



How to Prioritize When Everything is Pri 1

Ruth Tomandl

Sr. Producer, PlayFab

GAME DEVELOPERS CONFERENCE®

MOSCONE CENTER · SAN FRANCISCO, CA

MARCH 2-6, 2015 · EXPO: MARCH 4-6, 2015

Hi, and welcome to the Producer Bootcamp! You'll hear a lot of great talks today; this one is going to focus on how to solve the problem of having way too much important stuff to get done.

Which happens on every project.



My name is Ruth Tomandl, and I've been a designer and producer for about 14 years. I worked on the Dungeon Siege and Supreme Commander games at Gas Powered Games for 7 years, then on Lord of the Rings games at Monolith Productions for 5 years, and for the last two years I've been working in startups. Currently I'm at PlayFab, where we're building a live game operations platform.

What I love about working in games is that I get to work with a lot of extremely smart, ambitious, and creative people, but an unfortunate side effect is that there are always way too many good ideas to actually build. That's why I love being a producer, because that's a huge problem that we can help solve.



What Our Job Is

Producers are in charge of making sure the game gets **done**.

Every feature I have ever talked about WAS in development, but not all made it.
– Peter Molyneux

So first, what is a producer?

Producers get things done. Peter Molyneux gets criticized a lot, and a big part of that is that he gets his team to start tons of things that they can't finish. There are no rewards for starting games, only for finishing them. If you can't finish your game, nobody gets paid and your studio gets shut down.

More than anyone else on a game development team, this is the producer's job: Getting the game done.



What Our Job Is

Producers are in charge of making sure ~~all~~ ^{the most important}
the game's features and components
get completely **done** (that is, shipped)
within constraints
to the required quality level.

Every feature I have ever talked about WAS in development, but not all made it.
– Peter Molyneux

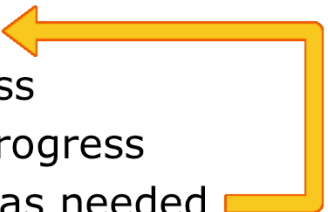
Getting the game done means making sure that everything — art, features, design decisions, UI elements, platform requirements, integrations, updates — gets decided on, planned, built, tested, bug-fixed, past cert, signed off on, translated, shipped, maintained, and updated. And that all of the preceding happens on schedule, under budget, and without hiring more people or lowering quality control.

This, of course, is impossible.

This is game development. Your team is full of extremely smart, ambitious, and creative people, who want to build something amazing. They already have way more features, characters, levels, etc., in mind than your team has time to build — and that's before development even starts. Since you can't do everything, your job is really to make sure the most important stuff gets done, and that time isn't spent on anything that's not needed. That's prioritization.



How We Do Our Job

- Make a plan
 - Follow the plan
 - Measure progress
 - Communicate progress
 - Adjust the plan as needed
 - Make sure the product owners make decisions
- 

The basics of prioritization are simple:

- Make a plan
- Follow the plan
- Measure progress
- Communicate progress
- Adjust the plan as needed
- Return to Step 2 and repeat until done

And, because I'll be using the term a lot, a "Product Owner" is someone who's responsible for giving direction on what stuff we need to build: i.e. anyone who adds items to the list of work. The lead designer, art director, publisher, could all be product owners. You usually have more than one (unless you're on a very small project with a very strong lead), but each one is usually responsible for their own area of the project. Some teams have this really formalized, and on some teams everyone is the product owner for their own work. You need to know who's responsible for making the final decision on each area of the project, so you can make sure important decisions get made correctly, by the right people, and on time.

In order for product owners to effectively make these decisions, you need to make sure they have all the information they need to make them.



Why That's Hard

- The project is always changing
- Coming up with cool ideas is fun
- Different people have different priorities
- You're working within constraints
- Scheduling polish time feels bad

No battle plan survives contact with the enemy.
- Helmuth von Moltke



The project is always changing.

Even if you had a perfect plan, it wouldn't last very long: Product owners are always coming up with cool new ideas or improvements, and the game's design is always changing based on internal reviews, playtests, other games coming out, and customer feedback. And even if the plan didn't change, priorities change based on when in the project you are. Right before an important trade show, having a great demo will be more important than whether matchmaking is fully functional.

Lack of scope discipline.

Dreaming up neat stuff is almost always more fun than actually building it. Your project scope has to match your team's capabilities or you won't be able to finish, and it almost never does. In my entire career, I've been on one project that was appropriately scoped from the beginning, and it was an expansion pack to a sequel the team had just finished, so we were very familiar with what our capabilities were.



Why That's Hard

- The project is always changing
- Coming up with cool ideas is fun
- Different people have different priorities
- You're working within constraints
- Scheduling polish time feels bad

No battle plan survives contact with the enemy.
- Helmuth von Moltke



Priorities are different for each product owner.

On the Lord of the Rings games I worked on, the lead designer's top priority was to make the highest quality game possible. The studio head's top priority was to make a financially successful, well-reviewed game. The VP's top priority was to get a game that fit well into the yearly product slate and that supported film ticket sales, while the license holders' top priority was to make sure our game matched their branding requirements.

You need to make sure that all of these priorities align with the priority list that your team is working off of (and that your whole team is working off of the same list). Strong disagreements or even fights aren't uncommon between discipline leads. You have to make sure that those disagreements are resolved and that everyone agrees on who's responsible for making the final decision on each area of the project. Only then can you make sure important decisions get made correctly, by the right people, and on time.



Why That's Hard

- The project is always changing
- Coming up with cool ideas is fun
- Different people have different priorities
- You're working within constraints
- Scheduling polish time feels bad

No battle plan survives contact with the enemy.
- Helmuth von Moltke



You're working within a lot of constraints.

Some of them are not negotiable, e.g., the game has to ship before the movie opens or you only have 3 gameplay engineers and can't hire more. It doesn't matter what your prioritization list is, if you hit a non-negotiable constraints, it will override everything else. If you have to ship on PS4, a PS4 cert failures will automatically go to the top of the priority list.

Scheduling polish time feels bad

Nobody wants to plan ahead for polish and bug-fixing time, but just finishing all of the Pri1 items will not make a great game. Polish and iteration are necessary for quality. It's extremely hard for product owners to accept a schedule that has any significant polish time built in if they don't have scope discipline. Think of how many features we could fit into that useless polish time padding!

Product owners have to be ambitious and optimistic to be effective, but as a producer, you need to make sure your project plan is realistic. Don't let product owners force you to schedule as though nothing will go wrong and everything will be perfect on the first try. That never happens, and we all know it.



Easy Solutions

That don't work



Easy solutions — that don't work

Everyone would like there to be a simple solution to this problem. There isn't one, but that doesn't stop teams from looking for one. (For a great book about this problem, I highly recommend "The Mythical Man-Month" by Fred Brooks.)

Here are a few silver bullets I'm sure you've heard people propose (or proposed yourself!) and why they don't actually work:



Let's Do Scrum!

- Scrum is a solution for a specific problem
- Scrum can mask problems it's not good at solving
- Even good changes have a price



Not only are there no silver bullets now in view, the very nature of software makes it unlikely that there will be any.

– Frederick P. Brooks

The first time I took a Scrum class, I was super excited. I had found the secret to fix all of my team's problems! I was wrong, of course, but I definitely understand why so many people cling to Scrum as a savior. The problem is that Scrum is a rigid solution to a very specific problem: Teams that are so bogged down or gridlocked that they literally can't produce anything. In that sense, Scrum is the Heimlich maneuver of project management: If you can breathe (i.e. if you are actually getting work done), it's just as likely to make things worse as better.

Scrum is still worth learning about. There are good, useful tools in Scrum that can help you create a good project plan and execute it effectively. But be aware that Scrum can also mask and encourage bad behaviors, including:

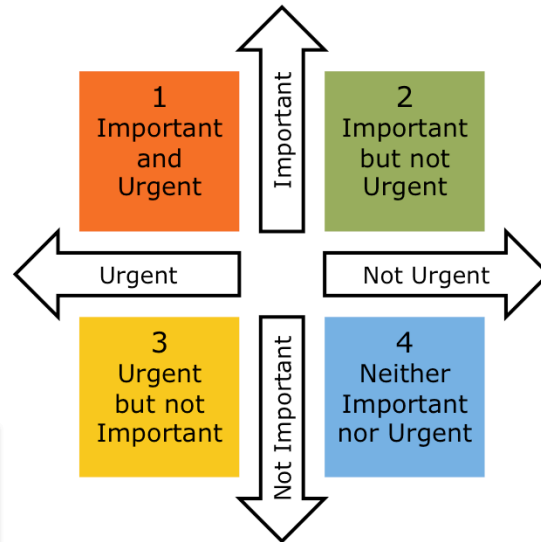
- Product owners who refuse to exercise scope discipline
- Team members who aren't honest (or realistic) about how much they can accomplish
- Spending too much time on minor features that are cool but that don't really help you succeed.



Eisenhower Matrix

- **Important** items help make the game successful.
- **Urgent** items are time-sensitive.

What is important is seldom urgent and what is urgent is seldom important.
– Dwight D. Eisenhower



This project management tool does help you understand your problem space better:

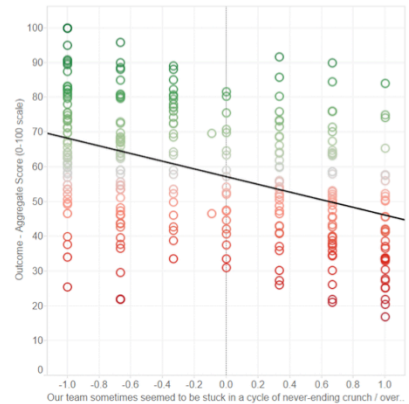
Importance and urgency are different, and they combine to determine a specific item's overall priority. The Eisenhower matrix is most useful for deciding which items are important enough to spend a lot of time deliberating (the top half).

Unfortunately, I've never worked on a project where this matrix would make a dent in the overall problem of "too much to do": Even if you separate all your work into quadrants, you'll still have way too much to do in quadrants 1 and 2, and you weren't ever going to get to the stuff in quadrant 4 anyway. Also, even if an E3 demo is less important to the overall project than getting your matchmaking to work, you do still have to make an E3 demo.



Just Do Everything

- “Find a way”, i.e. Crunch
- Crunch can’t make up for bad scope discipline
- Crunching means your plan has failed (and your team knows it)



Schedule 40 hours a week and you get 38. Schedule 50 and you get 39 and everyone hates work, life, and you. Schedule 60 and you get 32 and wives start demanding you send out resumes. – Game Outcomes Project Survey Response

At some point, every producer will be expected to find a way to get everything done. “Find a way,” “think outside the box,” and “don’t let constraints rule your schedule” often really just mean: “Get your team to work longer hours because we refuse to cut anything.” This is a recipe for disaster.

Crunching can let you get more work done, but not much, and not for long. The recent Game Outcomes Project is full of incredibly useful data about high-performing teams, and one of its clear conclusions is that crunch ultimately just makes things worse. Most teams might get 10% or 20% more done; really focused teams working in short bursts (2 weeks) might get 30 or 40% done during those periods. But burnout is high, and honestly if you have a scope discipline problem, the overrun isn’t going to be in the 10-30% range. It’ll be somewhere between 200% and 1000%.

Also, crunching means means you and your product owners have failed in your planning, and your team knows that. Like the other tools discussed here, crunch may have a role in very specific situations. But it is not a solution to the problem of too much to do.



Hard Solutions

That actually work

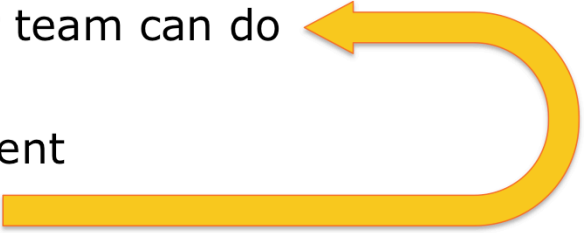


Here are some solutions that are a lot harder, but will actually give you results.

None of them will solve all of your problems, but they'll help your team be better at making games, and will help you be a good producer who is good at getting games done.



Cultivate Scope Discipline

- Product owners are rewarded for ambition
 - ... so you need to bring the realism
 - Understand what your team can do
 - Build credibility
 - Keep priorities consistent
 - Track your progress
- 

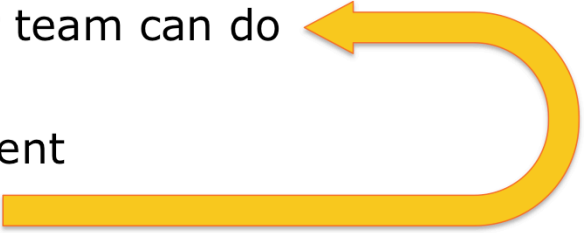
I think [Peter Molyneux] intentionally tries to say things to make them happen. ...
I think sometimes he does it to help push the team to shoot for the moon. – Gary Carr

Most of the actual solutions to the “Everything is Pri1” problems are really about scope discipline. Product owners are rewarded for their ambition, and there’s a strong culture in games (and software in general) of refusing to be disciplined in scope. Game studios could design a game that they were certain they could build, and assume that they’ll find cool things to expand on and add later. But I’ve literally never seen this happen.

The headline of this article is a bit tongue-in-cheek: If your product owners really believe that everything is Pri1, then you have a scope discipline problem. However, if they initially believe everything is Pri1, but you’re able to work with them to focus the scope to realistic levels and create a prioritized project plan, that’s a healthy tension that will benefit your team. They bring the ambition, you bring the realism, you get a good game done.



Cultivate Scope Discipline

- Product owners are rewarded for ambition
 - ... so you need to bring the realism
 - Understand what your team can do
 - Build credibility
 - Keep priorities consistent
 - Track your progress
- 

I think [Peter Molyneux] intentionally tries to say things to make them happen. ...
I think sometimes he does it to help push the team to shoot for the moon. – Gary Carr

Here are some ways of doing this:

1. Practice what you preach

Making a realistic plan starts with you. Do you have a good understanding of what your team is capable of? And do you have enough credibility that your product owners will believe you?

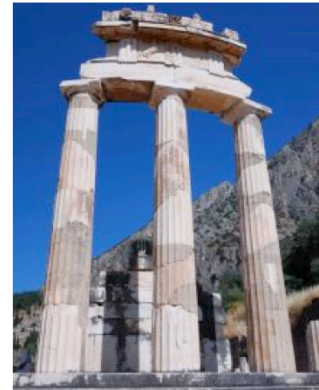
Don't commit to more work than your team can do. This is difficult, since you almost never have a really good idea of what your team is capable of unless they've been together for a while and have worked on similar projects in the past. You'll almost certainly overestimate what you can do, but do your best to make sure that that overestimation isn't too high, and that you have a plan to cut scope if (when) needed.

Be consistent. Work with your product owners to ensure that priorities are consistent across the entire team so that the most important stuff gets done. Then track and report on that progress, which will in turn contribute to both your understanding of the team's capabilities and your own credibility with the product owners.



Stay Focused on Your Pillars

- Will this be the difference between success and failure?
- Identify what success means
- Decide as a team on 3 pillars



Focus is a matter of deciding what things you're not going to do.
- John Carmack

2. Define 3 success pillars

It's hard to stay focused on critical work if you can't identify what's critical to begin with. You never really know which feature or polish item will be one that your players will love, or which bug will be the one that really frustrates them. But thinking in terms of "will this make us succeed or fail" helps keep your team in the right frame of mind. Asking this question may lead to features that seem critical being cut, and work that wasn't getting enough attention prioritized higher.

Start by defining three pillars that you plan to judge your game around. Pillars are major features, selling points, or unique aspects of the game that will lead to success. (Three is a good number because it will require some discussion to get your team to agree on three, and any more than that will be hard to remember.)



Pick Good Pillars – actionable, specific, positive

Good Pillars	Bad Pillars
Monster to the monsters	90% Metacritic
Sumi-e art style	Unique art style
Accessibility for all skill levels	John Romero's about to make you his bitch

Good pillars are actionable, specific, and positive. They should cover large areas of the project and make it easy for developers to know whether their work is aligning to the goals of the project. Bad pillars are those that won't help team members prioritize their own work or understand its context in the overall project.

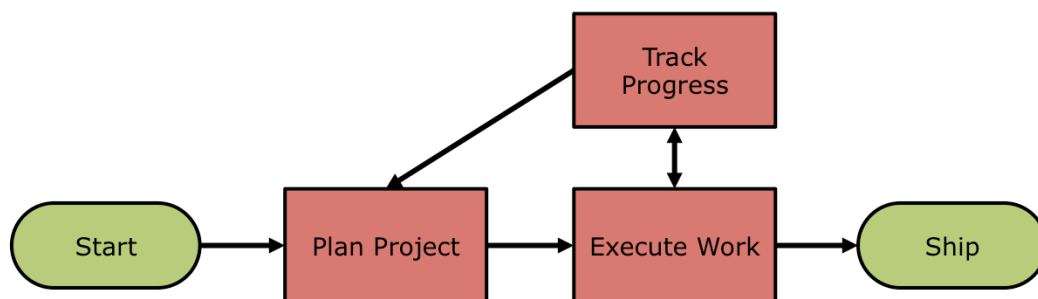
Pillars are often very similar to a game's unique selling points, or marketing points. Which makes sense, because in a way pillars are how your team markets your game to yourselves. It's a way for your team to have context for what you're working on and to understand what the final, finished product will look like and how each person's work fits into that finished project.

Finally, if a feature or work item doesn't align with your pillars, ditch it or change the pillars.

(Good pillars are from: Shadow of Mordor, Okami, Unreal Tournament 2)



Use the Right Process



4. Process is how you plan your project, execute on that plan, control the work being done, and use what you've learned to update your plan. Selecting the right process for your team is part of getting familiar with what your team is capable of, and how you can most effectively help them do their best work.



Use the Right Process

Agile (reactive)	Waterfall (proactive)
Iterative design	Strong design plan
Experienced team	Less experienced team
Small team size	Large or distributed team
Nebulous business goals	Clear business goals
Self-managed team structure	Top-down team management

I could spend the next year talking about nothing but agile vs. waterfall, but the key thing to remember is that whichever you pick, it needs to be the right one for your team. More experienced, smaller teams or projects with more uncertainty benefit more from agile processes because they let you continually change your plan in response to changing circumstances. Agile also works well with live games, such as Eve Online, because it lets teams iterate in response to player behavior.

Waterfall gives more predictability but makes it harder to respond to unforeseen problems. Larger, more distributed teams or projects with clearer paths to success (e.g. sequels, DLC updates) work well with waterfall schedules. A lot of people hate waterfall because it's less flexible, but for the right team it can work well: Rocksteady uses a very strict waterfall scheduling process for their Batman games, for example.

Most teams work with a mix of the two; most teams I've been on have used waterfall for art and level design, and agile for engineering and design.



Use the Right Process

- Familiarize your team with the process
 - And make sure they have ownership

The way a team plays as a whole determines its success. You may have the greatest bunch of individual stars in the world, but if they don't play together, the club won't be worth a dime.

– Babe Ruth

5. Familiarize your team with the process

The best process in the world won't work if your team doesn't use it. Another of the conclusions of the Game Outcomes Project was a positive correlation between team buy-in to the process and a game's success. Does everyone on your team know how to tell the team when something is done, or what to do if they get blocked? How do they track their task list and know what to work on next? Do they know how to find out what someone else is working on?

Give your team ownership over the process. They know what they can do, so involve them in milestone planning and decision making. At PlayFab we have quarterly and yearly planning meetings where the whole team works together to plan the main features to work on next. We have a lot of Pri 1s, but we're able to make a plan that everyone understands, everyone owns, where nothing important is overlooked, and that was decided on with the maximum amount of available information, and thus is likely to succeed.

The collage displays several project management and data visualization tools:

- Gantt Charts:** Multiple Gantt charts showing task durations and schedules. One chart includes a legend for 'Duration of a Normal Job', 'Spec. Time for a Normal Job', 'Duration of a Critical Job', 'Spec. Time to Holiday', and 'Actual Schedule'.
- Kanban Boards:** A Kanban board with columns for 'To Do', 'In Progress', and 'Done', showing tasks like 'Approve Plan', 'Drawings', 'Supply Market', etc.
- Project Timelines:** A project timeline for 'Platform 360' showing milestones from 2014 to 2015, including 'Project Greenlight', 'Initial GQ/TO delivery', 'Feature Planning/Prototype', 'First Production', 'Alpha', 'Beta', and 'Release'.
- Milestones Table:** A table listing milestones with columns for DATE, Milestone, and NAME.

DATE	Milestone	NAME
2/21/14	Greenlight	Project Greenlight
2/14/14	MD	Initial GQ/TO delivery
2/28/14	M1	Feature Planning/Prototype
4/25/14	M2	First Production
7/2/14	M3	Alpha
8/2/14	M4	Beta
10/7/14	WW	Beta Launch Submission
11/2/14	RL1	Post Launch Content 1
12/7/14	RL2	Post Launch Content 2
- Platform 360 Project Overview Table:** A table showing project details for Platform 360, including Project Overview, Milestones, and Release.

Project Overview	Milestones	Release
Project Greenlight	Initial GQ/TO delivery	2014
Feature Planning/Prototype	First Production	2014
Alpha	Beta	2014
Beta	Release	2014
Release	Release	2014

All models are wrong, but some are useful. Visualizing your schedule in different ways helps you make sure you're taking all costs into account.

These schedule visualizations are all tools that you can keep in your producer toolbox and use as appropriate. You might find that different ones are more informative to different teams, or that some of them help you plan your project and others help you track work or communicate the plan to your team.

We've already discussed a ranked backlog, so here are a few other planning and visualization tools you might find useful.



Swimlanes

Monday	1/13/14	1/20/14	1/27/14	2/3/14	2/10/14	2/17/14
Friday	1/17/14	1/24/14	1/31/14	2/7/14	2/14/14	2/21/14
Week #	1	2	3	4	5	6
Eng 1	Blackjack gameplay			Stats		Report Player
Eng 2				Casino Goals	Blackjack Goals and Jackpot	
Eng 3	Slots gameplay and animation pipeline			Leaderboards		Slot Jackpots and
Eng 4						
Eng 5	Poker Ring	Lobby	Overall	Chat	Economy	
Eng 6	Basic Goals		Goals General		Poker Goals	Account Authent
Eng 7	Slots Math			Integrations		
Artist 1	BJ art: 1st pass & Visual targets		Overall art	BJ polish		Casino goals
Artist 2	Slot 1 art: 1st pass & Visual targets			Slot 2 art: 1st pass & Visual targets		Slot 3 art: 1st pass & Visual targets
Artist 3		Slot 1 animations: 1st pass		Slot 2 animations: 1st pass		Slot 3 animations
Artist 4		Lobby art			Poker Content	
Designer 1	Initial economy design	Goals design	Detailed economy design	Minigame Design	Gameplay design refinement	
Designer 2	Detailed wireframes	Lobby Design	Gifting design	Telemetry Design	Goals design refinement	

Pros:

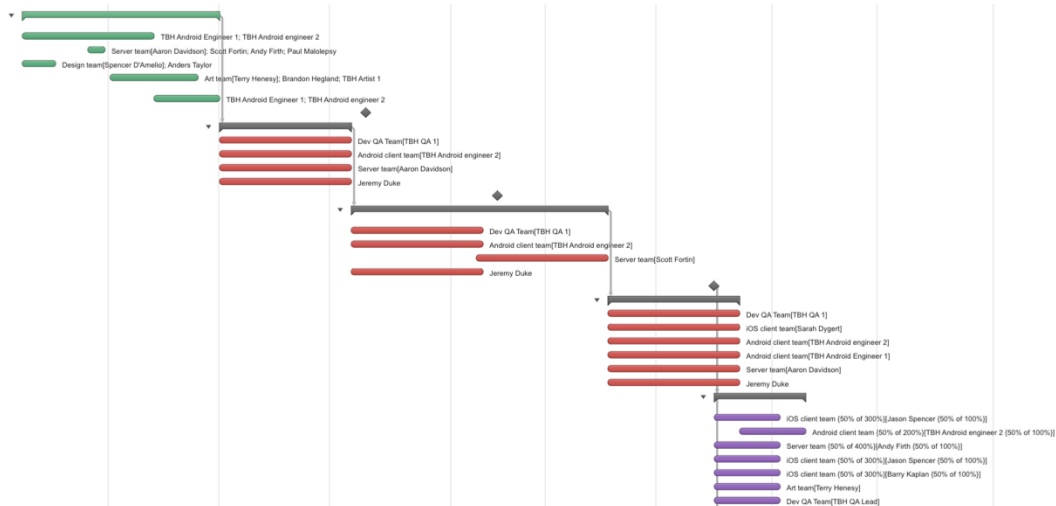
- Easy for team members to understand what they're responsible for
- Forces product owners to realize they've asked for too much at once
- Easy to update, and to customize (I use Excel)
- Lets you identify tasks with a deadline but no assigned team member

Cons:

- Almost impossible to schedule polish and bug-fixing time, because product owners will instantly see it on the chart and wonder why their favorite feature isn't there instead.



Gantt Chart



Gantt charts are a great visual illustration of why 'Waterfall' is called 'Waterfall'. They get a lot of hate from producers (because publishers often require them and they're a pain to make) but they are actually a very useful planning tool.

Pros:

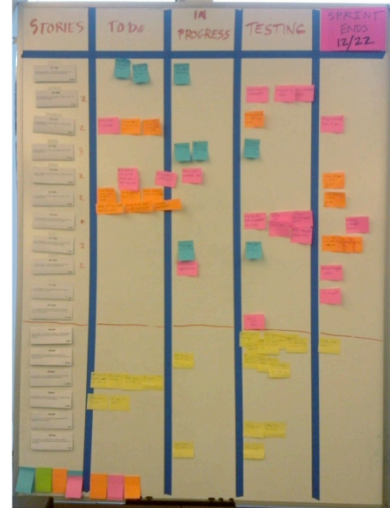
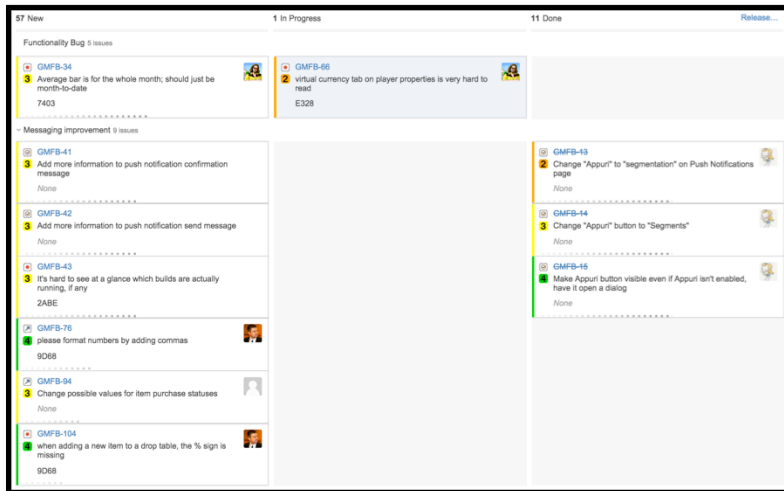
- No task is allowed to start before all its dependencies are finished, and no person can have two tasks assigned to them at the same time.
- Entering everything into a Gantt chart forces everyone to realize just how long it would take to get everything done. It's a good "reality shock" too.

Cons:

- They're very difficult to read and nearly impossible to keep updated
- There are tons of programs out there for making Gantt charts, but many people get stuck using Microsoft Project, which is expensive, hard to use, and doesn't run on a Mac.



Kanban (or Scrum) Board



A Kanban board is a tool for managing and communicating the state of individual tasks. It has 3 columns: To Do, In Progress, and Done, with a card representing each task. It's common to enforce restrictions on how many cards can be in the 'In Progress' column, or how long they're allowed to be there.

Pros:

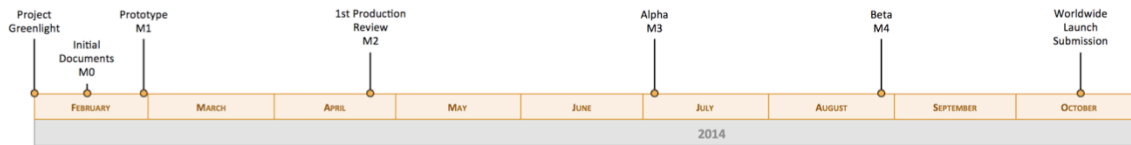
- Can be used either offline or online, and many good tools exist for creating them
- Useful for smaller teams or parts of a project, such as an art team

Cons

- Once your team is bigger than 10 people, it starts to break down quickly, and gets very hard to keep updated
- Less useful if your work has many external dependencies or milestones that can't be tracked on the board

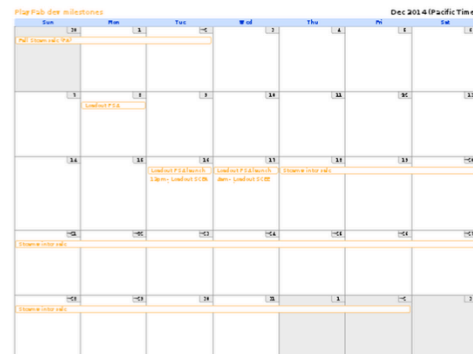


Milestone Calendar



Milestones

DATE	Milestone	NAME
2/1/14	Greenlight	Project Greenlight
2/14/14	M0	Initial GDD/TDD delivery
2/28/14	M1	Feature Planning/Prototype
4/25/14	M2	1st Production Review
7/4/14	M3	Alpha
8/29/14	M4	Beta
10/17/14	WW	WW Launch Submission
11/21/14	PL1	Post-Launch Content 1
12/19/14	PL2	Post-Launch Content 2
1/16/15	PL3	Post-Launch Content 3
2/13/15	PL4	Post-Launch Content 4
3/13/15	PL5	Post-Launch Content 5
4/3/15	PL6	Post-Launch Content 6



One of the big downsides of Scrum (and agile processes in general) is that it's tough to communicate and enforce deadlines because you're focusing hard on the next two weeks. If you have to ship by a certain date, and especially if you have non-negotiable deadlines (E3 demo), you can't just rely on your backlog or scrum board to visualize your plan. Also have a calendar with upcoming dates, and communicate regularly to your team what those dates are and what exactly needs to be done by each date.

Pro:

Forces everyone to focus on non-negotiable deadlines (that E3 demo again) that are more than 2 weeks ahead.

Con:

Not intended to be used on its own. Pair it with other tools to let the team see how their work fits into this bigger picture.



Be Prepared for Problems

Plans that only work if nothing goes wrong are bad plans.

- Your art team will get the flu
- Your only network engineer will quit
- Core requirements will change
- A critical feature of your game won't be fun

Hofstadter's Law: It always takes longer than you expect, even when you take into account Hofstadter's Law.

– Douglas Hofstadter

7. Be prepared for problems.

Plans that only work if nothing goes wrong are bad plans. Your art team will get the flu; your only network engineer will quit; core requirements will change; and a critical feature of your game won't be fun. This is where having 3 pillars is a good idea — if one of them collapses, you still have two others supporting the game.

Or, another way to say this: Schedules that only work if nothing goes wrong are fantasies. And:



Be Prepared for Problems

**Fantasy schedules
don't work.**

Hofstadter's Law: It always takes longer than you expect, even when you take into account Hofstadter's Law.

– Douglas Hofstadter

Fantasy scheduling is particularly a risk if you're surrounded by happy, creative, optimists (so, basically the rest of your team.) The producer's role is to gently (at least at first) remind them of reality. Get used to communicating bad news and pointing out risks in advance. This will not make people hate you. Good product owners appreciate that someone else is keeping track of the bad stuff so they can keep main focus on all the cool features they want to build. Point out fantasy scheduling when it happens, and make sure your plan has some slack to deal with potential problems.



Risk Analysis

List potential risks, with:

- Likelihood it will happen
- Size of impact to the project
- Plan to minimize likelihood
- Plan to mitigate impact

If you don't have a plan B, you don't have a plan.
– Harvey Mackay



8. Conduct a risk analysis

Get your team together and list potential risks to your project; for each one, give it a score for likelihood and impact. For example, your publisher cancelling the project is usually low-likelihood, high impact, but it could be high likelihood if you know the publisher has been cancelling a lot of projects recently or is in financial difficulties. Team turnover is usually medium likelihood, low impact, but if your only network engineer hates living in your city, it might be the highest in both categories.

Rank the risks based on those two scores. The top ones (aka the scariest risks) are the ones you should worry about the most. Make sure you have a plan for minimizing the likelihood and impact for them — a plan B, basically. Going through this process (and updating it at each milestone) will also give everyone a place to voice their fears for the project and to have some control over the things they're afraid of.



Answer the Right Questions

- "Is there any way we can do X?"
- "Can we do X and still do everything else?"
- "If we adjust our plan to include X, how does that affect our other top priorities?"

Effective leadership is putting first things first.
Effective management is discipline, carrying it out.
– Stephen Covey

9. Answer the right questions

Unless you're working on an extremely strictly scheduled waterfall project, new stuff is going to get added to the prioritization list, often at the top. When that happens, the question that's asked is usually, "Is there any way we can do [X new item]?" This is a bad question, because the answer is: We have a super smart, creative team. Of course there's a way we can do X new item.

DO NOT give that answer. It doesn't give your product owners the information they need to make good decisions, and if they have a scope discipline problem, it lets them get away with it. Also be wary of the question, "Can I add this item to the list and still get everything else done that I want?" The answer is no, because of the laws of physics. But that's not a useful answer and it won't help your project get done. These are both questions with no good answers. Don't answer them.



Answer the Right Questions

- "Is there any way we can do X?"
- "Can we do X and still do everything else?"
- "If we adjust our plan to include X, how does that affect our other top priorities?"

Effective leadership is putting first things first.
Effective management is discipline, carrying it out.
– Stephen Covey

Instead, answer this question: "We have a new priority: X. Can you adjust the plan to include X and tell us how that's likely to affect the other high-priority items on the list?" That's the question product owners should be asking, so that's the one you should answer. Verify with them where on the list X should go. Is it more important than our previous top priority items? Is it more urgent than other items, and therefore needs to get done earlier even though it's less important? If so, definitely let them know how that's likely to affect or delay the higher-priority items. Does it fit your established pillars, or conflict with them? If it conflicts, do we need to change the pillars?

Adding new top priorities needs to be done carefully, and it's never free. Don't let your team fall into the trap of thinking that it is. That's how scope discipline deteriorates, and that's how long, terrible death marches happen. Successful teams learn how to control that risk.



In Summary

- Your job is to get your game done
- Don't expect to do everything
- Lack of scope discipline is your enemy
 - But you have tools to fight it:
 - A realistic plan
 - Ways to communicate that plan
 - Well-defined Pillars
 - Back-up plans



Finally, don't let the coolness of the art or gameplay distract you from your role on the team. As I said at the beginning, the producer's job is to get the game done. Your team can't build every cool thing they think up, so your job is to make sure you get the most important ones done, and that your team agrees which ones are the most important — and what "done" is.

Lack of scope discipline is your enemy, but you have a number of tools to fight it:

- A workable plan. Start with a ranked backlog.
- Various ways to visualize and communicate your schedule, to catch problems with it early and keep people working on the right things.
- Clearly defined pillars to help you know whether you're working toward success and to help your team know how their work contributes to that success.
- Anticipating problems and making sure your team is prepared for them.



Resources

- GDC Talks:
 - Production Support Roundtables
 - Five Things You Can Do Today to Be a Bit More Agile
 - Leading High Performance Teams
 - The Vertical Slice Challenge
 - Using Earned Value to Course Correct and Deliver on Time
- Books and Articles:
 - The Mythical Man-Month (esp. "No Silver Bullet")
 - Game Outcomes Project
 - The Goal (a novel about constraints)

GDC talks (besides the rest of the Producer Bootcamp):

- <http://schedule.gdconf.com/session/production-support-roundtable-toolsprocesses>
- <http://schedule.gdconf.com/session/five-things-you-can-do-today-to-be-a-bit-more-agile>
- <http://schedule.gdconf.com/session/leading-high-performance-teams>
- <http://schedule.gdconf.com/session/the-vertical-slice-challenge>
- <http://schedule.gdconf.com/session/using-earned-value-to-course-correct-and-deliver-on-time>

http://en.wikipedia.org/wiki/The_Mythical_Man-Month

<http://intelligenceengine.blogspot.com/2014/11/game-outcomes-project-methodology-in.html>

[http://en.wikipedia.org/wiki/The_Goal_\(novel\)](http://en.wikipedia.org/wiki/The_Goal_(novel))



Q&A

Thanks! Any questions?