Code Quality – Interview Guide

ITiCSE 2017 working group WG5

May 2, 2017

1 Main Contacts

- Daniel Toll, +46 ..., daniel.toll@lnu.se, Skype: ...
- Jürgen Börstler, +46 ..., jubo@acm.org, Skype: ...

2 Project Plan

...

2.1 Communication strategy

...

2.2 Timeline

...

3 Open Questions

...

Details regarding communication, timeline and questions
that have been discussed have been deleted.

The guide continues on page 4 with interviewer guide
(Section 4) and interview script (Section 5).

4 Interviewer Guide

4.1 Research questions (for interviewer and WG report only)

In this working group, we are interested in the perceptions of code quality of students, educators, and professional programmers. In particular, we want to investigate which quality aspects they perceive as more or less important and which sources of information they use to get information about quality issues. To do so, we want to elicit actual examples of "good" and "bad" code from students, educators, and professional programmers.

More specifically, we are interested in the following questions:

- How do students, educators, and professional programmers define and perceive code quality and in which ways do these definitions and perceptions differ?
- Where do students, educators, and professional programmers look for information about code quality (and why)?
- How do students, educators, and professional programmers assess the quality of their code (including tool usage)?

In addition, we also want to collect code that can be used in further studies as examples of "good" and "bad" code (i.e. where we have evidence that it actually is considered as good or bad $\rightarrow$ *gold standard*).

4.2 Study Design

We are conducting structured interviews with three groups of participants: students, educators, and professional programmers. The interviews contain closed and open questions. The closed questions have short answers, typically from a selection of given alternatives. These are filled in directly by the interviewer or interviewee. For the open questions, we audio record the interview and transcribe it into written form for further analysis.

Participants should bring along actual code they have experienced to discuss in one of the open questions. This is done to focus on actual code that the interviewees have experienced and its particular quality aspects as opposed to general aspects of quality (like code should be maintainable). Doing so will also make it easier to elicit personal, first-hand opinions, instead of repetitions of general beliefs about quality. To relate answers and code, we need to make sure to get access to the code or record the screen on which this code is displayed using screen recording software.

- Students.
  Only students in computer-/software-related programs? We should guarantee a minimum level of programming literacy. Since we already have different groups of participants, it might not be a good idea to cover too many different student groups.
- Educators.
  Educators should have at least a few years of experience with courses that deal with programming, software design, or software quality, i.e. a TA who graded labs in a single programming course would not qualify as an educator.
- Professionals.
  Should be people who actually deal with software development for a living, i.e. people

who regularly read, write, test or review source code or low-level designs. Perhaps we would like to get some spread among domains and or tasks (e.g., testing safety-critical systems versus developing web pages).

Participant sampling process

All interviewers will conduct at least three interviews. To find suitable interviewees we should keep in mind that our main goal is a broad coverage of perspectives and not primarily representativeness of interviewees. Ideally, sample size is determined by "theoretical saturation," i.e. adding more interviews is unlikely to uncover further concepts or aspects.

To facilitate a good spread of participants we determine three factors determine the spread of participant.

- Group (student/educator/professional).
- Gender.
- Geographical spread (most active country).
- Domain. Field of study for students and educators, work domain and work task (for example testing safety-critical systems in the automotive industry) for professionals.

We want to distribute participants reasonably evenly over the three groups, domains and geographical regions. We also want to be sure that all participants are good "key informants". An educator or professional should therefore have significant experience in his/her role. A freshly graduated student is therefore not a good candidate for an educator or professional programmer. Since every interviewer decides on his/her participants, it becomes our collective responsibility to ensure a good spread. When you are considering whom to interview, please check the participant spreadsheet at Google Drive (`https://goo.gl/`[redacted]). If possible select a group, region or domain/task not yet covered. When you select a participant "register" your name, the participant's code, etc. in the category you think most appropriate for the participant (student/educator/professional).

It is important to recruit independent participants. You should not interview your own students, since they will very likely be "biased" by what you told them in your classes. If that cannot be avoided, please mark this clearly.

Recruit and prepare participants for interviews

Participants should get a copy of the Interview-Info document provided here: `https://www.dropbox.com/s/`[redacted]. The interview-info document provides an explanation on the nature of the interview and the need for bringing along example code to discuss during the interview. Participants might need some time to find code before the interview, so suggest a time and date that gives them some time to prepare.

The interview-info document also explains that all data will only be used for this study and that their identity is not disclosed. If participants have questions that you cannot or will not answer, they can contact us by email using the email address provided in the document.

Anonymization of participants

Study participants (interviewees) are assured anonymity. For the purpose of the study all interviewees will get a participant code. To facilitate this, they have to fill in and sign a consent form (`https://www.dropbox.com/s/`[redacted]`0`). Each of the interviewers should store their participants consent forms. We recommend you scan them and store them securely. If a participant wants a copy, please feel free to do so.

Participant codes should be of the form FLN, where F and L are the first letters of the interviewer's first and last names, respectively and N number the interviewees for each interviewer starting with 1. Daniel's interviewees, for example, would get the codes DT1, DT2, etc.

Before the interview

Read the interview guide (this document) and bring along the interview script (Section 5). Make also sure to print copies of the consent form ([redacted]), Part 3 and 4 of the interview script (Section 5.5 and 5.6), and the reflection form (Section 5.8). The consent form and Part 3 and 4 of the interview will be filled in by the interviewee. A reflection form is filled in by the interviewer after each interview.

Structured Interview

Stick to the interview script (Section 5). For the initial **closed questions** (Q1–Q3, Section 5.3), the interviewer fills in the answers. The script suggests options. Ask follow-up questions as necessary to determine a suitable option.

Q4 (Section 5.4) is recorded and transcribed. The script suggests follow-up questions regarding aspects that should be elicited in more detail, e.g.,

- Circumstances of experiencing quality issues (positive and negative).
- Exemplification using actual code.
- How the interviewee learned about a certain code quality issue (who told him/her that the code was "good" or "bad", what exactly was "good" or "bad" and why.)
- How a certain issue was resolved.

The remaining **open ended questions** (Q5–Q11, Section 5.5 and 5.6) are filled in directly by the interviewee.

Things to avoid

Do not influence responses by asking leading questions or conveying your own view ("priming"). Instead invite interviewees to provide their own personal view. Emphasize also that we are interested in "good" and "bad" examples/experiences. Do not move too quickly from one topic to the next. Do not interrupt or truncate prematurely, but try to actively lead away from irrelevant issues.

Tips for a good interview

- Know your interview guide and probes well. Get to know the informed consent statement, so that it is conveyed confidently and casual.
- Rehearse the introduction.
- Be aware of power differentials.
- Don't judge interviewees. We are interested in their personal views, whatever they are.
- Be comfortable with silence, don't force answers; take pauses.
- Let the interviewee talk freely.
- Ask non-leading questions to encourage more (broader or deeper) answers.
- Do not offer suggestions, interpretations, or answers that might "prime" the interviewee.

5 Interview Script

This section guides the actual interview session. The interview session comprises three main parts: (1) the preparations before the interview (Section 5.2), (2) the actual interview (Section 5.3–5.7), and (3) a reflection after the interview (Section 5.8).

5.1 Preparations

Before the participant arrives

- ☐ Check the recording equipment (battery level, available storage).
- ☐ Make sure that the interview room has no distractions (turn off phones, close doors, etc.)
- ☐ Make sure that you have copies of the consent form (preferably 2, in case the interviewee want to retain a copy).
- ☐ Make sure that you have clean copies of Part 1 of the interview script (Q1–Q3, where you will fill in the interviewee's answers) and Part 3 and Part 4 of the interview script (Q5–Q11, where the interviewee will fill in the answers herself/himself).
- ☐ Make sure that you have a copy of the reflection form for you to fill in directly after the interview.
- ☐ Make sure that you have a screen that can be viewed by both you and the interviewee and mouse that can be reached by both of you.
- ☐ Make sure that you have any source code the participant provided ready for viewing. To easily browse the code and to easily identify/refer to parts of it, use a IDE which displays both FILE NAME and LINE NUMBERS.

5.2 Interviewee introduction and interview set-up

Introductory statement

"Good morning (afternoon, evening). My name is <*your name*>. Thank you for coming.
We are making a study on attitudes and perceptions regarding code quality. The purpose of this interview is to get information about your perceptions of and experience.
Please note that there are no right or wrong, or desirable or undesirable answers. I would like you to feel comfortable with saying what you really think and how you really feel. The whole interview should take about 50–60 minutes."

☐ Completed.

Get consent and let participant fill-in consent form

"If it is okay with you, I will be recording our conversation, so that I can get all the details and at the same time be able to carry out an attentive conversation with you."
"Before we get started, please take a minute to fill in and sign the consent form. You will get a participant code to ensure anonymity."

☐ Consent form completed.

Please fill in the following

Date and time for the interview.

Interviewer name.

Location for the interview.

Participant code.

Verbally ID the recording

When the consent form is signed and returned, assign the interviewee a participant code. Note the code on the consent form and **start the recording**.
Please ID the recording clearly by stating the following:

1. Date and time.
2. Interviewer name.
3. Location for the interview.
4. Participant code.

☐ Completed.

5.3 Part 1: About the interviewee

The first section contains questions regarding the interviewee and his/her actual experience in programming. This part is filled in by the interviewer. Please feel free to clarify questions or ask follow-up questions to get clear answers.

> ”First, I want to ask some questions about your background and your programming experience.”

Q1. Gender.
Male ☐ Female ☐

Q2. “In which country did you get most of your experience?”

> Rationale: We want to figure out where the interviewees experience belongs to. Even if the interviewee is currently working/studying in one country his or her experience might be gained elsewhere.

Q3. **A1** “What is your current occupation and job title (if applicable)?”

> Rationale: Is the interviewee a student, educator, or professional programmer. We are also interested in sub-categories, like types of students, programmers vs managers, etc.

Interviewee is a student:

S1 “What is your study program and level?” (Text, e.g., Bachelor of Software Engineering)

S2 “In which study year are you in terms of full-time equivalents?” (Number)

S3 “Is programming one of your main study subjects?” (Yes ☐ No ☐)

S4 “How many programming courses did you take, i.e. courses with a significant programming component?” (Number)

S5 “How many programming languages can you program in?” (Number; count only the languages that the interviewee actually can name!)

S6 "How large was the largest program you developed?" (Lines of code, number)

S7 "How many years of experience in teaching software development do you have?" (Number in approx. full-time years)

If more than 2 years, we also ask the questions for the educator category.

S8 "How many years of experience as a professional programmer do you have and how recent is this experience?" (Number in approx. full-time years)

If more than 2 years and/or reasonably recent, we also ask the questions for the prof. progr. category.

Interviewee is a professional programmer:

P1 "What is your job title?" (Text, e.g., developer, software architect, manager, team lead, tester, etc.)

P2 "Do your formal responsibilities involve quality assurance?" (Yes ☐ No ☐
If Yes, in which role, e.g., software quality assurance manager or process leader, etc.; Text)

P3 "Which programming languages do you use most?" (Text)

P4 "How many years of experience in teaching software development do you have?" (Number in approx. full-time years)

If more than 2 years, we also ask the questions for the educator category.

Interviewee is an educator:

E1 "How many years of experience in teaching programming related courses do you have?" (Number)

E2 "How many courses related to programming do you teach per year on average?" (Number)

E3 "Which courses related to programming did you teach during the last five years?" (List them, or let them write them down)

If there are few courses, ask for a longer time back.

E4 "Do you talk about code quality in those courses?" (Yes ☐ No ☐
If Yes, ask for which courses)

E5 "How many years of experience as a professional programmer do you have and how recent is this experience?" (Number in approx. full-time years)

If more than 2 years and/or reasonably recent, we also ask the questions for the prof. progr. category.

For all interviewees:

A2 "Which programming language do you prefer most?" (Text)

Rationale: There might be differences regarding language preferences and the preferred language could differ from the most used one(s).

A3 "On a scale from *strongly disagree* to *strongly agree*, how much do you agree or disagree with the following statements regarding your personal experience related to software development."

- I read and modify source code from other programmers.
  Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
- Other people are reading and modifying the code that I write.
  Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
- I review or comment other people's code.
  Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
- Other people review or comment the code that I write
  Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree

5.4 Part 2: Liked and disliked code

Here the participants should provide, share, or describe in detail an actual example of code that they have worked with. We want them to describe and exemplify in detail what exactly they find particularly good or bad with this example code.

To succeed, the interviewee and the interviewer need to view and navigate the code either on a paper copy or electronically using a mouse or similar. To capture what is viewed interviewee and/or interviewer should mention the file and the lines of interest.

> ”We will now look at some of the code examples you provided. When we are talking about a piece of code, I will state aloud which page, file, line and/or feature we are discussing, for example “we are looking at class Test.java line 15 to 65”. You can help me in this by doing so as well to make sure we can identify a piece of code by means of the recording.
> Can you please show me a piece of code that you like or dislike particularly?”

Q4. “Please describe in detail, which properties or features you like or dislike with this code and how these properties or features affect the quality of the code. Please note that there are no correct or incorrect answers. We are primarily interested in code features that matter for you and why it does so.”

This part is repeated a few times, so that a range of features are discussed.

Open Question, will be recorded and transcribed.

The following probing questions can be used to elicit more detailed information:

UNCLEAR SIGNS: It is unclear how to detect or verify a feature, property or issue.

- ALT: “What shows you that the code has the feature, property or issue you describe?”
- ALT: “How can we identify in the code that we have the feature, property or issue that you like/dislike?”
- ALT: “How do you find code that has this feature, property or issue?”

UNCLEAR EFFECTS: It is unclear why a feature, property or issue leads to high or low quality.

- ALT: “What effect has this feature, property or issue on a programmer’s behaviour?”
- ALT: “Which benefits or costs has this feature, property or issue?”
- ALT: “What exactly is the quality issue with the code (class, variable, ...) you are pointing to?”

UNCLEAR SOURCE of information: It is unclear how the interviewee learned about a quality issue.

- ALT: “How did you learn about the importance or relevance for quality of this feature, property or issue?”

SUGGESTED FIX (for disliked features): It is unclear how a quality problem could be solved.

- ALT: “How would you change this code so that it becomes better?”

In case the interviewee strays away from the core subject, here are a few suggestion to get him/her back on track:

- “If someone else is reading this code, would he or she understand it?”
- “What would you say indicates high or low quality in your example?”

”Do you have another example of a feature or property that you like or dislike?”

NOTE: If the interviewee has only talked about code he or she disliked or finds has low quality, please try to explicitly ask for an example of code he or she likes or finds of high quality (and vice versa). We want to encourage interviewees to talk about properties, features or indicators of high and quality.

“You have only talked about code that you dislike (like). Can you point out a part of the code that you like (dislike)?”

If more, continue with Q4. If no, you can end this part.

While this is the most interesting part of the interview, it should not drag on forever. Make sure to keep the interviewee focused on code quality and keep in mind that everything needs to be transcribed as well. It would be sufficient to discuss 3–4 features, properties or issues in detail.

“In the following part of the interview, I want you to answer a few questions in writing. Please ask if you need or want to clarify something.”

Rationale: We want to investigate participant’s definitions and perceptions of code quality. NOTE: We must not influence (“prime”) participants about what we or others think code quality is. We want to know more about their personal perceptions.

Hand over a copy of Part 4 of the interview script (Questions Q5–Q11). Don’t forget to mark the copy with the participant code.

5.5 Part 3: Perceptions of source code quality

Participant code.

Q5. "How would you define code quality? Which properties, features or indicators show you, personally, something about its quality?"

Q6. "On a scale from *strongly disagree* to *strongly agree*, how much do you agree or disagree with the following statements regarding your personal experience related to source code quality."

- Code Quality is of high importance in my work/studies/teaching.
  Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
- I can easily tell good from bad code.
  Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
- I regularly work with code quality issues.
  Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
- I know how to measure code quality.
  Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
- I have learned a lot about code quality during my education.
  Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
- I have learned a lot about code quality from my colleagues.
  Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
- I have learned a lot about code quality from the Internet.
  Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree

Q7. "Please provide your top-3 recommendations for increasing the quality of code. Please indicate when a recommendation applies in special cases only."

- My top recommendation for achieving high quality code.

- My second most important recommendation for achieving high quality code.

- My third most important recommendation for achieving high quality code..

- Any further important recommendations you want to mention?

Q8. "According to your experience, what are the three topmost quality factors or indicators of high quality code?"

- The most important quality factor/indicator for high quality code.

- The second most important quality factor/indicator for high quality code.

- The third most important quality factor/indicator for high quality code.

- Any further highly important factors you want to mention?

5.6 Part 4: Sources of quality related information

Q9. "According to your experience, what are the three most useful sources of information about code quality? Are these sources reliable and trustworthy?"

- The most useful source of information about code quality.

- The second most useful source of information about code quality.

- The third most useful source of information about code quality.

- Any further highly useful resources you want to mention?

Q10. "According to your experience, what are the three most useful tools for improving code quality or achieving high quality code?"

- The most useful tool for improving code quality.

- The second most useful tool for improving code quality.

- The third most useful tool for improving code quality.

- Any further highly useful tools you want to mention?

Q11. "Is there anything more you would like to bring up?"

5.7 End of interview

"Thank you for your participation."

Remember

- Turn off the recorder. Make sure the recording is labelled with the participant code and stored safely, preferably at a location with backup.
- Make sure that the consent form is filled in and signed.
- Make a note on the consent form whether the interviewee agreed to further contacts.
- Make sure that the copies of Part 1 and Part 3 of the interview are filled in completely and have the correct participant ID.
- Make sure to collect and ID all other materials that the interviewee left, like notes, code examples, etc.

5.8 Reflection

After the interviewee leaves the room, please take a couple of minutes to reflect on the interview. Are there any particular observations that might be relevant for the analysis of the interview?

Please indicate the following (a separate form is available):

1. Interviewer name.

2. Participant code.

3. Date and time of the interview.

4. The participant's attitude toward the interview.

5. Unusual circumstances or events that might have affected the interview, such as interruptions, language difficulties, etc.

6. Anything related to the actual example code and/or its presentation that might have had some effect on the study's objective, e.g., unusual comments or identifiers used in the code, messages on the equipment that was used to show/project the example, files or web pages that turned up while locating the code.

7. Additional comments.