects	20	NA	NA	NA	Notes: Participants directly recruited. No payment; also no screening or open-ended questions.	NA	20	100%
Trinkenreich et al. 2025 [55]	None stated	Linux Foundation Diversity and Inclusion survey	2,350	NA	NA	NA	OT: Removed incomplete responses	1,644	706	30%

tect unsophisticated bots and fraudulent responses [12]. Griffin et al. assessed the average piloting completion time and based their minimum time on this [18].

Free text open-ended questions, where answers can be examined and parsed by researchers, frequently end up being used for survey screening [3]. In some studies, researchers were first alerted to fraud by inconsistent or gibberish responses to such questions [18]. Even if not intended for fraud detection, open-ended questions allow survey respondents to provide explanations, raise questions, and engage with the research topic. Griffin et al. recommend their extensive use, and that researchers should consider making at least one such question mandatory [18].

Inconsistencies between data points may also be relevant; Meem et al. found examples such as '*'No' for knowledge of APR tools but 'Yes' to experience using APR tools.*' This is different from repeating questions for attention reasons, in that it is fortuitous. Researchers may not have expected to detect fraud this way. Sometimes their first clue that something is wrong is when a developer has more years of experience than their age.

Detect Bots and Survey Farms. A useful tool for uncovering bot responses is to use **honeypots**: programmatically hidden questions that are only visible to bots [34]. The bots may not detect that the question is hidden from humans, and may answer the question, confirming the presence of a bot [18]. However, more sophisticated bots may detect this known strategy [58]. Guy et al. used **delayed compensation** as a deterrent to fraudsters, specifying in the survey agreement that payment would only be made after responses were examined for fraud, which could take up to a month [19]. This approach can deter both fraudulent and genuine respondents.

Best Survey Screening Approaches. Researchers should always plan a comprehensive approach to fraud detection in anonymous surveys. The survey platform's optional checks should be used. Prescreening questions are essential. A realistic minimum time to completion should be determined. Surveys should have at least one open-ended question allowing free text; more are preferred. Consistency checks and scrutiny of answers should be planned. Honeypots, attention questions and careful analysis of results all contribute to data quality.

However, we found no foolproof strategies to detect survey fraud, a finding reflected in Westwood's recent work [58]. We must broaden the debate to fraud *motivators*. The key driver of fraud is compensation [3, 10, 11, 24, 33, 58]. This is most vividly illustrated by Ruby et al.'s experience [34]. After offering a $5 payment they received 2,440 responses; all were discarded for poor data quality.

Recruitment strategy is key. Limiting recruitment to a specific group sidesteps the problem [7, 23, 28, 44, 48]. For widely-distributed surveys, removing incentives to cheat is the most powerful tool available. We see little to no fraud where incentives are not present [12, 14, 15, 36]. Therefore, like Ghazi et al. [16], we highly recommend that researchers should not pay for survey responses.

Answer to RQ3: A range of strategies can be adopted including use of open-ended questions, consistency checks and timing checks. They vary in the extent to which they are practical, and are effective deterrents. To avoid fraud, we recommend recruiting selectively, not compensating participants, or both.

6 Discussion

6.1 Limitations and Threats to Validity

6.1.1 Limitations. We analysed exclusions from only a single study: Ryan et al. (n=1048) [36]. We observe that for many studies (e.g. [5]), screening details, failure numbers and datasets are unavailable [3]. The Ryan data is published, and has multiple advantages. The target population included all coders, with no language, platform or other restrictions. The data was considerable (n=1048, 62 questions). It used widely adopted, empirically validated screening questions from Danilova et al. [9]. It closed before the arrival of free LLMs, so was not exposed to LLM fraud. Many questions allowed respondents to enter free text. Free text survey entries are widely used to detect fraud and aid the analysis of response quality [3, 18, 21, 24, 34].

We mitigated use of a single study by triangulating with other information sources. First, we compared the Ryan study's failure rates for CQ1 and CQ2 to those of the cohort of genuine programmers in Danilova et al. [9], observing that they corresponded. Second, we described authors' reflections on screening from many other studies [3, 8, 11, 18–21, 23–25, 33, 35, 59] in sections 2.2, 2.3 and 5.

6.1.2 Internal threats to validity. The original exclusions were done automatically, based on predetermined criteria. Evaluating the actual validity of respondents is subjective. We mitigated this by having the data independently analysed by three researchers.

There is a possibility that responses in the Ryan survey were subject to fraud. Our focus in analysing this data is on showing that genuine coders *could* have taken the survey and failed the programming tests. We found evidence suggesting that many excluded respondents were indeed coders. A mitigating factor is that this survey closed on January 31st 2022. The first version of the first freely available LLM, ChatGPT, was not released until ten months later in November 2022. Copilot appeared in February 2023. When the survey was open there was no easy way for non-experts to generate convincing responses to technical questions. Furthermore, respondents' free text was unstructured; their spelling and grammar errors, complaints and jokes added authenticity. The Ryan survey did not compensate respondents, removing the incentive to cheat.

The ability of a single LLM to answer the screening questions might not reflect LLMs in general. To mitigate this we tested the top five free LLMs as of July 8 2025.

LLMs may fail to answer screening questions different to those in the Ryan survey. One mitigating factor is that *survey* screening questions (unlike those for interviews or laboratory studies) must be relatively simple so that genuine programmers do not lose interest [9]. To succeed they should present approximately the same level of difficulty as those presented here. A second mitigating factor is that we analysed LLM ability to answer multiple different questions, and included all those we encountered in survey papers.

There may be screening techniques not found by our review of relevant SE literature. We mitigated this by searching Google Scholar extensively for survey screening techniques and fraud.

6.2 Discussion

As noted by Punter et al. in 2003, 'with the increasing pervasion of the Internet it is possible to perform surveys easily and cost-effectively over Internet pages (i.e., on-line)' [31].

The era of 'on-line' surveys may be ending. The simplicity of recruitment online is deceptive. It is difficult to know who is responding to surveys. When compensated, fraudulent responses may be generated by individuals, bots and survey farms. This is a documented issue in many domains including the social and health sciences [18, 19, 34]. Techniques such as setting minimum response times, checking for data inconsistencies, adding honeypots and evaluating free-text responses cannot prevent fraud. They may help to detect it, but are vulnerable to sophisticated actors [58].

In the SE context, researchers may wish to determine that respondents are genuine programmers, and may use screening questions for this purpose. However, our detailed analysis of the Ryan survey shows that using screening questions to exclude non-developers can result in inadvertently excluding weak or hasty developers. This can be considered unethical and, depending on the survey content, size and context, it has the potential to bias results. Meanwhile, the screening questions are no longer effective discriminators for true developers. Answers can be obtained by individual fraudulent responders via LLMs. Bots can code correct answers, and survey farms can distribute them.

What are the implications for SE research? Researchers may begin to focus more on other research strategies, such as case studies and interview studies. Surveys may fall into disuse, because survey data can no longer be considered reliable. However, we have found two recruitment strategies that limit survey fraud or reduce fraud incentives. The first is to send survey invitations to known populations. Choudhuri et al. recruited internally within the GitHub and Microsoft organisations, and used the organisations' internal survey tools [7]. This is the post-Internet equivalent of the traditional method of sending hardcopies of surveys by post to relevant organizations. It removes the potential for fraud. However, it is not usable for surveys that wish to have a broad contributor base. It would not have worked for our Ryan survey of software security practice and culture in industry. The second strategy, previously recommended by Ghazi et al. [16], is not to pay for survey responses. We observe little or no fraud in uncompensated surveys run before the advent of LLMs, indicating that compensation is the largest fraud incentive [12, 14, 15, 36].

The issue with uncompensated surveys is that recruitment is difficult and time consuming. Researchers are very resistant to removing compensation. It seems to be a blind spot. Studies we encountered comparing recruitment strategies [13, 20, 53] did not include any uncompensated groups. Others documenting massive issues with respondent fraud did not even mention complete removal of compensation as an option [3, 18, 19, 34] .

6.3 Recruitment strategies for unpaid surveys

Survey invitation response rates are usually low, including responses solicited via personal contacts. GitHub, often used in the past, now forbids email scraping. Indeed we received this reply from a developer we had messaged on Twitter with a request to retweet our survey: '*Done! Thank you for not taking the route of scraping and spamming emails from github.*' Another option, which also helps to reduce fraud, is using targeted recruitment within organisations. However, employees may be suspicious that evidence of poor performance or understanding obtained from such surveys may make its way to their employers [46]. We found that the best option was to publicise our survey on social media.

When appealing to strangers on social media to distribute a survey, professionalism is paramount. Researchers should be clear about the survey's origins and purpose [49]. Response motivation can be higher if surveys are anonymous and if participants expect to be told the research outcomes [16]. Platforms usually have a way for users to indicate that approaches from strangers are unwelcome, such as making direct messages private. Such social cues should be respected. Appealing to developers to disseminate a survey will not work if it is not well designed. Surveys should be engaging, interesting and respectful of respondents' time. Survey instruments are not always attractively, or even coherently, designed, as anyone who is part of a university *'surveys'* mailing group can attest. We specifically focused on designing an engaging survey, and received spontaneous positive survey design feedback from respondents. We found that recruitment is easier if a survey is enjoyable to undertake. Programmers themselves will promote it. Nevertheless, we can confirm that publicising surveys takes hard work and time. The first author spent at least an hour a day for three months drumming up interest in the survey on social media. On the other hand, combing through fraudulent responses is not only time-consuming but also disheartening [3, 18, 19, 34]. We considered it better to spend time on the design and promotion side. Researchers should work hard to create useful and entertaining survey instruments, build a rapport with the developer community over time, and find non-invasive ways to engage and inspire trust [14, 24].

6.4 Conclusion

We assessed excluded responses to a survey undertaken in prior work [36], concluding that excluded respondents were probably genuine programmers. We then assessed LLMs' ability to answer developer screening questions. We found that most such questions can be answered by LLMs. The results of short programs can be easily found online. We assessed recent survey studies and techniques to detect survey fraud in the SE and other arenas. We found that no techniques are foolproof against sophisticated fraudsters, except video verification which is impractical in a survey context.

We conclude that screening questions exclude weak developers without excluding bots. They should no longer be used to assess developer skill. Researchers should adopt a range of survey screening techniques to minimise and detect fraud. Importantly, we found two recruitment strategies that deter fraud. First, recruit for surveys from pools of known developers in a targeted way. Second, if recruiting more broadly, do not offer survey compensation. Plan to spend time and effort on survey design, imaginative recruitment strategies, and persistence. This will allow you to maximise the quantity and quality of survey responses.

Acknowledgments

This publication has emanated from research supported in part by a research grant from Research Ireland under the US-Ireland R&D Partnership Programme Grant Number 23/US/3939 and by Research Ireland grants 13/RC/2094_P2 and 13/RC/2077_P2.

References

[1] 2026. Humanize AI text. https://www.humanizeai.pro/ [Online; accessed 14 Jan 2026].

[2] 2026. Undetectable AI Rewriter. https://rewriterpro.ai/ [Online; accessed 14 Jan 2026].

[3] Adam Alami, Mansooreh Zahedi, and Neil Ernst. 2024. Are you a real software engineer? Best practices in online recruitment for software engineering studies. In *Proceedings of the 1st IEEE/ACM International Workshop on Methodological Issues with Empirical Studies in Software Engineering*. 52–57.

[4] Bilal Amir and Paul Ralph. 2018. There is no random sampling in software engineering research. In *Proceedings of the 40th international conference on software engineering: companion proceeedings*. 344–345.

[5] Hala Assal and Sonia Chiasson. 2019. 'Think secure from the beginning': A Survey with Software Developers. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 1–13. doi:10.1145/3290605.3300519

[6] Sebastian Baltes and Stephan Diehl. 2016. Worse than spam: Issues in sampling software developers. In *Proceedings of the 10th ACM/IEEE international symposium on empirical software engineering and measurement*. 1–6.

[7] Rudrajit Choudhuri, Bianca Trinkenreich, Rahul Pandita, Eirini Kalliamvakou, Igor Steinmacher, Marco Gerosa, Christopher Sanchez, and Anita Sarma. 2025. What Guides Our Choices? Modeling Developers' Trust and Behavioral Intentions Towards GenAI. In *IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE Computer Society. doi:10.1109/ICSE55347.2025.00087

[8] Anastasia Danilova, Stefan Horstmann, Matthew Smith, and Alena Naiakshina. 2022. Testing time limits in screener questions for online surveys with programmers. In *Proceedings of the 44th International Conference on Software Engineering*. ACM, 2080–2090. doi:10.1145/3510003.3510223

[9] Anastasia Danilova, Alena Naiakshina, Stefan Horstmann, and Matthew Smith. 2021. Do you really code? Designing and Evaluating Screening Questions for Online Surveys with Programmers. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. 537–548. doi:10.1109/ICSE43902.2021.00057

[10] Anastasia Danilova, Alena Naiakshina, and Matthew Smith. 2020. One size does not fit all: A grounded theory and online survey study of developer preferences for security warning types. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering (ICSE '20)*. ACM, 136–148. doi:10.1145/3377811.3380387

[11] Felipe Ebert, Alexander Serebrenik, Christoph Treude, Nicole Novielli, and Fernando Castor. 2022. On recruiting experienced GitHub contributors for interviews and surveys on Prolific. In *International workshop on recruiting participants for empirical software engineering*.

[12] Madeline Endres, Kevin Boehnke, and Westley Weimer. 2022. Hashing it out: A survey of programmers' cannabis usage, perception, and motivation. In *Proceedings of the 44th International Conference on Software Engineering*. ACM, 1107–1119. doi:10.1145/3510003.3510156

[13] Madeline Endres, Westley Weimer, and Amir Kamil. 2022. Making a Gamble: Recruiting SE Participants on a Budget. In *Proceedings of the 1st International Workshop on Recruiting Participants for Empirical Software Engineering*.

[14] Javier Escobar-Avila, Deborah Venuti, Massimiliano Di Penta, and Sonia Haiduc. 2019. A Survey on Online Learning Preferences for Computer Science and Programming. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. 170–181. doi:10.1109/ICSE-SEET.2019.00026

[15] Tiago Espinha Gasiba, Ulrike Lechner, Maria Pinto-Albuquerque, and Daniel Méndez. 2021. Is Secure Coding Education in the Industry Needed? An Investigation Through a Large Scale Survey. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. 241–252. doi:10.1109/ICSE-SEET52601.2021.00034

[16] Ahmad Nauman Ghazi, Kai Petersen, Sri Sai Vijay Raj Reddy, and Harini Nekkanti. 2019. Survey Research in Software Engineering: Problems and Mitigation Strategies. *IEEE Access* 7 (2019), 24703–24718. doi:10.1109/access.2018.2881041

[17] GitHub Corporation. [n. d.]. GitHub Acceptable Use Policies. https://docs.github.com/en/site-policy/acceptable-use-policies/github-acceptable-use-policies

[18] Marybec Griffin, Richard J. Martino, Caleb LoSchiavo, Camilla Comer-Carruthers, Kristen D. Krause, Christopher B. Stults, and Perry N. Halkitis. 2022. Ensuring survey research data integrity in the era of internet bots. *Quality & Quantity* 56, 4 (2022), 2841–2852. doi:10.1007/s11135-021-01252-1

[19] Arryn A. Guy, Matthew J. Murphy, David G. Zelaya, Christopher W. Kahler, and Shufang Sun. 2024. Data integrity in an online world: Demonstration of multimodal bot screening tools and considerations for preserving data integrity in two online social and behavioral research studies with marginalized populations. (2024). doi:10.1037/met0000696

[20] Harjot Kaur, Sabrina Amft, Daniel Votipka, Yasemin Acar, and Sascha Fahl. 2022. Where to Recruit for Security Development Studies from: Comparing Six Software Developer Samples. In *31st USENIX Security Symposium (USENIX Security 22)*. 24. https://teamusec.de/publications/conf-usenix-kaur22/

[21] Makena L. Kaylor-Tapscott, Maddison N. Tolliver-Lynn, and Maureen A. Sullivan. 2024. Improving data quality in online parenting research. 33 (2024), e2525. doi:10.1002/icd.2525

[22] Ella Kokinda, Makayla Moster, James Dominic, and Paige Rodeghero. 2023. Under the Bridge: Trolling and the Challenges of Recruiting Software Developers for Empirical Research Studies. In *2023 IEEE/ACM 45th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*.

[23] Jenny T. Liang, Chenyang Yang, and Brad A. Myers. 2024. A Large-Scale Survey on the Usability of AI Programming Assistants: Successes and Challenges. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. ACM, 1–13. doi:10.1145/3597503.3608128

[24] Fairuz Nawer Meem, Justin Smith, and Brittany Johnson. 2024. Challenges and Opportunities for Survey Research in the Age of Generative AI: An Experience Report. In *2024 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 423–428. doi:10.1109/VL/HCC60511.2024.00066

[25] Aaron Moss. 2018. After the Bot Scare: Understanding What's Been Happening With Data Collection on MTurk and How to Stop It. https://www.cloudresearch.com/resources/blog/after-the-bot-scare-understanding-whats-been-happening-with-data-collection-on-mturk-and-how-to-stop-it/. [Online; accessed 15 July 2025].

[26] Emerson Murphy-Hill, Ciera Jaspan, Caitlin Sadowski, David Shepherd, Michael Phillips, Collin Winter, Andrea Knight, Edward Smith, and Matt Jorde. 2019. What Predicts Software Developers' Productivity? *IEEE Transactions on Software Engineering* (2019), 1–23. doi:10.1109/TSE.2019.2900308

[27] Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, and Matthew Smith. 2018. Deception Task Design in Developer Password Studies: Exploring a Student Sample. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*. USENIX Association, Baltimore, MD, 297–313. https://www.usenix.org/conference/soups2018/presentation/naiakshina

[28] Kaia Newman, Sarah Snay, Madeline Endres, Manasvi Parikh, and Andrew Begel. 2025. "Get Me In The Groove"': A Mixed Methods Study on Supporting ADHD Professional Programmers. In *IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 778–778.

[29] Ofir Pasternak. 2019. Market Research Fraud: Distributed Survey Farms Exposed. https://www.greenbook.org/insights/insights-technology/market-research-fraud-distributed-survey-farms-exposed.

[30] Nikhil Patnaik, Joseph Hallett, Mohammad Tahaei, and Awais Rashid. 2022. If You Build It, Will They Come? Developer Recruitment for Security Studies. *ROPES - ICSE 202* (2022).

[31] Teade Punter, Marcus Ciolkowski, Bernd Freimut, and Isabel John. 2003. Conducting on-line surveys in software engineering. In *2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings.* IEEE, 80–88.

[32] Austen Rainer and Claes Wohlin. 2022. Recruiting credible participants for field studies in software engineering research. *Information and Software Technology* 151 (2022).

[33] Brittany Reid, Markus Wagner, Marcelo d'Amorim, and Christoph Treude. 2022. Software engineering user study recruitment on Prolific: An experience report. In *Proceedings of the First International Workshop on Recruitment of Participants for Empirical Software Engineering*.

[34] Emma Ruby, Serine Ramlawi, Alexa Clare Bowie, Stephanie Boyd, Alysha Dingwall-Harvey, Ruth Rennicks White, Darine El-Chaâr, and Mark Walker. 2025. Identifying Fraudulent Responses in a Study Exploring Delivery Options for Pregnancies Impacted by Gestational Diabetes: Lessons Learned From a Web-Based Survey. 27 (2025). doi:10.2196/58450

[35] Daniel Russo. 2022. Recruiting software engineers on Prolific. In *Proceedings of the First International Workshop on Recruitment of Participants for Empirical Software Engineering*.

[36] Ita Ryan, Utz Roedig, and Klaas-Jan Stol. 2023. Measuring Secure Coding Practice and Culture: A Finger Pointing at the Moon is not the Moon. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. 1622–1634. doi:10.1109/ICSE48619.2023.00140

[37] Ita Ryan, Utz Roedig, and Klaas-Jan Stol. 2023. Unhelpful Assumptions in Software Security Research. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 3460–3474. doi:10.1145/3576915.3623122

[38] Ita Ryan, Utz Roedig, and Klaas-Jan Stol. 2024. Training Developers to Code Securely: Theory and Practice. In *Proceedings of the 2024 ACM/IEEE 4th International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS/SVM)*. ACM, 37–44. doi:10.1145/3643662.3643956

[39] Ita Ryan, Utz Roedig, and Klaas-Jan Stol. 2025. Supplementary material for ICSE 2026. https://figshare.com/s/7da48d7bfdd6b519f69e

[40] Ita Ryan, Utz Roedig, and Klaas-Jan Stol. 2025. "ImmediateShort-Term3MthsAfterThatLOL": Developer Secure-Coding Sentiment, Practice and Culture in Organisations. In *2025 IEEE/ACM 47th International Conference on Software Engineering - Software Engineering in Practice (ICSE-SEIP)*.

[41] Ita Ryan, Klaas-Jan Stol, and Utz Roedig. 2023. The State of Secure Coding Practice: Small Organisations and 'Lone, Rogue Coders'. In *2023 IEEE/ACM 4th International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS)*. 37–44. doi:10.1109/EnCyCriS59249.2023.00010

[42] Ita Ryan, Klaas-Jan Stol, and Utz Roedig. 2023. Studying Secure Coding in the Laboratory: Why, What, Where, How, and Who?. In *2023 IEEE/ACM 4th*

International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS). 23–30. doi:10.1109/EnCyCriS59249.2023.00008

[43] Ita Ryan, Roedig Utz, and Klaas-Jan Stol. 2025. Data and scripts for "'Immediateshortterm3mthsafterthatlol": Developer secure-coding sentiment, practice and culture in organisations'. https://osf.io/tfxp9/?viewonly=5fb840be1d6b43d19c54ebade77419de. [Online; accessed 19 July 2025].

[44] Sadra Sabouri, Philipp Eibl, Xinyi Zhou, Morteza Ziyadi, Nenad Medvidovic, Lars Lindemann, and Souti Chattopadhyay. 2025. Trust dynamics in AI-assisted development: Definitions, factors, and implications. In *IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE Computer Society.

[45] Lucas Schalewski, Jamie Utt, and Bryant Valencia. 2018. Increasing Survey Data Quality Using Screening Validity Questions. *Oracle:The Research Journal of the Association of Fraternity/Sorority Advisors* 13, 2 (2018).

[46] Raphael Serafini, Marco Gutfleisch, Stefan Albert Horstmann, and Alena Naiakshina. 2023. On the Recruitment of Company Developers for Security Studies: Results from a Qualitative Interview Study. (2023).

[47] Saijal Shahania, Myra Spiliopoulou, and David Broneske. 2025. Gotta Catch 'Em All... Or Not?: How LLMs Bypass Traditional Checks & Mimic Human Response Behavior in Web Surveys. In *Proceedings of the ACM Collective Intelligence Conference*. ACM, 113–128. doi:10.1145/3715928.3737491

[48] Shazibul Islam Shamim, Hanyang Hu, and Akond Rahman. 2025. On Prescription or Off Prescription? An Empirical Study of Community-prescribed Security Configurations for Kubernetes. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 707–707.

[49] Edward Smith, Robert Loftin, Emerson Murphy-Hill, Christian Bird, and Thomas Zimmermann. 2013. Improving developer participation rates in surveys. In *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, San Francisco, CA, USA, 89–92. doi:10.1109/chase.2013.6614738

[50] Elisa J. Sobo, David M. Goldberg, Sean W. Hauze, and James P. Frazee. 2025. Cheating or Competing? University Students' Experience of AI Marketing and What It Means for AI Literacy Programming. *Annals of Anthropological Practice* (2025), e70029. doi:10.1111/napa.70029

[51] Klaas-Jan Stol and Brian Fitzgerald. 2018. The ABC of Software Engineering Research. *ACM Transactions on Software Engineering and Methodology* 27, 3 (2018), 1–51. doi:10.1145/3241743

[52] Mohammad Tahaei and Kami Vaniea. 2022. Lessons Learned From Recruiting Participants With Programming Skills for Empirical Privacy and Security Studies. In *RoPES-ICSE 2022*. ROPES.

[53] Mohammad Tahaei and Kami Vaniea. 2022. Recruiting Participants With Programming Skills: A Comparison of Four Crowdsourcing Platforms and a CS Student Mailing List. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. ACM. doi:10.1145/3491102.3501957

[54] Marco Torchiano, Daniel Méndez Fernández, Guilherme Horta Travassos, and Rafael Maiani de Mello. 2017. Lessons learnt in conducting survey research. In *2017 IEEE/ACM 5th International Workshop on Conducting Empirical Studies in Industry (CESI)*. IEEE, 33–39.

[55] Bianca Trinkenreich, Zixuan Feng, Rudrajit Choudhuri, Marco Gerosa, Anita Sarma, and Igor Steinmacher. 2025. Investigating the Impact of Interpersonal Challenges on Feeling Welcome in OSS. In *IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE Computer Society.

[56] Dirk Van Der Linden, Emma Williams, Joseph Hallett, and Awais Rashid. 2022. The impact of surface features on choice of (in)secure answers by Stackoverflow readers. *IEEE Transactions on Software Engineering* (2022). doi:10.1109/TSE.2020.2981317

[57] Daniel Votipka, Desiree Abrokwa, and Michelle L. Mazurek. 2020. Building and Validating a Scale for Secure Software Development Self-Efficacy. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. ACM, 1–20. doi:10.1145/3313831.3376754

[58] Sean J. Westwood. 2025. The potential existential threat of large language models to online survey research. *Proceedings of the National Academy of Sciences* 122, 47 (2025). doi:10.1073/pnas.2518075122

[59] Jim Witschey, Olga Zielinska, Allaire Welk, Emerson Murphy-Hill, Chris Mayhorn, and Thomas Zimmermann. 2015. Quantifying developers' adoption of security tools. In *2015 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings*. 260–271. doi:10.1145/2786805.2786816

[60] Yanfeng Xu, Sarah Pace, Jaeseung Kim, Aidyn Iachini, L Bailey King, Theresa Harrison, Dana DeHart, Sue E Levkoff, Teri A Browne, Ashlee A Lewis, Gina M Kunz, Melissa Reitmeier, R Karen Utter, and Melissa Simone. 2022. Threats to Online Surveys: Recognizing, Detecting, and Preventing Survey Bots. *Social Work Research* 46, 4 (2022), 343–350. doi:10.1093/swr/svac023