# Lean Game Development

Apply Lean Frameworks to the Process of Game Development

*Second Edition*

Julia Naomi Rosenfield Boeira

Apress®

# Lean Game Development

## Apply Lean Frameworks to the Process of Game Development

## Second Edition

Julia Naomi Rosenfield Boeira

Apress®

***Lean Game Development: Apply Lean Frameworks to the Process of Game Development, Second Edition***

Julia Naomi Rosenfield Boeira
Winnipeg, MB, Canada

ISBN-13 (pbk): 978-1-4842-9842-8 ISBN-13 (electronic): 978-1-4842-9843-5
https://doi.org/10.1007/978-1-4842-9843-5

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Jessica Vakili
Development Editor: James Markham
Coordinating Editor: Spandana Chatterjee
Copy Editor: Kezia Endsley

Cover image from Freepik (www.freepik.com)

*To all women and non-binary folks who work as game developers.*

# Table of Contents

# About the Author

**Julia Naomi Rosenfield Boeira** has been a software engineer for almost two decades, focusing on game development, Rust, and online services. Currently, she works as a team lead at Ubisoft; previously she worked as tech lead engineer for an online service. She has also worked with premium Agile consultancy companies such as Thoughtworks and is extremely active on GitHub with a focus on game development and Rust.

# Acknowledgments

This book would not have been possible without the support of my family: Diego, Kinjo, and Ffonxios; the contributions of Jay Kim, Evan Boehler, and Vitor Severo Leaes, who helped me review the book and gave me awesome feedback; to all the great Lean and Agile discussions I had with Paulo Caroli, Enzo Zuccoloto, and Marcelo Porcino at Thoughtworks; to the Rust in PoA Community: Julio, Douglas, Ruan, and Eva; to my parents who have always supported me and my studies; to my colleagues at Ubisoft: Evan, Denys, David, Madison, Poliana, Lune, Ryan N, Ryan H, Ryan F., Jesse M., and Sue; and, lastly, to my friends who helped me in this journey: Isabela Goes, Eva Pace, Thais Hamilton, Alexandra Perreira, Diego Ferreira, and Icaro Motta.

CHAPTER 1

# Introduction

This book's goal is to present a new way of developing games to teams with little or no experience in Agile or Lean methodologies. In addition to that, this book uses a fictional game, which is the combination of three other anonymous games that were produced with the techniques presented in this book. Unfortunately, it is not possible to go into more detail about these games due to NDAs (agreements of secrecy).

---

**Note** The word *Lean* comes from the Toyota Production System, which is a systematic method for waste minimization without sacrificing productivity. Also, the concept of Lean was popularized by the book *The Lean Startup*, by Eric Ries, which aimed to create startup business models following five concepts, which are discussed later. This book specifically focuses heavily on Lean techniques from a game development point of view.

---

If you have some experience with game development, now is the time to put it aside. It's time to let go of the things you know and, with an open mind, learn something new, or at least from someone else's point of view. The goal of this book is to provide you with a game production model that prevents waste, reduces bugs, and offers continuous reviews. The book even offers a sequence of steps to eliminate unnecessary tasks. When I developed this methodology, I was thinking of small- to mid-sized game companies, but it can be used in large enterprises as well.

J. N. Rosenfield Boeira, *Lean Game Development*,
https://doi.org/10.1007/978-1-4842-9843-5_1

Besides that, I like to believe that this book is focused on small- to mid-sized *indie* game development companies—that is, perhaps, the group that would enjoy the most advantage by this methodology. Of course, large companies can also take advantage of the methodologies presented here; however, larger companies typically have more difficulty adapting these methodologies, since they have more bureaucratic processes, more vertical opinions, and more secrecy around their projects, making them more resistant to changes. In addition, they often need outside help to identify their strengths and weaknesses, as well as help to identify which points in the process are good or bad, since their more hierarchical nature usually prevents bottom-top improvements. Therefore, this book can be a tool for large companies in middleware projects, specific features, art, and tooling.

## Why Lean Game Development, Not Agile Game Development?

Lean is something beyond Agile. In fact, many game companies have been unsuccessful in their first attempts to adopt Agile methodologies. This has generated some weird confusion about Scrum and Agile, which can be observed with tons of blog posts bashing Agile, mostly Scrum, for game development.

Another important factor is that many companies confuse Agile methodologies with Scrum, considering Scrum the only Agile tool available. The same happens with extreme programming (XP), and this confusion can have disastrous results. In fact, it's common to see companies adopting Scrums but not adopting the basic principles of Agile, which overloads the development teams. Another common case is those small "flexibilities" in the wrong direction of Scrum, which generate an even more waterfall process. How many times have I heard "This is not true Scrum!," "This is not really Agile," and "Now we have become Extreme Go Horse."[1]

Lean game development can meet the main needs of the game industry, but there are certain game-related aspects to take into account. For instance, game production is never 100 percent efficient, since you can't predict every possible problem, and it is far more difficult to find the "minimum" in a minimum viable product (MVP) in game development than in other industries. If you set fixed deadlines, the best you can expect is to get very close to them because unexpected problems and unexpected changes in scope will continue to happen, even after the deadlines. It's necessary to behave organically regarding changes, building in the ability to adapt to the environment.

Lean game development offers a methodological alternative to game development that can help you eliminate waste, get results as fast as possible, strengthen and empower teamwork, and get a better view of the whole work. How do you improve this visualization of the work? Kanban (which literally means a visualization card) is a classic tool that Lean development teams use.

That said, it's important to emphasize that in no way are Lean, Scrum, XP, or Kanban exclusive. They can be used together, thereby enjoying the best features of each.

[1] https://medium.com/@dekaah/22-axioms-of-the-extreme-go-horse-methodology-xgh-9fa739ab55b4

## How Do Lean and Agile Relate to the Game World?

Lean game development is, above all, strongly inspired by Agile and can take advantage of Agile's tools to develop software. Therefore, let's look at the Agile Manifesto and interpret it to represent the vision of games. For such, I suggest the following point of view for games:

- *Individuals and interactions* over processes and tools
- *Games running* over comprehensive documentation[2]
- *Audience collaboration* over sales
- *Spontaneous development* over following a strict plan

## Games and Software Relate Much More Deeply

To successfully understand Lean game development, you should first understand that digital games are also software and that software can be seen as a cooperative game of innovation and communication. Games are not only for children and teens; games are used to describe everything from romantic experiences to advanced military strategies, and they can also be used as another form of software development.

[2] www.gamedeveloper.com/disciplines/agile-game-development-part-3-working-game-vs-gdd

## GAMES FOR MILITARY STRATEGIES

The Blitzkrieg board game was used for a long time to help train army officers. The game is set in World War II, in which two armies confront each other: Great Blue and Big Red. There are five countries, but the alliances are not built strictly and can vary depending on how the game is played.

The game has three modes: simple, advanced, and tournament. One of the most interesting aspects is that advanced mode offers many combat units, such as infantry, artillery, armored, assault, shots, bombing, and so on.

Unfortunately, the game is usually hard to find, maybe because it's old, and it usually takes a couple of days to finish gameplay. Figure 1-1 shows the (gigantic) board of the game with the different colored pieces.

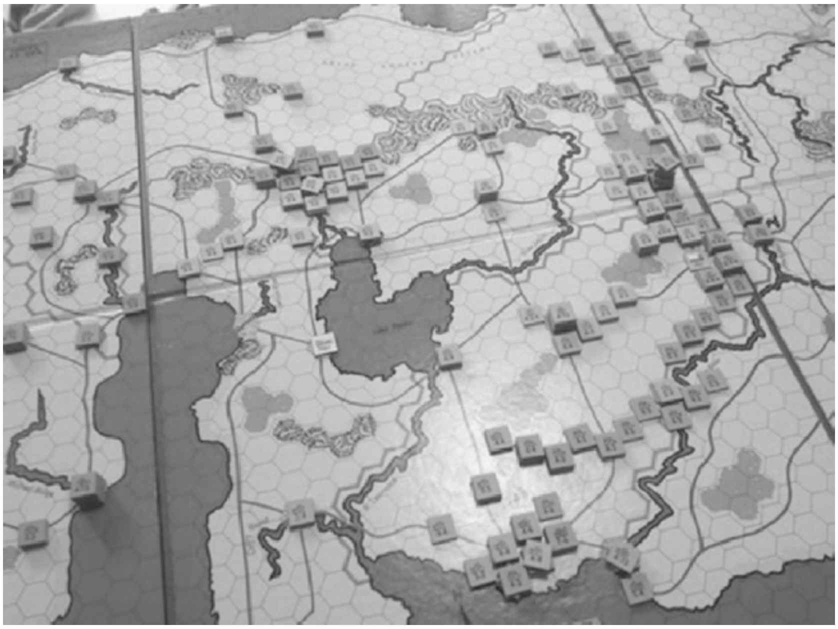

***Figure 1-1.*** *Avalon Hill's Blitzkrieg boardSource: boardgamegeek.com*

When someone proposes to play a game, hundreds of alternatives come to mind: tic-tac-toe, checkers, chess, poker, 21, hide-and-seek, ping pong, and so on. Games usually fall into certain categories that help players realize how they are played and what the goals are.

- *Zero-sum*: These are games in which each user plays on an opposite side, and if one side wins, the other loses. Examples include checkers and tic-tac-toe.
- *Non-zero-sum*: These are games with multiple winners and losers. Examples include poker and hide-and-seek.
- *Positional*: These are games where the overall state of the game can be determined by looking at the board. Examples include chess and tic-tac-toe.
- *Competitive*: Games in which there's a clear notion of winning and losing.
- *Cooperative*: In these games, people play together to win, or until they find it necessary.
- *Finite*: These are games that have an end.
- *Infinite*: These are games where the primarily intention is to keep playing. In other words, the goal is to remain in the game.

## What Kind of Game Is Software Development?

Many people see software development as a positional game, with a cycle of small victories and a clear goal. But a software development game is much more than positions on a board and much more than a team trying to overcome obstacles until the project can launch.

Software development is a cooperative game, in which all pieces must help each other in order to reach each one of the goals. Think of a survival game, in which each member of the team has a specific and unique skill that is useful to the group's survival. The software development process is similar to the concept of a cooperative game. There should be no leader; instead, a group of people unite to make the best decisions and divide tasks the best way possible to survive (win).

## Where Did We Go Wrong?

Unfortunately, over time, people got the idea that stricter and heavier methodologies, with more control and more artifacts, would be "safer" for a project's development. However, no one likes to play games with hundreds of rules that need to be remembered every minute in order for the player to perform correctly.

The best games are those that can be played in a relaxed way so that if a rule is broken, there won't be any big consequences for the result. Furthermore, games that allow the player to be creative and imaginative tend to provide much more cooperation. You just have to observe kids playing board games to realize this.

Taking this into consideration, why not apply this to software development? Let's look at an example of the negative effect that rigidity and heaviness can have on the classic board game Monopoly. Imagine that, besides the players, you have a person solely responsible for administering the bank, one for administering the real estate, another one for administering the real estate bank, one for administrating your life (chance cards), one police officer for administering prisons and the flow of characters, another one to roll the dice, and so on.

This type of model is the software development model used in most companies: it's highly hierarchical, it has several types of control over individuals, it has strict rules, micromanagement is encouraged, it's difficult to play, and it's aimed at exploring others. How can this model be superior to a relaxed, fun, creative, and cooperative model? The Agile Manifesto, the framework, and the methodology should not be applied in a rigid and immutable way. After all, a game should allow for fun, cooperation, and creativity.

Probably, this presumption that heavier methodologies are safer comes from the assumption that project managers can't look at the code and evaluate the degree of development, the status, and the project situation. Adding weight and rigidity to the model won't make the fear and insecurity regarding the project better, however. In fact, the consequence will be making your team delay their work and miss the deadline.

To achieve satisfactory results, always keep in mind that developing software is a team game, in which the production manager is not superior to anyone else. Remember, there are ways to document while the code is being written, and there are ways to visualize the software development without increasing the pressure on the team. A production manager shouldn't think of the team as people to boss around and coordinate, but rather as colleagues that they need to help.

The following are some prejudices of the game industry regarding the Agile methodology[3]:

- Test automation in games is much more complex than in other software industries.
- The game visual aspect cannot be tested automatically.
- Making open betas and demos for kids to test the game is much cheaper.
- The current business model in the sector is based on feature-complete games.
- "I don't like Scrum," because Scrum was the answer to Agile methodologies.
- The game's sequences and sequels are not iterations.

[3] www.gamedeveloper.com/programming/agile-game-development-is-hard

- Art cannot be iterated, and games are art.
- Games are developed so that the users play longer, not to save time like in e-commerce.
- It is impossible to create an automated test pyramid for games, especially large productions.
- From a production's point of view, continuous delivery is not attractive to games.

All of these points are discussed in depth in later chapters, but I think it's important to point out now why each of them is wrong:

- Test automation may be more complex, but it is certainly as or more valuable than in other industries.
- Gameplay tests and tests that identify errors in images, as well as how close an image is to ideal, are fundamental features for games. These kinds of tests are regularly done in mobile development and frontend development. For games, we have a few resources that can help measure this, like OpenCV.
- Having children test games, even if it generates some degree of satisfaction in them, is morally wrong and can greatly affect the reception of a game. If they are your target audience, be sure to have parents involved.
- Scrum most definitely is not the only Agile methodology.
- Later, you will see that art is a creative and iterative process.
- You want users to play longer, but spend less time trying to learn the game, where and when to click, and its mechanics.

- Not only is it possible to test games, I wrote a book on automated testing for games.
- In the old days, when the game came in a cartridge and there was no Internet to update the game, this phrase could even make sense. Nowadays, some games are released without even being ready.

Thus, it's important to understand the basics of Lean and remind yourself that software development is a cooperative and fun game, in which all pieces are important. It's a game that always produces more knowledge. Ideally, it must be managed organically and with low hierarchy to prevent waste (of human potential, of time, of excessive documentation, of conflicts, of stress, etc.).

This book covers several aspects of Lean development—such as the basic aspects, the inception, and MVPs—and applies them to games. You also learn how to use test-driven development, how to use continuous integration in games, and how to generate hypotheses. Lastly, you see how design and build[4] are different and learn more about tests, measurement and analysis, and the generation of ideas.

## Summary

In this chapter, I talked about the relationship of Lean and Agile in the game development world. I also discussed the deeper relationship that games and software have, including how software development can be seen as a game.

[4] A general reference to software engineering and its practices.

CHAPTER 2

# First Steps with Lean

This chapter explains Lean in a deeper sense and how it relates to game development. It also presents a visualization of the Lean game development cycle. Finally, the chapter introduces some places where Lean game development can take advantage of Agile methodologies.

## Seven Key Principles of Lean

When starting to talk about Lean in more detail, it's important to cover the seven key principles of Lean:

- *Eliminate waste*: This includes avoiding the following: producing disorderly and unnecessary inventory and requirements, giving excessive importance to defects that don't affect the user experience, processing unnecessary information, and creating long wait times. To reach those goals, avoid unnecessary code and features. Another important consideration is to not start more activities than can be completed.

  From a business point of view, it's necessary to elaborate on the requirements so they are easily understood and to avoid changing them constantly. Especially avoid bureaucracy. Inefficient communication can lead to misunderstandings

J. N. Rosenfield Boeira, *Lean Game Development*,
https://doi.org/10.1007/978-1-4842-9843-5_2

regarding the job to be done. From a developer's point of view, it's important to ensure that the job is complete and that you don't end up with defects and quality issues in the finished code. But maybe the most important issue is to prevent unnecessary changes in the job tasks.

- *Build with quality*: Quality problems lead to waste; in addition, it's a waste to test something more than once. To avoid quality problems, you can use pair programming and test-driven development. Both are fundamental tools, and both are e described in the coming chapters of this book.
- *Generate knowledge*: Generate knowledge while you're working so that the whole team can follow the software development process and have the technical ability to deal with problems. A usual way to generate knowledge is through pair programming and code reviews. Wikis, dev huddles, and docs are other tools you can use to share knowledge.
- *Postpone commitment*: Complex solutions should not be treated rashly, and irreversible decisions should not be made hastily.
- *Deliver fast*: It's common for people to spend a lot of time thinking about requirements that may or may not come up. Workers can also become mentally blocked or start thinking of solutions with excessive engineering. You can avoid this problem by gathering the right people, keeping things simple, and using teamwork.

- *Respect people*: The workplace should be pleasant, so never forget to be courteous with people. No matter what your position is in the company, you must always seek for equity between roles.
- *Optimize the whole*: This principle seems simple and intuitive, but it's usually not taken into account. It's important to identify failures, propose solutions to them, and look for feedback. A whole is not made solely by its parts but by people interacting.

While using the following methodologies, it's always important to keep these Lean principles in mind.

## Lean Inception

An interesting stage of software development in the Lean methodology is the Lean inception. Briefly, the *inception* is a workshop, done typically in a week, with many activities of alignment and goal setting. The product evolution ends up being represented by a minimum viable product (MVP) and a sequence of iterations over the MVP, each with its own features. If the team has enough time, defining alternative MVPs in case of failure is also a nice strategy.

The main goal is to define the scope of what is being created so that the team has a clear view of the path to follow, that is, the minimum game that needs to be built to generate results and verify its viability. If an MVP proves to be non-viable, new MVPs can be generated. Figure 2-1 provides insight into an MVP and its sequences. The MVP 1 part corresponds to the people who are going to prove that your MVP is worth it, known as early adopters. They will cut their own grass with the best tool available, while MVP 8 will allow people to cut the grass in a football field.

Another important point to learn from Figure 2-1 is that the MVP is not only about how feasible or how valuable the product is, but how delightful and usable is. When considering an MVP, you need to add elements from all of these aspects (feasibility, value, delightfulness, and usability). Another, more recent, idea is the MLP (Minimum Lovable Product), which differentiates itself from an MVP by focusing on delivering something that is more effective in an already saturated market.[1]

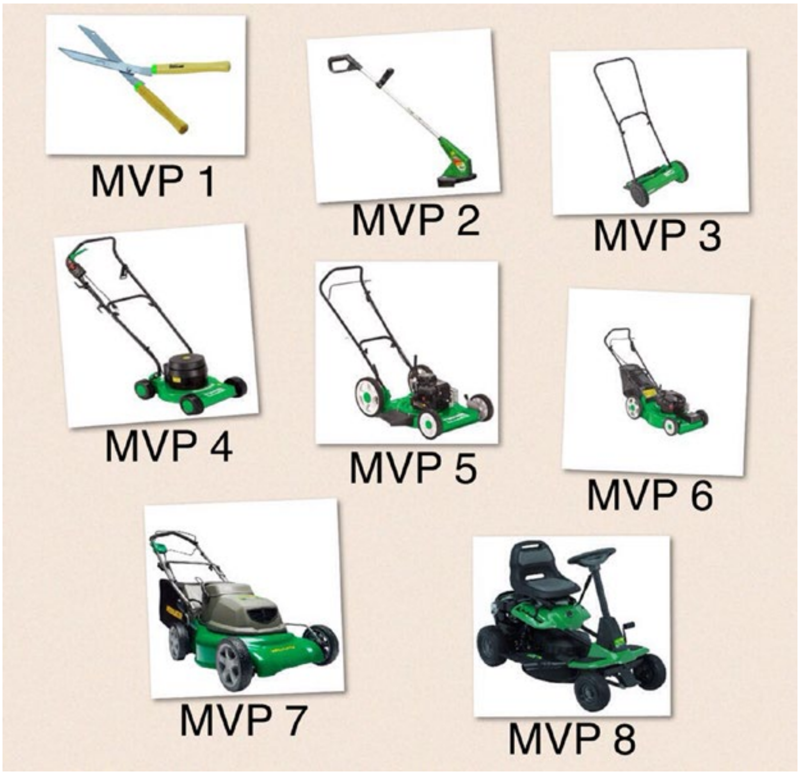

***Figure 2-1.*** *MVPs must occur incrementally, so that each increment provides a new return. Source:* *https://caroli.org/en/mvp-examples*

[1] https://medium.com/codica/what-is-a-minimum-lovable-product-and-how-to-build-one-22cb61f67e8a

**MVP CANVAS**

The MVP canvas gathers elements from design thinking, Lean startup, and business directives. It's a template to validate new ideas and question existing ones. It's divided into seven branches.

*MVP vision*: What product vision must this MVP deliver?

*Metrics for hypothesis validation*: How do you measure the results of this MVP? And from what business point of view?

*Outcome statement*: What knowledge are you seeking with this MVP?

*Features*: What do you intend to build with this MVP? Which actions can be taken in order to simplify the MVP?

*Personas and platforms*: For whom is this MVP?

*Journeys*: Which user journeys will be improved in this MVP?

*Cost and schedule*: What are the cost and schedule for this MVP?

Read more at www.caroli.org/.

## How Does Lean Inception Apply to Games?

The main tasks of the Lean inception are to come up with the game features, its basic game design, and the tools to be used. In short, there's a whole series of possible applications of the inception. It's also important to get the whole team engaged to increase motivation and build empowerment in the group. A more horizontal team is more engaged, has more respect for what is being developed, and enjoys a greater sense of ownership over the product.

## Lean PMO

The *project management office* (PMO) is a group of people (or even a single individual) responsible for keeping an integrated vision of the strategic plan throughout the whole product development, including managing deadlines, project rescoping, and costs. These people are responsible for gathering the company's portfolio to guide, plan, and organize the activities of the projects in the best way possible.

A Lean PMO manages the game development and organizes and keeps track of requests and MVPs. The PMO does this by taking into consideration the body of work, without getting mired in Agile technical details, like with what can happen with extreme programming (XP), Scrum, and Kanban.

The PMO's main role is to guarantee the continuous delivery of the game. This person/group needs to be aware of the whole and not the details and periodically monitor the product development.

## How Does a Lean PMO Apply to Games?

When it comes to game development, the Lean PMO is usually the team that gives the go-ahead for a game to move to the next stage, release an open beta or demo, or the game launch itself. This may vary when considering the company size, as larger companies tend to have more steps and be less Lean. Talking about continuous delivery when the game is not yet on the market might seem like shooting yourself in the foot, but continuous delivery doesn't necessarily have to be for the gamer. The idea behind this is managing the development steps, so that the product continues to evolve and receive feedback.

## Lean DevOps

The function of *DevOps* is to connect the practices of DevOps to the Lean MVP perspective (which is explained in the next chapter). DevOps refers to the practices that the team uses while creating and delivering the game.

DevOps doesn't have to be executed by a single person; it can be used by a group of people, by different people in different moments, and in different practical activities. It includes working with features and user stories. Some teams have specific people on the project who lead DevOps initiatives, such as the tool team.

### How Does Lean DevOps Apply to Games?

You can, for instance, designate people who are responsible for organizing and applying the techniques, methodologies, and tools that the team has, as well as guiding the game's deployment.

## Kanban

*Kanban* is based on Toyota's just-in-time model. The method consists of visualizing the workflow and acting on the process in order to not overload the members of the team. The process is exposed to the team members using the visual management approach, from the early stages to the final delivery.

Physical or virtual boards can be used, such as Trello. Some Kanbans are divided into columns and rows, but not all of them need to be. There are some very creative implementations; just search for and use the one that matches your team's needs.

A Kanban is composed of several "cards," with each one representing an action that must be taken. The action's degree of completion is marked on a panel, usually called the *Heijunka board*. This system has some advantages; it's easy to identify waste and can lead to faster cycles that increase productivity. When associated with virtual boards and processes, collecting information about cycles is very efficient; in the case of physical boards, it depends on the feeling and expertise of each team.

Color coding can indicate the status of the card and enable people to identify delays, allowing them to be solved first. Different cards in the same zone usually mean a failing flow and must be resolved. Many companies automate parts of the process with light signals on machines/user stories that are struggling with certain tasks. I worked with a developer experience team that developed an internal tool connecting Jira with Grafana to alert teams about user stories that they were struggling with.

From my point of view, this helps eliminate long daily meetings, as the board state is visible to everyone; the need of a Scrum Master, as blockers are visible and transparent to the whole team; and other wastes. Laborious tasks can often be divided into smaller ones, also increasing efficiency.

The work-in-progress (WIP) concept is used to limit activities that will be "pulled" in the Kanban method. From the Kanban's point of view, the next activity is only pulled when the WIP has work capacity. The WIP restrictions identify bottlenecks and possible problem areas in the process, helping to make team decisions (you can read more at `www.caroli.org/`).

A common concept to deal with work that is blocked is to label the task with a `Blocked` tag or label to communicate flow issues to the team and leadership—and, if necessary, take on another task until the blockage is resolved.

## How Can You Take Advantage of Scrum?

As you probably know, *Scrum* is an Agile framework for completing and managing complex projects. It is highly recommended for projects that have requirements that change rapidly.