

The Product Minded Software Engineer

Cem BAŞARANOĞLU

To all of my family,

“I start where the last man left off.”

Thomas A. Edison

Chapter I	10
From Past to Present	11
The Product	14
Conflict of Interests	17
Common Characteristics of a Well-Thought Product	21
Functionality	22
Usability	24
Efficiency	26
Flexibility	27
Reliability	28
Maintainability	29
Portability	30
Integrity	31
Chapter II	32
The importance of understanding the problem	33
Rephrase the Problem	35
Expose and Challenge Assumptions	36
Chunking	37
Use Multiple Perspectives	39
Use Effective Language Constructs	39
Make It Engaging	40
Reverse the Problem	41
Gather Facts	42
Clarify the Solution Space	43
Impact Mapping	43
The Goal: Why?	45
The Actors: Who?	46
The Impact : How?	47
The Deliverables: What?	48
Test Your Solution Space with Data	54
What is a Data-Driven Approach	54
The Benefits of Data-Driven Decision Making	55
Chapter III	57
Key Characteristics of Product Minded Software Engineers	58
Start with WHY	58
Protest by Nature	59
Think Twice	60
Think outside the box	61
Smooth Communicators	62
Trade-off Juggler	63

Always Patient	64
Edge-Case Master	64
Strike Home with Customers	65
Product Ownership	66
Become a product-minded engineer	67
Discover your company in time	67
Make a strong relationship with your product persons	68
Find who frequently interacts with the customer	69
Don't be afraid to suggest new ideas	70
Talk about tradeoffs	70
Feedback is important	71
Code First vs. Product First	72
Chapter IV	74
The Cygnus method	75
Do Collaboration Sessions frequently	76
The Community Contribution is undeniable	77
Develop your reading habit	78
Develop your Engineering Perspective	79
Pragmatic Thinking and Learning	80
Always consider the context	82
Use rules for novices, and intuition for experts.	83
Know what you don't know	83
Learn by watching and imitating	84
Keep practicing in order to remain an expert	85
Avoid formal methods	85
Capture all of your ideas	88
Strive for good design	88
Use your all senses	89
Step away from the keyboard	90
Change your viewpoint to solve the problem	90
Watch for outliers	91
Be comfortable with uncertainty	92
Hedge your bets with diversity	93
Act as you've evolved, breathe, and don't hiss	93
Trust intuition, but verify	94
Create SMART objectives to reach your goals	94
Plan your investment in learning deliberately and developing your mind	97
Learn from similarities, unlearn from differences	97
See without judging and then act	98
Give yourself permission to fail; it's the path to success.	98

Grab the wheel, you can't steer on autopilot	101
Final Words	102

Prologue

Software engineers have been trying to answer only an essential question since the 1960s;

“How can any product be delivered faster and tested better?”

All the tools and technologies used today actually exist for this reason. This question is as complex as it looks simple and its solution is not linear. Unfortunately, there is no silver bullet in the solution to this problem as Fred Brooks coined in his iconic book "The Mythical Man-Month"¹. The main purpose of the conference² held by NATO in 1968 to address problems related to software was to establish guidelines and best practices for software development.

In this conference, the main factors for the software projects that ended poorly were also discussed. Let's look at the result when we group all the factors defined in this conference;

- Projects that ran over-budget
- Projects that ran over-time
- Software that made inefficient use of calculations and memory
- The software was of low quality
- Software that failed to meet the requirements was developed to meet
- Projects that became unmanageable and code difficult to maintain
- Software that never finished development

The failure story of each of them was reprobed and They tried to draw conclusions from these stories. After a while, The software development industry sought to counter these problems through a variety of efforts:

- The development of new programming languages with features intended to make it harder for programmers to make errors.
- The development of Integrated Development Environments (IDEs) with developer-centric tools to aid in the software development process, including syntax highlighting, interactive debuggers, and profiling tools
- The development of code repository tools like SVN and GIT
- The development and adoption of code documentation standards

¹https://www.goodreads.com/book/show/13629.The_Mythical_Man_Month

²<https://www.scrummanager.net/files/nato1968e.pdf>

- The development and adoption of program modelling languages like UML
- The use of automated testing frameworks and tools to verify expected functionality
- The adoption of software development practices that adopted ideas from other engineering disciplines

As a result of all these solutions, the software industry has constantly tried to sharpen its solutions. New ways are always discovered when solutions are strangled. Many concepts like Agile, XP, Functional Decomposition, etc. that most of us know very well today arose as the solution to the problem I mentioned at the beginning of the article.

However, we have not completely eliminated all these problems despite all the precious techniques or tools. In fact, as software engineers ,we ignored the keystones Dijkstra³ and Hoare⁴ mentioned in their priceless masterpieces;

We must first understand the design purpose to develop good, cost efficient, and reliable software. The best way to understand this design purpose is to think like those who define the purpose.

There are numerous gems that relate to the importance of good design that can lead to good products. In addition, the most important thing is to be able to understand those who define the design purpose by communicating in the common language. A common language requires common ground. Thinking like product persons and other stakeholders is the first step in building a good product.

Humankind has always used some techniques and tools to solve problems since the foragers. As software engineers, we have focused on solving specific problems with many different techniques and tools since the 1960s. However, Many of us have been over-focused on mastering these techniques and tools, so our solutions have strayed from the design purpose of our products.

In this book, I outlined how to think like product persons or other stakeholders – to understand the design purpose – without getting stuck with the code. In addition, I mentioned that Allen Newell and Herbert Simon stated in problem space theory in their iconic book 'Human Problem Solving' , the solution is not linear, it actually has more than one dimension. In software engineering, it is necessary to understand the problem space, namely the product, in order to reach a flawless solution. It also deals with the phenomenon of being a product-minded engineer, which has been ingrained in our

³<https://dl.acm.org/doi/10.1145/355604.361591>

⁴<https://www.gwern.net/docs/math/1996-hoare.pdf>

genes since the existence of humanity but lost over time for many reasons. The “product” mentioned in the book refers to the problem in contrast to its general form.

Who is this book for?

First of all, you do not need to be an experienced or a veteran engineer to read this book.

- If you believe that the codes you write are a tool to improve your product,
- If you are thinking about the product and the problems it solves instead of being stuck in the code,
- If you care about the pain points of your end users,
- If you are wondering about the impact on end users when you develop a feature or write a few lines of code,
- If you have ideas about your product not only technically but also in terms of features,
- If you are questioning what the requests from your stakeholder will add to your product in sessions such as grooming or refinement, and more importantly, how they will affect your end-users

The first chapter stated about understanding the product from past to present and the characteristics of a good product.

In the second chapter, the problem space in which the main concerns and design purposes of the product are addressed and the solution space in which the final form is described in the simplest way are explained.

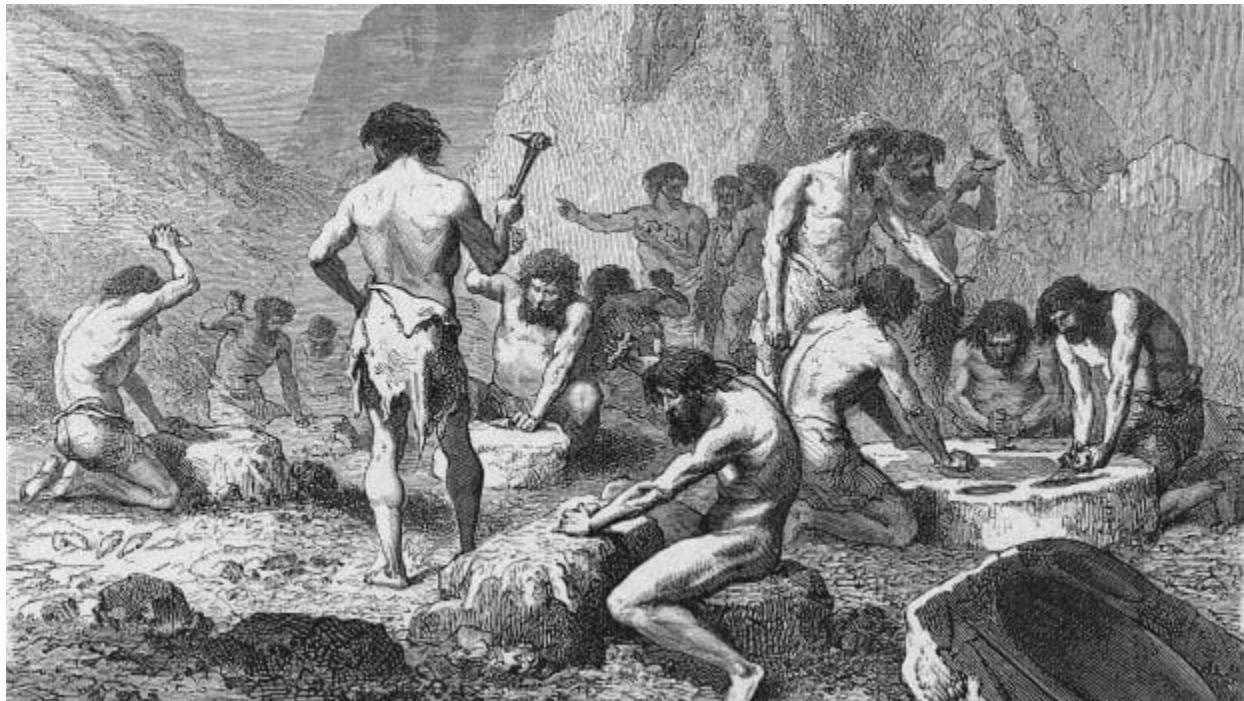
In the third chapter, the characteristics of product-oriented engineers and how to gain this qualification are summarised.

The last chapter is about the cygnus method. This method is a framework that can be used to become a product-minded engineer. It also represents the cygnus constellation in the image on the cover of this book.

Chapter I

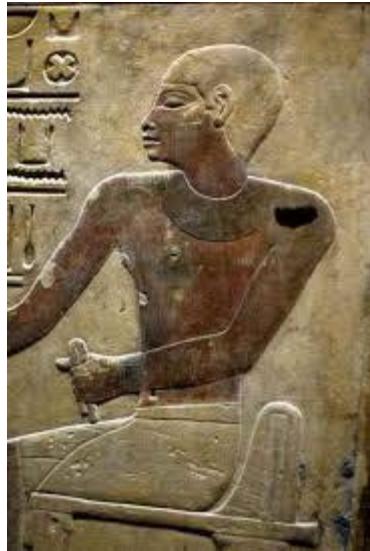
“Understanding the product from past to present and the key characteristics of a good product.”

From Past to Present



The first age

Humankind (human race; mankind, humanity; homo sapiens.) has tried to find a way out of the difficulties it has faced in order to make life easier since the earliest ages of the known history of humanity. Our hunter-gatherer ancestors had to constantly solve problems in order to adapt quickly to nature and survive. Our problem-solving ability is based on our survival instinct and is inherited from our ancestors through their genes. If there were fancy titles like today in these first ages history of humanity, they would probably call these people engineers who solved the problem.



Imhotep

We know that in the early ages there were untitled engineers who made life easier in the societies they lived in with their problem-solving skills, but the first known engineer in history lived in ancient Egypt.

The first engineer in recorded history, Imhotep is thought to have built Egyptian pharaoh Djoser's step pyramid in Saqqara, the earliest large-scale cut stone project. Before Imhotep, pharaohs were buried in mastabas — flat rectangular structures built over an underground burial chamber. But because Imhotep chose to use stone, rather than the traditional mudbrick, he could build much higher structures, essentially stacking six mastabas on top of one another. If you think imperial measurements are a headache, then spare a thought for Imhotep, who used royal cubits while constructing Djoser's pyramid.

A cubit is an ancient unit of measurement considered to be the distance from the elbow to the middle finger. The ancient Egyptians did attempt to standardise this through cubit rods, but the rods could range from 523.5 to 529.2 mm. Not exactly precise.

Djoser's pyramid wasn't the only structure Imhotep designed. It's understood he also worked on a step pyramid for Sekhemkhet, Djoser's successor. However, Sekhemkhet's reign was short-lived and construction was abandoned. We know of Imhotep's involvement in the project thanks to a piece of graffiti on the wall surrounding the unfinished pyramid.

Imhotep certainly wasn't the first to build with stone, but he was the first to attempt such a grand project using the material. The design of Djoser's step pyramid is believed to have set the stage for later pyramids, including the great pyramids of Giza.

I think Imhotep was not the first engineer, although he designed masterpieces. He is considered to be the first engineer, as the historians who made this research took into account the lexical meaning of the word.

The term engineering, and by extension engineer, is as follows:

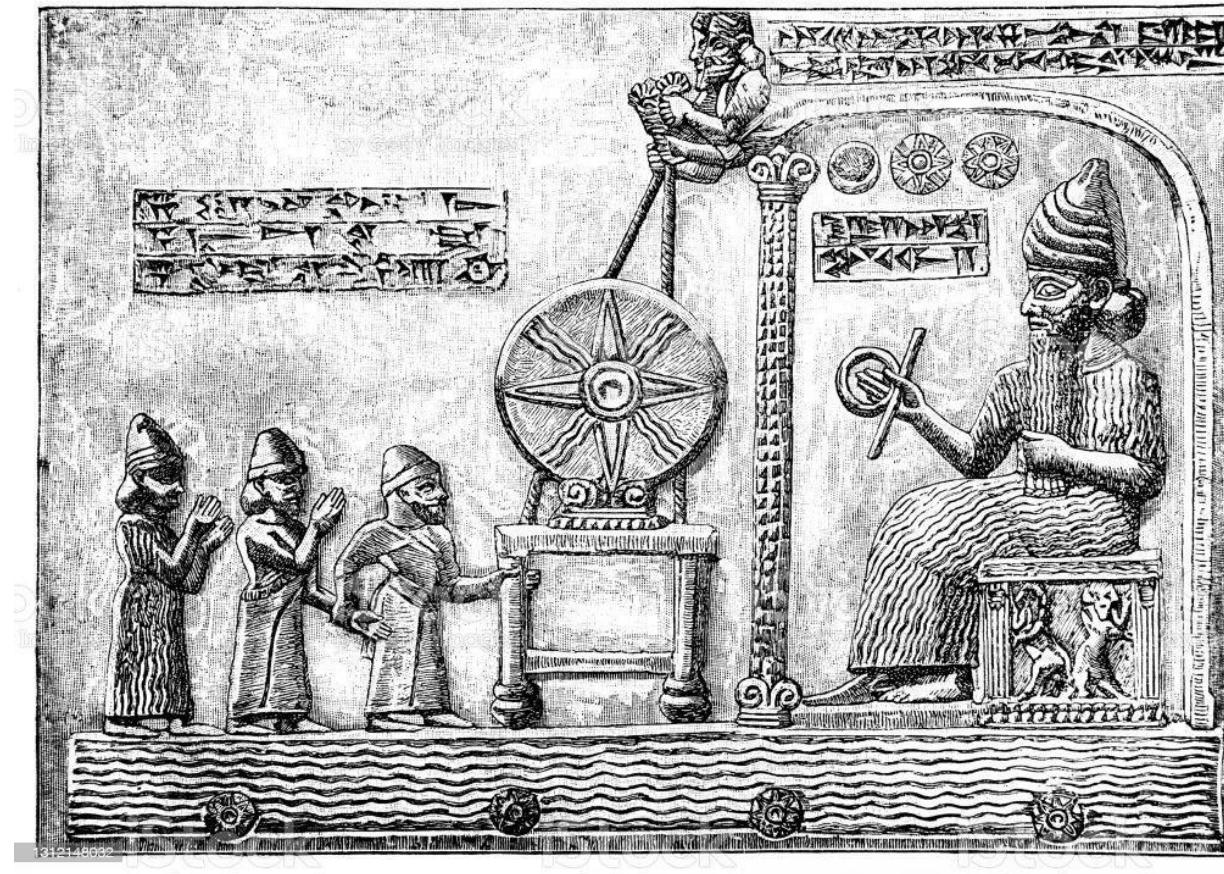
"A person whose job is to design or build machines, engines, or electrical equipment, or things such as roads, railways, or bridges, using scientific principles" - Cambridge Dictionary.⁵

The English word "engineer" itself has a Latin origin. Its root-word "gene" means, in effect, to create, invent, or more specifically bring forth. The first part of the word comes from, in English at least, the word "engine". While we associate this word with a large piece of complex engineering today, it actually has its roots in the Latin terms "ingenium" or "ingeniare". The former means to contrive or devise, while the latter refers to a device or machine. Today, the term "engineer" entered English, well Middle English, through Middle French, and had a relatively broad usage.

Historians may have identified the first engineer with the lexical meaning briefly defined above. As for me, the first engineers were people who developed solutions to hard conditions in the dark early ages of humanity. It is a great chance for us that the problem-solving abilities that form the basis of the engineering philosophy have survived to the present day thanks to their genes.

⁵<https://dictionary.cambridge.org/dictionary/english/engineer>

The Product



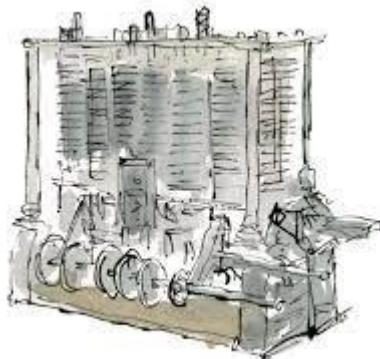
The Sumerians

The Sumerians were the first Mesopotamian civilization who lived in independent walled city-states. They were known to be very rich and inventive, having a varied culture, including farming, trading and playing music. Sumerians⁶ invented or perfected many forms of technology, including the wheel, mathematics, and cuneiform script.

Engineers have developed numerous solutions to solve the problems that societies encountered in daily life and to make their life easier from ancient Sumerians until today. When the Sumerians invented the wheel to facilitate their transportation and use in agriculture in the 4000 BC, they were quite sure of what problem they were solving.

⁶<https://oi.uchicago.edu/sites/oi.uchicago.edu/files/uploads/shared/docs/sumerians.pdf>

Also, the Sumerians changed the future by inventing the abacus⁷, which is seen as the basis of the computers we use today in the 2000 BC. All they really wanted to do was create a table of consecutive columns delimiting consecutive orders of magnitude of the Sixties (base 60) number systems. The only reason they wanted to do this was basically to be able to solve a lot of calculations that seemed complicated to them.



The analytical engine

Charles Babbage was actually trying to find a solution to a problem when he was designing the difference engine⁸ in 1820s and he developed this solution in the 1830s and built the analytical⁹ engine. The analytic engine would never have been invented if Babbage had concluded his research, assuming he had reached the perfect solution when he had completed the difference engine. This could actually cause some sort of domino effect. Ada Lovelace might never work for Babbage's analytical engine and Alan Turing might not have created software theory¹⁰ in 1935 - but none of these great scientists hesitated to perfect the solution of the problem they were dealing with. They found a solution to the problem they were dealing with and allowed us to create products that many of us are developing today to solve many different problems.

⁷<https://www.degruyter.com/document/doi/10.1515/9781501503696-023/pdf>

⁸<https://www.computerhope.com/jargon/d/diffengi.htm>

⁹<https://history-computer.com/charles-babbage-analytical-engine/>

¹⁰<https://www.ipl.org/essay/Alan-Turing-The-History-Of-Computer-Technology-FC5PMTK2FU>

When we consider that, humanity has developed different products to solve many problems since the beginning of time. The software is just a magnificent tool to create various products by solving different problems. Pete Mcbreen described software engineers as "Craftsman" in his iconic book "Software Craftsmanship" published in 2001 because of this.

The definition of the product is made in many different ways in many different sources. The Sumerians named all the solutions they produced to solve the problems of their own society as products. In addition, the product is defined as;

"something produced, something (such as a service) that is marketed or sold as a commodity, something resulting from or necessarily following from a set of conditions." Merriam-Webster¹¹

As software engineers, we can name the solutions that are produced as a result of every problem we solve as a product. The codes that we write may be a small part of a product, but when we look at it from a wider perspective, the problem that we solve refers to a product. We develop products to solve the problems of societies such as hammers produced in the first ages and asteroid anchors produced in our age.

¹¹<https://www.merriam-webster.com/dictionary/product>

Conflict of Interests



The software industry has developed many methods to simplify communication between product and tech persons with the spread of commercial applications. The problem that the industry was basically trying to solve was to build a language that people from these two different fields could speak in common. However, there are still conflicts between these two spaces, especially regarding which space has the authority to prioritise, plan, and deliver the tasks. Also, this completely causes a mind-boggling dilemma called throw-it-over-the-wall.¹²

According to this dichotomy, These people cannot speak the common language for a certain period of time and they no longer make an effort to understand each other. As a result of this, they pass into the phase of obeying, fulfilling, and producing sloppy solutions instead of acting with a common mind for the product.

The most well-known side effects of this are the corruption of the existing culture, the dramatic decrease in the benefit of the product for both the users and the company, and the negative impressions on the customer as the newly designed features are not thoughtfully thought out.

¹²<https://www.forbes.com/sites/forbestechcouncil/2018/12/14/how-to-resolve-8-common-conflicts-between-tech-and-sales-teams/?sh=73419e5271a4>

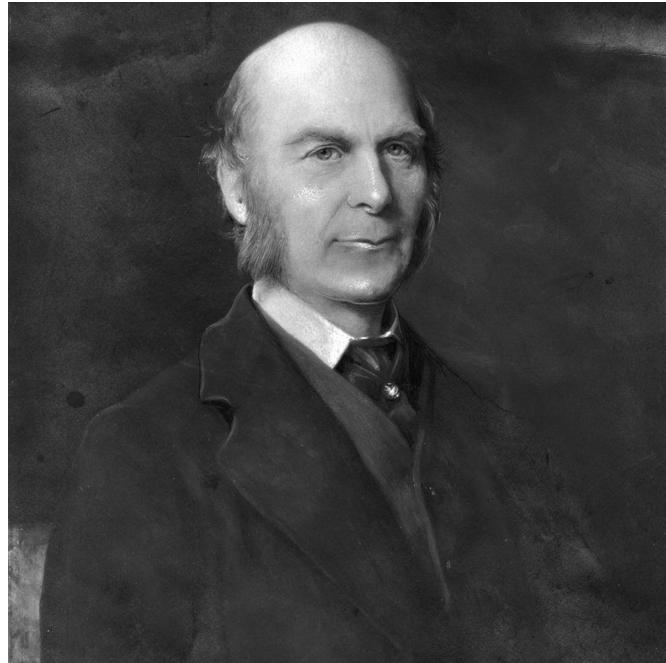
In fact, the most important purpose of being a product-minded engineer is to resolve the conflicts between these two spaces and develop high-quality and reliable products. It is also the best way to put our conflicts of interest to rest and focus entirely on a common interest.

Thinking of the software as a necessary tool to reach a common interest such as marketing, growth, sales and others is the first step to being product-minded. These communication accidents will be avoided if the software engineers understand the problem and are aware that they are really talking about the same problem with all the stakeholders.

To understand the problem, it is necessary to understand the problem space. Because people always look for the answers within the problem. It is also possible to prove how accurate a solution you will reach when you try to solve the same problem with your stakeholders in the common language, with the theory of the wisdom of crowds¹³.

¹³

https://en.wikipedia.org/wiki/The_Wisdom_of_Crowds



Francis Galton

The British scientist Francis Galton, who lived in the 19th century, was advocating a cranky idea that is still popular with many of us today. He believed only expert opinions, arguing that their non-expert opinions were worthless. However, these views of his were completely changed as a result of a strange incident he experienced during a wandering. He witnessed a "guessing the weight of the ox" contest held in a small village. Those who wanted to participate in this competition had to buy tickets first. Then they would write their estimates of the weight of the ox exhibited in the village square on a piece of paper and participate in the draw. The person who made the closest guess to the ox's true weight won the ox as a prize.

This competition was attended by what Galton would call experts, butchers, and ox breeders, as well as many enthusiasts, villagers, and visitors who did not fall under the definition of "expert". Those who did not fit Galton's definition of "expert" outnumbered the experts.

After the competition, Galton collected his papers with all the predictions and reviewed the results. The results completely changed Galton's philosophy of life based on expertise and expert vision.

The actual weight of the ox exhibited in the competition was 543 kilograms. Surprisingly, surprisingly, the average of the estimates of the 800 participants in the competition was 542 kilograms. This proved that the opinions of the community, even non-experts, were of great accuracy, with a margin of error of about 1 kilogram. He repeated this

experiment over a long period of time with different groups. But his conclusions were almost the same. The grain of truth in the ideas presented by irrelevant communities was enough to surprise and persuade Galton.

In fact, there are examples of this experiment even today. We all say, "Who wants to be a millionaire?" We know the TV competition.

the accuracy rate of the expert used by the competitor about the joker when answering the question asked is 66%; When questioned by audiences who would not be identified as experts in the studio, the accuracy rate was 90%. That's what's called the wisdom of the crowds.

The "wisdom of crowds" refers to the result of a very specific process, where independent judgments are statistically combined (i.e., using the mean or the median) to achieve a final judgement with the greatest accuracy.

In other words, Wisdom of crowds is the idea that large groups of people are collectively smarter than individual experts when it comes to problem-solving, decision-making, innovating, and predicting. The idea is that the viewpoint of an individual can inherently be biased, whereas taking the average knowledge of a crowd can result in eliminating the bias or noise to produce a clearer and more coherent result.

As you can see, collective intelligence can actually produce more accurate solutions than the solutions of experts. Therefore, you will find that our solutions become sharper when we work collectively and use a common language while trying to understand the problem.

Common Characteristics of a Well-Thought Product



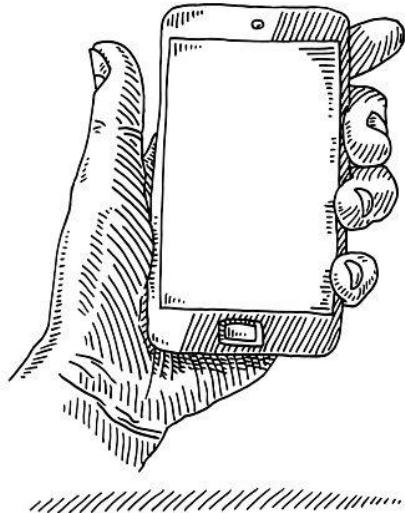
There is no doubt that there are essential characteristics that distinguish one well thought-out product from another.¹⁴ Software engineering is the process of analysing customer and business requirements and then designing, developing, building, and testing software applications to meet those requirements. The process focuses on the design, development, and improvement of software products using scientific standards, techniques, and methods. These result in effective and robust programming items or products. In the early days, software development was relatively simple, so software development was straightforward, but as technology improved, software became more complex and projects became more elaborate. A development team now had to be present, which could prepare detailed plans and designs, test them, devise intuitive user interfaces, and integrate everything into a system. An entirely new discipline emerged out of this new approach called software engineering.

¹⁴

<https://www.innotivum.com/Chapter%202.pdf>

Many software products are available on the market today that serve various industries. Often software developers focus on creating excellent products that work, but they forget to consider other aspects that can make a product extraordinary. Maybe you're a software developer interested in developing or improving your product but not quite sure what makes a great one. So, what you need is to incorporate fundamental characteristics of software into your software's core functionality in order to have an outstanding product.

Functionality



Functionality¹⁵ refers to whether a product works and helps the users meet their goals and needs. When a product is highly functional, it does what it's expected to do and does it well. Good product has a purpose and is crafted in such a way that it will consistently and reliable execute a particular function. Functionality is the goal of product development. The functionality of something is its usefulness, or how well it does the job it's meant to do. You might question the functionality of your new smartphone if you can't get it to send simple text messages.

¹⁵

<https://www.designsociety.org/download-publication/30775/ON+THE+FUNCTIONS+OF+PRODUCTS>

A good product must work completely in accordance with its purpose of existence. One of its most important assets is that it works as expected by nature. Products are responsible for fulfilling their design purposes¹⁶. They can only give their customers confidence when they work as they promise. Although the trust problem of customers has an equation with many obscures, they expect the products they use or decide to use to fulfil their design purposes.

Functionality refers to whether a product works and helps the users meet their goals and needs. When a product is highly functional, it does what it's expected to do and does it well. Good product has a purpose and is crafted in such a way that it will consistently and reliable execute a particular function.

Many customer experience studies have also proven that the first thing customers focus on when using products is functionality. It has been observed that around 90% of new customers give up using the product when the promised functionality is not provided. It is also known that these negatively affected users especially affect the future customers of the product.

¹⁶

Baker, MJ 2003. The marketing book. Butterworth-Heinemann Cravens, D 2002.

Usability



“Usability is about human behaviour. It recognizes that humans are lazy, get emotional, are not interested in putting a lot of effort into, say, getting a credit card and generally prefer things that are easy to do vs. those that are hard to do.”

David McQuillen

Usability is a measure of how well a specific user in a specific context can use a product/design to achieve a defined goal effectively, efficiently and satisfactorily. Although it is seen as a very simple concept, it is a very powerful tool when used correctly. It can derail even the most surprising product concepts and the most expensive investments if not implemented well.

The usability of the software is characterised by its ease of use. In other words, it should take less effort or time for customers to learn how to use your product. In addition, the usability of your product depends on five basic principles.

Memorability

Your customers, who have been away from your product for a certain period of time, should be able to remember the basic features and functionality of your product when they start using your product again.

Learnability

Your first-time customers need to be able to quickly understand the features of your product. Many studies have proven that first-time customers do not continue to use the product consistently in cases where they have difficulty understanding the product and spend a lot of time and effort to use the product.

Efficiency

Your customers should be able to quickly adapt to the features of your product. If they cannot use your product effectively during the time they spend with your product, they probably will not be a loyal user of your product.

Satisfaction

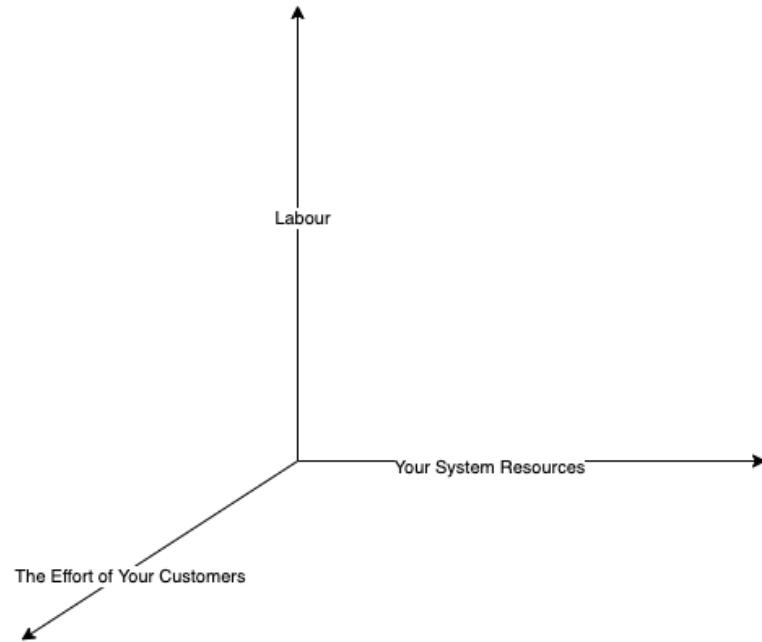
You need to make sure that your customers enjoy using your product and its features. A good product should satisfy its users with the features it offers. The results of surveys, one-on-one interviews, and NPS scores will provide you with this satisfaction in the simplest and most reliable way.

Errors

You know what they say, the err is human. A good product has to track customers' errors and identify the sources of these errors. Error messages should be simple and plain. Necessary improvements should be made to ensure that users do not repeat the same mistakes. It is known that users who make the same mistake do not understand the cause of the errors.

Efficiency

Your product needs to use both human and system resources as effectively and efficiently as possible. One of the biggest secrets of success is that it is efficient. The concept of efficiency cannot be considered from a single point of view. Your customers should use your product with minimum effort.



Efficiency has three dimensions like a cube.

The first dimension or x axis represents the system resources. Your system must be cost efficient when developing or maintaining your product.

The second dimension or y axis represents the human resources. Your team topology and your culture of the teams are the most important factors in the correct use of this resource.

The last dimension or z axis represents the effort and time that users spend while using your product.

A good product should grow exponentially to each other in the end axis. The growth state completely represents an increase in effective use.

Flexibility



“The Only Constant in Life Is Change.” - Heraclitus

It is essential to keep up with rapidly changing markets, technologies, and customer needs. Markets are constantly changing. Unfortunately, products that cannot keep up with these changes do not have a chance to survive in the market. Many products have killed themselves by ignoring the trends of the market or the needs of their customers. On the other hand, it is not only new generation enterprises that cannot keep up with these needs. Incomprehensibly, products that have dominated the market for many years are losing market share either because they do not understand the market's needs correctly or they are not flexible enough to meet those needs.

A good product must adapt to the demands of the market and the customer as quickly and flexibly as possible.

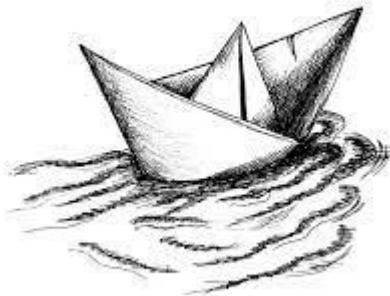
Reliability



A good product doesn't have to be perfect. It just has to be reliable indeed and It should work smoothly. Errors and outages abuse the customer's trust. You would lose all our hard-earned trust of your customers. Riveting, users who use non-reliable products immediately give up using these products only when they discover more reliable alternatives. So, even if your product meets all other values, unfortunately your worst fear is reliability.

Reliability cannot be simplified as "availability". Your entire product -customer services, technical and product teams, internal and external systems, communication channels , etc. – needs to be reliable.

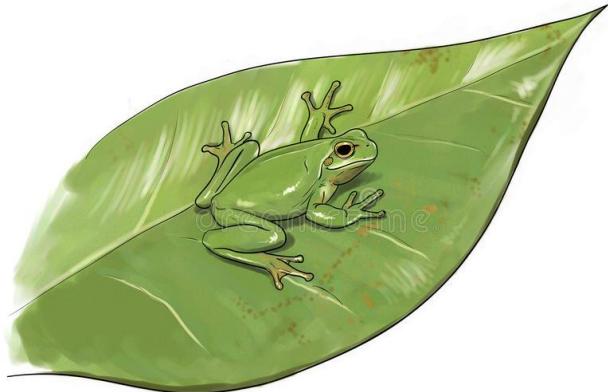
Maintainability



Maintainability refers to how easily you can repair, improve and comprehend your product. In some ways, maintaining is similar to being flexible. Maintainability deals with the modification of errors and minor alterations of the product, while flexibility focuses on major functional extensions. It also involves maintaining the services and functionality of the product.

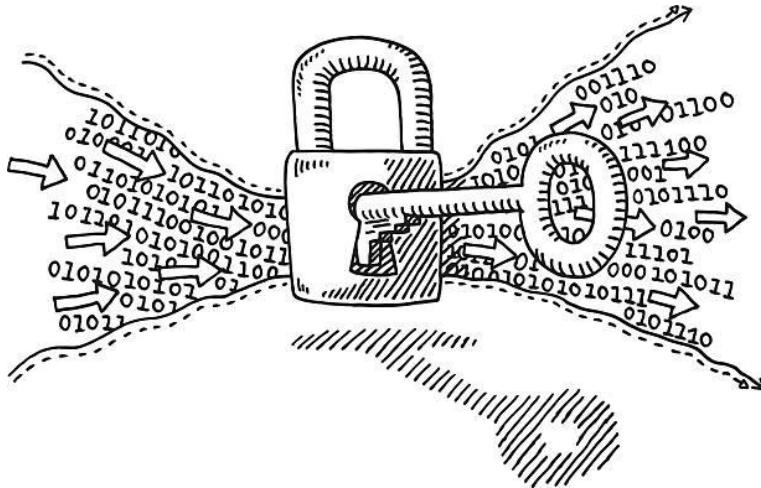
Maintainability is an important step in the product design process. There are many maintainability design criteria and measurement indexes used in product maintainability analysis. The maintenance costs of the product should also be considered when designing a good product. Literally, products that are quickly released to adapt to changes in the market lose a lot of money when they reach the growth stage due to maintenance costs.

Portability



Your product cannot survive only in the local market. Especially in underdeveloped countries, economic, sociological or cultural and many other factors disrupt the income-expenditure balance in local markets. It needs to be used by people from different cultures in different countries in order to grow your business. For this reason, your product must be portable. You may not need to focus on this in the early stage, but a well thought product should not ignore portability.

Integrity



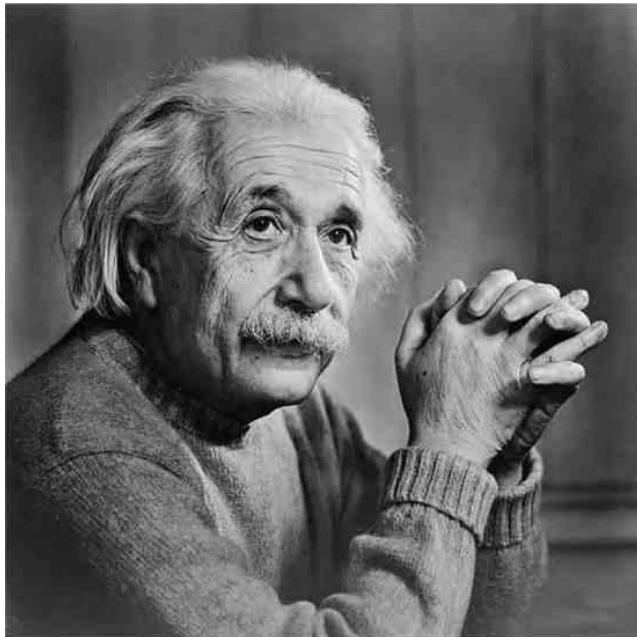
Integrity is key for demonstrating the safety, security, and maintainability of your product. In addition, a product that needs to be compliant with industry regulations and standards requires high product integrity. Achieving product integrity can be difficult. Yet, with the right practices to improve safety, security, and maintainability, the challenge can be easily overcome.

Integrity includes the robust protection of sensitive data of customers. If your product does not provide this integrity, you will probably lose your customers unbelievably fast. For this reason, a good product should clearly indicate to its users what data it will store and which data it will use.

Chapter II

“Understanding the Problem Space and Clarifying the Solution Space”

The importance of understanding the problem



Albert Einstein

“If I were given one hour to save the planet, I would spend 59 minutes defining the problem and one minute resolving it.” - Albert Einstein

Those were wise words. All words of wisdom are actually based on experience. This experience that Albert Einstein shared with us is actually proof that it is important to understand the problem. We should deeply understand and describe the problem as clearly as possible before finding and implementing the solution. If the problem is defined in its simplest form, no matter how complex it is, we actually have solved ~98.3% of the problem. We have to walk out with “the messes”¹⁷

In fact, most of us start dealing with any problem from the solution. To be honest, we act hastily. Unfortunately, we believe wholeheartedly trust all our assumptions. The only reason for this is hidden in our forager¹⁸ genes. For millions of years, our decision-making abilities have been working with our assumptions and instincts. In the last few centuries of humanity, we have learned to leave aside our assumptions and instincts and analyse the problem with scientific methods.

¹⁷<https://www.forbes.com/sites/forbestechcouncil/2018/12/14/how-to-resolve-8-common-conflicts-between-tech-and-sales-teams/?sh=73419e5271a4>

¹⁸<https://hraf.yale.edu/ehc/summaries/hunter-gatherers#hunter-gatherers-foragers>

We first need to analyse and understand the problem in depth in order to reach an absolute conclusion. Understanding the problem allows you to actually address the risks, fixing things that are broken and seizing opportunities.



Russell L. Ackof

“We fail more often because we solve the wrong problem than because we get the wrong solution to the right problem.” – Russell L. Ackof

Russell Ackoff (1979) has one of the most compelling metaphors for complex problems I have encountered so far. He called them “messes”. We all faced the same metaphor in many projects. The main reason for this complexity is that we do not understand the problem- no matter how much we console ourselves with excuses.

The only way to deal with all these messes is to find the causes of these messes by forgetting your assumptions and instincts.

We need to understand the first step in developing a good product after summarizing the characteristics of a good product. In this section, we will try to understand the requirements of creating a product rather than how to send a product to market. The first step in creating a good product is to analyze the problem space in which the product will produce as a result. There may be many different stakeholder, internal and external customers or other members of the organization within the problem space. The only thing that matters is to define the functional and nonfunctional requirements in the problem space as precisely as possible. Many different methods are used in problem

space analysis, but first of all, we need to know what methods we can use to really understand the problem.

There are basic tricks to understanding all problems, regardless of their complexity.

Rephrase the Problem

"If you keep rephrasing the question, it gradually becomes the answer" – Robert Brault

Trying to rephrase a problem is one of the simplest ways to understand it clearly. When trying to rephrase a problem, even the smallest parts need to be rethought according to psychologists. Also When we are handed the definition of a problem, it is often tempting to use that definition while pondering the solution. But often, rephrasing that definition will allow us to examine the problem from a new angle — "think out of the box", as some might say. Sometimes, rephrasing the problem makes what was not so obvious damningly apparent.

But rephrasing a problem is not just helpful in looking at product questions. Sometimes, just rephrasing a simple everyday question can make the answer readily apparent.

Suppose that you want to increase your sales. Look at the changing perspectives as the verb is changed in the following:

- In what ways might I increase sales?
- In what ways might I attract sales?
- In what ways might I develop sales?
- In what ways might I extend sales?
- In what ways might I repeat sales?
- In what ways might I keep sales?

Expose and Challenge Assumptions

'It's not that I'm so smart, it's just that I stay with problems longer' – Albert Einstein

It will contain assumptions — no matter how simple the problem is. This is an undeniable fact. There can be no problem without assumptions. You must prioritise all assumptions and challenge them one by one to find the right solution. Every problem — no matter how apparently simple it may be — comes with a long list of assumptions attached. Many of these assumptions may be inaccurate and could make your problem statement inadequate or even misguided. The first step to get rid of bad assumptions is to make them explicit. Write a list and expose as many assumptions as you can — especially those that may seem the most obvious and ‘untouchable’. That, in itself, brings more clarity to the problem at hand. But go further and test each assumption for validity: think in ways that they might not be valid and their consequences. What you will find may surprise you: that many of those bad assumptions are self-imposed — with just a bit of scrutiny you are able to safely drop them.

For example, suppose you’re about to enter the restaurant business. One of your assumptions might be ‘restaurants have a menu’. While such an assumption may seem true at first, try challenging it and maybe you’ll find some very interesting business models (such as one restaurant in which customers bring dish ideas for the chef to cook, for example)

Chunking

Making the problem smaller is one way to examine a problem statement. So is making it bigger. When solving messy problems (also known as ill-defined or wicked problems), the problem solver should use every trick in the book to craft the most useful problem statement. Messy problems, however, tend to be vague, their goals fuzzy, and the way to a solution, murky.

So what do you do? You can make the problem smaller by breaking it into pieces (see Which problem are you solving?), or make the problem bigger by making it more abstract.

Making a problem more abstract means identifying the core concept(s) of the problem, and moving up to a higher or more general category.

Chunk Up

Chunking up is the term most often used in creative problem solving for moving to a higher level. Chunking means breaking something into meaningful chunks. Chunking up is when you group related chunks into a higher category; chunking down refers to moving downward to specifics.

In psychology, chunking is a technique to memorise long lists of items. To memorise a telephone number, for example, you break the number into meaningful chunks such as country and area code. Objects can be grouped in whatever way makes sense to you—colour, shape, use, and so on. Such groups, because they are meaningful to you, will be easier to remember than an unordered, large group. The items in a group will also be easier to remember; because they are similar in whatever way you chose, one item can remind you of another.

Chunking can also refer to the way text is broken into meaningful chunks; for example, chapters, sections, paragraphs, and lists.

Each problem is a small piece of a greater problem. In the same way that you can explore a problem laterally — such as by playing with words or challenging assumptions — you can also explore it at different “altitudes”.

If you feel you’re overwhelmed with details or looking at a problem too narrowly, look at it from a more general perspective. In order to make your problem more general, ask questions such as: “What’s this a part of?”, “What’s this an example of?” or “What’s the intention behind this?”.

For a detailed explanation of how this principle works, check the article Boost Your Brainstorm Effectiveness with the Why Habit.

Another approach that helps a lot in getting a more general view of a problem is replacing words in the problem statement with hypernyms. Hypernyms are words that have a broader meaning than the given word. (For example, a hypernym of ‘car’ is ‘vehicle’). A great, free tool for finding hypernyms for a given word is WordNet (just search for a word and click on the ‘S:’ label before the word definitions).

- “What’s this a part of?”
- “What’s this an example of?”
- “What’s the intention behind this?”

Chunk Down

If each problem is part of a greater problem, it also means that each problem is composed of many smaller problems. It turns out that decomposing a problem in many smaller problems — each of them more specific than the original — can also provide greater insights about it.

‘Chunking the problem down’ (making it more specific) is especially useful if you find the problem overwhelming or daunting.

Some of the typical questions you can ask to make a problem more specific are: “What are parts of this?” or “What are examples of this?”.

Just as in ‘chunking up’, word substitution can also come to great use here. The class of words that are useful here are hyponyms: words that are stricter in meaning than the given one. (E.g. two hyponyms of ‘car’ are ‘minivan’ and ‘limousine’). WordNet can also help you finding hyponyms.

We can ask ourself these questions :

- “What are parts of this?”

- “What are examples of this?”

Use Multiple Perspectives

Before rushing to solve a problem, always make sure you look at it from different perspectives. Looking at it with different eyes is a great way to have instant insight on new, overlooked directions.

For example, if you own a business and are trying to ‘increase sales’, try to view this problem from the point of view of, say, a customer. For example, from the customer’s viewpoint, this may be a matter of adding features to your product that one would be willing to pay more for.

Rewrite your problem statement many times, each time using one of these different perspectives. How would your competition see this problem? Your employees? Your mom?

Also, imagine how people in various roles would frame the problem. How would a politician see it? A college professor? A nun? Try to find the differences and similarities on how the different roles would deal with your problem.

Use Effective Language Constructs

There isn’t a one-size-fits-all formula for properly crafting the perfect problem statement, but there are some language constructs that always help making it more effective:

- Assume a myriad of solutions. An excellent way to start a problem statement is: “In what ways might I...”. This expression is much superior to “How can I...” as it hints that there’s a multitude of solutions, and not just one — or maybe none. As simple as this sounds, the feeling of expectancy helps your brain find solutions.
- Make it positive. Negative sentences require a lot more cognitive power to process and may slow you down — or even derail your train of thought. Positive statements also help you find the real goal behind the problem and, as such, are much more motivating.

For example: instead of finding ways to ‘quit smoking’, you may find that

'increase your energy', 'live longer' and others are much more worthwhile goals.

- Frame your problem in the form of a question. Our brain loves questions. If the question is powerful and engaging, our brains will do everything within their reach to answer it. We just can't help it: Our brains will start working on the problem immediately and keep working in the background, even when we're not aware of it.
- If you're still stuck, consider using the following formula for phrasing your problem statement:

"In what ways (action) (object) (qualifier) (end result)?"

Example: In what ways might I package (action) my book (object) more attractively (qualifier) so people will buy more of it (end result)?

Make It Engaging

In addition to using effective language constructs, it's important to come up with a problem statement that truly excites you so you're in the best frame of mind for creatively tackling the problem. If the problem looks too dull for you, invest the time adding vigor to it while still keeping it genuine. Make it enticing. Your brain will thank (and reward) you later.

One thing is to 'increase sales' (boring), another one is 'wow your customers'. One thing is 'to create a personal development blog', another completely different is to 'empower readers to live fully'.

Reverse the Problem

The Reversal technique is a creative thinking technique that is based on the thought that to change your perspective, you sometimes need to change the question. By changing the order of the words in your problem definition, you will be forced to look at the situation differently. While a ‘reversed’ challenge sometimes sounds odd and illogical, it often sparks much more creative solutions. How you change the order of the words in your problem statement doesn’t matter much, as long as the key words are reversed. For instance, imagine you are responsible for limiting the traffic congestion in your area. Your challenge is ‘How do we make sure that fewer people take the car to their work?’. Swapping the key words, you could rephrase this challenge as ‘How do we make sure that fewer cars take people to their work?’ Where the first statement will make you think of alternative means of transportation, like trains or bikes, the second statement will probably make you think of solutions like carpooling (fewer cars for the same number of people).

Let’s look at another example. Imagine you are dealing with a shortage of staff in a nursing home. The problem definition ‘How might we make sure there is enough nursing staff to help out the elderly people in our nursing home?’ could become ‘How might we make sure there are enough elderly people to help out the nursing staff in our nursing home?’ The second problem statement is much more interesting and will probably spark more creative solutions. Maybe vital seniors could be allowed to live in the nursing home for free, as long as they help out with the chores and interact with the less fortunate housemates. Interaction with energetic peers might be good for the well-being of the occupants. Or perhaps pensioners could take care of the smaller chores like making coffee, doing laundry or cooking dinner, in exchange for a small supplement to their pensions.

Sometimes, the best way to solve a problem is to look at it in a new way. A logic or decision tree can help you tackle a problem by breaking possible solutions down into parts, and following those parts down new paths.

The process of creating a logic tree is simple – analyze the problem or question, offer solutions or answers, and generate ideas about how to accomplish them. Here’s an example to get started:

Reverse thinking can force a small team that’s been focused on a problem for a long time to think about it completely differently - and come up with a broad range of new ideas that might help to solve it.

Gather Facts

Investigate causes and circumstances of the problem. Probe details about it — such as its origins and causes. Especially if you have a problem that's too vague, investigating facts is usually more productive than trying to solve it right away.

If, for example, the problem stated by your spouse is “You never listen to me”, the solution is not obvious. However, if the statement is “You don’t make enough eye contact when I’m talking to you,” then the solution is obvious and you can skip brainstorming altogether. (You’ll still need to work on the implementation, though!)

Ask yourself questions about the problem. What is not known about it? Can you draw a diagram of the problem? What are the problem boundaries? Be curious. Ask questions and gather facts. It is said that a well-defined problem is halfway to being solved: I would add that a perfectly-defined problem is not a problem anymore.

Clarify the Solution Space

After we fully understand the problem, we need to clarify the solution space. In fact, the solution space is used to determine the borders of the absolute form of the solution and to create simple objectivity. You have to clarify your solution space to make a high-level design of the solution, plan your iterations, and plan resources, or you can able to see the distance between you and your goals more clearly. When clarifying your solution space, you need to be faced with the facts as much as possible.

Many products lie in the graveyard of "it was once a million-dollar idea" because the solution space was built solely on assumptions or instincts, avoiding reality, failing to analyze the market, ignoring customer demands, rejecting math, and not using data.

In fact, many different methods such as mind map, story map, gqm, stakeholder map, stakeholder interview, and empathy map can be used to create solution space. but I prefer the impact map because of its simplicity, clarity and visualization of the entire solution space in detail. In addition, the impact map is a goal-oriented method that focuses not only on certain parts of the problem space but on the whole.

Impact Mapping

Impact mapping is a tool for product development teams to choose which features to prioritise by working outward from an overarching goal and then locating the actions, whether big or small, that will accomplish those goals. Including it as part of the development process ensures that your product roadmap is grounded in your big objectives.

As it begins with the intended goal and extends out from there, all identified features have a direct impact on achieving that goal and a clear rationale for how they will do so. Impact Mapping was introduced to the world by Gojko Adzic in 2012 in his book Impact Mapping.

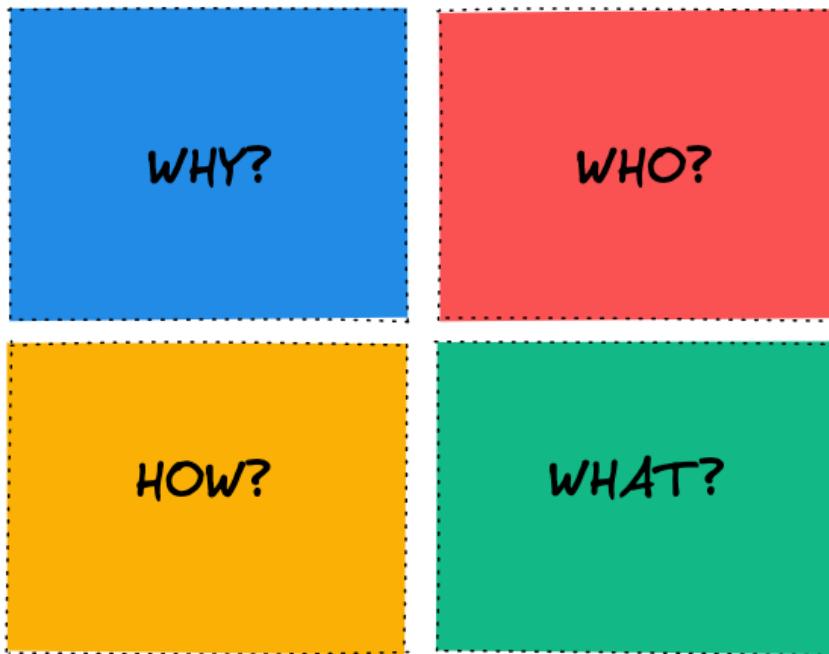
With similar origins and fundamental principles to story mapping and mind mapping, impact mapping is a visual method for feature identification and prioritisation. It quickly illustrates the path from the primary goal to a specific feature by identifying the relevant actors, how they can help achieve the intended goal, and what functionality is required for them to perform those desirable actions.

That way, with the goal at the centre of impact mapping, more time can be spent on identifying and defining the goal more than anything else. The rest of the exercise could lead the project in the wrong direction without the right goal. If you can answer the following SMART goal questions than your goal is in the right direction”

- Is it **S**pecific?
- Is it **M**easurable?
- Is it **A**ction-oriented?
- Is it **R**ealistic?
- Is it **T**imely?

Impact mapping incorporates multiple viewpoints, experiences, and opinions. When you conduct multiple impact mapping exercises with different groups, you can deduce where there are overlap and divergence of impact deliverables based on the biases of the different cohorts.

The impact map uses a framework consisting of 4 different steps to clarify the solution space.

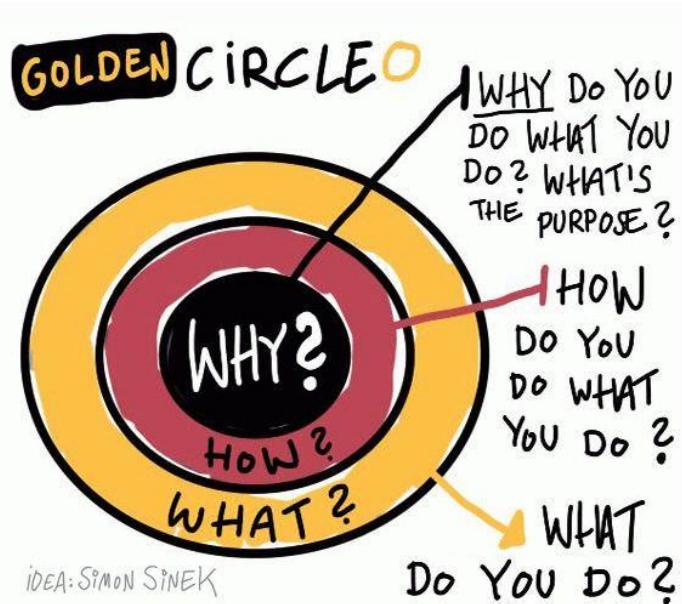


The Goal: Why?

This question totally represents your business goal. This business objective should be formulated as a SMART goal (Specific, Measurable, Action-oriented, Realistic, and Timely) and also clearly explain why achieving this goal will be useful for the organisation as a whole. Thus, the "Why"-question leads us to the following two questions:

1. Do you solve the right problem? and How do you know?
2. How do you measure the success of your product/service?

Simon Sinek proposes The Golden Circle theory in his book "Start with a Why"¹⁹.



The Golden Circle²⁰ consists of three layers;

¹⁹

<https://www.amazon.com/Start-Why-Leaders-Inspire-Everyone/dp/1591846447>

²⁰

1. Why - This is the core belief of the business. It's why the business exists.
2. How - This is how the business fulfils that core belief.
3. What - This is what the company does to fulfil that core belief.

Also, Ingrid Domingues gave the answer to WHY question;

"the impact map describes the business and user values that a new product or service is expected to generate.

- why this solution is a good investment for the business
- how people are going to gain from it
- what the solution should encompass in order to promote user satisfaction and business prosperity."²¹

Lisa Crispin shared her thoughts about impact map because of its effects in solving problems and creating solution space;

"Our stakeholders come to us and say, "Give us a UI that does X, Y, and Z." They often want to give us the implementation, despite the fact that they hired us because we're the ones who know how to develop the software. When customers do this, it's important to ask them the following questions: "Why do you want this feature? What business problem are you solving? What value will it add? How will we measure whether it was successful after we release it to production?" Once we understand the purpose, we can apply our own technical and domain knowledge as well as our creativity to come up with the simplest effective solution.²²

The Actors: Who?

In the "Who" part of impact mapping, we investigate "Who can help us reach our goal?", "Whose behaviour do we want to impact?", "Who can produce the desired effect?", "Who can obstruct it?" Think about people with whom you don't normally collaborate. Be as specific as possible.

<https://www.smartinsights.com/digital-marketing-strategy/online-value-proposition/start-with-why-creating-a-value-proposition-with-the-golden-circle-model/#:~:text=The%20Golden%20Circle%20theory%20explains,if%20they%20start%20with%20why.>

²¹

<https://www.methodsandtools.com/archive/impactdrivenscrum.php>

²²

<https://www.agileconnection.com/article/problem-solving-impact-mapping?page=0>

In other words, the who identifies the actors involved who can influence the final, desired outcome (i.e. achieving the stated goal). You should answer the following questions:

- Whose behaviour do we want to impact?
- Who can produce the desired effect?
- Who can obstruct it?
- Who are the consumers or users of our product?
- Who will be impacted by it?

The Impact : How?

The "How" part describes the impacts of our product or service: "How do we want to change the behaviour?", "How could our actors' behaviour change to help us achieve our goal?", "Which behaviour is most likely to get us to our goal?". Focus on what the actors most likely would do to support us to reach our goal, not everything they 'can' do. Consider behaviour that could potentially — or on purpose — hinder us as well. You should answer the following questions:

- How should our actors' behaviour change?
- How can they help us achieve the goal?
- How can they obstruct or prevent us from succeeding?

The Deliverables: What?

By answering the question "What can we do to support the behaviour change?" we find our "deliverables". With this approach, we put the deliverables in the context of "Who it's for?", and "Why it's important". If we design a technical product or service, these deliverables are the features and functionalities of the intended system themselves and related organisational activities. If we use impact mapping instead as a general problem-solving tool to design business objectives or improvements, the "deliverables" are small experiments we will try to achieve the impact that will help us solve our problem.

In a HBR article²³ Deborah Mills-Scofield answered her own question "What's the difference between outputs and outcomes?" as follows: outputs are extrinsic and outcomes intrinsic. Usually, outputs are artefacts like documents, software programs, or other physical "shippable stuff ", like training, and workshops; outcomes are knowledge transferred and behaviours changed. Outcomes are the difference made by the outputs: better support service, more convenient home banking experience. Outcomes are the benefit our customers receive from our output. Deborah makes this distinction exceedingly clear:

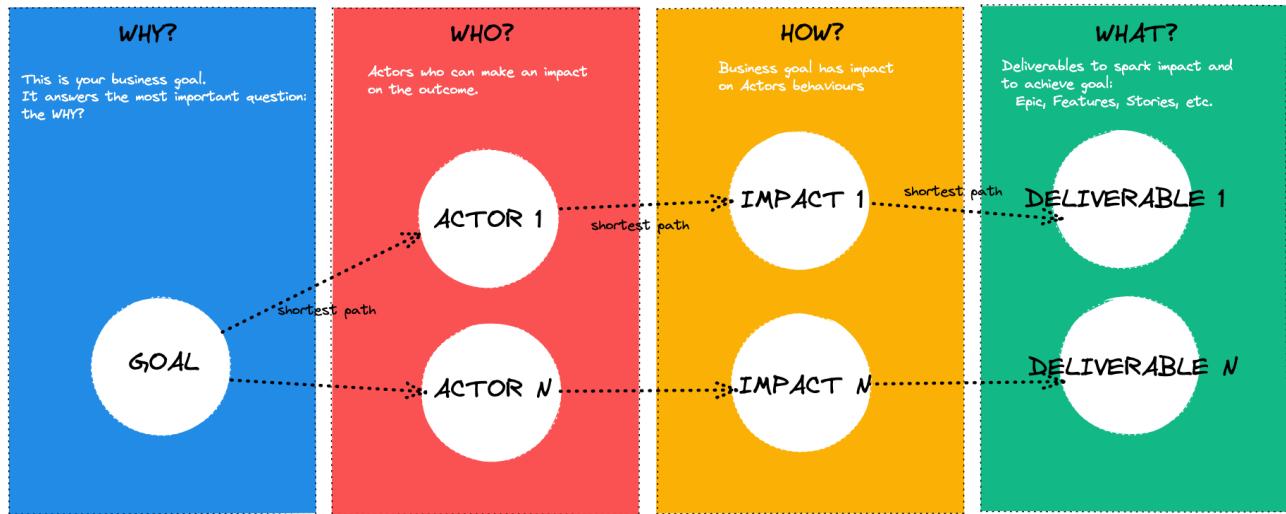
Outputs are important products, services, profits, and revenues: the What.
 Outcomes create meanings, relationships, and differences: the Why.
 Outputs, such as revenue and profit, enable us to fund outcomes; but without outcomes, there is no need for outputs.

Thus, when tracking the "What", we should focus on outputs and outcomes only, not on activities and effort to create them.

²³

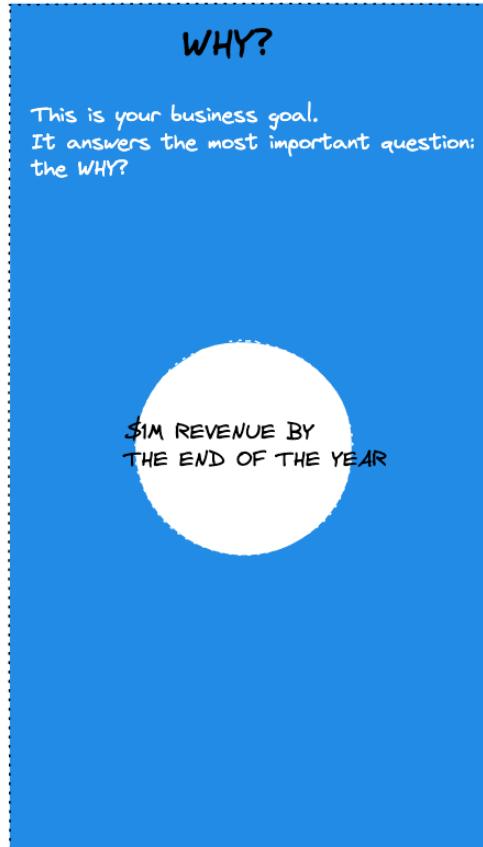
<https://hbr.org/2012/11/its-not-just-semantics-managing-outcomes>

Impact Mapping is, therefore, a great way to create a clearer link between deliverables and their associated impact and goals. This is useful for both reevaluating key business decisions and reprioritizing the product roadmap over time. You can see a plain format of this framework below



Use case: \$1M Revenue by the end of the year

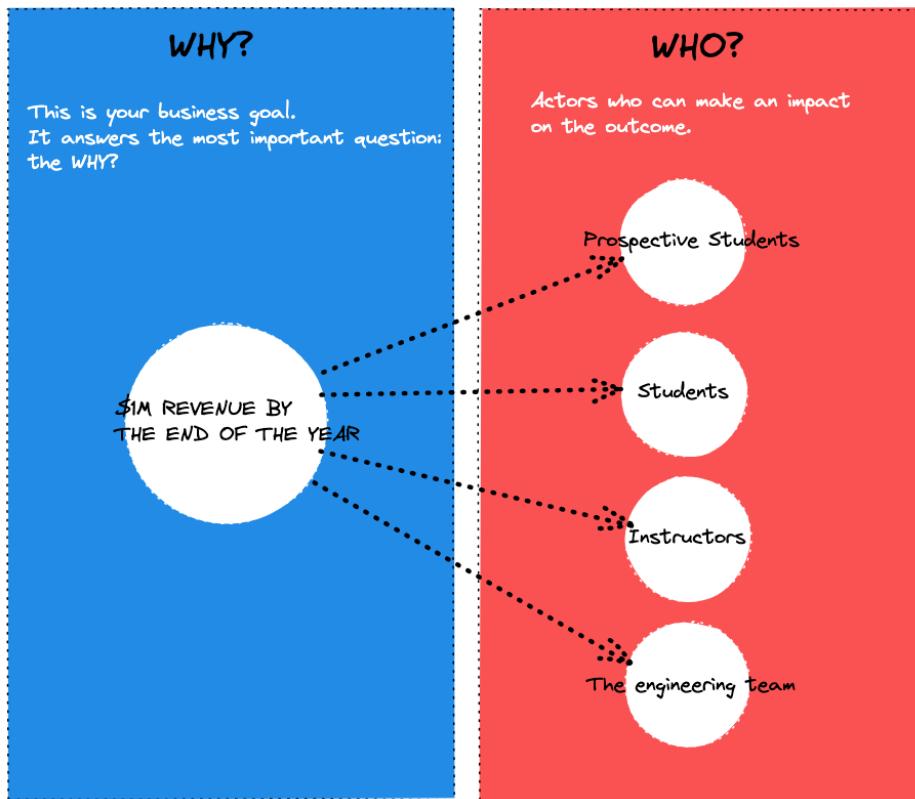
Let's say we are working on a platform like Udemy and our problem space consists entirely of this goal.



Why represents your main goal. This goal should be very clear. You should avoid confusion when defining your goal so that all team members, regardless of their level of knowledge, should understand this. If your goal is not understood, your resources will likely fail to resolve, no matter how experienced and competent your resources are. Therefore, You have to prefer simple and clear sentences instead of using general patterns and sentences that are difficult to understand. You should use more specific words instead of words such as "everyone" that contain a very large mass.

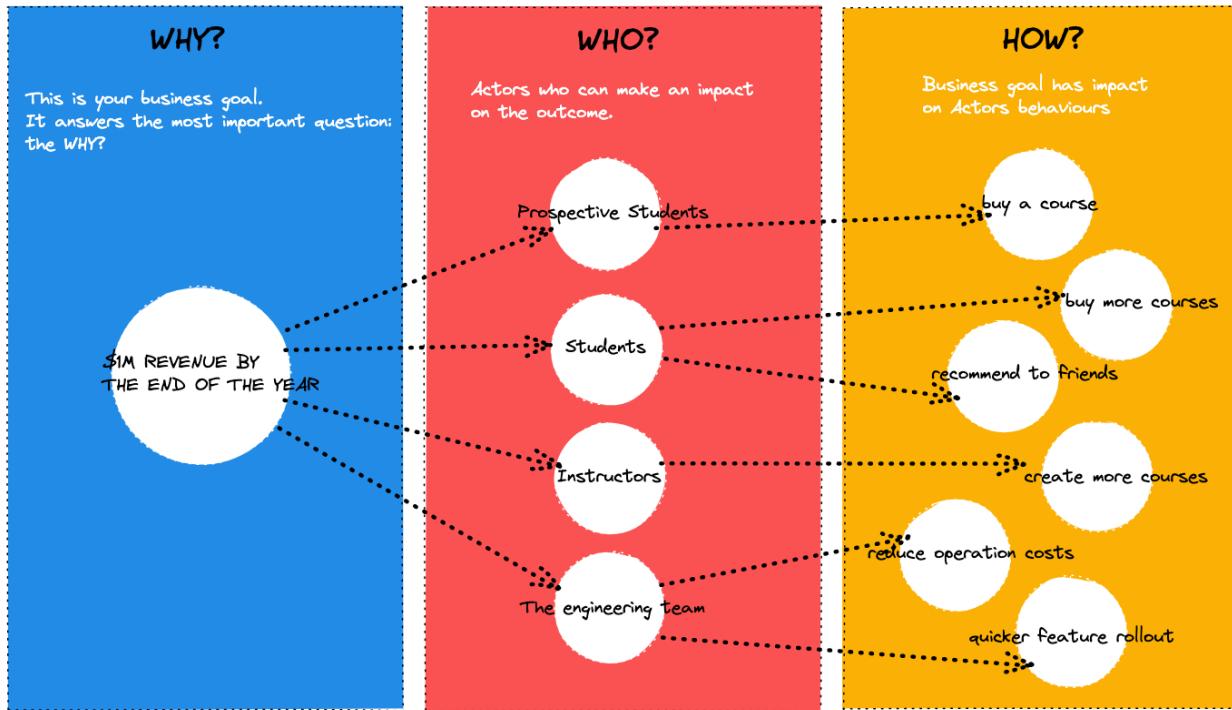
You have defined your WHY. Congratulations, you've taken the first step. What you have to do now is to ask yourself WHO is needed to achieve your goal?

We need to find which actors we can use in order to reach this goal. We can define prospective students, students, instructors, and our engineering team as our actors. Basically, all we want to do is identify possible actors that we can use for this goal.



Your actors need to represent the mass you can reach. A solution space consisting of actors that you do not have yet is unfortunately far from reality. WHO's answer may include actors you don't have yet, but targeting them alone won't make you successful.

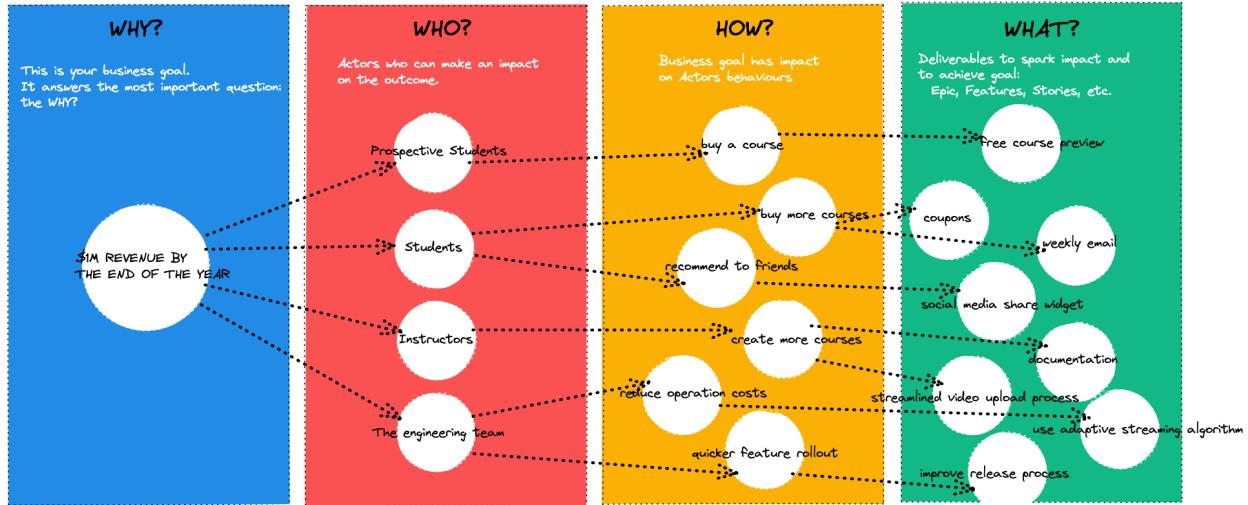
After determining our actors, we need to specify which impacts these actors can contribute to our goal.



Impact needs to be real impacts that your actors can create. You may not have these effects yet, but they must be possible. Otherwise, actors will never be able to create effects that are impossible for them to do.

You can never expect an elephant to fit into a small jar of jam. this idea is defined as impossible to realize according to the positive sciences. You also might think that an elephant could be digested by a boa constrictor. and again, that would just be a figment. It is best to go with the fact that elephants will find wetlands to drink water in warm climates. But with this fact, you can create water wells in the lands they are close to for your elephant actor to quench their thirst, or you can design deliverables to increase the productivity of your drylands.

After all, deliverables can be defined by considering the impacts. You must be completely objective when defining deliverables. You don't have to define only deliverables that you think are of great benefit to you. You can first define all your deliverables and then prioritize them based on their benefits. The only way to go after this stage is to identify and focus on the shortest paths to reach the final goal.



How can you determine which one will provide the greatest benefit in the shortest time, with positive sciences rather than your guesses? In the next section, I will touch on the importance of using data and I will explain what method we will follow when determining the shortest path.

Test Your Solution Space with Data

"Without data, you're just another person with an opinion." - W. Edwards Deming

After creating our problem space, we clarify our solution space, but we don't really decide which solution is effective. We can use our assumptions or instincts to make this decision, but there is a more accurate method; "trust data". Data-driven decision-making is the practice of collecting data, analysing it, and basing decisions on insights derived from the information. This process contrasts sharply with making decisions based on gut feeling, instinct, tradition, or theory.

Data-driven decisions are more objective and can be easily evaluated according to their impact on metrics. Without data, people run a much greater risk of being swayed by biases or acting on false assumptions. But of course, the success of data-driven decision-making depends on the quality of data collected and the methods used to sift through it. Data-driven decision-making is heavily quantitative — historically, its use has been limited due to the need to perpetually collect statistics and crunch numbers.

What is a Data-Driven Approach

The term "data-driven" may seem redundant, as people who make decisions might already rely on data. But in the case of data-driven decision making, companies collect data methodically and analyse it rigorously, so that the information represents reality much more accurately.

Business managers typically rely on both qualitative and quantitative sources of information to make their decisions. The difference between this approach and data-driven decision-making is how the data is collected and processed.

In the absence of a systematic process to gather and analyse data, what we may perceive as reliable data is actually a hodgepodge of anecdotal evidence, personal impressions, and selective information. This "data" consists partly of intuition and experience and is not very scientific.

Of course, intuition does have its place in business. But, as an organisation grows in scale and the dollar amounts at stake become larger, relying on gut instinct becomes very risky. As mentioned above, with data-driven decision making, the processes of

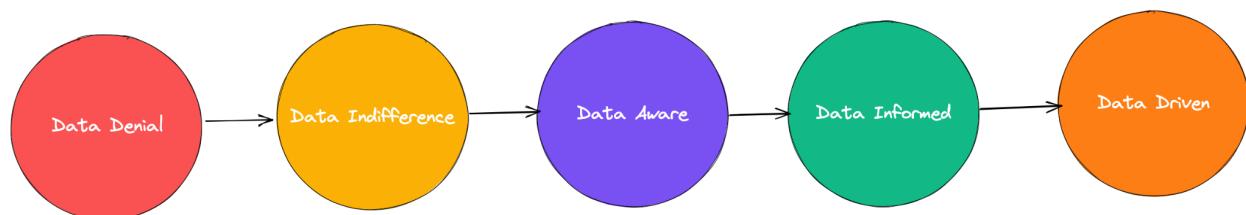
data collection and analysis become methodical and rigorous, so that the information yields a far more accurate reflection of reality.

The Benefits of Data-Driven Decision Making

Data-driven decision-making leads to greater transparency and accountability, and this approach can improve teamwork and staff engagement. Data-driven decision-making policies make clear that whims or fads are not driving the organisation, and morale improves because people see that objective data backs up management decisions.

In organisations that prioritise data-driven decision-making, goals are concrete, and results are measured. Team members often feel a greater sense of control because they can see the goalposts clearly. The tenor of interactions may become more positive because discussions are fact-based, rather than driven by ego and personality.

Data-driven analysis can pay for itself through cost savings and higher revenues. You first need to make sure that you have actually completed certain steps to be Data-Driven decision making.



The organisation starts with an active distrust of data and does not use it. This is the first step of being data-driven and this step is called data denial.

In the second step, The company has no interest in whether data is collected or used. this is called Data Indifference.

The next step is to become Data Aware. The business is collecting data and may use it for monitoring, but the organisation does not base decisions on it.

Once you are data aware, you start accepting and using data. Your teams use data selectively to aid decision making.

Data plays a central role in as many decisions as possible across the organisation after all these steps.

As companies mature in the data-driven stage, they typically progress in their use of analytics from descriptive to diagnostic, predictive, and prescriptive.

Data-driven organisations share some key attributes:

- An emphasis on data collection
- An investment in tools and skills to make sense of data
- A commitment to making data widely accessible
- A willingness to consider data-driven ideas that arise at any level of the organisation
- A dedication to ongoing improvement

If you decide based on data while deciding which path to proceed in your solution space, you will probably choose the shortest and most beneficial path. Consider our udemy example above. In this example there were four different actors and each actor had more than one impact. Your resources have a hard limit. You may not be able to produce many deliverables at the same time. At this point, how will you decide which deliverable to focus on? There is no objective way to know which one has the most accurate gain between two different deliverables, other than using data.

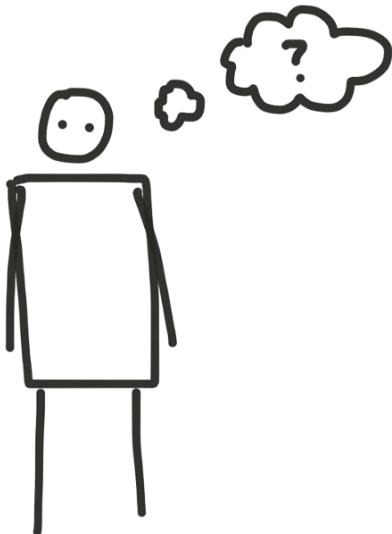
That's why you have to rely on your data to find cost efficient solutions when deciding which deliverables to crunch in your solution space.

Chapter III

*“The key characteristics of product-oriented engineers and
how to empower our competencies”*

Key Characteristics of Product Minded Software Engineers

Start with WHY



Product-minded engineers always start with WHY. They tirelessly ask the WHY question and repeat it until they are satisfied. Assuming that it will take an infinite time for the answer to the WHY question to become clear, you can be one hundred percent sure that they will take the risk.

It may seem strange to you, but there is a really valid reason why they do this. They want to understand the cause of the problem more than the solution and they are interested in it. Because, solutions to problems you don't really understand can be and will be deceptive.

They constantly answer WHY and keep doing this until they realize that there are no different WHYS. You should know that; If the scope of your solution changes every time you answer or test the WHY, it certainly proves that you cannot pinpoint the WHY.

That's why they want to find the root of WHY. As you know, the solution to the vast of problems in the software industry is never linear. The exact solution to any multivariate problem is when the variables are most accurately defined. They aim to clarify the variables to get the most precise solution to the problems you share with them.

They focus on the question of why rather than how. They compare your reasons with theirs and always wonder what could be better. They are autonomous in finding answers they can, by themselves.

Protest by Nature



They never accept the requests of the stakeholders they work with. They may be satisfied with their answer to the WHY, but they never hesitate to test the solution. Most of the time, they do not accept this idea as it is and try to improve the solution. They often challenge existing specifications, suggesting alternative product approaches that might work better. They seek different formats and different dimensions of the solution. They reject other people's solutions doesn't mean they don't like those solutions. they often want to sharpen the solution. While doing this, they enjoy asking questions and listening to different ideas.

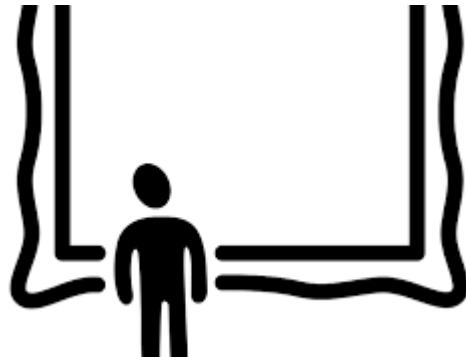
Think Twice



They often have well-thought-out ideas. They take the time to understand how the business works, how the product fits in, and what its goals are. They really care about the feelings of your customers. . They often dive straight to data about business and user metrics, getting their hands on this data however they can. Their ideas are often based on data. They are afraid to talk without data because they don't like to talk in vain. They always think twice before sharing their thoughts with their teammates, product persons, managers or other stakeholder in the organization.

When their ideas are hard to sell. They try to find out why. the purpose of doing this is not arrogance. they really want to understand the gaps of their ideas. If they truly believe in their ideas for the product, they are brave to eliminate all gaps and share their ideas again.

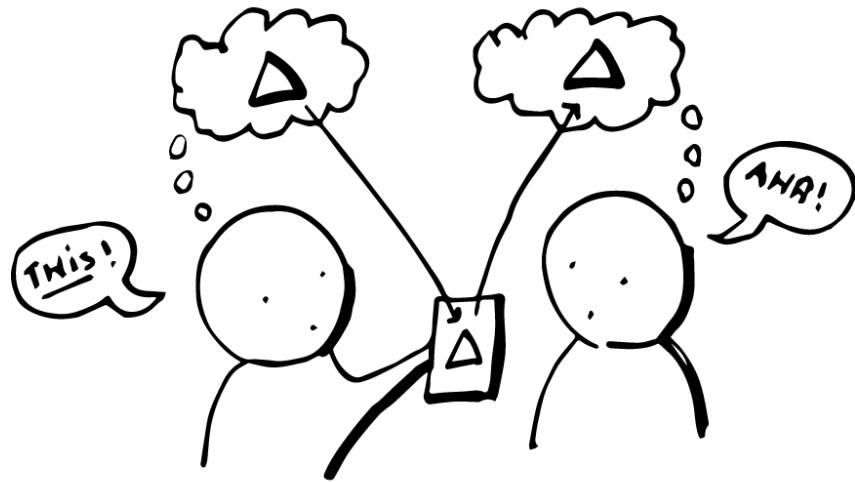
Think outside the box



They are adept at seeing the big picture. They never focus on a single point. They do not hesitate to use their instincts to see the future. but it should not be forgotten that the basis of their instincts is their experience and they make conscious decisions. They consider many open points, such as the pain points that any change will cause to customers, different factors in the market, and the reactions of competitors.

They address all the open points and share them with all stakeholders. They always wonder about open points addressed. Sometimes, even when they have to act fast and adapt to the speed of the market, they remain these open points on the agenda.

Smooth Communicators



They are almost always in strong communication with their non-engineered stakeholders. They are also smooth communicators, making it clear they're interested in learning more about how other disciplines work.

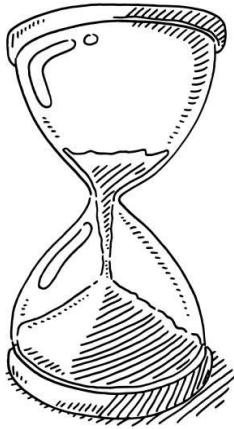
You often see them talking to stakeholders. In most of these conversations, they discuss their ideas about their product and clinch the idea of how it could be better. They love to get ideas from people with different disciplines. They never argue that they have the best idea. on the contrary, they know that the best idea can be obtained as a result of collaborative thinking. For this reason, their communication is strong. and generally everyone accepts them as part of the solution.

Trade-off Juggler



One of the most unique strengths that product-oriented engineers have is juggling both product and engineering tradeoffs and their effects. They have the engineering perspective and the product perspective at the same time. For this reason, they can find different and reliable solutions. They can quickly go back-and-forth between the two sides of the same coin: product features and engineering effort and tradeoffs. Because they do it all in their head, using their engineering and product insights, they get to valuable conclusions remarkably quickly.

Always Patient



They realize that Rome was not built in a day. They constantly strive to improve their ideas, but they know that the right idea can only make a big impact at the right time. They wait patiently for the right time to come. While doing this, they smell the market and follow the competitors.

When they realize that it is not the right time for their ideas, they never give up on these ideas, they just postpone them for the right time. There is always an idea waiting for its time on their agenda.

Edge-Case Master

They can predict many edge cases and address these edge cases to the right points. Sometimes you really need to have enough time and energy to deal with edge cases. However, it is more beneficial in many cases to address them and plan them instead of wasting your time and your energy overcoming these edge cases. In this way, you do not ignore edge cases. More importantly, you use your time really efficiently. Product-minded engineers focus on the "minimum lovable product concept". They put all their energies into tackling the really troubling concerns of the product. On the other hand, they plan their impacts on edge cases well and aim to perfect the product in the short run. They come to you completely with good middle-ground suggestions.

Strike Home with Customers



They always worry about their users not understanding their intentions clearly. "How can we be sure that our users will be able to use the features we have developed in the way we intended?" They constantly ask themselves the question. While the features are still on the anvil, they are worried about the answers.

We know that testing phases, sign-off processes, or even user acceptance tests do not guarantee that users will truly understand your intent. So they always worry about their users not understanding their intentions clearly.

They constantly follow the features they do to see if they are truly understood by their customers. They focus on metrics and want to make sure things are going well. When they realize that something is not right, they focus on how they can better explain it to their customers. They never consider themselves smarter than the customer. They learn from every case that they cannot explain themselves to the customers. They check the interview records with their customers or participate in the customer experience tests themselves to solve this problem.

Product Ownership



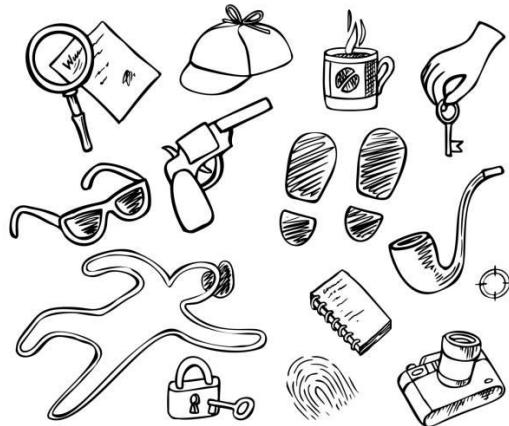
They are not just story crunchers. On the contrary, they want to measure the impact of even the smallest work they do on their customers. After the works they have developed are available, they work with data scientists, product persons or other stakeholders to track the metrics of these works. They consider their work done only after getting results on user behavior and business metrics. They work with data scientists, product persons, and all other stakeholders to do it. If the results are not what they expected, they delve deeper into the causes. They work tirelessly until they detect the root cause of the problem. And once they find the root causes, their only ultimate goal is to solve this problem. They relentlessly ask themselves and all their teammates and even the organization; how is our work really doing?

They want to know all the requirements for the projects they're involved in before trusting their instincts. They are not only concerned with short-term needs. They want to know all the long-term requirements so they can see the bigger picture. They use this data to guide their inner instincts.

Become a product-minded engineer

We now know the key characteristics of a product-minded engineer. Unfortunately, there is no one-size-fits-all method for this. Wait, there is no room for despair! We can still gain some habits by asking ourselves some questions frequently. We'll start with gaining these habits. In time, you will realize that you have all the competencies of a good product-minded engineer.

Discover your company in time



You can start by learning about your business model, how your company made money, or where your company spends its money. I'm definitely not talking about learning confidentials. Your intention should be to grasp the company's business model. You can research your company's most profitable products, departments, or parts. You can take note of which fields they are investing in to grow and keep track of it. You need to learn as much as possible about the reasons for all this, and you need to find out where your team is in this.

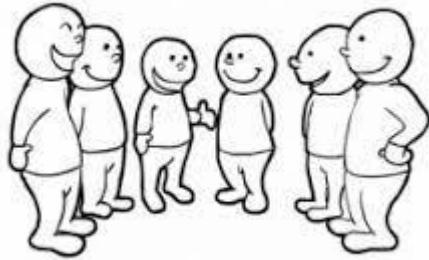
Make a strong relationship with your product persons



You can make a strong relationship with your product persons. You can let them mentor you so you can have some really good insights. In fact, product persons love engineers who are interested in the product because that means they can scale themselves more. You can really deepen your relationship before you start asking questions. I'm not talking about being hand in glove. just talk to them! You can tell the product persons sincerely that you want to be involved in product-related topics.

Chit chats are perfect for that! Even you will be surprised at what you learn in these small talks.

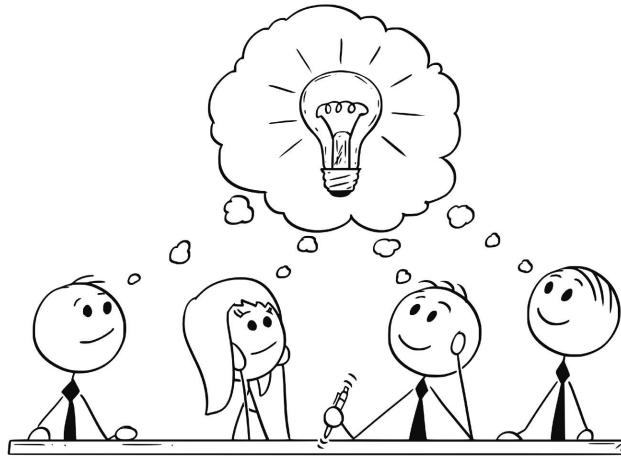
Find who frequently interacts with the customer



One of the best ways to understand a product is to participate in user research, customer support and similar activities. You can clearly understand how your customers see your product in these activities. If you don't have the opportunity to engage in these activities, You must definitely pair with UI designers, UX people, data scientists, and customerwho frequently interact with users.

You can be sure that they know their customers very well and know their wishes and complaints.

Don't be afraid to suggest new ideas



You can organize brainstorming sessions. You can bring well-backed product suggestions to the table in these sessions. Never be afraid to take initiative. If you really know your business, product, and other stakeholders, it's worth the risk! You can start with small suggestions for the project you are working on. Make sure that your first suggestions do not have huge efforts. because big efforts are dreadful! No matter how big the impact, the company may see it as a risk. Also, remember that every suggestion is an investment. Please ask yourself the following question; "If you had money, would you invest in it?" When estimating the effort of your idea, don't just think about the engineering costs. Consider product, growth and marketing costs!

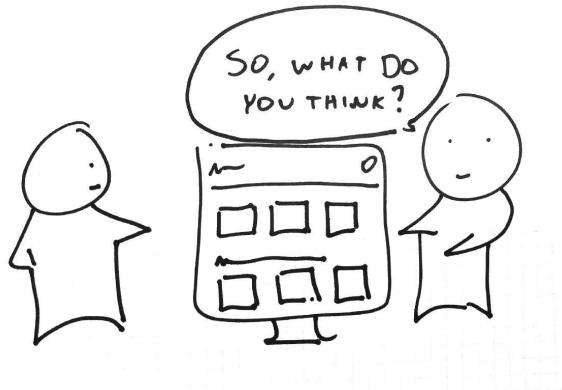
Talk about tradeoffs



You can try to find your tradeoffs. You do not try to find only the engineering tradeoffs of your product, but also the tradeoffs of the product itself. Thus, you can reduce the

engineering effort. Be open to feedback when you share your tradeoffs and close the gaps in your tradeoffs with feedback.

Feedback is important



If you frequently ask for feedback from the product persons that you work with, you can quickly realize your weaknesses and you can improve your weaknesses. You can often have feedback sessions with them. Thus, their valuable suggestions can guide you.

They are the people who can best test your ideas on this matter, as required by their role. you should observe carefully how they test your ideas. Thus, you can test your own ideas with the same methods while your ideas are still seeds.

Code First vs. Product First

Technical requirements should be defined to solve problems with your product completely. It is really difficult to choose the right techniques, programming languages, tools, etc. for your technical requirements. We should not forget that the choice of technology is made entirely according to functional and non-functional requirements. If you use different reasons while making these choices, you will probably break something in your product.

I know we don't like generalizations. But let's make a generalization to understand this subject better; There are actually two types of software engineers. Some of us are in love with writing code. For them, the code is the thing. I will refer to them as code-first engineers. These code-first engineers are obsessed with the architecture of the code, the techniques, and tools they use, the programming languages they choose, the test coverage of their code, the cognitive and cyclomatic complexity, and other kinds of stuff. They satisfy with making better abstractions, using newly released language features, refactoring, or even discussing almost anything related to the code. you can see it in their eyes!

On the other hand, the only reason some of us are really interested in such things is to see them as a means to an end. I will refer to them as product-first engineers. For Product-First Engineers, The code is just one of the scaffolds used to produce the end product. The product is not just about code. they compare its basic quality with how well it solves the problem. They focused on whether the train was actually working as expected. When the train works for its design purpose, everything is nearly perfect for them. On the other hand, they care about the passengers, the rails, and even the route, but they expect the engine to just start, how it works is just a detail for them. The product is the thin for them.

In fact, you can easily distinguish these two types of engineers with a few questions. Take a look at the conversations below and you'll know what I mean.

X: Hi buddy, Y. What's your unit test coverage like?

Y: Frankly, It is pretty close to zero; you know that this is a startup. We are just focusing on the early stage. We have to ship MVP to keep up with the market ASAP.

X: Do you hear what you are saying? You are high! We can't ship a product to the market that doesn't have code coverage of almost 90%!

Y: Why? Will we offer our customers our code coverage? Our product is called "the rarest artifact with the best code coverage?"

Foo: Do you guys use hot and sexy new technology fancy-panths.io

Bar: No, not really yet.

Foo: How so? Reddit, Gitter, Slack channels, and a lot of google groups are talking about it.

Bar: I don't get it. How will this help us to develop our product faster?

Luke: "Is there a lot of technical debt?"

Vader: yes for now. We will have to rewrite it at many points in the future, and we are taking this cost into account. The important thing, for now, is to be able to offer the right thing at right time.

Luke: Noooooo!!!

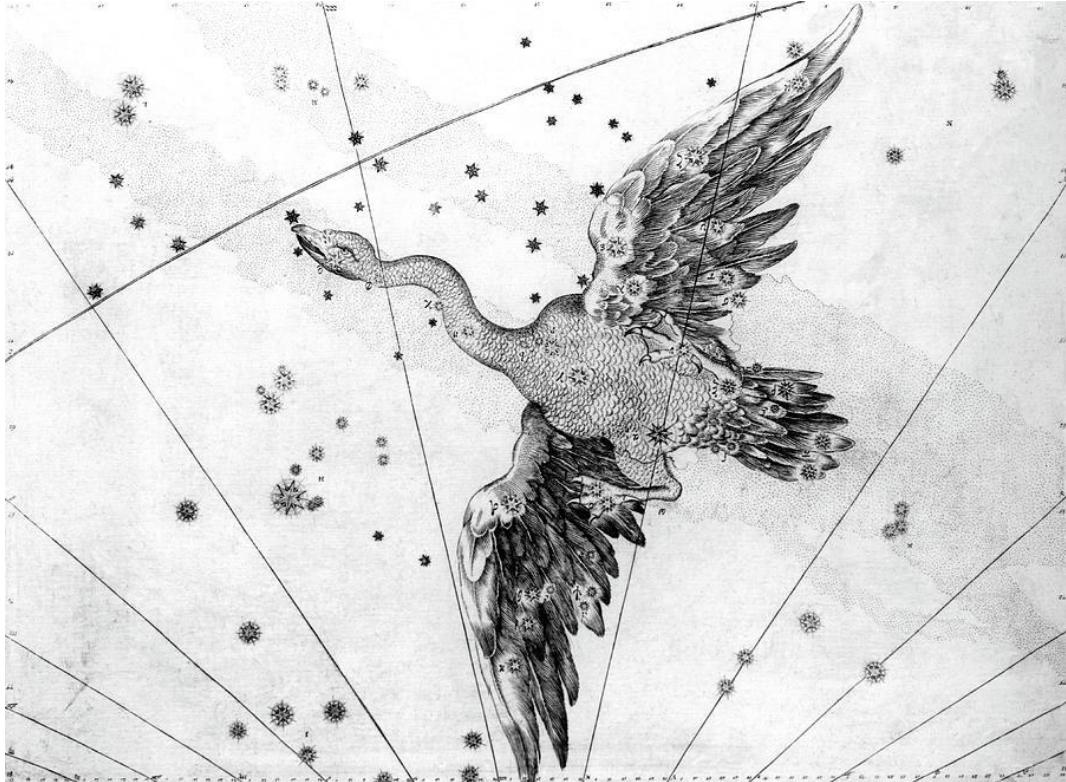
I hope these conversations are familiar to you as well. Most of the time, these conversations end as they started and both parties continue to protect their ideas. Code-first engineers are missing something very fundamental despite their good intentions. Programming is about building products that solve problems for users not about writing fancy code for its own sake. Users don't care about your technology stack, the quality of the code you write, or a lot of fancy terms that are really meaningless to them. They only focus on how the product works. Of course, this doesn't mean writing bad code or developing non-scalable products. These are really, really important, because the good engineering decisions will really get your product to its ultimate goals.

Please do not extend the delivery times of your features that you can really benefit from due to over-engineering. You can be sure that you will be a better product engineer when you balance both sides.

Chapter IV

“A simple framework that you can use to become a good product minded engineer.”

The Cygnus method



Cygnus constellation is the cover photo of this book. It is the eponym of this method. The Cygnus method is a set of rules that includes the basic principles of eight different approaches that I have applied in my engineering career and takes its name from the same constellation.

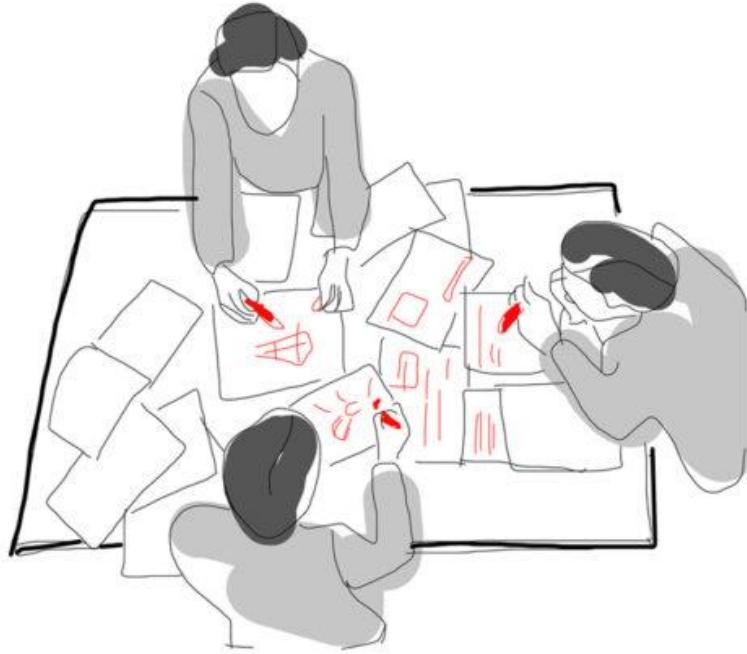
Cygnus²⁴ the swan is a distinctive cross-shaped constellation best seen in the Northern Hemisphere during the summer and fall months around September. The celestial waterfowl swims through the river of the Milky Way and is said to be the disguised Greek god Zeus on his way to a tawdry tryst.

Each perspective has different intentions in itself, but the goal to be achieved in the end is to be a good product minded engineer.

²⁴

<https://www.space.com/cygnus-constellation.html>

Do Collaboration Sessions frequently



As software engineers, most of us are introverts. There may be many psychological, sociological and other reasons for this but of course we can easily overcome this. Organizing collaboration sessions on any topic that interests us - technical or not - is one of the best ways to overcome this. In these sessions, we learn to express ourselves better. We also get the opportunity to listen to different ideas.

When we begin to express ourselves better and challenge our ideas by listening to other ideas, we reach a milestone for understanding the product. Usually, these sessions don't need to reach a conclusion. Therefore, they do not contain stress factors. That's why you should try to express yourself only in these sessions and challenge yourself with different ideas!

You can organize these sessions with your own team. Then expand your audience by including a few of your close colleagues in these sessions. In the next step, address your tribe. and when you really feel ready, target the entire organization.

You must not forget that, You have to get to know your team. You may need to spend as much time as possible with your teammates. The easiest way to do this is used to collaboration sessions. In these sessions, you can talk about your habits or your hobbies apart from technical issues. You can play games with them. Thus, you allow your teammates to get to know you. You cannot be successful without trusting each other! Because success is a phenomenon that you can only achieve if you have a good team.

The Community Contribution is undeniable



In order to contribute to something, you do not need to be one hundred percent sure of it. In fact, it is a naive intention on the basis of the idea of contribution, and the result is often positive. I know that the first thing that comes to mind when you think of community contribution is open-source projects. Of course, this is the most effective and well-known method. but you don't have to start with that first. The main purpose here is to learn to think, act and work collaboratively. Forget your individual ambitions and desires and start thinking about the values of your community. I do not advise you to completely give up on your ambitions and desires. Change your approach not personally but collaboratively. at least for a while!

There is a close connection between your improvement and the improvement of the community you live in. You do not have to contribute only to open-source projects to provide this.

You can make presentations within the team. You can write articles on many different platforms. You can use your youtube channel. You can even share any book or article you've read on a common channel. If you want to talk at any event; You can start by speaking with your own team.

Develop your reading habit



Reading is actually a very good method to reinforce your theoretical knowledge. It has already been proven that reading has many benefits, such as improving your memory and vocabulary, helping you learn new things, and improving focus and concentration.

Please don't view Books as old fashioned. Books don't just deal with the problems of the time they were written. Most of the books are timeless. The problems that were discussed years ago are still valid today. I'm not talking about pure technical books. I'm talking about the masterpieces. Dijkstra's excellent "humble programmer" still stands.

Focus on reading masterpieces that will change your perspective. The most important basis of your instincts and assumptions is your experiences. You can also gain experience from different experiences by reading.

Develop your Engineering Perspective



There is no silver bullet for this. Problem exploration²⁵ includes identifying, framing, and defining design problems and bounding problem spaces. Intentional and unintentional changes in problem understanding naturally occur as engineers explore design problems to create solutions. Through problem exploration, new perspectives on the problem can emerge along with new and diverse ideas for solutions. By considering multiple problem perspectives varying in scope and focus, engineers position themselves to increase their understandings of the “real” problem and engage in more diverse idea generation processes leading to an increasing variety of potential solutions.

How do we change our engineering perspective?

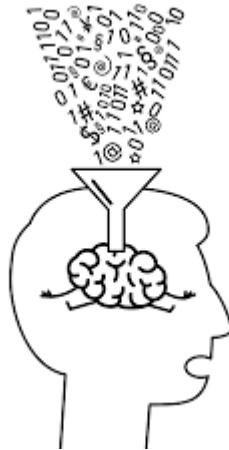
One of the best ways to do this is to follow different engineers. You can learn about their perspectives by listening to podcasts, attending events, or asking them questions directly. Different approaches to the same problems as you or different approaches to problems will be the facilitator for you to improve your perspective. You should know that there is always something to learn from others - no matter what their experience level is.

Spend your time listening to others as much as possible. Test your own ideas in as many different ways and different ways as possible. Even the smallest words you hear when you least expect it can help you solve the biggest problems.

²⁵

<https://onlinelibrary.wiley.com/doi/full/10.1002/jee.20263>

Pragmatic Thinking and Learning



One of the most precious books I've ever read is "Pragmatic Thinking and Learning: Refactor Your Wetware"²⁶ by Andy Hunt.

A practical and extensive collection of ideas, frameworks, tools and tips to supercharge your learning, ostensibly for programmers but relevant to anyone who plans on hacking their learning at school, home and work – by programmer and life-long learner, Andy Hunt of Pragmatic Programmers.

In fact, we need to know ourselves first. we need to determine which side of our brain we use more, what abilities we have, and what is our motivation source in order to think pragmatically. In fact, the key characteristics that the pragmatic programmer should possess are summarized by Andrew Hunt and Dave Thomas²⁷,

- We should take responsibility for our code and decisions.
- Do not leave broken windows unrepaired.
- Think critically
- Know your tools
- Program and refactor deliberately

²⁶

<https://pragprog.com/titles/ahptl/pragmatic-thinking-and-learning/>

²⁷

<https://www.cin.ufpe.br/~cavmj/104The%20Pragmatic%20Programmer.%20From%20Journeyman%20To%20Master%20-%20Andre%20Hunt,%20David%20Thomas%20-%20Addison%20Wesley%20-%20201999.pdf>

- Use Version Control
- Test your code
- Automate all the things

Many of these key characteristics are still valid today. But we first need to know how to think and learn pragmatically in order to have them.

But How?

First of all, let's try to understand what Pragmatics is.

Pragmatic means practical or logical. If someone calls you pragmatic, they mean that you tend to think in terms of the practical or logical rather than the ideal situation.

The term pragmatics is used in contrast to semantics. Semantics has to do with the actual definition of a word or text. Pragmatics refers to how words are used in a practical sense. Words can mean different things, and often the same word can mean something different depending on the context in which it is used. Words can also carry symbolic meaning and in practice, or practical situations, we will apply our understanding of symbols as we read or listen to others.

A pragmatic view means that one doesn't think in ideal or abstract terms. For example, words that attempt to explain abstract concepts-freedom, beauty-have no meaning in and of themselves. Instead, someone who looks at pragmatics would attempt to understand how they are being used in a given, concrete, practical situation. In other words, they look at how we apply these words in practical, everyday language.

Examples of Pragmatics:

Will you crack open the door? I am getting hot.

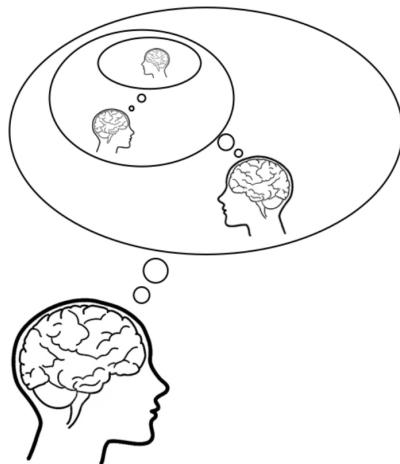
Semantically, the word "crack" would mean to break, but **pragmatically** we know that the speaker means to open the door just a little to let in some air.

If you eat all of that food, it will make you bigger!

Semantically, the word "crack" would mean to break, but **pragmatically** we know that the speaker means to open the door just a little to let in some air.

Now that we have learned the meaning of the concept, let's create a method about how we can think and learn pragmatically.

Always consider the context



In fact, everything you are a part of is a part of a larger system. When you focus on small parts of the system, you cannot see the big picture. this is why if you take things separately, you probably won't be a part of the solution.

Use rules for novices, and intuition for experts.



This is one of the core principles of the Dreyfus skill model. As you become an expert, you must learn to use your intuition and trust your intuition. Your intuition is based on your experience. therefore in many cases, they will help you be part of the solution rather than a hindrance.

Know what you don't know



Wisdom begins with being aware of what you don't actually know. Be humble about your understanding and assume you don't have a complete understanding of the full picture.

You can never know everything by yourself, the valid percentage of the human brain has been measured. in other words, you have a hard limit. Therefore, do not make yourself believe the lie that you know everything and accept the truth.

Learn by watching and imitating



The err is human. Sometimes you have to make mistakes. We all know that learning from mistakes is a very optimistic approach because this lesson can cost you a lot. Therefore, You can learn by watching and imitating these are safer and cheaper methods instead of learning by making mistakes.

Keep practicing in order to remain an expert



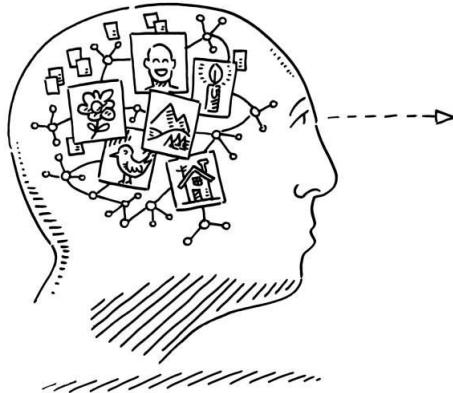
You have to practice constantly to sharpen your skills. The sector we are in is constantly expanding like space. The harder it is to follow an ever-expanding space, the more you have to practice to follow it as closely as you can. First, focus on your existing skills. then identify skills you don't have and chase after them!

Avoid formal methods



If you're not working for the queen, please remember that you don't have to be formal. If you need creativity, intuition or creativity, ignore the formalities and focus on them. bureaucracy may be your most formidable obstacle. but you don't need bureaucracy to really think. Creativity is beyond bureaucracy and formality.

Learn the skill of learning



There is no end to education. It is not that you read a book, pass an examination, and finish with education.

The whole of life, from the moment you are born to the moment you die, is a process of learning.

- Jiddu Krishnamurti, Philosophical and Spiritual Writer and Speaker.

Over the years many theories have been developed to examine the processes involved in learning. Most learning theories concentrate on the significance for the way that learning is delivered. There are many different ways of learning both formally and informally: as part of a group, such as in a classroom setting, one-to-one, such as in a mentoring or coaching arrangement, and self-learning. Furthermore people learn differently at different times in their lives and in different circumstances.

Most learning theories fall into one or more of these approaches;

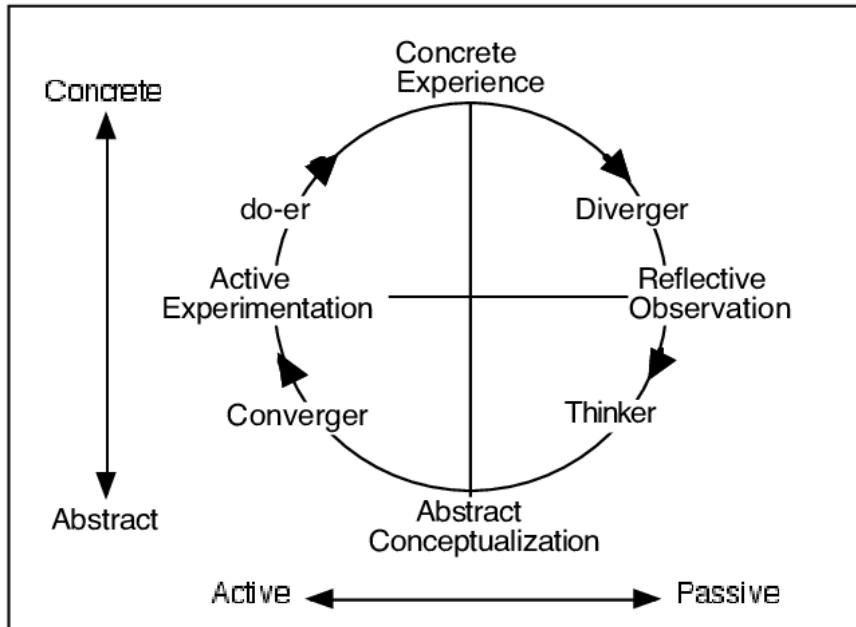
The Behaviourist Approach is concerned with learners responding to some form of stimulus.

The Cognitive Approach is based on knowledge and knowledge retention.

The Humanist Approach is based on explanations of individual experience.

Based on these three approaches, behaviorist, cognitive and humanist, researchers have proposed that we all have different Learning Styles, and put forward two very useful models.

But why are they useful? They are useful because knowing how you like to learn can help you to tailor your experiences so that you learn more quickly and effectively. This is a very difficult subject. I prefer to use David Kolb - The Experiential Approach.²⁸



The experiential model of learning that David Kolb proposes underpins much of the work of modern adult training providers.

Essentially, Kolb believes that learning is a dynamic process, in which we are constantly able to construct our own learning and development by moving through the following cycle.

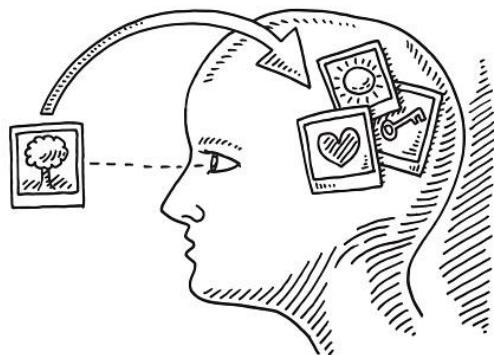
Kolb's four aspects of his learning cycle, in which experience is constantly reviewed and impressions challenged or confirmed, form the basis of experiential learning theory. The sequence is explained in the following way: a person's life experiences form the basis for his/her observation and reflection on what has been encountered encourages learning. This in turn becomes assimilated into what is already known, providing a new conceptual map on which further actions will be based, thus forming a new experience. To complete the cycle, people also need to be able to practice skills learned if the training is to have any true meaning for them.

²⁸

https://www.researchgate.net/publication/267974468_Experiential_Learning_Theory_A_Dynamic_Holistic_Approach_to_Management_Learning_Education_and_Development

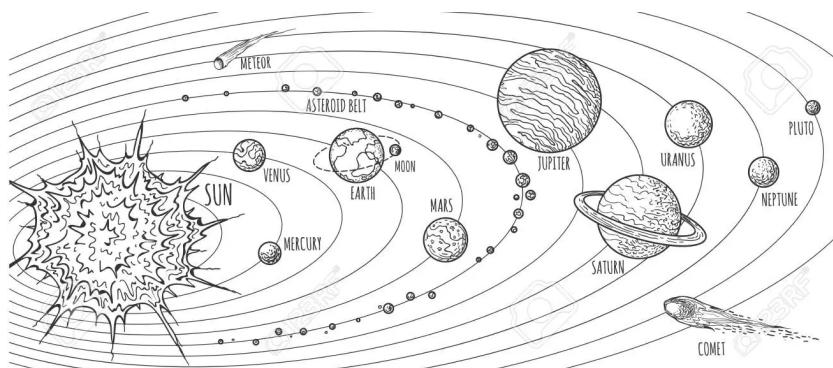
In training terms, therefore, learning is facilitated if the course content and process key into participants' existing experience and are so designed to encourage reflection and the formation of new concepts.

Capture all of your ideas



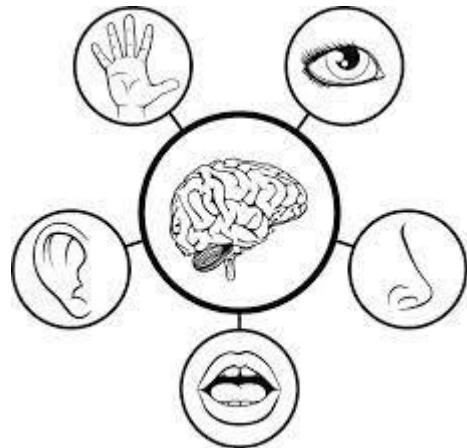
Please consider your ideas. You cannot keep everything in your mind. As I said before, you have a hard limit. For this reason, get into the habit of taking down all the ideas that come to your mind. When you have time, test your ideas one by one and challenge them with others.

Strive for good design



In this image you see our entire solar system. It would be really weird to expect you to design the solar system. Just focus on how perfect it is. As a matter of fact, Humankind always tries to design the best they can, due to their creation, and there are examples of this in nature. You should try to design really good things. It doesn't have to be divine.

Use your all senses



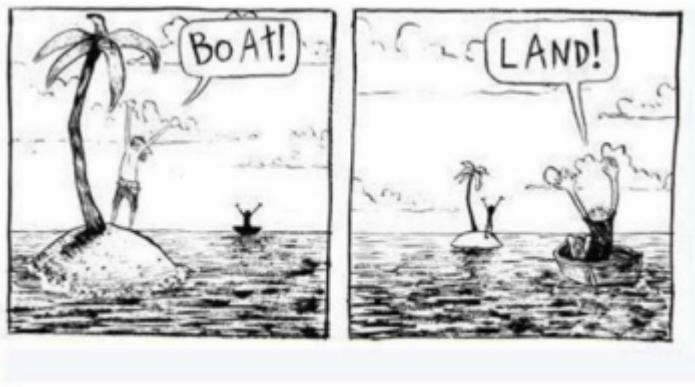
The more senses you engage in a task, the more involved and focused your brain will be. Fiddling, music, walking, etc. Try to use all of your senses as much as possible and engage them with the problem. This will cause your senses to become sensitive over time. As your senses become more sensitive, your solutions will also deepen.

Step away from the keyboard



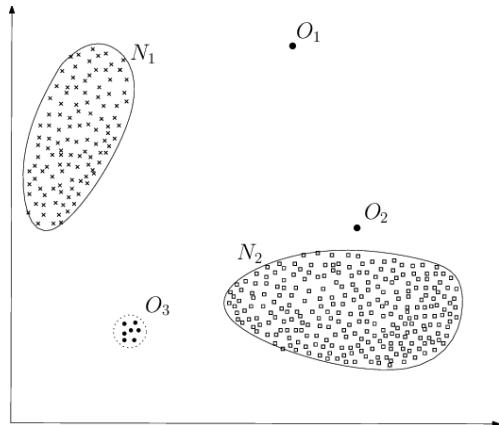
Sometimes you have to get away from the problem in order to solve it. Stay away from the keyboard when you feel burned out. You need the space to let your background processes figure out the problems you're encountering.

Change your viewpoint to solve the problem



Change your viewpoint to solve the problem: look at it in reverse, exaggerate it to the extreme, and change your point of reference. So you can clarify the solution. remember, the solution is not linear.

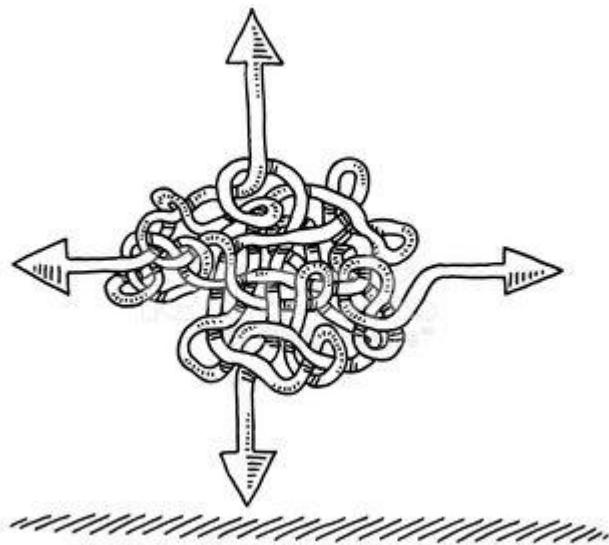
Watch for outliers



Watch out for outliers. Remember that rarely does not mean never. Something that has been proven mathematically can never be ignored. so don't ignore the rare things. The probability of everything that is said to never happen is at least 0.00000001%. (You can enlarge the zero value here as much as you want, but you can never change the fact that it exists.)

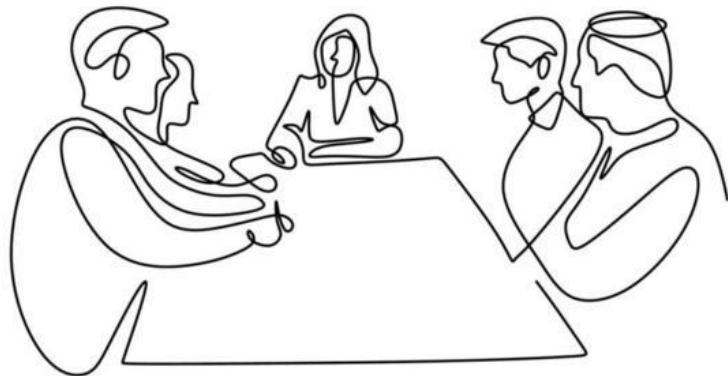
Sometimes a case that is seen as impossible as a percentage can actually create incredible costs for you. - when they happened.

Be comfortable with uncertainty



There will always be uncertainties. Uncertainty is a phenomenon in life itself. You cannot ignore this. Try to clarify the uncertainties as much as possible, but do not spend too much time with it. If you really think you've eliminated many of the uncertainties, don't worry about the rest. In time they will find their own meaning.

Hedge your bets with diversity



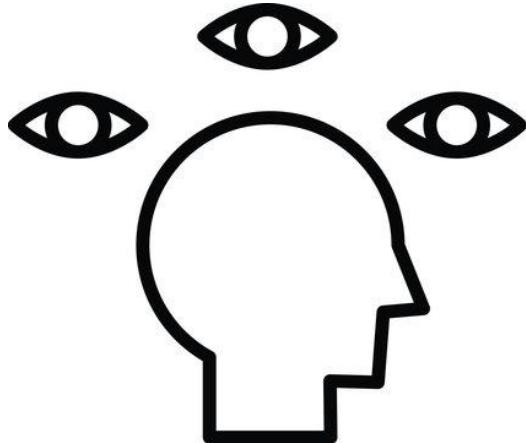
You may not always have accurate ideas. Remember that you are playing a bet when you have to trust your assumptions or your instincts. In order to win the bet, you must rely on the cards in your hand. If you don't trust your cards, don't play this bet.

Act as you've evolved, breathe, and don't hiss



The humankind has spent billions of years completing its evolution. Although we are not yet sufficiently humanized, we must admit that we are different from primates. For this reason, use your abilities that we have developed over billions of years, do not behave like different living things. The most effective way of communication is verbal. If you use the alphabet instead of hissing while doing this, you can be sure that you will get clearer answers.

Trust intuition, but verify



Really trust your intuition, but test your intuition as much as possible before offering it to others as a solution. If you can justify your own intuitions, people can of course do so. The best way to test intuitions is to confront them with reality. If your intuitions are too far from reality, you can be sure they are unreliable.

Create SMART objectives to reach your goals



I explained what SMART is in the section about clarifying the solution space, but the most important thing to know is that you will never be able to reach non-SMART objectives. You do not want to bear the cost of the things you think you have achieved.

Setting SMART Goals means you can clarify your ideas and focus your efforts, which allows you to allocate your time in a way that promises the most return and the highest chance of achieving your goals. Why work towards a goal you cannot achieve or will get nothing out of?

SMART stands for Specific, Measurable, Attainable, Realistic, and Timely.

A **Specific** goal has a much greater chance of being accomplished than a general goal.

To set a specific goal you must answer the six “W” questions:

- Who: Who is involved?
- What: What do I want to accomplish?
- Where: Identify a location.
- When: Establish a time frame.
- Which: Identify requirements and constraints.
- Why: Specific reasons, purpose or benefits of accomplishing the goal.

Establish concrete criteria for **Measuring** progress toward the attainment of each goal you set. When you measure your progress, you stay on track, reach your target dates, and experience the exhilaration of achievement that spurs you on to continued effort required to reach your goal. To determine if your goal is measurable, ask questions such as.....

- How much?
- How many? Make sure its always a set of measurable goals.
- How will I know when it is accomplished?

When you identify your goal and determine how you are going to achieve your goal, you see how **Attainable** it is and how much effort it will require.

In other words, When you identify goals that are most important to you, you begin to figure out ways you can make them come true. You develop the attitudes, abilities, skills, and financial capacity to reach them. You begin seeing previously overlooked opportunities to bring yourself closer to the achievement of your goals.

You can attain most any goal you set when you plan your steps wisely and establish a time frame that allows you to carry out those steps. Goals that may have seemed far away and out of reach eventually move closer and become attainable, not because your goals shrink, but because you grow and expand to match them. When you list your

goals you build your self-image. You see yourself as worthy of these goals, and develop the traits and personality that allow you to possess them.

To be **Realistic**, a goal must represent an objective toward which you are both willing and able to work. A goal can be both high and realistic; you are the only one who can decide just how high your goal should be. But be sure that every goal represents substantial progress.

A high goal is frequently easier to reach than a low one because a low goal exerts low motivational force. Some of the hardest jobs you ever accomplished actually seem easy simply because they were a labor of love.

A goal should be grounded within a **Time frame**. With no time frame tied to it there's no sense of urgency. If you want to lose 10 lbs, when do you want to lose it by? "Someday" won't work. But if you anchor it within a timeframe, "by May 1st", then you've set your unconscious mind into motion to begin working on the goal.

Your goal is probably realistic if you truly believe that it can be accomplished. Additional ways to know if your goal is realistic is to determine if you have accomplished anything similar in the past or ask yourself what conditions would have to exist to accomplish this goal.

T can also stand for Tangible – A goal is tangible when you can experience it with one of the senses, that is, taste, touch, smell, sight or hearing.

Plan your investment in learning deliberately and developing your mind



In fact, every person has limited but very valuable liquidity called time. You should be careful while spending this liquidity on your investments. Because unfortunately this is not a resource that you can regain later. You can never fill his place. So it would be wise for you to make your investments to learn.

Learn from similarities, unlearn from differences



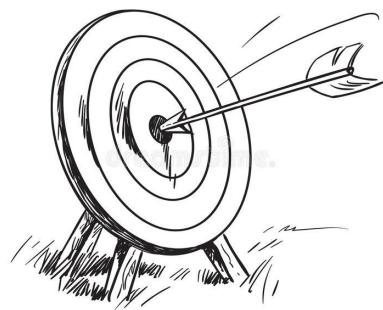
The similarities actually offer a lot of opportunity to learn something. You can compare similar things with each other to determine the differences between them. these differences will lead you to search for the reasons that create the differences. On the other hand, differences are something you should learn from.

See without judging and then act



You can be sure that you are not there to judge anyone wherever you are. Get rid of your biases before making a decision. Using biases when making your decisions will likely cause you to make subjective decisions. Whatever causes biases in your mind, first focus on them and try to handle them.

Give yourself permission to fail; it's the path to success.



Even the most professional archers can't be sure they'll hit their target right out of hitting the bull's eye on their first shot. There is always a margin of error. Archers who accept this margin of error are much more successful than others in hitting the bull's eye on their first shot. Don't be afraid to make mistakes if you really have no other choice.

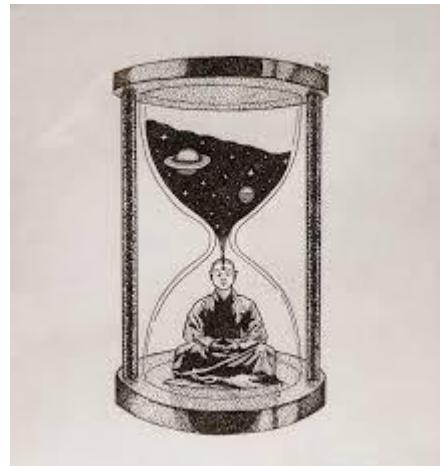
Fail fast and learn fast is a good method. You can fail fast—and learn fast—in many ways.

- The costliest failure that you can make is investing in a project that your customers don't want. The first step in failing fast is to prove that your intended

customers want and need what you're planning to create. Work with your stakeholders and validate your idea.

- During the workshop phase of a project where the team and stakeholders define the application features and minimum viable product (MVP), mandate that participants explore "crazy" ideas to see whether elements of those ideas lead to good ones. In this way, teams can quickly separate the good ideas from the others.
- In a hypothesis-driven development process, the entire team—design, development, and business—identifies the riskiest assumptions that underpin an idea. Then, the team builds experiments to test them. If the idea is based on flawed assumptions, it fails, and the team can pivot and avoid wasting its limited resources. When the team tests the riskiest parts of the project at the beginning, the cost, risk, and design of the project become more predictable.
- When you work with a new project that requires a large infrastructure and hardware investment, such as an Internet of Things (IoT) project, you must validate how the platform and devices affect your design. Implement as little of your infrastructure as possible to validate an assumption and then implement a bit more to validate the next assumption. If a significant part of your IoT platform isn't built before you encounter a failure, you'll have a better idea about where the failure is happening.
- From an application-delivery point of view, development teams that use DevOps can get immediate feedback to find out whether code or a deployment is flawed. In this way, a team can move to higher-quality code and a more stable deployment environment.

Make thinking time



You should spend some of your time thinking. You can just think with nothing to really distract you in a quiet environment. Although thought is an abstract concept, its results are tangible. If you don't have time to think at all, you can be sure that your mind will get blurry after a while. You may not be able to see the gaps in even the simplest of problems, and worse yet, it will affect your decisions badly. The most important reason behind bad decisions is not having enough time to think.

Use a personal wiki to organize your knowledge and learning



As I mentioned before, you have a hard limit and you cannot scale yourself even if you want to. We know that there is a capacity in our minds, so we can create personal wikis and transfer our knowledge to them instead of wasting time trying to store all the

information in our minds. although search, indexing, and other processes may not perform as well as in your brain, it still works.

Grab the wheel, you can't steer on autopilot



You are the captain of this ship. You cannot entrust your ship to someone else or really trust the autopilot. for this reason, you should constantly follow the works within your responsibility. You are only responsible for what you know. You are not responsible for the sea, creatures in the sea, the sky, or natural disasters. If you are only responsible for your ship, crew, cargo, and route, focus on them and pay attention to your responsibilities. The captain, who has sunk more than one ship, can set off again with only a wherry. - sure, if he can find it. Do not take your hand off the steering wheel in order not to lose the trust in you. even if you have autopilot. Because trust is lost quickly, but hard to gain.

Final Words

As software engineers, we are constantly trying to solve problems not only professionally, but also in our daily lives. we deal with problems for a substantial part of our time and try to apply the solutions we find to these problems.

Research shows that the biggest reason behind almost 60% of software projects failing in the last ten years; is stated that the software teams cannot really identify the problems. It is shown that the most important reason for this is that engineers focus entirely on techniques, tools, programming languages , and other things and pass on understanding the problem.

The main reason why engineers have behaved this way, especially in the last decade, is that the industry is changing more rapidly than in previous decades. Engineers spend most of their time specializing in certain verticals to keep up with changes in the industry. This causes them not to fully understand the problems they are dealing with and the projects failing. It is undeniable that many successful startups or ideas have disappeared due to the code first perspective of engineers in general.

In this book, I stated about how we can gain the product-first approach. I have shared methods and experiences that I sincerely believe may be beneficial for you. I hope this book will enable you to include product-first intent in your code-first perspective.