# Is it Viable to use Continuous Integration to Reduce Development Time for an RPG

**COMP130 - Software Engineering**

1604281

2018-03-19

**Abstract**

## 1 Introduction

At its core, continuous integration is an Extreme Programming concept where developers commit their code to the repository multiple times a day where it is automatically built and tested. [1] [2] Although the frequency of builds can vary, in an ideal situation, the code will be built whenever a commit is made. This helps to ensure that new code can be integrated successfully.

## 2 The Nature of RPG's

The term RPG - or Role Playing Game - refers to a broad range of games that are difficult to define due to the open-endedness of the term and the genre overlap that is common in video games. However, traditional RPG's often include key elements such

as a structured storyline and rules to restrict the player to that plot. [3] Mechanically, a levelling system, inventory and the use of quests to drive the game are also common.

Most relevant to this paper is the observation that RPG's tend to emphasise depth, and as such, combine many different systems within the game as a whole. For example, combat mechanics, xp and level-up systems, a world map, inventory and quest system may all be required for a relatively simple RPG. This breadth of content can make continuous integration especially relevant to the genre, as it is imperative that the game systems are compatible.

## 3 Benefits of Continuous Integration

One of the largest benefits of continuous integration is its ability to reduce merge errors when integrating code into the repository. If developers wait a long time before making committing code, resulting issues can be complex and hard to solve, often wasting valuable development time while they are fixed. Many small commits each day reduces the likelihood of merge errors, but also means that errors that do occur are easy to solve. [4]

Continuous integration also encourages and enhances communication between developers because of the quantity of commits that are made. More commits means that it is easier to see what changes have been made by other developers and provides the platform to query or make notes on individual changes via commit messages and comments.

Many developers find that the benefits of continuous integration allow them to release more often and have more successful pull requests. Both of these factors help reduce development time and streamline the project lifecycle. [5]

## 4 Disadvantages of Continuous Integration

One of the immediate concerns regarding continuous integration is the initial setup time and the associated costs. [6] Dedicated computers to build the code and run automated tests will be needed in any large development project, as running automated tests can take a long time, making it impractical to attempt to use developer computers. [7] Additionally, installing and configuring a continuous integration platform may take some time.

If the project includes hardware components as well as software, integration tests could require changes to the physical components to be successful, necessitating time and potentially money being spent to update the hardware. If this takes a longer than the continuous integration process, delays could be caused, as integration tests cannot be completed until the hardware is changed. [8]

## 5 In the Games Industry

The applicability of continuous integration changes somewhat in regards to the games industry. Games projects are often composed of large, complex development teams working on many different aspects of the game at once. Because of this, merge conflicts can be particularly disastrous and may delay multiple development teams if there are dependencies between departments. Because of this, continuous integration can be extremely useful for games development teams to try and avoid such a situation, while also providing good communication and transparency between individual development teams via multiple commits each day.

On the other hand, games can quickly become very large as they usually include assets such as 3D models, shaders and textures. This can make automated builds very slow which could cause delays in the continuous integration system. Possible solutions include only doing a full build of the game when it's necessary - such as once per day - and use

a minimal build at other times. [9]

## 6 Applicability to RPG-Style Games

From a programming point of view, designing an RPG game can be a challenging experience. Regardless of the style of RPG, there will almost certainly be many different aspects that need to be designed to work together. The complexity of the systems involved in a large-scale RPG may necessitate features being removed, reworked or added at the last minute. These changes can cause delays and complications for developers if the codebase is not well-maintained and planned to allow modification. Although there are many factors involved in making code maintainable and modular, continuous integration encourages the practice of committing small working portions of code, which in turn creates a more readable and maintainable codebase than if very large pieces of code are submitted at long intervals. [4] [10] Despite this, making code modular enough to have working code ready for committing multiple times a day can be difficult for developers. Although features should always be broken into smaller tasks during development, a few hours to complete even a small element of a feature can be a tight deadline in itself. Particularly when looking at RPG games, complex systems such as enemy AI can be difficult to divide into small enough tasks to make each one quickly completable.

One major cause of deadline issues and the need for crunches is poor communication between managers and developers. If managers do not have a good idea of where the development team is currently, they may assume that everything is on schedule, leading to problems when they discover that features they thought were complete are not implemented. [11] Continuous integration helps to facilitate this communication due to automated, verified builds. A manager can look at the repository and see that there is a working, successful build and see what has been contributed to it via commit logs.

The large scope of many RPG's - especially triple-A industry projects - often leads

to deadline slippage and harsh crunches to rectify this. In this context, continuous integration could be a valuable tool to help streamline development. However, RPG's are also a popular choice for indie studios, where teams may by very small. In this environment, the initial cost of setting up continuous integration as well as the time period where developers are still getting used to the new system may be perceived as outweighing the benefits as it can be difficult to change the established practices and developer processes that are already in place. [12] Dismantling an existing developer workflow in order to facilitate a new continuous integration oriented one could disrupt production at a critical time, although this could also by avoided by adopting continuous integration between projects, or very early on in development.

Despite this, continuous integration can benefit small studios, as having strict integration tests and automatic builds in place allows new developers to quickly begin adding code to the project without worrying about breaking the build. [13] This can aid indie studios by enabling them to expand their teams during a project, if and when they need more developers. This freedom could reduce development time, as the number of programmers can be changed to match new requirements for the project. Similarly, this can also enable much larger studios, who already have very big development teams, to easily reorganise departments or create new teams.

One the other hand, another concern with using continuous integration is that the automatically verified build can ironically lead to lowered standards for code contributions. This is due to developers simply needing to make sure their code passes the integration tests when they commit it. [14] This can be harmful to development as the quality of the code may decrease as developers do not feel the need to run further tests on their code or make sure that it conforms to good standards. In a complex RPG with many dependencies between features, poor quality in any code can be extremely detrimental later on in development and could cause delays.

## 7 Conclusion

In conclusion, despite the initial cost - in both time and money - of setting up continuous integration, it has the potential to significantly decrease development time during an RPG project. The ability to identify merge conflicts early on in development is invaluable when working on a large project with many different components that may be worked on by individual teams. Additionally, if dedicated computers are used, the automated build functionality of continuous integration can help speed up the project by removing the need for developers to build huge projects on their local machines, which would take large amounts of valuable time. [9]

Even in small projects with few team members, it can still be valuable to set up a continuous integration system. Not only can continuous integration improve error identification - even with small development teams - setting the system up early on can allow the studio or project department to grow in the future. It is much easier to change development processes early on than when the team is large and the project has grown very complex.

## References


[1] M. Leppänen, S. Mäkinen, M. Pagels, V.-P. Eloranta, J. Itkonen, M. V. Mäntylä, and T. Männistö, "The highways and country roads to continuous deployment," *Ieee software*, vol. 32, no. 2, pp. 64–72, 2015.

[2] W. Penson, E. Huang, D. Klamut, E. Wardle, G. Douglas, S. Fazackerley, and R. Lawrence, "Continuous integration platform for arduino embedded software," in *Electrical and Computer Engineering (CCECE), 2017 IEEE 30th Canadian Conference on*. IEEE, 2017, pp. 1–4.

[3] A. Primanita, R. Effendi, and W. Hidayat, "Comparison of a and iterative

deepening a algorithms for non-player character in role playing game," in *Electrical Engineering and Computer Science (ICECOS), 2017 International Conference on.* IEEE, 2017, pp. 202–205.

[4] Y. Zhao, A. Serebrenik, Y. Zhou, V. Filkov, and B. Vasilescu, "The impact of continuous integration on other software development practices: a large-scale empirical study," in *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering.* IEEE Press, 2017, pp. 60–71.

[5] M. Hilton, T. Tunnell, K. Huang, D. Marinov, and D. Dig, "Usage, costs, and benefits of continuous integration in open-source projects," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering.* ACM, 2016, pp. 426–437.

[6] C. Amrit and Y. Meijberg, "Effectiveness of test driven development and continuous integration–a case study," *IT professional*, 2017.

[7] F. Rken and D. Frommhold. Automated tests and continuous integration in game projects. [Online]. Available: https://www.gamasutra.com/view/feature/130682/automated_tests_and_continuous_.php

[8] R. Kasauli, G. Liebel, E. Knauss, S. Gopakumar, and B. Kanagwa, "Requirements engineering challenges in large-scale agile system development," in *Requirements Engineering Conference (RE), 2017 IEEE 25th International.* IEEE, 2017, pp. 352–361.

[9] L. Winder. Opinion: Is it green yet? improving our ci process. [Online]. Available: https://www.gamasutra.com/view/news/125977/Opinion_Is_It_Green_Yet_Improving_Our_CI_Process.php

[10] M. Brandtner, E. Giger, and H. Gall, "Supporting continuous integration by mashing-up software quality information," in *Software Maintenance, Reengineering*

*and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on*. IEEE, 2014, pp. 184–193.

[11] A. Maglyas, U. Nikula, and K. Smolander, "Lean solutions to software product management problems," *IEEE Software*, vol. 29, no. 5, pp. 40–46, Sept 2012.

[12] M. Shahin, M. A. Babar, and L. Zhu, "Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices," *IEEE Access*, vol. 5, pp. 3909–3943, 2017.

[13] D. Spinellis, "State-of-the-art software testing," *IEEE Software*, vol. 34, no. 5, pp. 4–6, 2017.

[14] R. Pham, L. Singer, O. Liskin, F. F. Filho, and K. Schneider, "Creating a shared understanding of testing culture on a social coding site," in *2013 35th International Conference on Software Engineering (ICSE)*, May 2013, pp. 112–121.