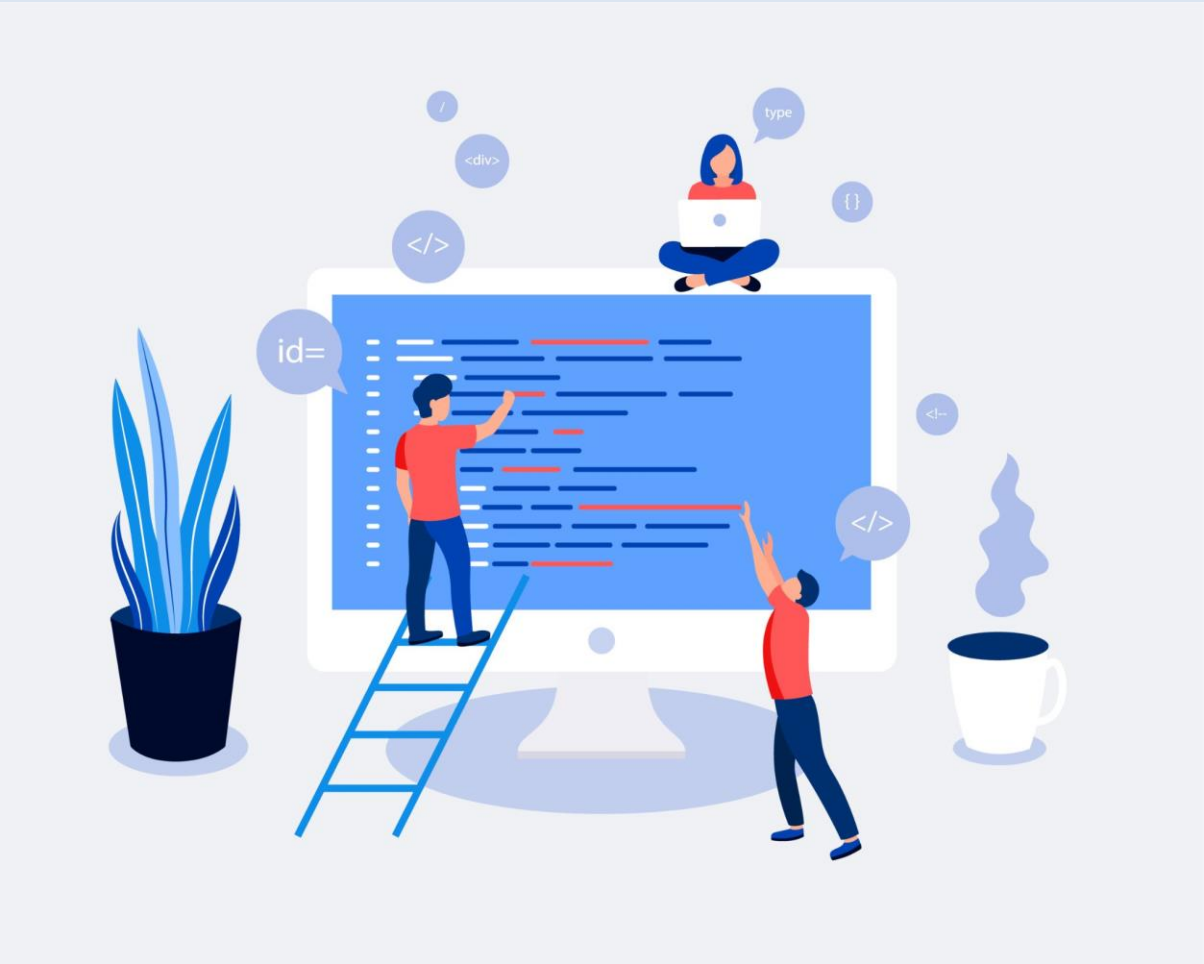

The Principles of Successful Software Development

CONTENTS

The 5 Most Important Things to Consider When Planning Your Software Development Project

In its development, the software goes through the software development life cycle (SDLC), from the conception of the idea to launching.

These phases are;

- identifying required software,
- analyzing software requirements (manpower and capacity),
- specifying tools needed,
- designing the software and programming,
- testing, and
- maintenance.

There are several Software Development Processes, which depend on factors like your goals, size of the project, and team. Examples are *Waterfall* (linear sequential model), *Agile and Scrum*, *Incremental and Iterative*, *V-Shaped* and *Spiral*.

The 5 most important things to consider when planning your software development project are:

1. Viability Examination and Planning

Software project planning has various requirements to be met to avoid dragging out timing, inevitably driving up the cost, or even frustrating your team.

First, identify if there is a need for something, what exactly is needed, and if the proposed process can deliver desired results. With the team, equipment, time, skills, and budget required, compare the same to available resources, select a process that maximizes resources in the allocated timeline, and best results.

2. Examining and Choosing the Requirements

Choice of requirements involves interaction with end-users, understanding user documents, and regulations in your line of business.

Have an outline of the project and software requirements from an expert. Infrastructure includes; PCs with advanced, integrated environment suite, a server to track and manage documents, a server to examine, produce and deploy software, a tool to follow requirements, defects, and tasks, an automatic building system, a tool to test regression, and system to report issues.

A comparison of strategies will help you determine the most appropriate one. The template selected should be followed, and actions tracked through selected tools.

3. Designing

The designer or programmer should have a document demonstrating how the program meets set requirements. Simple wireframes showing interaction with the software or prototypes should be made.

This validates ideas and gives valuable feedback before coding. Tools and indicators for data collection help identify issues on time, and tracking this data over time helps determine release time.

4. Developing

Executing outlined procedures has 4 phases; *Alpha*, the first version, tests functionality, *Beta* for internal examination and testing usability, *Release Version* is relatively stable but needs some changes, and *The Gold Master* is ready for release.
Feedback expected is on user experience, speed of task completion, and functionality in general. Automate repetitive tasks (building code, scanning, testing) to eliminate human errors.

5. Testing and Integration

This stage pushes the code into production and launching. Features should be listed, scripts for all user actions made available and tested to work, and fix issues detected (bugs and lags). Integration is slow and gradual to increases the chances of approval. Using UX tools, track how users interact with the software.

Automate this using a continuous deployment model or Application Release Automation (ARA) tool.

Maintenance and Updates

Bugs arise after use, and new features need to be added. Maintenance allows modification and updating of the software to improve efficiency as needs become dynamic.

There’s a wide range of factors to consider before starting your software development project.

How To Build a Roadmap For Product Development

Going to market with a new product is an involved, expensive process. It doesn't happen in a vacuum; there are stakeholders at every stage that need to be kept in the loop. Having a roadmap for the product development process helps keep you on track throughout the process.

Keeping Your Project Together

It takes strong leadership and clearly defined goals to keep a project on track. While the development and management teams are both important for the process, communication between the two is the real goal.

How do you know if your leadership team has what it takes to keep your product development project on time?

- Leadership has a clear set of goals
- There is a process in place to find bugs early in the development process
- Everyone involved has an understanding of where the profit lies in the product

Know Your Product's Story

The product roadmap you build needs to make sense. You need to understand the projected path your product will take, and that path needs to move progressively forward. Each step builds on the one before it.

How to get there?

The first step is to separate your goals. You will have customers and users (who will not always be the same) and you will have business goals. Break down goals for each segment into further subgoals. As you break the goals into smaller ones for your customer, user, and business development, you will begin to create a path.

A goal-centered road map doesn't get bogged down in the details and gives your teams the autonomy to provide their best work. During this stage, it is important to set boundaries regarding outside input on your project. This is prime time for a stakeholder to want features or additions that move the goal post, getting your project off course.

Stick With Your Plan

Once you have a roadmap in place you need buy-in from your stakeholders to stay on track. Keeping them involved through the development process makes it easier to keep the focus goal-oriented.

Just because you need everyone on board to achieve buy-in doesn't mean you need to acquiesce to every request. Keep your goals front and center, and listen to input. Then decide whether the idea just creates noise or whether it aligns with the product goals.

Using goal-based product development gives you a way to measure progress. With specific goals, it is apparent where you are in the process and whether it is on track.

What to do if you find your project moving off-course?

Short review cycles are a great way to keep your project on track. Your roadmap should not be so rigid that there is no room for adjustments. Having short periods between reviews allows you to notice problems before your project goes entirely off course.

Early detection of problems allows you to deploy fixes during the architecture or coding phase ideally, and by component testing at the latest.

Why Deployments Fail

The truth is, deployments often fail for simple reasons. Too much focus on the bottom line, ship dates, and other business matters, and not enough time focused on technical issues, is a sure path to failure.

Another reason for difficulties with deployment is the lack of clearly assigned roles. Without established roles, there is a lack of accountability. With a lack of accountability comes failure.

Regardless of everyone's role in the project, each should have a solid understanding of the goal of the project, the benefits the project will provide, and what will be the determining criteria for a successful project.

User stories - why you should write them

As someone working in software development, you've probably heard of (or worked with) user stories before. However, many people misunderstand and misuse them — these are not simply another set of software requirements.

The point of the user story is to make sure that the end-user is never out of sight; something that can easily happen if a team of developers suffers from tunnel vision. It's important to never forget the big picture — which is why user stories exist. This powerful format allows the development team to view their project from a non-technical point of view.

Once developers read a well-written user story, they will have a clear view of what kind of value they're providing to the end-user, and why they're working on their project, to begin with. This kind of clarity increases creativity and collaboration.

Why should you write them?

User stories are crucial for defining your end product more clearly. If you utilize a set of well-prioritized, properly defined user stories — you and your team will have a firm grasp on the entire point of the project and the value of the product.

It's what you would tell someone about the product without using overly technical terms; simply put, it's how you would describe your product to someone who isn't a developer, in simple plain English.

So, why is this so important for the development process? As you will see, pulling back from the technical nitty-gritty details and looking at the big picture in a meaningful way provokes thoughtful product discussions.

If you take the time to craft a user story thoughtfully, you will have the best possible basis for discussing the product as a whole within your team; which is sometimes just as important as the technical practicalities.

How do they help to keep the team aligned?

Larger projects can require quite a lot of teamwork from a huge number of people; in many cases, entirely separate teams that need to coexist and collaborate on a single project. In such a situation, it's difficult to keep the entire team aligned and focused on the common end goal.

That's where user stories come in to provide clarity across the board — knowing why you're creating something, and for whom, is crucial for achieving goals in a timely and productive manner. And they're equally useful when you have to make a technical deep dive as they are for a scope discussion.

Perhaps most importantly, user stories allow for more effective project participation from non-technical members of your team. The average contemporary software project is incredibly complex; meaning that the bulk of acronyms and technologies are not easy to grasp for non-devs.

Your user stories allow those team members to have a clear overview of the project as well, increasing the value that they bring to the table.

What should be included?

Most importantly, you need to ensure that you think like a particular user, or as more commonly referred to in user stories — a specific persona. This gives the story a perspective and an angle that informs the point of the project.

These persona definitions are the most crucial aspect of a user story because they allow the development team to connect to their target audience and understand their needs.

On a more practical note, a complex project may involve a multitude of smaller user stories for specific functionalities as well. This can make it difficult to administrate and navigate them; which means you need to use the proper metadata. Make sure that you tag, categorize, and name these stories in an efficient manner.

Translating user stories into code

At the end of the day, the task of your development team is to take these user stories and create a product that complies with their point. You can experience all of the benefits of extra development power from industry experts without hiring additional developers or committing in a meaningful way.

What makes a perfect user story?

You can build the best piece of software ever written, but if no one understands what it can do for them, its chances of being widely used are slim. Great user stories will crystalize for people what your product can do to make their lives easier or better.

Users come first

You need to gain an understanding of the actual users of your product. Creating a user persona is an excellent way to get in touch with their expectations. What problems are they trying to solve? Thinking like a user will help you see your product from their perspective. You will gain insight into how they see its functionality, their needs, and the expected value they're looking for.

Keep it simple

User stories need to be high-level but also accurate and straightforward. Avoid using buzzwords, jargon and acronyms. Using simple language is the best way to get a user story across so that team members and other stakeholders understand user needs.

Think big

If a backlog of user stories describes your product, there is no reason to be bound by constraints such as budget, time, feasibility, or cost. There is little cost in allowing some of your most wild user stories to enter the backlog. However, the value of allowing some of your most expansive thinking into the backlog is enormous.

Use epics

An epic is your big sketchy story. They typically describe large pieces of functionality that are broken down into smaller stories over time. It is like breaking down an engine into its moving parts. Epics are a fantastic tool for organizing your stories and keeping a clear view of the big picture.

Keep adding stories

As you get new ideas and develop new ways users will interact with your product, it is good to keep adding new user stories to your backlog.

It is essential that you can group and prioritize new stories according to the value of each story for the user and your business. This will also stop you from being overwhelmed by a huge number of stories and keep development on track. The difficulty, cost, feasibility, and effort required are also factors to consider when prioritizing stories. Cross dependencies in the stories can also force the order of

entries. It is important not to discard stories that are not a priority. Hanging on to them means you can use them down the track as your product develops.

You still need specs

Having a set of great user stories is a wonderful place to start, but they are just the beginning of the process. Your team will need to outline how the stories will come to fruition from a technical point of view. Linking stories to documentation that shows technical details from a software engineering perspective is a great entry point for more detailed technical specs.

Measuring success

How your product operates in the real world will measure its success or otherwise. Just getting to it works as it should or works according to the specs isn't enough. Actual user feedback, where you learn how happy and engaged your users are, will be the test for long-term results. Working according to user stories is fine for development, but you should be refining according to real-world results.

User stories are a great way to define your product and what you hope to achieve. They are brilliant at capturing your ideas and being the bridge to product development.

Contact:

Ben Rozell, CCO

NerdCloud

ben@nerdcloud.co