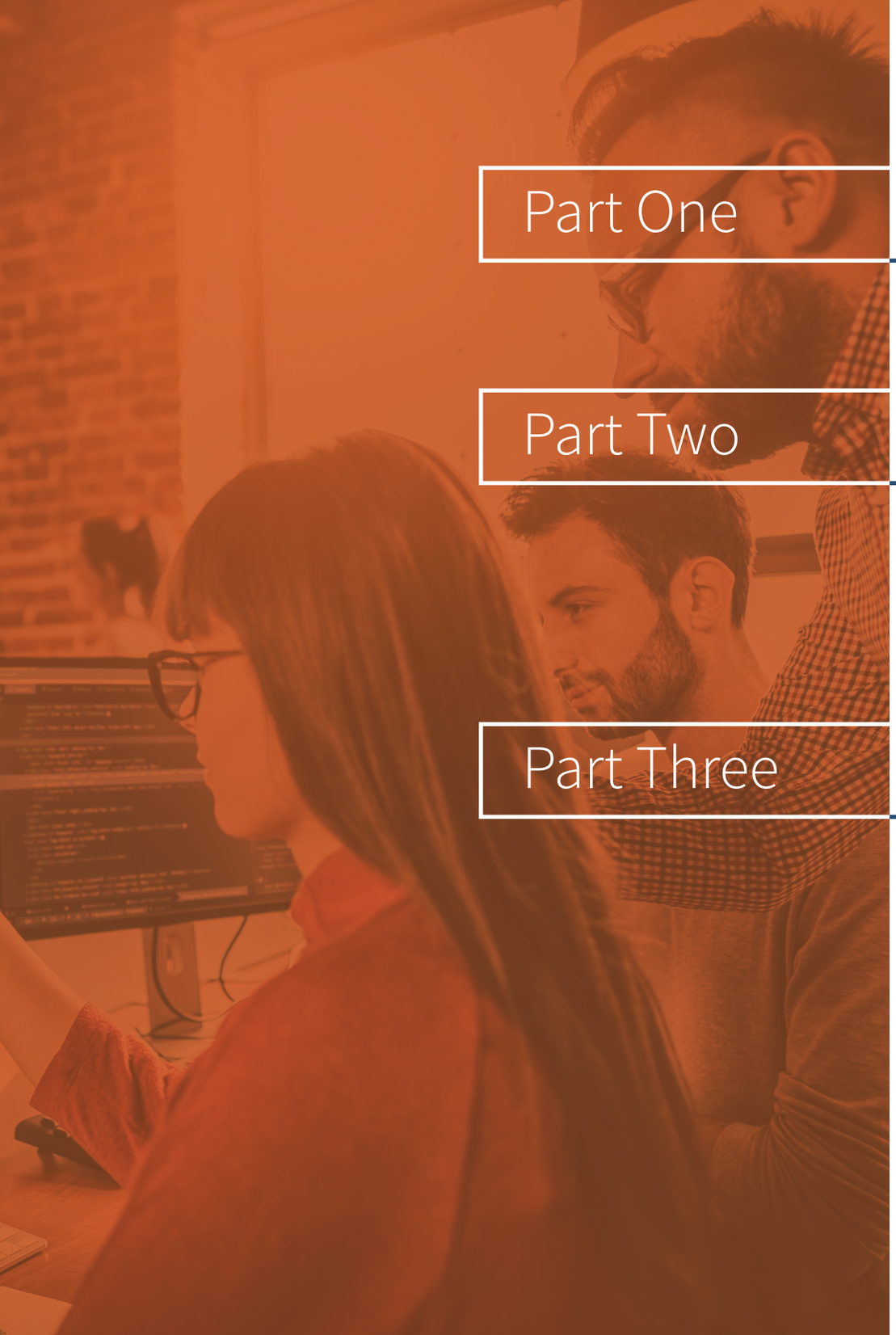
Playbook

Agile [Isn't] for Hardware

by The Playbook Team



Part One

We've Been There

Part Two

The Convergence of Methods and Tools

Part Three

The Solutions

Part One

We've Been There

Our higher-level purpose at Playbook is to help engineers love their jobs again.

1

Product Development is Tough

By the end of the first day of my first job, I was convinced I had made the wrong career choice.

When I was a kid, I used to dream about inventing cool products. So I got a degree in mechanical engineering. Although there was some evidence in college that I had made the wrong career choice, by the end of my first day at work it was clear that I wasn't going to like this job.

Sure, I got to solve challenging problems with creative designs. But what I didn't know was that I would have to collaborate with a bunch of other people who also had important input for the product. And they had a lot of questions: How much would it cost? Was it easy to manufacture? Did it generate too much heat? When would it break? Was it backwards compatible? How many customers had tested it? Could I make more room for the circuit board?

I didn't know. The prototype worked, and that was all I cared about.

I soon realized that product development in the real world was a team sport that was fraught with inefficiencies. Since every challenge seemed like an opportunity to invent a solution, I started fixing the problems that plagued product development and it eventually became my full-time job.

And I was happy again.

In fact, I loved it so much I never stopped.

Part Two

The Convergence of Methods and Tools

2

Years of Learning

After running out of things to fix at my company, I left and started helping other companies improve their engineering tools and processes. Soon after that I partnered with two other engineers who were doing the exact same thing, and over the next ten years we worked at over sixty different companies. They were from every possible industry and varied in size from a few engineers to a few thousand. And this created an ideal environment for us to rapidly learn — not only what worked, but also what didn't.

During this time manufacturing was implementing things like single piece flow and just-in-time inventory, and these became part of the larger body of knowledge called Lean. And it worked really well for manufacturing.

Since manufacturing was right downstream from engineering, Lean moved upstream. And while some of the ideas worked in R&D, most of them didn't.

A while later, software teams were frustrated with the results they were getting. So a core group of developers started a rebellion against the traditional "plan driven" approach and invented Agile. And it worked really well for software development.

Since the software teams were working alongside the hardware teams, Agile moved into hardware. And again, while some of the ideas worked, many of them didn't.

One of the most important books we read during this time was ***The Goal***. It was written in the 1984 — long before Lean and Agile were concepts. It sold over four million copies which was unheard of at the time for a business book — especially one about manufacturing. It was the beginning of what became Theory of Constraints and explained how to find and understand the constraints in complex systems, and then how to alleviate them. And it worked really well.

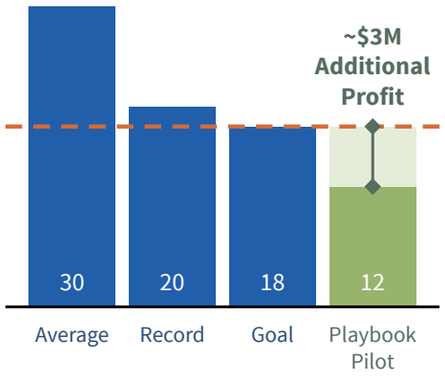
Finally, we read a book called ***Managing the Design Factory***, by Don Reinertsen, and it made ***a lot*** of sense. Instead of trying to copy methods without understanding the science behind them, it explained things like queueing theory and process science — boring stuff unless you're an engineer and need a tool to solve a problem. Then it's fascinating. And it worked really well in the design factory.

3

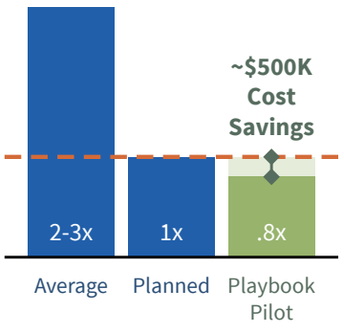
First Big Success with Playbook Methods

By now our heads were full of good ideas and we had implemented many of them as single solutions. So when we got the chance to work with a medical device manufacturer that wanted to make aggressive improvements, we made a list of forty things, prioritized it, and picked the top eight.

And they finished the project six months early and twenty percent under budget.



Product Development Cycle Time (months)



Budget ($)

"Cutting the project time by 50% is not something you can just sort of ignore."

— Ken T.
Engineering VP
Medical Device Manufacturer

The customer couldn't believe it and we were surprised too.

But the very next company we went to had a completely different type of project (a major PLM systems upgrade), and we implemented the ***exact same set of methods***, and they got an even bigger result. So big, in fact, that they won their company's Chairman's Award.

By now we knew we were onto something. The principles were real, and the methods not only worked, they worked really well.

But as we rolled these methods out to more customers it became clear that the current manual approach (Excel, sticky notes, Sharpies and web cams) could be improved, and all of the customers began to ask the same thing:

"Can we have software?"

So we went scoured the market for tools, but none of them would do what we needed. They all had different pieces, but none of them had everything. So we had to build it ourselves.

But before we started, we knew it was critical to understand the customers' highest-level goals. So we did a Voice of Customer process and the top two priorities were very clear.

They wanted their projects to have ***predictable end dates***. And they wanted to be able to ***respond to changes***.

But predictability and flexibility are two very different capabilities.

And their third goal was that ***everyone is well informed***.

Voice of the Customer

Innocuous daily slips accumulate until *they double the length of the project.*

So how do you keep everyone well informed in a system where things are changing rapidly, and still achieve a predictable end date? (Nothing like a challenge!)

Now that we understood the goals, we started studying the problem. And what we found surprised us.

For most projects, the biggest overall cause of delay is daily slips.

That’s right. A single day slip on the schedule. It doesn’t sound like much, but they happen all the time and are hard to detect. So people blame the lateness on big delays from test failures, long lead times, scope creep, and other things that are noticeable and memorable.

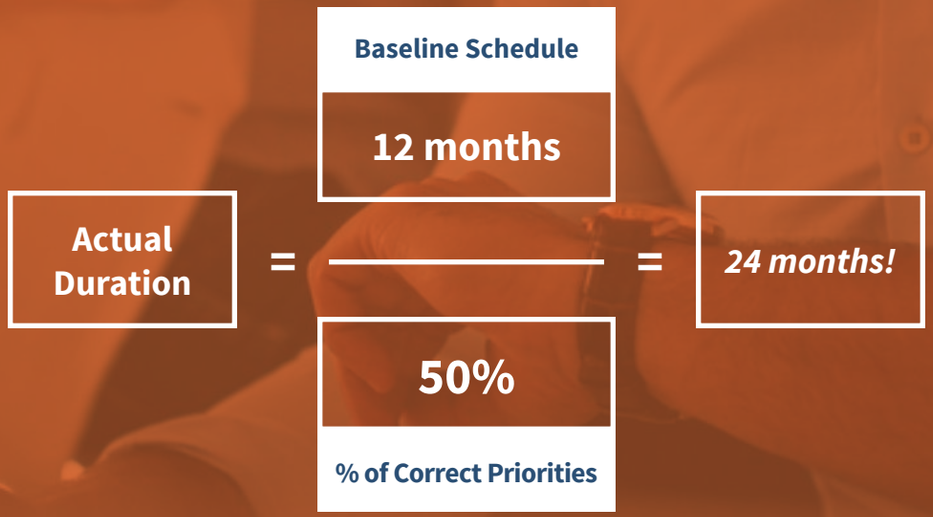
And all the while, the little innocuous daily slips accumulate until ***they double the length of the project without anyone noticing***.

Kind of like dark matter in the universe.

4

The Villain — Daily Slips

To determine the impact of daily slips, divide the baseline schedule by the percent of time people work on the correct priorities.



What is causing daily slips and how bad are they, really?

The simplest answer is that people are working on the wrong things at the wrong time. This is caused by many things: overloaded resources, multitasking, unclear priorities, etc.

And every day that the correct priorities are missed, the project slides one day.

While that might not sound like much, the impact adds up quickly. In fact, the actual project duration can be determined by dividing the baseline schedule by the percentage of time that you have correct priorities. In other words, if you have correct priorities 50% of the time, the project will double just from this reason alone.

So how do we fix it?

While there can be dozens or even hundreds of ongoing activities on a project, the only way to move the end date is to delay the work on the critical path, or delay something else long enough that it becomes the critical path. To prevent this we need to know where the critical path is. But that is hard to do when the projects are large and complex, and things are changing fast due to the uncertainty inherent in new product development.

There are many planning tools that can highlight the critical path, but it's the traditional process of developing the plan that is not effective. Most companies have a project manager create a high-level plan and stop there. But engineers know that the devil is in the details, and the details matter in this case. There is usually a lot of them, and they quickly become out of date.

So, the plan becomes nothing more than an approximation of the work that needs to be done. And the project manager spends time every two weeks asking for updates from the team members so they can tell senior management how the project is going. It's time consuming, frustrating, and inaccurate.

Fortunately, there is a better way, and Don Reinertsen recommended decentralized planning.

Part Three

The Solutions

5

Decentralized & Rolling Wave Planning

Decentralized planning simply means to involve the people who are going to do the work when you develop the plan. The team members are the subject matter experts for the details around exactly what needs to be done and how long it will take. And the benefits are obvious. The plan will be more 'correct', meaning it will have the right steps in the right order and they will be linked to the right things. And the estimates for duration and work (effort) will be more accurate.

What's the right level of detail?

The optimum level of detail is enough that the critical path is correct, and nothing will fall through the cracks that could delay the project. We want predictability, so there's no reason to leave out details now and hope someone will remember them later when they're busy. Capture everything now so it can be tracked and reused in future plans.

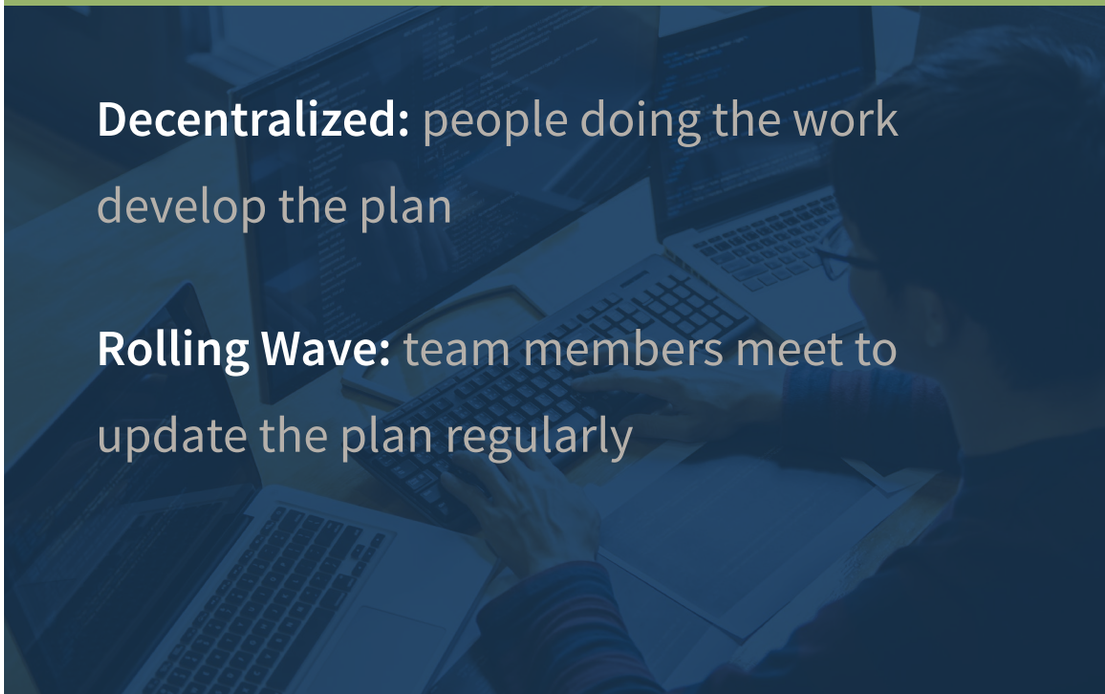
Doing this will create a plan with quite a bit more detail, but we don't need to plan the entire project at the beginning. New product development has so much complexity and uncertainty that it would be a waste of time to try to accurately develop the entire plan at the beginning.

How far ahead should we plan?

You only need to add the detail in the current phase to the next major milestone. This is a point where you know you have clearly achieved something. You should have a high-level outline of the rest of the project, but the future details will be added later during Rolling Wave Planning sessions.

Rolling Wave Planning simply means that you update the plan on a regular basis and have the team members there when you do it. This is necessary as things change rapidly during product development—new information is arriving frequently, and the path forward becomes clearer each day. The frequency of this can vary depending on the stage of the project and how often things are changing, but it's usually once every two weeks for a couple of hours.

The primary goal is to make sure the critical path is correct to the next major milestone. Because if the critical path isn't correct, the team member priorities aren't correct. And the project will slip every day until they are.



Decentralized: people doing the work develop the plan

Rolling Wave: team members meet to update the plan regularly

6

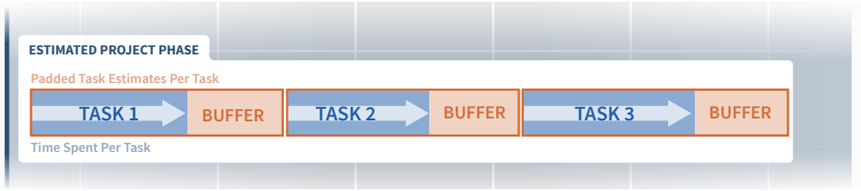
Shared Buffers

Shared buffers is an idea borrowed from Critical Chain Project Management (CCPM), which came from Theory of Constraints. Critical chain sounds very similar to critical path but has two key differences: shared buffers and load leveled resources.

One outcome of having the team members provide their own estimates for duration is they will usually pad their estimates. This is logical. They know things take longer than they expect, they know they will be multitasking, and they don't want to be the one who causes a delay.

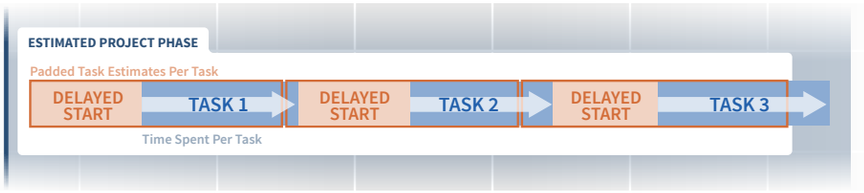
So if they think something could be done in five days, they will probably give ten days as their estimate. And if three people in a row do this, the project plan will be thirty days long, and have fifteen days of actual work in it.

Task Buffers Resulting From Padded Estimates



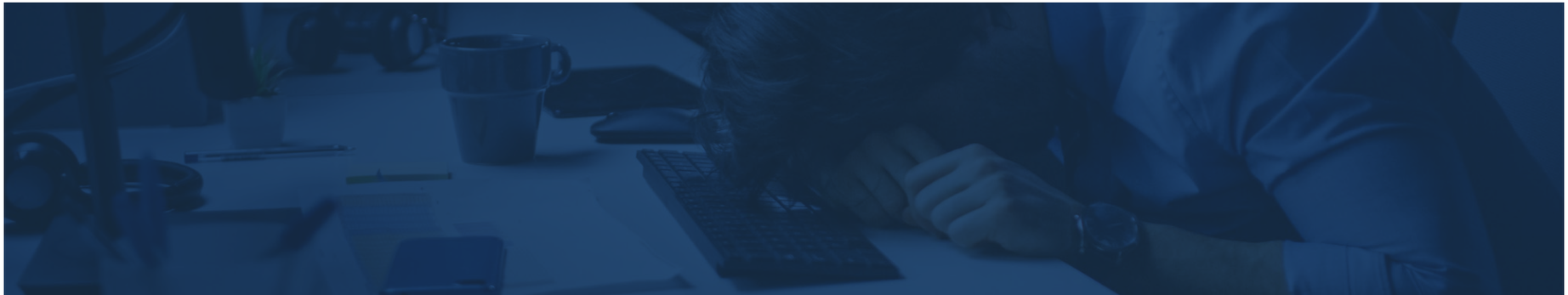
But there's this thing called Student Syndrome that says people don't work on something until right before it's due. This isn't because they're lazy. It's because they have more urgent things to work on. So they focus on those first, or they multitask. Then right before this task is due it becomes urgent, and then they focus on it until they get it done. So, the execution of these tasks looks like this:

Tasks Postponed Until Urgent

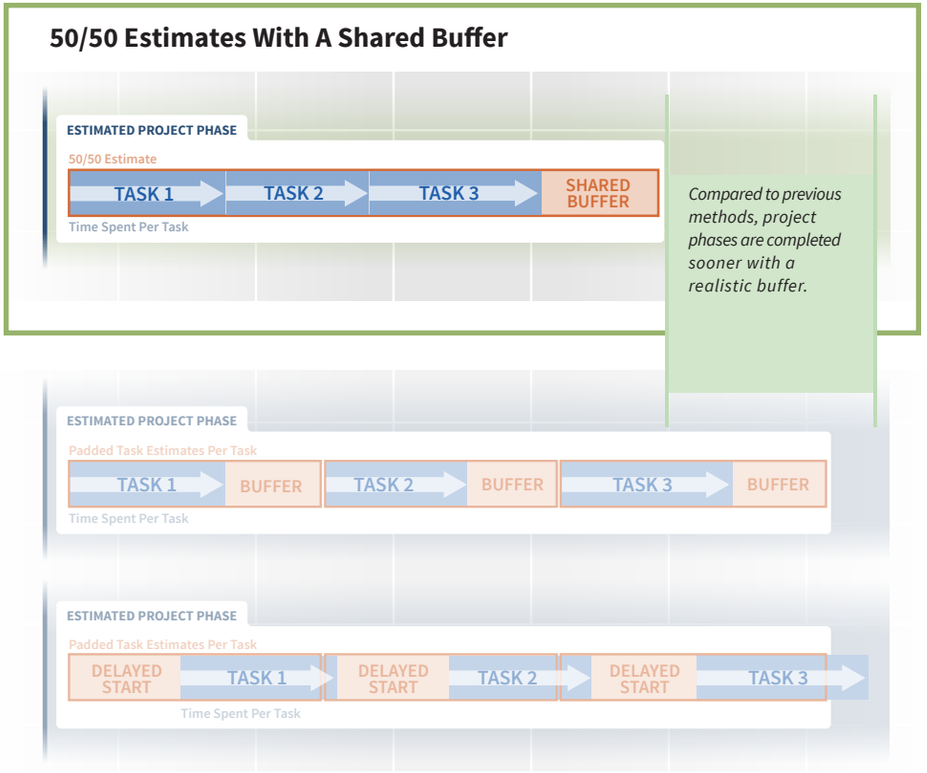


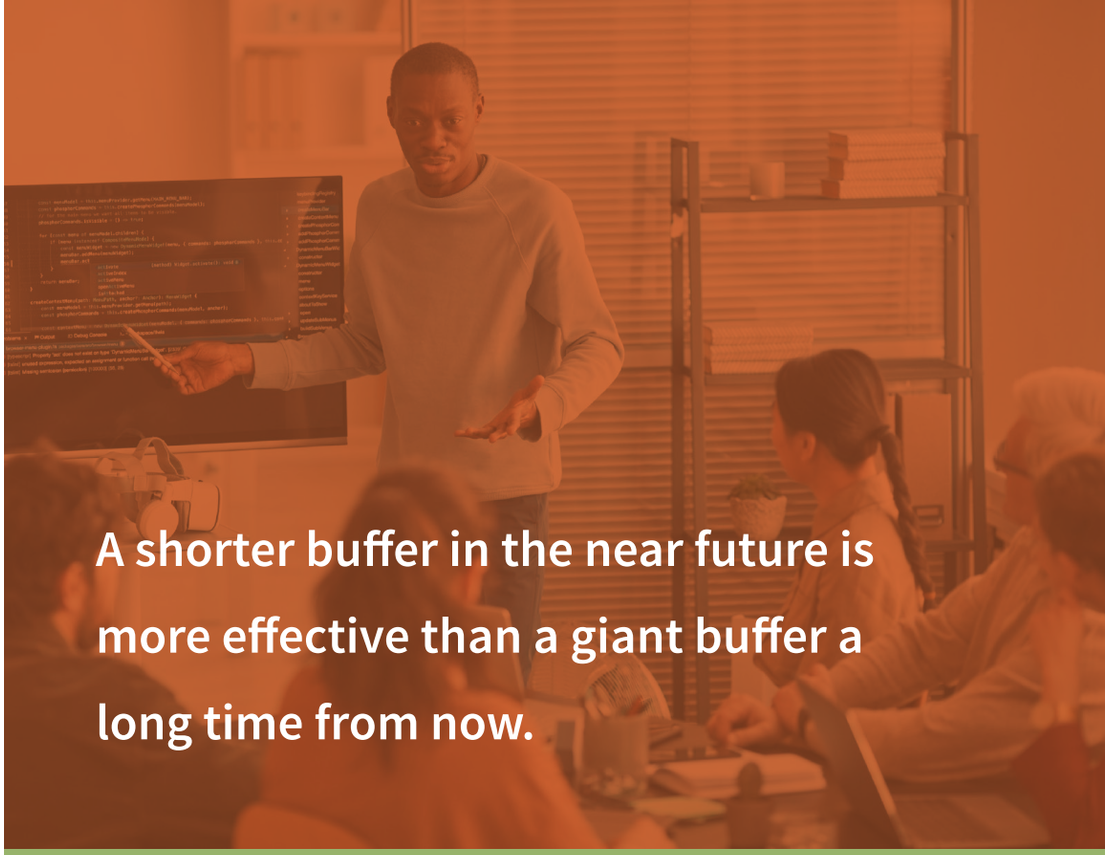
They use the buffer first, and then work on the task. But things usually take longer than we expect, so a thirty-day plan that had fifteen days of work in it will still take longer than we expect!

No wonder people are frustrated.



The solution to this came from ***Critical Chain Project Management*** (which came from ***Theory of Constraints***). It teaches to have people give a 50/50, or 'on average', estimate for their work durations. And then create a shared buffer at the end of the project that absorbs the normal cause variation. And the size of the buffer is based on the amount of uncertainty in the project and the number of tasks that are estimated. So now your project looks like this:





A shorter buffer in the near future is more effective than a giant buffer a long time from now.

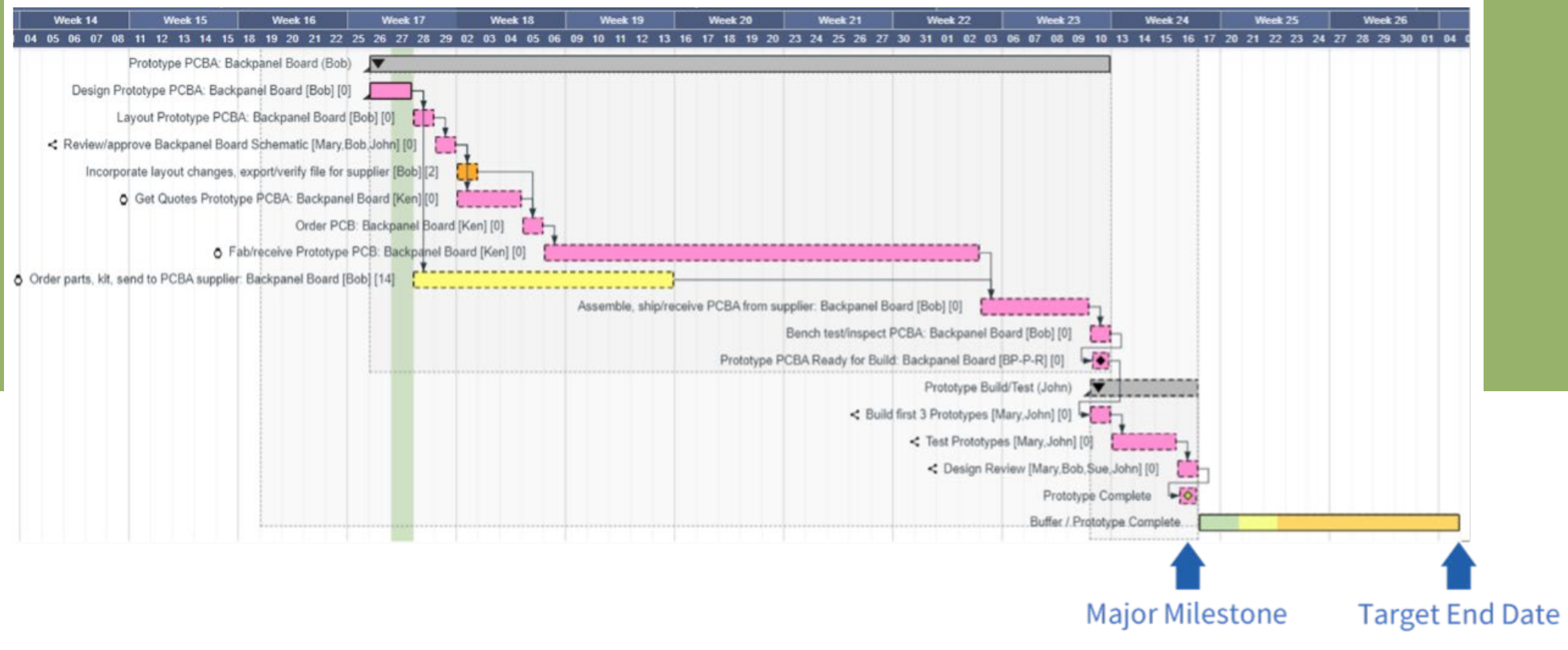
The benefits of this process are that the team members know that if they are late with their task, they will be consuming the team's buffer. So there is some psychological pressure. But in return for insisting they don't buffer their own tasks, we need to give them time to focus when they do start their task. We can't force them to multitask and still expect them to get their work done. So it's a two way street. And it works very well.

Critical Chain teaches to create one buffer at the end of the project. However, we modified this concept and create buffers at the end of major milestones within the project. Many new product development (NPD) projects are more than one year long, so ***it's much more effective to have a shorter buffer in the near future than it is to have a giant one a long time from now***. Also, the uncertainty or risk varies in different phases of NPD, so this allows you to size the buffer more appropriately for the phase you are in.

You can see in the image below that the major milestone is where the project would end if everything went perfectly, and the end of the buffer is the ‘commit date’ or date the team shares with management or the customer. Also, if this is the end of a phase, we plan the next phase to start at the end of the buffer. This prevents the final end date of the project from moving back and forth while changes are happening early in the project.

While it’s easy to see where the milestone is on any given day, this view does not show us the history of the changes, or how the milestone got where it is. This information is important if you want to understand what has impacted this project, and when you are trying to make improvements for future projects. ***So there’s a very slick way to track the buffer consumption that we’ll cover in the Reports and KPIs section later.***

Major Milestones Float while The End Date Remains Predictable



7

Standup Meetings and Visual Work Management

Now that there are tasks in the project plan, the traditional project management process is to have people work for two weeks, and then have a status meeting to see how things are going. But as we all know, that's not very effective. When the meetings are two weeks apart, it takes a lot of time to prepare, the meetings are long, and a lot of the information is old and not very helpful. In other words, it makes it difficult to respond and make course corrections.

However, the developers of Agile, or more specifically, Scrum, came up with the idea of having a short meeting every day.

Wait. A meeting every day?

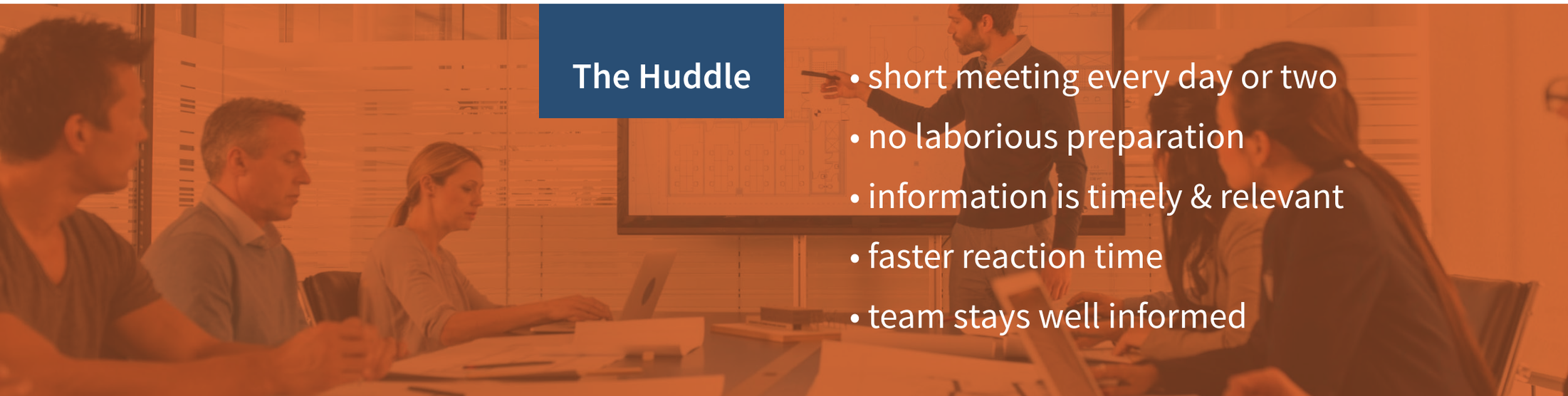
Most teams complain that there are too many meetings. So how could more meetings be a solution?

By having a very short meeting every day or two, there is no time spent preparing, and all the information is less than a day or two old. So it's very relevant and there's time to react to it. It also greatly increases communication and keeps people focused. And guess what. It also keeps everyone ***well informed***.

In the standard Scrum meeting format, each team member shares what they got done yesterday, and what they're planning to work on today. The goal is to finish the meeting in fifteen minutes or less, so any technical discussions that arise are tabled until after the meeting is over so the uninvolved people can go back to work.

Software teams usually look at a Kanban board during these meetings and move their tasks from left to right through a series of columns labeled with each phase of a software story (e.g., Backlog, In Progress, Review, Test, Complete, etc.) But tasks in hardware development don't all follow the same lifecycle, and most hardware tasks have to be done in a certain order — just like they're shown in the plan.

So, while Kanban boards work very well for software, they don't work for hardware.

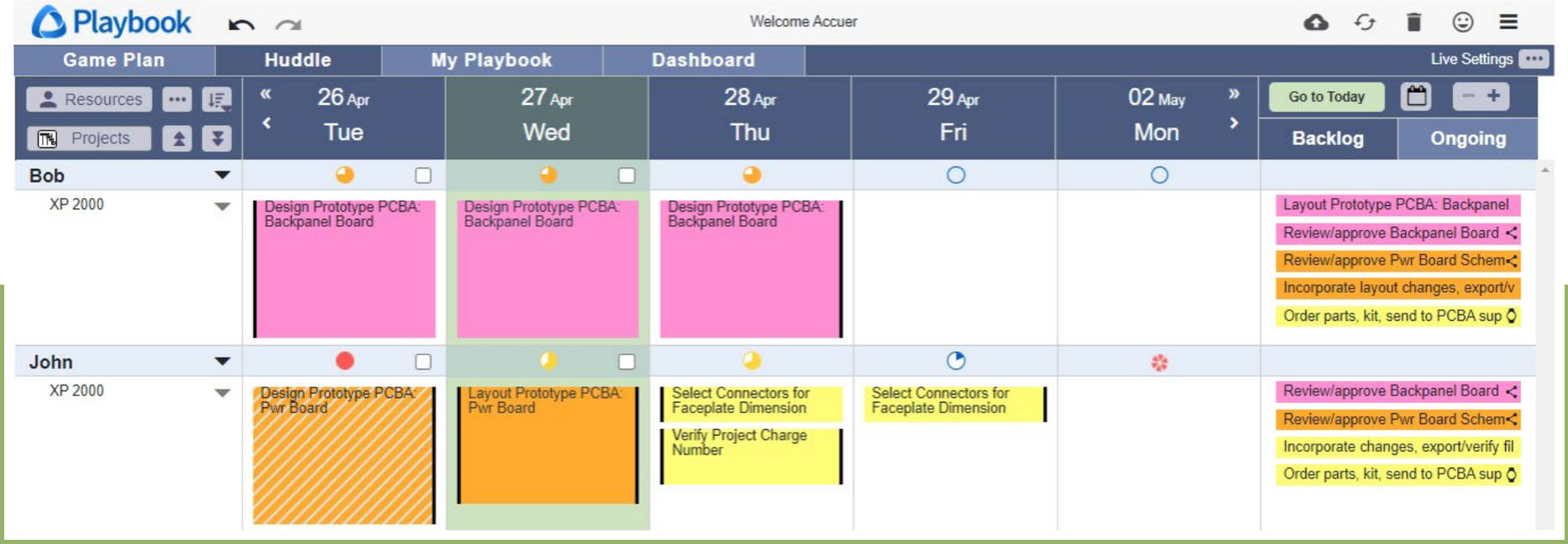


The Huddle

- short meeting every day or two
- no laborious preparation
- information is timely & relevant
- faster reaction time
- team stays well informed

So we developed a very different project board which allowed us to create a process for that standup meeting that is even more efficient. And once you know how to “read the board” in Playbook, you can see at a glance all of the important updates each team member would normally have to share verbally:

- What everyone worked on yesterday
- What everyone is working on today
- When a task was (or will be) completed
- When a task was (or will be) started
- What everyone will work on next (unless it changes before they finish their current task)
- If any tasks are out of date
- Whether anyone is overloaded
- Who is on the critical path task
- Whose tasks will become critical if they are delayed more than a few days
- Who has tasks that are blocked
- Who is on vacation
- Who is falling behind



Besides displaying all of that information, there one more thing this board does to solve our biggest problem.

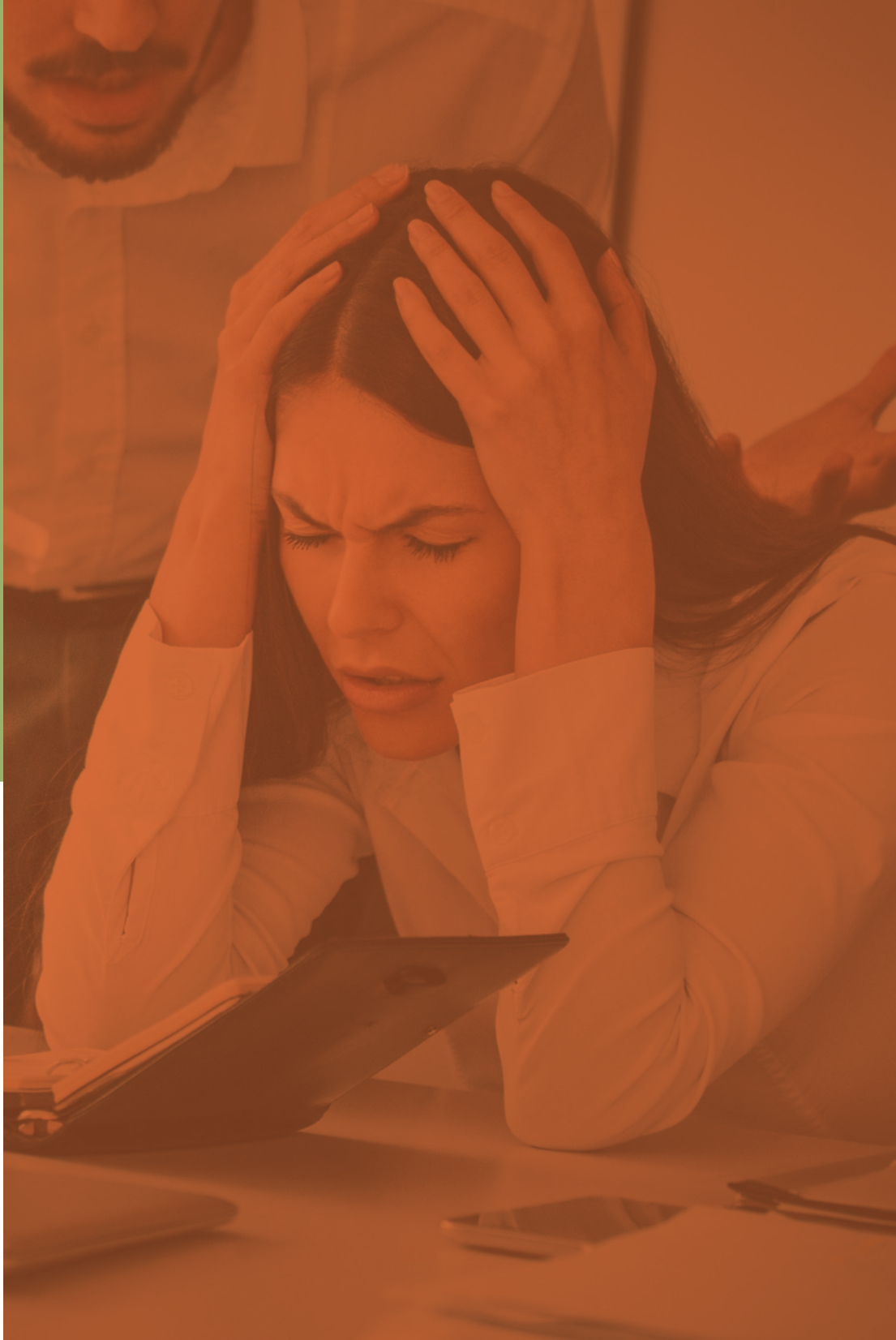
It prevents daily slips.

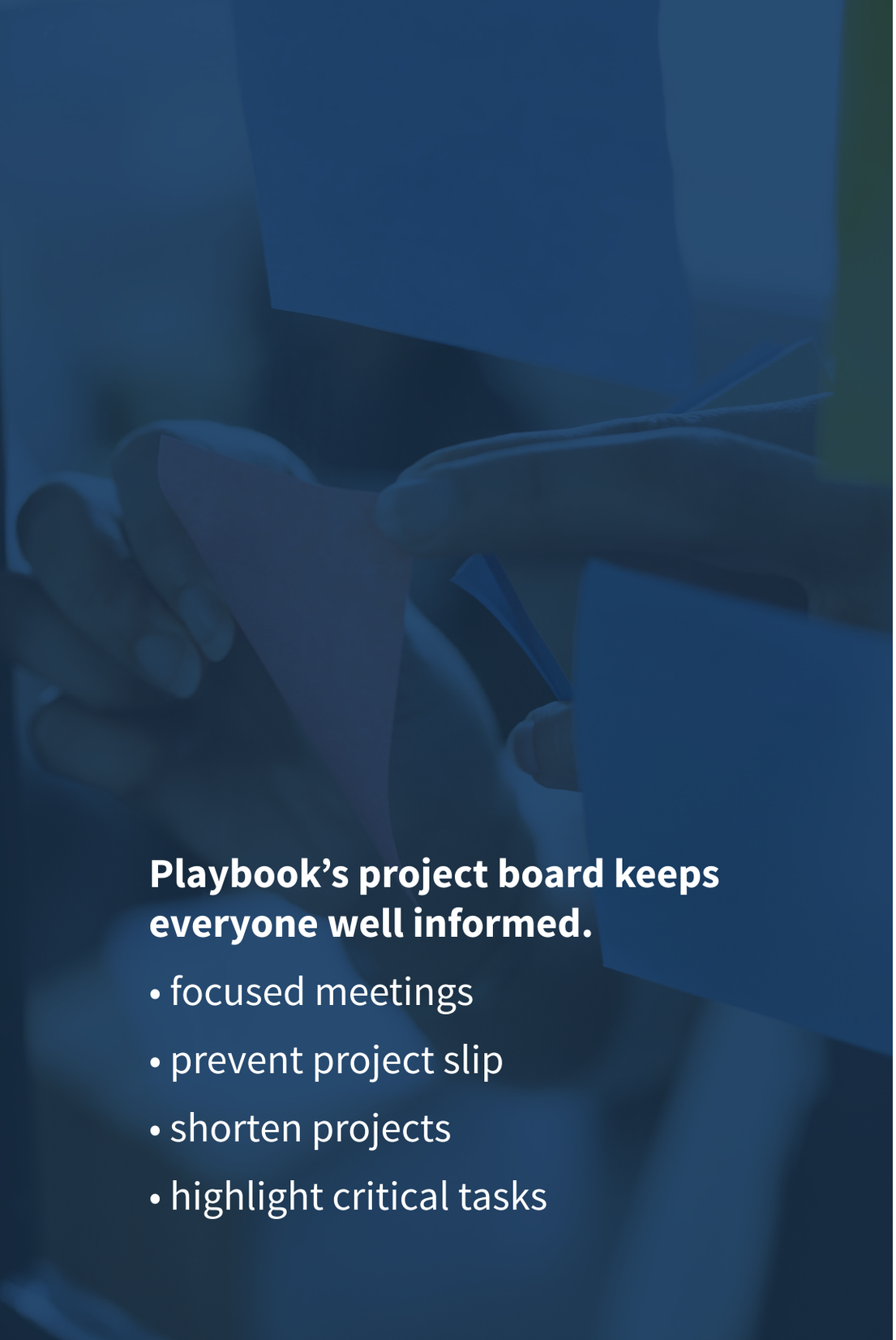
It not only prevents them, it predicts them in advance. And tells you instantly if they do occur.

On any given day, the only tasks that will delay the project if they don't get worked are the ones on the critical path. (For most small projects there is usually only one critical path at a time. But in large projects, or near the end of a project, it is possible to have multiple simultaneous critical paths.)

So Playbook's standup meeting board clearly highlights these tasks so everyone can see immediately if their task is on the critical path. And if it is, they know they need to focus until they get it done. And everyone else knows they need to support that person—don't interrupt them and offer to help if they can.

On any given day, the only tasks that will delay the project if they don't get worked are the ones on the critical path.





Playbook's project board keeps everyone well informed.

- focused meetings
- prevent project slip
- shorten projects
- highlight critical tasks

This simple feature prevents the biggest cause of delays on most projects.

It also allows us to change the focus of the meeting and make them much shorter and more effective than a standard Scrum meeting. Instead of everyone having to share their updates, the most important question each day now is, "Did you know you have a critical task? Are you working on it? Are you blocked? Is there any way we can help you?" And even those questions can be answered just by looking at the board.

So in reality, the most important outcome of a standup meeting with Playbook is to ***prevent the project from sliding***, and see if there's a way to ***shorten the project*** by helping the person with the critical path task.

This first step takes less than a minute, and then the team can continue the meeting and allow anyone to discuss other updates that might be important.

This is because there is no reason to have everyone verbally share what is clearly visible by looking at the board.

In summary, daily (or frequent) short meetings provide a lot of value. And Playbook allows you to skip the non-value-added chatter and get to the point. We have customers that have companywide standups where they review all of their high priority projects in less than ten minutes.

Imagine the entire company knowing what the critical task are each day, and who has them.

And if you happen to miss the standup? No problem. Just log in and look at the board and you'll know instantly what's going on in every project.

That's what we mean by "well informed".

8

Daily Task Updates

You may be wondering how all of these details are kept up to date.

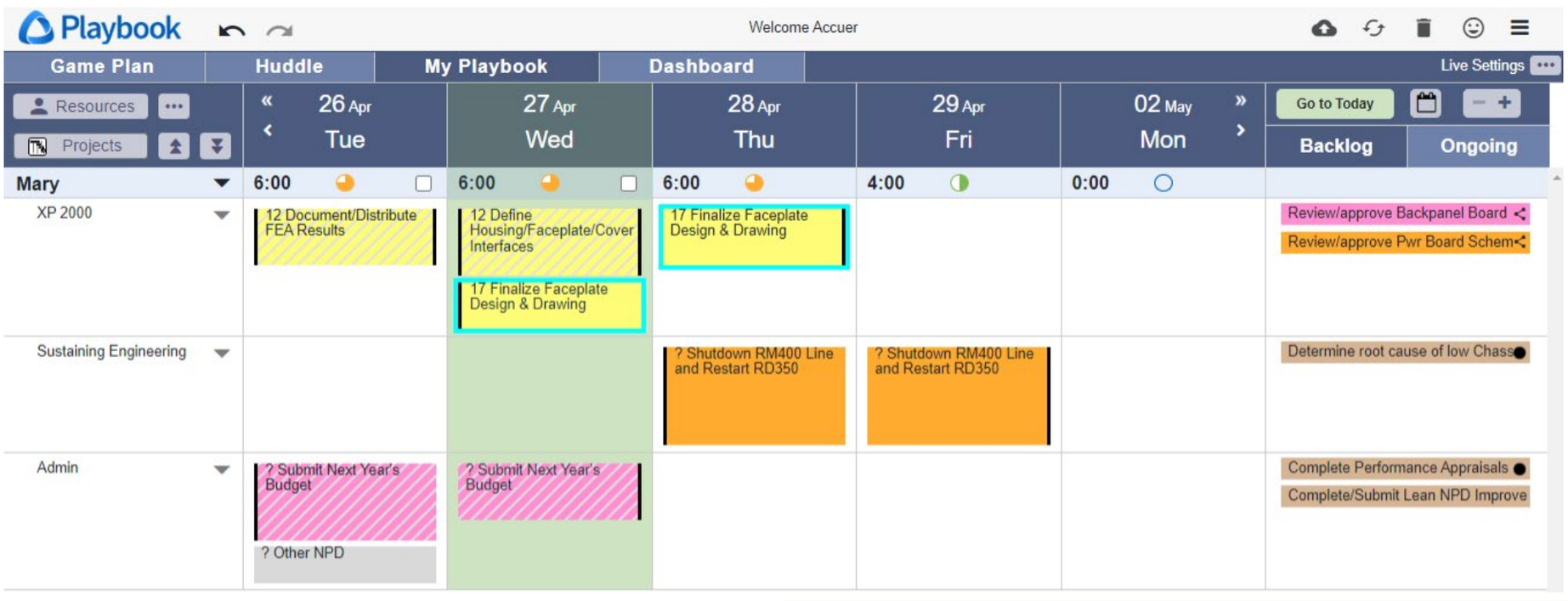
Good question, and the answer is surprisingly easy.

At the end of each day, each team member updates their tasks. The updates are very simple, and most people only have a few tasks per day, so it literally only takes two minutes.

And this has a lot of benefits.

It allows Playbook to look at all of these updates and re-sort the priorities for everyone based on all of the changes that occurred during the day. This also eliminates the need to waste time in the standup meeting asking what each person did. But most importantly, it ensures everyone has ***correct*** priorities, every day.

Playbook has a view for each team member that shows the projects they are working on, and what tasks on each of those projects belong to them.



9

Zero-Touch, Real-Time Reports and KPIs

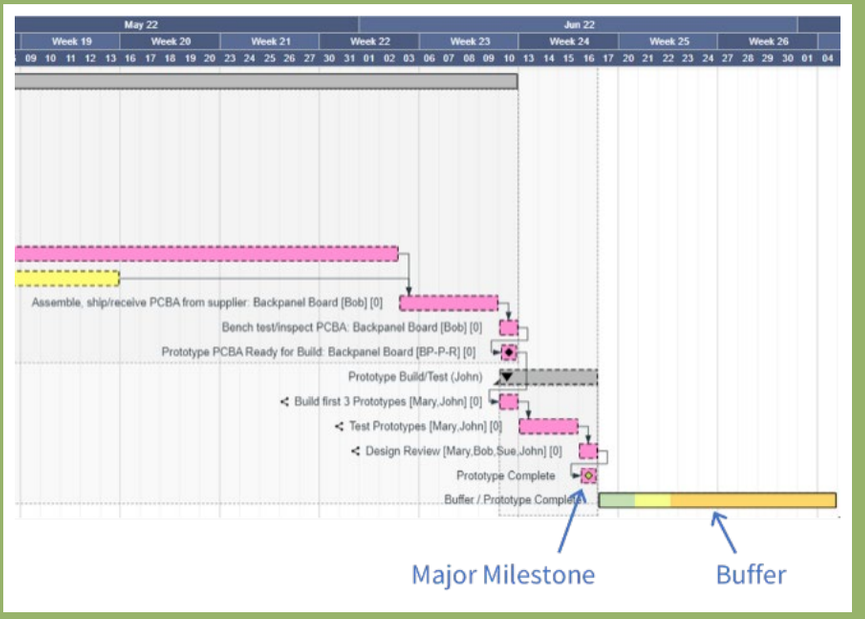
The final major component for any high performing system is the feedback loop. Reports on where we're at and what happened are great, but not all delays happen suddenly. Many of them are the result of things that can be measured a long time before the delay occurs.

So Playbook has KPIs that help you identify those situations and prevent the delay before it happens.

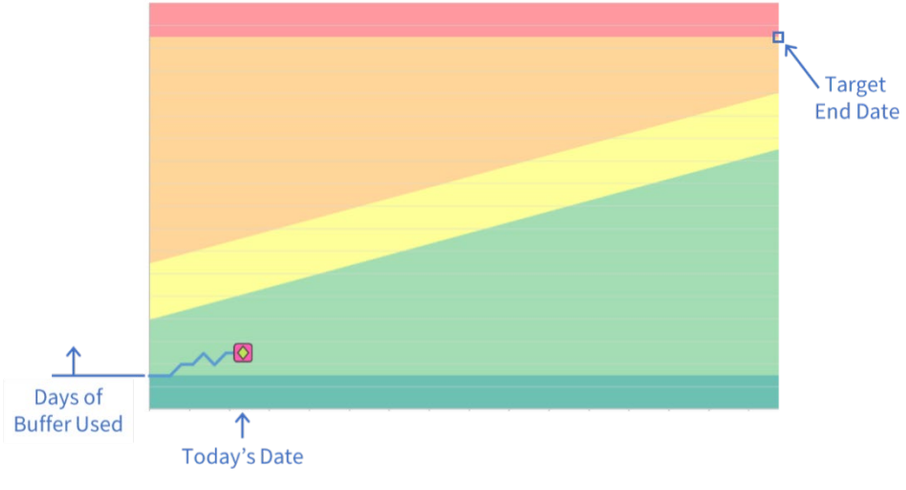
NPD projects involve many roles in an organization that each have their own needs around reports: senior management, functional managers, program and project management, and team members.

Senior management wants to know that their portfolio will be launching on time. The best view for that is a portfolio buffer chart that shows the status of every major milestone in the company.

But in order to understand it, we need to understand the buffer chart for an individual project. In the image below, you can see the buffer as it appears in the project plan.



Remember that the individual task durations are all unbuffered, or based on a 50/50 or "on average" estimate. But work estimates grow more than they shrink, so this buffer absorbs that variation. And in the view above, it's easy to see where the milestone currently is, but this does not show us the history of where it has been.



In order to see that, we create a graph with the duration of the project on the horizontal axis, and the buffer duration on the vertical axis. Then every time the major milestone moves, that change is plotted on the vertical axis. So it's very clear when every delay occurred, and how many days it was.

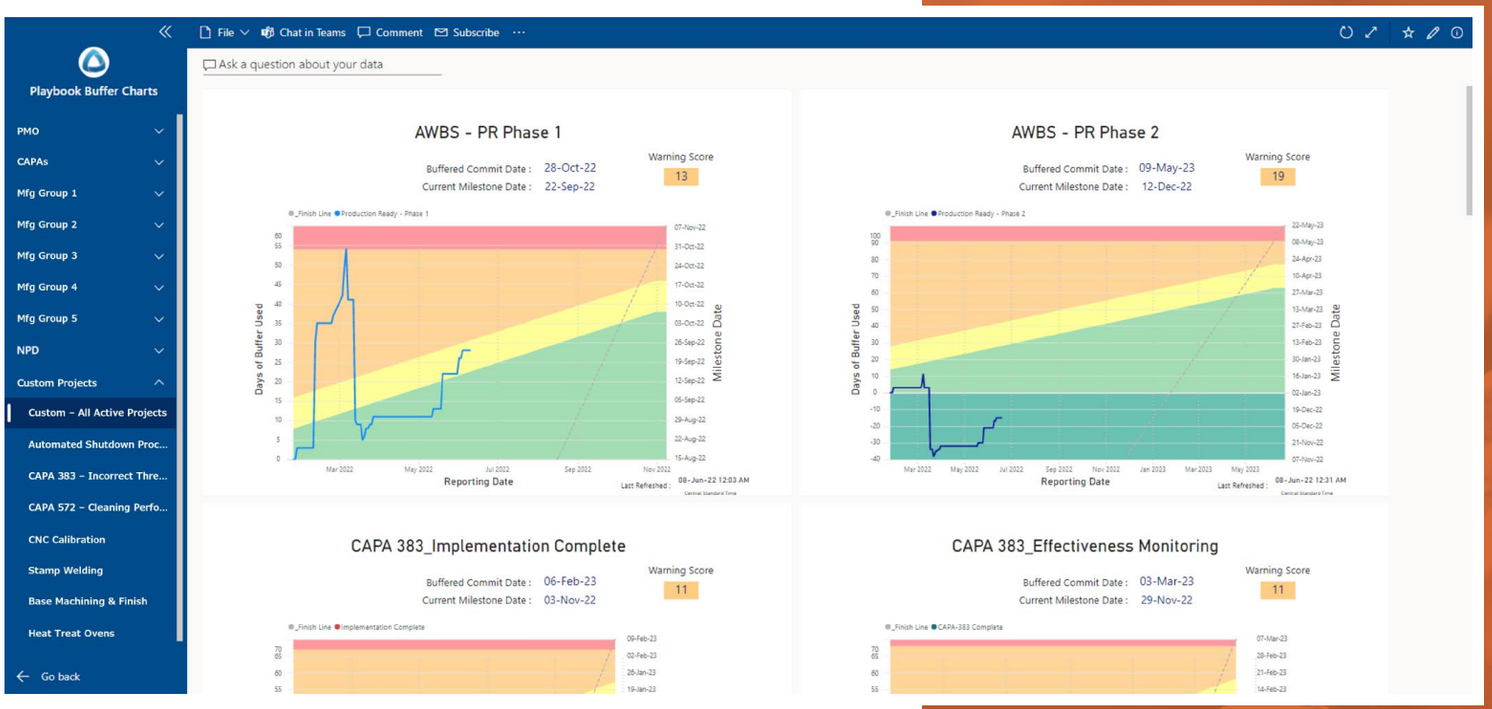
The predicted end date is at the top right corner of the chart where the orange turns red. And the colored zones represent the schedule risk based on how far through the project you are. This allows you to see the status of any project at a glance, as well as how rapidly the status is changing. You can also record the impact reason for any delay and see that when you hover over that data point.

Now that we understand the buffer chart, you can see how the portfolio status chart conveys the same information for every major milestone in every project in the company. And remember, this status is 100% up-to-date in real time. If someone updates a task that moves the milestone, you will see it here as soon as they save their update and you refresh your chart.

Portfolio Milestone Report

Once you've looked at the current status of all the milestones, you can look at detailed history of any milestone on its individual buffer chart.

Project Buffer Chart Dashboard



As mentioned above, this shows the current level of schedule risk as well as all of the delays (or gains) in the project so far. It also shows when the delay occurred, how many days it was, and what caused it. And the impact reasons can be gathered from all of the company's projects to look for systemic issues so they can be addressed and prevented in the future.

The reports above are important and answer "where are we at?" But the current status is the result of a lot of things that happened prior to today. So in order to predict the future, we need to look at the KPIs.

10

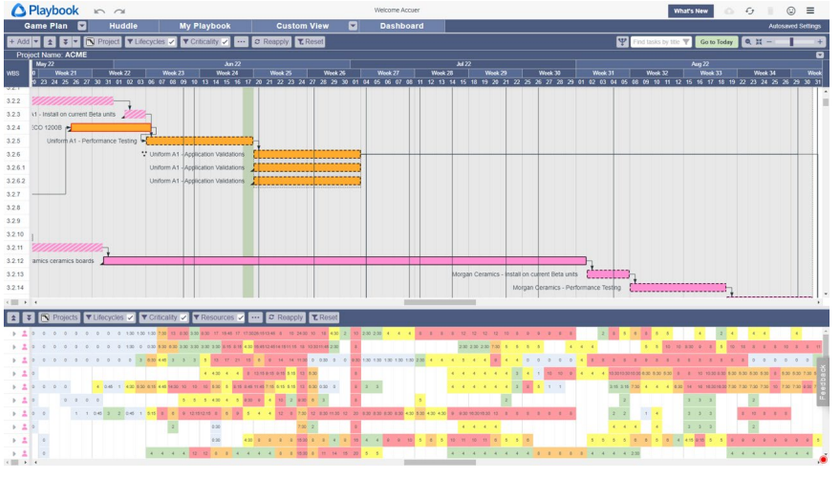
(Critical) Resource Loading

One of the key predictors of future delays is resource loading, and Functional Managers are generally responsible for this. Like every other project management tool, Playbook has a resource loading report that can look forward in any date increment. It can also report loading by individual resource, department, or role.

But Playbook's resource loading report has two additional features that are critical.

The first is the ability to look at resource loading 'by criticality'. Remember that ***Playbook places special emphasis on tasks on the critical path***. This is important because it's very possible for a resource to look overloaded in a report and cause a lot of extra work trying to mitigate that, when in reality all of the work might have enough slack that it will eventually get done and never impact the end date. Why worry about red cells in a loading report if they really aren't important?

Conversely, a resource might not look overloaded, but if all of their work is on the critical path, it's very possible they will cause the end date to slide before they complete it. In other words, it's critical that the loading report breaks down the work based on criticality.



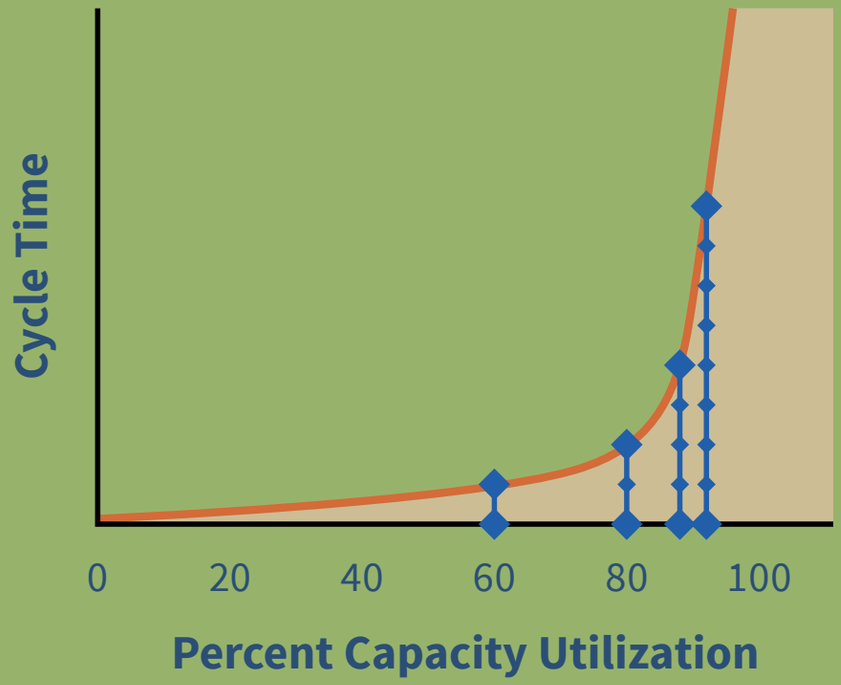
Also, because of the way Playbook sorts the future tasks for each resource, it is not necessary to perfectly load level the resources in the plan. As long as there is enough time in the future to get their total volume of work done before it runs out of slack, Playbook will help the resource self-level the work in the correct order. This is another big advantage over other planning tools that require the project manager to maintain the load leveling manually.

The other important difference is that ***Playbook also takes into account what a resource's true "availability" is for getting this work done***. Most companies load resources to eight hours per day. But it's nearly impossible for a person to get that much work done every day for an extended period of time. Most people are lucky to be able to focus for six or seven hours, and a lot of resources are really only available for three or four (or even fewer) hours per day for actual project work. So it's possible that the resource loading is off by a factor of two or more.

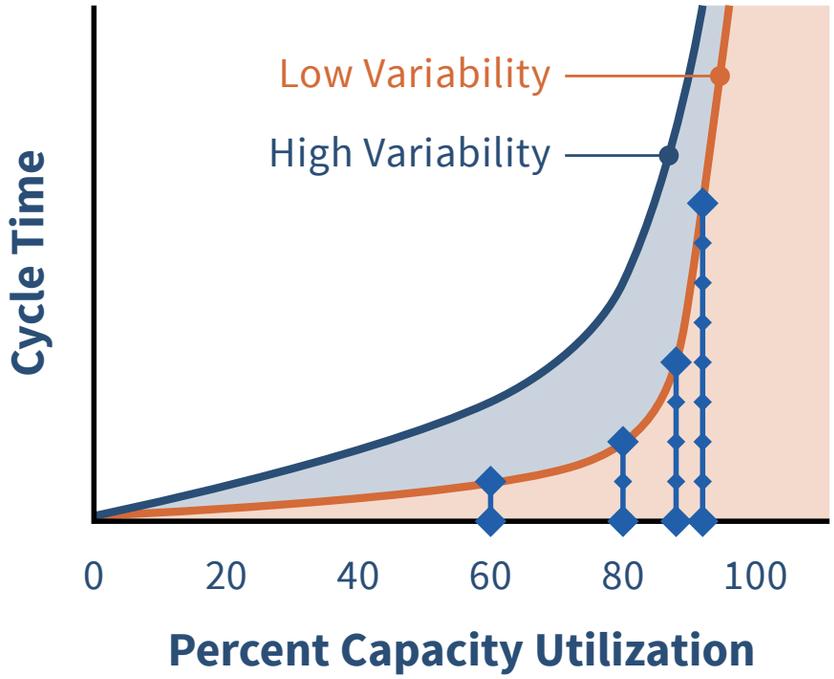
If that last sentence didn't scare you, we need to dig into resource loading for a minute. Overloading in a system is so important, and so often overlooked, that it needs to be covered here.

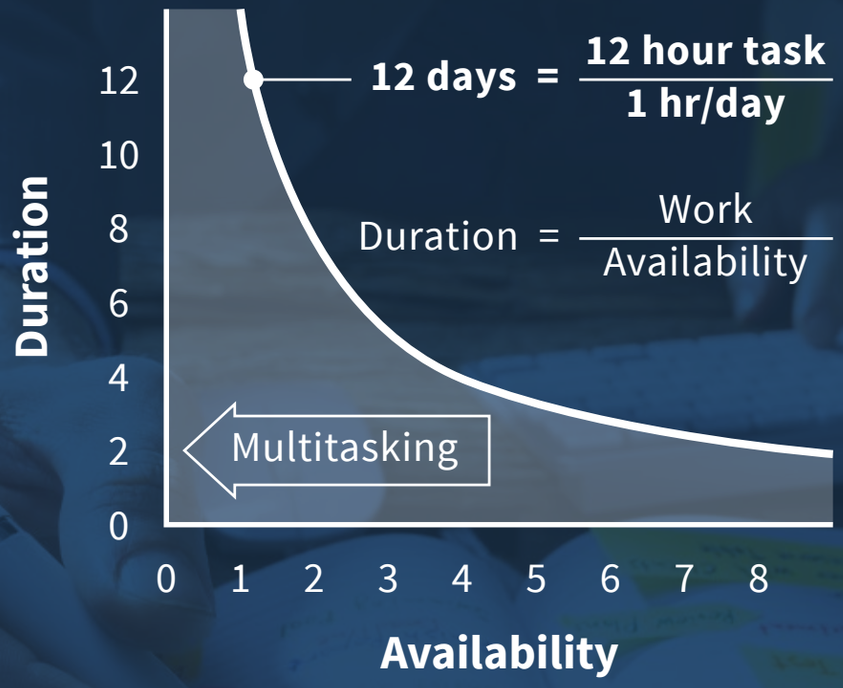
The graph below is of cycle time as a function of system loading. You can see that the cycle time increases gradually with increased loading until it gets to about 80%, and then the curve (impact) goes almost vertical. And to be clear, the endpoint of this curve is at 97.5% loading, so 100% is literally off the chart.

So why do companies load their resources to 100% in the project plans, and then are surprised when things take longer?



Manufacturing understands this principle very well and they maintain resource loading around 80%. Think about that—they intentionally plan to have about 1.6 hours of spare time each day in their system. (Wouldn't that be nice?!) But manufacturing has low variability in the arrival rate and cycle times of the incoming work. In environments like new product development, there is a high amount of variability in both the arrival rate of the work and the processing time. And systems with high variability ***need more spare capacity*** to keep the overall cycle times low and the throughput high. So that curve looks more like this.





But these curves are for overall systems. When you consider the loading on an individual, it is much more effective to consider their "availability". This is how many hours per day they are available to do project work. It's not realistic to expect a person to get eight hours of work done in an eight-hour day because there are always breaks and interruptions that deduct from that time. And many people have other responsibilities that take even more time from project work.

So it's possible that a person's availability for project work is more like five or six hours per day. And it's not uncommon for companies to discover that the true availability of some of their key resources is actually two or three hours per day. In other words, their system loading on the capacity graph above is actually two or three times higher than they think it is and probably in the 80% to 100% range.

No wonder things take so long!

That's why we designed Playbook to measure the availability of every resource over time so you can use the actual number for the loading reports. If your goal is predictable end dates, it's imperative you do this.

And finally, since Playbook indicates which tasks are critical, the people with low overall availability know exactly when they need to apply more time to a project or task so they don't hold it up. This is a big improvement over the traditional method of guessing and multitasking.

11

Summary

Hopefully by now you have a basic understanding of some of the principles from Lean, Agile and Theory of Constraints that can be applied to hardware projects.

And hopefully you've learned that the biggest cause of late projects are the invisible daily slips that accumulate over time and can easily double the length of your project without anyone knowing.

The cure to these daily slips is to ensure that everyone has ***correct*** priorities, every day, so the teams can be sure that the critical path tasks are getting worked on.

We mentioned at the beginning that our highest level purpose is to help engineers love their jobs again (and project managers too!)

So how do correct priorities help engineers love their job?

Gallup is a global analytics and advice firm that famously discovered the key drivers of employee satisfaction. The number one indicator of employee engagement was if they could answer "Yes" to the question, "Are my expectations clear?"

If correct priorities, every day, for all your projects, doesn't set clear expectations, we don't know what does.

Imagine coming to work every day knowing exactly what the most important thing is for you to get done.

And imagine what happens if everyone in the company worked on their most important task every day for a year.

The level of trust would be unprecedented, and the business results will follow.

Executive Summary

For teams that need to finish on time, Playbook creates ***unprecedented visibility***, which allows teams to ***respond to changes***, and still achieve ***predictable end dates***. Unlike every other project management or team collaboration tool, Playbook ensures every team member has ***correct*** priorities, every day, for all their projects.

As a result, ***team members*** are happier because they're not overloaded and forced to multitask, ***program and project managers*** have more time to do strategic work, and ***senior management*** sees business performance improve because programs are delivered on time, for less cost.

"I've had the luxury to deploy this software/ methodology twice at different companies with great success.

It's a life changer for me and my R&D teams."

— Raul Leyte-Vidal, VP R&D, Venocare, Inc.

We know that engineers like to test things before they decide to use them, so we have a Free Trial process that allows you to ramp up a small team and test the methods and software on one of your live projects—100% commitment and risk free.

To watch a four-part demo of these methods and software, **register here**.

To log into a Sandbox environment, **register here**.

To set up a call with an engineer to ask questions or start a Free Trial, **click here**.

Or email David at **dpaulson@accuer.com**.

Helping the best minds accelerate the delivery of world changing products.

Playbook

Visible. Actionable. Predictable.