How Is Video Game Development Different from Software Development in Open Source?

Presentation by Asim Waheed

CS846 – Mike Godfrey – Week 6

Introduction

- Builds on Murphy-Hill et al.'s 2014 "Cowboys and Ankle Sprains" study
- Shifts focus from industrial to open-source game development
- Aim: understand how OSS game development differs in team roles, bug handling, and developer perception

The Problem

Why study this?

What makes OSS game development different?

- SE research rarely studies open-source video games
- Game devs themselves claim the process is *meaningfully different*
- Authors ask:
  What actually makes OSS game development different from other OSS software?

New Idea

What did we learn?

Data Collection

- 60 projects total → 30 OSS games + 30 non-games
- Languages: mostly C/C++ and Objective-C
- Methods: repository mining + developer survey

Research Questions

RQ1: How do OSS game vs non-game developers contribute to their projects?

RQ2: How do they prevent & handle malfunctions?

RQ3: How do they *perceive* their development process?

RQ1 Highlights

- Classified project files into 10 categories (code, audio, images, etc.)
- Found heavy emphasis on multimedia files in games
- Developers tend to form specialized sub-teams (graphics, audio, etc.)
- Over time: initial focus on code → steady work on assets

RQ2 Highlights

- Categorized malfunctions (e.g., graphic, performance, security)
- Games → more graphic-related faults and security issues
- Non-games → mostly programming errors
- Games also show wider spread of fault types

RQ3 Highlights

- Game devs struggle with code reuse and automated testing
- Less clear requirements and architecture evolution
- Greater emphasis on creativity and cross-disciplinary skills
- Aligns with industry findings but quantified in OSS context

Positives

Did we like the paper?

Comprehensive Multi-Method Approach

- Authors did not just mine repos and leave it at that.
- Combined quantitative analysis *and* surveys

Novel Focus on OSS Games

- Prior work mostly studied industrial or commercial settings
- Studying open-source games gives a different lens:
  - Fewer deadlines
  - More volunteers
  - Different motivations
- Important step to understand wider ecosystem of game dev

Large Dataset

- Sixty projects is decently large
- Inclusion of both 2D and 3D games
- Dataset size itself is an achievement

Negatives

What ideas were a miss?

Different Domain Same Analysis

- Whole premise is OSS game development is different
  - Yet; analysis pipeline is standard.
- Analysis should be adapted to the domain
  - Files will *obviously* be different
  - Does not mean *developers* are creating those file

Fault Distribution Without Context

- Fault distribution should be normalized:
  - Obviously more graphics errors than command-line utility
  - Number of faults *compared to* amount of code is more important
- ~20% of faults fall into "unknown" category
  - Shows fundamental misunderstanding of domain
- Overall data model needs a lot of work

Perspectives from a Game Developer

What does a professional in the field think?

Unrealistic View of "Game" Projects

- Many *games* are simulation engines or niche prototypes
- Most real-world games rely on off-the-shelf engines (Unreal, Unity, etc.)
- Study more about hobbyist C++ projects than real-world game development

Lack of understanding of team dynamics

- No distinction between game designer and game developer
  - Many multimedia assets created by designers
  - However, they sometimes may be created by developers!
  - Studying distinction is important
- Files is not a good measure of proportionality of work
  - Many asset files would take a few minutes to create
  - A system file might take many days of iteration
- An interview would fix these issues

Reductive Analysis

- Paper treats games like ordinary applications with extra art assets
- However, a game is a system:
  - Engine is always running
  - Constant interaction with assets, physics, AI
- Complexity of interactions is ignored in paper

Rating: 1.5

Feels like it was written by someone who's never played a video game.

Future Work

How can we extend this?

Further Categorization

- Online vs single-player games
- Categorization by genre / POV:
  - Sandbox vs. Open World vs. Linear
  - Platformer vs. First Person vs. Third Person
- Differences in Game Engines:
  - Unreal vs. Godot vs. Unity

Discussion Points

- What do ☆YOU☆ think?
- Should SE research differentiate between game designers and game developers?
- How could we model the system-level architecture of a game to reflect its runtime interactions?
- Do game developers need a different curriculum than traditional software engineers?