Dissecting The Postmortem: Lessons Learned From Two Years Of Game Development Self-Reportage

By Ara Shirinian

[In this Game Developer magazine article, designer Ara Shirinian (The Red Star, uDraw) takes a look at two years' worth of magazine postmortems to mine them for common data points -- examining both what went right and what went wrong how often, and why.]

After having participated in numerous game projects throughout my career as a designer, including many failures and successes, I noticed that certain types of development mistakes appeared to recur with surprising frequency. Was this just another Twilight Zone-inspired idiosyncrasy of my career? Or is there something more going on here?

I imagined a development hell where throngs of teams all ran into the same pitfalls over and over, without knowledge of what they were doing wrong, and without the realization that anyone else might be making similar mistakes.

As developers, we have our own career histories to depend on for knowledge and experience about the rights and wrongs in game development. Beyond that, we can only rely on other developers' willingness to relate their own experiences, mistakes, and solutions to us.

To that end, things like postmortems, conferences, and just plain open discussion amongst developers are great tools.

But these are only vignettes in a sense, and often highly contextually dependent. For example a team's troubles with Lua integration are not helpful if you never use Lua.

Beyond that, I was curious whether there was a bigger picture, what it looked like, and if the results could help us learn something new about game development in general. That, in a nutshell, is the motivation behind this analysis.

Collecting Data about Game Development

Without any expectations about whether I would find any interesting trends or commonalities across game development projects (or whether I'd even find anything coherent in the first place), I decided that Game Developer's extensive history of postmortems was the best and most consistent source of data.

Conveniently, postmortems written for Game Developer all share a similar structure: the developer writing the postmortem is required to select and illustrate five things that went right during the project, and five things that went wrong. Despite the enormous range of project types and sizes, this made it relatively straightforward to collect and organize data about the good and bad things that were reported about projects.

The data set for this analysis consists of 24 successive postmortems published in Game Developer, covering a period of two years, from articles that were published from February 2008 to January 2010.

Collecting Data about Common Issues

Before data collection began, I defined a set of various issues or situations that I thought would be interesting to track across all postmortems. For example, whether the postmortem mentioned using scrum or some agile process, whether a successful experience with outsourcing was reported, and so on.

For each of these items, each postmortem was scoured for mention of that issue or situation. I didn't just search for specific words; collecting the data required a lot of extensive re-reading to ensure that if the postmortem was counted in a certain category, there would be no question about its inclusion.

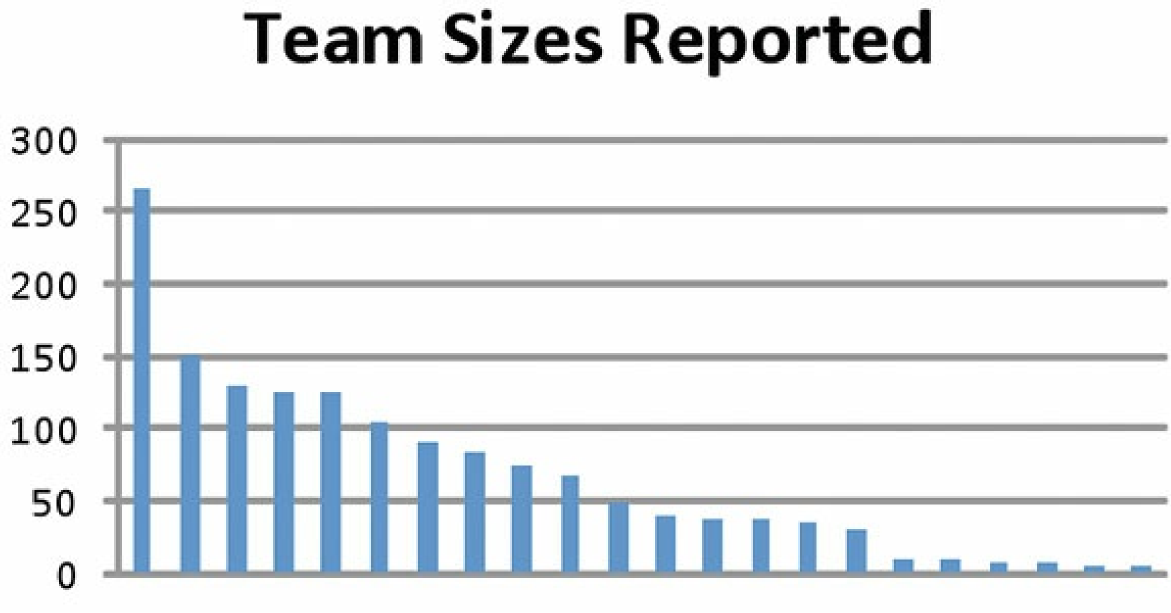
Results Part 1: Postmortem Metrics

Here are some metrics that characterize the projects whose postmortems were included in this analysis.

Team Sizes Reported

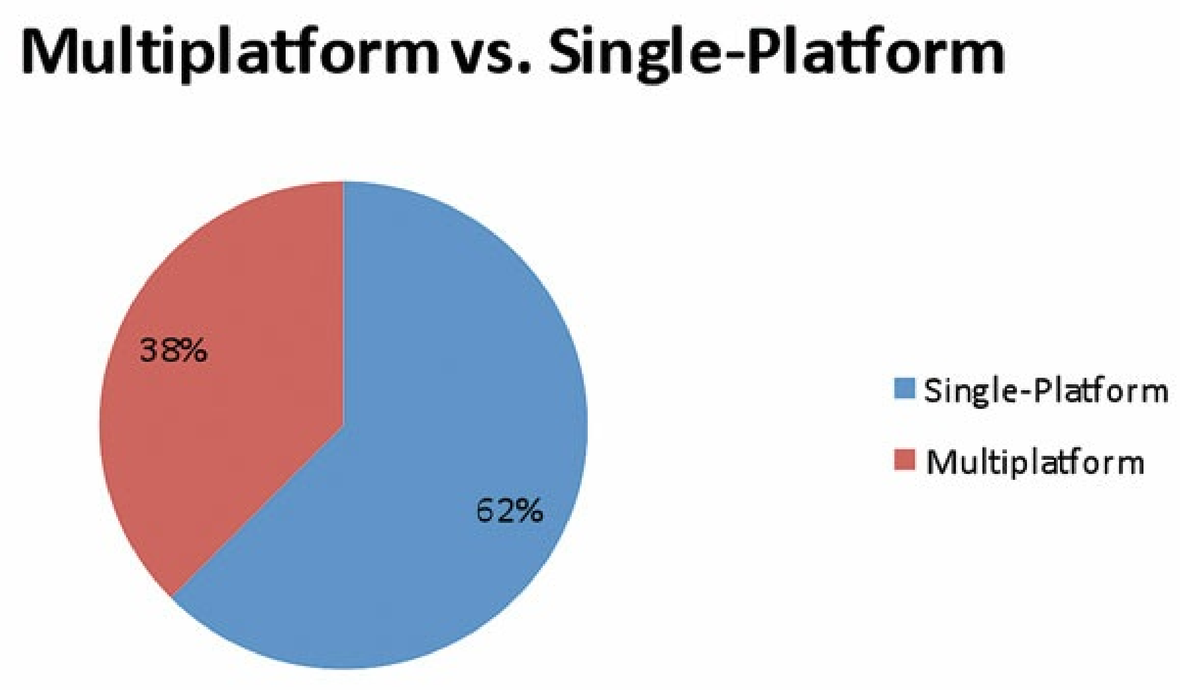
The largest project was *Far Cry 2,* boasting a team of 265 members. The smallest projects were *Age of Booty* and *n*+, both reporting just five team members. Four postmortems reported a range of team size; of those, the maximum was recorded.

Two postmortems did not report a team size and were excluded from this section. Team sizes tended to fall into four clumps, with the smallest teams consisting of five to 10 members. The next five largest teams consisted of 30 to 40 members. There was another clump representing 68 to 90 members, and six projects reported over 100 team members.



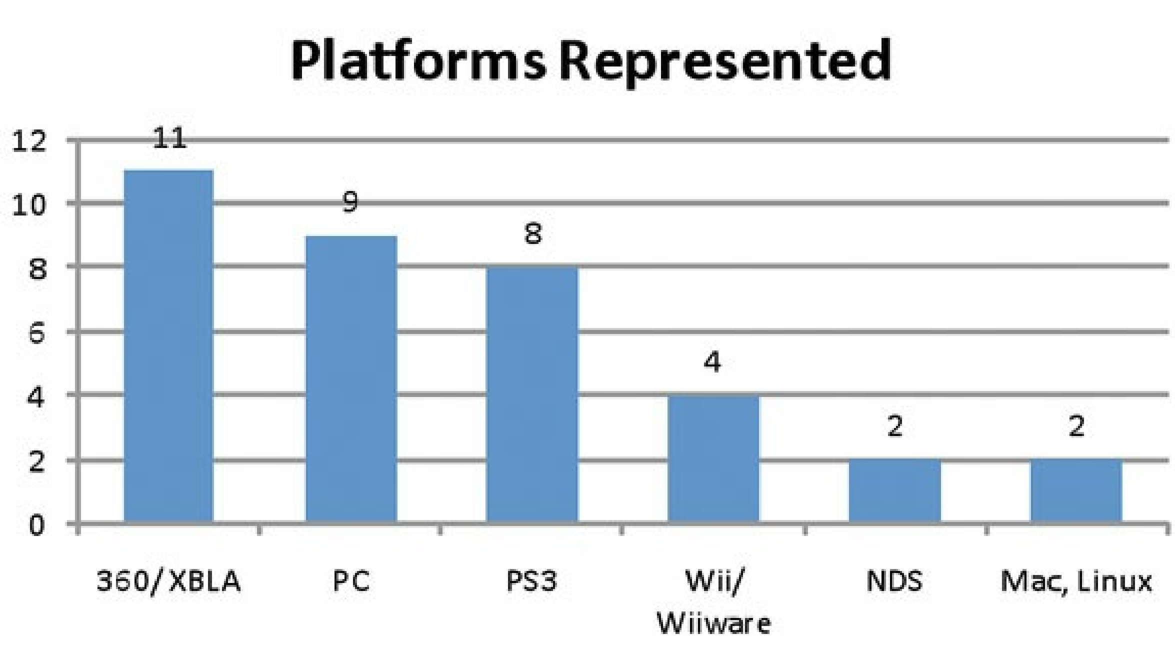
Multiplatform vs. Single-platform

Slightly more than one third of the projects were released on multiple platforms.



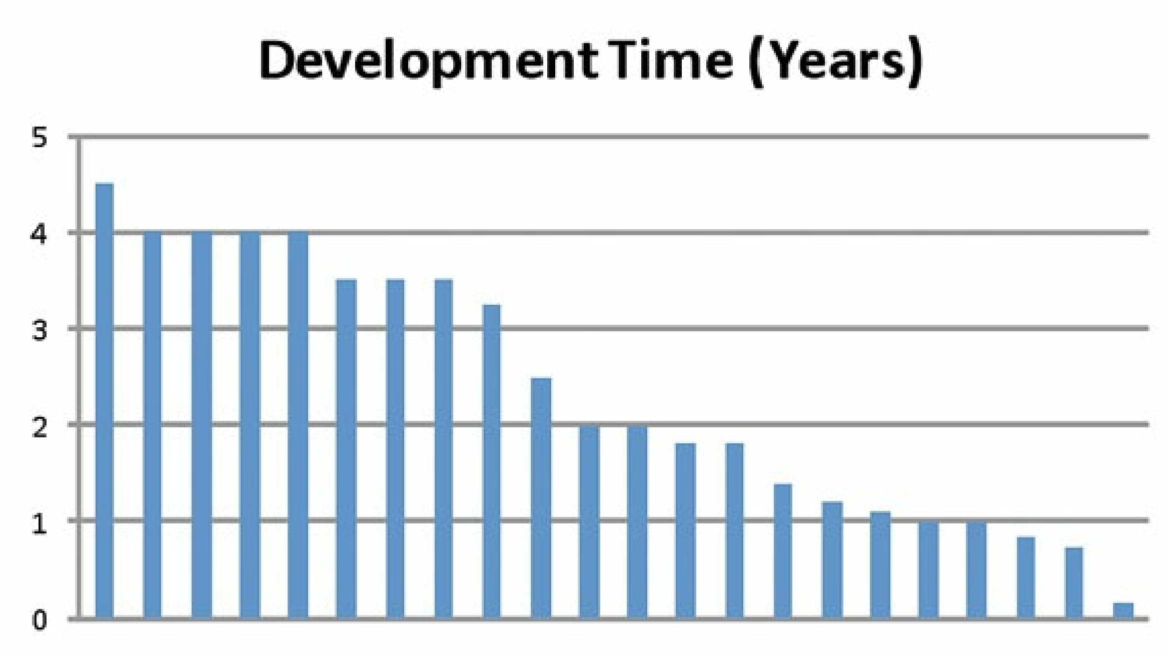
Platforms Represented

Multiplatform projects are counted once for each platform they were released on (there were no PSN games represented in this selection of postmortems).



Development Time

The average development time for projects was 2.4 years. Two projects did not report a development time and were excluded from this section. Interestingly, development time also seems to clump at yearly or half-year marks, although this is probably an artifact of how durations were reported. There is also a large jump between the two-year mark and the three and a half-year mark.



Use of Contractors or Outsourcing

Eleven of 24 postmortems reported utilizing contractors in the project. Only 7, or about 29 percent, reported outsourcing work to separate companies.

Caveats

It's important to mention some qualifications that are inherent in the nature of postmortems intended for public consumption. The first thing that's called into question is the sincerity of the report itself. Is the writer deliberately holding back certain information that would be considered "bad PR" for the company to reveal?

Some writers appear to be more forthright about dramatically bad outcomes of their projects than others. It seems safe to say that there is likely some amount of this type of informational restraint, but of course we can never really know who is writing with candor and who has one or more fingers tied behind their backs for whatever reason.

Fortunately, I have been pleasantly surprised about the willingness of many authors to expound on some quite disastrous situations in their postmortems.

The second major qualification is one of completeness of information. Because postmortem authors are free to pick their favorite five "bad things" and "good things" about their projects, and because each postmortem is written independently, there is no guarantee that any given topic will be mentioned at all in any given postmortem.

What's more, the absence of mention of any given topic gives us no information about that topic one way or the other. For example, if there is no mention of "working crunch" in any way, we still don't know if the team worked crunch or not.

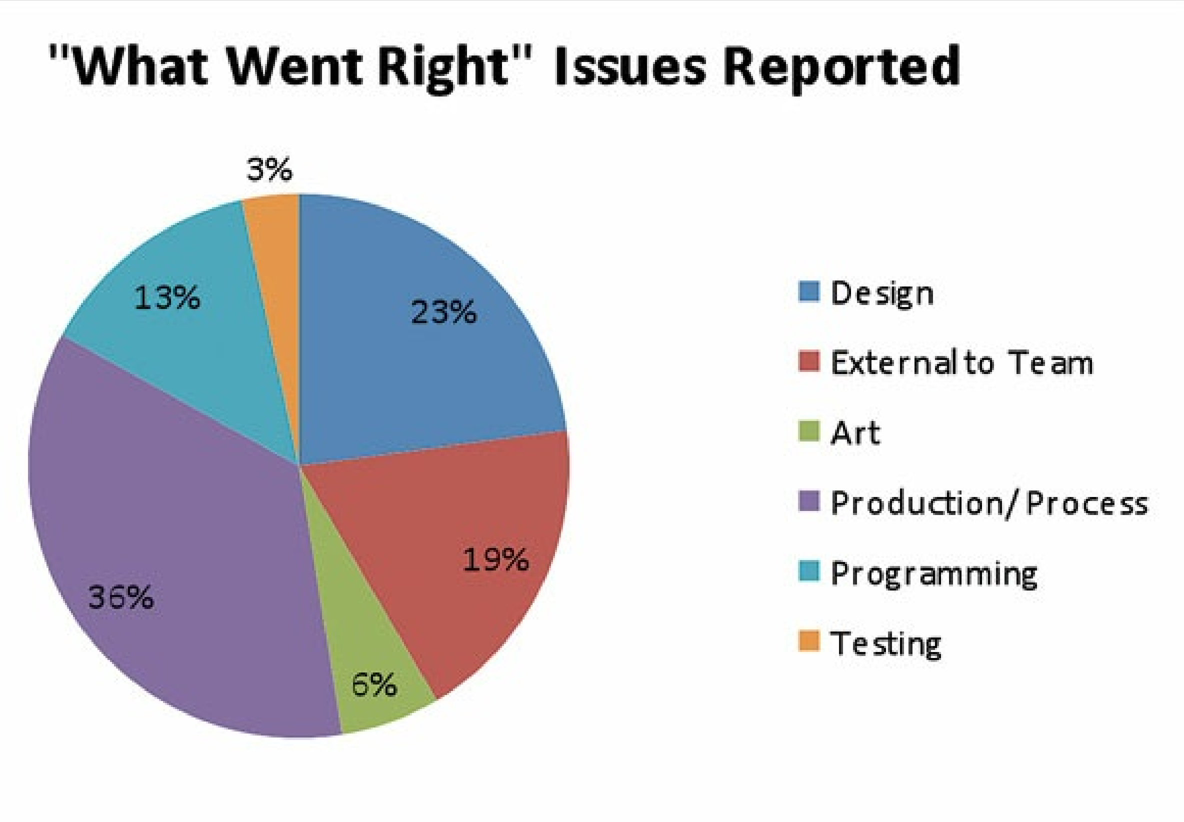
Either way, if the team did work crunch and it wasn't mentioned, we wouldn't even know if it was a brief finish-line crunch with everyone cheering to complete their most polished work, or a morale-obliterating death march with far-reaching casualties.

In short, we're at the mercy of the collective of postmortem authors, as the quality of information presented here is a function of their ability to present honest and complete information about their own projects.

Results Part 2: How Things Went Right and Wrong

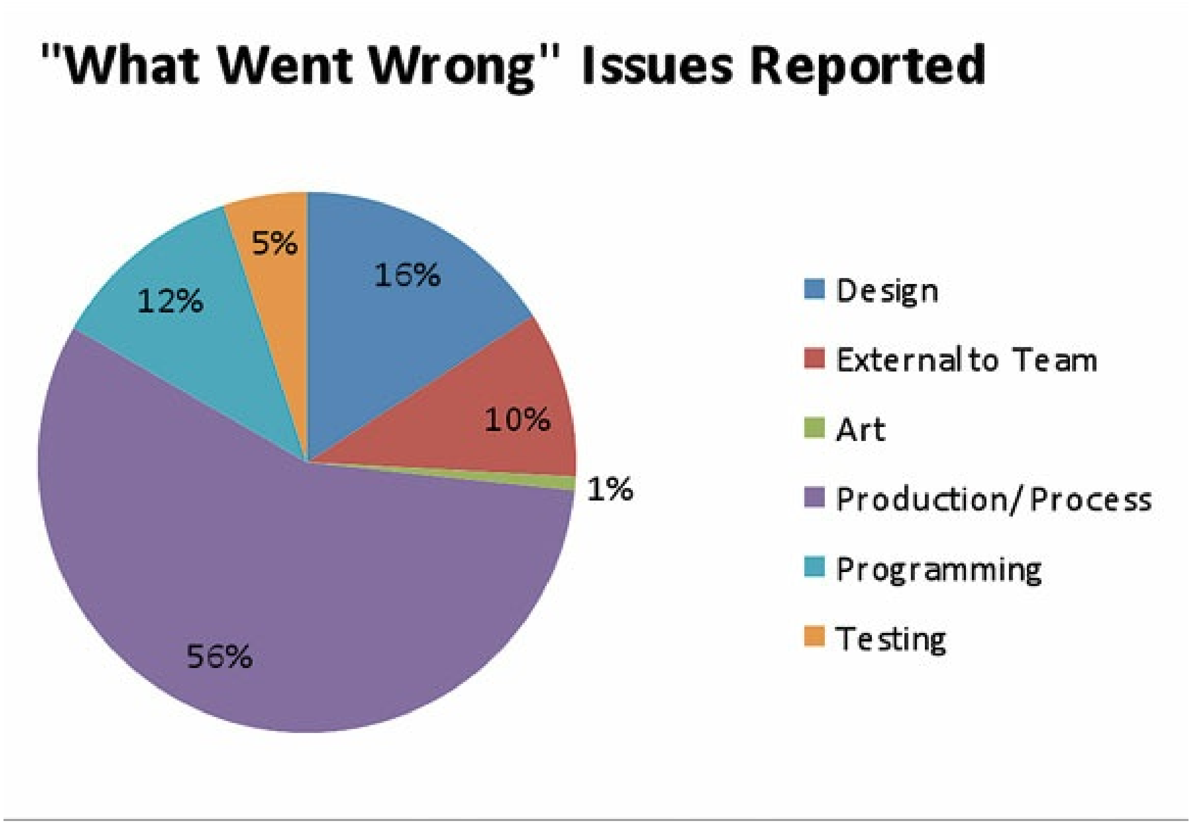
As described in the Methodology section, each "thing that went right" and "thing that went wrong" that was reported in each postmortem was classified into one of six categories. Each of these categories generally corresponds to a major discipline involved in game development. In total, this supplied 240 data points: 120 "rights" and 120 "wrongs."

If we look at the all of "right" issues under this classification, we see some interesting results:



What's going on in this pie chart? When things go well, production is most often credited for it, with design running second place and issues external to the team a close third. Art issues appear to take a disproportionately small slice of responsibility for the good things that happen on projects, and testing gets the least amount of representation.

Now, let's take a look at how issues were classified when things went wrong.



Immediately we see that production issues take an enormous share of the problems when things go wrong. What this means is that when things go wrong on a project, most often it's not inherently because of design mistakes, art mistakes, or coding mistakes, but problems in the way the process of development is prioritized, conducted, and managed.

In general, this seems to indicate that development teams are just much worse at planning, coordinating, and conducting the work required to produce a game as a whole, more than anything else. I think this finding will resonate with a lot of developers -- poorly managed projects unfortunately appear to be much more common than well-managed ones. Mind you, our sample of postmortems was all games that shipped, and at least by that account can be considered successfully managed projects.

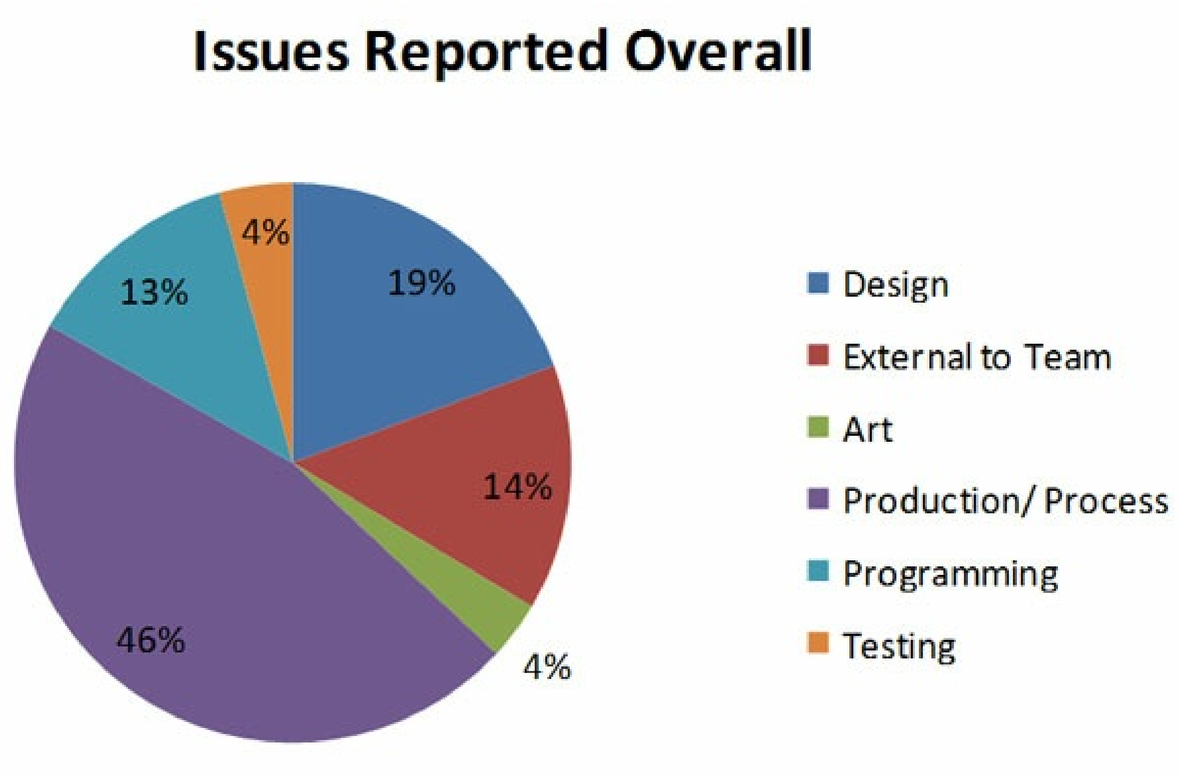
There were a total of 68 individual "what went wrong" issues classified under the "production" category. Within this group, the most common issues were related to scope, feature creep, and resource problems; this accounted for 16, or about 23 percent of production problems. The second-largest subset of problems within production was team-communication related, which accounted for eight, or slightly more than 10 percent of the issues. Six issues involved various critical events happening later than they should have in the development cycle.

While production increased dramatically in the "wrongs" versus the "rights," all other issue types decreased in frequency of reports (except for testing, which nearly doubled, but was still a small number).

It should be noted that the "wrong" issues related to testing were all about planning, managing, and being unprepared for the logistical burdens of the testing process, and not related to testers doing a bad job. For these reasons, most of the testing-classified items could arguably also be included in the production category instead.

Another interesting statistic is that programming issues decreased only by 1 percent, while design, art, and external categories decreased by 7 percent, 5 percent, and 9 percent respectively. This also seems to make sense, as when things do go wrong, technical issues can easily have much more salient effects on the health of a project. However, design issues are still the second most represented type, which also seems to indicate that designers may be creating more problems for projects than the other traditional disciplines.

Here's what the distribution of issues looks like if we combine all the rights and wrongs together:



This aggregate graph might be best understood as the amount of overall impact or significance a certain discipline can make on the smooth-running of a project's development. As expected, production is almost half of the pie, with design taking the number two spot again, and coding taking roughly equal weight to external issues. Note that since postmortems are explicitly about the process of game creation, this is not an indication of what is most important to making a game successful or artistic -- just to completing it.

Results Part 3: Common Themes

Are Small Teams Important?

Seven, or just under a third of postmortems, reported a "small team" as a positive aspect of the project. Interestingly, three of those projects actually consisted of 30 to 40 member teams, with the other four being reported from the smallest teams.

Developer-Publisher Relationships

Fifteen out of the 24 projects were developed under a typical developer-publisher system, where the entity publishing the end product is not the same as the entity developing it. Projects that were developed at a separate studio entity, even if owned by the publisher, were included in the count of 15.

Of those postmortems, we found that 40 percent, or 6 out of 15, reported a positive developer-publisher relationship. Three out of 15, or 20 percent, reported problems with their relationship with the publisher, while the remaining 40 percent did not report anything good or bad about their publisher.

Planning For the Team

Five out of 24, or 21 percent, reported deliberately planning their game with respect to the capability or expertise of the team. All of the projects that were in this category completed their development in under 2 years, and the majority of development cycles were completed in a year or less. These teams were not all "small," however -- two of the projects had 37- and 50-person teams respectively.

Crunch, Time Extensions, and Scope

Nine projects, or 38 percent, reported receiving a time extension to finish their project. A few also reported the length of the time extension, which was anywhere from "a few extra weeks" to a whopping total extension of 17 months (*Brütal Legend*).

The same number also reported some manner of a crunch period during their project, although few actually reported the duration of the crunch, which varied anywhere from six months to "almost a full year" to "always in crunch mode" (*My Life as a King*).

Most interestingly, 17 postmortems, or 71 percent, reported scope problems where there was either not enough time or resources to complete the game, or there was too much design that had to be cut, often repeatedly throughout the project.

This is the most obvious trend across the postmortems: teams are consistently underestimating the required amount of time and resources needed to create their titles. However, it's unclear how many of these scope problems are due to pressure to complete the game under a convenient timeframe for the publisher, and how many are the result of poor estimating from the developer.

Along similar lines, half of all projects reported making last-minute or exceptionally late feature additions or changes.

Development Agility

Five projects, or 21 percent, reported using some flavor of scrum or other explicitly agile development process.

Nine projects reported using a deliberately flexible design approach to at least one of the game elements they described.

Eighteen, or 75 percent of projects, reported iteration or rapid prototyping as a valuable component to development.

Conversely, a surprising 29 percent, (7 postmortems) actually reported that they committed to an inflexible design or plan partway through development. Five of those projects were 2.5 year-long cycles or longer.

Management and Communication

Eleven projects, or roughly half, made mention of some variety of team management problems, which included problems like overwork (separate of crunch mentions), lack of focus, problems with staffing, and morale issues.

Nine projects, or 38 percent, reported some kind of significant problem with communication across teammates, which included deliberate refusal of team members to communicate with each other, confusion about game vision and direction, and the ineffectiveness of leaders to adequately convey changes to the rest of the team.

Pipeline Problems

Nine postmortems also mentioned explicit problems with asset pipelines. These include pipelines making work unusually time-consuming or painful, not coming online early enough in the project, or otherwise not adequately supporting the actual work process of team members.

So How Did the Outsourcing Go?

Of the seven postmortems that reported outsourcing some of their work, three reported an overall successful experience, and four reported problems of some kind. The variety of problems included a lack of preparation, starting the process too early with respect to the overall cycle, underestimating the amount of management involved, and hiring a company to perform work that they did not have sufficient expertise in.

Conclusions

From looking at postmortems over the past two years and considering the data presented here, it would seem that the biggest takeaway is the importance of managing the development process itself.

Of course, we're not talking about just the "existence" of management, but good, methodical, careful management that keeps the interest of the game and the team at the forefront. Unlike some of the other disciplines, project and team management, when done well, may be able to compensate for inadequacies in other areas.

However, when executed poorly, it seems as though project management has the potential to unravel and destroy even the best creative work.

Beyond that, one of the most common and disturbing trends is the inability for game development projects to be properly scoped and scheduled.

We developers are constantly fighting a battle (that we aren't winning as often as we should be) between the resources we have available and the end product we are attempting to realize -- whether those factors are imposed by ourselves or by those paying our checks.

Methodology and Definitions

Data from the 24 postmortems was collected and organized in two different ways. These were the methods undertaken for each.

Organizing and Summarizing Things That Went Right and Wrong

The body of each postmortem consists of five expositions about things that went right and five about things that went wrong. Each of these atomic "things" was then classified into one of the following seven categories, defined as follows:

Design. Relating to game design, level design, gameplay and rule designs, and overall game vision external to team. This category covers all situations and decisions that were made that are clearly external to the direct team and development process, including business logistics, hiring, partnerships, funding, marketing, studio-wide decisions, and so on.

Art. Relating to art decisions, direction, or specific art processes.

Production/ Process. This relates to scheduling, work prioritization, production methodologies, development plans and processes, scope, team morale, team communication, team assignment, team management, and so on.

Programming. This category covers all technical issues, including tools, technology implementation, and anything code-related.

Testing. This category includes all traditional QA functions, including bug testing, gameplay/usability testing, localization, gameplay data collection, and metrics.

Other. There were only two items that did not fit any of the above categories, and were omitted from the results. Together they represented less than 0.5 percent of the data overall. One was a "right" item about sound direction (interestingly, this was the only item that could be categorized as relating to audio). The other was a "right" about obtaining nice office space.

A couple of specific types of challenges were encountered in categorizing each reported item from the postmortems. First, sometimes the original title of the item was not a good characterization of what was explained. This issue was avoided by categorizing the item based on the content of the explanations, not on how the explanations were titled. Second, sometimes the discussion on a certain item would have crossover with or would otherwise veer into other subjects.

For example, an item ultimately classified as "design" might make mention of some production process or something about art, or vice versa. In these cases, a good faith effort was made to select a category based on what the primary gist of the entire explanation appeared to be about.

Game Developer Magazine postmortems featured: *Age of Booty, Aion, Akrasia, Brütal Legend, The Conduit, Darksiders, Deadly Creatures, Far Cry 2, Final Fantasy Crystal Chronicles: My Life As a King, Free Realms, Golden Axe: Beast Rider, Infamous, Little Big Planet, The Maw, n+, Penny Arcade Adventures: On the Rain-Slick Precipice of Darkness, Saints Row 2, Scribblenauts, Tales of Monkey Island, Tomb Raider: Underworld, Trials HD, Uncharted 2: Among Thieves, Wizard 101, The World Ends With You*

Return to the full version of this article