DEVOPS METHODOLOGY IN GAME DEVELOPMENT WITH UNITY3D

Gabriel Barroso da Silva Lima, Cristina Souza de Araújo, Luis Cuevas Rodriguez,
Clairon Lima Pinheiro and Jucimar Maia da Silva Junior
Escola Superior de Tecnologia – Universidade do Estado do Amazonas (UEA), Brazil

ABSTRACT

The DevOps methodology comes to facilitate the interaction between developers and operations, automating many of the steps necessary to generate the final version of a software application. It is a current trend in software development. However, today, many game projects are developed using procedures that cause delays and crunches. The games industry could benefit from incorporating these models, and their associated tools, into the complex process of developing a game. In this article, a study of the application of integration models in the game development process and a software-tools evaluation available in the market for this purpose is introduced. In addition, an initial DevOps methodology experience is presented, serving as a reference to other similar projects, thereby demonstrating the real possibilities of using DevOps in game development.

KEYWORDS

DevOps, Games, Game Development

1. INTRODUCTION

Every software project in its life cycle goes through different analysis, design, implementation and testing processes. It is common for a project to guarantee its quality by having to perform hundreds of different tests. In a game, a failure can not only be caused by an error in the code, but also by a corrupt audio file or a 2D / 3D model. In the game creation process, in addition to quality controls and normal UX tests, play testing is required: checking if the game is fun, if has a fair difficulty, if has a good progression.

At the moment, AAA games have hundreds of people helping on the development, advanced logistics and dozens (or even hundreds) of millions of dollars of production costs. With scopes getting astronomical scales, even some AAA studios cannot handle it. Even before the pandemic, big games are already getting delayed [Jones 2020]. One of the areas game development still lack of infrastructure. Much of the time on a game production is wasted on motion, task switching, waiting, defected files or even unnecessary processes and features. All this done in detrimental to the integrity and reliability of the process, with high costs and, most of the time, manually. These problems can be avoided by using the DevOps methodology.

The DevOps tools and philosophies combine practices capable of increasing the capacity of an organization to deliver applications faster than traditional software development processes. DevOps can be applied in more areas, one of them is game development and, using Unity in the development, the process can be drastically optimized and problems that causes time lost, useless tasks and crunches.

The article presents the following structure: after this first Introduction section, a general review of the DevOps methodology, its phases and its application in software development is presented, including a summary of the main existing tools for each phase. Some of the main features of game development are detailed in section 2.3, particularly using the Unity game engine. Finally, the considerations made by the authors about the possibilities of using DevOps methodology for the development of games is presented in session 3, where a discussion of the possible tools to be used in each phase is carried out and is described the experience of a development team, using these tools.

2. DEVOPS IN GAME DEVELOPMENT

Emerged in the context of agile methodologies, DevOps responds to the need experienced by the technology sector to provide a faster response to the implementation and operation of applications.

DevOps is a work methodology that, based on code development through the use of new tools and practices, reduces the traditional distance between programming and systems technicians. This new collaboration approach that is DevOps allows teams to work more closely, bringing greater agility to the business and notable increases in productivity. It is also, therefore, a cultural change that allows companies to accelerate the life cycle of their applications.

2.1 Software Development using DevOps

The main purpose of the DevOps is to create a continuous development and improvement environment with small upgrades. This is achieved through a series of processes that extends from the base development until post production updates [Virmani 2015]. The DevOps lifecycle can be divided in phases: continuous planning, continuous integration continuous deployment, continuous testing and continuous monitoring. These phases are necessary to have an operational environment with continuous integration and delivery.

The continuous planning allows teams to quickly change and adapt plans or business strategies based on the market responses. Continuous integration means all the changes have to be early implemented, validated, and shared with the development team. Continuous Deployment depends on a hardware infrastructure that allows a quickly test provider anywhere and anytime. The principle of continuous testing is to automate every test. The continuous monitoring provides early details about the product operations and how to response to them. The Table 1 shows some of the tools used in each phase.

Table 1. DevOps tools

Phase	Tool	Language	License	Advantages
Continuous Planning	Trello	Cloud based	Commercial	Kanban based project management
Continuous Planning	Jira	Java	Commercial	User stories and issues tracking
Continuous Integration	Jenking	Java	MIT	Large active community and limitless integrations
Continuous Integration	TeamCity	Java	Commercial	.NET support and built-in features
Continuous Integration	Bamboo	Java	Commercial	Multiple notification methods and good integration with other Attlasian tools
Continuous Deployment	Puppet	Java	Apache	Focus on desired end state infrastructure
Continuous Deployment	Chef	Ruby	Apache	Client-server tool that can be used as an isolated installation;
Continuous Deployment	Ansible	Python	GPL	Open source, do not require installing agents
Continuous Testing	Selenium		Apache	Multiple browser platforms testing and highly flexible operations
Continuous Testing	Appium	C#	Apache	Focus on mobile applications
Continuous Monitoring	Nagios	C	Open source, GPL	End-user stations, IT services and active network components
Continuous Monitoring	New Relic		Commercial	Web and mobile applications
Continuous Monitoring	Cacti	PHP	GPL	High customizations and a template-base configuration

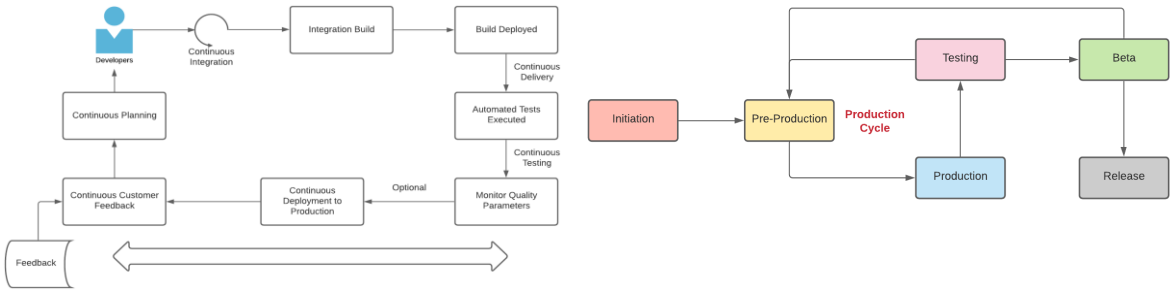
The DevOps cycle tries to automate all the repetitive processes and bring all the changes to the final product as fast as possible while obtain feedback from your costumes to future improvement. With this, the software can receive small but constant improvements that quickly respond to the consumer feedback. The DevOps often complements the agile cycle because the agile methodologies focus on the gap between developers and software requirements and the DevOps between developers and operations [Lwakatare, Kuvaja, Oivo 2016]. The Figure 1a shows a diagram of the DevOps cycle.

2.2 Game Development

A game development is similar to every other software development, with analysis, design, development, testing, implementation and maintenance. The difference when developing a game is doing these stages while creating a game that is fun and enjoyable to play, making the player want to play more. These elements are what make game creation one of the most challenging software development process on the digital industry.

The beginning of the game development generally starts with an idea. It is from this idea that the 'High Concept' is made. Providing the fundamental bases of the game design. With the High Concept, it is possible to elaborate the core concepts of the game: game design, level design, mechanics, story, art, music; even marketing aspects like target audience and divulgation strategies. These concepts, ultimately, become what is known as Game Design Document (GDD) [Bethke 2003], which is the primary font of description of a game.

With all defined, the development team can use the GDD to create a prototype of the game with the predefined concepts implemented. When the prototype is approved, the team creates a direction of what have to be done. From this point, the development continues until the eventual release of the game. After that, the focus point becomes the maintenance, bug fixes and additional content. The development cycle shown in Figure 1b projects the process flow, with the analysis and the High Concept in Initiation, prototype in Pre-Production, core development in Production, testing in Testing and Maintenance in Beta and Release.



(a) DevOps delivery pipeline [Virmani 2015] (b) Game development process [Ramadan, Widyani 2013]

Figure 1. Processes used as base to create an electronic game using DevOps

The tools required for developing a game is determined by the necessities of the team: art, music, prototype software; all of them are very important, but the most important is the game engine. The game engine is a game IDE that combines all the assets in a build. One of the biggest is Unity [Goldstone 2009]. Unity is a free and powerful game engine (with paid licenses) that works with C# language and an object-oriented architecture, capable to help create a lot of genres of games. Unity also works with external tools.

3. PROPOSAL FOR THE USE OF DEVOPS METHODOLOGY IN THE DEVELOPMENT OF GAMES WITH UNITY3D

On a team, the development planning has to be very clear with the tasks of the team. With continuous planning, the team keeps tracking of the feature workflow. One of the best continuous planning tools is Trello. In an unorganized team, with a big assets transit, problems like long waits, manual transit and defect files can occur. With continuous integration practices, every change is tested in the source and fast sent using a system like GitHub, making team members focus on their work and less on waiting and task switching.

The deployment with continuous deployment has to be made constantly with small changes each, making easier to track bugs and avoiding bug propagation. Using an automatic build-creator that can be trigged when detecting changes can avoid hours of manual creation of builds. Depending on the size of the project, even small changes can cause bugs, and the QA team have to detect them before they cause others. Automatic testing allows testing a one or more features, multiple times in various platforms with minimum efforts.

The development team have to be aware about eventual bugs. The continuous monitoring ensures that by allowing the team to fix a bug before that cause more serious problems. This not only help monitoring the build but automating the QA feedback. Trello for Unity is a tool that allows to create a Trello card directly from Unity. These processes allow the team to create more content without crunching. The Unity engine have its own tools, such as the Unity Cloud Build for continuous integrations and Unity Test Framework for continuous testing. Other external tools can help more, such as Trello integration or automatic build creator.

These game development processes with DevOps methodology and practices can be combined in a diagram with focus on production and automation (Figure 2). The initiation gives the game concepts. Pre-production involves the creation and the revision of game design and prototype, the continuous planning application allows quickly changes based on players and market feedback. The production involves the creation of assets and the integrations of these elements, with continuous integration, every change is committed directly in the source.

With the game on Beta or even on Release, customers can create their own feedbacks, which motion is also automatized, making the team respond quickly to the customers and release fast updates passing through the same processes of pre-production, production and testing.

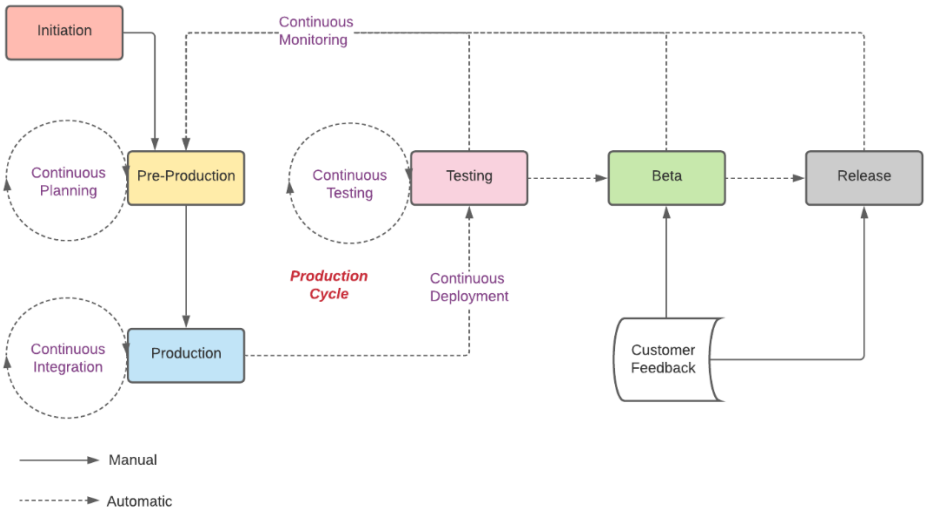


Figure 2. Game development process with DevOps integration

The authors of this paper, participating in an R&D project that includes game development, have used some of the phases of the DevOps methodology in the project. The DevOps phases implemented in the project were continuous integration, continuous deployment and continuous testing. Before, the development of the project had some difficulties, but after applied the DevOps methodology, the team had notice drastic reduction of severe or unnoticed bugs, a greater control of the functionalities in the project and a fast integration of changes.

Daily builds of the project were created to test and monitor each of the changes in detail. This versioning makes it possible to find bugs much faster, identify their origin more easily and monitor the changes and documentation of the team's progress, guaranteeing the quality of the product and its versioning.

4. CONCLUSION

This research has the purpose of studying and proposing the use of DevOps methodology in a game development environment such as continuous planning, continuous integration, continuous deployment, continuous testing and continuous monitoring. To facilitate the incorporation in the game development process, a workflow was proposed. The incorporation of these methods and tools in game development will allow obtaining higher quality products at a lower efforts and time cost.

In addition, some tools were proposed for each phase, particularly if the game is being developed using Unity. During the research, it was found that it is not yet fully formalized in the scientific community how to use DevOps for game development, although there are several tools that can be used for this purpose.

Also, the experience with DevOps in an R&D project, including the game development, was described. The use of this practice significantly improved the quality of the product, reducing the number of bugs and favouring the control of the implemented functionalities with creation of the different versions of the product. The use of these methodologies also had an impact on people. A greater commitment to the results was noted, since they were constantly receiving feedback on the work results. During the investigation, the feasibility of using the DevOps methodology in the development of games was evidenced.

ACKNOWLEDGEMENT

The authors would like to thank LUDUS Lab and the Amazonas State University (UEA) for the help in the making of this article. The results of this work were published by the R&D activities of the ARKADE Project, financed by TRANSIRE FABRICAÇÃO DE COMPONENTES ELETRÔNICOS LTDA, with the support of SUFRAMA on the terms of the Brazilian Federal Law Nº 8.387/1991".

REFERENCES

Abiteboul, S. et al, 2000. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann Publishers, San Francisco, USA.

Beck, K. and Ralph, J., 1994. Patterns Generates Architectures. *Proceedings of European Conference of Object-Oriented Programming*. Bologna, Italy, pp. 139-149.

Bethke, E. (2003). *Game development and production*. Wordware Publishing, Inc.

Bodorik P. et al, 1991. Deciding to Correct Distributed Query Processing. *In IEEE Transactions on Data and Knowledge Engineering*, Vol. 4, No. 3, pp 253-265.

Goldstone, W. (2009). *Unity game development essentials*. Packt Publishing Ltd.

Jones, C. 2020. *Every 2020 Game Release Date That's Been Delayed*. March, 26 of 2021. http://bit.ly/2020Delays

Lwakatare, L. E., Kuvaja, P., & Oivo, M. (2016, November). Relationship of DevOps to agile, lean and continuous deployment. In *International conference on product-focused software process improvement* (pp. 399-415). Springer, Cham.

Ramadan, R., & Widyani, Y. (2013, September). Game development life cycle guidelines. In *2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS)* (pp. 95-100). IEEE.

Virmani, M. (2015, May). Understanding DevOps & bridging the gap from continuous integration to continuous delivery. In *Fifth international conference on the innovative computing technology (intech 2015)* (pp. 78-82). IEEE.