CS846-001 Week 3 Reviews

Jun Qing Lim
University of Waterloo
jq3lim@uwaterloo.ca

1 Cowboys, Ankle Sprains, and Keepers of Quality: How Is Video Game Development Different from Software Development?

1.1 Problem Being Solved

The paper provides an investigation on the difference between game development and software development, aiming to obtain to better understand the impacts of software engineering research on game development. It studied the survey outcomes from engineers of both game and non-game development background and using questionaire as an approach of assessing the process of developing software (non-games) and games.

1.2 New Idea Proposed

The authors conducted surveys across engineers (primarily from Microsoft) and presented the following outcomes:

1. games and software are built with different requirements, with games focused on "play" and non-games focused on "functional requirements"
2. games have less design to produce more "fun" play thus they are designed for a different demography of society
3. games have higher emphasis on performance, which reduces the frequency of code reusability and increases innovative ideas
4. games have less automated testing because the whole experience of usability evolves around human playing through the game than a test script

It was then further concluded that games and non-games involve distinct set of skills and motivations and that software development in both fields are inherently different in its nature.

1.3 Positive Points

Firstly, the paper highlighted its arguments with several examples and case studies to support its claim, which is well-written and well structured.Secondly, the paper is written in simple-fashion and broadly, providing greater accessibility to a wide spectrum of readers not just software and game developers but also for scholars. Thirdly, the paper provided insightful reflection to developers in the industry today on how we could better improve the general software development process and the intersection of these fields.

1.4 Negative Points

Firstly, the surveys of the paper were conducted across a particular demography, which may be bias towards the outcome of the survey (i.e., sample size and study methods might be too narrow). Secondly, the study of the paper only covers a limited aspects of game industry and does not mention other facets of software that could affect the development process such as sales, commercialization, marketing, etc.

1.5 Future Work

One possible extension is to conduct a broader and thorough investigation to explore the differences and similarities between the development of games and non-games, which could help reduce the stereotyping or biases outcomes of a study. The other extension is that we could also fine-tune the research to investigate the ideas that could be adapted and applied from games development to software development (and vice versa), creating greater learnings on a cross-domain level.

1.6 Rating

4 out of 5. Paper that is thoroughly researched and well-written and offers unique standpoint on games and non-games.

1.7 Discussion Points

- Is there a different set of development practices across different countries which affect the general processes?
- How did technological advances in the past change the development flow of games and software?

2 Analyze This! 145 Questions for Data Scientists in Software Engineering

2.1 Problem Being Solved

The paper presents the survey results of data science in software engineering - of 145 questions that software engineers would like data scientists to answer involving software, software processes and practices. It aims to address the issue of data scientists often not knowing where to begin their research in software. As such, the objective of this paper is to provide data scientists a starting point for research in software engineering, allowing data scientists to easily focus on their topics that have higher relevancy to the software industry.

2.2 New Idea Proposed

The authors proposed a set of 145 questions grouped into 12 categories on software processes and practices. These range of topics include bug measurements, development practices, best practices, testing practices, evaluating quality, services, customers and requirements, development lifecycle, productivity, team collaboration, etc. These questions could help data scientists in establishing a foundation of research in software engineering. Data scientists can then further use these questions to conduct specific study of their research.

2.3 Positive Points

Firstly, the paper is written in a way that is easy to understand, making it able to reach wide range of readers not just people in the software industry. Secondly, the paper act as a useful resource for researchers, practitioners,educators and not just data scientists to help understand problems and advancements in their field. Thirdly, the 145 questions are grouped into 12 categories which are then ranked according to the importance of each question, which makes it easy for researchers to navigate the questions according to its research priority.

2.4 Negative Points

Firstly, 145 questions might be considered as a massive set of questions for any single individual to think deeply and not all questions are relevant to its research topic. Secondly, the paper lacks thorough review or study on the effectiveness of these questions on a practical sense, which may from time to time may not be completely relevant to the current state of software engineering.

2.5 Future Work

One extension that we could make is to conduct further research on the effectiveness of the questions in the current state of software industry on a domain-specific level, this will further solidify the relevancy of the questions. We could also expand this set of questions to other domains instead of only for data scientists, applying similar concepts, which will expand our scope of understanding of the software industry.

2.6 Rating

4 out of 5. It provides valuable resource for researchers to conduct study.

2.7 Discussion Points

- 145 questions is a massive list, how can we narrow our focus to specific (quality) questions that have the same impact?
- Did any technological advancements in the past affected the relevancy of these questions today?

3 Investigating Code Review Quality: Do People and Participation Matter?

3.1 Problem Being Solved

The paper investigates the factors that affect the quality of a code contribution of developers. It discovers that the practices of code review among developers help to increase code quality, promotes best practices and reduces bugs. It address the issue of code review quality by looking into aspects such as number of reviewers, experience/expertise of coding, and level of participation in code reviews.

3.2 New Idea Proposed

The authors propose the following ideas:

1. even with code reviews, they may not necessarily prevent all bugs completely
2. an experienced reviewer has a higher chance of producing higher quality of code reviews
3. too many code reviewers for a single developer to handle could diminish the quality of code reviews
4. active participation of code review process by all developers help to build skills in reviewing quality code
5. smaller code changes are less likely to introduce bugs

3.3 Positive Points

Firstly, the paper uses codebase from Mozilla as a sample size of study, which is considerably a large dataset, providing a solid evidence base of conclusions. Secondly, it illustrated detailed reasonings for the elements that affect code review quality such as level of participation, experience and quantiy of reviewers, which is convincing. Thirdly, it gives software developers and organizations awareness on how to improve review process, this could help reduce unnecessary bugs.

3.4 Negative Points

Firstly, the paper only uses single codebase, as such the findings of the study might not be easily generalizable to other organizations (i.e., exists some biases). Secondly, the paper does not take into account of other factors of a "quality code review" such as an organization's cultures, practices, and objectives, as such the findings may not be relevant for all types of organizations.

3.5 Future Work

As illustrated under the negative points, we could extend the work of this paper by examining codebases across different domains, this would provide a broader sample size of study and findings. Furthermore, we could also conduct another study to investigate the relationship between expertise and experience of reviewers and its code review quality, this would allow us to better understand if experience is a necessary factor for good code review.

3.6 Rating

3 out of 5. The findings of the study are not easily generalizable for all development circumstances.

3.7 Discussion Points

- What are some effective ways to increase the level of participation and engagement of developers in code review processes?
- Could we utilize any automated tools to help prevent missed bugs in a code review?