Software Development – 7 steps

Getting an idea from early concept to commercialization is a heavy lift.

Some of the common problems include:

- Inaccurate or incomplete requirements gathering
- Lack of clear project scope
- Unrealistic deadlines
- Unclear communication between stakeholders
- Poorly designed software architecture
- Issues resolution

For example, we were involved with a software development project that was behind schedule and over budget. The software had been developed using an inaccurate requirements-gathering process, which had led to scope creep and unrealistic deadlines. We were able to help by redesigning the software architecture and better managing the software development process. As a result, the project was completed on time and within budget.

Here are **seven steps** to help your software development project succeed.

1. **Defining the problem**

   Software development projects usually start with a problem that needs to be solved. To define the problem, it is essential to understand the project's goals and what stakeholders are involved. Our clients often come to us with an idea for a software solution, but they are not quite sure how to take it from concept to commercialization. This is where our Program Management skills come in. We work with our clients to help them define the problem they are

trying to solve, and then we help them develop a plan to take their idea from concept to commercialization.

2. Gathering Requirements

Once the problem has been defined, the next step is gathering stakeholders' requirements. This can be done through interviews, surveys, focus groups, or other methods. Requirements gathering is essential because it ensures that the software being developed meets the end user's needs. Without a thorough understanding of the user's needs, it would be not easy to develop software that is usable and useful. Additionally, requirements gathering can help identify potential problems early on in the development process, saving time and money later.

There are many different ways to gather software requirements. One standard method is to hold meetings with stakeholders and end users to discuss their needs. Another is to use software tools that allow stakeholders and end users to input their requirements directly.

3. Designing the solution

After the software requirements have been gathered, the next step is to design the software solution. This involves creating a blueprint of the software that will be developed. The software design should consider all of the requirements that have been gathered, as well as any other constraints that may be present (such as time or budget).

Software architecture is the high-level structure of the software, and it defines how the software will be organized. This is a critical step in the software development process, as it will determine how easy (or difficult) it will be to develop the software and how well it will meet the needs of stakeholders and end users.

There are a few different types of software architectures that are commonly used. The most popular ones are client-server architecture, mainframe architecture, and distributed architecture. The client-server software architecture is probably the most commonly used today. In this type of software architecture, there is a server that hosts software.

There are a few different types of software development frameworks that are commonly used. The most popular ones are the Model-View-Controller (MVC) framework, the Model-View-ViewModel (MVVM) framework, and the Model-View-Presenter (MVP) framework.

Some popular software languages include Java, Python, PHP, and JavaScript. These languages are widely used because they are relatively easy to learn, and they are also very versatile.

They can be used for developing a wide range of applications, from simple websites, Apps to complex software systems.

When deciding on the best software architecture, framework, and language for your project, it is essential to consider a few key factors. The first factor is the type of project you are developing.

PHP or JavaScript might be a good choice if you are developing a simple website. Java or Python might be a better choice if you are creating a more complex software system.

The second factor is the size and scope of the project. A more robust software architecture, such as Model View Controller (MVC), might be necessary if the project is large and complex. If the project is small and straightforward, then a less complex software architecture, such as Basic Four Tier Architecture (BFTA), might be sufficient.

The Basic Four Tier Architecture (BFTA) is a software architecture designed for small and simple projects. It is based on the four-tier model, which consists of the presentation layer, the business logic layer, the data access layer, and the database layer. The presentation layer

handles the user interface and interaction with the user. The business logic layer contains the business rules and logic. The data access layer is responsible for accessing and manipulating the data. The database layer is the back-end storage for the data.

The BFTA is a good choice for small projects because it is simple and easy to understand. It is also easy to implement and maintain. However, it has some limitations. The most significant limitation is that it cannot handle large projects very well. Another rule is that it is not very flexible and can be hard to change.

The third factor is the development team's expertise. If the group is experienced in PHP, then using PHP for the project might be a good choice. However, if the unit is inexperienced in PHP, then Java or Python might be a better choice.

4. Implementing the solution

After the software architecture is chosen, the next step is to implement the solution. This involves writing code to create the software application. Depending on the size and complexity of the project, this step can take anywhere from a few weeks to several months. In some cases, this can be done by a single software engineer. In other cases, it may require a team of software engineers working together.

Common problems that can be encountered when writing software code include:

– errors in the code (bugs) which can cause the software to not work as intended

– code that is not efficient and runs slowly

– code that is not well organized and is difficult to read or understand

There are a few solutions that can help address these typical problems:

writing code that is clear and easy to understand
peer review of the code
following software development best practices
testing code thoroughly

Testing can be done manually by running the software and trying out all the features. Alternatively, automated testing tools can be used to test the software. Automated testing is generally considered more reliable, as it can test more scenarios in a shorter amount of time but requires investment in setting up the automation.

5. Deploying the software

Once the software has been written and tested, it must be deployed. This usually involves setting up infrastructure, such as servers, and configuring the software to work in that environment so that users can access it. Deployment can be a complex process and often requires the help of a server administrator. However, there are multiple advantages with Cloud-based architectures that help.

One of the most significant advantages of Cloud based software architectures is that they are inherently scalable. As your business grows, you can add more resources to your account, and the software will automatically adapt to use them. This eliminates the need for you to purchase and maintain expensive hardware and software infrastructure, which can be a significant expense for businesses of all sizes.

Cloud-based software architectures also make it easy to collaborate with team members from around the world. All you need is an internet connection, and you can access your files and work on projects with team members no matter where they are located. This can save time and money on travel costs and make coordinating complex projects easier.

Additionally, Cloud-based software architectures are often more secure than traditional software installations. Your data is stored in a secure data center rather than your computer, making it less likely to be hacked or stolen. Knowing that your information is safe and secure can give you peace of mind.

The cloud-based software development industry has boomed recently, with many providers of cloud-based services. The top cloud-based software development services providers include

Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). Each of these providers has different strengths and weaknesses, making it essential for companies to select the provider that best meets their needs.

AWS is the largest cloud-based service provider, offering a wide range of services, including computing, storage, database, networking, and application services.

AWS is also the most expensive provider, and its services are not always as user-friendly as those offered by other providers.

Microsoft Azure is the second largest provider of cloud-based services. It offers a wide range of services similar to AWS but lower prices. Azure is also more user-friendly than AWS, making it a good choice for companies new to the Cloud.

Google Cloud Platform is the smallest of the three providers but offers some unique features worth considering. For example, GCP provides low-cost options for starting up new applications and websites, and its machine-learning capabilities are among the best in the industry.

6. Maintaining the software

Once the software is deployed, it will need to be maintained. This includes patching security vulnerabilities, adding new features, and fixing bugs. Once code is deployed, it needs to be held to ensure it continues functioning as intended. This typically involves tracking changes to the codebase and ensuring that any new updates are correctly implemented. In addition, code needs to be tested regularly to ensure that it still meets all of the requirements and standards set for it.

The most common tools used to maintain code once deployed in the field are revision control systems such as Git or SVN. These systems allow developers to track changes to their code and merge changes from multiple developers into a single, unified codebase. This can help prevent coding errors and ensure that all developers work from the same source code.

7. Getting help

There are many ways to get help with software development. One way is to join a software development community, such as an online forum or an offline meetup group. There are also many software development companies that offer consulting services. Finally, several books and websites offer advice on software development. Of course, Skillion can help too!

A lot goes into software development, from requirement gathering to design to deployment. It can be daunting, but many resources are available to help. Skillion's core value is that we offer Program management integrated as a critical part of a software development project. This helps to ensure software development companies remain focused on their business goals.

(*) The numbers in the Case Study are illustrative only and not intended to be accurate.

Pete Cooper is a CEO and Program Manager with 20+ years of diverse experience as a Program Manager and eight years as a CEO. His career started as a design engineer and grew to the executive level. He has worked in various fields, including Software Development, AI/ML, Product Design Aviation, App development, RF design, Electronics Design, Mechanical Design, Telehealth, Semiconductors, IoT, and more.

Pete is a thought leader in applying Program Management methodology as a CEO. He has received recognition for overseeing complicated projects in various sectors. He holds an Engineering Degree, MBA, an Airline Pilot's Licence, and multiple Program Management Certifications, including FAIPM.

At Skillion, where Pete is the CEO, we pride ourselves on our ability to implement and educate Program Management woven into our customer projects. If you need more than just a technical solution but need it managed end to end, don't hesitate to get in touch with us today to learn more.

info@skillion.tech

Pete Cooper