Putting the hours on fun, not manual labor: Incorporating DevOps in the Video Game Industry

William Skagerstöm
wska@kth.se

Gabriel Chang
gchang@kth.com

1 Introduction

The video game entertainment industry has grown immensely and simultaneously it has become increasingly more complex to create high quality games. Creating games can now cost upwards of millions of dollars [1] and the importance of being first on the market is crucial. Due to the high competition and market demand, the need to reduce development time is immense. This is where DevOps can be of use. DevOps is about rapid and reliable development, with a focus on streamlining the cooperation between the development teams and operations teams and the automation of tasks [2]. With the DevOps practices such as Continuous Integration, Continuous Delivery, and Continuous Deployment it is possible to reduce development time.

In this essay, we will explore the current state of DevOps in video game development. We will also look at how two game development companies have incorporated DevOps practices into their workflow and how doing this affected them.

2 Background

2.1 DevOps

There are many ways to define DevOps. The term comes from 'development' and 'operation', and can often be regarded as a set of operating principles that focus on breaking down the barrier between development and operation teams [3]. DevOps recognizes that there is a mutually beneficial relationship between them and by streamlining the process between them you can achieve faster and more reliable development cycles, which in turn results in faster delivery and a higher quality product. By following DevOps principles you can reduce miscommunication problems and accelerate problem resolution [2].

A big focus in DevOps is to use tools and automation in order to achieve the above. Tools like *Jenkins* are commonly used to automate many of the processes in software development and testing. Automating tasks is crucial to the three cornerstones of DevOps, namely: Continuous Integration, Continuous Delivery, and Continuous Deployment. With Continuous Integration, for example, automated builds and tests are run when a change is integrated in order to verify that the change upholds certain requirements. Continuous Deployment goes a step further and involves deploying each verified build to production [4].

2.2 DevOps in video game development

Video games are an inherently complex product and the industry as a whole is very competitive and somewhat unforgiving. A common trend within the video game industry is to take the core concepts of DevOps methodologies such as continuous integration and testing, but also extends this by adding concepts such as artificial intelligence and dedicated control systems that are tailored for video game development processes [5]. The general software development pipeline for video games can typically be defined as three separate production stages [6]:

One of the fundamental differences between ordinary software development and video game development is the overarching purpose of the software itself. Software is typically written with a specific set of requirements in mind to fulfill a business need. Video games as an entertainment, however, has a much more ambiguous need: Entertainment & fun.

This is much harder to write down into a project specification since what this term means depends on the target audience of your product. This places emphasis on the *pre-production* stage of development, with many products simply not progressing into a *production* stage before being terminated due to failing said core requirement. Since this is a large and very difficult requirement to approach, DevOps has been a widespread solution which allows developers to focus more on these specific areas of development rather than spend time on things which are not as critical as the primary entertainment loop of the game.

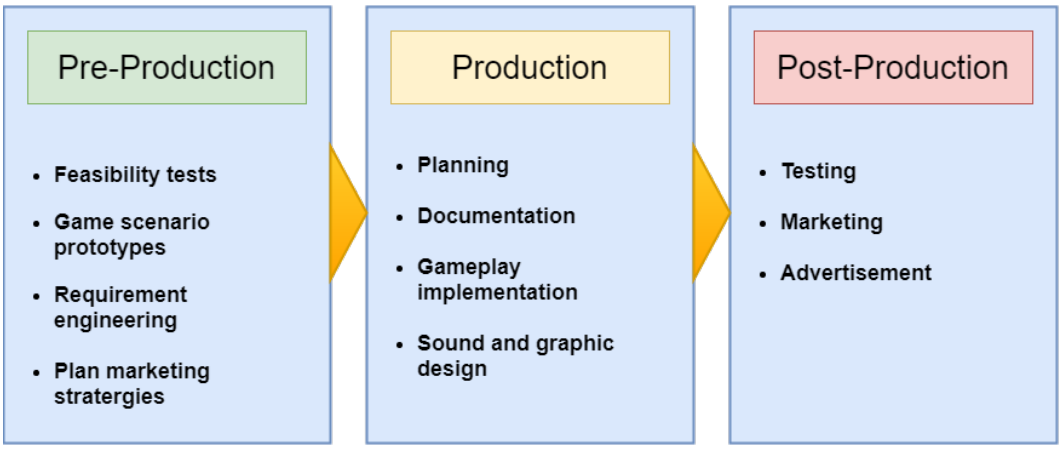

Figure 1: Video game development production stages

3 Sample cases of DevOps adoption

With the principles of video game development in mind, we will in this section look at how two game development companies adopted DevOps methodologies in order to achieve their purpose.

3.1 Electronic Arts

Electronic Arts (EA) is major video game development and publisher company based in *California, USA*, with major headquarters across the globe. Founded in 1982 by *Trip Hawkins*, it is as of May 2020 the second largest video-game corporation in the western world by market capitalization.

*EA*s transition to modern DevOps techniques was initially driven by the increasing complexity of their products and the quantity of different titles that were being developed simultaneously. Due to the sheer popularity of their products, it became apparent that any effort conducted by the dedicated quality-assurance team would be immediately dwarfed by the millions of players that would put their hands of their software when a new product released.

The general goal was thus to try to shift the efforts conducted by the QA and development team so that they may instead focus on ensuring the game design is fun and engaging, rather than doing tasks such as finding and squashing bugs. The way EA measures the efficiency of their integration of DevOps into their production pipeline is actually a finite discrete heuristic measurement of *years of effort saved*.

This initially started with about *20 years* during their initial adoption of DevOps during 2006-2010, but surpassed *1000* years in their third evolution in 2016 [7].

EAs initial transition to utilizing DevOps came in the form of a realization that they needed to make some adjustments to the way they deployed builds. Just before the adoption of their "DevOps 1.0", they had a dedicated developer that was charged with operating the build system, which ended up being quite a large and time-consuming task. *Nixdorf* noted that this person was usually the *"first person in"* and *"last person out"*. If this developer was ever moved around, this responsibility would simply to fall someone else.

They set out to break this pattern by trying to adopt some form of CI/CD methodologies, but before adopting modern tools they built and maintained their own system for just that, which ended up being overly complex and difficult to both maintain and learn for new developers.

Around the 2010 mark came some major breakthroughs when they shifted their infrastructure to a virtual platform and started adopting standardized industry tools such as *Jenkins* instead of their own custom software.

2015 marked the point where they set out to achieve their third revision, "DevOps 3.0". Compared to their previous iterations, this was first time where the entire development team was actually aware of

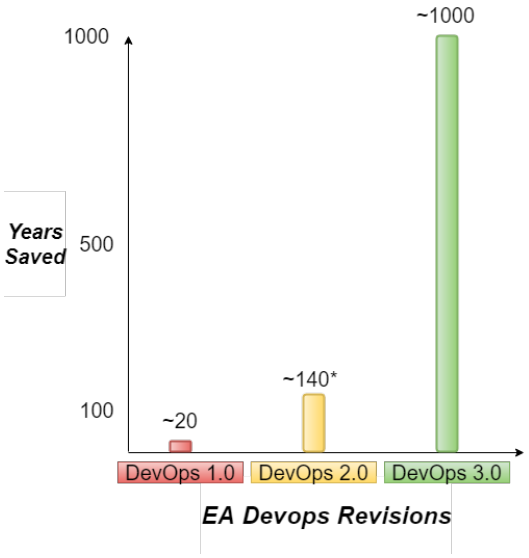

Figure 2: EAs years of of effort saved by their internal DevOps adaption revisions.
* While the years of effort saved for DevOps v2 is not explicitly mentioned, using exponential interpolation between the given numbers 20 and 1000 gives ≈ 140.

DevOps and its general methodologies, compared to how they previously were driven primarily by specific needs in order to streamline their software production procedures.

They had the mindset that they wanted to become *"active participants in the larger DevOps community"*, and wanted to catch up a bit. One of the major steps in their 3.0 shift was the embrace of cloud services and open-source software(both in utilizing it but also contributing to it).

Shortly after that they also started utilizing other solutions provided by *Cloudbees* in order to greatly reduce the developer burdens. *Nixdorf* notes that, while they made many mistakes on the journey to truly adopting modern DevOps methodologies, but concluded that all these mistakes were also lessons that the team learned from.

One of the key takeaways mentioned from EAs DevOps transition is the realization that the goal of this shift in methodology was not just to save time for themselves or the company, but instead about saving time for the developers and the QA teams so that they may spend said time improving the quality and authenticity of their games instead of other less important tasks [7].

3.2 Butterscotch Shenanigans

Butterscotch Shenanigans is an independent game development studio that was founded in 2012 by *Seth Coster*, *Sam Coster*, and *Adam Coster*. The studio currently consists of 6 employees [8]. It was during the development of the game *Crashlands(2016)* as the studio noticed a need for a better streamlined development process. Due to not having a clear and streamlined development cycle, this led to many game-breaking bugs that needed to be fixed. This was both physically and mentally taxing for the developers, but also demanded additional work hours from the team. For their next game, the team wanted to adopt DevOps practices.

The ultimate goal for *Butterscotch Shenanigans* was to have Continuous Deployment (CD) for their development. The benefits of having this was to: *"detect problems and fix them immediately"*, have *"fewer catastrophic failures"*, make it *"easier to identify problems where they emerge"* and *"quickly fix issues with deployments"* [9]. But their deployment process, seen in 3, was far from this goal. [9]

In order to achieve CD, they decided to reduce *waste*. According to *Coaster waste* is *"anything that the team is doing that is not delivering value to the customer"* [10]. To help reduce *waste*, the team needed to automate all the steps from implementing changes to sending builds to QA. The team built the automation tool *GamePipe*, which automated the process for writing patch notes, compilation of test builds and deployment of builds to QA testers. By deploying *GamePipe*, they

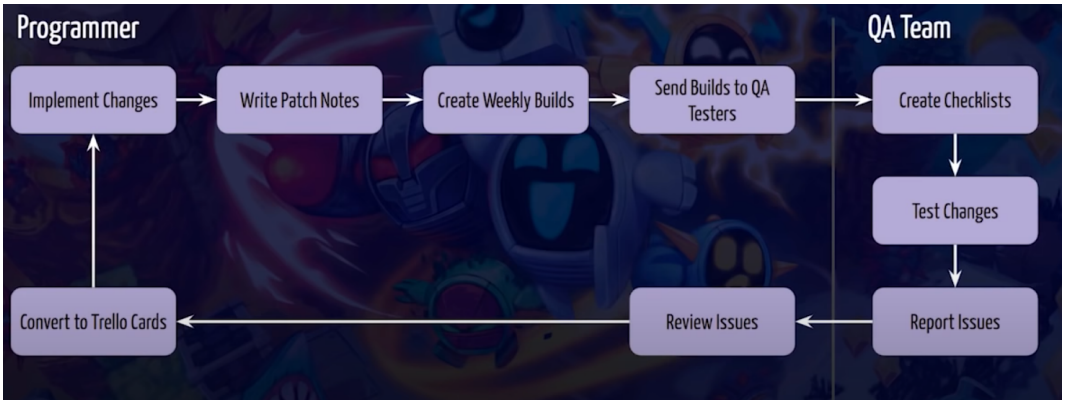

Figure 3: Previous deployment and QA pipeline of *Butterscotch Shenanigans* [9]

reduced time consuming manual work, for example In the past, *Coaster* had to manually send test builds to QA, which could only be done once a week. Now, when fully automated, every change would automatically get deployed to QA. In other words they went from weekly testing to continuous testing and the builds were also more stable. [9]

The next step was to automate the QA pipeline in order to improve the feedback loop process. Their current method of doing feedback and reporting issues included manually creating *Trello* cards to visualize the feedback. New tools were built. One used the *Trello API* so that the feedback received from the QA team would be automatically converted into *Trello* cards [11]. Their complete production pipeline with DevOps practices can be seen in 4. [9]

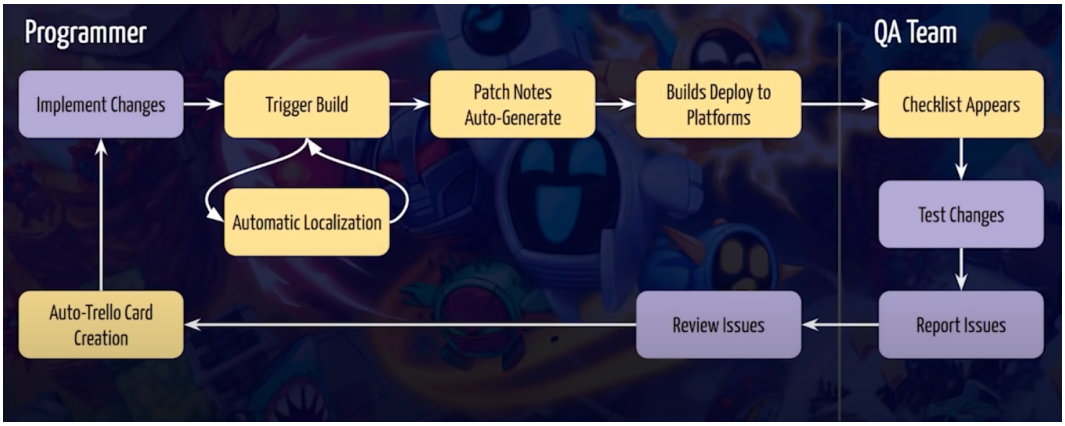

Figure 4: Current deployment and QA pipeline of *Butterscotch Shenanigans* with DevOps practices [9]

Coaster concludes that *"Stress and high turnover in game development is the result of bad practices"* and that by adopting DevOps practices, *Butterscotch Shenanigans* has been able to *"reboot itself into a studio built around efficiency"* [10].

4 Discussion & Conclusion

The insight provided by the material highlights greatly the fact that DevOps as a tool was something that was adopted not only to streamline software production within the game industry, but also for the ability to reallocate time in order to spend it on more important parts. Video games are ultimately an entertainment medium, whose primary purpose is to provide entertainment, fun and personal satisfaction. By freeing up precious hours through the utilization of modern DevOps methodologies, this allows video game companies to instead put those hours where they can provide the most value: Refining the core of the game and making it more enjoyable. And in such a competitive and fast-paced industry, this becomes precious and invaluable.

References

[1] Michael Smith. *How Much Does it Cost to Setup a Gaming Company and Make a Game?* June 2018. URL: https://www.gamasutra.com/blogs/MichaelSmith/20180604/319306/How_Much_Does_it_Cost_to_Setup_a_Gaming_Company_and_Make_a_Game.php.

[2] C. Ebert et al. "DevOps". In: *IEEE Software* 33.3 (2016), pp. 94–100. DOI: 10.1109/MS.2016.68.

[3] Arshad Ghole. *Game Development Future — How DevOps Is Changing it!* Nov. 2019. URL: https://askwonder.com/research/devops-and-game-development-0vefl5x8m.

[4] Atlassian. *Continuous integration vs. continuous delivery vs. continuous deployment*. URL: https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment.

[5] Hely Marleena. *DevOps vs. GameDev Software Engineering*. Oct. 2018. URL: https://medium.com/@helymarleena/devops-vs-gamedev-software-engineering-80b4b73bdc81.

[6] Saiqa Aleem, Luiz Fernando Capretz, and Faheem Ahmed. "Game development software engineering process life cycle: a systematic review". In: *Journal of Software Engineering Research and Development* 4.1 (2016), pp. 1–30.

[7] Josh Nixdorf. *The Electronic Arts DevOps Tale*. Mar. 2018. URL: https://www.cloudbees.com/blog/electronic-arts-devops-tale.

[8] Butterscotch Shenanigans. *About: Butterscotch Shenanigans*. URL: https://www.bscotch.net/about.

[9] Gamasutra. *Video: Less stressful game development via DevOps*. Apr. 2020. URL: https://www.gamasutra.com/view/news/361350/Video_Less_stressful_game_development_via_DevOps.php.

[10] Sam Desatoff. *Game development crunch: Is DevOps the answer?* Mar. 2020. URL: https://gamedaily.biz/article/1638/game-development-crunch-is-devops-the-answer.

[11] Nancy P. *DevOps & Game Development*. Jan. 2021. URL: https://askwonder.com/research/devops-and-game-development-0vefl5x8m.