

The Complete Guide to
**Managing Digital
Projects**

How to collaborate with clients, manage your team and projects, and get paid

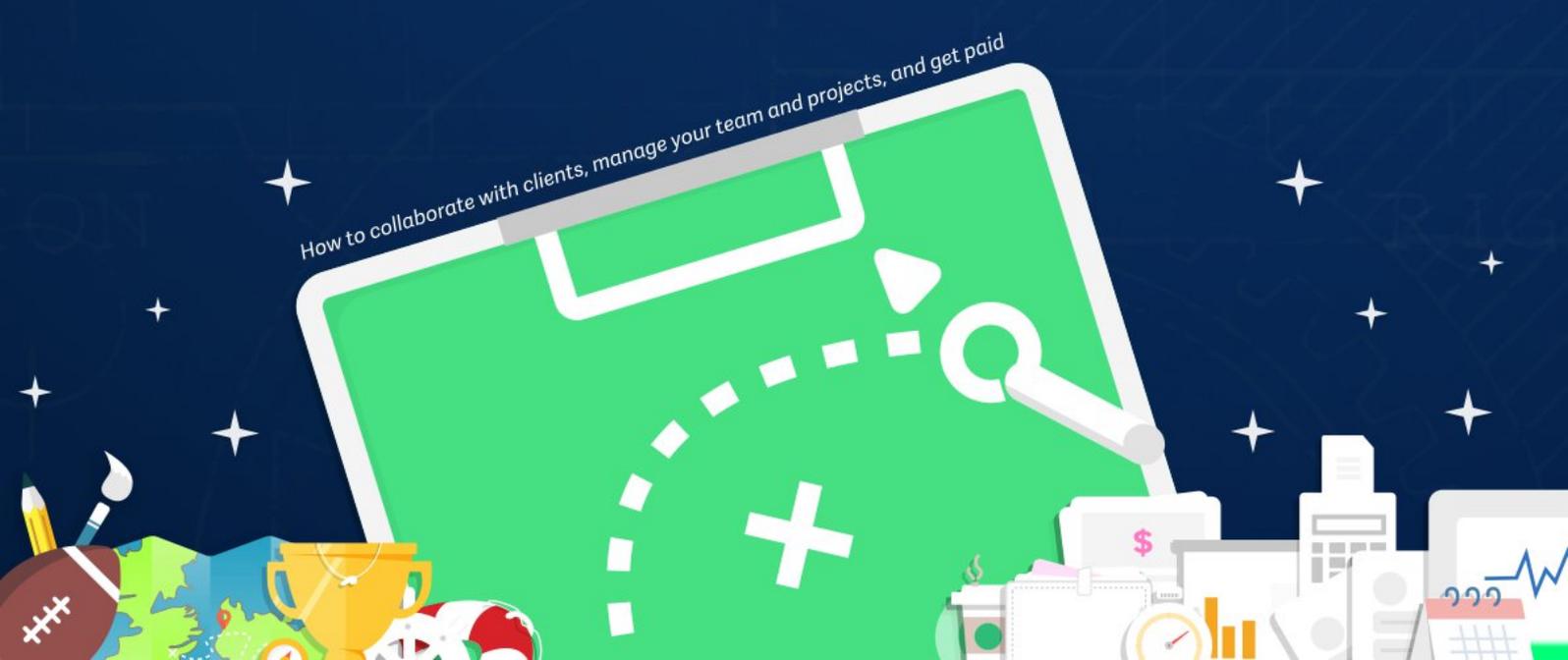


Table of Contents

04	<i>Chapter 001</i> Client Proposal Client Screening Giving an Estimate Tips & Tricks	71	<i>Chapter 008</i> Team Collaboration Keeping Everyone in the Loop Why Misunderstandings Happen How Not to Sound Like an Arrogant Jerk How to Collaborate With Your Team How Teams Form The Magic Formula for Good Teams How Can Developers Collaborate Better Why Developers and Designers Hate Meetings Collaboration Myths
10	<i>Chapter 002</i> Kickoff Meeting Before for the Meeting Starting the The Kickoff Meeting Agreeing on the Process Defining the Project Scope	90	<i>Chapter 009</i> Client Collaboration The Client Collaboration Process Is Broken Types of Clients Involving Clients From the Start Dealing With Non-Collaborative Clients Keeping Clients Updated Dealing With Change Presenting Work
18	<i>Chapter 003</i> Finance & Legal When and How You'll Get Paid Covering All Bases With a Contract Preparing for the Worst Case Scenario	102	<i>Chapter 010</i> Time Tracking Hourly Rates vs Flat Fee How to Track Time Estimating Work Time Cheating Being Too Efficient
27	<i>Chapter 004</i> Being Agile Modern vs Classical Project Management How Agile Helps Digital Projects	110	<i>Chapter 011</i> Billing Work How to Manage Invoices Special Invoicing Items Keeping an Eye on the Budget
32	<i>Chapter 005</i> Project Manager How Joe the Developer Perceives the Role Pixar Case Study The Many Roles of a Project Manager Re-Earning Cred Each Day	117	<i>Chapter 012</i> Beyond Projects Why Businesses Fall Apart The Three Elements of Strong Businesses
42	<i>Chapter 006</i> Project Planning Where to Begin Organizing 1,000 Tasks? Work Breakdown: Planning the "What" Project Timeline: Planning the "When" Task Assignment: Planning the "Who" Automate Planning		
55	<i>Chapter 007</i> Project Monitoring Watching Out for Bottlenecks Controlling Quality Working With People Staying on Top of Projects What to Do When Life Happens		

04	Client Proposal
10	Kickoff Meeting
18	Legal & Finance
27	Being Agile
32	Project Manager
42	Project Planing
55	Project Monitoring
71	Team Collaboration
90	Client Collaboration
102	Time Tracking
110	Billing Work
117	Beyond Projects

Chapter 001

Client Proposal



Negotiating with potential clients eats into your billable time - the time you could spend working on projects and earning money. But saying yes to the wrong client will cost you a lot more. That's why you need to screen clients, set the right budget expectations, and only take the projects that are right for you.

Client Screening

Everything starts with the client. They need something (say a new website) and they're interested in hiring you. But first, they want to know much it'll cost. How you respond to this question sets the course for the rest of the project; if you don't do it right, the project won't go right.

After they tell you what they need, don't be afraid to give them a **ballpark figure**. It doesn't have to be exact or detailed. Just say it looks like a \$15k project, but you need more info to give a precise estimate.

The point is to get the potential client over **sticker shock** as soon as possible and anchor their expectations. If they're not comfortable with that range, you can both save time and money by quickly ending a relationship that's not right.

If they say they understand, the window-shopper becomes a prospect who's worth your time. Now you can start thinking about the project and invest more time in the relationship.

Your time is limited so you have to manage it carefully between working with leads and your existing clients. Think of it like managing long-term and short-term goals: one is more urgent than the other, but they're equally important; your job, besides working with the current client, is to keep the pipeline full and be ready to jump on your next project.

A **screener** is the best way to quickly assess if you're right for each other. It's a set of questions you give to a potential client to determine if you're the right fit for the job (only you won't name it Client Screener, but something positive, like Project Assessment).

In it, you ask:

- what the budget is and whether it's approved,
- the timeline and the goals of the project,
- and the requirements.

Based on their response, you'll know if you can take the project and start devoting more of your (billable) time to pursuing the business.

It's perfectly reasonable to want to **know the client's budget**, not

because you want to charge them that same amount, but to tell them what they can get for that money and guide them to an acceptable solution.

Once you know how much they can set aside for the project, tell them if they can have all the bells and whistles (what their marketing team envisioned), or if something else would be more budget friendly, yet still aligned with their goals.

Giving an Estimate

Give them a general **item-by-item estimate** and explain the reasons behind each item. An itemized quote tells the client you're a pro and didn't just pull a number out of thin air. It should communicate the price as a meaningful calculation which took a lot of thought.

It'll also help them come to terms with the price. It's like when your mechanic gives you an unattractive price, but once they break down the cost (new parts vs. their hourly rate), it makes sense.

You don't have to create an estimate that's too detailed. Just **break down the project's major parts** and walk the client through. Highlight the benefit behind each cost and point out what's crucial and what's just nice-to-have. This will help your client prioritize their needs and agree on a budget they're comfortable with.

Some items (eg. research) **can't be removed** because they're an integral part of the project as other phases depend on it. Explain that with confidence and stand behind your quote. You have a good reason why you quoted that price and you have to show that. Don't cave in under pressure, as they're just doing the same thing as you - trying to get the best price.

They're not buying your time, **they're buying work and value** they get from that time. You can make a site for \$10k yourself, but you can't make \$100k from it - but they can. That's what you charge for: the value they'll get from your work.

Once you win the project, have a kickoff meeting to get a clearer idea of the scope of work, and create a **detailed estimate** that they'll sign off on. Keep in mind that the project scope is prone to change and your original estimate doesn't have to be the same as the final bill. That's why it's called an estimate. You'll have plenty of time to communicate any additional work and costs that pop up during the project.

← Estimates

Resend Mark as Won Mark as Lost Edit Estimate Download PDF 🗑️ ⋮

This estimate is won. ✔️ WON

[Edit design and your company info](#)



Digital Solutions

Digital Solutions
Creating creative works since 2006. Experts in the field of web design, custom development, and branding.

For
Ethel's Bicycles
6109 Heritage Drive
Ashland, OH 44805

Estimate for Bikes Website
Total: 11,327.00 USD

#	Item	Quantity	Unit Cost	Amount
1	Research	1	1,000.00	1,000.00
2	Information architecture	1	800.00	800.00
3	Design - main pages	5	150.00	750.00
4	Coding - main pages	5	250.00	1,250.00
5	Design - blog	1	140.00	140.00
6	Coding - blog	1	350.00	350.00
7	Content - articles	12	50.00	600.00
8	Feature - SEO optimization	1	400.00	400.00
9	Feature - analytics integration	1	300.00	300.00
10	Feature - search	1	100.00	100.00
11	Expense - images	16	25.00	400.00
12	Expense - hosting and domain	1	50.00	50.00
13	Testing	1	900.00	900.00
14	One month maintenance	1	200.00	200.00
15	Two reworks	2	1,200.00	2,400.00
			Subtotal	9,640.00
			VAT (17.5%)	1,687.00
			Total USD	11,327.00
			Amount Paid	0.00
			Balance Due USD	11,327.00

An estimate with an itemized breakdown

Tips & Tricks

Beware of small jobs. Small jobs have the same overhead costs as big jobs but come with a smaller budget and tighter deadlines. You'll have to set aside your time and resources for the work, only you won't get a good return on investment. What seems like a quick win almost always turns out needing more work - while the budget remains the same because, you know, it's just that small thing you can get done quickly.

Prepare a 14-page document which outlines everything you do, plus give examples or links to sample work. You can even go a step further and tailor it to each client, giving them a teaser of what they can expect if they hire you. For example, SEO agencies do a quick keyword analysis and include it in the document; it takes only a minute to analyze the website but it'll tantalize the client and make them want to know more.

When communicating with prospective clients, **work in shifts** with someone so you can respond promptly to each client message. Aim to respond in under 5 minutes. Timely communication is important but do this early and you'll score some major points.

Keep a **one-page portfolio** close at hand so you can quickly send it to a client if they want to know what services you provide (and by implication, services you don't provide). It's like an offering card that sets clear boundaries of your work and clarifies your range of activities.

Call yourself a consultant and not a freelancer. When people hear "freelancer", they perceive your work and time as less valuable. Freelancers are perceived as staff augmentation: because a company has no long-term commitment to you, it's ok to give you the dirty jobs which will save them time and money. Consultants, on the other hand, enhance business and bring unique expertise that the company can't afford to hire full-time.

Appear bigger than you really are. If you're a freelancer or a small team, register yourself as an LLC. This way, a client doesn't hire you, the person, but another business. To gain more credibility, some companies rent a virtual office in big cities like London so they can say they're UK based agency, even if most of their workforce is somewhere less glamorous.

Don't present yourself as the CEO when talking to the client. This puts you in a weaker position: you're telling them they're so important that

they get to talk to the boss. Instead, let a project manager communicate with the client under your supervision and get involved in the closing stage where you can leverage your title better.

Make it easy for a busy person to say yes when pitching to potential clients. Spend a few hours studying their business and create a document outlining every change you'd make to their website, for example. It's easier for clients to tweak the document and say "I want this and this, but not this" than to start with a blank slate. Then you're the one framing the discussion, which isn't about whether they should hire you, but what they should hire you for. Be cautious though: it's a very time-consuming strategy, so use it only if you don't have enough work and/or really want the client.

With that said, **don't accept spec work** before being properly commissioned and have a contract. If someone tries to convince you to work for free (and that it's actually a good thing for you), don't be shy to tell them how inappropriate their request is. Would they have the nerve to ask a lawyer or a maid to work for free? Respect your profession, stand up for yourself, and tell clients to refrain from proposing free work like it's an opportunity - the only opportunity any work should elicit is the opportunity to earn money and make a living.

Trust your instinct. The point of this initial phase is to see what type of client you're dealing with. If your gut tells you a client is more trouble than it's worth, you're free to turn down the proposal. You won't be the first one to do it and it's probably the best course of action for both parties. Explain that you're not the right person for the job and direct them to someone who is. There's no point in trying to fit a square peg in a round hole.

Chapter 002

Kickoff Meeting



After you give your client a rough proposal (and they say yes), you'll be tempted to start working right away - but that's not such a good idea. First, you need to flesh out the project scope, agree on how you'll be working together, work out a payment schedule, and sign a contract - and then start getting actual work done. It's important to set up ground rules from the start if you don't want to spend the rest of the project extinguishing fires.

Before the Meeting

After the client hires you, **find out everything** you can about them. You already know a lot from what you discovered during the business development phase. Now, delve deeper and focus on providing value by solving their problems.

Use your **previous work experience** to generate some ideas. Think of similar businesses that you helped, brush up on them, and have them in mind so you can offer examples on the fly during the meeting. In the client's mind, previous success translates to future success. Show them you know exactly how to get them from point A to point B.

You're running the show and you want the client to trust your judgement - after all, they hired you because of your experience. By establishing yourself as the **expert**, you're setting up the tone for the rest of the project, and ultimately the final presentation. This is the best way to prevent the client from hijacking your work in a pang of distrust.

Write a short **meeting agenda** on a piece of paper in bullet points so you don't ramble randomly while skipping the important bits. Your agenda should cover:

- getting-to-know-each-other phase,
- agreeing on the collaboration process,
- defining project scope and budget,
- signing a contract.

Don't worry if the conversation takes its own course - let it flow but make sure you've crossed off each item.

Before the meeting, ask the client to send you a **standard contract** beforehand (if they have one) so you can see what works for you and what doesn't. The bigger the client, the more likely they'll have their own legal department, and the more compromises you'll have to make. This way, you'll come prepared and settle on an agreement faster.

Note that you don't need to prepare a **detailed project plan**. In fact, it's detrimental. The plan will be more accurate and emerge naturally once you sit down and talk it out. You can't afford to plan every detail only to hear the client doesn't have the resources, had different dates in mind, or decides on more/less work. Have a general plan in mind, including the milestones and dates, but keep it flexible so you can tailor it to the client's needs on the go.

You also don't want to arrive to the meeting with sketches, mood boards, mockups, or preliminary user research. First, you can't do any of that before you don't fully understand the project goals. Second, the client didn't get a chance to be consulted and making the pitch can look like you're not flexible. And last, you haven't been paid to work on anything yet.

One way you can make sure your prep time doesn't go to waste is by charging a **commencement fee** before meeting. If they're serious about collaborating with you, they need to understand you don't work for free. It's about trust, risk mitigation, and gaining leverage. If you're dealing with a big organization or a government institution, you don't have to be so strict because it puts an undue burden on the client with a complicated bureaucracy. In that case, have the meeting and let them pay later.

Starting the the Meeting

Every project officially starts with a kickoff meeting. It's best to meet the client in person because communication is more natural and fluid. But if you're working for a client that's far away, Skype is usually the norm. In that case:

- check if your headphones and camera work 30 minutes before the call,
- dress up and smile even if you're not at your office and the camera is off,
- find a quiet room you know no one will enter and put an intimidating "do not disturb" sign on the door.

Start the meeting by expressing enthusiasm and thanking the client for choosing you. Tell them a bit about yourself and present your team like they're superstars in their respective fields. Show what each team member does and share a story that highlights their talent. Clients are looking for reassurance and when you present your as experts, It'll subconsciously reinforce their trust and help them rationalize choosing you - because if you succeed, they succeed.

Then let clients **introduce themselves** and their team. You should know everything about them before the meeting, but be polite and let them talk. Let them know how you're going to need their expertise and can't wait to work together. Note who's responsible for what and make sure they feel acknowledged by asking for their contact information - even if you think you won't need to consult them later. You want them to be excited to work with you, knowing their contribution is appreciated.

Start the project discussion with the **client's goals**. Ask them what they want to achieve with the project, what they've done so far, and what they expect from you. Write down the exact words they use and how they phrase the problem - you can reuse those exact word when you present your work. Remember that the first meeting is all about listening (and the final meeting is all about talking).

Big companies usually have a huge list of clearly **defined goals** and know exactly what they want. They probably use KPIs to measure them and have forecasts for each quarter. In contrast, small companies with less experience can have a vague notion about what they need, so you'll have to fill in the blanks and talk about their goals for them. This is a good time to mention your experience, and show you know exactly what they need and how to get there.

Business goals are pretty much the same across all industry types so if you worked on one e-commerce website, you'll already know they want to maximize the number of sales and make purchases easy. In that case, you don't need to focus on getting newsletter subscribers or worry about ad placement, like you would when designing a blog or a web magazine. If you brushed up on your previous projects, you'll nail this.

Ask the client about their **company structure** and available resources. You want to make sure they can maintain your work once you leave. You don't want to end give your client a website that needs four full-time Ruby developers just to keep it going. You want to show the client you're thinking about the project's long-term sustainability.

This will also help you know whether to include training and other services in the final estimate. You need to discuss the need for **future work** (like bug fixing, maintenance, training and education), and if they'll keep you on a retainer.

Agreeing on the Process

To avoid agonizing over why the client is calling you at 1AM or wants you to redo your work for the fifth time, always agree on the **collaboration process** beforehand, namely:

- how often you'll communicate,
- who contacts whom and when,
- how and when feedback is given
- who's in charge of the review,

- the number of reworks (and cost).

You can talk about your workflow and methodology, but it's usually too much information for them. They pay you for the final product and fulfilling their goals - talking about how you do sprints on a Kanban board doesn't contribute to that.

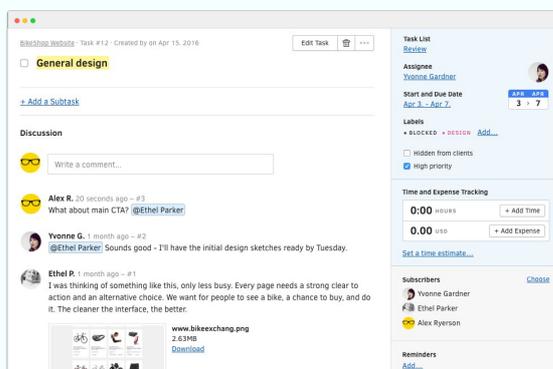
But they need to know **what you expect** from them. So for example, if you have To-Do, In Progress, and Review project phases, they need to know when you expect from them to provide feedback or reply to notifications where they're @mentioned.

Investigate how much they want to be involved and how willing they are to collaborate using a project management tool. Then you can propose to either:

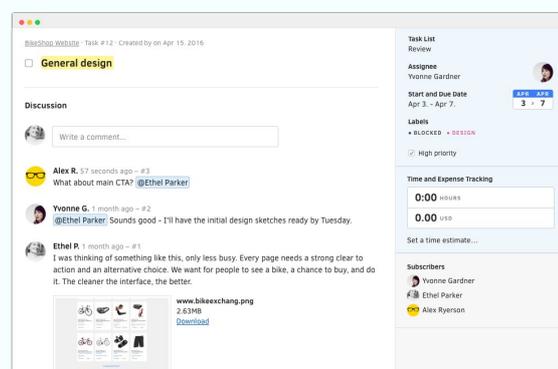
- invite them to Active Collab, let them see everything, and track your work, time logs, and budget
- communicate on an as-need-to-know basis via email and meetings/calls after each milestone.

Active Collab was made to ease collaboration so there's no separate client-side. This means clients can see everything you're working on and what they pay for. When you invite a client to the project, they'll be able to see tasks you're working on, comments, notes, discussions, and time records and expenses.

To make sure they see only what's relevant, each item can be hidden from a client. Hide with caution. When a client opens the project, they'll be unsettled by emptiness.



What a **team member** sees



What a **client** sees

Using a collaboration app lets you **open up your project** so clients can catch a glimpse of the process and see you're a hard-working professional. You'll also spend less time debriefing because they can track progress themselves. When you involve a client early and give unrestricted access, it gives them a sense of ownership and helps them champion the work to their management.

The **main concern with this level of transparency** is that clients will judge your work while it's unfinished and jump to conclusions. And it's a valid concern. If you make a mobile app that works perfectly but the design isn't polished, the client will latch onto the color scheme and won't be able to appreciate the complexity behind the app - all they'll only focus on how ugly it looks.

This is mainly a **client management issue** and a project leader's job. The above problem can be solved by using sketches. They don't look anything like the final product but they can serve to illustrate the functionality without distracting your client with other things..

The number one reason **why projects fail** is untimely communication. By opening up the project, you're letting your client see work in progress and raise red flags if they don't like something. It's a check-up mechanism that prevents false starts, and backtracking. Sure, you'll have to spend more energy explaining unfinished work to the client - but you also lower the risk of losing time and money. The earlier you can catch a mistake, the less it'll cost you.

You need to establish who all the **key players and decision makers** are and who has what authority over the project. When you have this information, you can involve those people early on and get their buy-in to avoid any grudges. This is especially true if their field of expertise overlaps with yours. For example, if you're a design agency, make sure you to make their in-house feel important: they can be your best ally or hinder you every step of the way.

Ask for the client's **activity timeline**. You want to know how much attention they'll be able to give you and when they're be busy. This includes vacations, company events, and each department's peak activity. You can later use this calendar to know the best time for contacting the client so you get their full attention. The last thing you want to do is schedule a presentation when they're stressed and prone to saying "yes" to anything just so they can get it over with - only later to tell you they don't like what you've done.

Finally, agree on the ground rules, like what are the **off-hours** when you don't work and can't reply. You don't want to let the client think they own you and that you must drop everything the moment they need you.

Defining the Project Scope

This is a good opportunity to talk about **milestones and deadlines** because that's when you're officially presenting your work and asking for feedback. This part overlaps with payment talk because the client will want to know the costs, how much time you'll spend on it, and the timeline.

But defining project scope shouldn't be **overwhelming for the client**. There's a lot to take in during the meeting and the client can get bored or lose sight of the big picture. So be brief. Outline all the steps and describe how each contributes to their goal. Always keep in mind that the point of the meeting is to see what the client needs, then agree on the dates and the budget.

You can start the talk by **iterating the agreed estimate**, going over it once again, and clarifying each item. Outline the project phases, deliverables, time estimates, and show how each phase contributes to the final product and how it benefits the client's goal.

First, suggest as a professional **what the client needs** and when. Then calculate how fast you can deliver it, how many man-hours you're going to need, identify whether you have the resources to do it, and if you'll need outside contractors. Then negotiate.

Focus on **fulfilling the client's goals** and let the price be an afterthought. You don't need to constantly bring it out unless you sense that the client is extremely price-sensitive. But if they use you for hand work, keep them updated about your hourly rates so they don't get surprised when they get a huge bill. If you do great work, better than the most, don't advertise your affordable pricing. Decide what your competitive advantage is and focus on that during the meeting.

CODING \$40-75

Backend development
Frontend development
Custom coding
Compatibility engineering
CMS customization

MARKETING \$45-65

User research
User testing
Consultation
Sales funnel setup
SEO, SEM

DESIGN \$35-60

Information architecture
UI design
Interaction design
Mobile design

PRODUCTION \$25-55

Video production
Motion graphics
Graphic design
Custom illustration
Copywriting
Content production

OTHER \$25-45

Maintenance
Localization
Content migration
Hosting setup
Technical support
Quality assurance
Education and training

If the client asks you for a **guarantee** that your work will improve their business, ask them to add an incentive clause to the contract and pay extra if you hit the goals. They'll move to the next subject before you finish the thought.

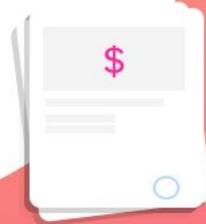
When talking about the scope, have a **checklist** with you because you're sure to forget something during the meeting.

Once you know everything that needs to be done, **outline all the project steps** and indicate where and when you need the client and what their responsibilities are. There are some services they may not even be aware they need - like content migration or accessibility compliance - this helps them discover and agree on whether they'll handle it themselves or hire you.

The project **scope will change** during the project, but if you do the project kickoff right, it shouldn't deviate much. But even if it does, it can be a good thing. If you get more work and the client is happy, it just means you're great at collaboration.

Chapter 003

Finance & Legal



Once you've covered the project scope and collaboration process during the project kickoff meeting, it's time to talk about money. You have to protect yourself from undercharging your work. Never start working on a project before you make sure who pays what, why, and when - and have a backup plan if the project fails.

When and How You'll Get Paid

You may be shy when it comes to talking about money. But what about your landlord, electricity provider, IRS, and your employees? They're not shy when it comes to money.

You provide a valuable service and if you're going to be coy about the payment, don't expect your good manners to be appreciated. If you want to keep the lights on and make the payroll, approach the conversation as a professional and never feel intimidated.

As you iterate the proposal and talk about the project scope, you'll **refine the final proposal** on which the client will sign off. The estimate doesn't have to be the same as the final total you're going to bill. You'll discover new information during the project and the scope will change; you can't predict everything this early. But it's good to have a foundation and then adjust the budget when you revisit the contract as the project unfolds.

Some clients aren't comfortable with hourly rates. You can say work will take 40 hours, but it might take more - the client has no guarantee the project won't exceed the budget. You can address this concern by **capping your fee**. For example, if you estimate 40 hours, cap your fee at 50, and work for 55 - then charge the client for 50 hours.

An alternative to hourly rate is a **fixed fee**. Clients prefer fixed fees over hourly rates because there are no surprises and the risk is transferred from them to you, which can be both good and bad. If you're productive and quickly finish the project, you're rewarded; but if you underestimate the time, you'll end up undercharging your services.

The client will **try to lower the price**. Don't jump and change your hourly rate - instead, negotiate the scope of the work or ask for concessions and more favorable terms (like extending the deadline, getting your

- Choose a **fixed fee** if the project is formulaic, simple, you've done similar projects and know exactly how long it'll take. This way, you can charge more and the client will feel more secure.
- Choose an **hourly rate** if the project has a lot of moving parts, you don't have much experience, and can't be 100% sure in your estimate. This way, you're protecting yourself from uncertainty and from clients who want more work than they've paid for.

name on the work, or the permission to reuse it). See how the project can be scaled down and where you can cut corners while still fulfilling the client's goals.

Save talking about the **payment schedule** for the end of the meeting, when you work out the project scope. For a smaller project, aim for 3-4 payments. The bigger the project is, the more often you should get paid.

Another benefit of **invoicing frequently** is that the amounts are smaller and don't need much paperwork. Some companies have strict guidelines when it comes to payments and if the amount is above a certain threshold, the payment has to go through a complicated approval process.

Make it clear that you **won't continue work** until you receive the payment after a milestone, as per the agreed schedule. You need to overcome your fear of confrontation and look out for your interests too. Have a clear deadline when the client needs to pay or you keep the deposit and aren't obligated to finish the work. Don't learn this the hard way.

Tie payments to **clear milestones**, the ones you can control and measure. "When the client is happy" isn't milestone, and neither is "when the site is live", especially if you don't control that part.

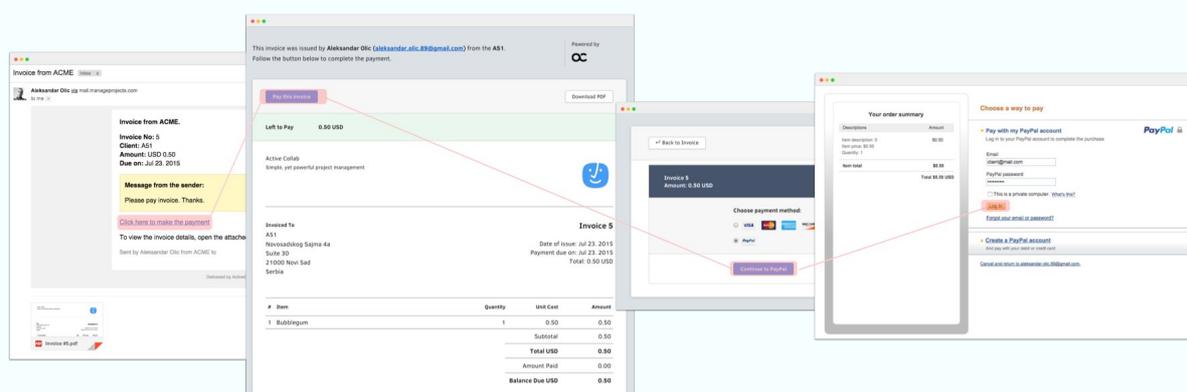
The milestones at which you should get paid are:

- before the kickoff meeting
- after the meeting,
- after doing research but before designing,
- after the comp but before coding,
- after the last presentation.

Don't start working before the client pays you a **deposit**, which is usually 50% of the total amount. The deposit can be non-refundable or you can use it as a collateral and return it if the project fails (subtracted by the amount the client owes you).

This also forces you to **work out the payment process** and the details so there's no room for stalling and excuses like "bank account blocked", "limited internet access" or some other "force majeure". This is your chance to raise red flags and run away before it's too late. Plus, once the client pays the deposit, they're more involved because they have skin in the game and are more likely to meet their appointments and deadlines.

Define when and how you'll get paid (PayPal, credit card, money transfer). You can use online payments in Active Collab so when you send the invoice, the client just has to click the link in the email, enter their PayPal or credit card, and pay with no hassle. This is good for you because it leaves little room for excuses like they didn't have the chance to go to the bank.



1. Client receives an **email**
2. Client sees the **invoice**
3. Client chooses the **payment method**
4. Client logs in and **pays**

A deposit **mitigates the risk** in case the client changes their mind but you already spent a good deal of time on research. Deposits are normal in any line of work where you create a custom product that can't be resold to anyone else and you set aside time for the client and can't take other work in the meantime. The deposit makes sure both parties have a vested interest in the project.

Don't forget to calculate **expenses** into the estimate. Expenses don't have anything to do with your work and include everything that you pay for, like stock photos, research incentives, domain registration, and subcontractors. Also, separate the expenses from your hourly rates in invoices and estimates so the client can see how much they're paying you. There's nothing worse than surprising the client with a bigger bill than they were expecting.

You should avoid paying expenses yourself and waiting for the

reimbursement. But if you do, include an **interest fee** (like banks do) because even money has costs. It's best to ask for the funds before you need them so there's no waiting. You can also instruct the client to make payments themselves but that involves more work, time, and coordination. Before spending money, always get the client's approval and then save the receipts/invoices so you can forward them to their accountant.

When it comes to **invoice due dates**, there are several options: upon receipt, NET 10, NET 15, NET 30, and NET 60. This refers to how many days can pass before the client is legally obliged to pay the invoice in full. The more time they have, the better it is for them and the worse for you. The earlier you can collect the payment, the less you risk missing a payroll or having to ask for a loan to cover overhead costs. This is where you can negotiate (and use as leverage if you have healthy cash flow).

You might want to add a **late fee** to the contract, which specifies what happens if the client misses the payment due date. The standard rate is 1.5% per month. This should be easy to add to the contract because the client doesn't want to appear flaky and of course they're going to pay on time. The clause shouldn't scare a good client, but it'll give you protection and leverage in case delays happen.

The next thing you should talk about is the **kill fee**, which you get in case the client decides to cancel the project. When you take on a client, you invest your short term future in them. In case they cancel, you don't have anything to work on next, don't have money, and can't pay your employees. This fee gives you time to find new work but it should be proportional to the project's total cost.

You can also add a **rush fee** if the client decides to move the deadline during the project. Changing dates can mess up your resource pipeline and jeopardize your other projects, so make sure this fee is high enough to make it worth your while.

Covering All Bases With a Contract

A contract won't mend a broken relationship or help you when things go wrong, but it can prevent bad things by raising red flags before it's too late. If there's a misunderstanding, you can always point to the agreement. Signing a contract is a normal business practice and if the client refuses, take it as a warning sign.

Think about having a **lawyer** on retainer, especially if you have big, high-budget projects. There are a lot of types of lawyers and you need the one who specializes in arts and intellectual property. They cost money but they can save you from financial mishaps further down the road. A lawyer will make sure you get paid for your work and not trap yourself in a bad contract. Plus, your lawyer can take a complex contract template and simplify it so it won't be so intimidating to smaller clients.

Some companies will ask you to sign **their contract** and won't sign yours. Having a lawyer go through it and handle their lawyer will ensure you get the most favorable terms possible and not sign something that can harm you. Never talk to the client's lawyer, even if it's a minor thing like changing a few words, because even the tiniest changes can have a huge impact. Instead, let lawyers handle all the minutiae among themselves. Your lawyer is on your side and will protect you from any loopholes and "alternative interpretations" of the contract.

An alternative to having a lawyer:

- Read books, like the "Business and Legal Forms for Graphic Designers", and write your own contract;
- Use templates, like the AIGA's Standard Agreement, and adjust it to your needs;
- Visit a local university's business law clinic for a free consultation.

In case the client has **their own contract**, negotiate. Just because something is in the contract, it doesn't mean you have to sign it that way. Ask them to remove things that don't suit you, add protection clauses, and explain your reasoning. Both you and the client will need to make concessions so make sure you know on which things you're willing to budge and use them as leverage.

It's best to **separate a contract from a statement of work**. A contract specifies the relationship terms between two parties. A statements of work, on the other hand, is specific to each project and covers its scope, list of deliverables, number of revisions, etc. You can sign a contract with a client once, which will cover all projects with them, and have a new statement of work each time that client hires you again.

Limit the **number of revisions** (and additional hours they'll incur) and be clear that you'll do only what's in the statement of work. You don't want to end up with a client who believes they hired you to get the project done, which to them means much more than what the statement of work covers. In case the client doesn't accept the work, you should have

the opportunity to fix or remake your work.

If there's more work, first sign a **change order** (describing the additional time and money needed) and then start work. If there are substantial changes, it's cleaner to make a new statement of work instead. A substantial change can be defined as anything that exceeds 10% of the original proposal's budget or schedule

You must add a **limitation of liability**, specifying that you aren't responsible in any way if something bad happens and can't be sued for damages caused by your work or be prosecuted for violated copyright issues which you didn't know about while creating your work. People sue people for all sorts of reason and this protects you from frivolous lawsuits and patent trolls. To assure the client of your good faith, specify that you won't include anything that might harm them in any way whatsoever to the best of your knowledge.

Sometimes **deadlines are missed**. This can happen because you miscalculated your resources or the client didn't respond on time. Define what happens if there's a delay in the schedule. If you miss a deadline, agree on what happens next, like whether you'll work harder, give a discount, or renegotiate the contract. Have a contingency plan in place and know what the repercussions are.

This goes both ways. Sometimes projects are late because the client doesn't respond on time, which hurts both the project and your business (as it disrupts your resource schedule and commitment to other projects). Negotiate for a **delay penalty** if that happens or at least find some other way to protect yourself.

If you're going to access their private accounts (servers, Google Analytics, etc.), add a clause to the contract that you won't use the information for any purpose that's against the company's goals. Assure that everything you find out during the project is **confidential** and you won't share it with anyone. Clients are reluctant with sensitive information so make them feel a bit safer.

Optionally, you can discuss **accreditation rights**, like including the work in your portfolio, submitting it for competitions, or showcasing it to other clients. Clients don't understand what these rights mean and will have reservations and fears - after all, they don't get any benefit out of this. Negotiate about including "made by" on the website because it can bring you new clients and offer the client an incentive, eg. a discount or something.

Add a **no solicitation** clause so the client can't make job offers to anyone on your team. These things do happen and you don't want to lose your best team members. Companies love to hire good workers and watching someone deliver results is better than any job interview. If they offer to hire anyone, you're at least entitled to a placement fee, if not damages.

Lastly, talk about the **IP rights** (intellectual property): who owns all authorship rights to the work and when those rights transfer. Most of the time, all rights should be transferred from you to the client upon full payment. This means a client can't use any part of your work until they settle the last invoice. You can also transfer the rights upon project cancellation but only once they settle the bill and pay the cancellation fee. Be clear that they can't use any part of your app or design until they pay you (or they risk a lawsuit).

You can also negotiate about IP rights and **structure the deal differently**. In addition to transferring all rights upon full payment, you can retain all copyright and grant a license for limited usage to the client. This way, they pay a lot less but you can re-purpose work and re-license it to other clients.

Keep a **checklist** of the things on which you're willing to negotiate. On one side keep all the things that would benefit your client, and on the other things that would benefit you. Keep the most important things at the top and weigh the tradeoffs during the negotiations.

Preparing for the Worst Case Scenario

Sometimes a client will take your work, disappear, and never pay. In that case, no contract can help you, especially if you're working for a foreign client. That's why you always need to have leverage to protect yourself - and the best leverage is **access to assets**.

Never give away **working files** until you're fully paid for a graphic or a photo. Export a PNG for feedback and send the vector or PSD only after you get paid in full. When you're collaborating and sharing work files, hide them from the client if you sense risk. Also, consider including a watermark across all the images.

If someone uses your unpaid work, **spread the word** until they pay you or take down the work. Go to their social profiles and sites where they

post ads and share your experience. Warn other people that the client is highly risky and has a habit of not paying. They'll usually offer to pay you so you stop damaging their reputation.

If you're building a website, it's a good idea to offer a client to set up **hosting** for them. If you want extra security, offer to do it for free. This way you control the log-in information and domain name which the client can't get until they pay.

When presenting the website for feedback, keep it on your **own server** and share a link. A client can see how everything works, but can't access the backend code and database. Making dynamic sites is safer because the client can't inspect the server-side source code and steal it.

Put **copyright information** inside the code (you can hide a txt file with copyright inside an inconspicuous image, like a favicon). This way, if all else fails and a client starts using your work without paying you, all you need to do is contact the hosting provider, request the website to be taken down due to copyright infringement, and show them the evidence.

If you don't have the time or resources to collect the payment, you can sell your debt to a **collection agency** for a fraction of the total due amount and let them collect the payment. This way, you'll at least get some money out of a bad deal and won't have to lose time and energy on lawsuits.

Chapter 004

Being Agile



Processes are tedious as hell, but if you don't set them up, you'll waste time, miss deadlines, go over budget, infuriate your clients, and never work again. Thankfully, most digital projects don't require elaborate processes. Simply breaking down work and tracking progress is all you need to pull off a successful project.

Modern vs Classical Project Management

Project management is an old profession. How else would Egyptians build the pyramids if there wasn't a processes to manage the work?

But project management didn't surface as a science until the 1950s. It was then that people started writing books on how to organize projects and systematically develop and apply techniques. Those books focused on really complex projects, like building a self-navigating missile system or a power plant - not something you can do in a month or even a year.

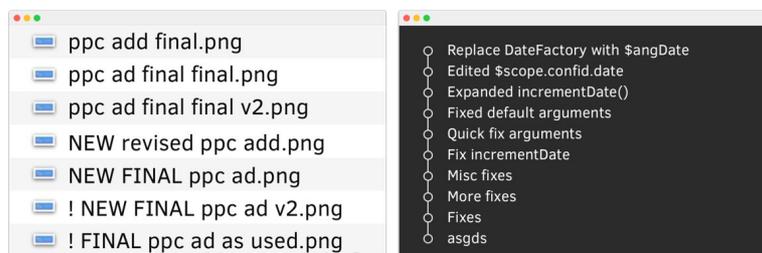
So most techniques and frameworks taught at universities and business schools are really complex (eg. PRINCE2, CPM, PERT, CPM). But they're not for you because they fail when it comes to digital projects, like building a website, making an app, or running a marketing campaign.

Why aren't they suited for you? Because digital products are malleable: they're easily changed, adjusted, and improved.

For example, once you build a bridge, that's it. You can't go back, refactor, improve incrementally, fix a bug, or pivot and change everything. You need to plan everything out upfront because the cost of a do-over can cost as much as the project itself.

Redesigning a website or rewriting an app from scratch also costs, but not as much as tearing down a bridge and rebuilding it. Plus, code and design are never truly finished. We need to update and improve websites and apps every day just to keep up with the latest technology and trends.

So we're more lax about digital. Just look at the way we name design files or our Git commit messages - even our "final version" goes through several "final" revisions.



How Agile Helps Digital Projects

To sum up: big, mission-critical projects need big, complex frameworks. Digital projects on the other hand only need flexible principles and a tool or two.

The principles that are better suited for digital projects, as summarized in the Agile Manifesto long ago (2001), are:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Pay special attention to the "over" part - it basically says:

"Don't bother planning, something's gonna change anyway. The best thing you can do is listen to your client/users, see how they use the product, and iterate until they're satisfied."

That's all there is to being agile. Experts and consultants sell trademarked methodologies they developed (like SCRUM, SAFe, RAD, RUP, etc.), riffing on the same principles. But they are overkill for smaller companies, plus their complexity goes against the core agility tenants.

All agile methodologies presuppose that projects change. Some give you general pointers on what you should aim for (Lean), some give you concrete tools (Kanban), and some are all-encompassing prescriptions for the entire workflow (Scrum).

Lean

Lean is agile taken to the extreme. It advocates delivering a working product as fast as you can and then work based on the feedback. "Fail fast" is a common advice for startups that best embody the lean spirit.

For example: if a client asks for a website, you don't do user research, plan a sitemap, or spend hours crafting perfect copy - it all takes so much time! No, you just put one call-to-action for visitors, launch it, and see how it performs.

Then, you see how people interact with the page and what they need. You can use heat maps, put a chatbox so they can ask you a question, or test it on a few users. Based on that, you can see what's missing. As you

gain insight, you start adding more elements based on the feedback.

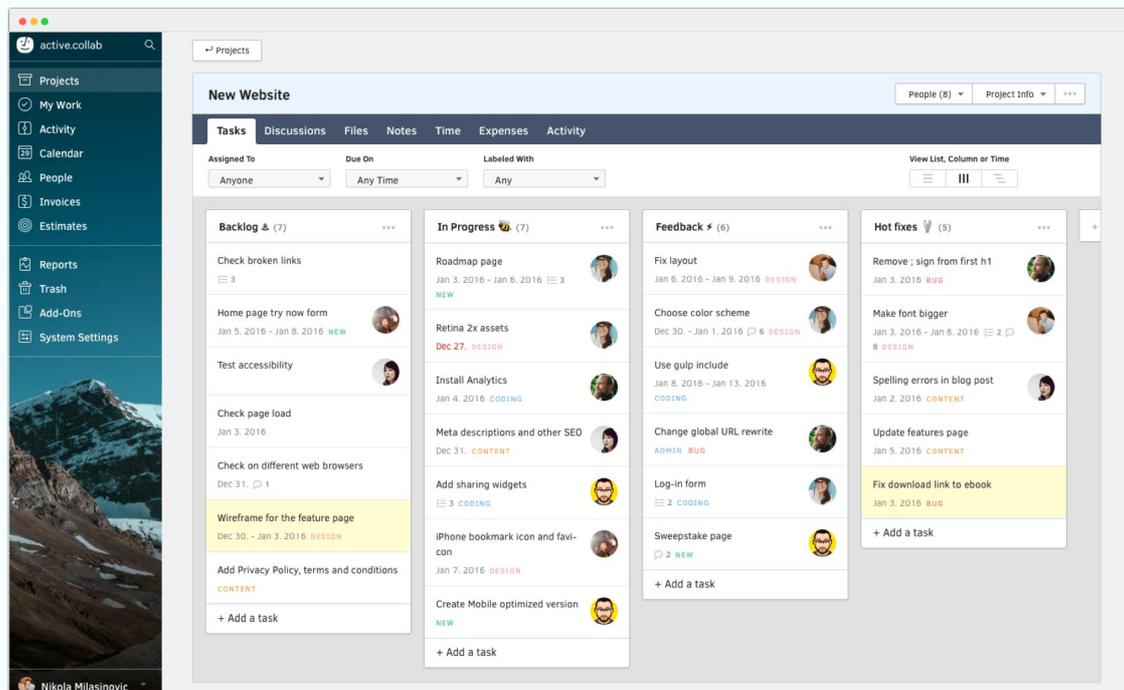
Everything in Lean revolves around the Minimum Viable Product (MVP) and eliminating waste. If some activity could be bypassed or the result could be achieved without it, it's waste: paperwork, processes, managerial overhead, switching between tasks, waiting, extra features, abandoned code, testing edge cases - who needs those?

Lean is not perfect, though. It's great when you're dealing with a lot of uncertainties and want to test an idea before you spend more money. If the client has an idea for an app but has no clue whether someone will actually use it, Lean is great. They can test the market fast and not lose money if it's a dud.

But if a client hires you to redesign their website, you don't need lean. You need slow, methodical research so you can make the most efficient website. Or if you're developing an app, it needs documentation, extensive interaction, research, and testing.

Kanban

Unlike Lean, which is more of a philosophy, Kanban is a concrete tool you can use. It's a board that helps you visualize processes that happen at the same time.



Kanban-like Column View in Active Collab

In Kanban, you write what needs to be done on a card and put it in a To-Do column on a board. When you start working, you move the card to another column. When you put all your tasks on a board, you can easily get an overview of the whole project.

Kanban is great for spotting bottlenecks on your projects because of its visual nature. You can see how many tasks you're currently working on and where you need to devote more time and resources so you can get more things done.

Most digital agencies and development teams use Kanban because it's simple, effective, and flexible.

Scrum

The most popular framework for software development is Scrum. Most organizations that use it have big projects and need the robustness, even if it comes at a price.

Scrum prescribes how everything about a project should function: how to organize a team, communicate, choose and estimate work, and track progress.

Here are the basic steps of a typical Scrum workflow:

1. Interview your client and write user stories (tasks) based on what they need.
2. Put those stories on a Kanban-like board in a backlog and assign points based on how much resources a story will need
3. Decide what stories you'll work on for the next version of the product
4. Try to finish what you've said you'd do
5. See how efficient you were on a nice burndown chart

Scrum isn't the most efficient use of your resources, but it's standardized, easy to implement, most developers are familiar with it, and management doesn't have to come up with their own solution.

It's like cooking a dish by following a recipe instead of coming up the best possible dish using all the ingredients you have in your pantry. It's not the best solution - just the most convenient.

Chapter 005

Project Manager



Project management is easier than ever before being a project manager somehow became tougher. Why is that? Software automated a lot of the work, but the way project managers use that time and a concrete value they provide is often questioned.

How Joe the Developer Perceives the Role

Many people think project managers don't do anything except boss people around and waste time. The trouble is, they're not that far off.

Ask a typical team member "What's a project manager's actual job?" and they won't be able to tell you exactly.

They know the textbook definition: a project manager manages projects, coordinates people, communicates objectives, allocates resources, monitors processes, etc. But what does it REALLY mean?

Well, in the eyes of Joe the developer, project managers don't do anything. Anything useful, that is.

In the best case scenario, a project manager doesn't waste other's time; they just sit quietly in their corner, creates to-dos, and mind their own business. Developers and designers - the people that actually create things - think they would be much more productive without a project manager.

Waste time

Communicates with clients
(which changes requirements)

Panics about budget and
deadlines

Asks for status reports

Organizes meetings

Criticizes other people's work
(but the feedback has no merit)

Useful

Creates and assigns tasks

Counter-productive

Rewards people based on
arbitrary numbers (like number
of fixed bugs)

Stands over shoulders to make
sure a person works

Presents work (but can't explain
the reason behind decisions)

Decides on technology (but
doesn't have a tech
background)

Neutral

Spends lots of time on
spreadsheets and YouTube

How an average developer and designer perceive
a project manager's job and their value

Pixar Case Study

This is not anecdotal evidence. The perception of project managers as professional time wasters is well documented. What happened in Pixar is one example.

When their team finished Toy Story, it was a stellar success. But Pixar's founders had trouble convincing project managers to stay and work on another film.

Why? The project managers described their job as a nightmare. They were constantly disrespected, marginalized, and treated like second-class citizens. When management asked artists and technical staff if that was true, they confirmed it.

The team felt **project managers impeded good work** by over controlling the process, micromanaging, and throwing sand in the gears. And this came from a team who made one of the most groundbreaking and successful animated movies of the decade.

Not only did the team deliver a product under extreme deadlines and budget, but also came up with a new workflow (feature length 3D animation was a completely new field and there were no standard workflows the team could rely on).

Of course, Pixar's management did everything they could to fix the situation. But it took them time to notice the problem of management disrespect and since then, they constantly work to make sure it doesn't happen again.

But other organizations either don't have the time to do that or don't want to admit anything's wrong. They happily trudge along and focus on the next project, while employees put up with bad management or quit once they've had enough - and you can guess what the best workers do.

Project	Budget	Time
Yet another website	\$45000	112 d



The Many Roles of a Project Manager

A project manager's role depends on the company size. For example, if you're a small development agency, hiring a **senior project manager from a big company** is a terrible choice because of the rhythm and skills mismatch.

In a startup, a project manager has to do a lot of different things, plus be the one responsible for driving work forward.

But in larger organizations, they have a more reactionary role. They wait for emails to come in, for a phone to ring, for meetings to get scheduled, for reports to get filed, and deadlines to get close before stepping in. In big companies, there's always someone who asks for their attention, so they expect the same at a smaller scale. But that doesn't happen - it's the other way around.

The things they're good at (like complex decision-making, prioritization, organizational design, process improvement, and organization-wide communication) are not needed in a startup because there are no processes to improve - only processes to create from scratch. Small companies also need domain expertise and creativity for initiating new directions.

In a smaller company, a project manager will have to **do all sorts of things** in addition to managing projects: create wireframes, send assets to developers, write bug reports, fix CSS, answer emails, organize Photoshop layers, write manuals, do SEO audits, etc. If a project manager is not a jack-of-all-trades, they won't fit in at a smaller company.

Project manager's responsibilities are many, and the benefits are few. If you want to succeed as a project manager, you have to be passionate about making others more productive, even if it comes at your expense.

Role #1: The boss of the project

The project manager and account manager are often the same person. They act as a liaison between the team and the client. Because they have more contact with the client (and are the closest to the source of power), they are susceptible to the **"boss" syndrome**.

They can feel like they can decide what and how something should be implemented without consulting with the team. Project managers with

the "boss" syndrome feel best about themselves when they figure out "how" to do the project. On the other hand, good project managers just **clearly define the "what"**, let the team decide the "how", and then manage the delivery.

For instance, a project manager shouldn't decide to use Laravel as the app's base without first consulting with the team - they should tell what the app needs to do and then let the team suggest the best framework.

Because a project manager's only goal is to create the right thing within time and budget, there's a misconception that they're more important than anyone else. But it's not true. Project managers are not the most important, just the **most responsible**.

Ancient Romans had a concept of "Primus inter pares", meaning first among equals. It's an honorary title for those who are formally equal to other members but were allowed to speak first during a debate. When applied to project management, it often means:

If a project fails, it's because of a bad project manager; if a project succeeds, it's because of a good team.

A project manager is not the "boss" of the team but the "boss" of the project. A developer may answer to a senior developer or manager, but their employment is not in the hands of the project manager. All the project manager can do is suggest who to hire or promote, but not do it single-handedly.

Role #2: Pseudo-creator

Project managers don't have a large role in the final outcome of the product, but do have some influence on the direction. A project manager that has to weave the client's wishes into the design has to ask the designer for help. So, the end product is a result of the designer's effort and the project manager's influence.

This can lead to an identity crisis. They can't directly control how a website or an app will look like, but they are held responsible for the final result. Also, when it comes to presenting the work, they are expected to know and talk about everything like they created it.

So project managers are **tempted to hijack work** and micromanage to relieve this existential pressure. But it rarely ends well. Hybrid project managers who can both manage a project/team AND design/develop

are extremely rare.

Because a project manager is a jack-of-all-trades, they can feel like they can design as good as the professional - and so they do it. But when they start being too involved with the design, other departments take a backseat. This leads to crunch time, a drop in morale as other team members start feeling disrespected, and an overall deterioration in communication.

The trick to surviving as a project manager is to keep the ego in check and not fly too close to the sun. All you can do is influence the outcome and create by proxy. It can be extremely frustrating, but it's something project managers have to learn to live with.

Role #3: Keeper of the balance

There are two types of products: perfect products and products that ship. Everyone on the team has to make tough decisions. The decision a project manager has to make is whether to ship an incomplete product or fall behind schedule. Priorities change each month and project managers have to anticipate what will be important next month and make tough calls on the spot.

While different departments fight for different goals, you have to **make sure no one wins**. Designers may want flashy animation and developers faster load times, but you can't have both.

You have to walk a fine line and not let anyone "improve" the work at the expense of the whole. The moment one side wins, the project fails. You're the only one who has the official duty to think of the big picture.

Role #4: Adopted parent

A project manager is like a parent to their team, and not in a "oh how wonderful it is to be a mother/father" way. You get all the downsides and barely any upside.

Like a parent, you always have to **be there for your children**, no matter how tired or angry you are. You have to support and protect them, teach them how to take criticism, and make unpopular decisions when needed. When a client doesn't like the work, you have to break the news to the team.

When you're working on a project, you have to be critical and point out

what's wrong, what's right, and represent the client's interest. This often means pushing your team to come up with a better solution. It's your job to inspire them to rise above mediocrity and become better.

But when you're representing your team's work to the client, it's your job to defend them - even if you disagree with some decisions. You may personally agree with your client, but your job is to represent your team. Anything less would be a betrayal. Work is a team effort - if a client starts criticizing the work, a project manager can't throw the team under the bus and side with the client.

Sometimes, your kids will barely acknowledge your presence. Or, they'll talk behind your back and say how you're no fun and simply don't "get it". But you have to swallow your pride and keep performing your parental duties the best you can.

So, like a parent, you **won't be very popular**. When you're with your team, you represent your client; when you're with your client, you represent your team. You're never one of the guys, and some people will subconsciously take it against you. And just as a parent, you have to do what's good for them - not what's popular.

With that said, you do need some popularity points in order to gain their trust and make work smoother. This means giving credit, saying "good job", complimenting on a specific detail, etc. In other words, you need to be a cheerleader. But beware not to descend into fake flattery. It's better not to say anything than say something you don't really mean or else you'll lose integrity and trust.

Role #5: Know-it-all

Team members are inherently distrustful of project managers because they don't actually create anything. But you do one thing no one has the time to do: you make sure everyone is on the same page, at all times. Sometimes, you'll spend your whole day making sure two people sitting next to each other are working on the same thing.

At any given time, you have to know who's working on what, who's on vacation, and which teams are understaffed. When the team is shorthanded, you have to find someone to fill in or do it yourself.

To fully utilize your resources, you have to **know everything without trying**. If someone has a question, you should be the first person they think of. Getting this kind of reputation is difficult, especially if you don't

have a technical background. But you don't want your developer to waste time trying to figure out how to restore their Git commit if you have someone who's a guru on the subject.

A developer shouldn't trouble themselves by getting to know every other developer's core competencies. But a project manager should so they can immediately hook up the person who needs help with the one who can provide that help (or at least know how to find that person). Gathering knowledge takes time, but the more things you know, the more stuff people will tell you.

In addition to networking and keeping an open ear, you should **systematically centralize information**. This includes keeping a contacts list, updating the company wiki, and using project management software.

Role #6: Productivity freak

Big organizations move slowly because projects are so easy to delay. All it takes is one person: a developer may be stuck waiting for requirement clarification or a manager may need to be consulted before approving an expense. All these seemingly minor hesitations slowly delay projects. One hour here, another hour there, and your project will end up slipping a month behind schedule - and you'll wonder how that happened.

The only way to stop those little, harmless hesitations is to constantly work on **removing roadblocks**. So if anyone's stuck on anything for any reason, a project manager should work on resolving the issue. The less time a task spends in the "waiting for approval" phase, the less likely you'll slip behind schedule.

A project manager's job is to **take non-work off their team's back** so they can spend more time designing and developing (i.e. doing what they're hired for). You don't make your team more productive by organizing meetings and asking for reports - this makes your job easier, which is a completely different thing.

Protect your team from outside work and they'll naturally become more efficient, without attending a motivation seminar. Do everything in your power to protect your people from meetings, extra work, or "quick consultations" with other teams.

Also, while designers and developers are expected to focus solely on tasks within their field, you're the one who should do odd jobs, like

booking meeting rooms, sending reminders, taking notes, entering tasks, and making sure the videoconferencing system gets fixed.

You should also work on optimizing processes. In poor organizations, people spend much of their time fighting organizational boundaries and broken processes, navigating through red tape, and dodging conflicts. Sometimes, they're not even clear what their job is, if they're doing it right, or why they do it in the first place. And if they work up the courage and speak up, management denies any problems exists, defend the status quo, ignore the problem, and mark the person as having an "attitude".

As a project manager, you should strive to make **each day more productive** than the last. The more you work, the more you can learn and optimize the process. This takes time and creative thinking. You can't just decide to improve things - it should come naturally, out of a desire and a need to be more efficient. As you work, try out new things and experiment. Most experiments will fail, but it's the only way to become better.

This sometimes means holding a post-mortem meeting, which, unlike other meetings, is an investment. Meeting after a project is finished will tell you what common problems are and your team will have a chance to suggest how to fix them. Then you can sum up your experience and have a chance to analyze your work, without looming deadlines breathing behind your neck.

Make sure these meetings have an agenda, start on time, and don't last too long. Also, come prepared, take notes, and actually follow-up on any action item. This sounds easy, but in practice, most project managers don't have the time and focus to actually do it. If you can't set some time aside to improve the process, all the more reason you need to. It's like that zen proverb:

"You should meditate 20 minutes a day - unless you're too busy; then you should meditate for an hour."

Re-Earning Cred Each Day

Team resistance is a burden you'll have to deal with every day. But always keep in mind that a project manager is equally important as the other roles that create tangible value.

Without project managers, the whole project would spiral out of control, teams would fight for resources, and no one would get paid because they wouldn't have who to work for.

Good project managers:

- Care more about a project than a particular department's goals;
- Praise and give credit when it's due;
- Protect team from extra work;
- Work on making others more productive;
- Are the first ones others turn to for information;
- Represent the client in front of the team and vice versa;
- Don't dictate the "how" but manage the delivery of the "what".

A project manager's job is not measured by how much they work but how much their team accomplishes. If all you do is twiddle your thumbs and make coffee, yet every team you manage outperforms other teams, you're doing a good job.

Everyone managing developers and designers should read **Peopleware**. Do a favor to yourself and your team and read it right now. It'll be the best time investment you ever made.

Chapter 006

Project Planning



It's exciting to start working on a new project. The team is tempted to just dive in and start coding, designing, and creating. But as a project manager, it's your solemn duty to make a game plan first.

Where to Begin Organizing 1,000 Tasks?

Planning can be tedious and unglamorous - but if you skip it, you'll end up wasting a lot of time on trial and error. Even if a plan proves useless, the act of planning is indispensable because it clarifies the goal and serves as a starting point.

"Give me six hours to chop down a tree and I will spend the first four sharpening the axe." - Abraham Lincoln

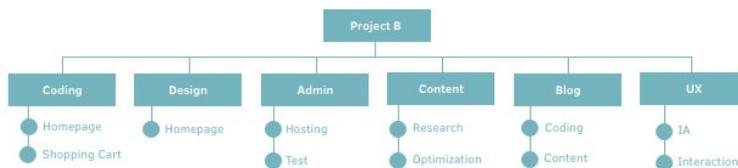
Before you start planning to-dos, time, and people, you need a consistent way to organize your work. Projects have a lot of tasks and you have several options by which you can group them:

- **Progress:** Backlog, Next, In-Progress, Review, Done
- **Type of Work:** Design, Marketing, Development, QA
- **Time:** Today, This Week, Next Week, May 2016, To Be Determined
- **Product:** Website, Mobile App, Desktop App, Blog
- **Feature:** Homepage, Contact Us, Header, Footer, Shopping Cart
- **Priority:** Low, Routine, Normal, Priority, Critical, DEFCON 1
- **Complexity:** Quick Fix, Minor Improvement, Big Feature, Security, Tech Debt
- **Hybrid:** a mix of the above



The way you group tasks depends on the project. For **continuous projects** that have no beginning and no end (like developing your own product), it's best to organize tasks by progress and projects by product. **Small-scale projects** (like website development) can be grouped by feature, while **bigger projects** (like a complete rebranding) are commonly grouped by the type of work.

There are two approaches on how granular you want to go when grouping tasks: you can use simple segmentation and complex segmentation.



Complex Segmentation

You segment tasks by many criteria and have a lot of task lists. This is good if you have big projects which you don't want to split but need to make task management easier.



Simple Segmentation

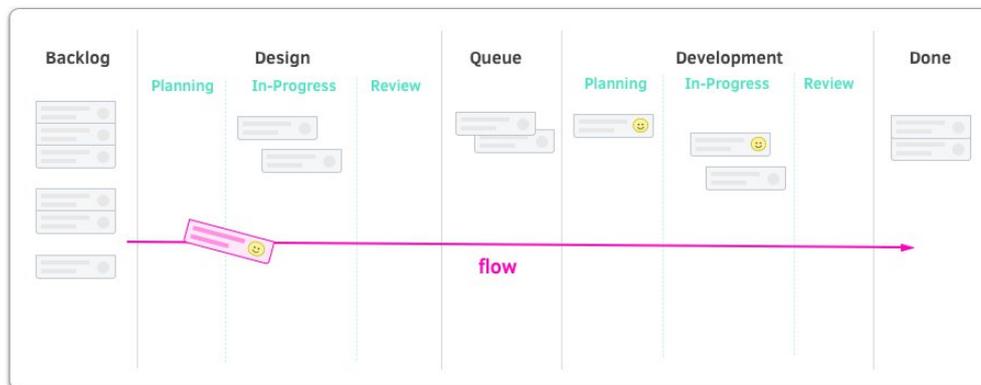
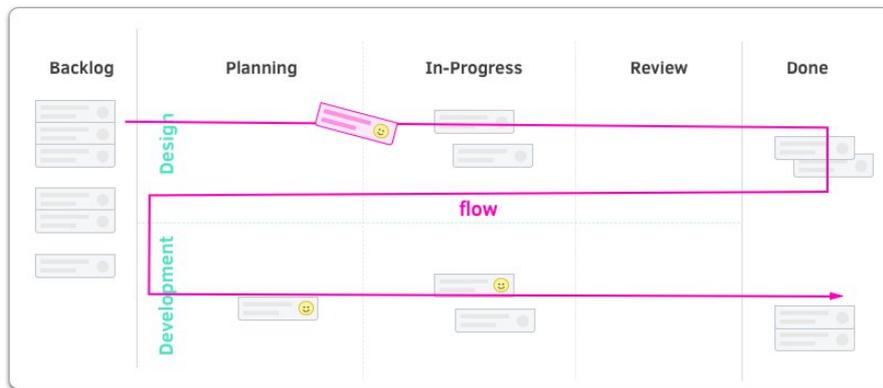
You segment tasks by just a few criteria and have a few task lists. This is good for smaller projects where you don't want to think where to put an ambiguous task that can't be categorized.

When in doubt, go with the **most flexible approach** and see how it works. Over time, you'll know which tasks are the most frequent and you'll know when to use which principle. For example, if you complete tasks as soon as you start working on them (there's are no complicated processes, just simple to-dos), grouping them by progress is useless; what you then need is to group by time or priority.

If your workflow is good, but you occasionally come across a task that demands it's own category, just add a column where you'll put all the odd jobs. You can also use the hybrid approach: if you categorize by complexity, feel free to add the In-Progress column. But don't overdo it. It can be confusing if you use more than one principle: for instance, does a new bug go into Backlog, Hot Fix, Development, or Feature A?

The problem of segmenting tasks more granularity is addressed by using **swimlanes**, but they bring more confusion than clarity because they make the flow non-sequential. For example, you need to segment both design and development phases into Planning, In-Progress, and Review. But with swimlanes, it's hard to track work items and a developer doesn't know what tasks he can pull next.

A better solution is to break a task list into several task lists so the flow is straight and manageable.

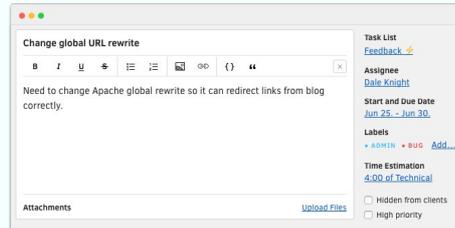


When deciding on how you want to organize tasks, keep in mind all the tools you have at your disposal and how you can retrieve information.

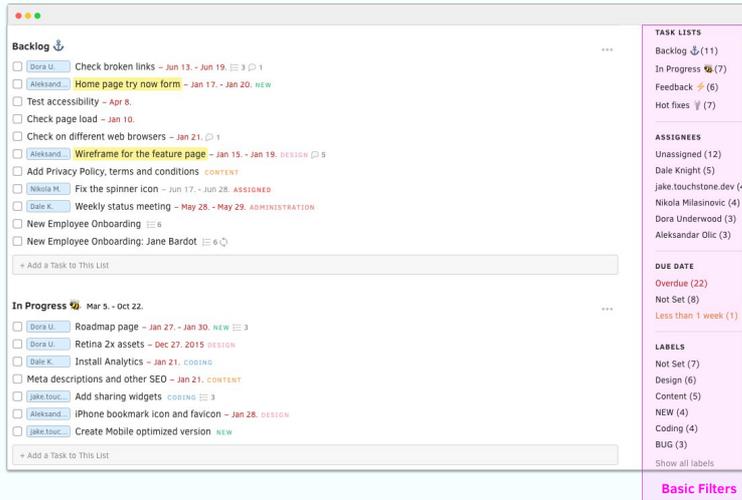
If you need more to segment tasks using more dimensions (eg. want to mark task as a bug that's in-progress with a high complexity), use labels in addition to task lists. Labels are like tags and can be anything you want; one task can contain multiple labels. You can communicate a task's complexity, type of work, priority, or anything else by using them. This way, task lists don't have to do so much work.

For example, if you organize task lists by feature, you can use labels to indicate progress. When you start working on a task, just switch the label to In-Progress; when you're done, switch the label to Review.

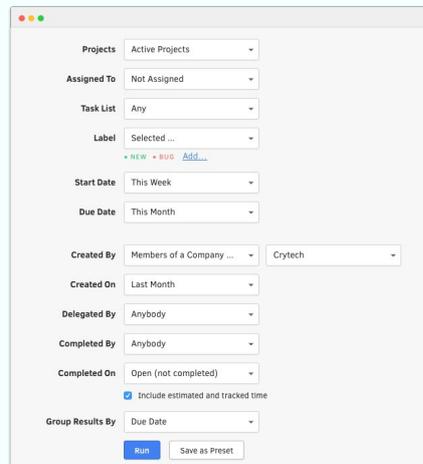
To recap: use task lists to group tasks by one principle (usually stage of progress or type of work), and add labels if you need more information. You can always filter tasks by assignee and date automatically when needed.



A task in Active Collab allows you to fill it with as much details as you need.



Once you create a task, you can then use those details (like assignee or due date) to filter what tasks you see on a project.



Advanced Filtering using the Tasks Report

If you need more control, there is a Tasks Report for project managers where you can get all the tasks you need across projects.

In Active Collab, a task has several filterable fields. It practically means you don't have to organize task lists by time or assignee because you can always get those dimensions automatically.

What to do with completed tasks?

When deciding on columns, keep in mind that Active Collab automatically gives you a Done column for each project, only it's not in the form of a column.

Every time you complete a task, it disappears from the main view. But, you can always access it by looking at your completed tasks. So in a way, it doesn't matter if the task is in the Done column or hidden somewhere out of sight - you still have a place where it ends its life cycle.

Tasks are on a separate screen because if a task is completed, you don't need to see it every time you open a project.

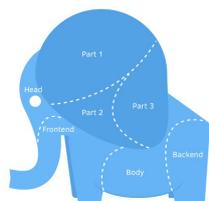
But if you have the kind of workflow where you want to see completed tasks (eg. a client wants to see what's done), then don't complete them.

Instead, create a Done task list and move the tasks there. Or, if you want to keep a task in its task list, put a checkmark (✓) at the start of the task's title to indicate it's completed. This way, you can organize tasks by type of work (as opposed to the stage of progress), and keep them in view but visually separated from the rest.

Keeping completed tasks on display is a great motivator for your team. It's easy to forget how much you've accomplished and get lost in day-to-day activities. Keeping finished tasks in sight may clutter the view, but it's great visual reminder of the progress you've made. It also helps you reassure your clients and make them appreciate all the work you've done.

Work Breakdown: Planning the "What"

How do you eat an elephant? One bite at a time. By breaking a project into several tasks, it suddenly becomes a manageable and achievable undertaking.



You first **break work** into several discrete units, then you break those units further, and so on until your tasks become doable. For example, you start breaking down "Web redesign" into several big tasks and those into subtasks, etc.

Research	Design	Development	Review
Survey users <ul style="list-style-type: none"> - create questionnaire - find users - analyze results 	Information architecture <ul style="list-style-type: none"> - Create interaction map - Create sitemap - Plan content - Copywriting - Discussion and review 	Front-end <ul style="list-style-type: none"> - Set layout - Write HTML - Write CSS - Responsiveness - Header - Footer 	Test <ul style="list-style-type: none"> - A/B tests - Usability - Test back-end
Competition research <ul style="list-style-type: none"> - make the list - analyze - document key points 	Visual design <ul style="list-style-type: none"> - Choose color palette - Decide on typography - Create comps - Asset production 	Back-end <ul style="list-style-type: none"> - Set integrations - Create admin area - Log-in - Cookies - Security 	Present to the client <ul style="list-style-type: none"> - Arrange a meeting - Prepare documentation - Present - Gather feedback - Evaluate next steps
Gather requirements <ul style="list-style-type: none"> - interview marketing department - interview sales department - get content 			

This doesn't have to be an exhaustive list of all work - it should be a comprehensive list on which you can start working right away and add other tasks as you go along. At this point, it's enough just to specify what needs to be done - you'll think about how and when after that.

Project Timeline: Planning the "When"

Some projects have fixed end dates, some have deadlines for milestones only, some have soft deadlines, and some don't have deadlines at all.

When you have an end date, it's best to plan your timeline backwards by adding tasks which are closest to the project deadline first and then gradually moving towards the start. This is **deadline-driven planning** and it's most often used in event planning where you have a date for an event that can't be moved so you plan everything before it.

On the other hand, if you're not constrained by a deadline and need to focus on the quality of the work, use **quality-driven planning**. Software companies who continuously work on a product use this approach because it's more important to get the feature right than to meet an arbitrarily set deadline.

Quality-Driven Planning

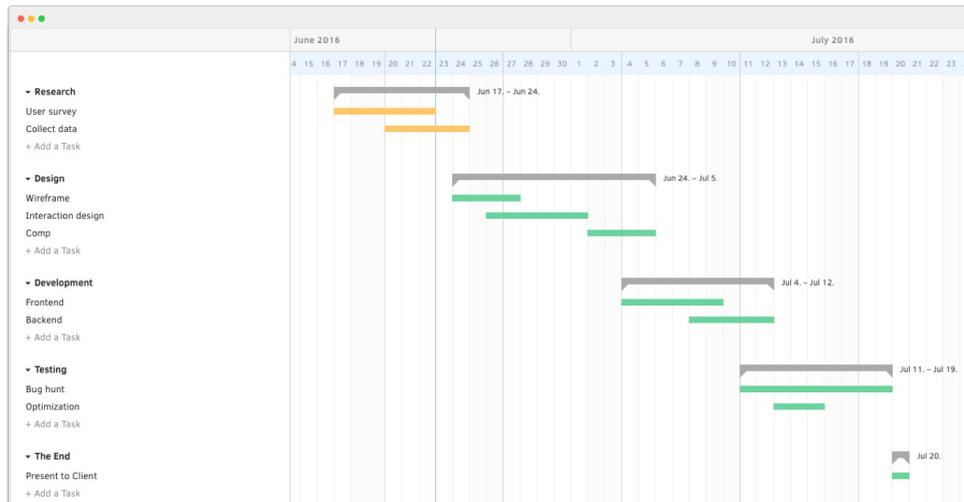
Step 1
Estimate how long you need for the first activity.

Step 2
Move on to the next activity.

Step 3
Give each task as much time as you need to finish it properly.

Step 4
If you need more time for some task, shift other tasks as well so you don't risk crunch time.

Step 5
The end date of the last task is your deadline.



Step 7
Adjust dates until you come up with a realistic timeline within time constraints.

Step 6
If you end up having to start "yesterday", decide what activities can be finished more quickly.

Step 5
Do so until you come to project's first activity.

Step 4
See what goes before that and repeat.

Step 3
Estimate how much time you need for that and set start and end date for that task.

Step 2
See what's the immediate previous thing before the deadline.

Step 1
Set the project deadline.

Deadline-Driven Planning

When estimating, you can first approximate **start and end dates** for task lists. For instance, estimate how long Design should last and, based on that, determine how much time you have for individual tasks within the list. Be careful, though: this estimate can delay your whole project if you're too optimistic with estimates.

To lessen your unwarranted optimism, add an estimate for each task BEFORE you start working with dates. If a task is estimated for 6 hours and you see the assignee has other tasks that same week, you'll be less tempted to set the duration of one day.

Incorrect estimates make project planning more difficult, but they are harmless when compared to the biggest enemy here - an **indecisive client**. Clients often don't know the exact features they need in the early project phases, but usually find out during the project. This can derail your whole plan. But it's not a bad thing. It'll mess up your perfect plan, yes - but it also means your project scope just got bigger, which means more money and a chance to revise the project scope agreement).

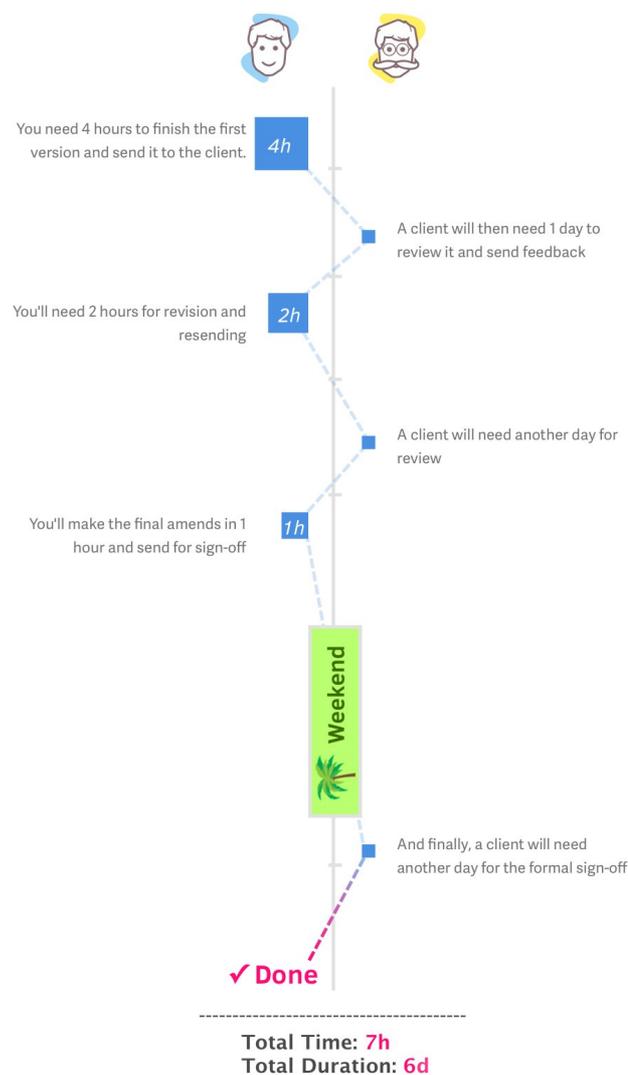
That's why you need to include **triage periods** in the plan. In every project, you're going to have periods of rough patches and smooth patches. Usually, the rough patches correspond to important milestones when the team has to decide how to move forward with any new information at hand. These periods require extra time for communication, reflection, and getting on the same page. It's at times like these that the team might get a feeling that a project isn't going

great. It's normal. You just need to anticipate those moments and add buffer zones in your timeline.

Add one triage padding after the research, because after research, the team knows what they have to work with so they can plan better. Add another triage padding once the design moves from sketch to mockup, because once the design becomes "real", clients and others on the team experience all kinds of reactions, and that can change the direction of the project.

Don't forget to account for **time lag** when you plan a task's end date.

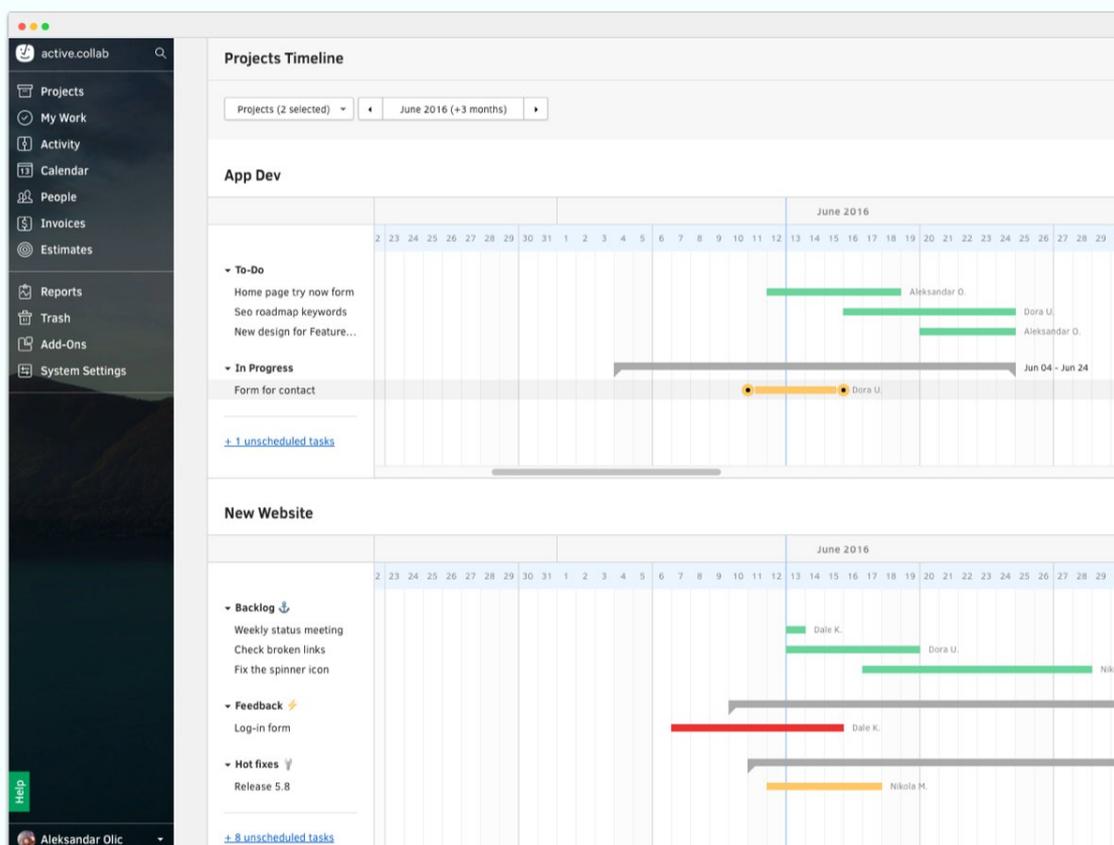
A 7-hour task can actually take up 6 work days to complete (or 2 weeks) - operating under an assumption that the client does their job on time. The point is, a task might need 7 hours of actual work, but that work can be distributed over several days due to time lag.



It's important to communicate the project plan to your team and the client. Make the project plan easy for others to see and access so they can better understand how they fit in the grand scheme of things. This doesn't mean you need heavy documentation - just the opposite. Present the timeline to your team, ask what they think, and improve

You have to monitor how a project interacts with your other projects. If your projects share resources (like time and people), they can overlap so your team members end up with too much on their plate, which will disrupt the whole project plan.

To see who works on what and when, run a Projects Timeline report in Active Collab before you start planning a new project. It'll give you an overview of all your projects and their tasks so you can see which dates are the busiest and should be avoided. You can even plan and tweak projects directly in the report.



Task Assignment: Planning the "Who"

Each team member needs to know what to work on. They expect a project manager will assign tasks for each day and keep them busy. This is a project manager's job because they have the complete overview of the project (not to mention the responsibility for the time and budget).

Most tasks aren't discrete work units that require a team of one. Most of project work is collaborative, and there are several types of collaboration.

Some people work on the task, some need to offer their advice, and others merely have to be informed of the progress because it impacts their own work. The best approach is to use the Responsibility Assignment Matrix (RACI).

The RACI matrix defines four roles a person can have on a task:

- **Responsible:** owns the task, works on it, and their duty is to see it gets completed
- **Accountable:** must approve the work and sign-off before it can be completed
- **Consulted:** has the necessary information without which a task can't be completed
- **Informed:** needs to know the final outcome, but doesn't need to be consulted

RACI matrix used to be done in a spreadsheet, next to the work breakdown structure, but Active Collab is better suited because of its interactive nature:

The screenshot shows a task page for 'BikeShop_Website - Task #12'. The task description is 'General design' with a subtask 'We need to come up with a general feel of the website. Attach all link and images we can use for inspiration and we'll decide on the direction.' The 'Discussion' section contains three comments: Alex R. (1 month ago) asking about the main CTA, Yvonne G. (2 months ago) replying 'Sounds good - I'll have the initial design sketches ready by Tuesday.', and Ethel P. (2 months ago) providing a detailed response about design choices. The right sidebar shows the 'Task List' with 'Assignee: Yvonne Gardner', 'Due On: Mar 30', and a list of 'Subscribers' including Yvonne Gardner, Alex Ryerson, Ethel Parker, Dale Knight, and Jake Touchstone. Annotations on the left and right explain the RACI roles: 'Accountable' (client mention), 'Consulted' (input from others), 'Responsible' (assignee), and 'Informed' (subscribers).

Accountable
When you need approval, mention the client in a comment (using @ClientName) so they get the notification and an email.

Consulted
If a task needs input from several people, create subtasks and assign them to the people who need to do a part of work. You can also use @mention when you need someone's input right away

Responsible
When you have all the tasks, select an assignee. They're the ones you'll ask if why the task wasn't finished. There should only be one because if someone doesn't feel ultimately responsible for seeing the work through, the work may not happen at all.

Informed
Subscribe everyone who needs to be informed of the progress

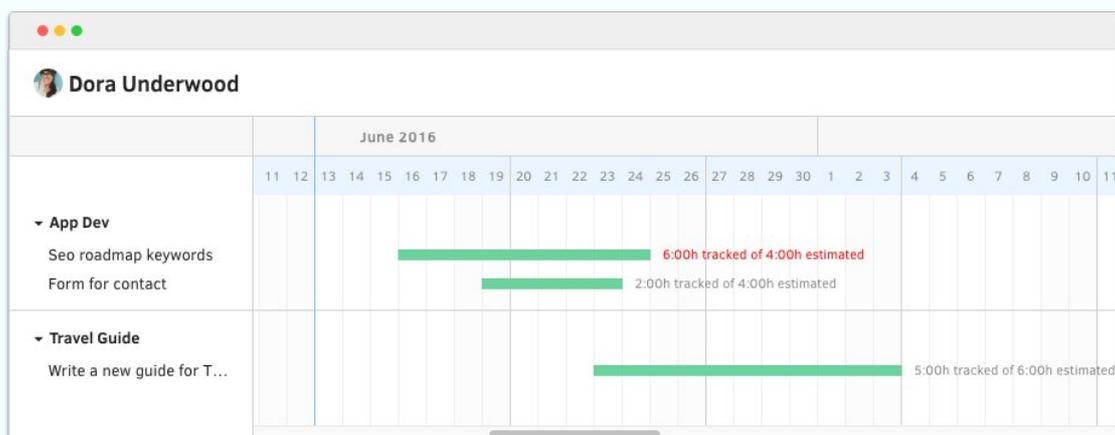
So for example, a project manager holds a designer responsible for finishing a wireframe. A client is accountable because they must approve work before a designer can complete the task. Marketing should be consulted because they know what the main call to action should be and what content they have; the information architect needs to be consulted too because he created the sitemap. A front-end developer only needs to be informed of the final result so they can start coding the layout.

If your team works on multiple projects simultaneously, you'll have to make sure their responsibilities on other projects don't overlap, so they don't get overbooked.

The Team Timeline report lets you see a timeline all the responsibilities a team member has across all their projects.

With it, you can see what a person has to work on each day so you can better plan their time. For example, while planning a project, you can quickly see all the tasks someone has on other projects. You can then tweak tasks (on the existing project or the new one) and rearrange them so they don't get overbooked.

With the Team Timeline, you can create more realistic project plans because you'll know how busy each team member is. You can even determine when to start a new project based on this information (or if you have the resources to start it at all).



If a team member finishes all their assignments and has leeway, it's a good idea to always have tasks in the backlog: a task list from which a person can pull work once they finish all their assignments. In software development, there's always a curated backlog full of bugs (arranged by priority), so when a developer finishes working on a feature, they can work on bugs while waiting for the review.

Automate Planning

Once you've planned one dev project, you've planned them all. Save time by using an existing plan as a basis for future projects. You do this by creating a project template, filling it with common tasks (and all the details), files, team members, discussions, and notes.

So when you get a contract for a new project, you just create a new project from the template, update it with specific details, and you're ready to go.

Recurring tasks are another time saver. They are common tasks that you create often (eg. every day, week, or month. They're perfect for services you provide each week/month. You define them once, set how often they should be created, and don't worry about creating them manually later.

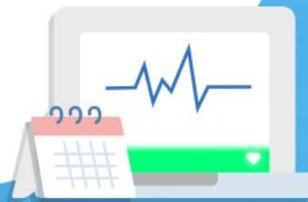
For example, if you want your assistant to send direct mail campaigns on the first of every month, you create a recurring task and it'll pop up on that day. Your assistant will see it under on their to-do for that day and send the campaign.

Here are some tasks you can put on auto-pilot:

- Daily standup meeting
- Monthly progress meeting
- Client meeting
- Review contracts with lawyer
- Check team performance every week
- Create a project report every Monday
- Perform backup each week
- Check unpaid invoices
- Pay office bills
- Invoice work at the end of the month
- Contact an old client to keep in touch

Chapter 007

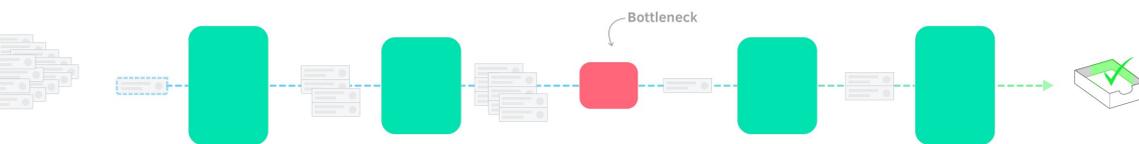
Project Monitoring



Industry stats say that less than 1/3 of all projects are completed on time and budget. It doesn't have to be this way. To prevent a project from spiraling out of control, a project manager has to constantly monitor progress, watch out for potential problems, and take corrective measures.

Watching Out for Bottlenecks

The biggest hurdle to how much you can accomplish are the bottlenecks: one or more resources which limit the output of the entire system. For example, if your marketing team can set up a thousand ad campaigns in one day but your creative team can create only a few landing pages, the number of campaigns you can have is limited by the output of your creative team.



If you aren't aware that the creative team is a bottleneck, then a backlog of work will begin to pile up in front of them and you'll have a conflict between departments. Also, the quality of work will suffer because the creatives will start cutting corners so they can clear the backlog; this will, in turn, destroy the conversion rate and render your whole work meaningless.

One way to deal with the problem is to redeploy your existing resources when a backlog becomes too big. For example, the marketing department can jump in and help the creative, or a developer may jump in if there are too many tasks in the QA backlog.

How do you know when someone needs help? By purposely **limiting work in progress (WIP)**. You can prevent work pileups (which results in bottlenecks) by limiting the number of tasks people can work on at one time. It sounds counterintuitive because, after all, you want everyone to constantly work on something and operate under full capacity.

But a large amount of work leads to problems. Developers tend to multitask and work on bugs while they're waiting for QA to complete. This ultimately results in more work for the QA down the road. Work will keep piling up until they end up with an enormous pile of partially-completed work that can't be shipped.

In competitive rowing, a key position is the coxswain — the person in the back of the boat yelling “row, row, row”. He coordinates the activities of all the rowers so they are rowing at the same speed. If one rower

outperforms everyone else, boat gets out of kilter and slows down - meaning, extra power and speed can actually slow the boat down. Same thing on projects. Making one department extra efficient can bury other departments in excess inventory and slow them down.

It's important to understand that limiting WIP doesn't directly result in more output (because you're still constrained by the bottleneck). Instead, what you get is:

- Better lead time (the time between the task's initiation and completion)
- Constructive frustration which drives improvement. If people complain about not having work, you have a clear signal you need to expedite the process and a justification to ask for more resources.
- Better quality.

Longer lead time leads to poorer quality, as evidenced by the Little's law. In fact, 6 times increase in average lead time results in a greater than 30-fold increase in initial defects. This longer average lead times results directly from greater amounts of work-in-progress. If you want to improve quality, you first need to reduce and then limit WIP.

With Active Collab, you can monitor how many tasks are in the In-Progress task list and act accordingly. You can also take a number of assignments a person has as a signal of a bottleneck. When you run the Workload report, it'll show you how many tasks each person is assigned to. If someone has way more than anyone else, assign some tasks or investigate the matter.

The screenshot displays a 'Workload' report interface. At the top, it shows '5 members' and 'All Projects'. Below this, tasks are organized into columns for each team member:

Aleksandar Olic	Dale Knight	Dora Underwood
Home page try now form	Remove ; sign from first h1	Prepare contract
Wireframe for the feature page	Install Analytics	Roadmap page
3. Complete tasks	Change global URL rewrite	Check broken links
iPhone bookmark icon and favicon	Log-in form	Retina 2x assets
Drag people onto a card to indicate that they're responsible for it.	Weekly status meeting	Write a new guide for Travel
Collect images		Seo roadmap keywords
Home page try now form		Form for contact
New design for Features page		

jake.touchstone.dev	Nikola Milasinovic
Use gulp include	Make font bigger
Create Mobile optimized version	Fix layout
Add sharing widgets	Fix the spinner icon
Sweepstake page	Release 5.8

Another way to deal with bottlenecks is to **design the team** so there are no bottlenecks to begin with. Do this by calculating how much features you need, how much features each role can process, and then hire accordingly.

Let's say you need designers, developers, and testers on a project. An average designer can design 5 features per week, a developer can develop 3, and a tester can test 10. How many people do you need?

After a bit of math, you find out you need: 6 designers, 15 developers, and 3 testers.

This is a pre-optimized system where your only job is to monitor if the people are accomplishing as much as planned. But digital projects can be tough to estimate. Even a concept of what exactly constitutes a feature can be ambiguous.

Pre-optimization at least gives you a starting point. Then, as you work, you'll see which team over- or underperforms, identify where the bottlenecks are, and add staff where needed.

When it comes to software development, beware of the mythical man-month: Adding manpower to a late software project delays it. This means you should think twice before you hire more developers in order to meet the deadline.

At other times, you simply don't have the possibility to throw more resources at a project. This happens when:

- A resource is scarce (like a developer who knows how to navigate through a client's legacy system),
- or impossible to increase (like the time you need to get approved by the App store).

In that case, you need to **design the system** around bottlenecks. For example, if each task has to be run by a senior developer, but you only have one in the team, this makes him the bottleneck. As such, you want to make sure he never runs out of work. After all, an idle junior developer costs less than an idle senior one - especially when they're the bottleneck in your team.

When working on a bottleneck, keep in mind that work items vary in size. This means a bottleneck can run out of work simply because there are a few large tasks in the pipeline that are coming too slow, and when they land, the pipeline will get clogged again.

Controlling Quality

When faced with a lot of work, people tend to cut corners intentionally. But sometimes, the quality suffers without anyone noticing it.

For example, a developer may see 100+ console errors when they start working on an app and may subconsciously think:

"If no one cares about compiler warnings, it means sloppy work is tolerated and I can cut corners in order to get home by 5pm".

When faced with a decision to make something good or to meet the deadline as long as it compiles, they'll always choose the latter because that's all management cares about.

This hurts long-term projects the most. **Overlooking one issue** leads to overlooking more issues, which snowballs into a tech-debt ridden project that, the longer you work, the more it costs.

You probably heard about the mythical 10x developer who can accomplish 10 times as much as work as an average one. They do exist, but the chances of you having one on your team are next to nothing. If you have a someone who accomplishes more in terms of quantity, better investigate. Chances are, you have a tech-debt loan shark.

How to spot tech-debt loan sharks?

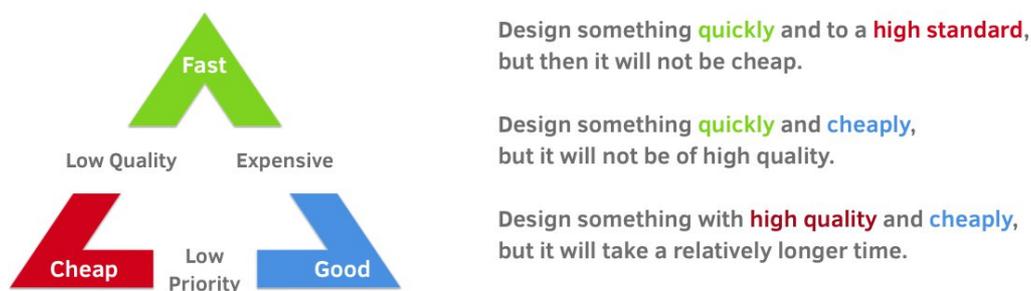
- They prefer to copy/paste than to spend time on DRY;
- As long as it works, how or why is irrelevant;
- They care more about finishing a task today than to make next week's task easier to finish;
- They don't care how their work affects the overall design;
- They don't write tests;
- They complain when there's no documentation but don't bother producing it.

It's **easy to lose focus on quality**. Project managers spend most of their time thinking about the budget and deadlines because they're the most visible and easy to quantify, analyze, and communicate. What gets measured gets done. Non-technical people don't get upset over

spaghetti code like developers do - but missing a deadline or going over budget is another matter.

When quality takes a back seat, it hurts your position on a competitive market. In the past, building something took a lot of work so selling was easy. Today, it's the opposite. The market is oversaturated with low-quality work and the trend will become even steeper.

To make sure the quality is good, involve a client from start to finish. They have the largest stake and should value the long term benefit of getting a good product over meeting a deadline. But if they care more about the bottom line (and pay accordingly), you don't have any obligations to worry about quality. You can't have something fast, good, and cheap at the same time - you can only pick two.



On the other hand, over-focusing on quality can hurt you. Always do a quick cost/benefit analysis before you ask "can we make this better/faster/stronger". If it takes a lot of work to make something perfect, but it doesn't make an impact, move on. Your team can only do a finite number of things, so focus on those that matter the most.

Think of finishing a project like sweeping a floor. You clean most of the dirt fairly quickly, but the last few specks of dust are impossible to remove. So you keep cleaning until you're left with an amount of grime you can live with, put the broom away, and get on with other chores.

Working With People

A project manager's job is to monitor the plan and tell others what to work on and when. If the team risks falling behind schedule, the manager steps in to see what can be done.

A lot of your time as a project manager comes down to managing people. But because you don't have the authority of a CEO, you have to lead, manage, and inspire people using other methods.

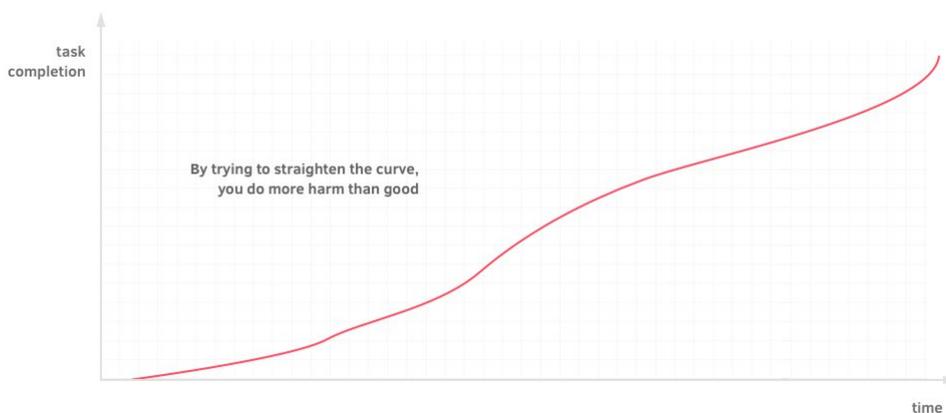
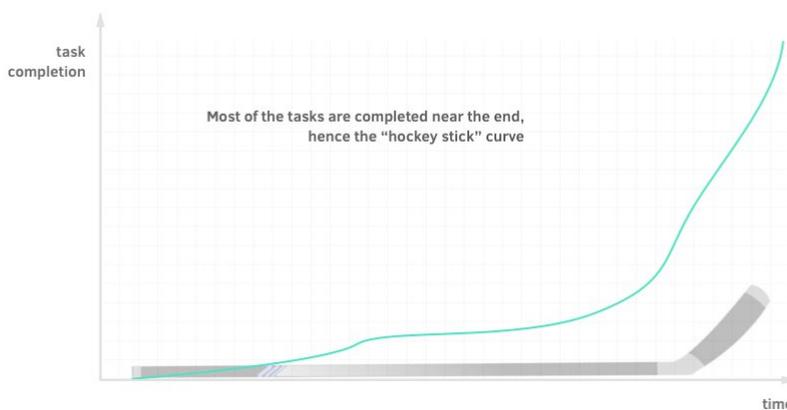
Bad project managers often go through this cycle:

1. Ask everyone for status updates.
2. Be ignored.
3. Try to attach yourself to the next project so you don't get fired.

Or even worse, they ask a developer for a date to implement something, tell the customer it'll be delivered faster, and complain when a developer misses the deadline. Or they may not even consult with the developer, promise the client something they can't deliver, sign the contract, celebrate with a big event, and leave the developer with figuring out how to make the impossible happen.

Another big mistake bad project managers make is **measuring success by using the wrong metrics**.

For example, you may notice you complete most of the work as you come closer to the end. This makes your progress chart look like a hockey stick. You think you need a "healthier" progress bar and nicer distribution. So, in order to straighten the hockey stick, you issue an incentive and give a bonus if a task gets closed earlier .



Then people start to pester others to complete the task and the overall productivity plummets because everyone is after their own gain. In the end, your chart becomes a bit more linear and “healthier” - but at what cost?

Same story if you measure work by meeting deadlines and the number of released features. You can ship the product on time, with no bugs, and with all the features - but the product will be mediocre because none of the features are well thought-out.

In both cases, you got what you asked for, but not what you needed. Managers make a common mistake of measuring the productivity by counting irrelevant numbers and creating wrong incentives. Then people find a way to hack the system and your incentives end up backfiring.

For example, you can measure the productivity of your tester by the number of bugs they report. They'll be incentivized to discover as many bugs as they can, no matter how small they are; then they'll write a bug report, argue with a developer whether something is really a bug, and waste time.

Measuring a number of completed tasks is also a bad idea. Such a system doesn't take into account anyone who helps others become more productive and as a result, everyone suffers. The true 10x developer is a person who makes others more productive.

It's been proven that **incentives don't help** but companies still use them. They're fine in sales, where the result directly depends on one person; but when your success depends on a team, the result is a collective effort where you can't easily measure contribution and rewarding individuals is never fair.

Your best option is to drop incentives altogether, focus on keeping good levels of enthusiasm throughout the project, and reward team effort.

Enthusiasm on a project can vary widely, depending on the stage. Everyone is super excited at first, but once you hit the first roadblock, it'll take some of the wind out of your sails. You'll push through, but could then reach a roadblock so big that you'll start to question whether you should continue at all.

Your job is to motivate people to keep working when the team starts to have second thoughts. Each time morale drops, keep in mind the

enthusiasm curve. Once you understand how your team is going to feel, you can prepare accordingly.



Also, keep in mind the 90/90 rule:

The first 90 percent of the code accounts for the first 90 percent of the development time. The remaining 10 percent of the code accounts for the other 90 percent of the development time.

Staying on Top of Projects

The three most important things you need to know are are:

- Who's stuck?
- Who's too busy?
- Who's idle?

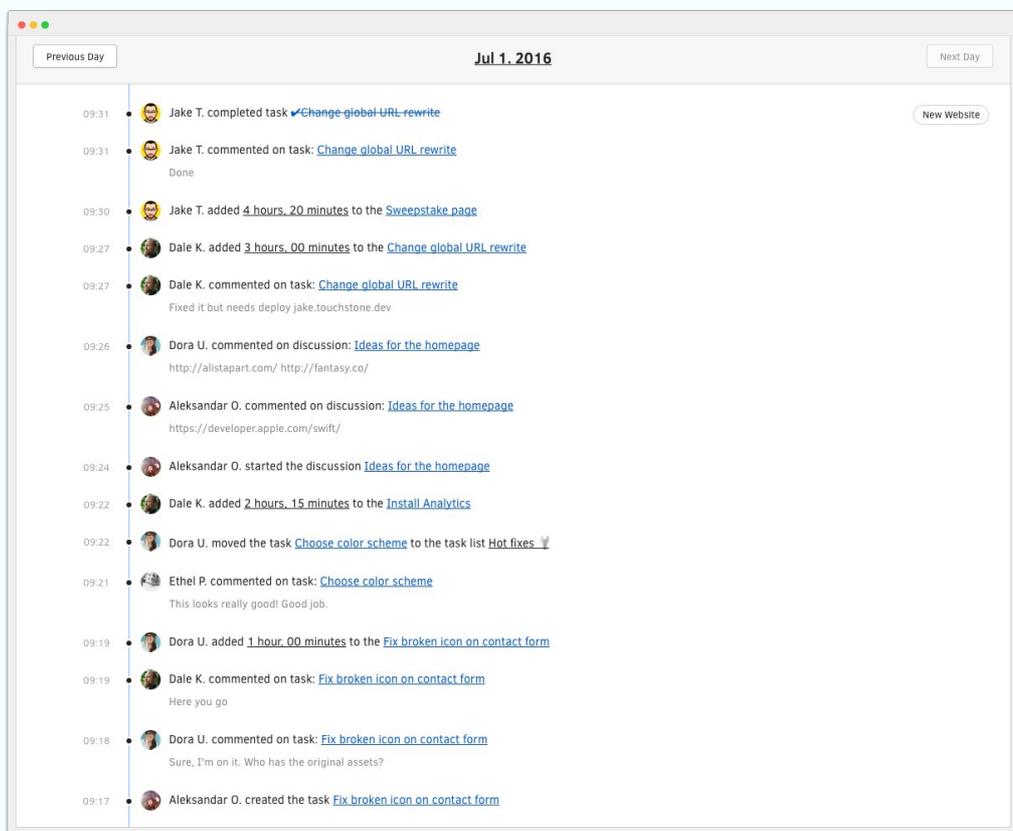
There are several ways you can get the answers to the questions above.

You can organize a status meeting, but meetings are generally a bad idea. A "quick" 1-hour meeting with 6 participants takes up 7 work-hours (taking into account one hour due to context switching). If the goal of the meeting is for you to get a status update, find a better way.

Daily standup meetings are popular because they are shorter and concise. They can be useful if you don't have any structure and need a

place to start. But they aren't always useful as you can skip some days without noticing; plus, keeping them short takes a lot of policing, which can be bad for morale.

The best solution for getting status updates is to encourage the team to update their tasks inside a project management app. It's a chore, but it's an even bigger chore when someone needs information or help, and has to pull others aside to get it. If tasks are updated regularly, you can catch up with what's been happening by going through a particular project's activity stream, instead of asking people what happened or having them send a report.



Each of the three questions from Scrum daily standups can be answered using Active Collab without having the meeting:

- What did I accomplish yesterday? Check the person's activity stream.
- What will I do today? Check the Team Timeline or run the Tasks report.
- What obstacles are impeding my progress? Encourage the team to use @mention in comments, instantly message or talk to the person if it's urgent, or consult the project manager.

Reports are a project manager's best friend but they require a lot of work. When you see a burndown velocity chart, it seems so easy to make - but it's not. The creation of that nice chart takes a lot of badgering. It's simple: if you don't feed the right information into the system, you won't get a useful chart - which is the whole point of having it in the first place.

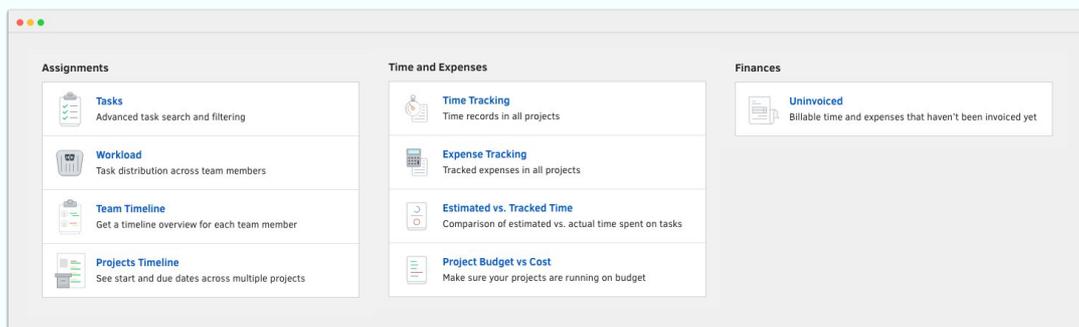
Reports operate under the principle "garbage in, garbage out". If your team member doesn't bother putting estimates or track time, your reports will be worthless.

The best way to get people to enter admin data is by **enforcing a default rule**. For example, when a bug is reported, it needs to be in a standard format or else you won't take it into account. Same way, each task should have an estimate and a label before it can be put in the backlog. If a task doesn't have time logs, it can't be completed.

At first, you'll have to check each task to make sure the team follows the procedure; but after a while, you won't have to. If you refrain yourself from feeding your team a fish (in this case, updating tasks for them), but teach them how to fish (update tasks themselves), you'll be much more productive as a whole.

When you have good data, you get good reports. With Active Collab reports, you can find out the answers to a lot of important questions:

- What's overdue? Run the Tasks report;
- What's coming up? Run the Tasks report;
- Who's too busy and who's idle? Check the Team Timeline report;
- What's someone up to? Check the user activity from the People section;
- What's new? Check a project's activity and the Global Activity;
- Who has too many tasks? Run the Tasks report and group results by user.



You can even get an answer to highly specific queries like: What are all the bugs related to feature X that the client reported last month but haven't been yet been completed? Just run the Tasks report and you'll find out. You can even save the queries as presets so you can use the report as a mini dashboard.

In addition to activity on the project, you can keep track of the time and budget and get answers to the following questions:

- How much has been spent and where? Run the Project Budget vs Cost report.
- How much time has someone tracked? Run the Time Tracking report.
- How good are our estimates? Run the Estimated vs. Tracked Time report.

Once you have the answer, you can correct the situation. For example, if some tasks take longer than expected, you can break them down into subtasks. If you see that you burned more of the budget than you planned, you can talk to the client about it.

This is another good reason to invite clients to Active Collab. There, they can see for themselves where the budget went, item by item. This level of transparency instills client confidence and makes them want to hire you again because they know they can trust you.

What to Do When Life Happens

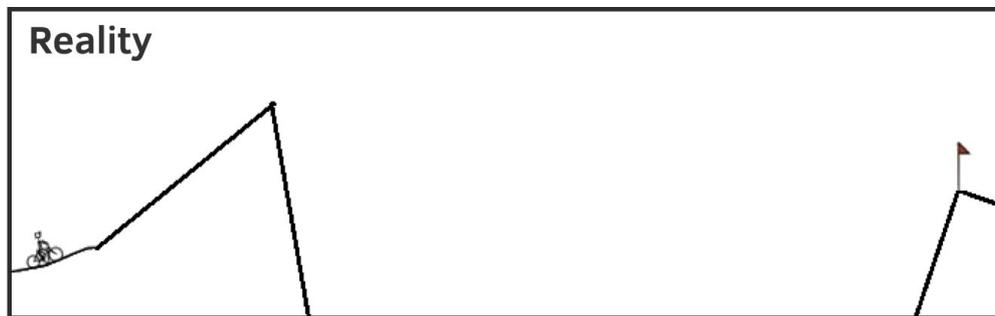
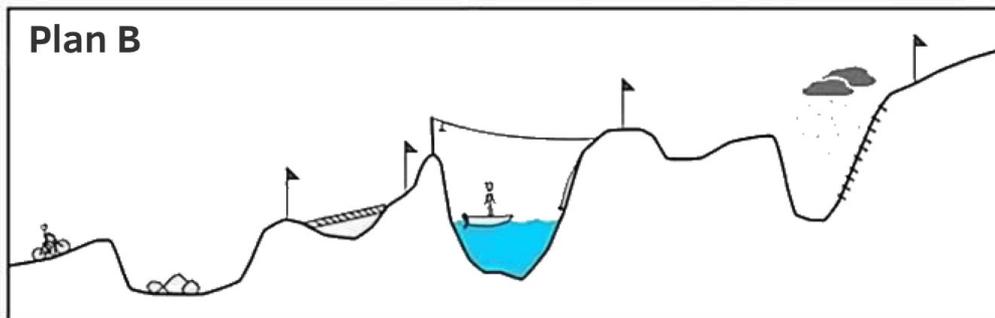
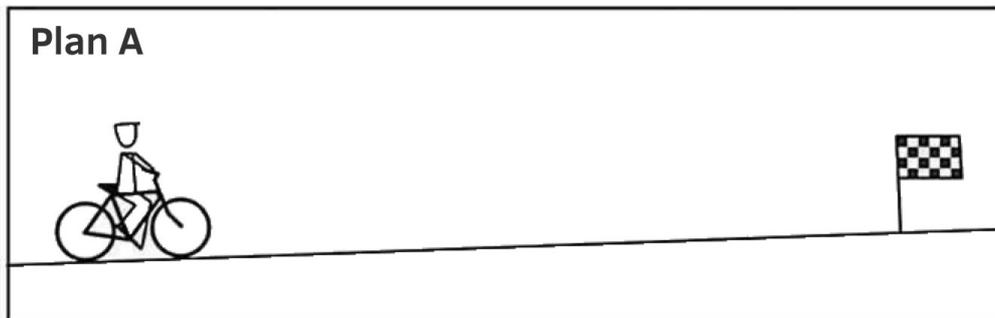
Things change. No matter how well you defined the project scope or planned your project, something won't go as planned. This is your chance to revisit the initial plan and talk to the client about updating the statement of work.

Six stages of a project

- Stage 1: Enthusiasm
- Stage 2: Disillusionment
- Stage 3: Panic and hysteria
- Stage 4: Hunt for the guilty
- Stage 5: Punishment of the innocent
- Stage 6: Reward for the uninvolved

When something threatens to derail your project, you can be:

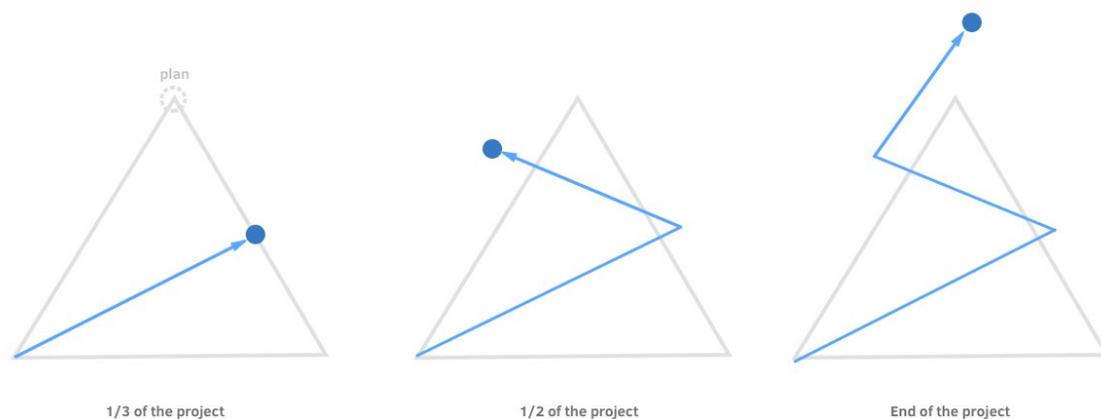
- Optimistic and try to push your team to finish the project on time,
- or realistic and make the plan fit the new circumstances.



DOGHOUSE DIARIES

When planning, know that a **plan will need adjustments** as you work and discover new things. Don't get attached to the plan so much that you are reluctant to change it or pivot. This is your opportunity to make the plan better.

For example, you might plan to end up at point Z in 2 months; but along the way, you're likely to visit points A and B during the projects. You may even end up somewhere beyond point Z.



Nothing is set in stone and it's ok to change the plan, even pivot, as long as you communicate with the client and they agree.

That's why it's good to make buffer zones in the plan: the periods when you can reassess where you are and update the plan. This often happens after research and after creating a high-fidelity design.

Having **frequent milestones** also help you keep things on the right track, especially with larger projects. If you only have long-term, high-level milestones, you won't have the chance to realize something is wrong until it's too late. Milestones give you a chance to reflect and discover issues before they become too dangerous.

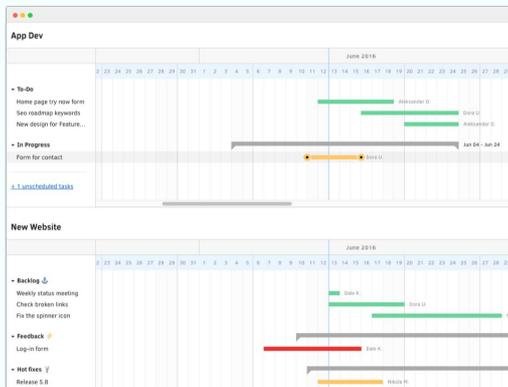
Often, half way through the project, someone will tell you that meeting the deadline is impossible because of a bunch of things. If you involved your client and invited them on a project, they'll be aware of the issue and you'll face the problem together.

When project requirements (date, budget, or features) change, there will be paperwork. You have to discuss the course of action with the client and add an addendum to your statement of work.

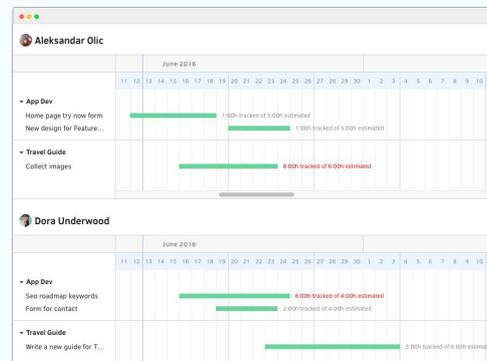
There are several ways to know when you won't make the deadline:

- The most obvious one is that someone will tell you, or at least warn you ahead of time.
- Another way to see if you'll miss the deadline is by reviewing the timeline on a weekly basis.

With the Project Timeline report, you have all your projects in one place and can see which tasks are late (red) and due soon (orange). Then you can open late tasks and leave a comment, asking for a status update.



Projects Timeline report



Team Timeline report

You can also run the Estimated vs Tracked Time report to see which tasks took longer than expected. You can also get this info in the Tasks report and Team Timeline.

When reviewing deadlines, keep in mind that the **project deadline doesn't matter**. The only deadline that matters is the next one because it affects every other deadline. If you miss the design deadline, you can't meet the development deadline, and, ultimately, the launch date.

A cool trick is to think about deadlines as time-limited challenges. This way, you don't panic and approach the whole thing from a positive perspective.

What to do when you realize that you're aren't going to meet the deadline? You can:

- Move the deadline;
- Increase the budget and hire contractors;
- Reduce the project scope.

No matter what action you choose, you have to work it out and negotiate with the client.

Sometimes, the **client will change the project scope** at the last minute but insist on the same timeline. If you defined the project scope and agreed on the terms of your engagement in a contract, this won't happen. That's why it's important to always have a written trace of what was approved and when. If you can point to what the client said in writing, you'll have an easier time negotiating.

Your team can also cause **scope creep** (scope creep is introducing new elements into the project that might overwhelm the available resources). This happens when a team member is so enthusiastic about the project that they want to make it perfect so they work on more than what's agreed upon. The problem is, the original plan doesn't account for the extra work and something more important can get delayed.

Chapter 008

Team Collaboration

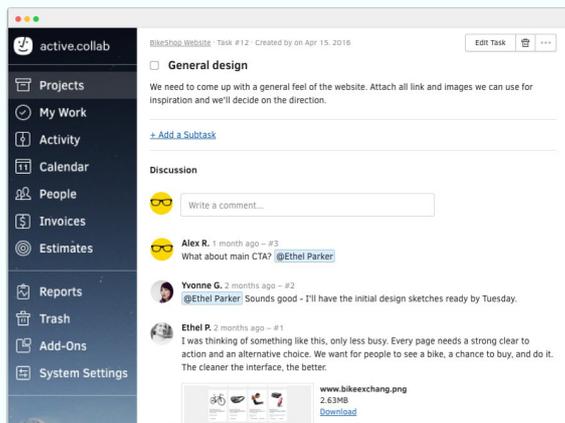


When it comes to team collaboration, you can either read hundreds of management books and studies - and still have no idea what to do - or you can simply get out of people's way so they can get things done.

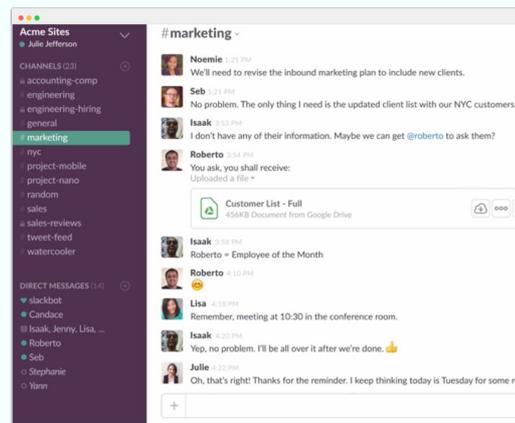


Keeping Everyone in the Loop

When you **keep the nexus of activity online** (across chat rooms, tasks, and discussion boards), it doesn't matter if your whole team works in the same office or remotely because everyone is on equal footing. Someone who works remotely doesn't miss out on the latest events if they're not by the water cooler - all they need to do is go online and they'll be up to speed.



Project management tool: Organized collaboration around tasks



Instant messaging tool: Collaboration around interests

Active Collab is specifically designed so people can focus on work. There are no intrusive notifications that ask you to drop what you're doing. When you're ready, you'll head to My Work page or your inbox and deal with the notification when it's best for you.

Another benefit is **increased productivity** because if everything is online, there's no reason to interrupt someone to get information. A typical developer, while in the zone, keeps a million of things in their head, from variables and loops to utility functions and API calls; when someone interrupts them every 11 minutes, they need 20 minutes to return to the zone where they can get something done. That's a major productivity killer.

When you bring a new person on the team, it's a great time to teach them your collaboration etiquette:

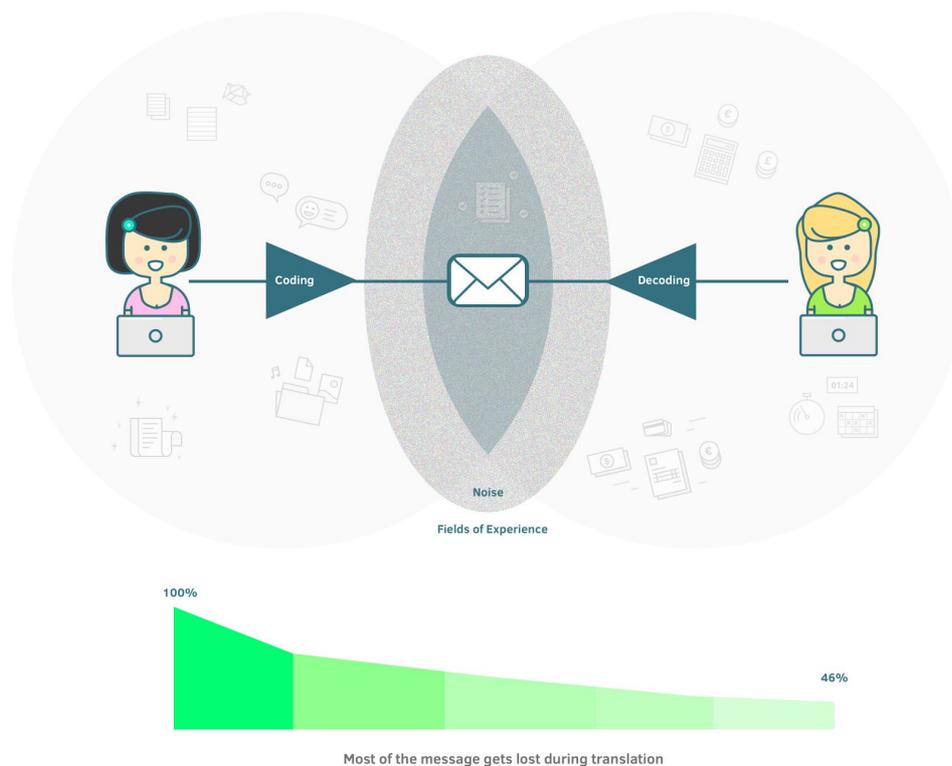
- When you have a non-urgent question, don't head to the person's office but ping them on chat or leave a comment on a task.

- For general questions and talk, send a message to the appropriate chat room or start a discussion so anyone can chime in when they have the time.
- If someone is working and doesn't want to be distracted, they'll have the chat and Active Collab windows closed.

Why Misunderstandings Happen

Misunderstandings happen because there's a big drop off between the sender and the receiver. When you send a message, it goes through a lot of processes and the original meaning gets lost. To make sure there aren't any misunderstandings, keep in mind how communication actually works.

Let's say a client wants to tell a designer what kind of website they want. A client first writes the message and right there, during that writing process, the **message loses its original meaning**. Maybe they can't communicate context, have a different interpretation of common terms, or maybe they're not so good with words. Even professional writers struggle with concise and clear communication.



The medium (eg. email or task comment) also takes away some of the message by limiting what can be said and how. Then, there's the noise around the medium that downgrades the message's quality: reading a long email in a noisy office while there are thousands of other unread emails is different than reading the same message on a piece of paper in your private office.

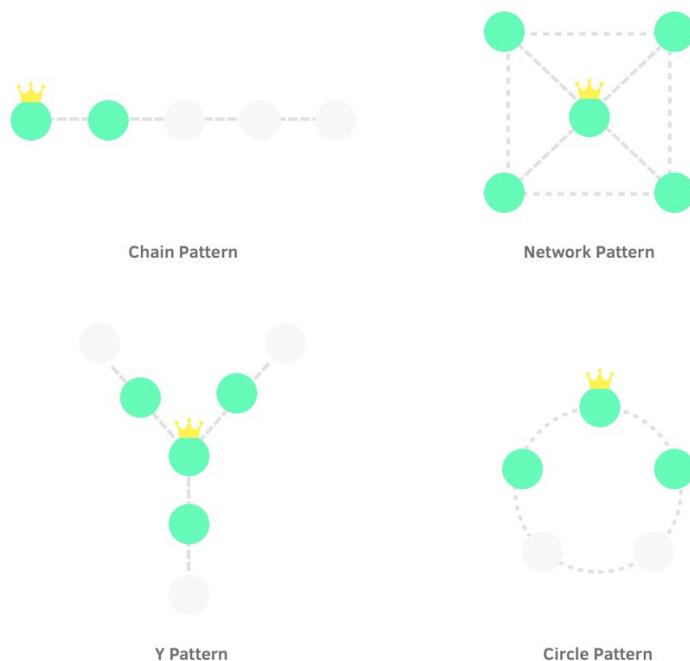
Then comes decoding, when a person interprets what you wrote and distorts the original message further. When the designer reads your email, they have to decode the message using their own field of experience.

For example, you may ask for a prototype and the designer may understand comp, so he'll spend a lot of time making pixel-perfect PSDs when all you wanted was a fancier wireframe. The good news is, the more you work together, the more your fields of experience overlap and there's less room for misunderstanding.

At the end, the designer gets a different message than the one client had in their head. So, when you communicate, take into the account:

- How you express something,
- the medium and the environment,
- and the other person's field of experience.

There are **four basic communication patterns**: circle, chain, Y, and network.



The network pattern is the most efficient and the one you can use only if you keep the nexus of activity online. In the network pattern, everyone can communicate directly with everyone else so there's less room for message distortion.

The network pattern saves you from "**monkey on the back**" problems (being responsible for someone else's problem). This problem happens when someone can't proceed without the manager's approval so they hand off the problem to the manager, thus giving him the monkey.

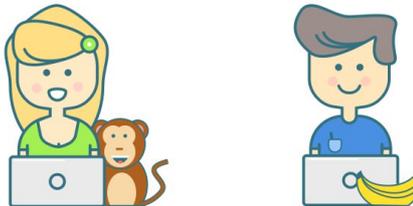
For example, a developer might run into you and say:

Hi there! Great to run into you, because you see, we've got a problem with...

You're in a rush so you say you'll let them know later. You might wonder what's wrong with that.

But let's examine what really happened: before the two of you met, the developer had a monkey on his back and you were free; once you parted, the monkey was on your back and the developer was free. Basically, you got stuck with the problem, thus neglecting your own work.

- Choose color scheme
- Remove ; sign from first h1
- Make font bigger
- Release 5.8
- Spelling errors in blog post
- Update features page



To make matters worse, people sometimes have no choice but to give away their monkeys to managers due to bureaucratic reasons. The more managers take on their back, the greater the bottleneck they become; all until they end up with so many problems that they don't have the time to do their job. While managers sit on a big pile of tasks, others will complain how they can't make up their mind.

The solution to the monkey problem is to set clear boundaries from the start and **never accept the ownership of the monkey**. At no time, while helping, will someone's problem become your problem. It means that if someone asks for a consultation, it's their job to leave with the solution.

It's like in school: if someone asks you to help them with their math homework, you should help them but at no point should you touch the pen or do the homework yourself. By taking the pen, you set yourself up for more work down the road and people know they can take advantage of you.

Never have issues on a project: issues are talked about, problems are solved.

How Not to Sound Like an Arrogant Jerk

Digital projects are different when it comes to communication because people who work together aren't necessarily in the same place or time zone. This means most of the communication is in written form.

The trouble with electronic communication is that it's **easy to misconstrue**. You can't just write:

"It's good."

"Send the assets, pls."

You have to write:

"Great job! This is really excellent, I love it!"

"I need the assets to finish the design. Could you please send them as soon as you have the time?"

Both messages say the same thing but they make us feel and respond differently. The first pair of comments seems harsh while the latter makes us like the other person more.

Talking face to face is a lot simpler as we don't have to bother with "padding" our message because of voice, rhythm, and body language do the job. This is why emails tend to be long and artificial - we need to compensate for the lack of body language by writing more or else we risk sounding brusque and rude.

It's also why we sweat over each word when we're about to send an email to the boss, because we have to think of all the possible interpretations. So we spend 10 minutes writing and rewriting a request that could be done faster if we met them in a hallway. What's worse, we all complain when we get long emails - but we'd be offended if it didn't

have some fluff.

To avoid sounding like a jerk, you should **go out of your way to explain your reasoning**. This may feel phony, but it works. It softens the in-your-face tone your message could otherwise have. Even starting a message with a cheerful "Hi, Amy" instead of just "Amy" can soften the impression. Adding a few emojis and informal "hey"s and "what you think"s can also soften the message without creating too much padding.

Electronic communication suffers from two additional problems:

- Delay - a problem can take much longer to solve using email than talking;
- Polarized decisions - we are less likely to make a compromise when we try to reach a solution electronically; this happens because we lack the social awareness and nonverbal channels which reduce our need to compromise.

The solution is to mix electronic communication with conference calls and an occasional face-to-face meeting. This is especially important for cross-functional project teams who work on complicated problems. Without a real meeting from time to time, projects take much longer to finish.

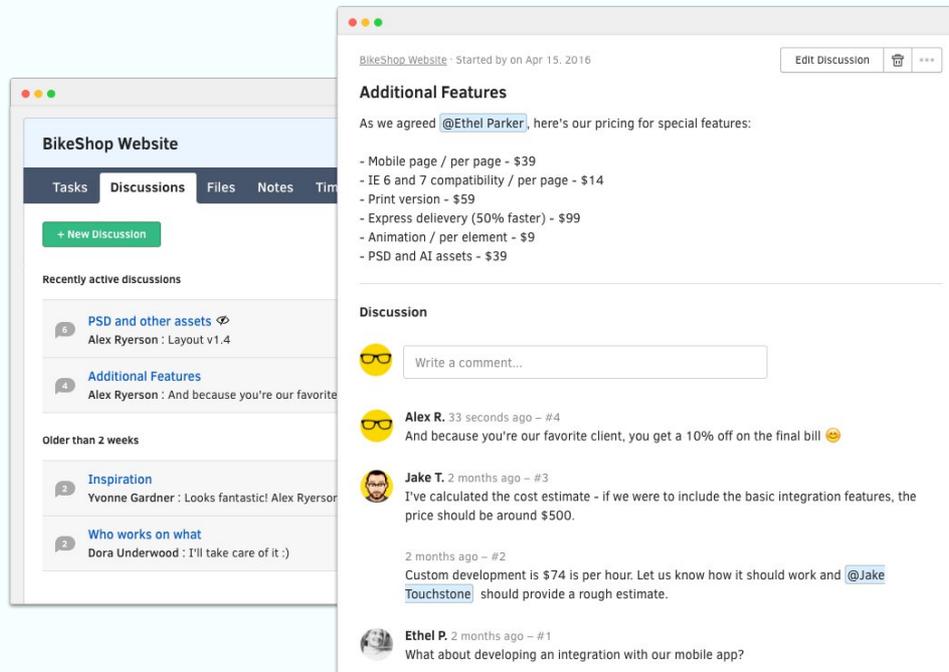
How to Collaborate With Your Team

Most of the collaboration is straightforward. For example, a designer quickly mocks up several ideas, almost all of them awful. After a few days, the team picks 3 strongest ones. The designer gets feedback and chooses the best approach. Then they refine the idea, consult with the client, and have the final design ready for coding.

Early work is always awful and that is perfectly ok; you need to start somewhere. Once you have something, people will have ideas for improving it. The initial work needs to be good enough to convey the basic idea so you can have a conversation about it. From then on, it's about polishing and communication.

Spend time prototyping just enough to figure out what it is you're building. Comps are always an unfinished mess and don't look anything like the final product; so treat them as a throwaway work and an intermediate step to the final goal. Don't include them as part of the final deliverables; the last thing you want is to spend time updating pictures when the client paid you for a website.

During a project, people work alone on their task and then ask for feedback using task comments. For general things, project discussions are better.



Discussions are great for status updates, agreeing on the sitemap, what to work on next, the color palette, or even sharing preliminary mockups. For example, you might want your client to see that you're working on a task but you don't want them to see the result until it's finished. In that case, create a discussion and hide it from them. This can even serve as the place where you'll share all your PSDs and high-res files you don't want your client to access before they pay.

During discussions, there will be **disagreements**. The key to overcoming them is to ask each side to explain their reasoning and then keep asking "why" until they run out of arguments. Most discussions will unravel once a person is asked to explain their line of thought. If it turns out both sides have good arguments, the person who can back up their argument with data usually wins.

The best argument you can have in a report or discussion is a user testing video. Watching actual users struggling with the system is more convincing than reading about it:

How can't he see the button? It's right there!

Some people are **better at arguing using email** than talking in person or via instant messengers. They can prepare intricate arguments to justify their positions because they have time to shape their thoughts. Others are better at real-time communication like instant messaging and meetings because they can react faster. By using comments on tasks and discussions, you give everyone time to make their case so the best idea can win. Don't let someone win just because they can outtalk their competition - it's about the quality of the idea, not the presenter's charisma.

Everyone on the team should be able to add tasks, start discussions, and see everything. Active Collab is about getting things done together and restricting who can do what on a project is against that. You should trust your team and not be afraid to let them see everything related to that project. If you can't trust your people, you have much bigger problems than restricting access.

If you're bringing in **contractors** to your team, make sure they have access to all the information and insight they need. Don't pigeonhole them and restrict their access. You can't distribute information on a "need-to-know" basis because you can't know what a contractor needs to know. If they're a designer, for example, they need to have access to research and brief. Make them feel like they're a part of the team, even if it's temporary. It's good for morale, productivity, and quality.

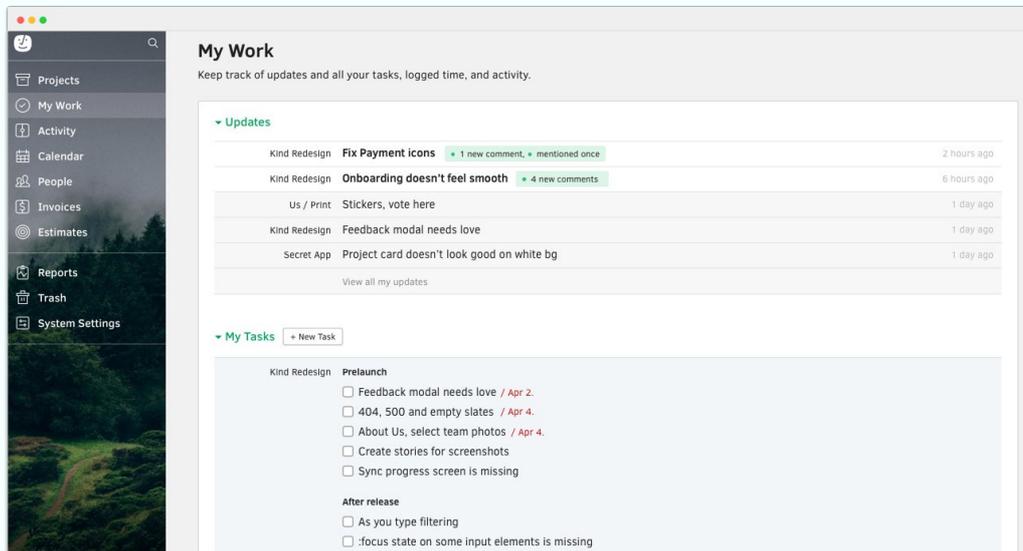
You can **keep all project-related information** (such as project estimates, PDF reports, service agreements, contracts, research, credentials, etc) inside notes and discussions, or even tasks - they support attachments and you can hide them from clients.

Keep all your assets in one place and share them with your team. Keep working files in Google Drive or Dropbox so others can access the latest version whenever they need without asking you to send it. When a file is ready for feedback, attach it to the task so people can comment. You can keep finished assets in Discussions too.

As you work, move tasks from list to list or change labels to indicate progress. You can even prepend the status as the first part of a task's name (eg. Round1 TaskA, Round2 TaskA).

When you need input from someone, use **@mentions** so they get a notification.

All the notifications will appear on their “My Work” page. Plus they'll get an email to which they can directly reply from their inbox.



Each member has their own personal dashboard, My Work. It contains all their updates, assigned tasks, time logs, and activities. It's a one-stop place from which they can track everything they're working on.

How Teams Form

When a new project starts, people get together and start working. Each team goes from a shy group of people too polite to disagree to a well-oiled machinery that gets the job done; this happens in four team forming phases: forming, storming, norming, and performing.

The first stage is **forming**. People are enthusiastic, positive, and polite to everyone. They may not know fully what the project is about and how they fit in. They don't have established modes of communication so they're extra careful not to offend anyone. This is the phase when a project manager's role is the most dominant and everyone turns to them for guidance.

After people get to know each other, they're less inhibited to disagree and enter into conflicts. This the **storming** phase. It's when people find out what they do and how they'll work - and some people won't like it.

Expect resistance and lots of compromise at this stage.

Once people start resolving their differences and appreciating their colleagues' strengths, they'll enter the **norming** stage. They may develop internal jokes, bond, and start hanging out after work.

The last stage is **performing**, a point where the team can handle anything you throw at them. They share the same goal, everyone pulls in the same direction, and nothing can disrupt their performance because they know how to work with each other.



In smaller companies, forming new teams doesn't happen often. Instead, there's always the same group of people who work on every project. In those cases, these phases occur when there's a new hire only it's not so dramatic; the new person just inherits the culture already in place and tries to make it work with their previous team culture.

The Magic Formula for Good Teams

What separates a great team from an average one? Managers at Google asked the same thing and decided to find answers with Project Aristotle. Some of the things they asked were:

- How frequently do particular people eat together?
- Which traits do the best managers share?
- Should you put introverts together?
- What happens when you put together only the best workers in a team?
- Are teams more effective when everyone is friends outside work?

They took everything into the account: socializing outside office, shared hobbies, educational backgrounds, outgoingness, memberships, age, gender balance, team longevity, etc.

In the end, researchers **didn't find the magic formula** for the right mix of specific personality types or skills or backgrounds. No matter how they

arranged the data, they couldn't find a pattern. How a team was composed didn't make any difference.

But they did find out what made one team better than another: **psychological safety**. In successful teams, people:

- Are comfortable to be vulnerable in front of others,
- and feel safe to take risks.

There were other important dynamics, but psychological safety was critical.

The reason was simple: we don't speak up when we don't understand something because we don't want to risk sounding stupid. Although this self-protection is normal, it's detrimental to effective teamwork.

People also don't want to put on a "work face" and leave a part of their personality at home. They want to be who they are without fearing someone might say something negative. To fulfill our maximum potential, we must know that we can freely talk about messy and sad things without fear of recrimination. We can't just focus on efficiency - we need to know others really hear us and that work is more than labor.

This psychological safety is difficult to create. There's no memo or a rule that can force us to respect others. It's part of the company culture which takes years to evolve. In today's fast-paced age, where companies open and close every few years and people drift from one company to another every few months, and sometimes culture doesn't have a chance to get created in the first place.

The best you can do is:

- Discourage negativity;
- Lead by example by sharing your own personal highs and lows;
- Appreciate every idea, no matter if you like it or not;
- Focus on creating a positive workplace where people aren't afraid to speak up or ask for clarification;
- Build trust by giving trust.

For teams that are spread among 3 continents, this is a bit trickier because there is rarely the right time to talk about personal life. As teams became more virtual, cooperation naturally declines. The only way to prevent it is to take concrete measures to establish a collaborative culture, like organizing get-togethers every month (or at least every year).

This happens because people don't know each other and the gap is more difficult to bridge. But if people work together long enough, they'll develop tacit understanding: they'll know each person's strengths and weaknesses, have shared experiences from which to draw upon, and unwritten rules that'll help them collaborate more closely.

How Can Developers Collaborate Better

Developers work best when they work alone, but there are a few instances where collaboration brings better results - namely peer review and pair programming.

Peer review is when a developer finishes coding and gives the code to someone else to examine for defects or improvements. The code that goes through the peer review is of higher quality and has fewer bugs - which means lower costs in the long run.

It's much cheaper to fix a bug before deployment than to spend time documenting the bug, entering it into the backlog, deciding on its priority, and then figuring out how the code works in the first place.

If you're very particular about what goes into your codebase, code quality will improve. If code is rejected and goes back to In-Progress, people will grumble at first. As you work more and more, code acceptance on the first pass becomes a matter of pride and people will do their best to produce clean, maintainable, well-designed code. That is, if the management values quality over deadlines.

Code review does take some sacrifice. Because another developer goes through the same work, they don't produce code of their own. This means you get less done in the short run, but you save the project from work down the road.

A good solution to the productivity problem is to make developers pull tasks from a Code Review task list while they wait for QA (they can't pull more work due to the WIP limit).

The only problem with peer review is that it takes time for someone to review new code and get familiar with it (especially if it's a major feature). Some reviewer will tend to skim and give very general feedback. Later, when the developer gets to work on the reviewed code, they'll have lots of feedback and suggestions - but it'll be too late.

Be careful when you introduce code reviews to developers who didn't have them before. A harsh review can destroy a person and sink morale. Once you do it a few times, people will be more open and will appreciate it as a chance to learn and grow. You just need to make sure to soften the blow until they get to that stage.

Code reviews aren't the same for every task:

- Some need to be big and official, like when you're releasing a big feature or a new module.
- Some are minor, like when you fix a bug and a fellow developer just needs to run a quick check.

Another collaboration technique is **pair programming** when two developers work behind the same computer, sharing a keyboard and mouse. It's much more collaboration-intensive. One dev is the driver and the other the navigator, and they switch roles as needed. One is writing the code and the other is reading and checking it while thinking through problems and what to do next. If one person gets stuck, the other is there to take the wheel.

Advantages of pair programming:

- People with different expertise can attack the problem from multiple angles;
- People learn new things as they show each other neat tricks and workarounds;
- More people know the code and can maintain it in case one of them leaves;
- The code gets written quicker, has fewer bugs and code smells;
- People who work together are less likely to procrastinate;
- It builds morale and interpersonal relationships, which make people enjoy their job more;
- There's less wandering off down blind alleys, losing focus, and banging head against a wall.

Challenges of pair programming:

- Bad in big doses as spending 8 hours working together is too fatiguing;
- It doesn't work when people's personalities or motivations don't match, eg. when a low-skill dev who doesn't care is paired with a senior dev who takes over the process;
- It's hard to convince management it's a good investment when all they see are doubled hours for the same work.

Pair programming is great if you have the resources and can focus on high-quality code. It's also good for risky and experimental projects that are too complex for one person; or if you have a very complicated

debugging process and you need to have another person thinking why something won't work.

Everyone should do code review. Pair programming, on the other hand, should be done from time to time and only if the situation calls for it.

Why Developers and Designers Hate Meetings

To understand why developers and designers hate meetings and interruptions, you have to understand the difference between a maker's and a manager's schedule.

People who create things (designers, writers, developers) operate under a maker's schedule, which is completely different from a manager's.

The **manager's schedule** is neatly divided into 1-hour blocks. They have appointments throughout the day, which can be changed on the fly. They timebox every task and can block off anywhere from half an hour to several hours for a task. Changing the schedule is easy for them, plus they don't sacrifice any productivity while doing it.



Makers don't have this luxury. They can't divide their work into discrete 1-hour units; they need at least half a day of continuous work to get anything done.

Makers carry an overhead cost when they start working. For example, before a developer can start coding, they need at least half an hour to navigate around code and plan what they need to do. Once interrupted,

they need time to get all those things back in their short term memory before they can start working.

Makers need continuous time periods of focused work. Managers don't - they can switch between tasks and modes at no additional cost. Plus, because managers operate under a schedule, they can always protect themselves from interruptions by saying they're busy and point to their schedule to prove it.

That's why meetings are a disaster for makers, especially if they're in the middle of the day. If you work from 9:00-17:00, and have a meeting from 12:00-14:00 (plus lunch), by the time you get to work, you'll only have maybe two hours of continuous work - and that is if no one interrupts you. That's barely enough to even get started. That's why people finish their day without getting anything done.

Since managers have more power in organizations, they make everyone submit to their way of scheduling. Business people also have the privilege to have speculative meetings because they can afford them. A designer will have more trouble agreeing to "a cup of coffee" as the only way to get things done is to have a large, contiguous chunk of time to do it.

There are several ways to make collaboration between management and makers better:

- Let your team have flexible work hours or work from home. For example, most freelance developers code at night because no one can interrupt them and they can get things done while the world is quiet.
- Schedule meetings at the very start of the day. This way, people don't have to switch gears in the middle of work and can work without keeping one eye on the clock to make it to the meeting.
- Limit work to 40 hours/week, not 8 hours/day. If the person is in the zone, the worst thing they can do is leave the zone because the time's up. Plus, if they have one hour left, instead of waiting it out, they'll go home and use that hour better next day.

Productive people also **know when to work on what**. There are some tasks that require a lot of work and focus, and others that are comfortingly routine. Sometimes, you just don't have the willpower to start working on a large feature or solve a complex problem.

Debugging is perfect at times like those because it's so straightforward. The program is supposed to do x, but instead does y. You have to find out why. The problem is constrained and you know you're going to win in the end. This makes debugging a relaxing low-level activity when compared to starting something big.

Some low-level tasks are: checking email, responding to messages, admin stuff, catching up on news, updating tasks, checking results, etc.

Collaboration Myths

Lately, we came to revere collaboration as something that makes or breaks an organization. Collaboration become a silver bullet for everything. If you can't solve a problem or can't meet milestones on time - collaborate! We begin to insist on collaboration even in places where it hinders productivity.

Take Linux or Wikipedia for example, all projects that became successful because of the sheer power of collaboration. They make us revere the hive mind, the wisdom of crowds, the miracle of crowdsourcing.

But we're missing the big picture. If we take a closer look, all those projects were created by people working alone. There were no brainstorming sessions or huddle ups. They were all asynchronous, relatively anonymous interactions. This doesn't sound anything like a typical, politically charged, face-to-face open office.

We need alone time to get things done. For example, the best musicians and athletes become the best not by playing with the orchestra or in a team - they **become the best by practicing alone** for long stretches of time. Same way with students: those who study alone learn more than those who work in groups.

You can engage in deliberate practice - the thing that actually makes you better - only when you're alone. When we practice deliberately, we identify tasks that are out of reach, learn how to do them, monitor progress, and revise the process.

This applies especially to developers, designers, and writers. Kafka for example couldn't write when his fiancée was near him:

"You once said that you would like to sit beside me while I write. Listen, in that case I could not write at all. For writing means revealing oneself to excess; that utmost of self-revelation and surrender, in which a human being, when involved with others, would feel he was losing himself, and from which, therefore, he will always shrink as long as he is in his right mind. That is why one can never be alone enough when one writes, why there can never be enough silence around one when one writes, why even night is not night enough."

That's why **open-plan offices reduce productivity**. Open office plans simply squeeze more employees in less space, while management hopes that'll make people collaborate more. But it doesn't work like that.

People at open offices need to deal with a lot of interruptions and noise. For instance, a guy to your right might have allergies and is constantly clearing his throat; or a person in front of you may constantly interrupt everyone with a non-funny joke. To make matters worse, the non-productive employees only get louder and louder as time goes on.

Open offices make people generally more hostile, unmotivated, and insecure. Research found that people in open offices:

- Change job more often,
- take more sick days,
- suffer from higher blood pressure and stress levels,
- argue more with colleagues,
- worry someone is eavesdropping or spying their computer screens,
- have fewer personal and confidential conversations,
- are more socially distant and slower to help others,
- have elevated heart rate due to loud and uncontrollable noise.

Collaboration is important but sitting people next to each other doesn't translate automatically to more of it. What people need to be productive is:

- A quiet space to work without distractions (2-4 person per office);
- A place where they can casually mingle and exchange ideas (during lunch or chat);
- A place where people can have a meeting (conference room).

The **presence of others can impair our problem-solving skills**. Due to peer pressure, we tend to follow what others say. No matter how smart we are, we're all susceptible to the herd mentality.

In one experiment, students were given a test so simple that 95% of the group answered every question correctly. But when the experimenters

planted an actor who intentionally gave wrong answers, the percent of students who gave all correct answers dropped to 25%. And the funny thing is, when everyone was asked if they were influenced by the actor, everyone truly believed that they came up with the answer on their own.

Group brainstorming is another popular concept that doesn't work as advertised. Common wisdom says that people in groups generate more ideas than individuals - but that's not true. People produce more ideas of equal or higher quality on their own. And the performance gets worse as the group size increases: groups of four perform better than groups six, which in turn perform better than groups of hundreds

There are three possible explanations to why group brainstorming fails:

- Social loafing: people work less to achieve a goal when they work in a group than when they work alone.
- Production blocking: only one person can talk and generate ideas at a time while others have to sit passively.
- Evaluation apprehension: people are less likely to suggest an idea in fear of looking stupid.

Even though group brainstorming doesn't work, it's getting more popular than ever. That's because people need to believe the group performed much better than it really did; they are attached to the activity and need to justify it, or else admit they wasted time.

It's ok to have a group brainstorming, as long as you know that the main benefit of the activity is social cohesion and team bonding, and not getting best ideas.

The exception is **online brainstorming**. It combines the best of both worlds: people get to think alone and produce more, while at the same time get to bounce ideas off each other. Online brainstorming, when properly managed, gives better results than either group or solitary brainstorming. Even the group size positively affects the results: the more people, the better.

So next time you need to brainstorm some ideas, open a discussion, invite people, and let them collaborate alone yet together. You'll get better results than if you'd organize a meeting and force people to sit passively while the speaker gets to finish their monologue. Plus, you'll have a written trace of all the ideas so you won't have to type them out manually.

Chapter 009

Client Collaboration



No collaboration workflow is complete without the client. They should be the biggest collaborator on your team but many agencies make a mistake by building a wall around what they're doing..

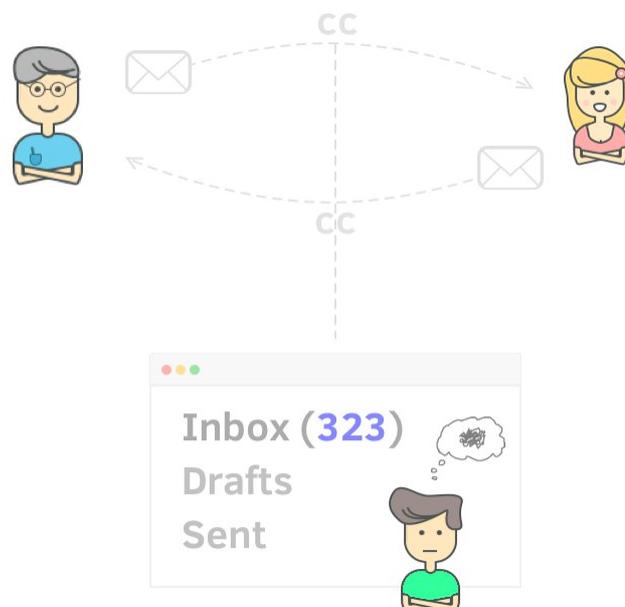
The Client Collaboration Process Is Broken

In most agencies, only the project/account managers deal with clients directly. A client never gets in contact with the designers, developers, and copywriters; they never see how the sausage gets made. That makes sense sometimes - creatives have a reputation for being divas and difficult to deal with. If a client asks to make the logo bigger, a creative's response might lack the required social grace and diplomacy.

In order to successfully deal with clients, all a person has to be is normal and detached. It's very difficult to remain neutral when someone criticizes your work, and respond with tact. That's why project managers take on all the communication. Plus, it's easier for clients to deal with one person than several.

Sometimes, **clients micromanage everything** on a project if they get access. They tell your team members what to do, when their deadlines are, and introduce panic. The team ends up frustrated and feel like they can't collaborate among themselves because the client is watching, monitoring, and judging their every move.

So what usually happens is that **project managers act as intermediates** between the team and the client - thus excluding the client from the process. When a designer needs something, they need to ask a client through the project manager. It's like asking person A at a dinner table to ask person B to pass the salt - it's ridiculous. Not to mention all the CCs and BCCs that overflow your inbox with 300+ emails a day.



Many agencies exacerbate the problem by **building a wall** around what they're doing and leave the client on the other side. The client then feels animosity and distrust, because they're shut out. That's a huge mistake - the client should be inside, doing the heavy lifting and being your biggest collaborator.

These are some of the reasons why agencies build walls around what they do:

- They don't want the client to know the agency outsources their work;
- They're afraid that the client will see how messy the process is;
- Creatives are like wild animals and totally lack soft skills;
- Employees mustn't know anything about hourly rates, tracked time, or other financial details;
- The agency fears the client will poach employees;
- There are a lot of restrictions about who can see what, NDAs, red tape, C.Y.A, and general paranoia;
- Clients tend to micromanage or require too much hand-holding through projects.

These are all valid concerns if you have a broken processes and a shaky organization. It's like when you have bad software and instead of replacing it, you make suboptimal decisions - but sooner or later, that house of cards will collapse.

Types of Clients

Other Agencies

Most creative and marketing agencies outsource work to other specialized agencies and freelancers. So one agency will do a marketing strategy and hire agency A for production, agency B for post-production, agency C for promotion, and so on.

Other agencies are easiest to collaborate with because they know how things work. In Active Collab, the main agency will create a project and invite people as team members from other agencies (and categorize them by company).

SMBs

Small and medium size businesses aren't usually familiar with the collaboration process and need a lot of hand-holding. The good news is they're flexible and you can get a lot of things done rather quickly. Invite them to a project and show them how you work behind the curtains -

they'll appreciate the show and get a chance to learn something new.

The hardest question for them seems to be "name 3 websites you like, 3 you dislike, and why" because they don't know where to start; so make their job easier by suggesting some solutions yourself and asking them what they think.

Corporations

Corporations are the most difficult to collaborate with because there are a lot of stakeholders involved, with varying degrees of authority, needs, and wants. There are a lot of approval stages, hoops to jump through, and CCs to exchange before any decision can be made. You often get assigned a contact person in the organization through whom you'll have to go to in order to get anything, which can be time consuming.

The key to dealing with corporations is to make the job for the other side easier and save them time by keeping all the information structured in one place, with a written trace of every discussion and decision.

Involving Clients From the Start

The most pain free way to work with clients is to involve them from the start. Make it clear that you see them as a team member and expect cooperation and support during the entire project. Go one step further and specify in your contract what exactly you expect from them.

Honesty is the always best policy, for both sides. Disclose what you do and if you outsource work to contractors. The client probably won't care because they hired you to get something done and how you do it doesn't matter as long as you deliver.

Good designers take orders and hand over exactly what a client wants. Great designers dive deep to uncover what a client actually needs.

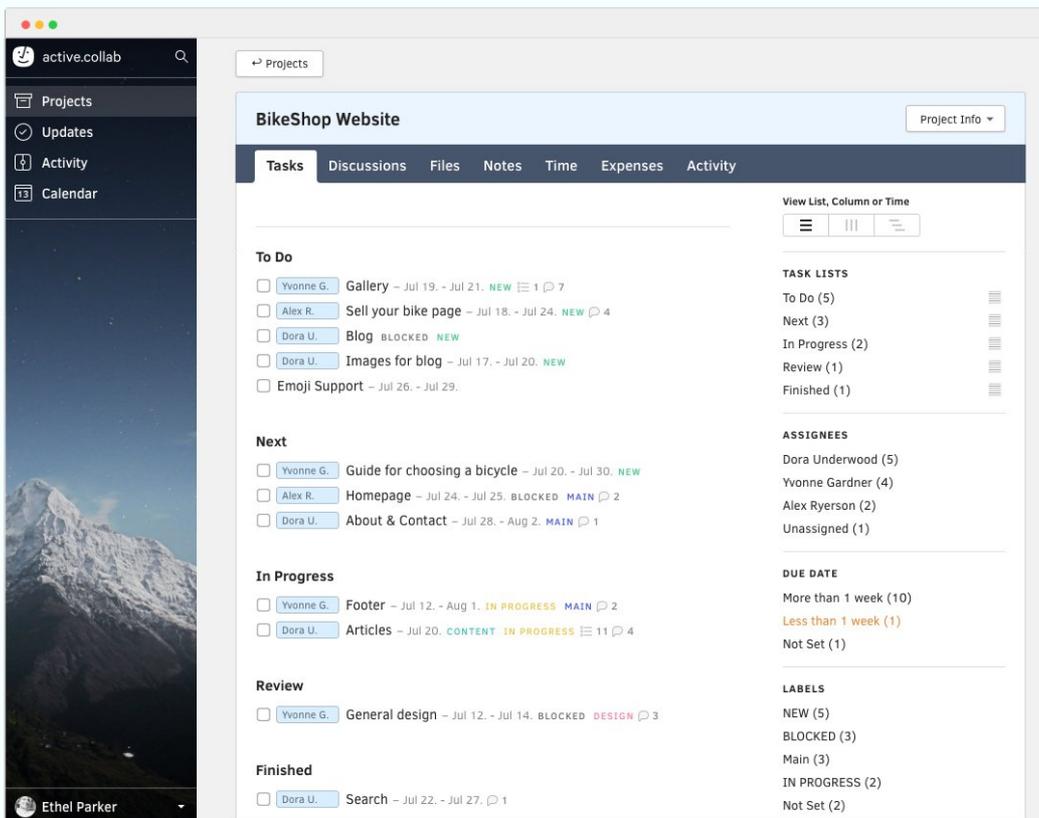
Invite a client to a project so can see what you're working on - that is, if they're interested. Even if they don't track progress by going through task themselves, it's an act of good faith. When you show them you have nothing to hide, they'll trust you more. Plus, you show them how hard you work so they know where their money goes. Then they can see what you're working on, provide feedback, and upload the content and assets you need.

In Active Collab, clients can see what you're working on, provide feedback, and upload the content and assets you need.

Some agencies don't want the client to see everything so they create two projects: one for the client and one for the team. This is not a good solution as it requires double the work and upkeep - and neither are billable activities. It's much easier to let clients see.

If you have some sensitive information or just want to discuss something internally within the team, you can hide individual tasks, discussions, and files. This is particularly useful when you don't want clients to see preliminary designs or hi-res documents before they pay.

Clients have limited access, but if you have a client who's especially collaborative, enabling the Client+ add-on will let you assign tasks to the client - after all, clients do have responsibilities, just like everyone on the project. They still won't see other projects, hidden stuff, people on other projects, or any contact info.



Dealing With Non-Collaborative Clients

Involving client sounds nice in theory but it's difficult in practice. Some clients don't want to be overwhelmed with information or too involved, which is perfectly fine.

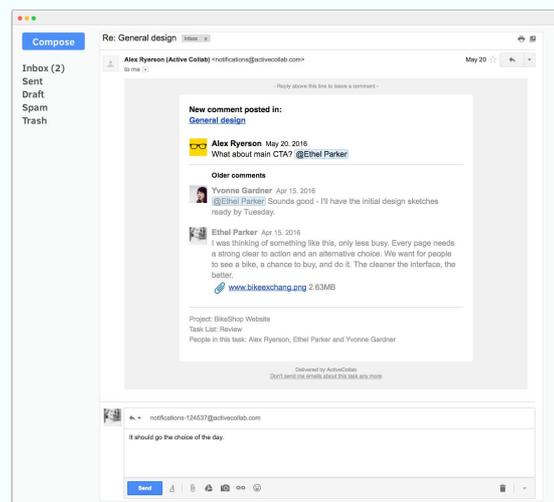
In those cases, a project manager will personally work with the client, update them, ask for feedback, and review progress.

Sometimes, it's not the collaboration itself that bothers a client but the medium: **all they want is simple email communication**. If you work with contractors, your project management software is yet another thing they have to learn. For your clients, the collaboration tool shouldn't interrupt their day or annoy them. So don't force them to use anything they're not comfortable with.

Clients can reply to Active Collab tasks and discussions via email without ever having to use Active Collab. You simply set up an account for your clients using their email and subscribe them to tasks where they'll receive notifications when there's a new comment. This way, they get updates automatically and they can reply from their inbox.

Clients can even send an email to a project address and it will be converted to a discussion. The team can then treat those discussion as new tasks initiated by the client, convert them to a task.

Each project gets a random email address, which is hard to remember for clients. To make it easier to remember, set up an email forwarding so all email from one address goes to the project. For example, if you own @agency.com domain, you can create a new account, like client-project@acme.com, and set up email forwarding so every email that goes to that account is forwarded to the project's email address. This way, a client can easily remember the address when they want to make a request.



Keeping Clients Updated

Clients hate surprises or not knowing what's going on. So you have to make it a habit to update them on your progress, even if they don't ask for it explicitly.

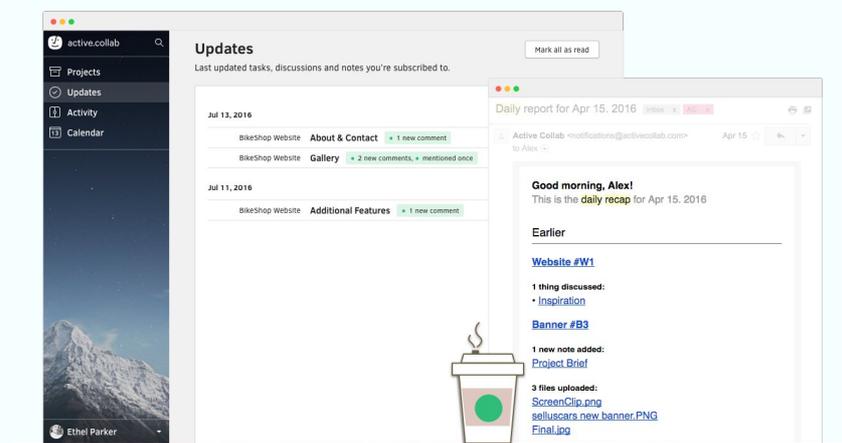
Not getting updates on progress is incredibly frustrating for the client. You can't just hide yourself and all the work from them - you have to **constantly communicate progress**, even if it doesn't contribute to the work directly.

Update your client weekly on what's going on by sending them:

- A list of completed tasks from the previous week;
- Tracked time broken down by type of work;
- Spent vs remaining budget;
- Planned tasks for the upcoming week;
- What you expect from the client this and next week;
- Tasks that are slipping over budget/schedule, why, and how they'll impact the project;
- An executive, one-page summary.

Projects somehow always end up slipping behind schedule. By keeping clients updated each week, you can address the problem on time together. Plus, negotiating additional time or budget will be a lot less awkward.

Active Collab can save you a lot of time by automating reporting for you, especially if your client wants to stay in the loop. The client gets an automatic recap email every morning at 7 AM so they can see what happened the day before. If they want more information, they can access the Activity page to see what happened on all their projects or check Updates to see if they have any notifications that concern them personally.



Answer client messages as soon as possible. When a client doesn't hear anything from you, they can get paranoid - the primary concern being that you're twiddling your thumbs while collecting a fat paycheck. When you're in the same city, the client feels that if something goes wrong, they can at least get a hold of you in person. But when they're remote, they get agitated and suspicious when you don't return their messages.

No matter how much you communicate through messages, find some time to schedule a call to keep them updated. It's better to lose some time rather than damage the client's trust. If you're working remotely, be very available and return calls and emails as soon as you can.

Let clients see for themselves how much you work. Even if you're working like a dog and can't spend additional time debriefing the client, you have to. Agencies often spend more hours than they bill because they don't have any proof of work. The solution is to track time on everything you do and let the client see the work and timesheets for themselves. They'll start valuing your work much more once you get organized and gain real leverage.

Dealing With Change

During the project kickoff, you agreed on the scope, budget, and schedule with your client. But things you can't predict change the full project scope. Sometimes a client finds out they need something else in the middle of the project; or a project is more project complex than you estimated at first.

There's nothing wrong with changing the scope - it's actually advised that you set aside time in your plan for reviewing the initial agreement. Revising the statement of work will come up naturally during the conversation with your client if you kept them updated throughout the project.

When you first notice a project slipping in small places, you have to **inform the client**. Small slips here and there build up into significant schedule and budget problems over time. Don't sweep things under the rug hoping not to let the client down. Slips happen and it's better to talk about them openly than to inform the client the budget is spent, without them knowing where it went. People are optimistic by nature so don't let them assume everything's going great when it's not.

Sometimes you'll miss a milestone because the client didn't leave feedback on time. This is another problem that regular updates solve. If you make it clear to your client that you're running late because of them, they'll know it's not your fault. Plus, you can catch the problem early and discuss how to improve the workflow in the next stages of the project.

But what usually happens when scope or schedule change is that people:

- Keep quiet about the delays from the client, hoping the team will catch up,
- take on additional requirements and "small" jobs without updating the timeline, budget, and statement of work.

For example, the client forgot to think about the content or icons and asks you to come up with something. You need to revisit the statement of work and see how it impacts the project and your bottom line.

Sometimes by taking work, you're actually hurting your profit. **Never take on "small jobs"** without negotiating. Some clients are reluctant to revisit the contract because they want to use you as much as they can and get their money worth. The solution is to add small jobs as extra line-items in your weekly budget reports so the client can see how they impact your budget and schedule.

Try not to please a client too much or else they'll start pushing you around. If you try too hard and take additional work without negotiating, you'll lose money. Project managers usually justify the loss by cross-selling or upselling services, hoping to gain more business in the future, but that's a risky road. What's worse, you'll drive your team to work more (without paying them more), and you risk losing your best workers that way.

When you change a project requirement to factor in more work, give the client a preliminary estimate, a revisited schedule, and the proposed addendum to the statement of work. Just like you did during the client proposal, explain the benefit behind each item. If you're doing a responsive website, don't just tell it'll look better on mobile - explain how it impacts their SEO, user satisfaction, and their bottom line.

Be sure to consult the Project Timeline report to see how this new work impacts your responsibilities to other clients, and price accordingly.

Presenting Work

People think a client hires you to slave for days and finally present your masterpiece in a grand reveal, which the client loves and showers you with praise and money - except it doesn't work like that. In fact, it's the opposite.

Work approval isn't a one time event but a process that happens quietly throughout the project. The exceptions are small items (like banners and icons) but even that's up for debate.

Your goal is to make the final approval just a formality since the client knows the end product and already gave their blessing. To get to this level, you need to:

Know who the decision maker is

Going into a project, you need to know who on the client side provides input, who gives feedback, and who approves it. If your point of contact doesn't have the decision-making authority, plan for it by setting aside some time for decision making as your contact will have to push their busy bosses for feedback, which will take time.

Get political buy-in

Try to talk to each stakeholder on the client side even if you don't need their input. You never know who can affect the decision making so talk to each department that's affected by your work purely out of political reasons. Plus, inquiry is flattery and inviting people to participate empowers them.

Remember that there are more stakeholders than you might initially think. Executives have to defend your work as a part of the overall strategy; customer service has to support it; sales has to sell it; and production has to maintain it. Make them all your allies by involving them, and they'll defend your work because they are in part responsible for it.

If you don't include others, you'll have a hard time presenting your work because the group you skipped will challenge your recommendations and ask why you didn't consider this or that - thus turning the presentation into a debate. But if you involve them, by the time the formal proposal comes up for approval, the decision will already have been made and the final meeting will just be a formality.

Show work often

When you show work often, you can get a feel of how the client thinks and steer the project accordingly. Plus, you chip away at a client's natural situational anxiety. Once a client pays you a big deposit, they're anxious to know what they're paying for. When they see results, they'll feel better about the relationship. That is one more good reason to invite them to projects.

Be careful what you present though. If you're presenting backend, keep the frontend very rough because clients cling to visuals; if there are no visuals, they can focus on functionality.

Don't let clients have a lot of options. It's better to have three good initial designs than presenting ten great ones - more choices make clients indecisive and less happy with the end product. In fact, presenting only one is a completely viable strategy. Steve Rand had a rule of making only one logo and no rework, and the client had to pay fully for the work even if they didn't like it. It's suffice to say, clients were very happy. Sometimes, less is more.

When showing work, be sure to let a client know how you arrived to the latest iteration. Show the the first draft, why it didn't work, what you did then, your thought process, and the steps between. This will stop comments like "make logo bigger" because you've educated the client and made them like you.

Get rejected early

The faster you can find out what the client wants, the faster you can finish a project and the less time you'll spend on solutions that don't work. 90% of your job is being rejected and getting upset isn't worth it. Being rejected is the hazard of doing creative work.

It's extremely helpful when you learn the "why" of the rejection because you can address and bypass the objection. Sales pros have a clever technique called the "alternative close". Instead of asking:

"Do you like the solution?"

...you tell them:

"We can pursue this direction or that direction."

By doing this, the negotiation becomes not a matter of "if", but "how".

But if arguing about a task is going to take as long as doing the task, it's better to just do it.

Some clients reject the first draft and don't even bother reviewing it. They expect the first idea won't be good and they're waiting for you to step up your game. Once they reject you a few times, you earn their attention.

Other clients are afraid to hurt your feelings by saying no and end up unhappy. You need to make it clear to them that if they see something that isn't working, they need to point it out. Otherwise, you will show them the same thing again and again until the project runs out of time and money and the client gets stuck with a solution they don't want.

Arrange pre-meetings

When it's time to show your work during an official meeting to multiple stakeholders, arrange several informal pre-meetings with each of them to avoid the "big reveal". These may seem like unnecessary conversations, but during your presentation, the stakeholders will smile knowingly and affirmatively nod their heads. There won't be any puzzled "I'm processing it" expressions because you gave them the time and space to think, establish their opinion, and give you their support.

Chapter 010

Time Tracking



When you charge by the hour, time tracking is a must. After all, you can only charge a client if you can show them the timesheet. But even if you charge a flat fee, it's a good idea to track time so you can gauge how productive you are.

Hourly Rates vs Flat Fees

Most agencies and freelancers charge by the hour. They have hourly rates for different job types (like design, writing, development) and quickly create invoices based on the time they spend on particular tasks.

Charging by the hour has become the norm, but there are still some clients who don't like the open-endedness of time tracking. They have a budget, a clear idea what they need, and paying by the hour seems risky because they can't control it.

When searching for an app developer, a client just wants to know if they can make the app and how much it'll all cost. They don't care if you'll spend 30 hours on design and 60 hours on development, or vice versa. They just want to know the bottom line.

When managing a client proposal, you first need to determine whether to charge by the hour or a flat fee.

Charging by the hour is the best solution for you because it has the least amount of risk. The client is less likely to ask for rework or change the project scope because they have to pay for any extra work. As far as you're concerned, they can ask for a thousand revisions as long as they pay you by the hour.

A common error first-time freelancers and agencies make is to charge a flat rate for a whole project. Then, they're stuck with a huge project in development and the client keeps asking for more and more work, thus delaying milestones and payment. In the end, they spend a lot of time making the client happy but end up earning nothing.

There are some instances where charging a flat, until-the-client-is-happy fee is acceptable:

- When you can estimate with 100% certainty how long the work will take,
- You can finish work faster than estimated,
- Small jobs that take less than a couple of hours at most,
- You love working on a project and don't care how much time you spend.

A good **rule of thumb** when it comes to charging:

- If a project is complex and prone to scope creep, charge hourly.
- If you're doing routine and small jobs, charge a flat fee.

How to Track Time

The best way to track time is to keep time records associated with the tasks they belong to. So if you're working on a task "Roadmap page" and you spent 5 hours on it, log that time directly on a task so you can keep all the information in one place for later reference and billing.

Let's say you agreed on 200 hours for a project and you divide that time like this:

- "Roadmap page" takes up 40 hours of design and is assigned to Dora
- "Log-in form" takes up 120 hours of coding (Dale) and 40 hours of testing (Jake)

You create the tasks, put estimates next to each one, and then let team members log time records against them.

For example, Dora works for 3 hours on the "Roadmap page" task on Monday and 5 hours on Tuesday, and each day she adds a time record with comments on what she did (either manually or via a timer app). For "Log-in form", both Dale and Jake add their own time records and comments.

The screenshot displays a time tracking application interface. It features a main window titled "Time records for : Roadmap page" and a secondary window titled "Time records for : Log-in form".

Time records for : Roadmap page

- Job Type: General
- Date: Jul 20, 2016
- Billable:
- Time Record: 5: Done HTML and basic structure
- Buttons: Add Time Record, Cancel
- Summary: Jul 19, 2016, 3:00 | Design, Dora U., Made initial wireframe

Time records for : Log-in form

Date	Time	Job Type	Assignee	Description	Billable	More
Jul 20, 2016	3:10	Testing	Jake T.	Web page when logged out	BILLABLE	...
	3:50	Coding	Dale K.	Setting cookies	BILLABLE	...
Jul 19, 2016	5:50	Testing	Jake T.	Member log-in form	BILLABLE	...
	4:20	Coding	Dale K.	DB planning	BILLABLE	...

Time and Expense Tracking

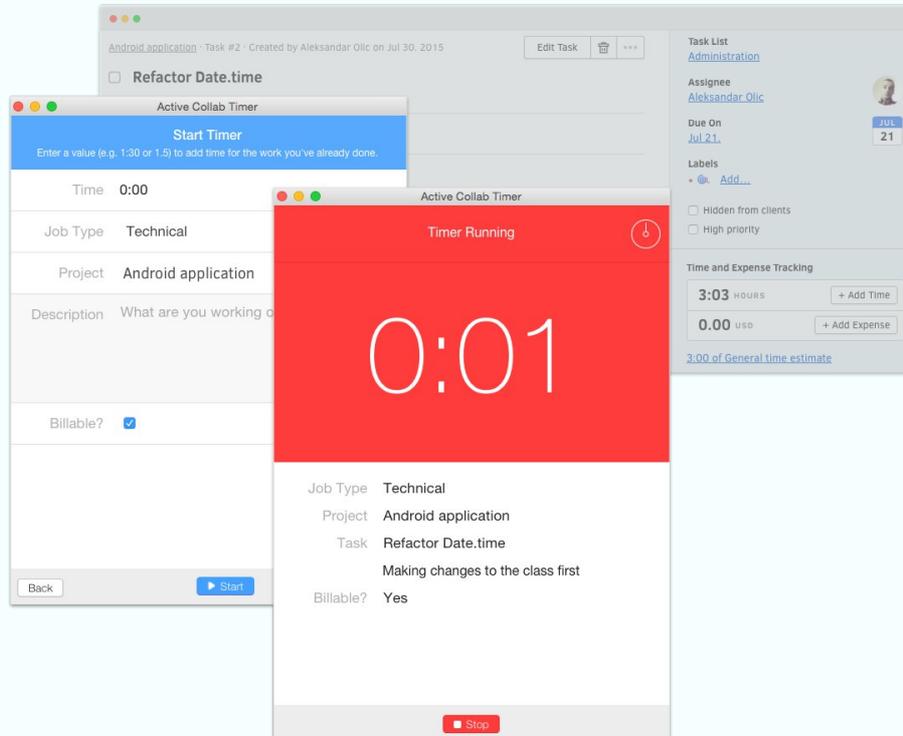
- 17:10 HOURS (+ Add Time)
- 0.00 USD (+ Add Expense)
- 160.00 of Coding time estimate
- Subscribers: Dale Knight, Jake Touchstone

Once you have a system in place and people track time, you can:

- Monitor how much of the estimated time/budget you've spent;
- Identify tasks that are taking longer than they should;
- Learn where you make bad estimates;
- Present the client with a detailed breakdown of what you worked on;
- Invoice your work in under a minute.

Agencies often spend more time working on some tasks but they can't bill for that time because they can't prove it. But once you can prove to a client how much time you've spent working, you can charge for it. If you're organized and keep track of everything, there's no chance you'll lose money on billable work.

The best way to track time is to use a timer that has all your tasks in one place and which will automatically store time records when you stop working. This saves you from administrative work like entering time records in a timesheet.



Tracking time with Active Collab Timer

In some instances, manually entering how much you've worked is faster. For example, you're in a meeting and running a timer would be inconvenient. In that case, it's easier to finish the meeting and log the time afterward. Or, if you're switching between different tasks often and don't want to bother stopping and running the timer every few minutes, just work and dedicate the last 5 minutes to add the time logs.

Getting the whole team to track time is a chore. People don't like filling out timesheets, no matter how important the activity is.

To make sure people on your team log their time, follow these rules:

- A task can't be worked on before it gets an estimate.
- Each time log has to have a comment, explaining what was worked on.
- A task can't be completed if it doesn't have logged time.
- Check each day if the time logs were entered.
- Let each team member install a timer app and use it to track time.
- Keep track if someone logs time manually and ask why they did it.

Don't incentivize entering time records or creating bonuses for people who log the most time. People will hack the system, one way or another.

Estimating Work

If you're just starting out, you'll work a lot because you can't accurately estimate how much time you need for various tasks. The key is to **make an estimate and track time against it**. Any kind of estimate is good for a start. As you work and see where you underestimate/overestimate, you get better at it. But to get better, you need data - and that comes only if you track time.

You need discipline to estimate and track time regularly. Some people don't track time because they think it "slows" them down; but spending too much time on a "quick" task hurts their income much more than the few minutes it takes to make an estimate and add time records. Never work on a task if you don't have at least a ballpark idea of how long you're going to need and then see if the estimate is correct.

Once you made an estimate, **recalibrate it as you go** so you can improve your project proposals. When you have realistic estimates and proof behind them, you'll quote and manage client proposals better.

When estimating, take the task's **complexity index** into account. A

complexity index helps you measure the chance a task will take longer than you think. If you know a task might derail your schedule, you can plan extra buffer time accordingly.

In creative work, how long something takes depends on:

- The time you need to make the thing;
- The time it takes to communicate (gather requirements, feedback, get buy-in, and other admin stuff).

It would be ideal if you could bill for admin stuff too because sometimes you **spend more time on communication** than actually creating the thing. If you accurately estimate and agree on the first part, but not the second, your project profitability drops significantly. So you need to manage both. To manage the first, know yourself and the work - to manage the second, know your client.

Creative tasks such as writing and coding can **take up as much time as you give them** (also known as Parkinson's law). To avoid procrastination, allocate a fixed amount of time for each task (using estimates) and ship what you have when the time is up. When you have immovable boundaries boxing your time, you'll be more motivated to finish the work.

Time Cheating

Hourly rates make clients nervous. For all they know, you might be fabricating time records, or working more than necessary just so you can charge more. As a result, a lot of time tracking apps have **proofing features**.

For example, some timers take screenshots of your desktop at random moments so clients can later review them to make sure you were doing work stuff. Some timers count the number of keystrokes and clicks you make, read titles of web pages you visit, or even get the first few sentences of emails you write.

That's bad for a number of reasons:

- Privacy invasion and stealing the know-how;
- Hacks and workarounds that can be easily googled;
- A person needs time to think about work and that can't be measured;
- Pressure to always look your best in screenshots and fear to take a break;

- Tasks where you don't actively participate (like video rendering) are billable too.

When you're dealing with new clients, it's best to start with small projects and **build trust**. When it's time for bigger projects, agree on:

- Manually adding time records for tasks that require thinking or research (as long as it doesn't exceed the estimate),
- Tracking time for everything else,
- Taking pauses without stopping a timer as long as the pauses don't last longer than 5 minutes.

No matter how honest you personally are, clients get cheated by others and have a good reason to be distrustful. These aren't necessarily big cheats; small cheats are way more common as a person can rationalize it and still believe they're a good person. For example, studies found that people are more prone to stealing a \$2 pen than a 1\$ bill because stealing money is harder to rationalize.

Time cheats can happen in a number of ways:

- When a deadline is approaching fast and there's no time to add time records, people tend to take the total time they were in the office, subtract two hours (one for lunch and one for dinner), and log the result, even if they worked less by taking several long pauses during the day.
- Honest people who refuse to overbill have an overall billing rate that's 20% lower than the average; as a result, when it's time for lay-off, they're the first to go - thus sending a clear message to others that honesty isn't a good thing.
- Some people are constantly "on-call" and bill every hour they spend monitoring email for a project, no matter if they receive work or not.
- Management gives out bonuses to people who log more time or penalize people who make a wrong estimate so people tweak their numbers either directly or by creating artificial work.

The best you can do is earn trust by inviting a client directly to a project so they can see for themselves where each hour went.

Being Too Efficient

On a rare occasion, you'll finish a project much quicker than you told your client. **Don't punish yourself** for that. If you nail something on the first try, there's nothing wrong with that. Maybe the first design is the winner and you don't have to create others to justify your estimate.

A client is paying you for the solution, not the time it took you to get there. The important thing is you were honest in your estimate, which was based on real numbers and your experience. It just so happened that you finished earlier.

There are several ways to deal with this happy occasion:

Charge More

Stipulate a bonus clause in your contract if you deliver before the deadline. Or raise your prices because you know you can deliver faster than others.

Do Extra Work

Delight your client by adding a few more features than they asked for, as a gift. They'll owe you for that and be more loyal. Just don't overdo it and get a client used to something extra on every project - gifts have an impact only when they're unplanned.

Sit on Work

Put away the finished work and wait a few days to show the work closer to the estimated date. This way, you'll avoid the conversation on why you're so early and make the client suspect you estimated later on purpose.

Chapter 011

Billing Work



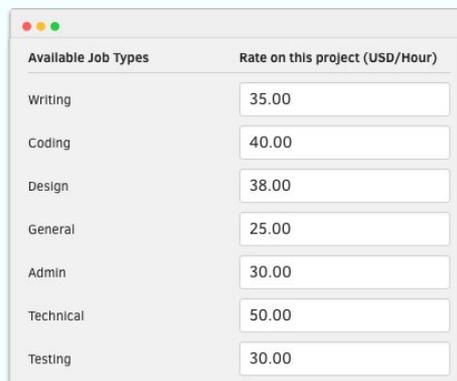
Billing is too important to leave it to your accountant. You need to bill quickly and bill often to keep money coming in at all times - the last thing you want is to miss payroll and lose employees. But to bill often, the process has to be really quick and simple. Otherwise, you'll put off of creating invoices, uninvoiced work will pile up, and when you finally do send an invoice, you'll have to wait for the client to pay it - which leads to serious cashflow problems.

How to Manage Invoices

The main trick to invoice quickly is to have all the data already in place and just press a few buttons. Every task you worked on and the time you logged should easily become an invoice item.

To get to the stage where you can invoice in under a minute, you'll first have to make sure you've planned the project and divided the work into tasks. Then, you and your team need to track time on those tasks.

Not all time records are equal though. For example, to make a landing page, you'll do designing and coding - which have different hourly rates. Each time record should specify the type of work so you can bill accordingly. Same with clients - some clients you'll charge more and others less for the same type of work, depending on your agreement.



Available Job Types	Rate on this project (USD/Hour)
Writing	35.00
Coding	40.00
Design	38.00
General	25.00
Admin	30.00
Technical	50.00
Testing	30.00

To make invoice management easier, start working on a project on either the 1st or 15th of the month. Alternatively, you can invoice every week or when you reach a predetermined milestone.

When the time comes to send an invoice for the first batch of work, see what time records you didn't invoice on the project and invoice them. To make it easier for clients to review the invoice, group the items by either task, project, or job type.

Each invoice should have:

Your logo, company name, and address

Client's company name and address

Itemized list of individual costs (item name, quantity, cost)

A note describing the invoice or clarifying something

Invoice ID number

Date of issue

Payment due date (on receipt, or 10/15/30/60 days after issue)

Optional discount

Calculated tax rate for each item

Total amount due and currency

When sending an invoice, write a **quick summary email**, explaining where the money went and what the benefit is. For example, an invoice may say "Implementing AJAX - 14 hours" but the client won't get it; explain it in simple terms, eg. the site will load faster. Also, reassure them that everything is going well and as planned. They can't understand your work and check up on you, so all they want to know is if everything is ok.

When you create an invoice, send it as a PDF by email to your client. If the client forgets to pay it after the due date, set up an automatic email reminder to apply constant (but reasonable) pressure.

If the client misses a payment, immediately stop working. It's better to be safe than sorry when it comes to money. This is why, during contract negotiation, your goal is to specify the payment schedule and the payment due date. The earlier you catch a non-paying customer, the sooner you can take action and not bury yourself in more work, hoping they'll pay eventually.

Overdue Reminders

Send first reminder after:
1 day

Then repeat every:
2 days

Reminder message
This is an automated reminder that your invoice is overdue. Please make a payment at your earliest convenience.

The addressed invoice will be attached to these reminders.

If you can, go for convenience and let your clients pay in one click by setting up **online payments**. You can receive payments on your account using any of the online payment processing companies (like PayPal, Stripe, AuthorizeNet, and Braintree).

Use **recurring invoices** for clients that keep you on a retainer. Simply creating a standard invoice and setting it to go off automatically each week/month/quarter is enough to save you a few hours and make you more productive.

Keep track of all your invoices and their status levels. When you have multiple clients, it's easy to forget who paid what or how much. Keep invoices visually recognizable and organized (know which are due, unsent, or paid partially) to know if you need to take action.

The screenshot displays a web application interface for managing invoices. It is organized into four sections: Unsent, Sent, Partially paid, and Recently paid. Each section contains a table with columns for Invoice Number, Client, Due Date, Amount, Left to Pay, and Status. The 'Unsent' section shows one invoice (433) for Crytech with a due date of Apr 13, 2016, for 2,700.00 USD. The 'Sent' section shows two invoices: 6 for A51 (12,190.00 USD) and 434 for Crytech (500.00 USD). The 'Partially paid' section shows two invoices: 10 for A51 (11.75 USD) and 9 for A51 (1.18 USD). The 'Recently paid' section shows two invoices: 432 for A51 (474.41 EUR) and 2 for Eric's Used Cars (343.00 USD). Each row includes a 'LEFT TO PAY' column and a 'STATUS' column with a corresponding button (UNSENT, SENT, PARTIALLY PAID, or PAID) and a three-dot menu icon.

Unsent					
INVOICE NUMBER	CLIENT	DUE DATE	AMOUNT	LEFT TO PAY	STATUS
433	Crytech	Apr 13, 2016	2,700.00 USD	2,700.00 USD	UNSENT

Sent					
INVOICE NUMBER	CLIENT	DUE DATE	AMOUNT	LEFT TO PAY	STATUS
6	A51	Jul 23, 2015	12,190.00 USD	12,190.00 USD	SENT
434	Crytech	Apr 14, 2016	500.00 USD	500.00 USD	SENT

Partially paid					
INVOICE NUMBER	CLIENT	DUE DATE	AMOUNT	LEFT TO PAY	STATUS
10	A51	Sep 18, 2015	11.75 USD	6.00 USD	PARTIALLY PAID
9	A51	Aug 28, 2015	1.18 USD	0.10 USD	PARTIALLY PAID

Recently paid					
INVOICE NUMBER	CLIENT	DUE DATE	AMOUNT	LEFT TO PAY	STATUS
432	A51	Sep 19, 2015	474.41 EUR	0.00 EUR	PAID
2	Eric's Used Cars	Jul 20, 2015	343.00 USD	0.00 USD	PAID

Special Invoicing Items

You'll have expenses on projects that don't relate to your services. These things can include hosting, domain registration, stock photography, etc. Before you buy anything, double-check with the client and use their money only after they approve.

If you pay for the expenses using your own money and get reimbursed through the invoice, make sure to visually separate the expenses from hourly rates. You don't want your client to see a huge bill and think they paid you more than agreed.

Think about whether you'll include the **project manager's and sale agent's fee** in the invoice, or absorb it through the profit the company makes. A project manager can spend a lot of time communicating with a client, but they don't get to log their time and include it in the invoice; still, it's a project expense.

Not all clients are the same. Some are wonderful to work with and some are a nightmare. You have the power to reward good clients by giving them an **angel discount** and punish difficult ones by including a **jerk tax** (only it won't be called like that on the invoice).

A client earns the angel discount by being polite, providing feedback on time, paying invoices dutifully, and generally being a nice human being. Grant them a 5-10% discount or do something more than you promised.

Good clients are hard to come by, so give them a little extra service. **Care about them a little more than you have to.** It'll leave a strong impression and make it more likely they'll come again - good clients are

Online accounting software (like QuickBooks and Xero) is great for both bookkeeping and billing. The best thing is, you can manage your tasks and time records in Active Collab and send them to QuickBooks and Xero for further manipulation. This will save your team and your accountant a lot of time because they don't have to update each other. You just create an invoice in Active Collab from the tracked time and let accounting software take care of it in their accounting app.

worth it if it means you get to work with them again.

Keeping an Eye on the Budget

During the project kickoff, you agreed on the budget with your client. The problem is, you sometimes run out before you finish the project (due to scope creep or underestimating time and expenses).

To prevent going over budget (or at least spot the problem early on to take action), you need to track the budget on a project-by-project basis.

You should get in the habit of **updating your client weekly** on the budget

$$\text{Remaining Budget} = \frac{\text{Total Budget}}{\sum (\text{Logged Time per Task} \times \text{Its Hourly Rate}) + \text{Expenses}} \times 100\%$$

Spent so far

JOB TYPE	HOURLY RATE	BILLABLE HOURS	COST SO FAR
General	100.00	35:07	3,511.00
Design	60.00	5:00	300.00
UX	65.00	0:00	0.00
Backend	80.00	0:00	0.00
Frontend	75.00	5:00	375.00
Writing	55.00	6:00	330.00
Other Expenses	-	-	1,560.00

How much budget you've spent, as calculated by Active Collab

If you spent more in the first phase than you initially planned, you need to know why that is. The best way to find out is to see the **estimated vs actual time tracked** breakdown for each task. By comparing the estimated time, spent time, and the % done, you'll get an idea on whether you'll be on time and on budget. Then you can see if certain tasks took more time than planned, ask why, and talk with your client with actual data at hand.

Estimated vs. Tracked Time

Everyone ▾ All Projects ▾ All Open Tasks ▾

Alex Ryerson

TASK	PROJECT	TRACKED TIME	ESTIMATED	% OF ESTIMATE
Homepage	BikeShop Website	7	5.00	140 %
Sell your bike page	BikeShop Website	15.07	20.00	75.35 %
Analytics	BikeShop Website	20	20.00	100 %
KW analysis	SEO eHouse	7	10.00	70 %

Dale Q. Anderson

TASK	PROJECT	TRACKED TIME	ESTIMATED	% OF ESTIMATE
Backlinks	SEO eHouse	12	5.00	240 %

Dale Knight

TASK	PROJECT	TRACKED TIME	ESTIMATED	% OF ESTIMATE
Install Database (WP)	Website Design	8	7.00	114.29 %

status. This way, you save both from having an awkward conversation when something goes wrong or catching them by surprise and making them wonder where the money went. If they're updated constantly, they'll know where the budget went and when it's time to talk about the best course of action.

Don't sweep problems under the rug, hoping the team will catch up. Be upfront and make things right before the situation goes out of hand - it's much safer than trying to correct the mistake for which the client will ultimately pay for.

Chapter 012

Beyond Projects



You need to think where you want to be, beyond your current projects, and then build a philosophy that supports that vision. Once you have the philosophy, invest in people and let them improve the processes so you get to where you want to be.

Why Businesses Fall Apart

Most agencies never become big. In fact, an agency has a greater chance of falling apart than becoming a well-known brand. There are several reasons to this:

High Turnover

There's a great demand for technical talent in today's market, which makes switching to a better job extremely easy. Also, when a developer/designer gets tired of working for someone else, being paid peanuts, and dealing with bad management - they start an agency of their own; and the cycle repeats.

Unprofitability

Agencies lose money on projects because they work more than they charge due to bad organization, last minute changes, scope creep, subpar processes, etc. A badly run agency can last for a surprisingly long time before it dissolves (or gets acquired by a bigger agency) and the founders start another company (hoping they'll get it right this time).

Deteriorating Quality

Technology is constantly changing and agencies who are initially agile don't stay that way for long. They don't innovate and get better. As a result, they get complacent and become happy with churning out the same work again and again until they become a shell of their former self. They slowly lose clients as they stop doing the things that got them hired in the past - which are great ideas and execution.

Project Myopia

A typical agency doesn't have the time to think and develop a long-term strategy because it's consumed by everyday activities. The teams are constantly in crunch, catching up with deadlines and work. And the work is always given priority because reflection isn't a billable activity. So they keep taking new projects instead of taking a time-out, catching their breath, and checking if they're not heading over a cliff.

Too Many Layers

The more an agency grows, the more managers it hires. As a result, processes get sluggish and costs increase; suddenly, there are several communication layers between clients and teams, meetings and

interruptions abound, and nothing gets done. An agency that initially had 1 manager for 12 workers ends up with 100 people - and 1 manager for every 3 workers.

The Three Elements of Strong Businesses

Starting a business is easy because you have the enthusiasm, good ideas, and flexibility. There's no management, hiding who makes how much money, coordination, or rules. It's just a few people trying to make it on their own, pouring their heart and soul into the work.

But growth means hiring more people and taking on more serious projects, where the chance of tripping up rises exponentially. A CEO doesn't have the time to manage and bond with new people like they used to because it's not scalable, so they delegate it to a manager. But new managers don't share the same values and can't pass on the spirit of the organization as well as the founder.

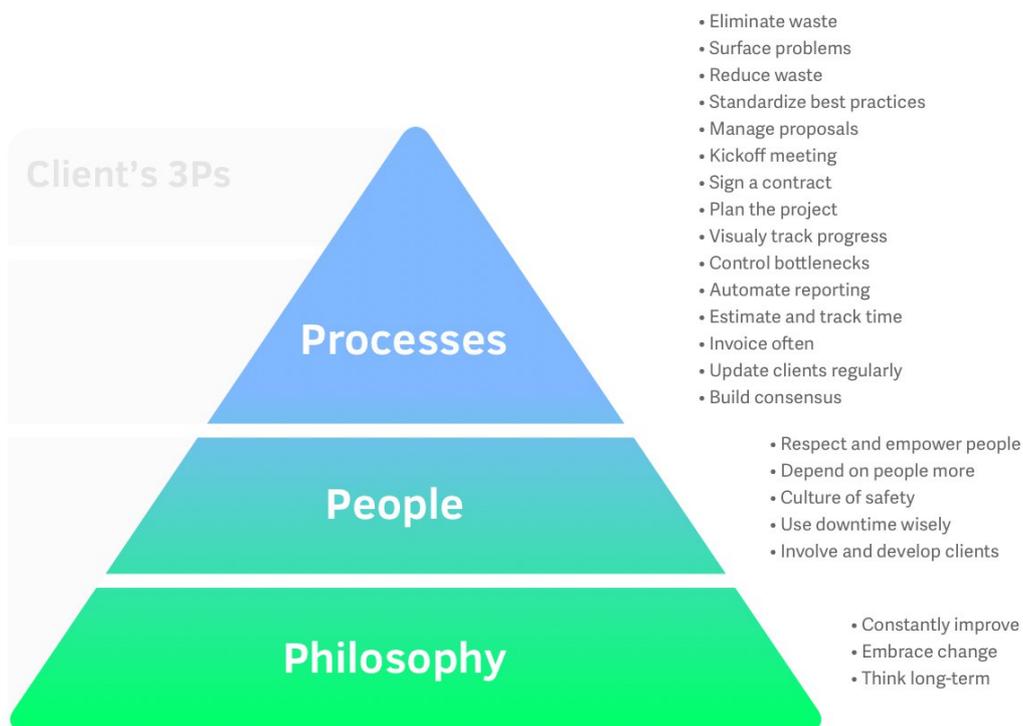
As a result, employees treat the agency like just another job, a way to pay their bills. They'll move on in a few months so they don't care about improving the workplace. On the other hand, the management doesn't care about long-term impact because they're pressed by deadlines and budgets.

So somewhere in the process, the agency begins to rot. They hire a consultant and implement agile or lean or some trademark framework, hoping it'll solve everything. The initial enthusiasm gives everyone a short productivity boost, but that efficiency is soon lost as they go back to their old decadent ways.

The whole problem with introducing agile processes and techniques is that they don't address the root - the underlying philosophy and the people. It takes more to be great than using Kanban boards, limiting work in progress, writing a vision statement, or hiring the best talent.

To have a strong and healthy business, you need 3Ps:

- Philosophy
- People
- Processes



Improvement Philosophy

At the base of the pyramid is the philosophy that inspires everything you do. It's the foundation on which you base how you work and think. People and processes are built on top of the philosophy and they are its visible manifestation.

Truly great companies are **constantly trying to improve** in all things, no matter how small they seem. When you reward someone because of a tiny improvement, you send a clear message that no improvement is too small, which in turn leads to bigger ones.

For example, after a project ends, take the time to reflect on what happened and what can be better. Each project should make you better, smarter, faster, and more efficient. Then, as you gain more experience and excellence, you can adjust your hourly rates. After all, you should charge more if you can deliver work of higher quality and faster than when you first started. If you don't change the pricing every few projects, you're not growing.

Having a philosophy of continual improvement doesn't mean implementing Scrum, rapid prototyping, or some other buzz word. Every innovation is doomed if your people are discouraged to think and act independently, trained only to do their job, and not make waves.

Most agencies for example can't pull off of rapid prototyping because their culture isn't built that way, both structurally and emotionally. They can't put out a prototype and tell their clients:

"We know it's not perfect but tell us what doesn't work and we'll fix it".

No client will do that, and more importantly, employees are too scared to risk losing their job. The culture simply isn't there to support the process, no matter how good it may seem on paper. Rapid prototyping works in software startups, but not in agencies because of a fundamental philosophy mismatch.

Innovation takes time and nurturing, something that's foreign to project-focused agencies. You need to think about the **long-term plan**, even if it means sacrificing short term goals. You need to think about your strategic market position and why someone would hire you. But building a brand takes more than that - you need long-term thinking to get away from the clutter.

Long-term planning is so neglected that most organizations hope project managers can fill in as a strategist. This is a serious mistake. A project manager only cares about projects - what happens to the rest of the universe doesn't matter as long as projects come on time and budget. They simply don't have the time or the energy to think about a bigger picture. When it hits the agency that they need change, hiring a strategist will be too late: it'll be like calling in the fire-prevention officer when the building is already ash and smoke.

Agencies start out with a lot of expertise, but they slowly lose that expertise because they don't keep up with the speed of change and new technology. They're too busy monetizing their existing knowledge and so they don't grow - a typical example of short-term thinking.

You need to base your management decisions on a lasting philosophy, even at the expense of short-term financial gains. If you want to be in business 10 years from now, you need to **embrace change**: your goals, market position, capabilities, even specialization - they will all change.

The most dangerous time for any organization is not when it starts

falling apart but when it gets successful. When business is booming, companies become complacent and the team doesn't feel the urgency to become better or push themselves harder. They go on autopilot and stop growing. As a result, the quality erodes.

You need to take every opportunity to stir the pot, question your success, and remind yourself you're mortal in order to keep the company vigilant and ready for change. Never tell things are going well; always keep working and innovating like there are millions of others who have 24 hours each day to take everything away from you - because there are.

A philosophy can't be created in a month, or even several years; there's no rule or a memo that can make your employees instantly loyal, hard-working, and innovative. Instead, it's like a seed you plant while you're young that, with enough care and time, grows into a mighty tree that bears fruits for many generations.

You build a philosophy every day with everything you do. The more authority you have in an organization, the more you can influence its culture. Every time you say "thanks", "good job", or "it needs to be better", you're building a culture and steering the company in a certain direction.

To expect excellence, you need to be excellent yourself, walk the walk, and empower people to do the same.

For example, if you're pressed by a deadline and cut corners, your team will notice and think it's ok; it's a slippery slope and every action counts. If, on the other hand, you notice a defect, anticipate the client's complaint, and delay the shipment to fix it because you know it's not right to disappoint the client - you show others how important customer satisfaction really is.

Amazon is famous for their cheapness, but they have a good story behind it. Their office needed a table, but instead of buying a new one, they improvised by taking a door and laying it down so it served as the table. Instead of spending money on their own needs, they used the situation to teach their employees what Amazon is really about .

Each new employee is shown the table to illustrate the importance of cost-cutting because each penny saved can be passed on to the customer so they can get the best price - even if it means dining on improvised tables.

Empowered People

The philosophy is the underlying foundation of your company but the people are its physical manifestation. They are more important than your processes because they make and improve processes and they're flexible and can learn and grow beyond what they're doing right now.

Digital agencies are not like classic businesses where the talent is concentrated at the top; in agencies, the talent is concentrated in the middle to lower parts of the organization, as those people actually deliver the work.

An organization is nothing without its people as they bring the system and values to life by working, communicating, resolving issues, and growing together. They should be the main engine behind improvements as they're the ones in the trenches every day and know the process better than the management.

Project managers and other executive coordinators aren't there only to guide the company's development but the development of the people and allowing them to do business the way they know it should be done. Management should encourage, support, and demand employee involvement.

Most companies make the mistake of treating employees like cogs that can easily be replaced. But you should **depend more on people**, not less. They're the ones who design new things, come up with ideas, identify hidden workflow problems, and fix them. If people rely on a supreme wisdom of the all-powerful process and are met with resistance when they suggest improvements, don't later wonder why you get less and less business.

Before you can build supreme digital solutions, you need to build people. Everyone should know how to work effectively in small groups, solve problems, improve processes, collect and analyze data, and self-manage. Both decision and proposal making should be offloaded to workers so it's their responsibility to discuss suggestions and arrive at a consensus before implementing any decision.

A culture that promotes improvements should also encourage **admitting a mistake** and making amends. If people are conditioned to take criticism as a sign of weakness, they'll avoid speaking up in order to avoid blame or sounding stupid. In fact, the number one reason that

makes a team successful is a chance to be vulnerable in front of others without fear of repercussions.

You should praise people who openly address things that went wrong, take responsibility, and devise a plan to prevent these things from happening again. Instead of breeding a culture of passing the buck, you should encourage and reward people who try to improve the organization.

Project managers believe that people have to be fully utilized and no person should ever run out of work. This may satisfy a typical 20th-century management accounting practice and look efficient, but it impedes the development of an improving culture.

If people are working all the time, they don't have the time or the energy to think about improving the processes - all they want is to finish their tasks on time and go home. They know that if they make a suggestion, they'll get stuck with meetings, bureaucracy, and fighting with authority. Why would they do that when they can do what they're told, collect the paycheck, and move on to the next job when they get a better offer?

Don't be afraid of downtime as people can use that time to learn new things and improve the organization in the long run. Google has a rule that makes it ok for people to spend 20% of their time on other activities. Most people don't use it, but it's the idea that's important. A 100% (even 60%) productivity is a myth so don't base management decisions around total productivity or it'll backfire and hurt you without you noticing it.

In order to really increase productivity, focus on:

- Making the environment free of noise and distractions,
- Limiting the size and frequency of meetings,
- Keeping the nexus of activity online so all the information is readily available and there are no interruptions,
- Letting people work alone or on a flexible schedule when they need,
- Empowering people to make decisions on their own, have autonomy over their work, and constantly learn new things.

This focus on people includes your clients as well. You need to involve them on projects and work together. They should be your biggest collaborator. You need to update them on progress so you can solve problems together on time.

While working with you, clients should also pick up your philosophy of continual improvement. Learn how you can become better, but also use every chance you have to teach the client something new. They don't know about your field of work as much as you do - instead of viewing it as a hindrance, view it as a chance to make them a bit smarter than they were before they hired you.

For example, a client may complain that the website isn't pretty enough. Instead of making it prettier, explain that conversion is much more important than the visual look, which often only makes a site more difficult to use. It's best if you can teach them subtly throughout the project and guide their feedback so they end up feeling like they came up with the solution.

Lean Processes

At the top of the pyramid are standardized processes you use. They are designed to help your people be more productive and get things done, while leaving room for continuous improvement.

A good agency has a number of processes that are liable to improvement:

- Screen potential clients, making sure you match their needs.
- Give a rough proposal before arranging a meeting.
- Have a kickoff meeting to precisely define the project scope and payment schedule.
- Sign a contract and a statement of work before you start working.
- Plan the project by breaking down work into tasks, with deadlines and assignees.
- Use a Kanban board or some other visual management tool to track progress.
- Watch for bottlenecks and limit work-in-progress.
- Automate reports so people don't waste time creating them.
- Estimate and track time you spend on tasks.
- Invoice often and quickly.
- Involve clients and update them regularly.
- Build consensus with stakeholders.

When looking to implement a new process, you may come across a system like Kanban that may seem inappropriate because it's too elaborate for your small team. But you have to consider the hidden perks that go beyond the traditional cost-benefit analysis. Take into the account how the process will shape the people's minds. The new system may intrigue your team, get them interested in improving client

collaboration, and ultimately find new, better ways to collaborate. This is why the underlying philosophy is so important to processes - in unleashes people's creativity and inclination to innovate .

When someone makes an improvement, **standardize** it as a best practice and make it an official part of your processes. For example, a designer may come up with a good workflow for getting feedback; instead of passing the know-how ad hoc, standardize the process so others can use and build upon it. Don't reinvent the wheel with each new project but take something that works and build on it. If you don't standardize, the knowledge will pass into oblivion.

Processes should aim to **reduce waste** (waste is any activity that doesn't provide value or contribute to the client's satisfaction). So how to distinguish the value-added work from waste? Consider a developer who's busy coding, reading the specifications, and having meetings with the product owner; are they doing value-added work? You can't know by simply looking at what he does.

You have to follow the progress of the actual software they make. When you start analyzing each step of the process, you'll learn that they make a huge amount of information, some of which doesn't impact the final product at all (eg. progress reports or arbitrary documentation). This waste should be removed.

Time-consuming emails, lengthy reports filled with technical descriptions, and information overload - they're all waste; same goes for lengthy meetings (especially ones that are about sharing information everyone knows and there's no conclusion).

Identify activities that add value and get rid of everything else. If the customer isn't willing to pay for the work, drop it. For example, don't create assets or illustrations that don't end up in production, unless they're used as a step needed to achieve the final product.

Don't have robust processes that minimize the impact of potential problems. You want the problems to surface quickly so people have to deal with them immediately or they'll sink. This way, everyone is motivated to fix the problems and inefficiencies or they can't work. The best time to make an improvement is immediately.

When you put fixing an important but not urgent problem on your to-do list, you'll never get around addressing it and the technical/organizational debt will pile up. More importantly, by hiding

vast inefficiencies, people will just assume a process typically takes days or weeks to complete and that it's normal. It's not. With a good process, the same thing can be accomplished much quicker. So don't bullet-proof yourself against problems because burning problems are resolved quicker than the ones under your rug.

The Complete Guide to Managing Digital Projects

Writing and design: Aleksandar Olić

Editing: Miha Ambrož

Publisher: Active Collab

Date: Aug. 05, 2016

Edition: First

Words: 33,630

Language: English

ISBN: 978-1-370-84975-8

Primary Category: Nonfiction » Business & Economics » Project management

Secondary Category: Nonfiction » Computers & Internet » Software development & engineering / project management

Copyright © 2016 by Active Collab

All rights reserved. No part of this publication may be used in any form or by any means without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law. For permission requests, write an email to the publisher at marketing@activecollab.com.

The author and publisher have extensively researched the topic and prepared this publication with great care, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information contained in this publication.



Active Collab is a powerful, yet simple project management software. It helps your team stay organized when you outgrow email and offers a one-stop solution for all your business needs.

Active Collab gives you an overview of your team's activity across projects and bring clients on board to collaborate more closely. With it, you can delegate tasks to your team, keep information in one place, estimate and track time, and issue invoices.

For more than 10 years, over 200.000 people have used Active Collab, ranging from small businesses to Fortune 500 members, universities and government institutions.

activecollab.com

