What Questions Developers Ask During Software Evolution? An Academic Perspective

Renato Novais[1], **Creidiane Brito**[1], **Manoel Mendonça**[2]

[1]Federal Institute of Bahia, Salvador – BA – Brazil

[2]Fraunhofer Project Center for Software and Systems Engineering at UFBA
Salvador – BA – Brazil

{renato,creidianebrito}@ifba.edu.br, manoel.mendonca@ufba.br

Abstract. *Many studies on software evolution propose new approaches. One needs to validate the approaches. For that, it is common to conduct experimental studies where participants have to answer questionnaires with questions related to the research topic. Those questions must be relevant, otherwise the validation may be invalid. In this context, this study investigates which questions (and so the tasks) developers really answer (and perform) during software evolution. To this end, a survey comprised of 11 questions was applied to 42 participants from academia. This study allowed to derive an initial model on questions developers ask during software evolution, and to understand how the participants agreed with the relevance of the questions.*

1. Introduction

Several studies on software evolution and maintenance propose new methods, techniques, and tools. To validate their approach, it is common to conduct experimental evaluation, where they try to answer questions or to perform maintenance tasks. For example, in [Wettel et al. 2011], the authors used a set of ten program comprehension and design quality assessment questions to investigated the efficiency and correctness using software visualization. In [Bavota et al. 2012], the authors analyzed the impact of test smells on program comprehension during maintenance activities. For that, they applied a questionnaire with 16 questions related to software test. In [Hattori et al. 2013], the authors also evaluated their tool based on a set of six evolution tasks. And in [Novais et al. 2012b] and [Novais et al. 2012a], the authors evolved a software evolution visualization tool motivated by the need to answer two feature evolution comprehension questions.

The relevance of the questions or tasks used is, in general, based on literature reports. However, if those tasks are not really used on the state of the practice, the approaches and therefore, their evaluation, may not guarantee their applicability.

Based on this assumption, we decided to investigate which questions (and so the tasks) developers really answer (and perform) during software evolution. The study presented in [Sillito et al. 2006] already investigated this topic. In our study, we decided to go further to see if the questions reported in that study, and in others found in the literature, are really relevant for developers and practitioners.

[Sillito et al. 2006] conducted two qualitative studies of programmers (newcomers and industrial programmers) performing change tasks to medium to large sized programs. Based on those studies, they cataloged and categorized 44 different kinds of questions asked by their participants. [de Alwis and Murphy 2008] highlighted the difficulty

of programmers have to answer questions as they investigate the functioning of a software system. They must piece together results from different tools to determine an answer to the initial query. To overcome this issue, they proposed a model that supports the integration of different sources of information about a program and a tool that implements this model. On the same token, [Fritz and Murphy 2010] also pointed out the challenges on answering a variety of questions that require the integration of different kinds of project information. To investigate this topic they interviewed 11 professional developers. From this step, they identified 78 questions developers want to ask, but for which support is lacking.

The general goal of this work is to investigate how academics and practitioners see those questions, and discover if they are following the same directions. If the answers are no, we may have a big issue on the directions of the research on the academic context that should be always well aligned with real industrial problems.

In this paper, we present the results of the investigation with the academic researchers. To this end, we performed a survey with 11 questions and with 42 people from academia. This paper contributes with the results of this survey on the academic perspective and also presents an initial model for questions developers ask.

This paper is organized as follows. Section 2 presents the experimental evaluation. Section 3 shows the results of the study. Section 4 discusses some threats to validity. Finally, Section 5 concludes this paper, presenting directions for future works.

2. Experimental Evaluation

2.1. Goal Definition

This study is part of a larger study that aims to investigate what questions are relevant during software evolution. The goal is to collect the academic and practice perspective and compare the results. In this paper, we present the first part: the academic perspective.

2.2. Planning

Selection of suitable set of questions. The first step on this study was to identify a set of questions that could be used on the survey. Therefore, we conducted a literature review on the study context.We selected eight papers that report questions developers ask while performing software evolution tasks [Sillito et al. 2006] [de Alwis and Murphy 2008] [Fritz and Murphy 2010] [D'Ambros and Lanza 2007] [Tu and Godfrey 2002] [German et al. 2004] [Ackermann and Zazworka 2010] [D'Ambros and Lanza 2009]. The first three papers are the main reference points since they are focused on the same topic: investigate questions programmers ask during software evolution tasks.

This step generated a list of 212 questions. This set of questions is very large, which is fairly impossible to apply a survey with all of them. To overcome this issue, we decided to group the questions. We observed that the questions had intersections and could be grouped in what we called Representative Questions. Each representative question grouped similar questions we found in the papers. For example, a representative question 'What module has been recently modified?' may refers to similar questions as 'What classes have been changed?', 'Which API has changed (check on web site)?', or 'What methods or functions were changed?', etc.

Regarding the grouping step, we rely on the three most qualified experts to ensure the soundness of our clustering. In short, the experts validated the mapping of 212 questions to the 11 questions reported in Table 1.

Table 1. List of Questions.

Question	Description
Q1	What is the impact of a change in the source code?
Q2	Which module has been recently modified?
Q3	Which module has undergone more changes?
Q4	When a change occurred in the source code?
Q5	Who has changed a module?
Q6	Who has more impact in a given module?
Q7	How much work has someone performed in some module?
Q8	Where particular person has worked?
Q9	For a given module, what are the changes that affected it?
Q10	What was the reason behind the changes made in a given module?
Q11	How was the process of changing a module?

Response options for each question. The participants had to answer the questions according to two perspectives. First, the level of relevance for each question. The options were: i) *Completely Irrelevant* (CI); ii) *Little Relevant* (LR); iii) *Relevant* (R); and iv) *Very Relevant* (VR). Second, whom the information that the question addresses is related to. The options were: i) *Software Developer*; *Software Manager*; iii) *Both*; and iv) *None*.

Participants. The survey had 42 participants. We selected master and PhD students from an experimental software engineering class at Federal University of Bahia, and students from a specialization from a scientific seminar class at Federal Institute of Bahia.

We asked the participants to answer five questions about themselves. They were: c1) How many software deployed for the client have you developed any software maintenance activity?; c2) How many years have you worked with software development activities?; c3) How do you consider yourself regarding to someone else that most know about the related topics? (I am fair below, I am little below, I am at the same level, I am above); c4) What is your position on the current company (if working)?; and c5) Do you consider yourself more academic or practitioner?

Figure 1 shows the participant characterization. Question c4 had very different answers so we omitted it.

2.3. Execution

The execution process was conducted during master and PhD class sections at Federal University of Bahia and during a specialization class section at Federal Institute of Bahia. The participants took no more than 30 minutes to answer the survey. They had to answer the survey, and then fill a form characterizing them.

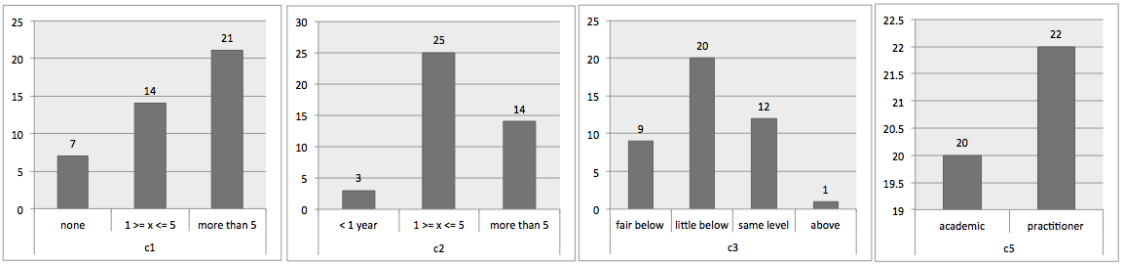

Figure 1. Characterization of the survey's participants.

3. Results

3.1. An Initial model for questions developers ask

Figure 2 presents an initial model derived from the grouping questions step. Most of the questions had a relationship among them. We observed that there are three important entities of interest when asking question related to software evolution: change (modification report), source code (module), author (user, developer). The questions may be only related to the specific entity (e.g. what are the *authors*?, what are the modules of the software?), but generally they relate two entities of interest (e.g. which *author* modified this *module*?).

Figure 2 depicts six questions relating two by two of these entities. The arrows link the two entities, and the direction of the arrow means the main entity on the question. The arrows highlight the historical information of interest. The color of the arrow portrays the stakeholder (developer, manager, both) that can use the information provided by the task that answers the question. We observed that the question is important to the manager, the manager and the developer, but never only for the developer. This is acceptable since the manager should understand every think about the project.

There are still two important dimensions that can be added to these questions: time and effort. It is common to find question like: what changes have *recently* affected this module? and which authors have *mostly* worked on this module ?

We believe that developers of software evolution may benefit from this initial model. They can use it as a initial guide of questions that their tool should try to answer.

3.2. Questionnaire's answers

Figure 3 shows the total of answers per each question of the survey. It presents the big picture of the survey answers. For each question there are at most four bars. From left to right side, the bars mean: *Completely Irrelevant*, *Little Relevant*, *Relevant*, and *Very Relevant*.[1]

The first insight one may have from this picture is that most questions had the *Relevant* as the main response. Only questions Q1, Q4, and Q9 had the *Relevant* option response less than 50%. The others overlay this mark. Even more, only question Q1 the *Relevant* option was not the highest option.

[1] View it on the computer screen, or in a color printed version for best results

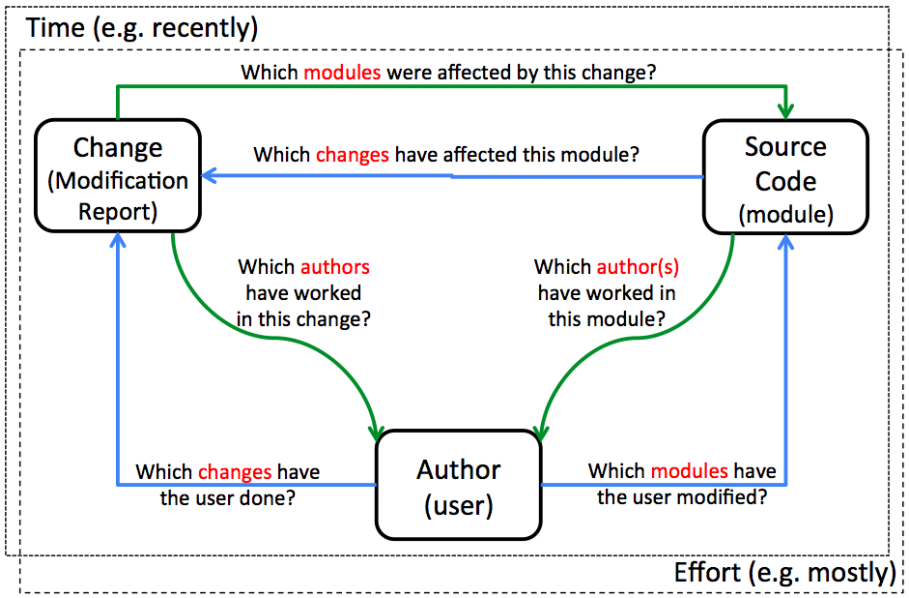

Figure 2. An Initial Model for questions developers ask

As can be observed, few questions had *Completely Irrelevant* answer (Q1: 2; Q6: 4; Q8: 1; and Q10: 2). The question with highest value for this category was the Q6, with 4 respondents. This question is related to who has more impact in a given module. In other words, who has changed more the given module? On the other hand, 50% of the respondents agreed that this question is *Relevant*.

The category *Little Relevant* is presented in all of the questions. The highest values are for the questions Q5 and Q7 with 14 respondents. In all of the questions, it was higher or equal to the category *Completely Irrelevant*.

On the same token, the category *Relevant* is also presented in all of the questions. The highest values were for the questions Q2, Q3, and Q8 with 29, 26 and 27 respondents, respectively. It is also important to notice that, for all the questions, this category had higher value than the category *Little Relevant*.

All questions have at least four respondents on the category *Very Relevant*. For the question Q1, 71.43% of the respondents selected this category. This is also the highest agreement of the participants.

Figure 4 shows how the participants evaluated each question considering whom the information that the question addresses is related to. Again, for each question there are at most four bars. From left to right side, the bar means: *Software Developer*, *Software Manager*, *Both*, *None*.

The category *Software Developer* was presented in all of the questions. Question Q7 had the lowest value for this category: 1, while question Q2 had the highest.

The category *Software Manager* was in almost all questions. Only question Q1 had no respondents for this category. The three highest score for this category were on questions Q7, Q8 and Q7, with 31, 29, and 22, respectively. Moreover, question Q7 had the highest value among all questions and categories.

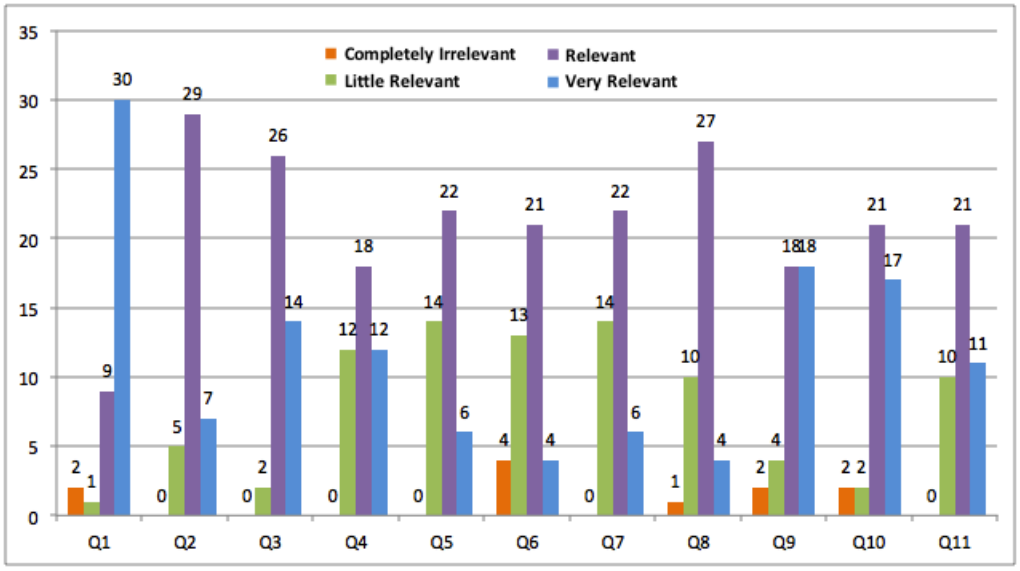

Figure 3. Participants' answers per question.

The category *Both* considered *Software Developer* and *Software Manager*. It was also presented in all of the question. The highlights were for the question Q1, Q10 and Q9, with 30, 23 and 22 scores, respectively.

Finally, the category *None* is presented in 7 questions, with no more than 5 answers (question Q6). Analyzing both Figures 3 and 4, it is possible to observe that question Q11 has two respondents for *None* category, and zero respondents for *Completely Irrelevant*. This may mean that the participants believed this question was *Little Relevant*, *Relevant*, or *Very Relevant*, for someone else not the *Software Developer* or *Software Manager*. Questions Q2, Q4, Q6, and Q7 present the same pattern.

3.3. Discussion

The results showed that most participants aggreed with the relevance of those selected questions. To support this affirmation, let's considers grouping the results between the categories, as follows: negative category (*Completely Irrelevant* and *Little Relevant*) and positive category (*Relevant* and *Very relevant*). In this case, it is possible to observe that all questions had positive answers with more than 50%. The questions Q5, Q6 and Q7 had the lowest level, with 28, 25 and 28 respondents, respectively. The others were higher than 30 respondents. Questions Q3, Q1, and Q10 had the highest values, with 40, 39 and 38, respectively.

As a general conclusion, according to these results, the questions/tasks are important on the state of the practice, when one needs to evolve his/her software. This guide us to conduct the second part of the larger study, that will investigate how expert (and real practical) people agree with the relevance of those questions/tasks.

4. Threats to Validity

This study has some threats to validity. The first one we identified is the way we used to select the first set of questions. We started by the most referenced papers on the topic we know [Sillito et al. 2006] [de Alwis and Murphy 2008] [Fritz and Murphy 2010]. As

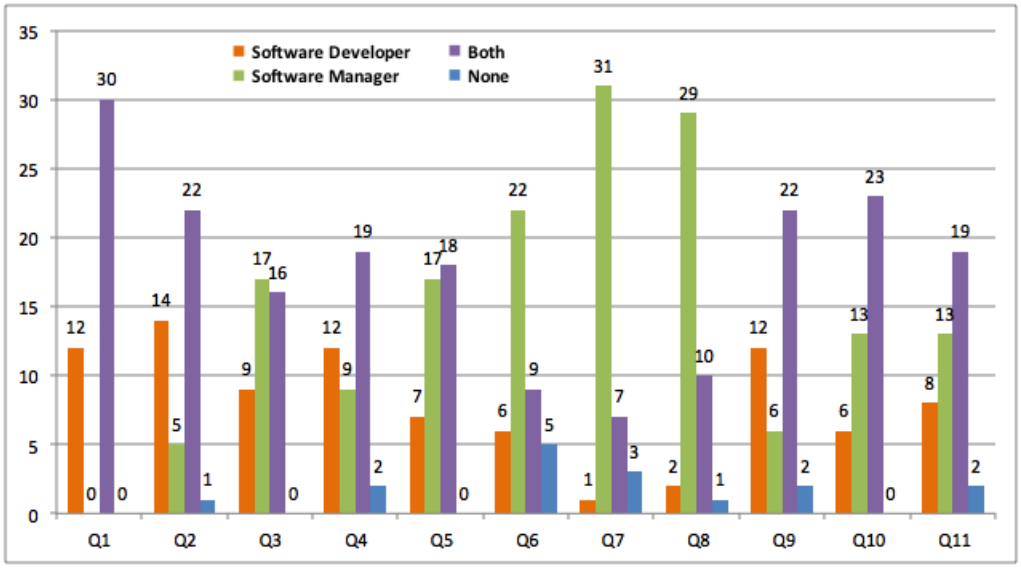

Figure 4. Relevance of the questions for stakeholders.

they are from the same research group, we decide to add other papers we found that also had questions/tasks that developers ask/perform.

Other threat may be related to the first group of task. We needed to reduce the number of questions. Therefore, we requested experience people to help us on this step. Even we based on expert opinions, this may be biased. One could for example, conduct a larger survey with independent expert people.

The lower statistical power is a threat to the conclusion validity since we did not used strong statistical techniques to analyse the results. Finally, but not lastly, we point to the participants we selected. They may not be representative, however we tried to select the most heterogeneous we could count on.

5. Final Remarks

This paper presents an academic perspective survey on questions developers ask during software evolution. It was motivated by repeatability that researchers use such question on the validation of their studies. The survey was composed of 11 questions and was applied to 42 respondents. The studies allowed us: i) to derive an initial model on questions developers ask during software evolution, and ii) to understand how the participants agreed with the relevance of the questions.

The work presented here is part of a larger study, which intends to investigate and compare how academic and practitioners envision those questions on the state of practice. Thus, as a future work, we aim to apply the same survey with industrial participants and compare the results with the academic participants. Other direction is to analyze the produced information per participant level. This may guide us to a further understanding about the produced data.

References

Ackermann, C. and Zazworka, N. (2010). Codevizard: Combining abstraction and detail for reasoning about software evolution. Thecnical Report - University of Mariland.

Bavota, G., Qusef, A., Oliveto, R., De Lucia, A., and Binkley, D. (2012). An empirical analysis of the distribution of unit test smells and their impact on software maintenance. In *Proceedings of the 28th IEEE International Conference on Software Maintenance*, ICSM'12, pages 56–65.

D'Ambros, M. and Lanza, M. (2007). Bugcrawler: Visualizing evolving software systems. In *Proceedings of the 11st European Conference on Software Maintenance and Reengineering*, CSMR'07, pages 333–334, Washington, DC, USA. IEEE Computer Society.

D'Ambros, M. and Lanza, M. (2009). Visual software evolution reconstruction. *J. Softw. Maint. Evol.*, 21(3):217–232.

de Alwis, B. and Murphy, G. (2008). Answering conceptual queries with ferret. In *Proceedings of the 30th ACM/IEEE International Conference on Software Engineering*, ICSE'08, pages 21–30.

Fritz, T. and Murphy, G. C. (2010). Using information fragments to answer the questions developers ask. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, ICSE'10, pages 175–184, New York, NY, USA. ACM.

German, D., Hindle, A., and Jordan, N. (2004). Visualizing the evolution of software using softChange. In *Proc. Int. Conf. on Software Engineering & Knowledge Engineering*, SEKE'04, pages 336–341, New York NY. ACM Press.

Hattori, L., D'Ambros, M., Lanza, M., and Lungu, M. (2013). Answering software evolution questions: An empirical evaluation. *Inf. Softw. Technol.*, 55(4):755–775.

Novais, R., Lima, Paulo, R., and Mendonça, M. (2012a). Timeline matrix: an on demand view for software evolution analysis. In *Proc. of the 2nd Brazilian Workshop on Software Visualization*, WBVS'12, pages 1–8.

Novais, R., Nunes, C., Lima, C., Cirilo, E., Dantas, F., Garcia, A., and Mendonca, M. (2012b). On the proactive and interactive visualization for feature evolution comprehension: An industrial investigation. In *Proc. of the 34th International Conference on Software Engineering*, ICSE'12, pages 1044–1053.

Sillito, J., Murphy, G. C., and De Volder, K. (2006). Questions programmers ask during software evolution tasks. In *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, SIGSOFT'06/FSE-14, pages 23–34, New York, NY, USA. ACM.

Tu, Q. and Godfrey, M. W. (2002). An integrated approach for studying architectural evolution. In *Proceedings of the 10th International Workshop on Program Comprehension*, IWPC'02, pages 127–, Washington, DC, USA. IEEE Computer Society.

Wettel, R., Lanza, M., and Robbes, R. (2011). Software systems as cities: a controlled experiment. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE'11, pages 551–560, New York, NY, USA. ACM.