October 08, 2019

Surveys to assess developer perceptions of code complexity

Jason Wilkinson

Jeffrey Warshaw

Elizabeth Kammer

Edward Smith

Ciera Jaspan

See next page for additional authors

Inventor(s)
Jason Wilkinson, Jeffrey Warshaw, Elizabeth Kammer, Edward Smith, Ciera Jaspan, Collin Green, and Yvette Nameth

Surveys to assess developer perceptions of code complexity

ABSTRACT

Although a codebase can be assessed objectively for computational complexity, e.g., time-to-execute, memory occupied, etc., there are currently no techniques to assess code complexity as perceived by a developer. Questions of relevance to software development, e.g., how easy or difficult it is to work with a codebase, which parts of the codebase are more complex than anticipated, etc. are not readily answerable. This disclosure describes techniques that automatically send surveys to developers at pertinent points during code development or maintenance. Developer responses are collated and analyzed to determine developer-perceived complexity, e.g., sections of code that are of excessive complexity, are difficult to work with, etc.

KEYWORDS

- Perceived complexity
- Code complexity
- Subjective complexity
- Code review
- Survey
- Developer feedback
- Experience sampling

BACKGROUND

Although a codebase can be assessed objectively for computational complexity, e.g., time-to-execute, memory occupied, etc., there are currently no techniques to assess code complexity as perceived by a developer. Questions of relevance to software development, e.g., how easy or difficult it is to work with a codebase, which parts of the codebase are more

complex than anticipated, etc. are not readily answerable. Questions relating to developer-perceived complexity are only subjectively judged and lack effective ways of objective measurement.

The occasional survey that does attempt to capture perceived complexity is not automatic, is infrequent, e.g., quarterly, and too broad to produce accurate or relevant data. Besides, present-day surveys are conducted long after developers have completed their work, making them hard to target correctly. Current techniques do not produce a point-in-time snapshot of developer's thoughts.

DESCRIPTION

This disclosure describes techniques to send to developers small surveys in real time at various stages of the developer's workflow. The surveys are characterized by the following:

- sent directly to the individual doing the work.
- sent immediately after the developer's work is complete.
- just enough in number to enable large-scale data collection over time.

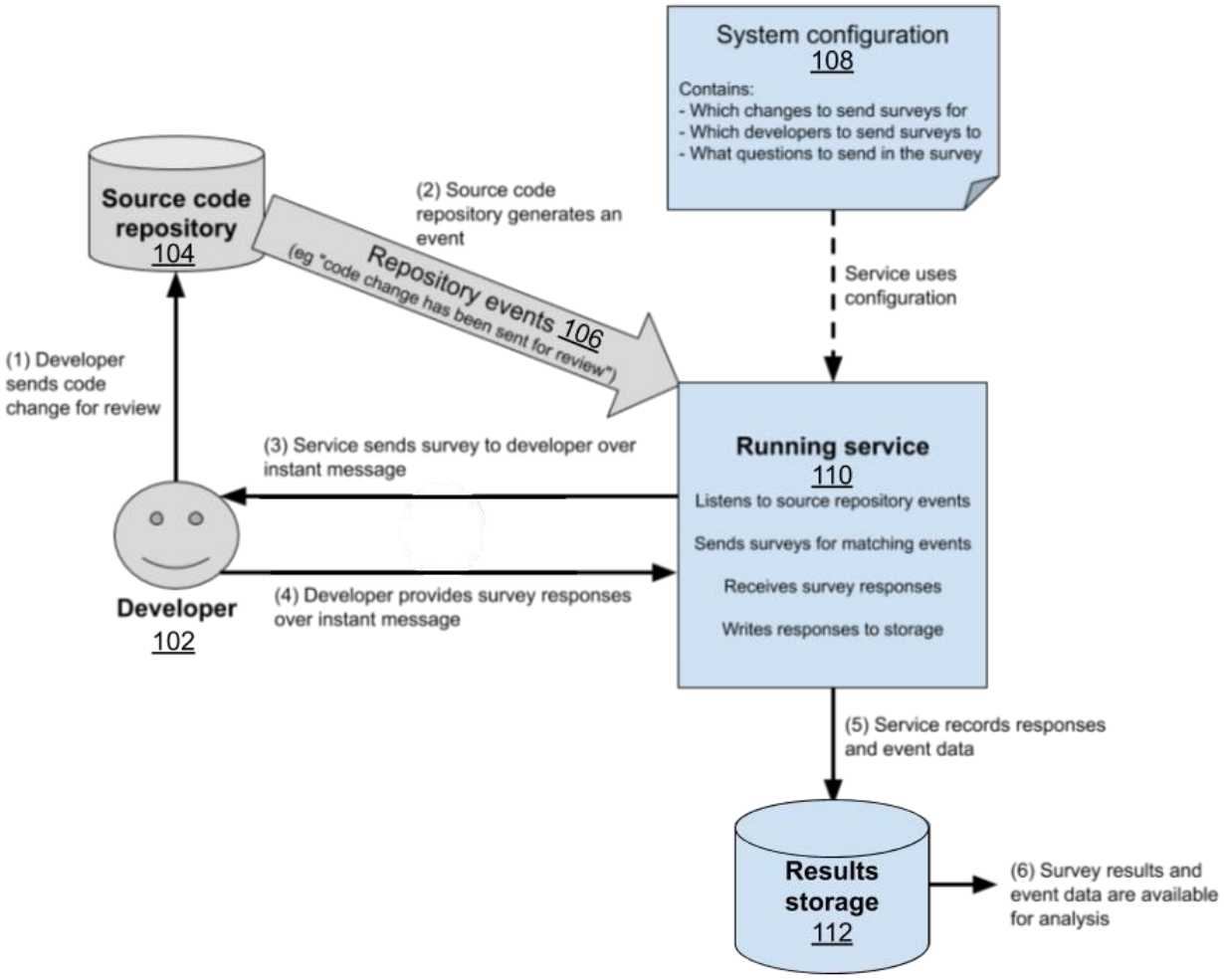

Fig. 1: Assessing code complexity using surveys

Fig. 1 illustrates assessing code complexity using surveys, per techniques of this disclosure. A developer (102) sends in code changes for review to a source code repository (104). A running service (110) monitors the source code repository for events (106) that occur during the development workflow. Events occur in different phases of the development workflow. Examples of code development phases include the phase when a developer sends in code changes for review; the phase when a change-list is submitted to the codebase; etc.

Within each phase, the running service listens for events that come from an external source. When events arrive, these are processed in order. An event is processed based on a

system configuration (108), which includes, e.g., triggers organized on a per-project basis. A trigger is essentially a filter that describes the survey that is to be sent if the trigger matches. Example triggers are as follows: which changes to send surveys for; which developers to send surveys to; what questions to ask in the survey; to which parts of the codebase the trigger applies; etc.

If the event being processed matches a trigger, the running service determines the survey to send and the person to send the survey to. For example, the person to send the survey can be the author of the change. The running service customizes the specified survey and sends it to the relevant person. The recipient of the survey responds to the survey. The responses are received by the running service, which sends them to results storage (112) for analysis.

A survey includes one or more configurable questions. Questions generally relate to changes to the codebase made by the respondent to the survey, and can be open-ended (free text) or multiple-choice. Some example survey questions are as follows:

- How complex did you (the developer) expect it to be when you started the project?
- How complex was the change actually? (Alternatively, were your expectations regarding complexity met?)
- Where in the codebase were your expectations regarding complexity not met?
- Where in the codebase was your work hindered by what you consider to be unnecessary complexity?
- Any comments or examples to illustrate your answers to the above questions?

The survey is an assessment of the codebase and of its complexity by the developers. The questions of the survey are customizable such that the survey can be generalized to measure other subjective qualities, e.g., effectiveness, happiness, etc.

The survey is designed to be frequent enough to achieve accurate measurement while minimizing impact on developers. For example, it is ensured that a certain minimum time elapses between surveys being sent to the same person. Submission of code changes is not contingent upon a response to a survey. Respondents can choose to opt in or out of the survey, check statistics, learn about the survey and its goals, etc. using, e.g., a web-based user interface. The survey itself is customizable to suit different projects, triggers, questions, times-between-surveys, etc. For example, different teams can, by a change in configuration, ask developers questions specific to their teams or projects.

The survey can be sent using any suitable channel, e.g., via an instant messaging service using a chat bot or another suitable mechanism. When the survey is sent via a chat bot, the user can respond simply by interacting with the chat bot, which stores the survey results. An online user interface can also be provided to enable users to set survey preferences (e.g., opt in or out of surveys), check statistics, and get more information about the survey system.

Since the survey is sent on a real-time basis, e.g., triggered by code repository events, responses can be obtained while the work is fresh in the minds of the developers. Since the respondents are individuals that actually work on the codebase, and since the surveys are conducted at a relatively large scale and on a real-time basis, the accuracy of the results is high. The survey techniques described herein can be provided for large-scale code repositories, including private codebases at large companies, public codebases of open source projects, etc. The techniques described herein therefore enable large-scale collection of subjective reports and/or qualitative data from developers in the course of their work.

CONCLUSION

This disclosure describes techniques that automatically send surveys to developers at pertinent points during code development or maintenance. Developer responses are collated and analyzed to determine developer-perceived complexity, e.g., sections of code that are of excessive complexity, are difficult to work with, etc.