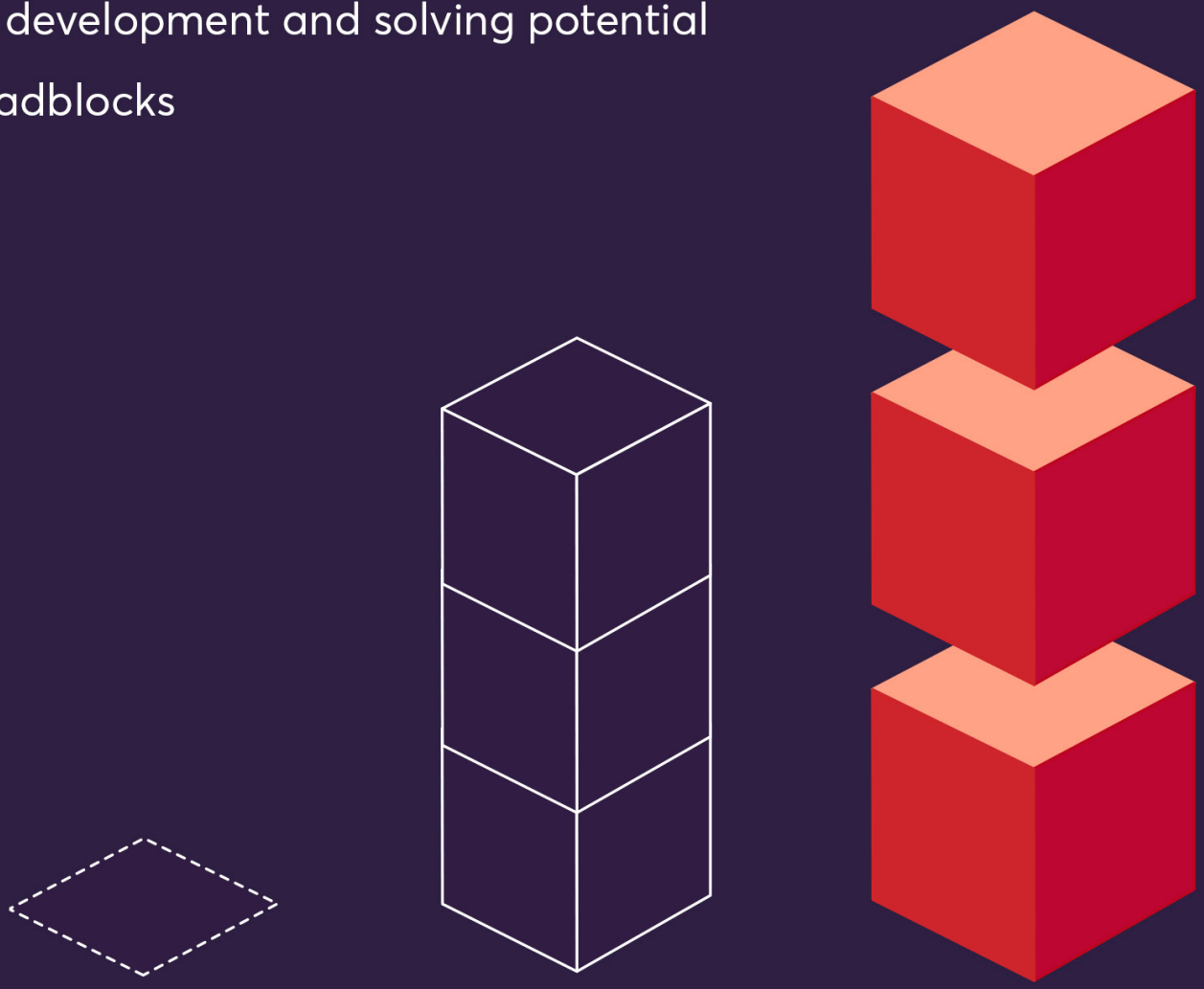
From Concept To Completion

A Full Agile Development Walkthrough

From project idea, through planning, to development and solving potential roadblocks



From Concept To Completion - A Full Agile Development Walkthrough

Take the full trip from coming up with the idea for a project, through planning and initiation, all the way to development and potential fixes to development roadblocks.

Introduction

This is a guide for entrepreneurs in the early stages of the software product development process. Whether you're just starting out with an idea for a project or have already begun development but found yourself facing unexpected obstacles, this ebook will help you take appropriate next steps.

You will learn:

- How much planning should go into developing a product.
- How to approach an external development agency and prepare for the initial meeting.
- What the 7 phases of starting a new project are.
- How to set up a team.
- What it means to hire freelancers instead of using an agency or hiring your own employees.
- How to set up effective team communication.
- How to create user stories for your product.
- What the essential elements of design are.
- How to map out and manage the development process.
- What the most crucial elements of the development process are.
- What pitfalls to avoid.
- What to do when you need to start over with the development process.
- How long it takes to develop a product.

And more!

We hope you'll enjoy your read, and we would like to encourage you to give us any feedback you might have or ask any questions that come to mind. We're always happy to chat.

Editors: Aleksandra Prejs, Olga Trąd
Authors: Aleksandra Prejs, Olga Trąd, John Waldron, Radek Zaleski

What to Expect From The Application Development Process

Whether you have a great idea for an app, or you've gone so far as to create a design for your idea, you probably have some questions about the development process when working with a development team outside your organisation.

If you have never created an app before, the process can be confusing and overwhelming, especially if you come from a non-technical background. In this guide, you will learn the process of creating an app from concept to completion, and at the end, you will have a stronger understanding of what you can expect from your outsourced, professional development team.

Agile Math = Multiple Levels of Planning

Projects start from a simple mathematical rule: time, cost, and scope. Agilists define the terms as follows:

Time is the amount of time required for finishing the project. It is perceived in the context of the deadlines resulting from the marketing strategy, budgeting or related events. **Cost** is the budget that the team receives for the project. **Scope** is the part of the project that includes the work that needs to be done in order for the project to be delivered.

Planning development in Agile is an undervalued task. Agile is all about making quick adjustments to plans and goals for the project, which can't be achieved in a chaotic environment. You can read more about the methodology here.

Setting and revising goals is a crucial part of the project. Agilists are able to achieve this by planning at various levels: project, release, iteration, and daily planning.

- Project planning is the initial plan for the whole project, which includes the product vision statement.

- Release planning determines the project's scope, schedule, and resources. This type of planning occurs at the beginning of the project, and it should be updated regularly as the project develops.

- Iteration planning occurs at the beginning of every iteration (e.g. a week-long sprint). This plan is based on the tasks accomplished during the previous iteration. The team prioritise the tasks and schedule the plan for the next iteration. Iteration planning also includes previous iteration review.

- Daily planning is used for coordinating the work. During daily stand-up meetings, teams assess their tasks and revise their plans, focusing mainly on discussing the present tasks. Daily planning focuses on coordinating individual activities of team members.

The project initiates with a set of objectives. It is crucial to clarify and understand the relationship between the team's definition of completed tasks, and your conditions of satisfaction for every single task in the backlog. Conditions of satisfaction are the criteria according to which the project's success is assessed. Tasks done must fulfill a special set of conditions of satisfaction that should be added to every task in the backlog. Conditions of satisfaction must always be verified at the very start of the project and then subsequently monitored.

What You Need For Your First Scoping Session

After you find a development team and sign on the dotted line, you're ready to begin the process of creating your application. Since you trust a group of new people to take your idea and turn it into reality, the best thing you can do is to come to your first meeting with as much information as you can.

Your first scoping session is your chance to meet with the team. Each company handles this part of the process differently. At Netguru, our formal scoping meetings include a project manager who will oversee the project, and the software developers who will be working on it.

You'll learn a bit about their background and experience and hear what each person's role will be on your project.

Then, it will be your turn to present your idea to the group.

You might not have the idea completely fleshed out, but the more detail you have about what you are looking for, the better.

What You Should Prepare

You don't need a whole briefcase full of printed handouts or a 100-slide-long presentation. You can bring those materials with you, of course, but you don't have to. Before your scoping session, simply ask yourself the following:

- What problem will my app solve?
- Who is my target user?
- Are there other applications on the market that do this?
- If yes, what are their strengths and weaknesses?
- What are the apps I like using? What do I like about them?
- What is my budget?
- When do I want this application to be complete?
- Am I looking for a mobile app (i.e. an app for smartphones or tablets) or a desktop app (i.e. an app for traditional computers)?
- Do I have designs, drawings, or mockups of my application idea? If you do, send them over for review before the scoping session. You can also bring them along.

Don't worry if you don't have all the answers. The more you prepare, the better, but your project manager and your team will be there to help you, not to judge you.

Your Vision

As a product owner, you have a vision of the final product. However, it sometimes turns out that it's impossible to carry out the development process strictly as you planned. You have to create a new actionable plan together with your development team.
In this section, we share our lessons learned over the 10 years of working with clients. We will also shed some light on the risks associated with poor communication during this step. Learn how to keep a smooth transition from ideas to outcomes.

Positive Attitude

Remember that you and your team work towards the same goal. This rule is valid both at the beginning of the process when you discuss the project's details and at the end when you're given a working product. The same applies to the whole development process. You're a part of it, which means you shouldn't try to take over, but you should feel comfortable with the level of influence you have on what is happening. Your project manager is the person who sets rules for the whole team – they will make decisions with your (project's) best interest in mind. They have the skills and experience necessary to do this, so you can be sure all the hard work and resources that go into the project won't go to waste.

Drafting up a Common Vision

To kick off your project successfully, you have to determine your product's target group and the problem it will solve, so that the

user can understand it from the very beginning. Remember that you have to highlight the benefits for the user. For example, at Netguru, the process of drafting the shared vision takes place during a free scoping session.

Together with a client we analyse their competitors and try to precisely describe the target group. This step is crucial for every IT project. If we skipped it, we would end up making an app for everyone, which makes it less useful and less likely to attract customers. An app for everyone is an app for no one.

Try to describe three types of users who will use your app and find the differences between them, determine what motivates them and what can make them happy about using the app. After this step, your every design decision will be simpler – you will avoid creating an app for yourself and let your team design for the actual end users. It's too late to determine your target group when the project has already kicked off. If something goes wrong, you risk a failed product and wasting your money on developing features your app's real users don't need in the first place.

Another very important aspect here is your team's motivation. If the client doesn't have a very precise vision of a product, the team won't feel motivated because they feel they won't be proud of the final product. Remember that your team don't develop simply what you dream of. They always have in mind the perspective of the users, and that's why your views and your team's views on various topics may differ. And this is a good thing! Your team will help you see your project from several different perspectives.

The Most Important Thing You Need For A Scoping Session

The most important thing you can bring to the scoping session as a client is an open mind.

Remember, you will be meeting with experienced developers. They will have ideas on how to simplify your idea, make it better, make it more palatable to users, and how to save you some money along the way.

An open mind and a bit of flexibility will help you achieve your ultimate goal.

What To Expect From Your Development Team During Scoping

The purpose of a scoping session is to flesh out your idea and move it towards a solid concept. At the first meeting, the team will have a lot of questions, because they will want you to help you paint a complete picture for them. The aim is that they fully understand your concept and ultimately create the app you're looking for.

Every single scoping session is different. These meetings are personalized based on your unique situation, but every scoping session includes the following elements:

- Understanding your business: the development team will get to know your company, your industry and the business goals of your application.

- Understanding your vision: what do you want your application to be?
- Understanding your users: who is going to be seeking out and using your application?
- Understanding your commitment: your expectations in terms of timeline and budget, and how much time you can commit to meetings related to the project.

Before You Begin – The 7 Phases of a New Project

Best practices dictate that the development team already have a workflow model structured right from the moment the first meeting between client and development team commences. They should be able to help you understand the whole process. But if you'd like to take a more hands-on approach, you can involve yourself with the planning and development more deeply. Here's a guide to launching a successful project without any unnecessary fuss or stress.

Phase 1: Aligning Your Vision with the Development Team's Plan

Before any development work commences, it's absolutely crucial to ensure that everyone is working towards the same goal and has the same vision for the project. The whole team needs to understand which features will be included in the MVP (minimum viable product), and what the goal of the first release is. The vision, features, and goals can, and most likely will, change with time. That's why it's essential to define the purpose of the MVP to prepare for any future changes.

If the vision isn't yet clear, we recommend running a Product Design Sprint, in which a team of designers will help you shape and solidify your idea before you commence the development work. This way, you can avoid wasting time and resources on pushing on in the wrong direction.

At the beginning, you will be assigned a project manager and a designer. Those two will take you through a 7-day sprint involving:

- intense brainstorming,
- creating the personas of your users,
- creating the user paths of those personas,
- creating first mockups and views.

The key benefits of this process are:

- definitions and goals,
- the critical path diagram,
- the user journey,
- a prototype storyboard,
- a testable prototype,
- a basis for a further development plan.

Product Design Sprint is a tool that will give you a precise road-map of where to go next.
You will be provided with an extensive report showing how to forge your idea into a live product.
This is the first step you need to take to kickstart the development.

The technical aspects of the application will not be defined in detail, but they will be defined sufficiently enough to attract the interest of new investors and draw the funding necessary for starting the development.

We recommend Product Design Sprints to our clients and everybody else who wants to develop a software product. We have found that it solves many problems before they even have a chance to occur and speeds up the development process, while also making it more streamlined.

If this idea sounds intriguing, read more here.

The most important things to establish in the initial stages of planning are:

- **User and customer personas:** the development team need to understand exactly who is going to be using the final product. If it's a piece of bespoke office software that has been commissioned, then the development team need to know how the office employees operate and exactly what the software will be helping them achieve. If it's an app that you're developing, what is the target market?

- **What need the project will meet:** it's important that you, as a client, provide the team with any market research that you have done regarding the project to help the team understand the product's purpose.

- **How the end product will meet your needs:** in relation to the last point, it is important that the development team manage to clearly communicate that they are committed to your vision of the project. Each member could bring his/her own ideas to the table so that they become active participants in the formation of the idea.

- **SWOT analysis:** what are the strengths, weaknesses, opportunities and threats to the project? It's important that all competitors are identified, and you know how the new app will improve upon what's already available.

- **Available budget:** clients will have a maximum amount they are in a position to spend on the project, and the development house will try and create the best deal it can in accordance with that. But this needs to be clarified from the outset so that there is no chance for a disagreement later (not unheard of when dealing with money). Also, check whether the development house of your choice uses fixed prices or daily rates – at Netguru, we prefer the latter.

- **Any time constraints:** when would you like the final product to be completed? Does the development team consider that to be realistic and will they be able to deliver in this time frame?

Phase 2: Drawing up a Software Requirements Specification (SRS)

A Software Requirements Specification describes your product and contains a list of requirements for it. The goal of your SRS is not only to create a very clear and concise document, which everyone is up-to-date with and able to adhere to, but, more importantly, to assist in the creation of great software.

During Phase 1, you will have managed to outline your main goals with your development team. Now, you can create a general specification.

This should address:

- What the software under development is required to do.
- How the external interfaces of the software function.
- The expected performance of the software.
- A product feature list.
- Attributes, i.e. security, maintainability and portability considerations.
- Constraints, i.e. whether or not there are any required standards in effect, policies for database integrity, limits on resources, etc.

A good SRS will be an excellent resource for the development team, and will establish a strong basis of agreement between the developer and the client before the work begins.

The specification will do so much more than that. It will form the basis for forecasting the costs and timelines and considerably reduce the later efforts in development. The drawing up of an SRS will compel the development team to sketch out all software requirements rigorously, thus reducing the need for redesigning and recoding later.

'Sketching out' ideas in the specifications phase may be taken literally. Much like creating storyboards before taking a screenplay to a film, developers should physically sketch out how they envision the interface of the software to look. You can do it by using a mockup tool such as Sketch or Balsamiq, or you can go the good old-fashioned way with pencil and paper.

Written descriptions of each module are also invaluable. And to this effect, creating effective user stories is invaluable. Put simply, user stories describe exactly what the customer wants from each module.

Phase 3: Giving and Receiving Feedback

Before the project progresses any further, it is imperative at this point for you as a client to go and discuss the project specifications with all those who are involved in the project. This may include your employees who might be using the software on a daily basis in the office or the potential users of your new app.

By taking the time to do this now, you can save many hours in the future by preventing the development team from starting off in the wrong direction or from not including an essential functionality of the app.

Phase 4: An Interview With the Project Manager

Your project is now moving closer and closer to getting underway. But, before this happens, it's important that you have an in-depth meeting with the PM of the development team. The interview will give you a chance to clear up any anxieties that you may still harbour with regards to communication and processes.

Here are the 10 most important questions to ask your project manager before a new IT project gets underway:

- How should I communicate my ideas?
- Which communication tools are we using?
- What is the structure of the communication flow?
- What type of process model will be used?
- How will the workflow be managed? How will we be alerted if we are behind schedule or if unforeseen problems arise?
- Who is responsible for the project?
- Can all of my requirements be met?

- How are project priorities translated into iterations, estimations, and processes?
- How will the final product be distributed?
- How can I help and support you throughout the development process?

After taking questions from the client, the PM will then have some questions himself/herself, in order to ensure that the client's requirements can be fulfilled.

These questions will include:

- Does the client have his/her own servers already?
- Does the client have any preferred delivery or deployment tools?
- Does the client have a technical person on his/her side, a CTO (chief technical officer) for instance?

Phase 5: Preparing for the Kick-off Meeting

Before a project starts, there's always a kick-off meeting. But, even before this, there are a few things that the development team will need to have prepared.

This will include:

- Information on whether the development team will be

conducting all of the work, or if they will be sharing some part of the project with another team. Should the latter apply, then the client may have some concerns about how the workflow between the in-house and remote teams will work, and how any issues related to working with remote teams are solved (see 4 Common Concerns About Remote Teams - Solved).

- All communication channels to be set up. At Netguru, we use a combination of daily email updates, weekly calls, and various project management apps and tools.

- The first draft of the app architecture, which will be discussed over with you during the kick-off meeting, or during the first days of collaboration.

Phase 6: The Kick-off Meeting

The final stage before development begins is the kick-off meeting. Here's what will happen:

- The team will be introduced, and all of their roles will be outlined – the developer(s) in charge of coding; the QA team in charge of testing and noting bugs; the project manager in charge of planning and communication with the client.

- The client will be asked how s/he plans to earn money from the application.

- The first drafts of the app's architecture will be discussed, and any necessary changes will be made.
- The communication tools will be outlined for all team members and the client (see 5 Communication Hacks for Better Business Relationships).
- A staging server will be set up with access to it granted to the client and relevant team members. The client will then be asked if s/he has a production server. If the client doesn't have one, then the development team should provide one.

Phase 7: Go!

Everything should now be in place for work to begin. In the initial stages of collaboration, milestones will need to be set with iterations laid out that all devs are to follow consistently. It will also be at this early stage that everyone will have a clearer idea whether or not they have all the materials and information they require to complete the project. If anything is discovered to be lacking, it can be sorted out now so that everything is ready when the time comes.

Setting up Your Team

The Team You'll Need

Software development can be seen as a team "sport". You won't be able to develop your product all by yourself. You need a number of people behind you who will be responsible for different areas of your development process. Take a look at who a perfect development team would consist of and what skills they would need.

The Perfect Team for Your Development Game

Visionaire

It's probably you, the founder, or the CEO or CTO of the company, but sometimes a product manager or the product owner may perform this function.

Deliverables: a well-written document with a product idea, initial designs or mockups, and user stories.

Obligatory skills:

- Excellent communication skills
- Ability to create business models
- Perfect understanding of the product idea, its strategic goals and the way it will be monetised in the future
- Outstanding grasp of the problems the product solves
- Strong decision-making skills

Nice-to-have skills:

- Ability to create an app wireframe
- Product designer

A person who is able to turn the founder's idea into prototypes and designs.

Deliverables: a product prototype (including high-fidelity mock-ups, market research, user stories and more).

Obligatory skills:

- Ability to solve users' problems
- Ability to design user-friendly interfaces
- Good understanding of user experience
- Eye for detail, good taste
- Ability to ask questions to users and business owners and turn the acquired knowledge into new features of the interface
- Ability to come up with user stories
- Ability to conduct A/B tests of proposed solutions
- Knowledge of web analytics necessary to optimise the product

Nice-to-have skills:

- Ability to tinker with frontend code and mockup

Project manager

A person who makes sure that the project proceeds smoothly and keeps every team member in the communication loop. The project's informal leader.

Deliverables: division of tasks, schedule monitoring, defining priorities.

Obligatory skills:

- Outstanding organisational skills
- Ability to effectively communicate via online channels such as e-mail and Slack
- Knowledge of business and technology jargon
- Knowledge of project management tools such as Jira
- Ability to monitor the budget and keep it under control
- Ability to set the scope and choose a team for the project
- Charisma and ability to persuade and motivate

Nice-to-have skills:

- Knowledge of data management tools

Developers

A team of people responsible for turning images into a working software.

Deliverables: high-quality and well-documented code.

Obligatory skills:

- Responsibility and sensibility
- Resourcefulness
- Ability to effectively communicate with the project team and the client
- Good understanding of the business goals of the project
- Ability to spot technical risks within the scope of the project
- Ability to suggest compromises to work around technological limitations
- Ability to look for already existing solutions and use them to reduce costs and speed up the project
- Knowledge of open-source projects and various development platforms

Nice-to-have skills:

- Lots of experience is not always needed. What you need

instead is also the perspective of less experienced programmers who can have a lot of abilities unrelated to programming itself, e.g. strong communication skills or the understanding of client's business.

Quality assurance specialist

A person whose work may seem invisible, but if a QA Specialist does not pull his/her weight, the effect becomes striking for everyone, because of the number of bugs soars. QA Specialists know the project better than anyone involved in the development process. They check every single feature multiple times and create documentation on how to test them after the following iterations.

Deliverables: accepted or rejected tickets (tasks tracked within your project management software) with detailed explanations.

Obligatory skills:

- Ability to describe bugs and show how they occurred
- Ability to automate recurring actions
- Ability to spot bugs undetected by automatic tests
- Precision and eye for detail
- Knowledge of tools which facilitate the testing process
- Ability to write own automation tests
- Analytical thinking and ability to break bugs down into easily analysable parts

Nice-to-have skills:

- Courage to speak their mind – a QA Specialist is someone that will not be afraid to prove to you that you're wrong.

Players' bench

During different stages of the development process, you will need different abilities of the people involved. For example, after some time, the presence of a product designer may become less useful than more effort from a front-end developer. In order to feel secure about your product's development, you need to have the option to switch your team's assigned roles in the process and get certain people more involved than others.

You also need to take into account the fact that people will take days off, they will fall ill, and some of them will even leave the company altogether during the development process. None of those events should affect the development process. Having some additional people on the players' bench will improve the consistency and ensure the continuity of work.

Managing Development: Expectations vs. Reality Check vs. Trust

Trust your team but be involved in their work. You will have access to a staging infrastructure (a testing environment in which you will be able to see your product as it's developed) and multiple communication channels to keep all the flow transparent. Always ask questions if you are unsure about something – it's your team's duty to make you feel safe and well-informed. Be active, because your team will appreciate it greatly if they'll have easy access to you when business decisions need to be made or when they want to share ideas they think will benefit your project.

Treat your development team as your partners, which means that you should give them honest feedback and communicate all important matters to build a good work relationship. Try to be precise and clearly express your expectations. Give your team the right to say “no” – they will use it only on the basis of their expert knowledge. Listen to your team, because you’re working towards the same goal.

How to Estimate the Cost of Managing a Group of Freelancers When Developing A Product

You may not be aware of the unknown or unrealised costs associated with hiring freelance developers. You might also be unfamiliar with the processes and infrastructure required behind the scenes to successfully launch a project, as well as some of the risks related to hiring freelancers in comparison to a coordinated team approach.

Today, startup owners can instantly find freelancers of various skills and at different prices, thanks to portals such as Freelancers.com, Upwork, etc. This makes hiring a freelancer not only easy but also very appealing. What is more, employing a freelancer means that you don't have to pay for their health care or give them equipment to work on. A startup hiring freelancers doesn't have to worry about providing office space, because the freelance will probably work remotely anyway. This will obviously save money and time. Sounds great, doesn't it? Beware, though, that there are a few aspects you will have to pay attention to before hiring a freelancer.

Become a Mentor in a Kindergarten

When hiring a group of freelancers you don't hire a team – you hire a group of random strangers who most likely have never worked together. At the beginning, it's more similar to a group

of kids in a pre-school rather than a team of professionals working towards the same goal. There is a chance that after some time they will get along with each other well. For the time being, though, you have to accept the fact that they won't share the same working flow. For example, they may not be used to frequent commits, they won't observe the rules of code review, and their code will differ in quality. What's more, they will likely use different tools and won't be familiar with the tools you want them to use on your project. This means spending extra time on their learning process.

Individualists with Strong Personalities

Being individualists, freelancers are typically not used to working with others, so you have to find a way to manage your remote team effectively. For instance, you will need to set up a communication flow, define methodology and tools, and finally clarify the deadlines. You need to bear in mind that different people have different working styles. Some make very accurate reports on their work progress; others don't bother at all to regularly contact you. Some may prefer to work during the day, while others are productive only at night. Remember that everybody has their own pace and work habits as well. It's very difficult to build a well-integrated team out of such people. There is a chance that an unexpected conflict will arise. On top of that you need to regularly monitor and motivate your employees, or else they might abandon your project, and your costs will grow.

Taking Care of Your Project's Coherence

As a product owner, you should also decide whether you need a short-term solution or you will need to support your product in the future. Freelancers might be a good option if your project has a defined timeframe and you are not concerned about its continuity. If you are looking to maintain your project in the future, you want to make sure your team know the project through and through – this requires clear documentation and code base maintained regularly. Introducing these to a group of freelancers might take some time and time is money, right?

Major Risk – Lack of Commitment

Finally, you should never expect from a group of freelancers a real commitment to your project. They are simply not familiar with it and its aim, and they won't have any sentiments about it. Moreover, if they get a better-paid offer, they will most likely abandon your project and go to someone else. To add insult to injury, freelancers usually have many jobs and many bosses at the same time, so you are just one of many, and their career doesn't depend on the success of your project.

There are obviously many advantages to hiring freelancers, but also many drawbacks are involved. The matter whether to hire a group of freelancers or follow a more team-oriented approach stays a question of the startup owner's personal choice.

Effective Teamwork and Communication

The success or failure of each project depends on whether all stakeholders are on the same page. Shared concepts will save you time, money and a lot of stress. It is essential that you establish a set of shared concepts before you get invested in actions.

Why It Is Important

Imagine you're part of a football team. Your team will soon play in a very important match (the World Cup final for example). The problem is you don't know the rules at all. You have no idea what you are supposed to do, where to run, and why everybody is screaming at you. And now imagine that everybody else in the team has forgotten the rules. It would be a real mess, right?

You can't win if you don't know the rules of the game you're playing.

The same applies to any other kind of teamwork. Without a shared understanding of all the concepts, starting a new project will always be a struggle for team members.

Terms, Rules and Goals

It's becoming obvious that you need to understand specific vocabulary (pass, goal, offside), rules (each team consists of a maximum of eleven players), and goals/objectives (you need to score

more goals than the opposite team) to enable your team to even attempt to win the game. Naturally, concepts will vary depending on what you are working at. In web development, we don't deal with offsides – we deal with pull requests, devops, and code review. It is crucial to make sure everybody understands what those terms mean and how important they are.

Terminology is important because it defines the vocabulary shared among team members. It helps us understand what we are doing. We can have different types of terminology:

- Names of the processes and departments in your company
- Product design guidelines
- Names of components, classes or methods in the code
- Tables and values in the database
- Rules, which are actions which you should or shouldn't perform

Those actions can be required, permitted or forbidden. Rules help us define how we're going to do our work. Let us consider a few examples:

- Project workflow
- Deployment or QA checklist
- Weekly, monthly or quarterly procedures (reviews, planning, etc.)

Goals help us understand which direction we should follow and what we want to achieve in the end. Goals answer the question why we're doing it in the first place. Some examples:

- Sprint/release goals
- Product requirements
- Revenue/sales targets
- Company's v2mom
- Personal goals

How to Create Shared Concepts

Below you can find some recommendations on how to create and maintain shared concepts in your teams.

- Start with basic terms. You need to agree on the main subjects of your work and the most important definitions.
- Make sure everyone has access to relevant information. Use team collaboration tools to create a knowledge base. We are really into Confluence, but you just can start with Google Docs.
- You won't have a lot of rules at the beginning, but they'll emerge naturally due to mistakes people make and obstacles they come across. Make sure you document the rules properly.

- Set clear goals and make sure everybody knows and understands them.
- Document changes and irrelevant terminology, rules or goals could cause a lot of problems. Make sure everybody is always up-to-date with any changes.

The above concepts form the core of a team's existence. Terminology represents the things that they do, rules show how they should do it, while goals indicate the purpose of why they do it. This sort of a map can help you manage your team more effectively and help them go in the right direction.

Using Processes

Don't forget about the processes and the game strategy. In football, there is a saying that fans are the 12th man on the pitch. In the development process, your 12th man is the variety of processes that will govern the work of your team. You need to have your processes worked out and written down BEFORE you get down to work. Determine the duration of iterations, velocity of releases, and communication rules. That said, remember to stay open and flexible, because your expectations and needs might change during the project.

Designing web and mobile applications is a complex process that consists of multiple stages. The complexity of the process makes it very easy to make seemingly insignificant mistakes. Unfortunately, those mistakes can have serious consequences, even though they might seem small and unimportant at first – for in-

stance, an unnecessary line of code. In order to minimise the risk of making such deceptively harmless mistakes, it is crucial to create processes and procedures.

The Importance of Processes and Procedures

Some people underestimate the importance of creating processes and procedures, because they think it is just a waste of time. They couldn't be more wrong. In Netguru, we constantly take advantage of various processes and checklists. We do it for many different reasons.

Our everyday working lives can be really stressful and hectic. It is so easy to make a mistake when you work under the pressure of time. You can avoid making those mistakes, however, if you follow every single step of procedures you've established earlier. This is precisely what we do in Netguru. Creating coherent processes and making easy-to-follow checklists offers yet another benefit: it significantly reduces the amount of time you need to complete tasks, which translates into increased efficiency.

The 5 Key Features of a Good Process:

Efficient

A process must be created for a reason. It either needs to prevent mistakes and failures caused by our imperfect human nature or optimise the tasks we carry out. In other words, it needs to help us work faster or better.

Useful

A process should concern tasks that we carry out frequently or actions that bear significant risks and could potentially lead to serious consequences.

In regular use

A process needs to be followed. Each person that a process is relevant to must know that it exists and should take care to follow it under all circumstances. The people who fail to observe a process need to be given adequate feedback and should immediately receive a link to the relevant process or checklist.

Measurable

Metrics should be in place that would allow you to verify whether a process is being followed and help assess the impact the process has on the company's performance.

Constantly improved

Processes should not be set in stone. Every process needs to be updated and improved as soon as its circumstances change.

How to Use Processes and Checklists

Keep your checklists up-to-date

Checklists and processes must be easy to update. To allow this, we keep them on a shared Google Drive, which all the company's employees have access to. For instance, you can have a look at a checklist we follow before kicking off a project.

Always follow your checklists and processes step by step

It's really easy to think that you've remembered to do all the things on a checklist, but then it turns out you were wrong and you've forgotten to do something.

Double-check your checklists

We often ask our colleagues to check how we've performed a process or procedure, and whether we haven't missed anything important. We also find it useful to go through a checklist together with another team member, for example at a Google Hangout.

Set a benchmark

A procedure must have a point at which you can assume that the checklist or procedure has been completed.

Inform everyone about procedure checklists

Everyone needs to know the checklists and consider them the ultimate source of information about the processes.

Good processes lay the foundations for a company's success. They reduce the uncertainty and chaos of our daily working lives. That's why it is crucial that everyone in the company appreciates how important they are and reacts promptly, if the processes aren't followed carefully or plainly ignored.

Creating User Stories

Before any development can begin, the team must create user stories. After your first, detailed scoping session, the group will take the information they collected from you and start creating user stories.

But what is a user story? Your users are people who want to achieve something, and they will use your app to try and accomplish their goal. Your team will document the journey that your users will take using your product until they reach their desired end result. This will help guide the development process that will ensue.

A typical story: the PM gathers a list of information from a client of what needs to be implemented in a web application in order to deliver its business value – we call it features. Essentially, a feature is a user story, or a group of stories that are related (called an epic), and delivers a bunch of functionalities that end users would generally expect to get all at once. There are several characteristics of what a feature should look like and how it should be applied.

First and foremost, a feature should provide business value. This means that it should determine the health of the app in the long run. Business value expands the concept of value beyond economic value (also known as economic profit) to include other forms of value such as customer value. As you know, these values are not necessarily measured in monetary terms.

Secondly, a feature should be possible to estimate, which means it must have enough definition for the developers to provide an estimate of the amount of work needed to implement it.

If the feature is too time-consuming, it should be broken down into smaller user stories – preferably, the broken-down version should be small enough to fit within one iteration. There are two key advantages to breaking down features like this. Firstly, it makes estimation easier and more accurate – things that take a long time tend to be very difficult to estimate. Secondly, it allows us to track progress on the feature accurately. Breaking down stories is pretty hard, though, because you have to make sure that they still deliver value. Usually, the more complex features will be described in epics (bundles of user stories).

Within a project, the QA team can take care of the quality of the code. At Netguru, we apply the concept of test-driven development, which means that a feature should be testable and understandable for all team members.

We always encourage others to remember that user stories are not a contract – they are usually hints or reminders of features for the team and clients to negotiate and collaborate on to clarify the details when the time of development nears.

A user story (or several user stories in an epic) that describes a desired feature (functional requirement) should be written as a simple narrative, understandable for all team members. User sto-

ries are often created by clients. The format is usually not standardised. We try to avoid technical stories though.

A Typical User Story

Not every user story will consist of the following elements, but consider following this structure, especially if you're not experienced at creating user stories.

Title

For example: As a [user], I want [function], so that I can [value].

As an admin, I want to have access to an admin panel, so that I can get information on basic settings (users, passwords, editing and deleting profiles).

As a logged-in user, I want to be able to edit my profile, so that it is updated.

As a user, I would like to be able to search through the website so that I can find the content I need easier and faster.

Points

Points are a relative scoring system, usually estimated by a developer as a measure of complexity and, indirectly, the effort needed to complete a feature story.

Requester

You, the project manager, a quality analyst or another developer.

Owner

The developer assigned to the task.

Descriptive text

Such as acceptance criteria or conditions of satisfaction. A descriptive text defines the details of the work to be done and determines when a story is completed, ready for testing, and working as expected. A good way of gathering details is to ask additional questions such as 'What if … ?', 'Where …?', 'When …?', 'How …?'. We also use examples and graphs to visualise our way of thinking.

References to external documents

Such as screenshots, mockups, etc.

Comments

Left by any of the team members.

Inventing good features and writing user stories around them is not an easy task, especially for new projects and members who are used to different approaches and methodologies. Mistakes happen, so the communication between the team and the client is an essential part of our work. Efficient communication allows us to avoid (in the worst case) the implementation of a wrong

deliverable. The best idea is to follow agile methodologies' best practices and thoroughly discuss every feature before we start working on it, and then once more during the development. Because of all that, a fundamental part of our daily schedule is to attend daily standups, planning meetings, work review meetings, and retrospective meetings. Other forms of both oral and written communication help a lot!

The Essential Element of Design

You cannot build a house without an architectural design. Similarly, you cannot build an application without a design. The pictures in an app's design help clarify your vision so that the developers have something concrete to work with. It also helps you to see a visual representation of what your app will eventually look like when it is up and running.

If you do not have in-house designers you can work with, your Project Manager can connect you with a designer, based on your industry, your preferences, and other factors. Ultimately, you will choose the person you want to work with so that you are comfortable with the process.
You will work with the designer to ensure that the app incorporates the visual and branding elements you need to make the app your own.

Mapping Out The Work

You've hashed out your ideas. You have a design in place. Next comes the portion of the project where the work is laid out in detail for the developers.

Going back to the house-building analogy, you can't start constructing a building from the roof. First, you must have a solid foundation and a framework. You also need to determine when the electrical works complete, when the plumbing gets installed, etc. All of these elements must be coordinated in advance.

Your Project Manager and developers, along with your assigned Quality Assurance Specialist will begin to break down each requirement of the application into smaller, more detailed "chunks" of work. They write detailed instructions for each step of the process.

Reaching Product Goals

Reaching a goal on time and within budget requires a clever strategy and a battle-tested process. Without them, attaining your goals may not only be stressful but also inefficient, and may result in the product's failure. Following a structured development process suggested by a software development agency with all key roles filled will help you avoid it. A client may see progress faster during the first stages with a full-time employee or a freelance contractor. But over time, the investment in an established process will show its benefits – usually as early as a few weeks into the partnership. How do you reach your goal fast and smart?

Flexible from Day One

We prepared some tips you should take into consideration when choosing a software development agency and managing the co-operation with them. But before you start managing it, you need to choose an agency, and this choice should be based on a number of aspects. When deciding on which agency to choose, look for one that is flexible in every phase of product development. The agency should be able to help regardless of whether you have an idea for your product and want to start a project, or you want to switch from working with freelancers or another agency. They should be able to help even if the idea is the only thing that you have, and offer a workshop that will help you translate the idea into a vision of a working product.

Demand a Lot – From Yourself, Too

A perfect client-agency co-operation requires an active approach from the client. Your feedback is essential for the agency to successfully create your product. The agency, in turn, is obliged to let you express your opinions, and then take them into consideration. Before you ask an agency for help, prepare all materials related to the project. It's not a dealbreaker, if you don't have any materials yet – a good agency should help you create them. Think carefully which type of project methodology you want to follow: fixed price vs. time & material. The choice is not as obvious as it might seem. Time & material – which we recommend at Netguru – is more flexible, because it enables you to change your ideas throughout the project, but the final price may change as well.

You can always choose the fixed-price method, but you have to bear in mind that every change of plans will incur additional costs. You won't be able to develop your product fast – instead of focusing on the development itself, you will waste your time on negotiations.

Make People Focus on What They Are Good at

Remember that product development is a team sport, and you need more than one person to deliver your product successfully. Try to fill the key roles in your project with people who are naturally capable of performing the functions you assign to them:

- You need a developer to think about your product's architecture and write code.
- A quality assurance specialist will check if the code is of the highest possible quality and resolve all potential issues.
- A product designer will build a product with a potential of bringing high profits and provide you with both a conceptual idea for your product and its visual interpretation.
- A DevOps engineer will make sure that what you're trying to create is scaling properly and that the product is secure. They will also ensure that the solutions you choose satisfy the needs of your clients now and in the future.
- A project manager will keep all the communication channels open and will make sure you reach your milestones on time by carefully choosing priorities and managing the team to achieve the highest efficiency possible.

Remember that even when someone is good at all of those functions, "context switching/putting on different hats" is super hard, and making one person perform all the functions will end in only one way – you will end up with extensive delays in your project.

Experience is More Important than the Price

When looking for an agency, choose one that has a well-structured process, which they will make available to you from the very beginning. You should start with an initial kick-off as soon as you meet, get to know each other and plan initial milestones. The company that you pick should have existed on the market for some time, because it means that it has the relevant experience and – which is even more important – that its processes have already been tested on many projects. It's worth paying a little bit more, as it will decrease the chance of the project's failure. Look for an agency which will help you define your MVP and will limit the initial phase of development to 3 months so that you can show a working product to customers and investors as soon as possible. Of course, it doesn't mean that the agency should reject your ideas – they should be stored in the backlog for future milestones. In the meantime, you can conduct market research that will help you to adjust the next steps to the market's needs.

Feedback and Transparency Are Key

To hit your goals smart you should ask for feedback and give it to your agency on a regular basis. To be able to do it, look for an agency which has processes build on the culture of feedback

and effective communication. The agency should encourage you to stay in touch with all people engaged in the project. Try to stay active and get involved in the project as much as you can, because a successful initial phase will have an impact on everything that ensues. It allows you to understand technical challenges better, which will translate into better results. You should also have access to the code, staging, and a project management tool to be able to track the project's progress on a regular basis. Only a very efficient and fast feedback loop will result in successful iterations.

Hitting Product Goals

To sum up, when you want to hit your product goals fast and smart, you should think about the project's completion from the beginning. At Netguru, after having delivered over 160 products both for startups and well-established corporations, we know that experience and efficient processes are an essential and invaluable part of every project. So, if you want to be smart in achieving your goals, be smart when setting them.

Small Tasks

Breaking up work into smaller tasks is better for the team's morale and planning. Sadly, many teams fail to do this and take on huge, week-long tasks. This is a serious problem. Luckily, it can be avoided by an experienced team.

Why Are Small Tickets (Tasks) Better for Your Product Development?

Reason 1: Small tasks are much easier to estimate incorrectly
When you'd like to have a new feature of your product finished within two days or more, it is almost certain that something else will pop up. Let's be honest — it is impossible to plan a whole work schedule for two days when there is just one task. It's extremely important to break big tasks down into smaller ones. So when thinking about introducing a new functionality, make sure your Project Manager splits the job into smaller tasks for your development team.

Reason 2: Small tasks are way more transparent
With smaller tasks, both the team and you can easily see what is going on and track the progress of each task. It is much easier to quickly pinpoint a bug or a problem, which makes the whole procedure a much smoother one. Even if a bug does eventually rear its ugly head, it won't break too many things, and so it will be much easier to fix! Transparency is key, because the very last thing you want to have when running an IT project is confusion about who did what, how and when.

Reason 3: It's much easier for a PM to plan the work of the whole team
A PM can react quickly when a member of your team sees that something is going wrong and they have a blocker to cope with. The PM will also be able to see blockers right at the planning stage of the project.

One of the worst things that can happen is the team not finishing the sprint on time. The solution is simple: dividing their tasks into smaller ones and estimating each one separately. It's always better to finish faster and start another task from the backlog rather than not finish them on time at all.

Reason 4: There will be visible progress in your product's project management tool

This really motivates the team. They can literally see the work going forward each week. Nothing more exciting than that! Also, it's easier to track the progress of each sprint so you will be able to really be on track with what's happening in your project.

Reason 5: Small tickets mean better task descriptions

Your PM will be much more accurate while describing each step of the tasks for the team rather than the task in general. Also, your QA will thank you for precise descriptions. It's easier to review smaller chunks of code rather than 150 lines of a massive task.

Developing The Application

The development process is where the rubber hits the road. Once you have signed off on your designs and you, your Project Manager and your QA Specialists are clear about the requirements of the application, the developers begin the work of building out your app.

While working through the development stage, you should remain connected to your Project Manager and/or Your QA Specialist. Mistakes or changes in this stage of the application creation process can be costly, so continuous, proactive communication is essential for success.
At Netguru, you will stay connected through email, Slack, phone, video conferences, and JIRA (our preferred task management program), which will allow you to see where your project is at every stage of the process.

Your Alpha Tester: The QA Specialist

Your Quality Assurance Specialist is there to test the development at every step. It is the job of the QA team to think of the things that could go wrong with the application before a user lays his/her hands on it.

QA Specialists advocate for your user. As the application takes shape, they keep the user experience in mind and ensure that the application will help users achieve their goal from the time they log in till the time they log out.

QA Specialists are responsible for:

- Ensuring that the requirements are defined in as much detail as possible.
- Aligning the features of the app with the defined requirements of the project.
- Filling in all of the missing gaps so that developers have the information they need to tackle each step of the project.
- Managing all documentation related to the project.
- Managing project credentials.
- Testing of the app to ensure that requirements are met at each stage.
- Creating a list of edge cases and testing of each of them.
- Managing internal functional notes, i.e. a document that contains essential project information for any new developers joining the project after it has kicked off.
- Participating in “bug bashing” sessions, where the team put the app through extreme testing to ensure usability.

What is an Edge Case?

An edge case is defined as a problem or situation that occurs only at an extreme (maximum or minimum) operating parameter. For example, a function that divides two numbers might be tested using both very large and very small numbers to ensure it works at both ends of the spectrum. If it works there, it should work correctly in between. Your QA Specialist develops, maintains, and tests edge cases throughout the development.

Iterations

How can you make sure that there is ongoing improvement in your project once it has entered an active development phase? How can you eliminate potential stumbling blocks that pop up from time to time? How can you ensure that best practices are maintained and problems minimised? The answer is simple: Iteration Retrospectives! Let me tell you how we do it at Netguru.

What are Iteration Retrospectives?

Iteration Retrospectives are short internal meetings held at the end of an iteration, preferably before the next one starts. During an IR, the project's team members, moderated by the Project Manager, have a short discussion about how the last iteration went. The best moment to organise an IR is alongside an internal standup/planning session shortly before the weekly call with the client – and of course, the client is more than welcome to participate. With Netguru's 1-week iteration flow, an IR should last no

longer than 10-15 mins, however, the exact time will depend on the given team.

Giving Complex Feedback on an Iteration

The ultimate aim of an IR meeting is to identify things that negatively influenced the team's workflow last week (for instance: overestimated task(s), poor organisation, too much time spent on calls, various blockers, unexpected refactorings, tests, etc.). Simply mention anything that bothered you or hindered your team's workflow. Next, discuss how these problems can be eliminated in order to enhance the team's workflow and future performance.

At the same time, try to identify and shed light on practices that brought extra value to your team's workflow. These "best practices" ought to be promoted and repeated over future weekly iterations, if possible.

Proposed Questions & Survey Templates

OK, but how do we get this information from each team member? The following simple questions should help you and your PM initiate and stimulate discussion among your team:

- How was the iteration? Rate it on a scale from 1 to 5, 1 being the lowest and 5 the highest (Make your choice based on your overall subjective feeling)
- What went well? (Mention at least one thing!)
- What didn't go well?

- What actions can we take to improve our process going forward?

In order to automate the process of collecting the feedback from all your team members, an iteration feedback survey can be used. We suggest that your PM sends (via Slack or email) the survey to your team members 1-2 hours before the meeting, so that they have enough time to respond. The survey should only take about 5 minutes to complete, and is a great resource for the discussion at the retrospective.

Furthermore, what's great about individual surveys is that team members do not have to share their initial feedback in front of the team. Believe it or not, but this kind of privacy can lead individuals to be more willing to share their insights about the last iteration! Remember that the survey is optional and doesn't have to be used every time. Check with your team whether they prefer a discussion only, or an initial survey and then a discussion.

IR Follow-up

It is crucial to monitor and check whether the actions undertaken on the basis of an IR have actually been implemented and if they deliver the desired results. Moreover, be open to your team's opinions and feedback regarding the new initiatives. Do not hesitate to organise a quick team follow-up after a week (perhaps even merge it with the next IR session) in order to see if ideas generated during the last IR have actually worked out.

Frequency of Iteration Retrospectives

Generally, it's a good idea to do an IR after every iteration, but never feel forced to do it. If you have a feeling that an IR will not bring any value, just skip it and wait for a more opportune moment. You may notice that over time that the retrospectives get shorter and shorter as the team gets used to discussing and resolving issues.

You'll be surprised how willing the team are to share their workflow, insights, and feedback with you and yor PM. Just create a 15-minute "space" for them to talk, and you'll be able to identify many strong and weak points in the team's workflow.

Checks And Balances

When the app is complete, it is returned to you, the client, for launch. Once the handover wraps up, you provide feedback to your Business Development Specialist, letting them know what went well, what could have gone better, and any other details you wish to share.

That feedback is passed on to the team, and addressed during an after-action review as soon as your project is closed.

Our team at Netguru is also subject to peer reviews to ensure that each project team member is working as effectively and efficiently as possible. These sessions ensure that we are always improving our processes, and your input as a valuable piece of the puzzle.

Pitfalls

How Not to Run Your Project

Following the best practices discussed in this guide will only get you so far. It' just as important to learn from others' mistakes and experience. Let's now list some of the most common problems that can occur in a dev house, and what should be done to avoid falling victim to such errors.

Typical Development House Problems (and How to Spot Them)

Meetings

- Too much time spent talking, and not enough time spent on doing actual work.
- Meetings can become a dreadful waste of time if their agendas are not adhered to rigidly.
- Time wasted on irrelevant things.
- Of course, it's a great thing when everyone in a dev house gets along swimmingly, but team meetings are not the place for personal or professional digressions.
- Devoting too much time to a single topic.
- We've all been in meetings when a particularly hot topic

seems to have everyone excited, and each person in the room has their own opinion on what should be done. This is no good and the PM needs to step in and take control of the meeting. A solution should be decided upon, everyone must agree to it (whether they are in favour of it or not), and the meeting must progress to the next item on the agenda.

- Reopening already closed issues.
- Once a topic is finished with, it's finished with. No ifs, buts or maybes.
- Decision stalling.
- The only way to progress is to make a decision – even if it's a wrong decision. It's far too easy to contemplate one idea or approach after another. Decisions need to be made so that everyone can get started on actual work.
- Failing to predict potential obstacles.
- As the old adage goes – if you fail to plan, you plan to fail. Good practice dictates that you should spend a little time forecasting the potential pitfalls of an iteration. Time needs to be factored in to account for this, and alternative solutions should at least be thought of in advance so that no one is left completely floundering when they hit a wall.

Communication

- Agreeing to everything the client wants, instead of opening a constructive dialogue with meaningful feedback.

- Ok, if it's possible to deliver precisely what the client wants every time, then great! But the reality is that your team won't always be able to do so – and even more unlikely that you will have the very best plan and ideas for the whole project from the outset. The client is not always king when it comes to development – which is why they hire a development team to do this bit for them. A constructive dialogue needs to be ensured, with appropriate feedback flowing both ways.

- Taking things personally.

- Nobody likes to be the one who got something wrong, but when you do, you should own

- your mistakes and look at them objectively. The ultimate end goal is to create a product for which there is no rival, and sometimes that means that mistakes had to be made in order to find the best solution. This, indeed, should be the attitude adopted by everyone in a dev house – and PMs should lead by example.

- Inadequate mastery of project management tools.

- Great project management lies at the heart of all successful projects. But, the PM can't do it all alone. S/he will de-

pend on the whole team's mastery of the prescribed project management tools in order for tasks to be completed adequately and efficiently.

Outdated Project Management Practices

One problem that we are becoming increasingly aware of is the persistence of outdated project management tactics that do nothing but demotivate employees and cause all sorts of dissonance and delays to an otherwise promising project.

So, let's take a look at what these common glitches are so we can all know what to avoid in the future. If you see your PM doing any of the following, don't hesitate to talk to them about it. Remember, constructive feedback is always a good thing.

Which Project Management Strategies to Forget

Overplanning

The old adage that goes 'if you fail to plan, then you plan to fail' will always hold true. But there is such a thing as restraint when it comes to planning your projects. Far too often this stage – which, in the age of Agile, should really serve only as a guide from one iteration to the next (i.e. about a couple of weeks at most) – is allocated a disproportionate amount of time from the get-go. Development teams don't need to be tied down to the drawing room for 6 weeks before they even start coding – they just need to start coding.

The work, and the world too, is changing so fast that even planning for mere two months ahead can prove to be futile. A better approach is to plan in small segments – perhaps just from one week to the next. This way, everything is adjustable to any unexpected changes that will occur.

A scope that is too rigid

Following on from the last point, too rigid project scopes (i.e. the hows) and too rigid product scopes (i.e. the whats) can be disastrous for a project. When beginning a project, nobody knows for absolute certain what's going to be needed 3, 4, 5 or 10 iterations down the line – so why should a project manager (PM) insist on how a given should be done right at the outset? Similarly, although the client will, of course, know what he/she wants the end product to do, a flexible product scope is the only way to ensure the delivery of a well-designed, great product, as we need to account for all the lessons learned along the way.

Continuous risk management

Once again, continuous risk management falls under the overplanning umbrella. It's impossible to account for all possible risks, as these will continuously change in line with a flexible, agile plan and scope.

Relying only on verbal communication

Even the best of us can't keep everything that needs to be done – nor everything that has already been done – in our heads all at once while we're trying to concentrate on our current project. And

when a team relies only on verbal communication to organise its work, things very quickly become muddled and, later, forgotten. By writing things down, all team members are able to follow the workflow, trace back what has already been achieved, and what they need to prepare for.

Lack of order in documentation

Developing clear and written communication pathways is only ever as useful as your preparedness to document all correspondence. Therefore, once communications have been written down, they need to be stored somewhere – ideally online – where all team members can access them. See our blogpost: '5 Remote Team Management Apps To Rock Your Performance'.

A fixed mindset instead of a growth mindset

In many ways, having a fixed mindset is the error that summarises all of the above points. Indeed, if we, as the human race, had forever had a fixed mindset, then we would never have evolved into the great community-oriented people that we are today. Indeed, we would have stopped at inventing the wheel (if that!), and never bothered to progress to the stage of motors, automobiles and all the rest. My goodness, if we had stopped at the calculator, then we would never have invented the internet! It doesn't bear thinking about.

No, development teams need to take an evolutionary lesson and always be prepared to adopt a growth mindset, be willing to

learn new things, and thus grow and improve, and progress, and evolve. Only this way great products are made.

Not learning from past mistakes

To err is human, according to Alexander Pope. We're pretty sure that even developers qualify as being human, which means that they're going to make mistakes every now and then. And so too will marketers, sales teams, CEOs, and even good ol' project managers.

Of course, making mistakes is OK – as long as we learn from them and do not repeat them. Indeed, the smartest amongst you won't even need to learn from your own mistakes – you can learn from the mistakes others made before you.

Starting Over

Are you having problems with adding new features to your project easily? Are you unable to quickly make significant changes to the project? Maybe you spent a lot of money and aren't seeing results from the development team? As a product owner, you might want to rewrite the application from scratch. Take a look at the possible advantages and disadvantages of such a solution and make a wise decision.

Developers as Creators?

Developers have their world of code. Every single project is a small world that the developer has created. Every project, every single line of code is the result of the developer's creativeness, imagination, and understanding of the project's scope. When you are the master of your project, you know it from the first line right down to the last, you understand what you've created, and you know what you had in mind when you began. That is why sometimes a creator prefers to work on a project from the very beginning rather than first analyse and understand the creation of some other project master. Therefore, every IT Project Manager should consider creating the project from scratch and allowing creators to realise their ideal vision.

Rewriting from Scratch?

Some developers assume that it is easier to start from the very beginning than to improve something that already exists. They believe that it is more difficult to read code than to write it. Sometimes even the creators have problems understanding their own code, so following up on some other creator's idea is a more time-consuming, difficult and boring task.

Reasonable Arguments that Speak in Favour of Re-writing an Application:

- The creator finds it difficult to understand the source code.
- The language of the source code must be changed.
- The source code has become problematic.
- The source code is not available.
- The existing code cannot be compiled for some reason.
- Debugging has become challenging.
- It is not possible to transfer the code to a new platform.
- New changes require rewriting.
- New techniques require rewriting.
- New code could fix previous problems.

- There is no way to figure out how the previous code works.
- The architecture is going to be changed significantly.
- The code is bug-ridden and poorly structured.
- You like rewriting code as a hobby.
- You'd like to have an exercise for your development skills.
- There are Intellectual Property-related or licensing issues.
- The new creator is going to be dealing with the project for a long period of time, and they feel uncomfortable with accepting responsibility for somebody else's source code base.

Terrible Reasons for Rewriting the Code from Scratch:

- Ugliness.
- Disliking the coding style.
- Laziness.
- It is not done the way the new creator would have done it.
- The application is too slow (simply find the bottlenecks and optimise them).
- Lack of respect for the previous creator.
- The app was not written by "me".

Inadequate Arguments for Avoiding Rewriting the Code from Scratch…

- Not trusting the creators.
- Sticking to "If it's not broke, don't fix it".

What about Refactoring the Code?

Some of the above problems may be solved not by rewriting the whole code but simply by introducing small changes, i.e. refactorings. Reusing the work of other creators is difficult, but it is not completely impossible. Refactoring is a task which takes more time, but in some cases it is understandable.

The following arguments suggest that you should refactor the code:

- No information about the features and the legacy code.
- No information about requirements.
- Reinventing the wheel is useless.
- Re-writing usually takes longer than we expect.
- Re-writing the code step by step does not impact significantly the user experience.
- There is no guarantee that the new creator will do a better job than the previous creator.

- The new creator may introduce the same problems as the previous creator.
- The new creator may even introduce new problems to the software.
- The old code has been already tested, bugs were fixed, and the project was improved.
- Throwing away the previous project requires accepting having wasted the time and money previously invested.
- The investors are put in an extremely dangerous position since users may move to other products

Suggestions from the Professional Creators at Netguru

We suggest taking into account the factors listed above and the fact that rewriting an application from scratch is only one of two options. Refactoring is a viable alternative. There is no good answer for the question whether to rewrite the application from scratch or to refactor it. It depends on the case under consideration. It is important to be knowledgeable about both options in order to identify the best solution for the project. Being professional means facing up to challenges.

How Long Will This All Take?

Developing an application takes time, and the full project cycle will vary depending on your unique requirements. During the scoping process, you will be given a timeline for your project. Keep in mind that changes are inevitable as you and your team work through the project. If and as changes arise, the timeline may be impacted, and your team will keep you up-to-date on any alterations to the delivery date.

Summary of Netguru's Process

Your idea + initial budget from investors = success.

At Netguru, we love working with startups and supporting them from the very beginning. Likewise, we don't mind jumping into an ongoing project. For the time being, however, we'd like to discuss the possibilities you have if you are beginning the journey to a great product.

So, let's start from the beginning and assume we will create a product from scratch.

- You have spent quite a long time figuring out what your potential customers/users need.
- You have talked it through with your partners, friends, family, potential investors.

- You know how to fulfil the needs of your potential customers.
- You feel confident that your application will meet the market's demand (perhaps because of the marketing research you have conducted?)
- You have an initial budget to get started.

If you have and have done the above, you have all you need to get in touch with us and start the machine rolling.

If you'd like to introduce the process described in this book at your organisation and make your product development more efficient, get in touch!

- hi@netguru.co
- **+44 2038 713 389**
- Our website

Ready to Take the Next Logical Step for Your Project?

We hope this book helped you see the product development process from a bird's eye view and that you now understand it as a whole. Our goal was not to prove to you that developing an app is neither simple nor cheap. It was to show you that a software product is something you can easily add to your business, so long as you follow the best practices and avoid common pitfalls. This way, you'll be off to a good start and ready to think about scaling your product.

For a comprehensive guide on scaling software products, check out our ebook Scaling SaaS for CEOs.

And as a cherry on top of what we've shared in this ebook, here's an infographic presenting the 7 steps to successful IT project management.

At Netguru, we have over 10 years of experience in developing software products, from early-stage startups to projects within mature organisations. We specialise in taking the whole burden off of our clients' shoulders and delivering quality web, mobile and desktop applications on time, within budget and regardless of project- or industry-specific limitations. If you have any questions about our process or would like us to take a look at your project, we'll be happy to help.

You can:

- Talk to us on Twitter or Facebook
- Subscribe to our newsletter for more tips and resources
- Contact us directly: hi@netguru.co, +44 2038 713 389