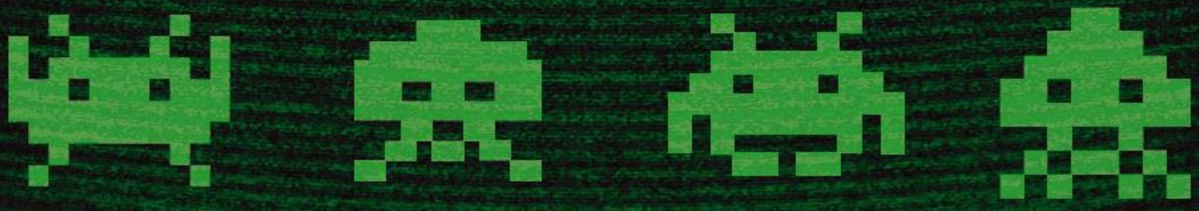
GAME OVER

Please try again



Game Design & Production

Jack Murray

Contents

Executive Summary

Purpose of the Document

The purpose of this document is to inform production and design teams on effective methods to minimise the negative reception at launch and provide an adaptation of Agile development frameworks in order to help games that have released to bad reception to recover.

Overview of the Proposed Framework

The proposed adaptation of agile reinforces the best practices to make it suitable for the challenges of game development, addressing key industry challenges such as ambiguity, creative iteration, stakeholder management and realistic marketing. In order to achieve this a flexible planning method approach promotes teams to use milestone-driven progress and stakeholder centred communication to facilitate sustainable development. The framework also covers post-launch support where games have released to negative reception. Drawing on proven agile principles already used in the industry it aims to improve on specific practises to enhance creative processes, transparent communication and player trust.

Key Benefits and Objectives

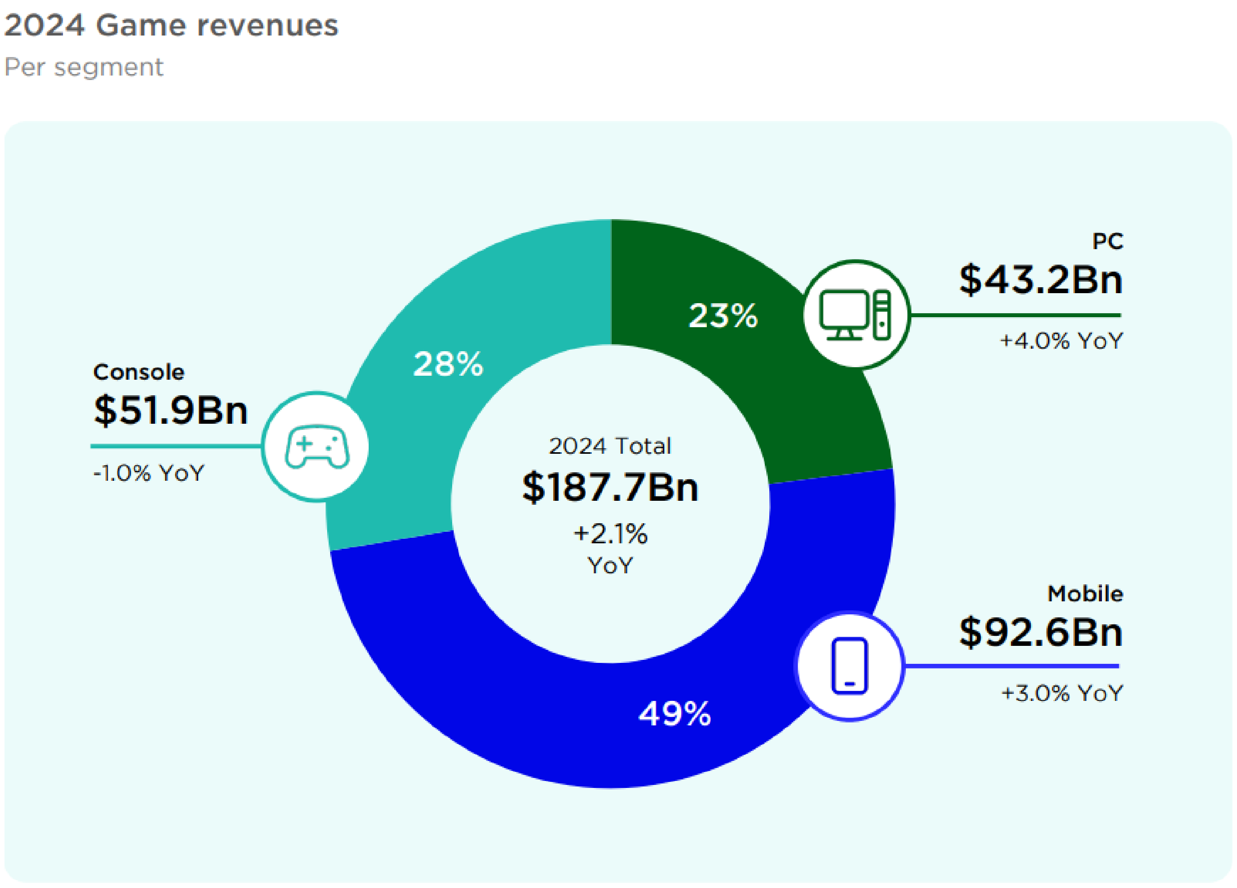
- **Align Creative and Technical Priorities**: Enables realistic planning without affecting creative processes
- **Enhances Stakeholder Management**: Establishes shared understanding and alignment through a switch to continuous visual and playable deliverables.
- **Support Sustainable Development**: Reinforces pacing strategies and buffers to reduce burnout and crunch.
- **Emphasis on Market Readiness**: Encourages responsible marketing by making use of a feature maturity metrics to ensure marketing is aligned with expectations.
- **Recovery and Post Launch Growth**: Provides advice on reframing negative game narratives and rebuilding player trust after launch.

Introduction

Background on Game Production Challenges

Development challenges are an inherent part of creating any project, however producing and shipping a game is not a simple task, it is plagued by common problems and is still very far from a healthy and unified work process. (Petrillo et al., 2009)

In 2024, the video game market revenue reached $187.7 billion (Newzoo, 2024) reinforcing the increase in player counts worldwide. However, with the major advances in technology, players are increasingly changing their demands and expectations of games. This challenge often forces developers to release unfinished projects or delay the release of their games. (Grewal et al., 2020)



(Newzoo, 2024)

As games become more ambitious, developers face increasing challenges in meeting deadlines, managing expectations and ensuring a successful launch. Feature creep is one of the most common challenges that projects face, where features are continuously added throughout development, leading to delays and increased strain on resources (Blow, 2004). A lack of a strong vision and project management causes games to become bloated with non-priority features taking up more development time ultimately meaning games release unfinished. (Gershenfeld et al., 2003)

Another major challenge is stakeholder pressure, particularly form publishers, investors and platform holders. External factors such as these often impose strict release schedules at the expense of quality. Publishers can force developers to release before its polished, leading to post launch controversies and financial implications. (Robichaud, 2023)

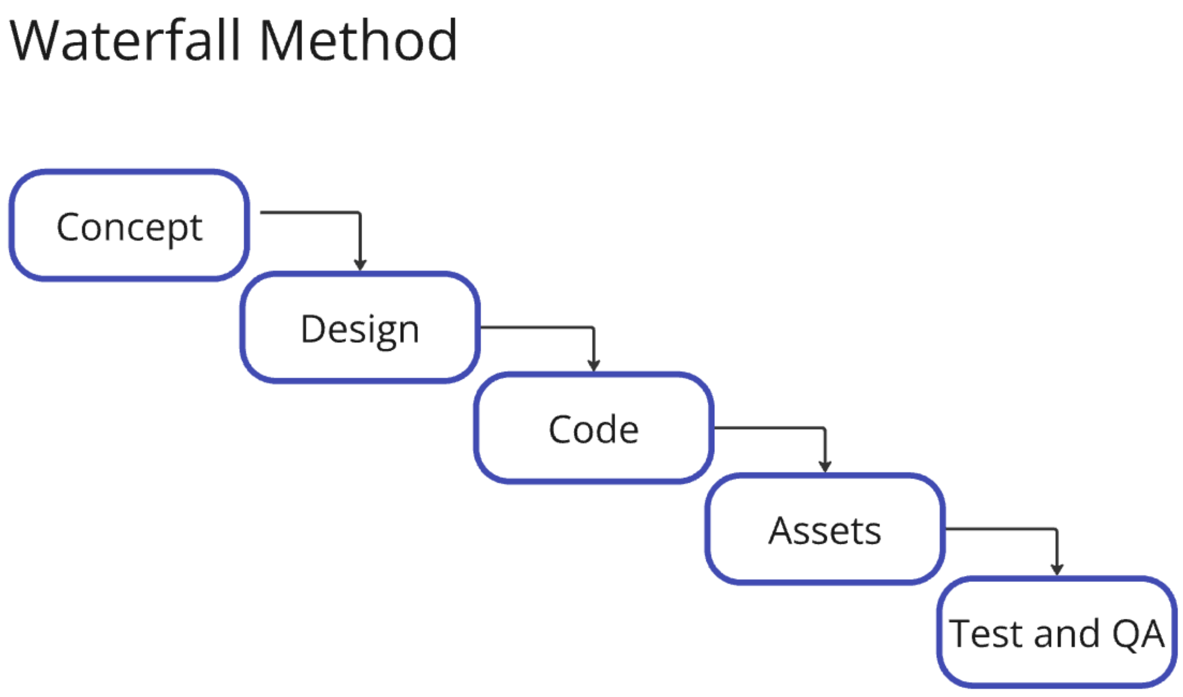
Additionally crunch culture is deeply rooted in the industry. To meet these tight deadlines, developers are forced to over excessive overtime which in-turn leads to burnout, reduced productivity and in most cases a lower quality output. Sustainable development practises such as realistic scheduling and effective resource allocation are essential in employee well-being and productivity. (Dyer-Witheford & de Peuter, 2006)

Finally, dedicated quality assurance and testing time is often reduced by over-running tasks leading to technical issues at launch. High profile cases such as No Man's Sky and Cyberpunk had inadequate testing resulting in considerable performance issues, missing features, and negative player reception. Having dedicated time that isn't moved or reduced to test the games features or integrating testing throughout the project development pipeline can stop game breaking bugs effecting a games launch.

In a 2020 GDC talk Oculus' Ruth Tomandl argues that problems with game vision, team organisation, partner decisions and market forces stem from ambiguity and unknowns in development (Game Developers Conference, 2020). The challenges outlined are crucial to consider when adapting Agile as I aim to prioritise realistic planning, stakeholder management, sustainable development practices and quality assurance – ensuring that games launch in a finished, functional state allowing for continued post-launch improvement without the need for damage control.

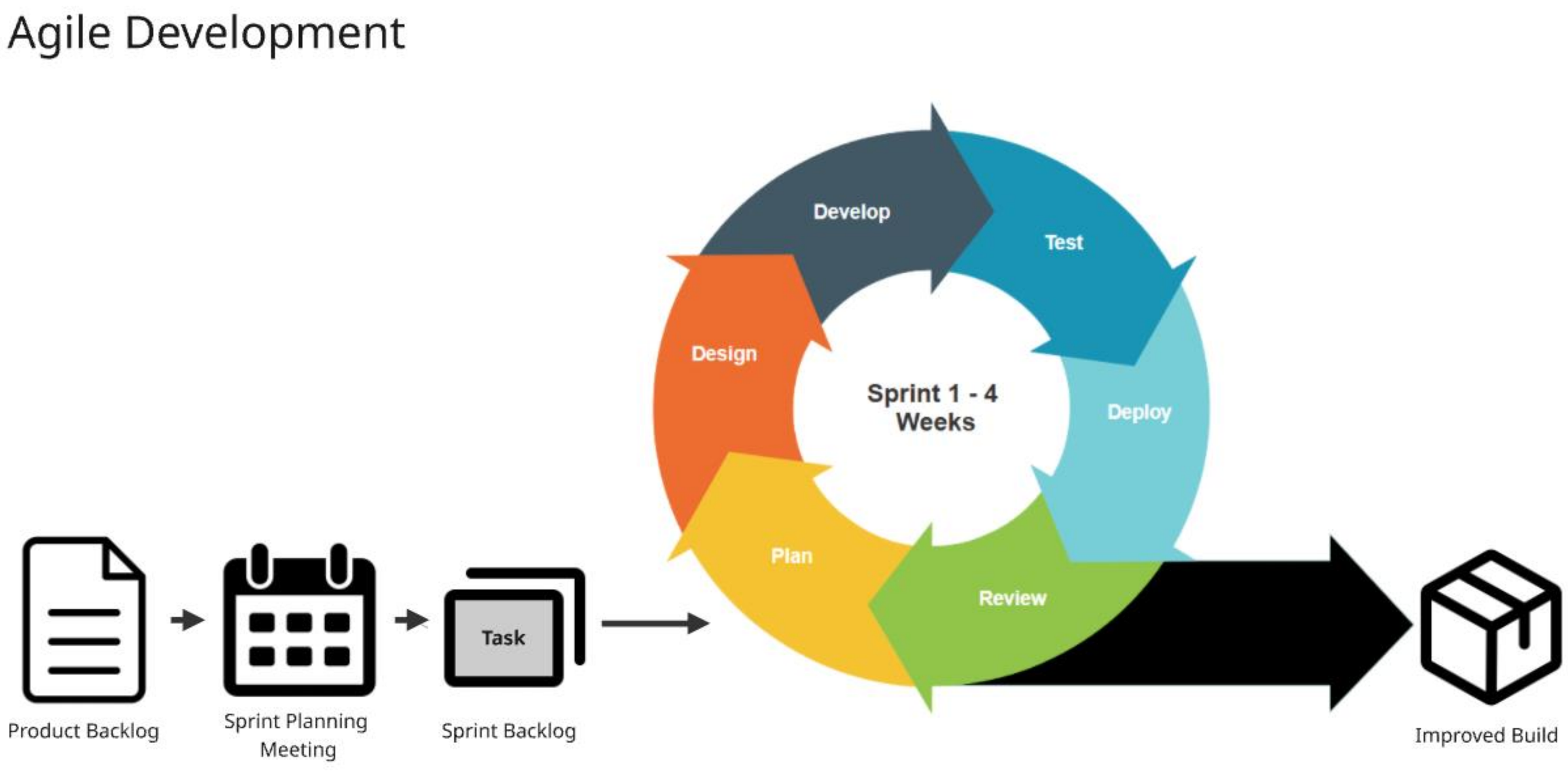
Current State of Industry Practices

Modern day game development is complex, necessitating a balance between creative flexibility and project management. The development landscape has changed drastically since the early days of ‘one person teams’ so it makes sense that industry practices have also changed. A single programmer could create a video game in a matter of months in the early days of the industry, but as user expectations increased and hardware development costs increased, an approach to handle the growing risk was identified. Many companies quickly adopted the Waterfall method which was commonplace in many other industries. (Keith, 2020)



The Waterfall method is based around the idea that everything in one section must be completed before moving onto the next – code doesn’t get started until all the design and concept is finished, because of the rigid nature of the method it had very little chances of succeeding in a flexible industry such as games. (Cohen, 2009)

In the past decade, the games industry has seen a significant shift as developers and producers have shifted from standalone titles and physical games to continuously updated live service models (Sotamma & Svelch, 2021). Indeterminately, studios release content for these existing games instead of developing new games or standalone sequels. Since these games don't have a defined end point to development Agile was brought in to address live-service games, Agile is derived from the software industry where the requirements cannot always be defined at the start of the project and is the most commonly used method today. (Cohen, 2009)



Agile development is built around close collaboration, meetings and fast iterative design cycles. The method at its core has each team member to focus on their own small task which allows them to create rapid high-quality work while being pieced together through the multiple design cycles.

Need for a New Framework

While Agile addresses the issues with un-defined project end goals, I think that it fails to address the critical game specific problems I outlined previously. The launch failures of both No Man's Sky and Cyberpunk 2077 highlights the need for a structured rework to Agile development that prioritises realistic development, transparent communication, and sustainable post launch support. While it is important to note both games did eventually recover, it took years of costly updates to regain player trust – which is something that is not feasible for most studios.

There have been numerous notable titles across the games industry that have suffered as a result of overambitious marketing, rushed production cycles, and poor stakeholder management. The adaptation that I aim to produce focuses on addressing ambiguity in development in order to ensure clear production milestones, effective communication between developers and stakeholders, as well as an emphasis on post-launch support. The framework aims to provide a roadmap for delivering a successful game launch while also offering a structured recovery strategy for games that release to negative reception. The goal of my adapted Agile goes beyond just preventing project failure but to create a system that allows studios to adapt to changes in the industry, maintaining long term player engagement without relying on post launch fixes to salvage a games reputation.

Case Study's

Cyberpunk 2077 - Case Study Summary

When Cyberpunk 2077 (CD Projekt Red, 2020) was announced by CD Projekt Red, it quickly gained traction and became one of the most anticipated games of all time. The game promised a vast open world, deep RPG mechanics and innovative graphics built for the next generation of consoles. All these factors contributed towards immense hype surrounding the game, bolstered by the studio's success with The Witcher 3: Wild hunt. However, when the game released in December 2020, Cyberpunk 2077 (CD Projekt Red, 2020) became one of the most notorious launches in game history, plagued by game breaking bugs, terrible performance on last gen consoles and under-delivering on multiple key features highlighted in the pre-release. This ultimately led to a severe backlash from fans who felt dissatisfied with the game which saw the game being refunded, lawsuits and even the game being pulled entirely off the PlayStation store.

In the face of this disastrous launch, CD Projekt red started a multi-year recovery effort, releasing major patches, offering free next-gen upgrades and introducing expansions that overtime helped to restore the games reputation. This case study examines the contributing factors behind Cyberpunks failed launch, the company's response and the steps taken to rebuild confidence in the players.

The launch and subsequent recovery of Cyberpunk 2077 (CD Projekt Red, 2020) highlighted several critical failings not just for CD Projekt Red but the games industry. One of the biggest takeaways is the importance of transparency when communicating to the players. The decision to withhold information from not only the players but project stakeholders about the games state on older generation consoles created unrealistic expectations, this contributed massively to the widespread backlash that the game received and gave stakeholders grounds for a lawsuit further damaging the company reputation. This situation reinforced the need for developers to provide accurate representation of their games including in any cinematic and gameplay trailers.

Another lesson is the danger of a mismanaged development and crunch culture. Reports suggested that Cyberpunk underwent years of mismanaged production with shifting design goals, unmanaged scope, and technical shortcomings. The issues on release of the game highlighted the negative consequences a long term fundamentally-flawed projection can have on both a game and the company's reputation. In stark contrast the Phantom Liberty DLC proved that when there is time taken to refine and polish a game, a team can a produce a game that is received well. The game's success story underscored the power of post-launch commitment and community management.

No Mans Sky - Case Study Summary

When No Mans Sky (Hello Games, 2016) was first revealed in December 2013, it promised to revolutionise the gaming industry with its procedurally generated world, limitless exploration and seamless multiplayer experience. Developed by small independent studio Hello Games, it quickly gained traction with its ambitious scope and pre-release marketing spearheaded by studio founder Sean Murray. When the game eventually released in August 2016, it became infamous for failing to deliver on a considerable number of key promises. This led to widespread dissatisfaction and backlash from players who felt they had been deceived by the marketing.

Despite this launch, Hello Games persevered and instead of abandoning the project committed to multiple years of updates and DLC transforming the game into the expansive and dynamic game they had originally envisioned. This case study explores the failures of No Mans Sky (Hello Games, 2016) at launch, the steps taken by Hello Games to recover and the lessons that can be learned from its eventual success.

The release of No Mans Sky (Hello Games, 2016) highlights the importance of managing player expectations and hype, transparent marketing and post-launch commitments. The key takeaway is that every feature you mention is a promise when communicating pre-release. Hello games fell victim to creating an image of the game that was unrealistic and more ambitious than what was delivered on release, leading to widespread anger and disappointment among fans. This highlights the need for studios to be transparent about games features over making aspirational features and promises that are unrealistic for release.

Framework Overview

Agile is built around predictability, iterative development and clear requirements, but game development is often

- Highly creative and subjective
- Ambitious (it’s hard to define what “fun” is)
- Dependant on multiple different teams syncing up (art, tech, design)

The proposed modifications establish a framework better aligned with the unique characteristics of game development. The following sections break down the core principles and concepts and examine how each adaptation specifically addresses the challenges of ambiguity, realistic development/marketing, and effective stakeholder engagement.

Core Principles and Concepts

Dealing with Ambiguity in Game Development

Ambiguity is an inherent part of any project, especially during the early stages of development such as conceptualisation. In contrast with common software products, that often have well-defined user goals and functionality, games have a subjective and dynamic objective: *producing entertaining and engaging experiences*. As a result, features might not have a measurable success criteria and game visions are often hard to define and communicate to teams. (Game Developers Conference, 2020)

The most damaging result of ambiguity is Churn, this is where you are doing a bunch of tasks but not getting anywhere, usually because your vision is not clear enough to know if you’re getting closer to it.

Problems to Watch for:

- Your Game doesn't change - weekly design reviews are not showing progress, milestones slip because people are blocked by something they cant define.
- Rat-holing or bikeshedding - focusing on small problems that are easy to fix while ignoring major problems that are hard to fix.
- Throwing away work

(Yassine et al, 2003)

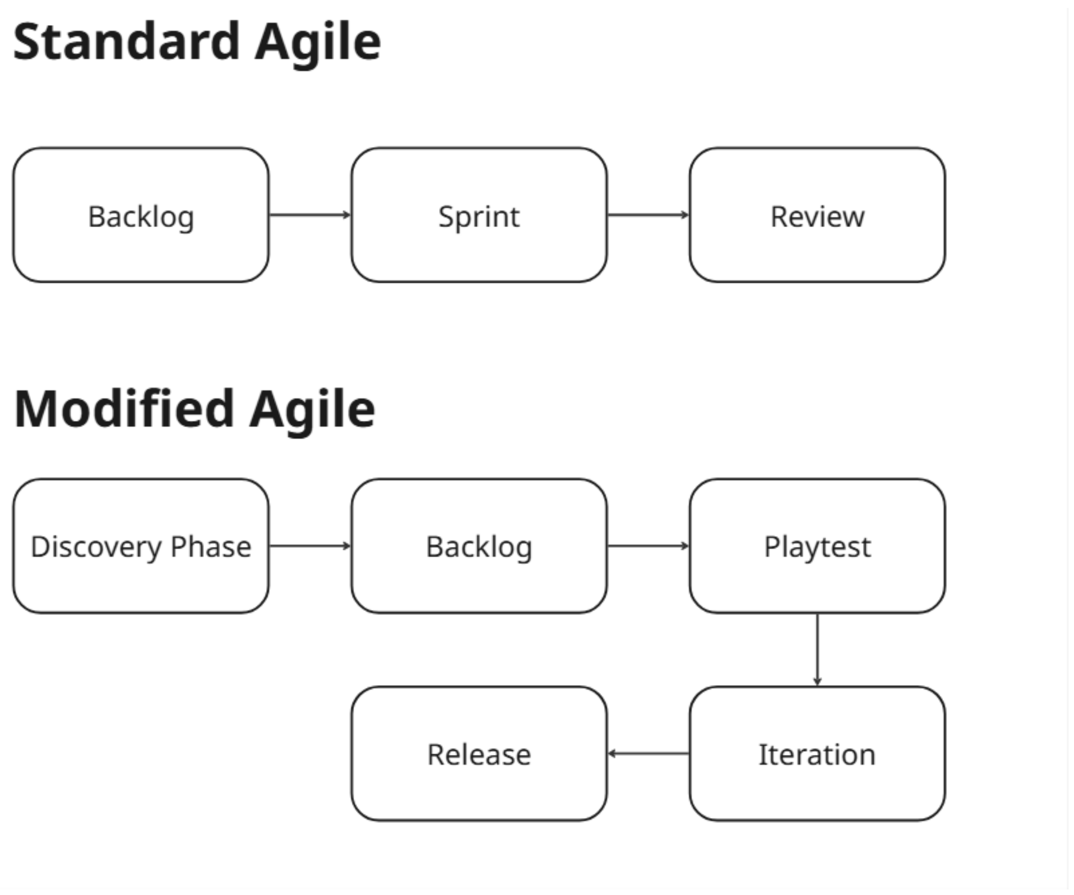
Traditional agile frameworks, such as scrum emphasize well-defined user stories, and measurable outcomes within time-boxed iterations. While effective for traditional software, this structure can be too ridged in practice where ideas need to evolve before becoming actionable. (Keith, 2020)

To achieve this several adaptations to Agile are proposed to better accommodate ambiguity in game development.

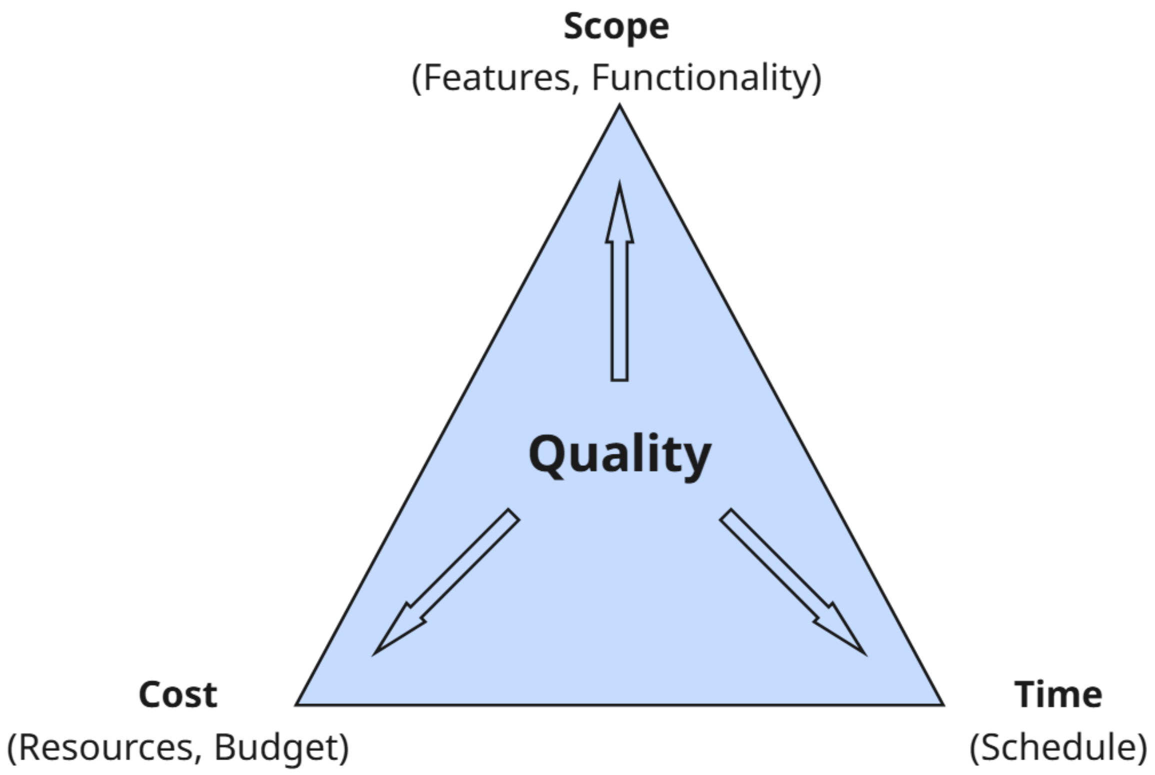
Discovery Phase Prior to Sprint Planning

Before features are added to the products backlog, a “Discovery Phase” can be incorporated into project workflow. This phase acts as a structured period for ideation, exploration and experimentation. Rather than focusing primarily on deliverables, the objective is to validate core gameplay concepts, mechanic, art styles or narrative design through rapid prototyping and collaborative discussions. (Dunlop, 2014)

- This phase may include short-lived "concept spikes," narrative developments, or mechanic mock-ups.
- Outcomes from this phase may be either discarded or refined into actionable backlog items.



It is important to note the longer the time spent in this phase the lower the level of ambiguity there will be going into development, however in a commercial studio environment deciding how much time to dedicate to this phase is influenced by factors such as development time, cost and scope.



The iron triangle is a framework that helps you understand your constraints and therefore figure out how much time you can dedicate to discovery.

If your budget is small, it means your schedule needs to be shorter. Also, if you have to include many features, there's less room for ambiguity. This lack of ambiguity means there's less space for innovation and creativity leading to less time in discovery. Instead, you have to focus mainly on executing the project.

On the other hand, if you have all the time and money you need, you can afford to prototype and spend time in pre-production without rushing.

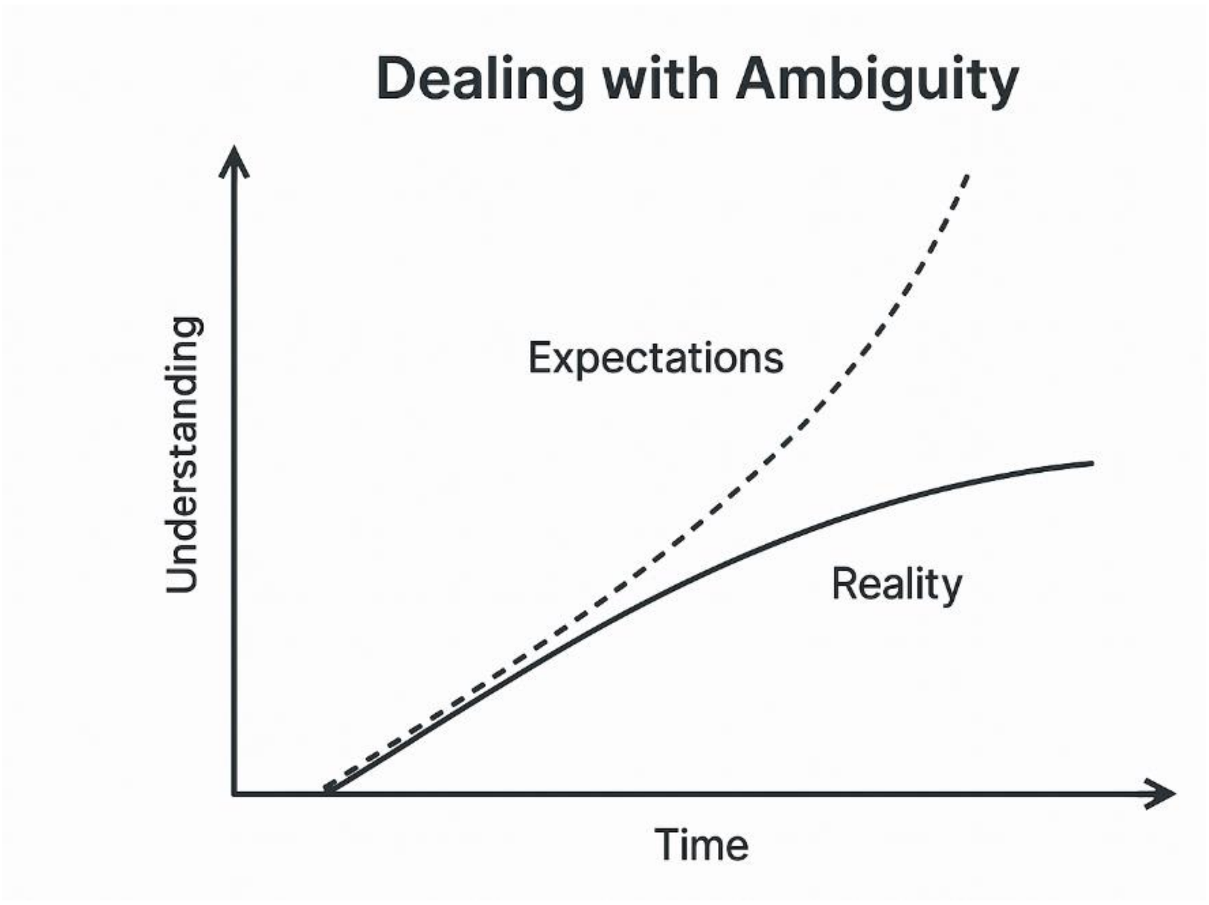
Redefining Requirements as Hypotheses

In early development stages, requirements should be reframed not as fixed specifications but as hypotheses to be tested. (Olsson & Joelsson, 2019)

For Example:
“Implement grappling hook traversal system” may be instead defined as explore whether grappling hook mechanics enhance player mobility and engagement.”

- Acceptance criteria for tasks shifts from functional completeness to experiential validation (e.g., "Is it fun?", "Does it support core gameplay goals?", “Does it align with the games pillars”).

- This approach supports iteration and informed decision-making, particularly during pre-production and early prototyping.



Comparison between expected and actual reduction of ambiguity over time. While traditional development frameworks suggest a rapid decline in ambiguity shortly after project start, the reality of creative processes—particularly in game development—demonstrates a gradual clarification of ideas. This discrepancy highlights the need for *an adapted Agile methodologies that accommodates uncertainty during early stages of production.*

Using Playtesting as a Feedback Mechanism

Continuous playtesting is essential for reducing ambiguity over time. Incorporating frequent, lightweight internal playtests within sprints allows teams to evaluate whether evolving features are achieving intended emotional or gameplay effects. (Bromley, 2023)

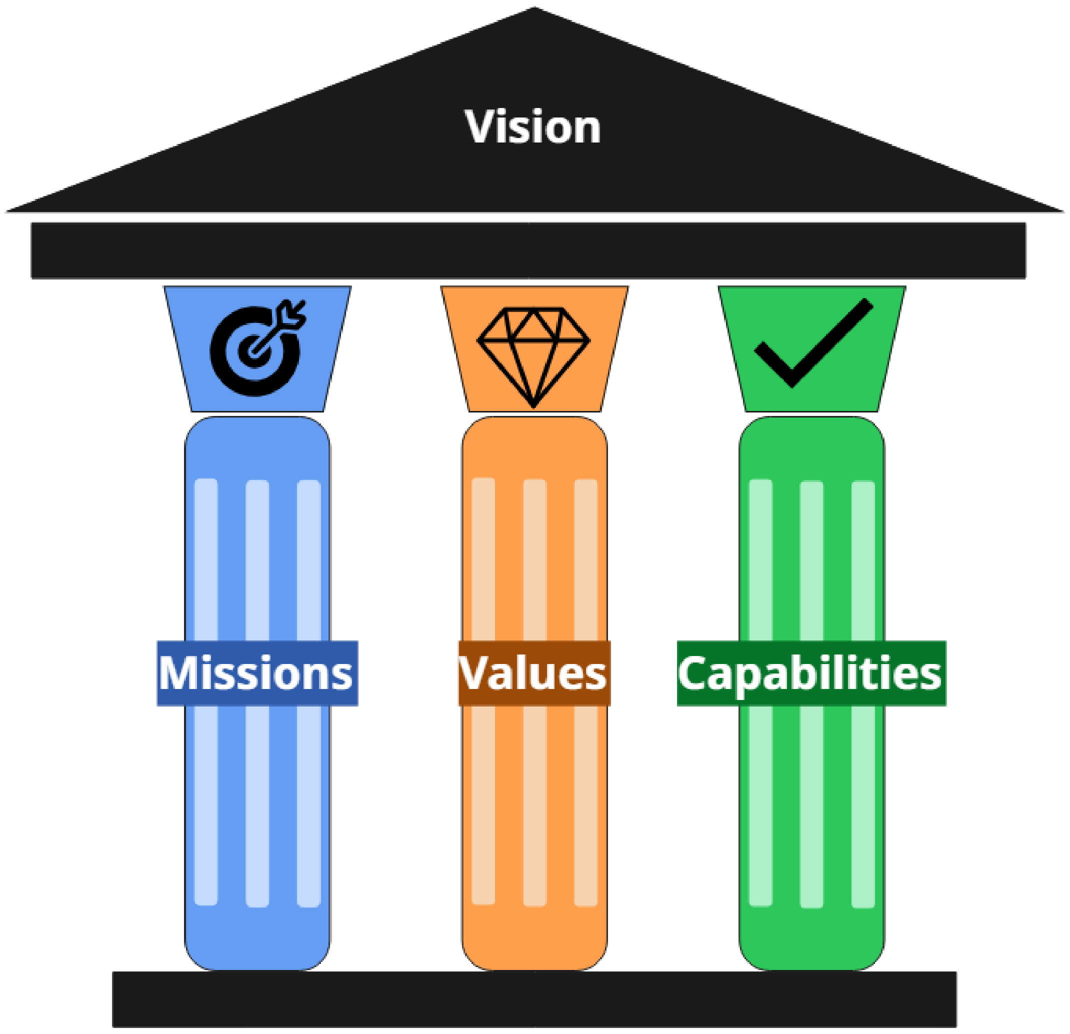
- These playtests can act as qualitative sprint reviews, especially when progress is intangible (e.g., mood, pacing, immersion).
- Feedback loops should include not just developers but also stakeholders and target audience proxies when possible.
- Team can get into the habit of finishing tasks, and something can be built to play test (this helps to identify what is good about the games vision, what are the things that took up a lot of time and effort but aren't that good)

Using Pillars to communicate vision

Effective communication of creative vision is essential in any project, particularly in Agile environments where interdisciplinary teams iterate constantly.

One widely adopted approach to maintaining shared vison across diverse teams is the use of design pillars. At its core pillars are a distillation of your GDD to a couple of bullet points that are easy for your team to communicate and understand. (Pears, 2017)

Using pillars to communicate what the game will be like - if you're working on a game use the pillars as a yard stick to think about the work you're doing - Use the pillars to determine if what you are working on directly supports the vision if it doesn't it's probably extraneous and you don't need to do it. (Game Developers Conference, 2020)



This diagram visualizes the foundational structure for communicating and sustaining creative vision throughout game development. The overarching "Vision" represents the desired player experience and long-term creative direction. This vision is supported by three key pillars: ***Missions****, which define the game's goals and thematic intent;* ***Values****, which articulate the emotional and experiential qualities the game aspires to deliver; and* ***Capabilities****, which represent the core strengths of the development team, tools, and technologies.*

These pillars serve as guiding principles in Agile workflows, ensuring cross-functional teams remain aligned even as features evolve. By embedding these pillars into planning, review, and design discussions, teams can navigate ambiguity and iteration without compromising the integrity of the game's original vision.

Emphasising Realistic Development

Game development is especially liable to scheduling conflicts, scope creep, and burnout – often resulting in "crunch" periods of development. While traditional Agile frameworks do offer iterative structure, they fail to inherently prevent these issues. Agile in game development studios is often misapplied to accelerate delivery instead of to foster a sustainable development environment. Addressing realistic development timelines requires adapting agile in a way that better accommodates for creative driven iterative cycles (Jozin, 2019)

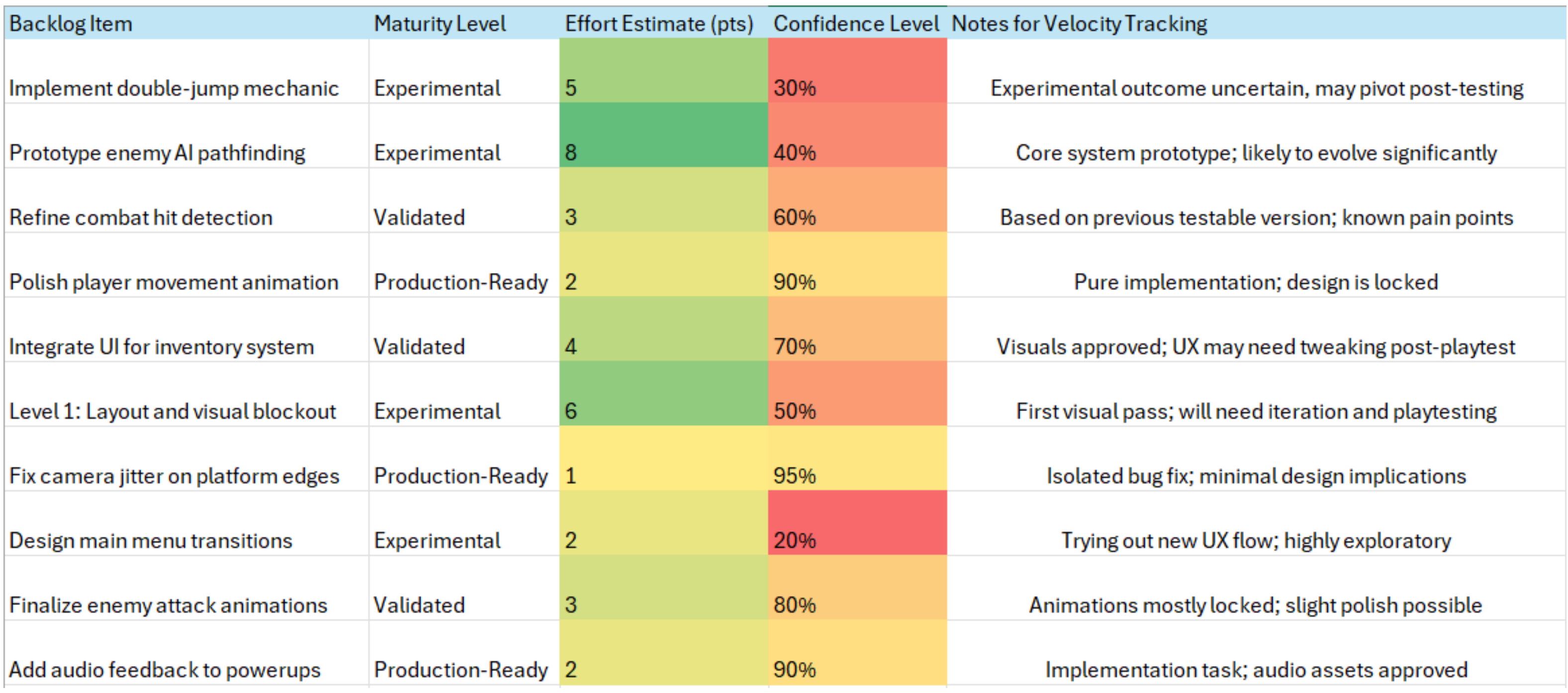
The proposed changes explore how Agile can support a more realistic pace of production in the game development context.

Velocity Tracking based off Feature Type and Creative Maturity

Traditional Agile velocity metrics assume tasks are similar in terms of complexity or can be estimated with confidence (Atlassian, 2025). However, in game development, tasks can range from straightforward such as implementing basic UI features to tasks with high levels of uncertainty such as prototyping new game mechanics. On top of this game development often lacks linear progression and there is a fine line between temporary exploration and permanently lost.

In order to address this:

- Teams should classify backlog items based on **feature maturity** (e.g., experimental, validated, production-ready).
- Task estimations should reflect **confidence levels**, not only effort, and avoid penalizing experimentation that fails constructively.
- Velocity metrics can then be contextualized rather than strictly compared across sprints.



Backlog Item	Maturity Level	Effort Estimate (pts)	Confidence Level	Notes for Velocity Tracking
Implement double-jump mechanic	Experimental	5	30%	Experimental outcome uncertain, may pivot post-testing
Prototype enemy AI pathfinding	Experimental	8	40%	Core system prototype; likely to evolve significantly
Refine combat hit detection	Validated	3	60%	Based on previous testable version; known pain points
Polish player movement animation	Production-Ready	2	90%	Pure implementation; design is locked
Integrate UI for inventory system	Validated	4	70%	Visuals approved; UX may need tweaking post-playtest
Level 1: Layout and visual blockout	Experimental	6	50%	First visual pass; will need iteration and playtesting
Fix camera jitter on platform edges	Production-Ready	1	95%	Isolated bug fix; minimal design implications
Design main menu transitions	Experimental	2	20%	Trying out new UX flow; highly exploratory
Finalize enemy attack animations	Validated	3	80%	Animations mostly locked; slight polish possible
Add audio feedback to powerups	Production-Ready	2	90%	Implementation task; audio assets approved

Confidence Level (%) reflects how likely the team believes the estimate will hold. Lower confidence = higher volatility.

Velocity Contextualization:

- Sprints heavy in *experimental* items will yield less predictable velocity.
- "Failure" in an experimental task isn’t waste—it informs design and moves features toward validation.

Sprint Planning Strategy:

- Mix maturity levels to balance creativity and deliverability.
- Track *confidence-weighted velocity* (e.g., 8 pts @ 40% = 3.2 effective velocity).
- Retro insights should focus on *why* an item over/underperformed; not just how many points were completed.

Development Buffer Time

Creative processes such as games are non-linear; ideas may fail or evolve unexpectedly. Introducing creative buffers – dedicated time for exploration, ideation, polish or even failure – within sprints or between milestone deliverables can help alleviate the impact of creative uncertainty. (Tromp, 2022)

- An Example of this could include 1-2 days per sprint set aside for iteration, game feel tweaks or narrative improvements
- It's important to emphasise that this time is not "slack", but an essential buffer that addresses the experimental nature of creative development.

Flexible Scope, Fixed Cadence

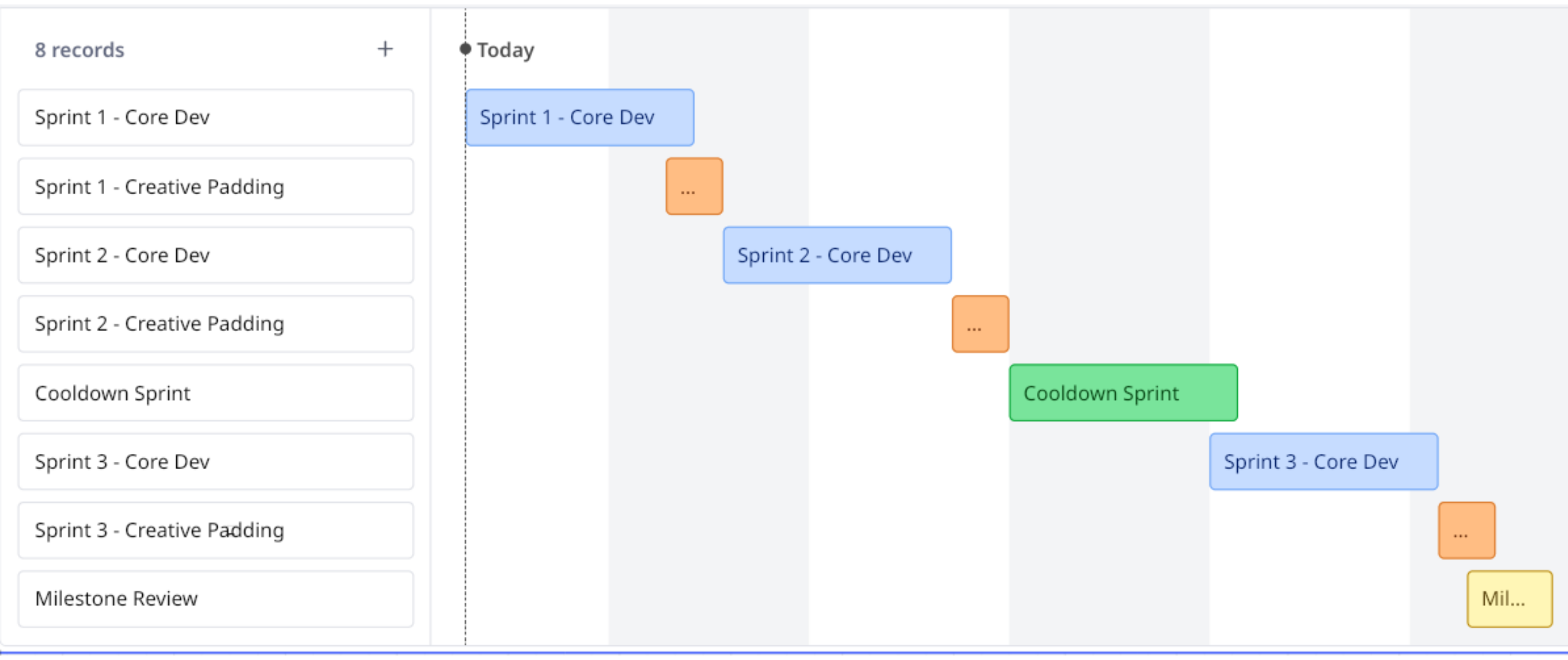
Instead of imposing a strict fixed scope per sprint (which in many cases can lead to overcommitment and crunch) Agile adapted for games can emphasise fixed cadence with adjustable scope allowing teams to deliver work at a consistent rate, with the understanding that features may be scaled, split of deferred based on emerging issues. (Rubin, 2012)

- Using the Minimum Viable Product definitions on a sprint level to determine what must ship and what can shift
- Feature roadmaps should be flexible enough to accommodate re-prioritization based on creative feedback and the results of playtesting.

Effective Milestone Planning

Agile development often fails to encapsulate long term planning (Ittycheria, 2019), whereas game development often has large milestones such as (Alpha, Beta, Gold) and a lot of the time these are externally imposed. A hybrid approach is necessary in order to address both short- and long-term development.

- Break down major milestones into internally defined Agile goals.
- Plan around *functional slices* rather than layers—for example, creating one "complete" mission or level rather than building all systems in parallel.
- Incorporate **"review and adjust" sprints** post-milestone to allow teams to decompress, assess what worked, and reallocate resources if needed.



Sprint 1 - Core Dev
Sprint 1 - Creative Padding
Sprint 2 - Core Dev
Sprint 2 - Creative Padding
Cooldown Sprint
Sprint 3 - Core Dev
Sprint 3 - Creative Padding
Milestone Review

Core Dev Sprint Cycle (1-2 Weeks)

- **Work Phase (Days 1–8)**
  The majority of the sprint is dedicated to typical Agile tasks: implementing features, testing systems, refining design work. This phase follows standard Agile ceremonies (daily standups, task board updates), but with an emphasis on mixing feature maturities (e.g., experimental + validated).

- **Creative Padding (Days 9–10)**
  The final 2 days are *deliberately reserved* as "creative padding":
  - Space for playtesting, tuning game feel, experimenting with mechanics.
  - A safe zone for failure, iteration, or even just reflection.
  - Prevents over-scoping by providing margin for unexpected design challenges.

Inspired by studios like Supergiant Games, which emphasize "no crunch" and value intentional polish time over frantic delivery.

2. Post-Sprint Cooldown (Optional)

- **Cooldown Phase (Every 3–4 Sprints)**
  After 2–3 sprint cycles, a cooldown sprint (or "breather sprint") is introduced:
  - Dedicated to bug fixing, documentation, knowledge sharing, and retrospection.
  - Allows the team to pay down **technical debt** and **creative debt** (features that "work" but need refinement to feel right).
  - Acts as a pressure release valve to avoid long-term fatigue.

Reflective of practices adopted post-Cyberpunk 2077 by CD Projekt RED, where teams introduced structured rest cycles and more realistic milestone pacing.

3. Milestone Integration

- **Milestones Are Flexible and Feature-Focused** Rather than fixed by date, milestones are defined around "vertical slices" of the game (e.g., one fully playable level or combat prototype).
  - Creative buffers ensure these milestones are hit *without sacrificing quality or health*.
  - Feedback from each milestone feeds into reprioritizing the backlog.

Stakeholder Management in Agile Development

In order for a game to have effective stakeholder management it must extend beyond traditional agile roles such as product owner and customer, this is necessary as in the context of games stakeholders include a complex list of groups and individuals these range from executive producers and marketing teams to platform holders, community representatives, and even the players during open development or early access periods.

The proposed changes explore how through Agile teams can support a more efficient communication plan with project stakeholders.

Redefining the roles of Stakeholders

Game development when compared to traditional software development operates under greater creative ambiguity, this makes it difficult for stakeholders to specify fixed requirements (Marklund et al, 2019). This often leads them to demand strict feature commitments and over imposing on developers negatively effecting games development and release.

To help address this stakeholders should be encouraged to:

- Align towards *experience goals* (e.g. Make movement feel more responsive)
- Be Involved in co-creating *player personas* or *emotional journey maps* in order to act as shared decision-making tools
- Actively be aware of and encourage the iterative nature of game development, where outcomes evolve through playtesting and feedback

These shifts in thinking reframes stakeholders as valuable tools to collaborate with in discovery, rather than clients of fixed outputs.

Focusing Communication on Milestones, Not Dates

In order to manage expectations and reduce misalignment with project stakeholders:

- Replace rigid delivery dates with "Experience Milestones" – e.g. "First Playable Combat Loop", "First Complete level with Audio"
- Use visual milestone maps or timeline trackers that show progress by quality and completeness, not just task count
- Integrate playable builds or video walkthroughs as stakeholder deliverables, fostering shared understanding of progress and risk

Embedding Feedback without impacting teams

Agile is built around continuous improvement, however too much reactive input from stakeholders can destabilize creative progress. In order to strike a balance between these:

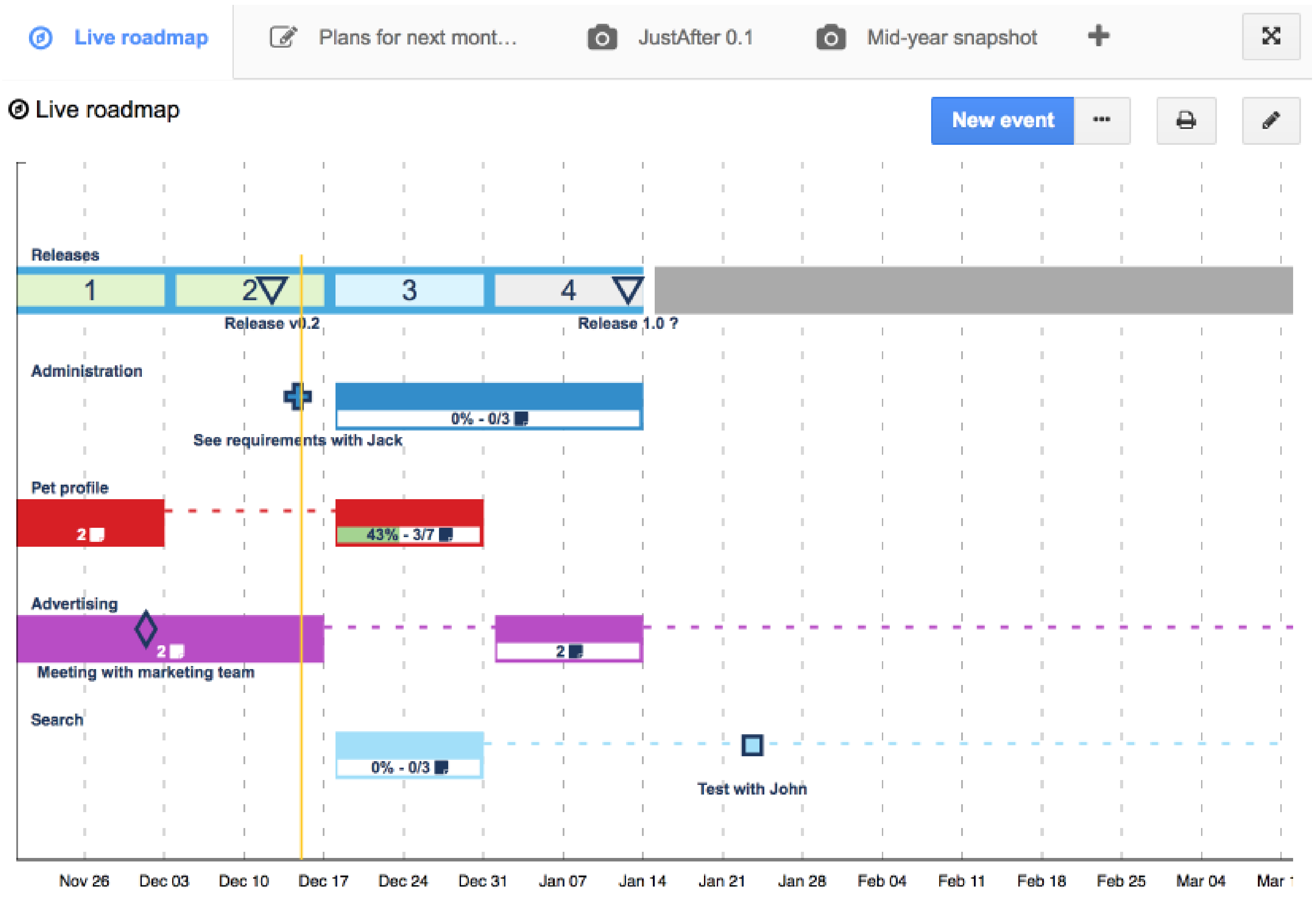
- Introduce feedback windows (e.g.mid-sprint- or at review milestones) to prevent constant changes
- Categorise stakeholder feedback (e.g. critical, nice to have, future considerations, not relevant) in order to manage stakeholder input
- Maintain a single or small point of stakeholder translation – a creative producer of department lead – to avoid feedback fragmentation.

(LeanWisdom, 2023)

Managing External vs Internal Stakeholders

External stakeholders such as publishers and investors require transparency in communications, while internal stakeholders such as QA, narrative, art need creative alignment. To bridge both:

- Implement layered updates — high-level progress for execs, detailed design breakdowns for internal teams
- Use living roadmaps that reflect shifting creative priorities, with justification for changes clearly logged



(IceScrum, 2025)

Key Elements in a Living Roadmap

1. Evolving Feature Blocks

The timeline is broken into segments such as:

- *Combat Iteration Phase*

- *Narrative Design Sprint*
- *Visual Polish Pass*

Each block shifts slightly forward or backward depending on creative discoveries or user feedback.

These shifts are not errors, but signs of an adaptive creative process. Instead of being locked into static deadlines, the roadmap shows *creative priorities adjusting dynamically*.

2. Change Justification Markers

Beside any shifted or altered block, a small flag or annotation could appear with:

- Reason for Change - "Combat loop required rebalancing after playtest"
- Impact Summary: "Visual polish delayed by 1 sprint to allow audio integration"

These annotations create a log of transparent decision-making, useful both for internal accountability and external stakeholder trust.

3. Feedback Loop Integration

The roadmap might feature icons for:

- Milestone approvals
- Playtest checkpoints
- Stakeholder reviews

This signals that change is driven by actual testing and stakeholder alignment, not arbitrary rescheduling

Addressing realistic Marketing in Agile Game Development

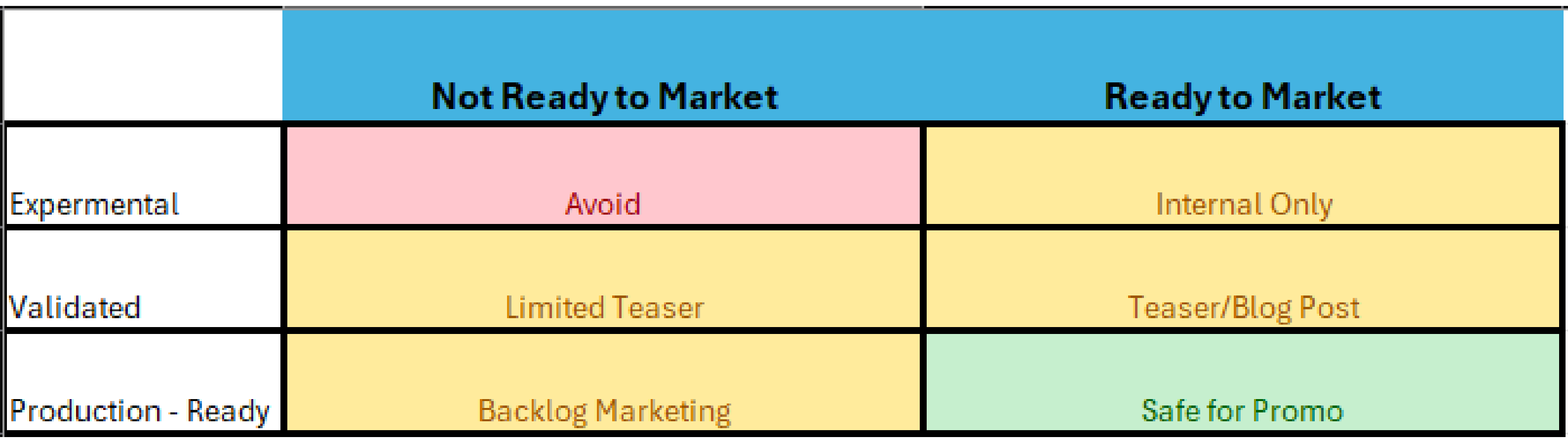
How a game is marketed is often the key to its success generate massive hype and anticipation surrounding a game, however it can also be its failure with many high-profile cases such as No-Man's Sky and Cyberpunk 2077 highlighting that every feature you mention in marketing is a promise to fans. These marketing expectations are shaped by early trailers, internal pitch decks, and milestone promises made to publishers or platform holders. (Wesley & Barczak, 2010)

Due to the creative non-linear nature of game production, it makes it difficult to guarantee specific features or polish levels months or even years in advance and traditional Agile frameworks often overlook the pressures of marketing cycles, hype management and public perception – all of which are key contributing factors in the games industry. In order to address this game studios should push to adopt marketing - aware development pipelines that are flexible, transparent and facilitate the creative nature of games.

Aligning Marketing Milestones with Development Progress

Instead of using marketing based on early projections such as trailers at major expos or fixed launch dates, games studios should:

- Base marketing aims on “Experience Readiness” — i.e., only promoting features that have passed internal playtest thresholds for stability and player impact.
- Create cross-functional milestones early on in development that include input from both developers and marketers to assess whether a gameplay feature or system is *both ready and demonstrable*.
- Encourage vertical slice validation before promotional campaigns — a playable, polished section that accurately reflects final quality and ensures promotional items align with actual progress.



	Not Ready to Market	**Ready to Market**
Expermental	Avoid	Internal Only
Validated	Limited Teaser	Teaser/Blog Post
Production - Ready	Backlog Marketing	Safe for Promo

Shows how features can be categorized by their development maturity and marketing readiness. Visually justifies why not everything should be promoted, even if it’s in development.

Avoiding the Pitfalls of Overpromising

The case of CD Projekt RED's Cyberpunk 2077 highlights the implications of an environment where marketing outpaced development. It is essential for any studio that a case like this doesn't happen, in order to achieve this, it requires:

- Marketing teams should be involved in feature maturity scoring, not just narrative or brand messaging.
- Reinforce the use of feature volatility ratings with the marketing teams *(outlined prior in the Realistic development section)* so that marketing understands the risk of delay or removal.
- Establishing effective communication line in which internal delays or descoped features are logged transparently and communicated to marketing partners early.

Make Marketing Playtest Driven

Game development is Agile driven, so it logically follows that the marketing for those games should also become playtest-informed and adaptable.

- Take forward feedback from play testers and internal QA into marketing messaging - if players are especially connecting with a feature, spotlight that in campaigns.
- Use demos and closed alphas as milestone marketing, rather than premature trailers that showcase incomplete systems.
- Include marketing teams in sprint reviews or vertical slice playthroughs so they are grounded and understand the games actual experience.

Communication with Publishers and Platform Holders

Publishers and Platform holders such as Sony, Xbox and Steam are major stakeholders (Chandler, 2020), often demanding comprehensive well-polished feature lists or fixed deliverables for store-front pages or promotional deals. In order to effectively manage these relationships teams should –

- Provide living feature roadmaps (*Similar to those outlined in Stakeholder Management)*
- Focus on milestone review summaries that include development confidence ratings alongside media assets - "Feature X is 80% stable but may shift based on player feedback in the next sprint"

- Align publishing deliverables including box art, pre-order descriptions with validated, tested features only.

Implementation Plan

The implementation plan for this adapted framework is designed to be straightforward due to the foundation of well-established agile methodologies. By building on proven practices like iterative development, multidisciplinary collaboration, and backlog driven planning, the framework aims to maintain familiarity with teams while integrating the changes for the unique demands of game production. The proposed approaches can be layered onto existing workflows with minimal disruption, another benefit is that not all features have to be implemented at once allowing teams to adopt the framework incrementally while preserving production momentum.

Steps for Adopting the Framework

Step	Description
1. Framework Introduction & Training	Conduct onboarding sessions with team leads and stakeholders. Introduce core concepts: creative padding, experience milestones, feature maturity, marketing alignment.
2. Establish Cross – Disciplinary Alignment	Ensure game design, production, marketing, and QA leads define shared terms for “quality,” “completeness,” and “experience goals.”
3. Integrate New Agile Artifacts	Introduce new planning tools: • Creative maturity-based backlogs • Feature volatility metrics • Living roadmaps • Marketing-readiness scoring
4. Adapt Sprint Loops	Update sprint planning to include creative buffers and stakeholder demos. Add milestone reviews (e.g., vertical slice readiness, first playable loops).
5. Test with Vertical Slice	Test the framework on a limited portion of the game such as core combat loop or tutorial.

	Collect feedback and refine processes.
6. Full Rollout and Retrospective Loops	Roll out framework studio-wide, with quarterly retrospectives to refine practices, velocity metrics, and marketing alignment.

Stakeholder Roles and Responsibilities

Role	Responsibilities
Game Director/ Creative Lead	Outline player experience goals, Prioritise features based on emotional and game play impact.
Producers	Ensure that the frameworks planning structure is implemented, creative buffers and experience milestones are in place.
Developers	Estimate features based on confidence not only effort, flag volatility early.
QA / UX Teams	Conduct playtests that focus on validating experience quality not only just bug counts. Provide relevant feedback that informs both development and marketing
Marketing Teams	Coordinate with development teams to ensure that marketing is based off validated, stable features.
Stakeholder Pannel (Publishers)	Engage with milestone reviews and roadmap updates, Receive playable builds and walkthroughs.

Timelines and Milestones

Month	Milestone
Month 1	Team training, onboarding materials, role definitions complete
Month 2	Initial implementation begins on some feature group (tutorial + UI)

Month 3	First playable experience milestone: e.g., “Core Combat Loop with Placeholder Assets”
Month 4	Stakeholder marketing alignment meeting: Feature readiness scoring in effect
Month 5	Expanded implementation to art, audio, and narrative teams
Month 6	First “Living Roadmap” presented with risk levels and justifications
Month 9	Vertical slice milestone, with integrated QA/UX and marketing input
Month 12	Studio-wide framework review: success metrics collected, refinements made

Adjust Timeline to your project length and development model.

Risk Assessment and Mitigation

Potential Risks and Strategies for Risk Mitigation

This framework is built off agile foundations and is tailored to address common challenges in game development, however this adaption does carry inherent risks and recognising and proactively planning for this is imperative to the successful implementation.

Risk 1 – Misinterpretation of Creative Buffers as Slack Time

There is a risk that team members or external stakeholders may perceive creative padding or buffer zones as inefficient and un-needed down time and may push against their implementation.

Mitigation Strategy:

- Ensure that the purpose of creative buffers are documented and communicated effectively as tools to manage ambiguity and allow for constructive iteration
- Align with leadership and stakeholders on pacing expectations (using visual tools like experience milestone maps)

Risk 2 – Resistance to Non-Linear Road mapping

Stakeholders may be resistant to shift from traditional fixed delivery dates towards living roadmaps and experience driven milestones.

Mitigation Strategy:

- Inform Stakeholders of the value of Experience Milestones over rigid dates - "First Full Combat Loop" over "Level 1 Done"
- Provide consistent playable builds/ walkthroughs to visually communicate progress and maintain trust.
- Include Stakeholder feedback sessions at regular intervals build into the project's roadmap

Risk 3 – Marketing Promoted Features not ready for Exposure

Marketing teams run the risk of promoting features that still in early development stages, setting unrealistic expectations for the player and damage company creditability.

Mitigation Strategy:

- Ensuring that the Marketing vs Feature Readiness is based on feature maturity
- Encouraging marketing team participation in milestones reviews aligning them with the development reality
- Maintaining a shared promo ready backlog by both dev and marketing leads

Risk 4: Cross-Disciplinary Misalignment

With game studios often being made up by different large-scale Design, engineering, art, and marketing teams the definition of "complete" between departments may become blurred or distorted causing delivery issues and delays.

Mitigation Strategy:

- Ensuring that design pillars are established early and communicated effectively, Using shared experience goals to define quality
- Include creativity maturity levels (Experimental → Validated → Production-Ready) in task tracking between teams
- Focus on UX and QA in early-stage feedback loops to close the communication gap between intention and execution.

Evaluation and Feedback Mechanisms

Establishing an effective evaluation and feedback mechanism is imperative to ensure that the framework remains relevant, effective and adaptable. These mechanisms enable continuous improvement by using evidence, experience and stakeholder engagement to ensure that development is grounded. (Markiewicz & Patrick, 2015)

Framework Testing and Iteration

Due to the limitations of this proposal, in order to validate the effectiveness of the adapted agile framework in a real-life game development scenario, an iterative testing approach is recommended.

- **Pilot Implementation:**

  Rolling out the framework in a controlled pilot phase either with a smaller team or vertical slice of the project. Focus on testing creative pacing, milestone flexibility and using feature maturity with backlog features.

- **Review Stages:**

  Use a mix of velocity tracking, sprint reflections and developer sentiment surveys to see performance. Ensure that any creative friction, misalignment or positive outcomes are documented.

- **Iterative Improvements:**

  Apply insights from the pilot implementation to refine processes, tools and templates before moving to a full-scale roll-out.

- **Version Control of Framework Elements:**

  In order to have efficient documentation to refer back to, maintaining a living document model of the framework. This includes logging all the updates to track its evolution across projects.

Post – Launch Recovery: Responding to Negative Reception

This framework has focused on addressing the issues that can lead to a negative release in game development before they happen, However, it fails to acknowledge that games may have already been released and a post-launch plan may need to be implemented.

The post launch phase is critical, especially when a game has entered the market to widespread criticism or underperforms when compared to user expectations like the cases of Cyberpunk 2077 and No Mans Sky. Rather than viewing this as a failure, the idea of post-launch recovery should be approached in order to provide an opportunity to restore player trust, realign the development teams, and improve the projects long term health.

Stabilisation Phase

When a game releases to negative player reception, whether due to technical issues, missing features or underdelivering on player expectations – the most urgent phase is the stabilisation. This involves assessing the damage, prioritising critical fixes, and establishing effective communication between the team and community about the progress. (Georgiou, 2024)

Rapid Response and Issue Triage

The immediate objective post-launch is to prevent further damage by resolving high-priority issues quickly. This is best managed using a cross-discipline “Triage Team” formed of leads from engineering, QA, design, and production. Their role is to:

- Establish a Triage Board by severity such as game breaking bug, progression blocks, UX confusion.
- Prioritise game fixes based on real player impact, using data, support tickets, and community reports to identify most prevalent issues.
- Assign response windows - hotfix 1 within 48 hours, patch in 7 days and communicate them openly with teams and community.

This phase benefits from a high degree of Agile flexibility such as the one in the proposed framework, where short hotfix sprints can replace traditional patch cycles.

Transparent Communication

Following any difficult launch where trust between the player and developers is lost, developers must shift from marketing speak to a developer-to-player tone that's honest, specific, and frequent. Effective transparency includes:

- A Launch Retrospective Blog or Video outlining:
    - What went wrong - late-stage scope shifts, performance issues etc..
    - What is being addressed immediately (acknowledge the issues and what you're planning to do)
    - What will take longer and why (re-evaluation, development times)
- Regular, accessible updates (via Steam, Discord, Reddit, Twitter/X) with:
    - Patch notes written in human-first language make it as easy for the players to know what to expect
    - Clear timelines and status indicators communicate what is "In Testing" or "Scheduled"
    - Open acknowledgment of community feedback trends – look at forms and reviews, know what features are working in the community and what aren't.

Strategic Repositioning

After addressing critical issues outlined in the stabilisation phase the next critical phase for post launch recovery is a focus on improving how the game is perceived, marketed and supported over time. This isn't phase is not only for technical fixes but about reframing the games reputation and long-term feasibility for both players and stakeholders.

Reframing Narrative

The goal of repositioning focuses on shifting the public narrative from 'What went wrong' to highlighting 'how far the game has come'. This usually requires a shift in how the game is presented to the public:

- Re-Launch Marketing Campaigns: Using the updated trailers, patch recap videos or platform features to gain attention with key messaging surrounding the feedback from the community.

- Utilise Live Ops: Consider switching to a live service style of updates where you can roll out post-launch events, seasonal updates or community challenges to drive community re-engagement without needing to relaunch the entire product.

- Put developers at the centre of the story: the recovery narrative should be framed as human-first: highlight developers explaining what's changed and why, use community facing language such as "fixed this because the community said..." to build emotional reciprocity.

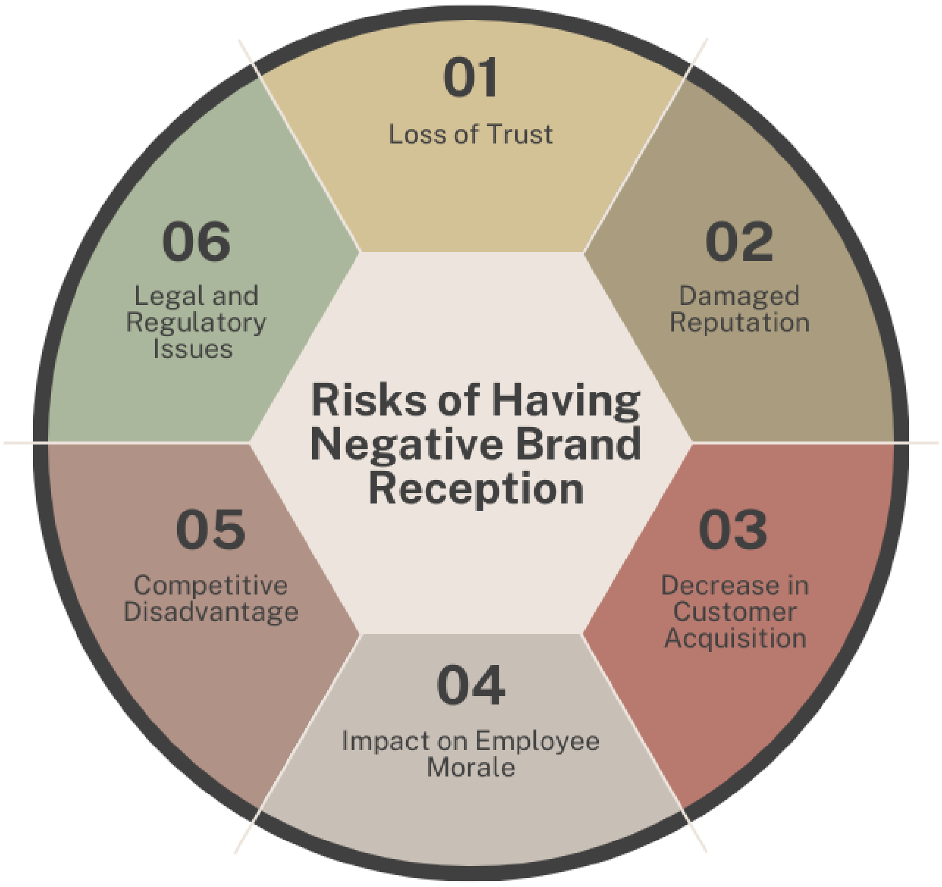
- Each subsequent update should highlight a thematic progress or change such as Improved stability, core feature expansions or new features. Avoid "just fixing bugs" each update should reinforce intentional growth.

It's important to acknowledge that every game might have a different recovery strategy depending on its context, genre and audience, the practices outlined here represent principals that could be implemented in a post launch scenario.

The unique case of No Man's Sky is also something I think is important to cover here, the developer's stance of media silence following its controversial launch was unconventional but did ultimately succeed. By focusing on development and letting the updates speak for themselves, the studio over time rebuilt trust through consistent improvements. However, this approach does carry significant risk – not communicating with media can create a space where external narratives are extremely prevalent, often they are shaped by frustration misinformation, or speculation. Without clear communication, studios lose the control over the public story, which can further damage reputation and delay recovery momentum.

Risks of Unchecked Brand Perception

The risks of abandoning a failed launch cannot be overstated while it might seem easier to shift work to a new project it majorly effects brand perception which plays a crucial role in shaping the future success of any business. When a brand or company is perceived negatively by its target audience, it can face numerous challenges and risks that can impact its reputation, customer loyalty, and overall business performance.



01
Loss of Trust
02
Damaged Reputation
03
Decrease in Customer Acquisition
04
Impact on Employee Morale
05
Competitive Disadvantage
06
Legal and Regulatory Issues
Risks of Having Negative Brand Reception

1 – Loss of Trust: negative brand perception can erode the trust that consumers have in a brand. When customers perceive a product and company by association then question its credibility, reliability and authenticity. This loss of trust can lead to a decline in sales

2 – Damaged reputation: A negative brand perception can ruin a company's reputation, especially in a digital area such as games where negative reviews, social media backlash and viral content can spread quickly creating narratives about a brand which is extremely hard to get on top of and recover from.

3 – Decreased Customer Acquisition: When a company has negative perception, it becomes difficult to attract new customers because of their own pre-conceived conception of a company which results in them being hesitant to engage with a brand that has a poor reputation or negative reviews. This can lead any future games being hindered by the failings of previous titles.

4 – Impact on Employee Morale: Negative brand perception doesn't just effect customers but has significant implications on employees. When a company is perceived negatively, employees may feel demotivated, disengaged and be less likely to want to be associated with the brand. This has a major effect not only in productivity but employee turnover rate which is especially detrimental as the resources needed to onboard people in a creative industry such as games is usually high.

5 – Competitive Disadvantage: In a competitive market such as games, a negative brand perception can give competitors an advantage. Customers may choose to switch to competitors who have a more positive brand image this becomes bigger issue if the game is releasing to an oversaturated market.

6- Legal and Regulatory Issues: Negative brand perception can also lead to legal and regulatory issues. If a customer's perceive a brand's product as misleading it can result in lawsuits, fines and damages as seen with both cases of Cyberpunk 2077 and No Man's Sky.

(FasterCapital, 2025)

Bibliography

Atlassian. (2025). *Sprint Velocity in Scrum: How to Measure and Improve Performance*. Retrieved from Atlassian: https://www.atlassian.com/agile/project-management/velocity-scrum#:~:text=In%20Scrum%20and%20other%20Agile,timelines%2C%20and%20manage%20stakeholder%20expectations. (Accessed: 14 April 2025)

Blow, J. (2004). Game Development: Harder Than You Think. *Ten or twenty years ago it was all fun and games. Now it's blood, sweat, and code.*, 28 - 37.

Bromley, S. (2023, January 19). *Run better internal playtests*. Retrieved from Games User Reseearch: https://gamesuserresearch.com/run-better-internal-playtests/(Accessed: 16 April 2025)

Chandler, H. M. (2020). *The game production toolbox.* Boca Raton, FL: CRC Press.

Cohen, D. (2009). *Producing Games - From Business and Budgets to Creativity and Design*. Burlington: Focal Press. .

Dunlop, R. (2014). *Production Pipeline Fundamentals for Film and Games.* New York: CRC Press LLC.

Dyer-Witheford, N., & de Peuter, G. (2006). "EA Spouse" and the Crisis of Video Game Labour: Enjoyment, Exclusion, Exploitation, Exodus. *Canadian Journal of Communication*, 599-617.

FasterCapital. (2025). *The Risks Of Having A Negative Brand Image And How To Avoid Them*. Retrieved from Fastercapital: https://fastercapital.com/topics/the-risks-of-having-a-negative-brand-image-and-how-to-avoid-them.html(Accessed: 22 April 2025)

Game Developers Conference. (2020, December 18). *Embracing Ambiguity: How to Do Good Work When You Don't Know What to Do*. Retrieved from https://www.youtube.com/watch?v=4DWdnoLosZ8&t=381s(Accessed: 03 April 2025)

Georgiou, M. (2024, July 18). *Step-by-Step Guide to Rescuing a Failing Software Development Project*. Retrieved from imaginovation: https://imaginovation.net/blog/rescue-failing-software-project/#:~:text=1.%20Prioritize%20Critical%20Issues%20Identifying%20and%20prioritizing,to%20get%20the%20project%20back%20on%20track. (Accessed: 28 May 2025)

Gershenfeld, A., Barajas, C., & Loparco, M. (2003). *Game Plan: The Insider's Guide to Breaking In and Succeeding in the Computer and Video Game Business.* New York: St. Martin's Press.

Grewal, B. L. (2020). *An Empirical Study of Delayed Games on Steam.* Alberta: Cornell University.

IceScrum. (2025). *Documentation*. Retrieved from Icescrum: https://www.icescrum.com/documentation/roadmap/

Ittycheria, P. (2019, October 10). *Is Agile Failing Long-Term Planning?* Retrieved from Forbes: https://www.forbes.com/councils/forbestechcouncil/2019/10/10/is-agile-failing-long-term-planning/(Accessed: 09 April 2025)

Jozin, R. (2019). *Crunch time in software development: a theory.* Stuttgart: Institute of Software Technology - University of Stuttgart.

Keith, C. (2020). *Agile Game Development: Build, Play, Repeat, 2nd Edition.* Addison-Wesley Professional.

LeanWisdom. (2023, December 25). *Managing Stakeholders in Agile Projects: Mastering Effective Engagement*. Retrieved from Lean Wisdom: https://www.leanwisdom.com/blog/managing-stakeholders-in-agile-projects-mastering-effective-engagement/(Accessed: 12 March 2025)

Markiewicz, A., & Patrick, I. (2015). *Developing Monitoring and Evaluation Frameworks.* SAGE Publications.

Marklund, B., Engström, H., Hellkvist, M., & Backlund , P. (2019). What Empirically Based Research Tells Us About Game Development. *The Computer Games Journal*, 79–198.

Newzoo. (2024, August). *Global Games Market Report 2024.* Newzoo.

Olsson, A., & Joelsson, G. (2019). *REQUIREMENTS ANALYSIS FOR AI SOLUTIONS - A STUDY ON HOW REQUIREMENTS ANALYSIS IS EXECUTED WHEN DEVELOPING AI SOLUTIONS.* Borås: University of Borås.

Pears, M. (2017, October 12). *Design Pillars – The Core of Your Game*. Retrieved from Game Developer: https://www.gamedeveloper.com/design/design-pillars-the-core-of-your-game(Accessed: 14 April 2025)

Petrillo, F., Pimenta, M., Dietrich, C., & Trindade, F. (2009). What went wrong? A survey of problems in game development. *Computers in Entertainment*, 36.

Robichaud, F. (2023, November 4). *Stakeholder Risk Management and Social Acceptance*. Retrieved from Boreal-is: https://www.boreal-is.com/blog/stakeholder-risk-management/ (Accessed: 20 March 2025)

Rubin, K. S. (2012). *Essential Scrum - A Practical Guide to the Most Popular Agile Process.* Addison-Wesley Professional.

Sotamma, O., & Svelch, J. (2021). *Game production studies.* Amsterdam: Amsterdam University Press.

Tromp, C. (2022). Creativity From Constraint Exploration and Exploitation. *Psychological Reports*, 1818-1843.

Wesley, D., & Barczak, G. (2010). *Innovation and Marketing in the Video Game Industry.* London: Routledge.

Yassine, A., Joglekar, N., Braha, D., Eppinger, S., & Whitney , D. (2003). Information hiding in product development: the design churn effect. *Research in Engineering Design*, 145–161.