WHITEPAPER

Product Engineering

Designing and Building Winning Solutions

A Comprehensive Guide to Product Engineering

Part 2 Of 4

Gleecus TECHLABS INC

Forward Thinking Digital Innovation

Table of Contents

04 **Executive Summary**

05 **Software Product Engineering Process**

05 **Setting up the Roadmap**

05 *Figuring out the bigger picture*

05 *Breaking down the goal into epics*

06 *Breaking down the roadmap into sprints*

06 *Assigning teams to sprint*

06 *Integrating the roadmap with project management tools*

07 **Implementing Agile**

07 **Designing a Software Product**

08 **Building a Software Product**

08 *Translating design into code*

08 *Programming languages and tools*

09 *Writing modular and reusable code*

09 *Adhering to coding standards and best practices*

09 *Version control*

09 *Code reviews and collaboration*

Gleecus TECHLABS

Table of Contents

09 *Testing code snippets*

10 *Continuous integration and deployment*

10 *Code optimization and performance tuning*

10 *Documentation*

11 **Checking for Quality**

11 *Functional tests*

12 *Non-functional tests*

13 **Winning the Product Owners**

14 *Highlight industry trends*

14 *Project the ROI*

14 *Perform a competitive analysis*

15 **Conclusion**

Executive Summary

The product engineering phase deals with the product development process, the product design process, the roadmap setting, setting up the Agile or Lean process, the QA process, and the software engineering involved.

93% of Agile organizations reported better customer satisfaction, 76% reported better employee engagement, and 93% reported better operational performance.

This phase requires collaboration between multiple teams such as product owners, developers, testers, and designers. During this phase it is important to have clear communication between stakeholders to ensure that everyone is on the same page regarding the goals and objectives of the project.

86% of employees and executives cite lack of collaboration or ineffective communication for workplace failures.

The software product development process, often known as SDLC (Software Development Life Cycle, is a structured approach to building a new product. This consists of design, planning, roadmap setting, prototyping, and testing. There can be several approaches to software development like waterfall, spiral, agile etc. Modern software development approach mostly adopt the Agile process.

An engineering roadmap summarizes the goals and plans of a product development team keeping in mind the resources available and development philosophy to be followed. This whitepaper deals with the product development process and setting up the roadmap for design to QA process. It attempts to alleviate the scare of failure among product owners through strategic goal setting using the product engineering roadmap.

Software Product Engineering Process

The software product engineering process is a structured approach to building a new product. This consists of design, planning, roadmap setting, prototyping, and testing. The software product engineering process is often known as SDLC (Software Development Life Cycle). There can be several approaches to software development like waterfall, spiral, agile etc. Modern software development approaches mostly adopt the Agile process.

Setting up the Roadmap

An engineering roadmap summarizes the goals and plans of a product development team keeping in mind the resources available and development philosophy to be followed. Setting up an engineering roadmap consists of several steps:

Figuring out the bigger picture

At the onset of preparing a roadmap it is important to understand the high level goals of the product. It is important to ask the necessity of the roadmap and the definite role it will play throughout the product lifecycle. A clear vision of what defines success is required.

Breaking down the goal into epics

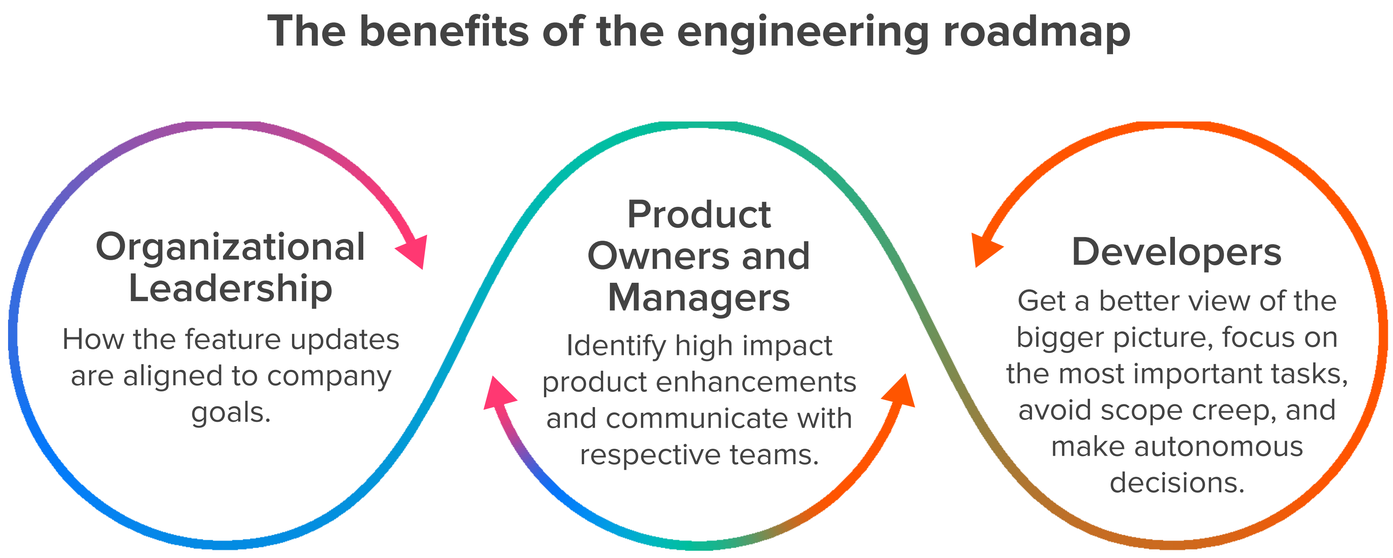
Next is the stage to break down the high-level goal into smaller themes or epics. This will help in translating the high level vision into tangible goals. At this stage it is important to prioritize which epics need to be completed first.

Breaking down the roadmap into sprints

Modern software development approach adopt an iterative method to develop software for example, Agile methodology. In Agile methodology each epic is broken down into several tasks and the tasks are allotted to sprint which is a very short span of time ideally fortnightly or at most a month.

Assigning teams to sprint

Once the boundaries of a sprint are set, assign teams to individual sprints. The engineering roadmap is broken down into lanes so that each team and engineer can figure out their lane and what they will work on.



Integrating the roadmap with project management tools

The roadmap shows what each team member will work on and their objectives. Teams may be comfortable with using particular software tools to assign and monitor day-to-day tasks. One of the most popular project management tools is Jira. Integrating project management tools helps to prepare a to-do list while also watching the high-level overview of the project goals.

Implementing Agile

Following the Agile or Lean process while creating a roadmap comes with many advantages. The development team gets into customer interaction at a very early stage. Iterative process makes the development very fast keeping frequent checks on quality and bug fixing. The development team and the organization at large goes through a learning process as it keeps getting feedback on the incomplete prototype and rebuilds with this information in mind.

> *The billing cycle of Agile also follows a flexible approach instead of having a fixed price. There are several billing models that agile offers:* ***"Time and Materials", "Agile Fixed Price", "Agile Mini-Fixed Price".***

Agile Mini-Fixed price offers the advantages of a fixed price model from the customers end while keeping very small billing units called stories to offer the lean nature.

Scrum is an Agile methodology which provides product teams with iterative and incremental framework. The sprints in Scrum are very short around 2-3 weeks and a sprint team consists of 6-9 people. Sprints hold team members accountable on a regular basis in a measurable way.

Designing a Software Product

Once the roadmap has been set, the project architect starts designing the software architecture. A software architecture is an abstraction of the runtime elements of a software system during some phase of its operation.

Designing software architecture starts with abstraction of components, planning the interaction with different components of the software. The way a component integrates with the product and specific states it fulfills need to be assessed. Characterizing various aspects of a component for example, in the case of a payment gateway, how it integrates with the system or does it maintain any state, in a rather abstract way is a good starting point.

Abstraction is at the heart of a software architecture and the level of abstraction increases as the complexity of a product increases.

Software solutions need to be designed keeping maintainability in mind. Design principles serve as a guide to build application from discrete components that are loosely coupled and communicate with each other via explicit interfaces or messaging systems. Software architecture design is therefore influenced by the principles of encapsulation, dependency inversion, encapsulation.

Building a Software Product

This phase, also known as the software construction phase, follows the requirement gathering, analysis, and design phase and precedes the testing, and deployment phase.

The objective of this phase is to transform approved architecture and design into a functional prototype. In order to achieve that the processes and **functionalities involved with this stage are:**

Translating design into code

Developers take the system design, architecture, and specifications created in the previous phases and translate them into a programming language. They write lines of code to implement the desired features, algorithms, data structures, and user interfaces.

Programming languages and tools

Developers use programming languages such as Java, C++, Python, JavaScript, or others, depending on the requirements of the project. They also utilize Integrated Development Environments (IDEs) or text editors that provide features like code highlighting, auto-completion, and debugging to aid the coding process.

Writing modular and reusable code

To enhance maintainability and extensibility, developers strive to write modular and reusable code. They break down the software into smaller components or modules that perform specific functions, making it easier to understand, modify, and maintain the codebase.

Adhering to coding standards and best practices

Following coding standards and best practices is essential for producing high-quality code. These standards define guidelines for formatting, naming conventions, code documentation, and overall code structure. Adhering to these standards ensures readability, consistency, and facilitates collaboration among developers.

Version control

Developers utilize version control systems like Git to manage the codebase efficiently. Version control allows for tracking changes, branching code, merging changes from multiple developers, and reverting to previous versions if necessary. It also enables collaboration and provides a centralized repository for the codebase.

Code reviews and collaboration

Code reviews play a significant role in the coding phase. Developers share their code with peers or senior developers for review. This process helps identify potential issues, improve code quality, and ensure compliance with coding standards. Collaboration tools and communication channels are used for effective collaboration between team members during this phase

Testing code snippets

Developers often test small portions of code, known as code snippets, to verify their correctness and functionality. These tests can be done through unit testing frameworks that allow developers to write test cases specifically targeting small sections of code.

Continuous integration and deployment

In modern software development, continuous integration (CI) and continuous deployment (CD) practices are adopted. CI involves integrating code changes frequently into a shared repository and performing automated builds and tests. CD automates the deployment process, allowing software to be deployed to production environments quickly and reliably

Code optimization and performance tuning

Developers also focus on optimizing the code during this phase. They identify areas where the code can be improved in terms of performance, efficiency, and resource utilization. Techniques like algorithmic optimizations, caching, and minimizing database queries are employed to enhance the software's performance.

Documentation

Alongside writing code, developers document their work to provide guidance and understanding for future developers or maintenance teams. This documentation includes comments within the code, code documentation generated by tools like Javadoc or Doxygen, and external documentation explaining the software's architecture, APIs, and usage.

Checking for Quality

Ideally a product is tested at all stages of the product development lifecycle right from the high level test scenarios accompanying the requirement analysis stage to regression test post release.

Tests are further classified as functional and non-functional tests. While functional tests are done to check the functionality of a feature or system, non-functional tests are done to check other things which go into building value for the product like performance ability, load capacity, user acceptance etc. Some of the common tests pre-release are:

Functional tests

Unit testing

This type of testing is performed on individual units of code. It helps to ensure that each unit of code is working as expected.

Integration testing

This type of testing is performed on groups of units of code. It helps to ensure that the units of code work together as expected.

System testing

This type of testing is performed on the entire system. It helps to ensure that the system meets all of its requirements and works as expected.

Acceptance testing

This type of testing is performed by the customer or end user. It helps to ensure that the product meets the customer's needs and expectations.

Performance testing

This type of testing is performed to measure the performance of the software. It helps to ensure that the software can handle the expected load and that it meets the performance requirements.

Reliability testing

This type of testing is performed to measure the reliability of the software. It helps to ensure that the software is stable and that it does not crash or have errors.

Security testing

This type of testing is performed to identify and mitigate security vulnerabilities in the software. It helps to protect the software from unauthorized access and malicious attacks.

Usability testing

This type of testing is performed to measure the usability of the software. It helps to ensure that the software is easy to use and that it meets the needs of the users.

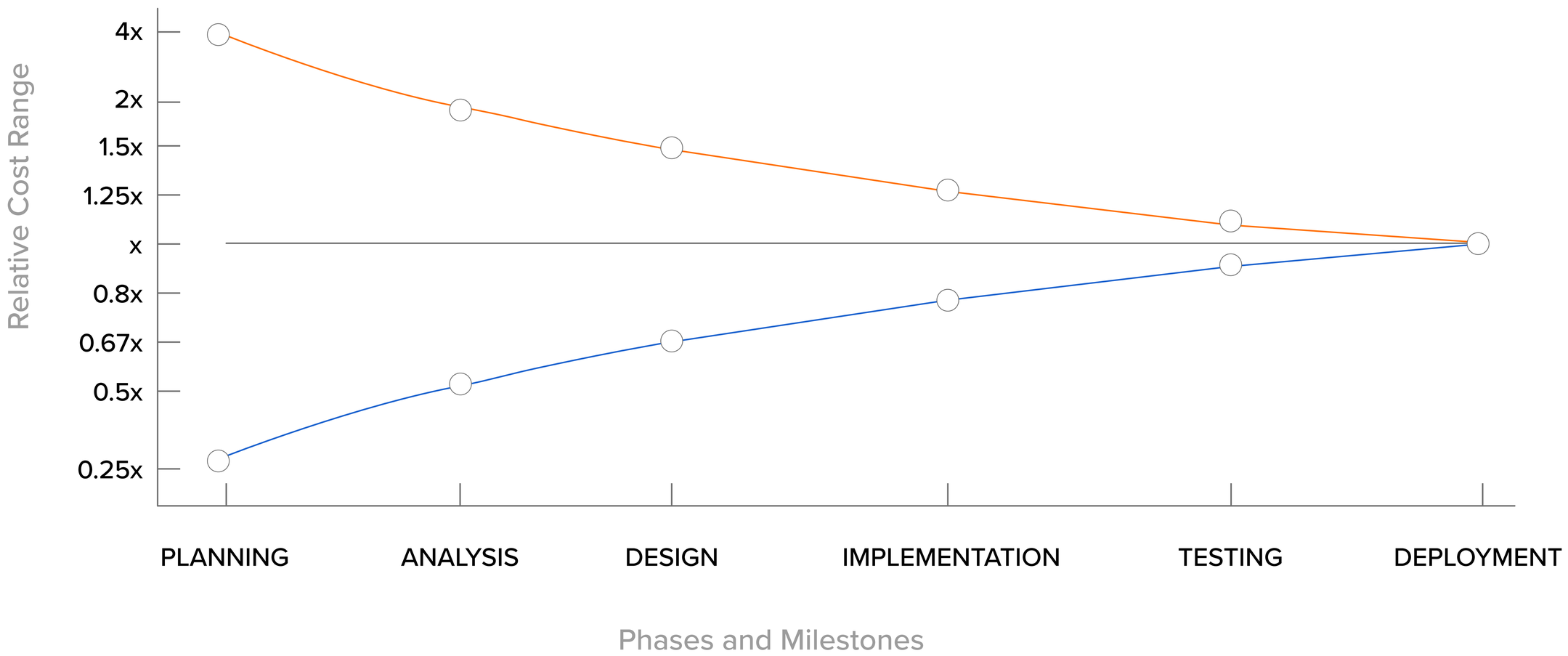
Winning the Product Owners

Seven out of 12 of the most famous failed projects, state cost overrun and delays in delivery as major problems that led to the failure. It means that about 60 percent of the project success depends on meeting the cost and time estimates, provided by the engineering team.

Often an interesting product idea may not see the light of the day as the product owner might be skeptical about its success. Even after starting the product engineering cycle it may face roadblocks or get shelved if some of the C-suite executives are not confident about the business proposition of a product.

It is important to allay the fears of a product owner and bring him into an active role in support of the product. A product owner must be convinced that an MVP or some of its features may fail at launch and there is a lot to learn and improve from that. Building an innovative product needs to go through countless iterations and as it nears to becoming the desirable product the ROI will improve. This is also reflected in the Cone of Uncertainty introduced by Barry Boehm in his book Software Engineering Economics. It is practically impossible to define the scope of work early in the process. Usually, the clearer the project requirements become, the more accurate the quote will be.

The Cone of Uncertainity



Highlight industry trends

The product owners should be made aware of industry trends through various facts and figures in support of the product. Facts and figures can be sourced from various channels like publications from survey agencies, social media trends, and customer feedback.

Project the ROI

Projecting a possible ROI with defined metrics will generate interest about the product among the business leaders. The ROI can be defined through some metrics such as the cost of building the MVP, potential size of the target market, customer acquisition cost, time to market, pricing model, churn rate etc.

Perform a competitive analysis

A competitive analysis can help to establish the uniqueness of the product and present it as a potential solution to convert a competitor's customer. Competitive analysis can be done by reviewing their product, analyzing social media, and watching their product page reviews. Watching for digital disruptors in the same industry can also point out how they are changing the customer experience and convince potential product owners.

Conclusion

The product engineering phase is where the IT team actually start to get its hand dirty. This is when the project is broken down into smaller and clearly defined job. It has also got a roadmap and timeline of achieving those milestones. This is the trust building phase for the product owners through actual demonstration of work. At the end of this phase the product is ready to taste the waters in the market and gather feedback on what the end user approves or disapproves of. Feedback gathering leads to the scaling stage where it is decided what features need to be scaled up, brought in, or discarded. This phase sets the future of a product from the core.