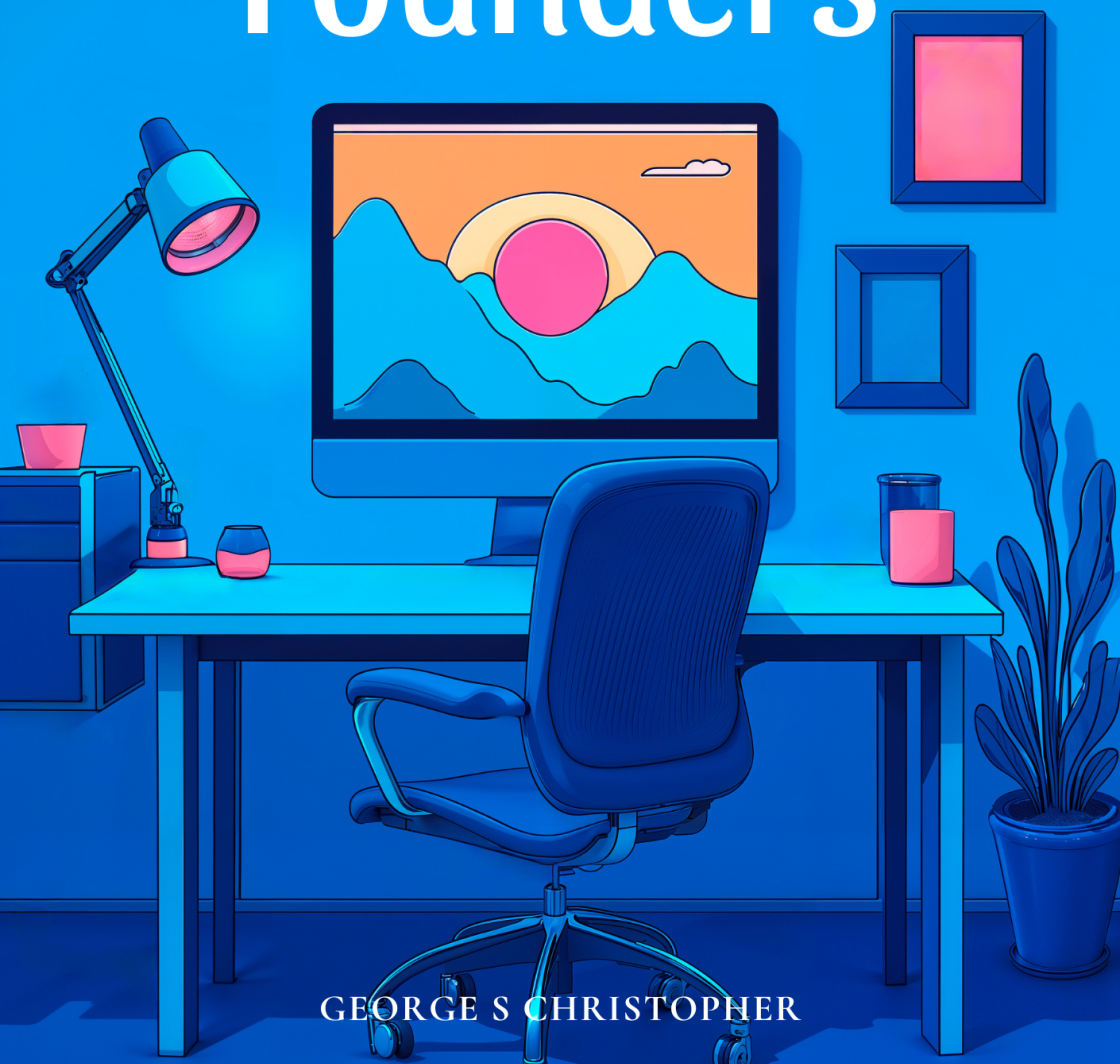
Tech for Non-Tech Founders



GEORGE S CHRISTOPHER

Software is a great combination between artistry and engineering.
- Bill Gates

Table of Contents:

Tech for Non-Tech Founders

1. Is this your story?

You are a founder of a beautiful company. You could be in working in education, logistics, supply chain, content delivery, research publishing, retail, finance, insurance or any other business domain.

You have used your industry and domain expertise and the wonderful connects that you have created to build a core company that solves the problem of customers in that domain.

You have many paying customers, gained a lot of trust, generated a lot of relationships and company is going on smoothly because all the activities happening are under your area of expertise. Now that you have paying customers, steady revenue and good forecast, you want to scale.

The first thing that comes to mind when you want to scale is software or apps. Software is not just one of the ways but in many cases it is the only way to scale your services or products across your region or the country or the world.

But the moment you start creating software you start moving into a domain where you move out of control.

You start to hire people, technical experts and they decide something called as a tech stack, they create a product design, they create a product roadmap, put together a project plan and delivery milestones.

Now at this moment, none of the above makes sense for you and you feel out of place and out of control but you will proceed anyways in the interest of the project and the overall scaling and business objectives. Software being like air cannot be straight away measured or metricized.

2. The Software Paradox:

> *Software applied to an efficient process will magnify the efficiency. Software applied to an inefficient process will magnify the inefficiency.*
> *-Bill Gates*

Your software product is being released, and you begin to realize that what looked perfect on paper isn't as efficient or effective in real-world use as you had expected.

But you have promised to customers on the deliveries and dates. So you proceed with the release to the customers and proceed to the next work on the product and a few releases go like this.

Now the product deployed in customer business starts to fail and causes disruption in customer's business. This causes escalations and you will deploy your entire strength of developers to address the customer issue as it impacts a live business and a production issue.

This will cause the regular release cycles and delivery plans and product roadmap to collapse completely and the product evolution stops.

Now the other customers will start escalating that their promised features are not coming and the entire development team becomes a fire fighting team.

> ***During Product Start:***
> *Innovation - 90% and Fire fighting - 10%*
>
> ***After a few customers and deployments:***
> *Innovation - 10% and Fire fighting - 90%*

This causes customers to be unhappy, revenues and growth are affected, owners / investors are alarmed, top management is in a bind and developers are in high stress environment working day and night.

This is exactly the opposite of the environment needed to create and innovate.

3. No one is happy:

In any transaction, the aim is that both the parties should be happy. Both or all the people involved in the transaction should be happy. In a good transaction all are happy.

In a scenario, if one person cheats another person in a transaction, the person who got cheated might be sad but the person who gained will be happy. So even in a bad transaction one person is happy.

But how come, in our development environment, no one is happy.

The customers are not happy as they are not getting the business impact because of our product. The management is not happy as revenue and brand and customer relationship is affected. Developers are not happy as the environment has become like a furnace with a high stress environment.

The environment becomes a stressful place and even though all stakeholders come with sincere intentions and with hard work and honest efforts, somehow all of them go as waste and everyone is stressed and unhappy.

How did a place, with all good intentioned and talented people in the ecosystem, including customers, management and developers, become toxic and unhappy for all involved? It is because there are some more missing pieces in the puzzle. Those are the 3Ps of software.

4. 3Ps of Software Engineering:

The three important Pillars or 3Ps of Software engineering are the following:

1. People
2. Process
3. Platform

> *Great software comes from the right people, following the right process, using the right platform to build the desired final product.*

5. People:

The first thing you need to do is to check if you have the right people in your team in terms of roles, capability and culture.

5.1 Roles:

The different technical team members that you might need for software engineering are the following:

1. Business Analyst - Documents the business case or the product requirement specifications. They define 'what has to be done?'.

2. Designer - Creates the UI/UX for the product. Creates the brand guidelines, product overall visual theme. They define 'how it has to be done?'.

3. Developers :

3.1 Front end developers - Develop the front end systems using the UI/UX provided by the

designers. They create front end apps for web, mobile, tab, TV, watch and other relevant devices.

3.2 Back end developers - Create the backend of the product which has the business logic and persistence logic. The business services are exposed to the front end pluggable endpoints called APIs.

3.3 Database developers - Creates the Database tables, relationships and mappings to store the data of the product permanently. The backend developers connect to the database to do the CRUD (Create, Read, Update and Delete) operations.

4. Quality Analyst - Tests and validates the product against the requirements provided and the test cases created. They are responsible for both Functional and Non-Functional (Performance, Security, Load etc) Requirements.

5. Server Engineer - Manages the servers in physical location or on the cloud. Manages the

deployments, CI/CD pipelines, containerization, backups, load balancing, failover, disaster management etc.

6. Project Manager - Defines the product roadmap and release milestones in consultation with the customer, management and business analyst. The delivery roadmap will have what will be delivered and when and the clear scope of each release.

5.2 Capability:

1. Skills - Ensure that each member allocated to each role is a specialist in that area. The expertise of each member should be constantly and consistently verified using internal and external certifications.

2. Communication - The team members should be able to communicate and collaborate with each other seamlessly. Open communication and a transparent environment gets work done faster and easier.

3. Collaboration - The team should work together for successful releases. All teams should work only 3 goals each Quarter. The goals should be simple, easy to measure and transparent to track for everyone.

5.3 Culture:

1. Attitude - The attitude of each team member is very important as even one person with a bad attitude can pollute the team.

2. Self Improvement - All team members should constantly be in the pursuit of excellence and improve themselves everyday. There should be career growth path for each employee.

2. Leadership - The team members should be able to always do the role of one level up. Leadership should be measure by the ability to create more leaders. Else they are just managers.

6. Process:

The second thing you need to do is to check if you have the right process for the people in your team to work efficiently and effectively.

The following are the top process structures that you should have for your team to run like a well oiled machine. Else even if you have a great team, they will not be able to ship great products.

1. Product Roadmap - The product should have a clear roadmap for the next quarter, year and 3 years. If this is not there, then there is no common goal for all team to come together. This is the overall scope and plan for the project.

2. Release milestones - The upcoming releases and their dates. I recommend having releases every 15 days. 15th of the month and last day of the month. So you have two release cycles or sprints every month - 1-15th and 16th - 30th (or end of the month).

3. Bug Tracking - When the release is done, the internal QA or customer will have feedback. All of them are tracked in a bug tracker system. These will be prioritized and added to a release.

4. Release Scope - The scope of work that is planned for each release that is upcoming every 15 days. So if there are 500 overall roadmap items, we need to split it for each 15 day release based on priority. In addition the release scope can have bugs also. So a release can have a combination of roadmap items and bug items.

5. CI/CD - Continuous integration and continuous deployment is a must for any software development process. Whenever there is a commit, the build will happen and the deployment will be done. Or it can happen at regular intervals or at night time.

6. Regression Suite - The entire platform should have a clear regression test suite, that will run the full tests covering all functionality

and give a test report. The code coverage of test should be more than 95%.

7. RTM - There should be a requirement traceability matrix where you can map a requirement to its test cases. Each requirement should have its own list of test cases. If a functionality fails, we should be able to pull up the test cases of the function to see what was missed and add more test cases to make it robust.

8. NFR Testing - There should be regular Non-Functional requirement testing and audit for performance, load, vulnerabilities, security, disaster management, load balancing and failover.

9. Project Management - The overall process flow should be managed seamlessly connecting all stakeholders and ensuring the deliveries are done on time, fulfilling the scope and with the highest quality. The KPIs for user engagement should also be tracked by the PM.

10. Client Engagement - The customers will have clarity on the product roadmap ahead of time and also will have a briefing after each release on what was released. The clients can upvote to prioritize a feature in the roadmap.

7. Platform:

The third pillar of software development is the platform. The platform is the set of tools and technologies that are needed for building efficient software. I have given some references for you to get started, but this not a full list.

7.1 Software and Tools:

1. Design:
Figma
Adobe XD
Sketch

2. Web Development
React
Angular
Vue

3. iOS / Android Apps
Swift and Swift UI
Kotlin and Jetpack Compose
React Native / Flutter

4. Backend Development
Java / Springboot
NodeJS
Python / Django
PHP / Laravel

5. DB
MySQL
PostgreSQL
MongoDB

6. PM / BA
Notion / Zoho Docs / Office 365
JIRA / Zoho Project / Asana
Firebase / Mixpanel / AppsFlyer

7. Coding Standards:
ESLint / Prettier
SonarQube
New Relic

7.2 Deployment Architecture:

The architecture defines how all you technologies and tools come together and work together. The architecture defines how you system will receive requests from the front end of other systems and process them and give response. The application layers, inter-layer communication, containers, servers, deployment and other NFR specifications are covered in the architecture.

7.3 Non Functional Specifications:

The non functional specifications of a software are very important and they also define how a product will work in different situations:

1. Load
2. Performance
3. Security
4. Vulnerability
5. Reliability
6. Load balancing and Failover
6. Disaster management

8. Video Overview:

Incase you need to get more details on the overview of the software architecture, the tech stacks and database, you can view the following videos. This was meant as a educational series of young people joining the programming career path and it will help you understand what happens under the hood of software applications.

Basics of App Development:
https://youtu.be/YYcyoqSs9CM?si=2LIsysMhw_iEC3kk

Basics of Database:
https://youtu.be/DdRN2nZnGwE?si=WYKdj6SE8wdjJsOJ\

Now that you know the basics of software engineering and how it all works together, you should ask the following 10 questions to your team at regular intervals to see how the health of the product is at any point in time.

9. The top 10 questions for every quarter:

1. Is the full product roadmap available and is the release roadmap for upcoming quarter published?
2. Have all releases (every 15 days) in the previous quarter released on time to the customer?
3. Have all the releases contained 90% or more of the planned scope of work for that release?
4. Did the customer accept 90% or more of the release items?
5. Do you have a plan for making bug tracker count as zero?
6. Is the percentage of code coverage in regression suites more than 95%?
7. Was there server or service downtimes (scheduled or unscheduled) in the last quarter?
8. Was there any CI/CD build failures?
9. Was there any security, load, vulnerability or other threats?
10. Is everyone improving (1 external certification per quarter), happy (stress free and enjoying work) and excited (looking to innovate) to work on the product?

10. Top 10 KPI Metrics:

The health of a person is measured by metrics like temperate, blood pressure, heart rate etc. The same way the health of your software is measured by metrics which are given below.

If you do not measure them then you cannot manage the development process.

User Activity & Engagement

1. Daily Active Users (DAU)
Number of users using the system daily
Measures adoption

2. Monthly Active Users (MAU)
Unique users per month
Measures overall reach

3. DAU / MAU Ratio
Stickiness metric
Are users coming back daily?

4. Average Session Duration
Time spent per session
Indicates engagement depth

5. Sessions per User
How often users log in
Measures dependency on system

Product Usage & Efficiency

6. Feature Adoption Rate
% of users using key modules (HR, Finance etc)
Shows which ERP modules matter

7. Task Completion Rate
% of tasks completed successfully
Example: invoice created, order processed
Measures usability + effectiveness

8. Bounce Rate
Users leaving quickly without action
Indicates poor UX or onboarding issues

Business Impact Metrics

9. Transaction Volume
Number of business operations (e.g., invoices, orders, entries)
Measures real usage value

10. Error / Failure Rate
Failed transactions or system errors
Measures system reliability

10. Conclusion:

Software is not rocket science. But at the same time, it is not as simple as it looks either. Software engineering is a science that can be perfected using the three pillars detailed above.

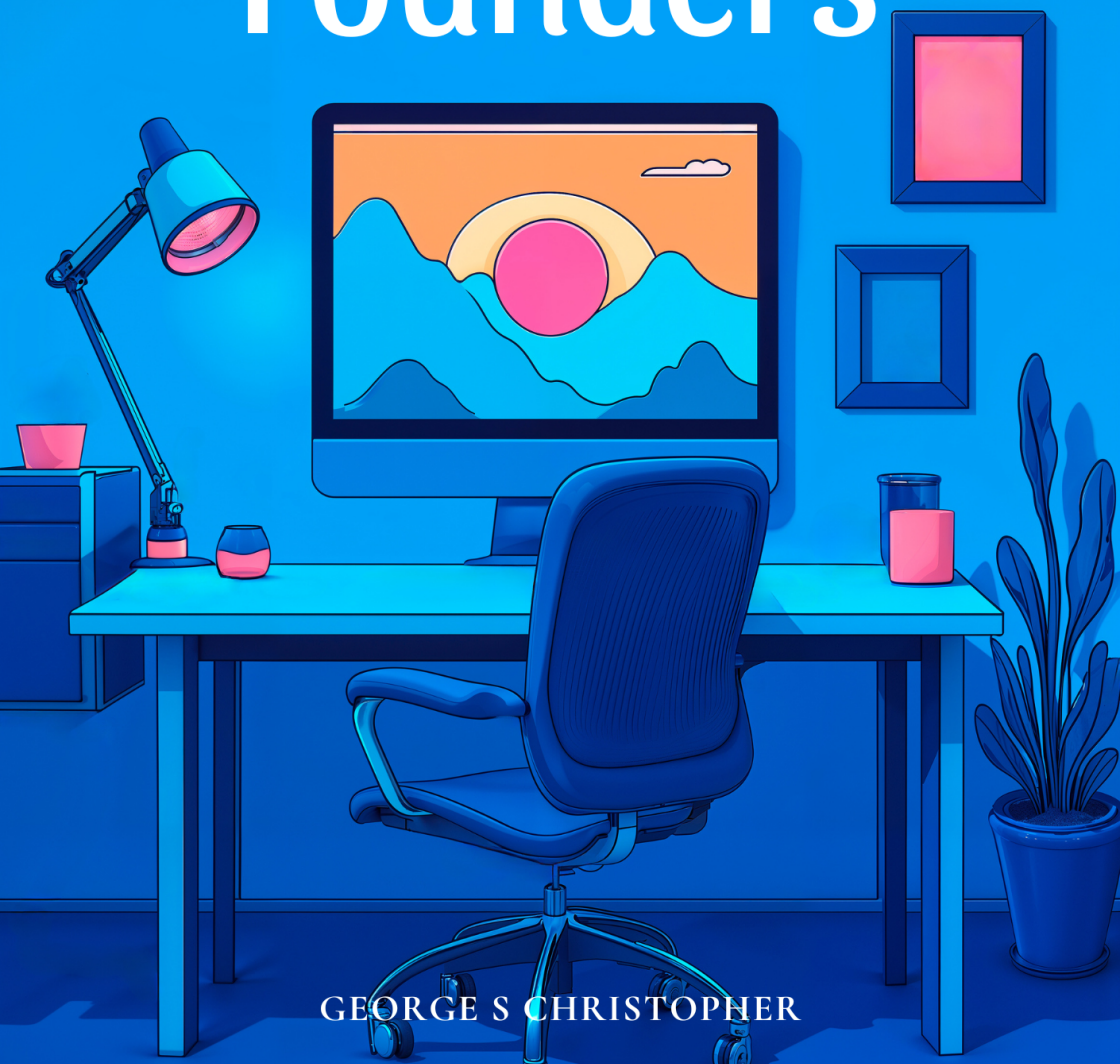
If we can set up a work place with these three pillars then great software can be created that can become the delivery rocket that takes not only the company but the lives of every single person and stakeholder to the next level.

Notes:

Notes:

Why do most developers fear to make continuous changes to their code? They are afraid they'll break it! Why are they afraid they'll break it? Because they don't have tests.
— Robert C. Martin

Tech for Non-Tech Founders



GEORGE S CHRISTOPHER