Paper Review – Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development?

Paper Author: Emerson Murphy-Hill et al.
Paper Review Author: Felix Wang

Problem Being Solved

The central research question of this paper is: How is video game development different from software development? Most likely, the answer is that it's different in some ways but similar in other ways. Although we already know the answer, it's still worth investigating, because it's rarely studied – in the past 2 years, only 3/116 software projects studied at major Software Engineering (SE) venues were games, and game developers often hold themselves apart from formal SE. This paper, as the first study that systematically and empirically investigated the differences between traditional SE development and video game development, it has the implication that if we could prove they are different, the tools and practices for traditional SE development will not be applicable to game development, so this could open up a new research direction; or if we could prove that they are similar, game developers will be able to benefit from the established ad validated tools and practices of traditional SE development.

New Idea

The methodology used in this paper combined qualitative interviews and quantitative surveys. In the interview, they recruited 14 interviewees found on LinkedIn, each of whom had at least 2 years of game development and at least 2 years of non-game development in the past 10 years. And in the qualitative surveys, we used a 5-point Likert scale survey consisting of 28 statements, and the survey had 364 participants working in the Games, Office, and Other departments of Microsoft.

In the interview results, on the requirements aspect, interviewees perceive that functional requirements are for non-game development, games are supposed to be fun and subjective, so no strict requirements are needed. And game's user experience is different from non-games, and unfulfilled requirements in games tend to be less problematic. Requirements are from multiple sources, and gamers, as the ultimate users of games, have less say in the game's requirements. On the design aspect, they perceive that there's less up-front thought on architecture, and since it's very hard to predict a game's lifespan, it's hard to determine if reusability is important. Game development also tends to reward more for creativity. From the construction, tools, and methods perspective, they believe that project-specific performance tuning cannot be reused. Also, games tend to focus more on innovation than similarity. Some still admitted that there is some code reuse on game engines or general SE development tools. Games are considered to be less testable because the state space is too large, there's no clear definition of what is correct behaviour, many game algorithms are non-deterministic, automated testing is fragile to frequent changes, and software testing is more expensive than human testing, etc. Maintenance is often delayed in game maintenance, sometimes because of less management buy-in, sometimes because new content is already enough for product release, and there's no need to modify or extend the program behaviours. One believes that the cloud-based platforms like Steam might enhance this. For configuration management, games normally have a significant amount of content, and configuration management is often chaotic. Part of the problem is due to a lack of code review. Developers perceive that people in game management positions tend not to have technical backgrounds, and it's hard to communicate engineering issues with non-engineers, because they don't respect engineering activities that have no immediate impact. Many think agile is a good fit because of the game's unpredictability. Some admit that there's a lack of process. But there's a concern that such a way of imposing control may oppress creativity. Many think game development has much more pressure on releasing software on time.

For the general work topics, people perceive that game development has distinct technical challenges, and people's understanding of fun is subjective. Also, game developers require a wider variety of skills, specialists like sociologists, anthropologists, and economists are required. Game developers perceive that they need a high level of autonomy, and communication is crucial to bridge interdisciplinary gaps. Game developers believe that they tend to have a stronger tie to the customer. Profitability and game rewards are good indicators of results. And game developers find meaning in their work by knowing how many people use it. They perceive themselves to have "celebrity status" if the game they developed is popular, and they feel meaningful to provide positive emotional experiences for others. A lot of people reported that the video game industry is notorious for long working hours, and sometimes developing motion-based games is physically demanding.

The results of survey data confirmed that, in game development, there are less clear requirements, they tend to use Agile more, creativity is valued more, the ability to communicate with non-engineers is valued more, they require a more diverse team, and people are more impressed by game development. But they disconfirmed that the management team's technical background shows no difference.

Positive Points

One positive point is that they have combined qualitative data with quantitative data analysis methods. The authors themselves admit that the interviews and surveys individually provide limited insights, but by combining together, they can cross-validate each other and increase the depth and breadth of insights.

The next thing I like about this paper is the soundness of their interview analysis. Normally, in an interview study, we emphasize the interpretive depth and emerging patterns rather than the counts or distributions of the perceptions. So sometimes, even if only a single participant mentioned something, it could be analytically meaningful. This paper did a great job on that, they focused on the conceptual richness and the theoretical relevance of responses and didn't simply ignore non-mainstream opinions.

Negative Points

Although it's been considered acceptable in software engineering studies, I still don't like the fact that this paper is taking the mean of Likert-scale responses. The number of Likert-scale responses is an ordinal number, which has no numerical meaning. Because we could never guarantee that the interval between "strongly agree" and "partially agree" is the same as the interval between "partially agree" and "neutral". This departs from the strict survey analysis methodology.

The next thing I don't like about this paper is that the participants for their surveys are: Games (145), Office (100), and Other (119). The statistical method used by this paper tends to have more statistical power in classes with more samples and less statistical power with fewer samples. With participants in the game being 45% more than in offices, this would possibly overshadow subtle effects in smaller groups. This paper guarantees that participants should have at least 2 years of experience in game development and at least 2 years of non-game development in the past 10 years. But by this kind of selection criteria, some biased participants who are more senior in game development and less senior in non-game development (like 8 years of game dev + 2 years of non-game dev) would be selected. So there's a possibility that these developers will have a mindset of game developers and hold themselves apart from software engineering. This will possibly introduce biases like "We shouldn't be restricted by unit tests", "We shouldn't be restricted by considering reusability", "We shouldn't care more about gamers' requirements that much", etc. Because the original URL in the paper is down, I couldn't get access to the original interview data to verify this, but I would suggest at least putting the median years of game and non-game development experience in the table on page 3.

Future Work

I'm still concerned about whether the unit tests would truly benefit game development, so I propose a lab-based controlled experiment on unit tests in game development. My research question is, will using unit tests in game development using MVC architecture enhance the quality of games? I will organize a coding session for 40 participants to complete two game dev coding tasks (P1 and P2) of a game's Model part, with or without writing unit tests. We will equally and randomly assign 40 participants into four groups following a Latin square design. For example, in group A, we will give them P1 first without letting them write unit tests, then we give them P2 and let them write unit tests to reach at least 70% statement coverage. We alter the order of questions and whether developers are required to write unit tests in the four groups. And eventually, we measure the quality of code.

Rating

4.5/5. I think it's an extremely novel paper, which identifies the underexplored domain in software engineering, and opens the research perspective on a series of game-related studies, including game dev practices, empirical standards on gamification, etc. But it contains minor imbalances and is not exhaustive enough.

Discussion Points

(1) What caused the disparity that the interview study suggests that people in game management positions tend not to have technical backgrounds, but survey studies disconfirmed that? Which one should we believe in?

Class responses:

Some students in class believe that most managers do have certain levels of technical backgrounds. One student mentioned that some managers do call themselves to have a technical background because they can code, but it's been a long time since they participated in real hands-on development, and they actually lack an in-depth understanding of the modern frameworks and domain tools. One student highlighted that the interview study only has 14 interviewees, while the survey study has 264 participants, which makes it more reliable. The presenter added that both sources can be correct: interviews capture perceptions; surveys capture large-scale demographics; it's worth studying why perception and measurement diverge.

The professor mentioned an anecdote about promotions in the 1990s until people plateaued (Peter Principle), while today’s flatter hierarchies may change how “technical” managers are perceived.

(2) In section 4.1.8, it says “interviewees reported being under significantly more pressure to release the software on time for games than for non-game software”. Do you think this is inherently why game dev is different from non-game dev, or is it the consequence of a lack of process?
Class responses:
Some people mentioned that it’s a multi-factor issue. Many pointed to the intrinsic complexity of games, with huge state spaces and heavy testing demands, as a major driver of pressure. One student mentioned the studio culture and how developers are treated as amplifiers of pressure regardless of formal process. Some perceive that deadlines are not the core problem compared to overall complexity.

(3) In section 4.1.3, it says “there is less code reuse in games … because reuse implies similarities between software, yet games emphasize innovation”. Do you think traditional non-game devs are pursuing similarities rather than innovations? Do you think there is a trade-off between similarities and innovations?
Class responses:
One student said teams often prefer familiar tools and patterns to avoid relearning costs; non-game applications learn on standard features, while games still follow genre conventions. Others noted that mechanics evolve during development, making stable reusable components is difficult. The presenter asked whether reusable components should exist for specific genres. One student highlighted that, just like developing non-game software, people build a reusable framework or architecture as its root, and customize its branches that are not reusable, the game engines are the code reused for most of the time.

(4) After reading this paper, do you think the game development is more “art”, more “science”, or more “engineering”?
Class responses:
One described game design as “art”, game development as “engineering”, and neither of those is “science”.
The professor said he couldn’t agree more. He said building engineering systems isn’t the same as doing science. Science is more theoretical (e.g., mathematics) and focuses on building reliable models of how things work. Interviewing, constructing empirical standards is more “science”, however, developing software is more of an engineering process, despite it’s called computer science.