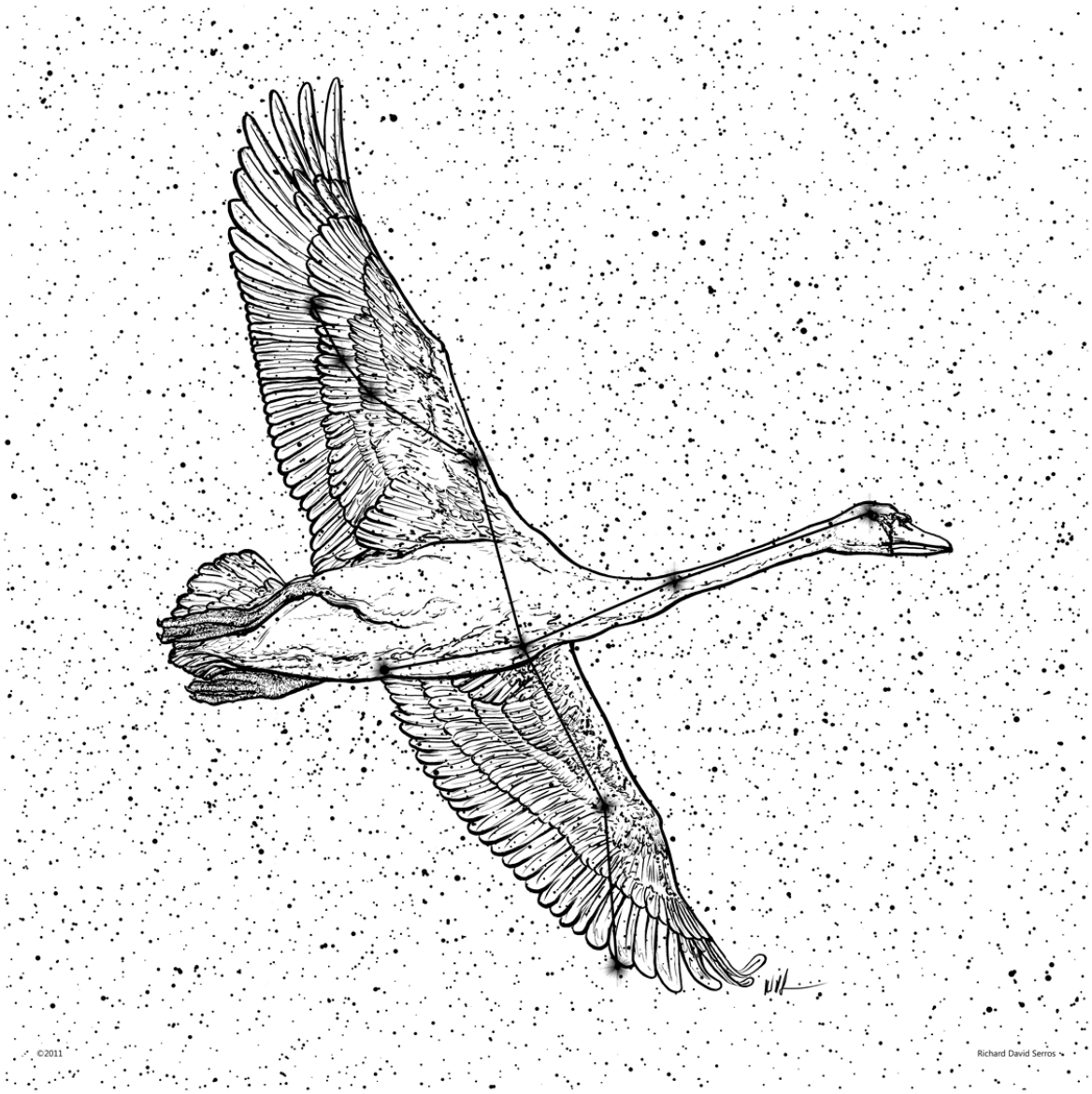


The Product Minded Software Engineer

Cem BAŞARANOĞLU

To all of my family,
and to my dear friend Ilkay AKSAMAZ

“I start where the last man left off.”
Thomas A. Edison

Prologue

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

"How can any software product be delivered faster and tested better?"

All the tools, technologies, and techniques used today were actually created to reach the answer to this question. This question is as complex as it looks simple, and its solution is not linear. Unfortunately, there is no silver bullet in answer to this problem, as Fred Brooks coined in his iconic book "The Mythical Man-Month"[1]. The primary purpose of the conference[2] held by NATO in 1968 to address the issues related to software was to establish guidelines and best practices for software development.

The main factors for the software projects that ended poorly were also discussed at this conference. Let's look at the result when we group all the characteristics 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[3]:

- 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

[1] https://www.goodreads.com/book/show/13629.The_Mythical_Man_Month
[2] https://www.scrummanager.net/files/nato1968e.pdf
[3] https://arxiv.org/pdf/1805.02742.pdf

- The development and adoption of code documentation standards
- 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 such as Agile, XP, and Functional Decomposition, which most of us know very well today, have been raised as a solution to the problem I mentioned at the beginning of the article.

However, we have not entirely eliminated all these problems despite precious technologies, techniques or tools. In fact, as software engineers, we ignored the keystones Dijkstra[4] and Hoare[5] 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 goal.

Numerous gems 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 a 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 methods 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.

This book 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[6] in their

[4] https://dl.acm.org/doi/10.1145/355604.361591

[5] http://176.9.41.242/docs/math/1996-hoare.pdf

[6] http://cognitivepsychology.wikidot.com/cognition:problem-solving#:~:text=In%20this%20theory%2C%20people%20solve,another%20are%20known%20as%20operators.

iconic book 'Human Problem Solving'[7] that the solution is not linear. It actually has more than one dimension. It is necessary to understand the problem space in software engineering, namely the product, to reach a perfect solution. It also deals with the phenomenon of being a product-minded engineer, which has been ingrained in our 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 is about understanding the product from past to present and the characteristics of a good product.

The problem space in which the product's primary concerns and design purposes are addressed and the solution space in which the final form is described most simply are explained in the second chapter.

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

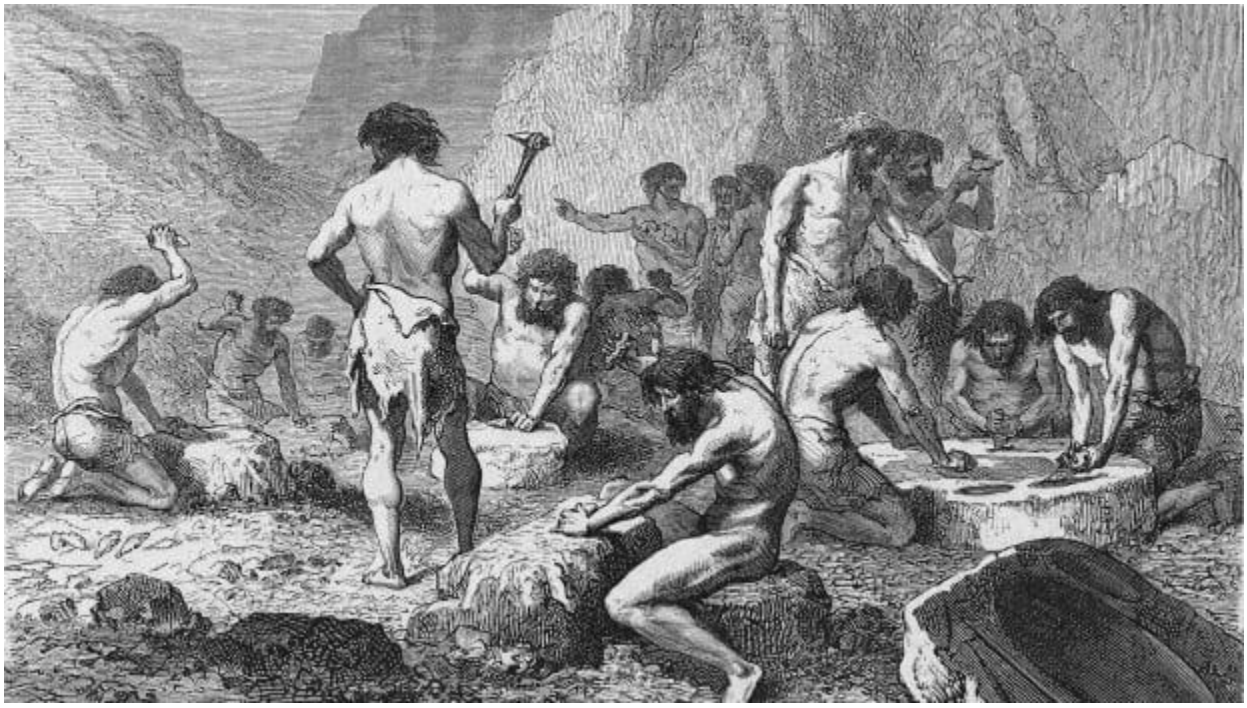
[7] https://www.goodreads.com/en/book/show/4286601-human-problem-solving

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

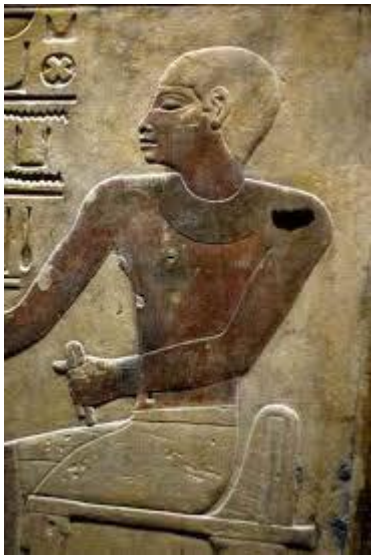


The first age

Humankind[8] (human race; mankind, humanity; homo sapiens.) has tried to find a way out of the difficulties it has faced to make life easier since the earliest ages of the known history of humanity. Our hunter-gatherer[9] ancestors had to constantly solve problems 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.

[8] https://www.goodreads.com/book/show/52879286-humankind

[9] https://hraf.yale.edu/ehc/summaries/hunter-gatherers



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[10].

The first engineer in recorded history, Imhotep, is thought to have built Egyptian pharaoh Djoser's step pyramid[11] in Saqqara, the earliest large-scale cut stone project. Before Imhotep, pharaohs were buried in mastabas — flat rectangular structures constructed over an underground burial chamber. But because Imhotep chose to use stone rather than the traditional mudbrick, he could build much higher systems, 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 attempted 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. He also worked on a step pyramid

[10] https://www.engineering.com/story/who-was-the-first-engineer

[11] https://www.worldhistory.org/article/862/the-step-pyramid-of-djoser-at-saqqara/

for Sekhemkhet[12], 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 graffito 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[13]. I think Imhotep was not the first engineer, although he designed masterpieces. He is considered the first engineer, as the historians who did this research took into account the linguistic 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 [14]

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 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 has a relatively broad usage.

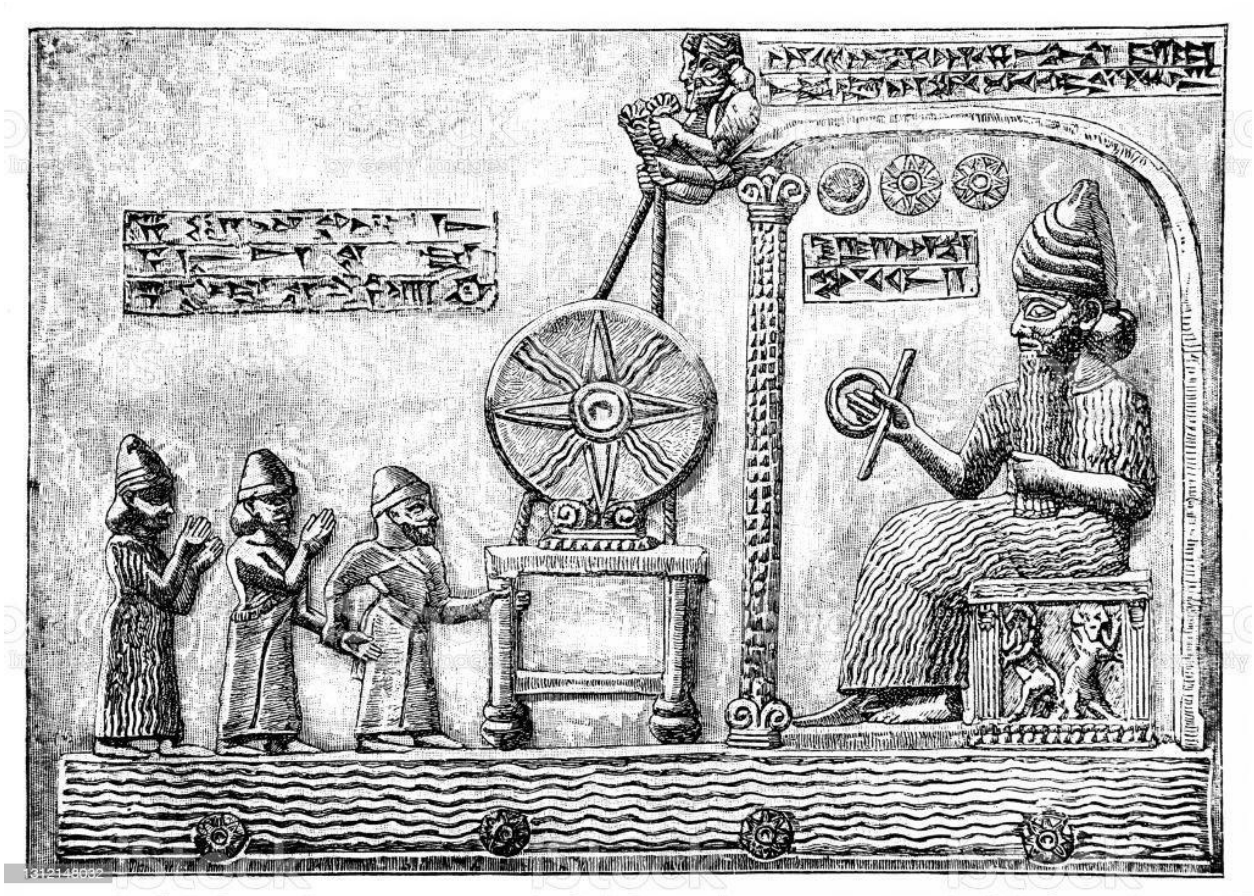
Historians may have identified the first engineer with the lexical meaning briefly defined above. As for me, the first engineers were people who developed solutions to challenging conditions in the dark early ages of humanity. It is an excellent chance for us that the problem-solving abilities that form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

[12] http://www.ancient-egypt.org/history/early-dynastic-period/3rd-dynasty/horus-sekhemhet/sekhemkhet-funerary-complex.html

[13] https://www.britannica.com/topic/Pyramids-of-Giza

[14] https://dictionary.cambridge.org/dictionary/english/engineer

The Product



The Sumerians

The Sumerians[15] were the first Mesopotamian civilization that 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. Sumerians invented[16] or perfected many forms of technology, including the wheel, mathematics, and cuneiform script.

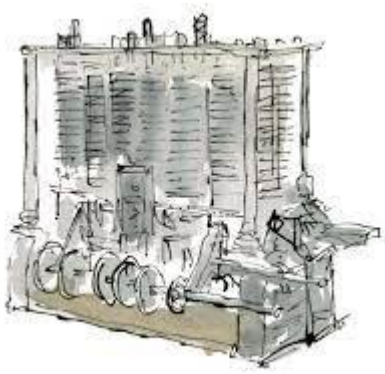
Engineers have developed numerous solutions to solve the problems that societies encountered in daily life and make their lives easier from ancient Sumerians until today.

[15] https://www.history.com/topics/ancient-middle-east/sumer

[16] https://www.worldhistory.org/Mesopotamian_Science/

When the Sumerians invented the wheel[17] to facilitate their transportation and use in agriculture in 4000 BC, they were pretty sure of what problem they were solving.

Also, the Sumerians changed the future by inventing the abacus[18], which is seen as the basis of the computers we use today in 2000 BC. They really wanted to 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 solve a problem when designing "the difference engine"[19] in the 1820s. He developed this solution in the 1830s and built "the analytical engine"[20]. The analytic machine 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[21] might never work on Babbage's analytical engine, and Alan Turing might not have created software theory[22] in 1935 - but none of these great scientists hesitated to perfect the solution of the problem they were dealing with. They found a

[17] https://www.citeco.fr/10000-years-history-economics/the-origins/invention-of-the-wheel#:~:text=The%20wheel%20was%20invented%20in,advances%20in%20two%20main%20areas.

[18] https://www.degruyter.com/document/doi/10.1515/9781501503696-023/pdf

[19] https://www.smithsonianmag.com/history/what-a-difference-the-difference-engine-made-from-charles-babbages-calculator-emerged-todays-computer-109389254/f

[20] https://history-computer.com/charles-babbage-analytical-engine/f

[21] https://www.computerhistory.org/babbage/adalovelace/

[22] https://blogs.scientificamerican.com/guest-blog/how-alan-turing-invented-the-computer-age/#:~:text=In%201936%2C%20whilst%20studying%20for,the%20foundation%20of%20computer%20science.

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 issues.

When we consider that, humanity has developed different products to solve many problems over time. The software is a magnificent tool for creating various products by solving other problems. Pete Mcbreen described software engineers as "Craftsman" in his iconic book "Software Craftsmanship"[23], published in 2001 because of this.

The definition of the product is made in many different ways in many various 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 a set of conditions.” Merriam-Webster[24]

As software engineers, we can name the solutions produced as a result of every problem we solve as a product. The codes we write may be a tiny part of a product, but when we look at it from a broader perspective, the problem we solve refers to a product. We develop products to solve the problems of societies, such as hammers[25] produced in the first ages and asteroid anchors[26] made in our generation.

[23]https://www.goodreads.com/book/show/1035377.Software_Craftsmanship

[24]https://www.merriam-webster.com/dictionary/product

[25]https://hausoftools.com/blogs/news/the-history-and-evolution-of-the-hammer

[26]https://www.hindawi.com/journals/ijae/2019/1257038/

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 software industry was basically trying to solve the problem of building a language that people from these two fields could speak in common. However, there are still conflicts between these two spaces, especially regarding which area has the authority to prioritise, plan, and deliver the tasks. Also, this ultimately causes a mind-boggling dilemma called throw-it-over-the-wall.[27]

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

The most well-known side effects of this are the corruption of the existing culture, the dramatic decrease in the product's benefit for both the users and the company, and the

[27]https://www.interaction-design.org/literature/topics/socio-technical-systems#:~:text=A%20socio%2Dtechnical%20system%20(STS,communities%20of%20people%20and%20technology.

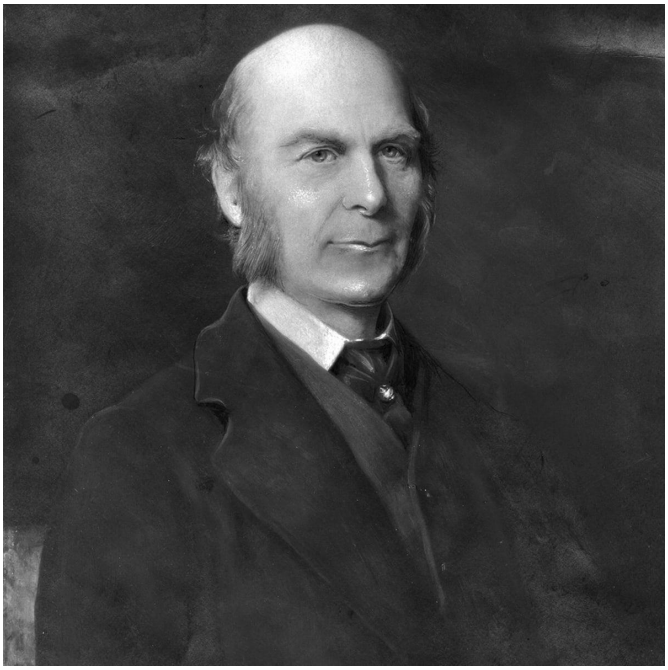
negative impressions on the customer as the newly designed features are not thoughtfully thought out.

In fact, the most essential 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 rest our conflicts of interest and focus entirely on a common interest.

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

It is necessary to understand the problem space first. Because people always look for the answers to 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 crowds.[28]

[28]https://www.goodreads.com/book/show/68143.The_Wisdom_of_Crowds



Francis Galton

The British scientist Francis Galton, who lived in the 19th century, advocated 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 utterly changed due to a strange incident he experienced during a wandering. He witnessed a "guessing the ox's weight" 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 ox's weight 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 proper weight won the ox as a prize.

This competition was attended by what Galton would call experts, butchers, ox breeders, and 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, the average of the estimates of the 800 participants in the match was 542 kilograms. This proved 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 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 particular process, where independent judgments are statistically combined (i.e., using the mean or the median) to achieve a final conclusion with the most remarkable accuracy.

In other words, the 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 eliminate the bias or noise to produce a more precise and more coherent result.

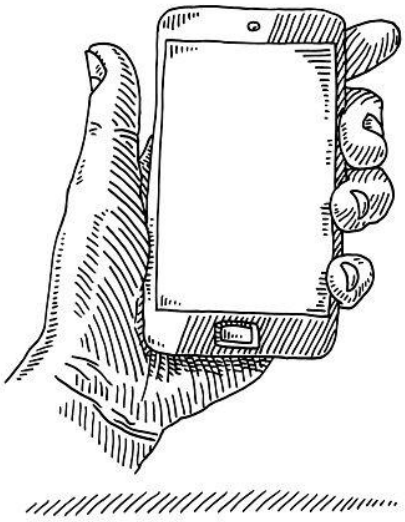
As you can see, collective intelligence can actually produce more accurate solutions than the solutions of experts. Therefore, you will find that our solutions become sharper when we work collectively and use a common language to understand the problem.

Common Characteristics of a Well-Thought Product

There is no doubt that essential characteristics distinguish one well-thought-out product from another. Software engineering is analysing customer and business requirements and then designing, developing, building, and testing software applications to meet those requirements. The process focuses on designing, developing, and improving 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.

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 sure what makes a great one. So, you need to incorporate fundamental characteristics of software into your software's core functionality to have an outstanding product.

Functionality



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 well. A good product has a purpose and is crafted so that it will consistently and reliably 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.

A good product must work entirely following 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 purposes. 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 well. A good product has a purpose and is crafted so that it will consistently and reliably 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, significantly affecting the product's future customers.

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 measures how well a specific user within a particular context can use a product/design to achieve a defined goal effectively, efficiently and satisfactorily. Although it is seen as a straightforward concept, it is a potent tool when used correctly. If not implemented well, it can derail even the most surprising product concepts and the most expensive investments.

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, 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 using it.

Efficiency

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

Satisfaction

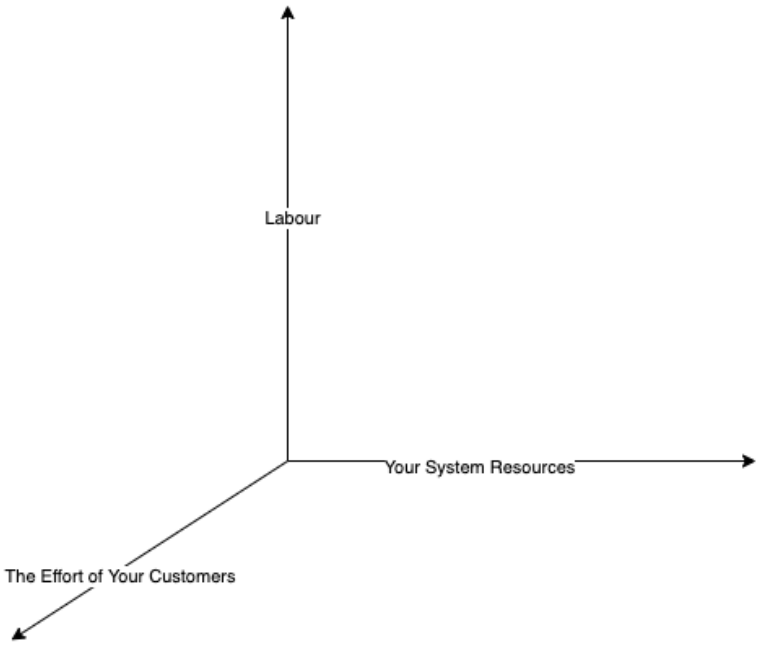
You need to make sure that your customers enjoy using your product and its features. A good product should satisfy its users with the features it offers. The results of surveys, one-on-one interviews, and NPS scores will provide you with this satisfaction most simply and reliably.

Errors

You know what they say; the err is human. A good product has to track customers' errors and identify the sources. Error messages should be simple and plain. Necessary improvements should 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 the y-axis, represents human resources. Your team topology and the culture of the groups are the most essential factors in the correct use of this resource.

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

A good product should grow exponentially to each other in the end axis. The growth state ultimately represents an increase in practical 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, not only new generation enterprises 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 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 in your customers. Rivetingly, users who use non-reliable products immediately give up using them 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. – must be reliable.

Maintainability

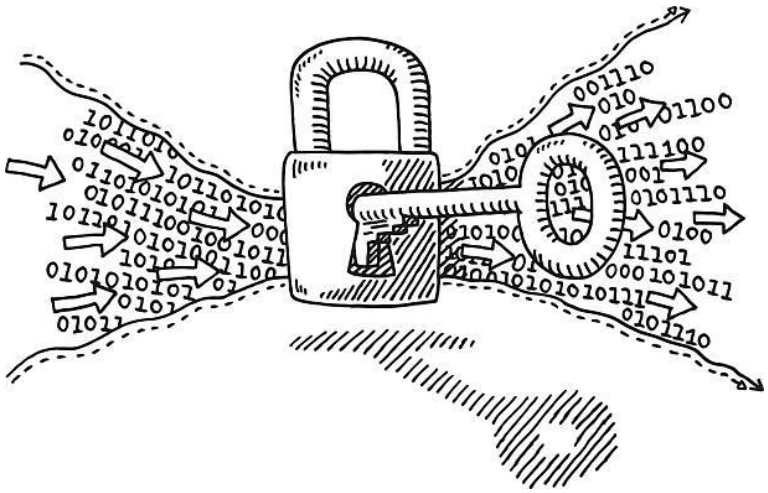
Maintainability refers to how easily you can repair, improve and comprehend your product. In some ways, maintaining is similar to being flexible. Maintainability deals with modifying errors and minor alterations of the product, while flexibility focuses on major functional extensions. It also involves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

Maintainability is an essential step in the product design process. Many maintainability design criteria and measurement indexes are 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 factors disrupt the income-expenditure balance in local markets. It needs to be used by people from different cultures in different countries 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 vital for demonstrating your product's safety, security, and maintainability. 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 exemplary 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 quickly. For this reason, a good product should clearly indicate to its users what data it will store and which information it will use.

Chapter II

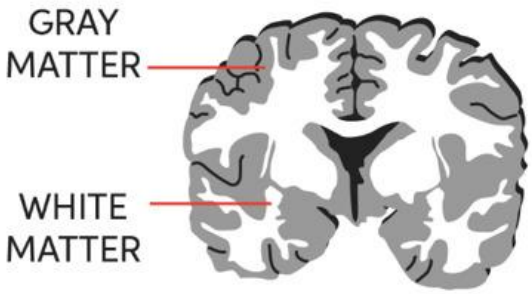
“Understanding the Problem Space and Clarifying the Solution Space”

How do we solve hard problems?

The human brain[29] is truly a masterpiece because of its outstanding design and extraordinary functionality. On the other hand, the concept of thinking is also profound and complex.

As you know, The brain is a complex organ that controls thought, memory, emotion, touch, motor skills, vision, breathing, temperature, hunger and every process that regulates our body. The brain and spinal cord that extends from it make up the central nervous system[30] or CNS.

Weighing about 3 pounds in the average adult, the brain is about 60% fat. The remaining 40% is a combination of water, protein, carbohydrates and salts. The brain itself is not a muscle. It contains blood vessels and nerves, including neurons and glial cells.

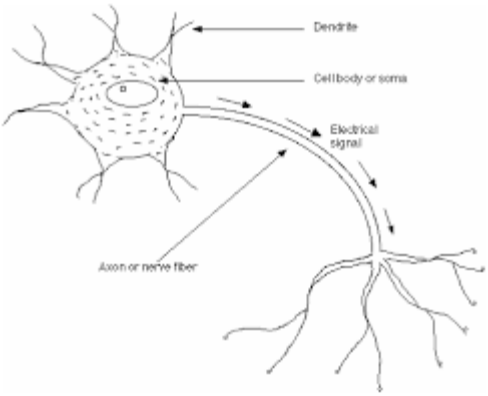


Gray and white matter[31] are two different regions of the central nervous system. In the brain, gray matter refers to the darker outer portion, while white matter describes the lighter inner section underneath. In the spinal cord, this order is reversed: The white matter is on the outside, and the gray matter sits within. Gray matter is primarily composed of neuron somas (the round central cell bodies), and white matter is mainly made of axons (the long stems that connect neurons together) wrapped in myelin (a protective coating). The different composition of neuron parts is why the two appear as different shades on specific scans.

[29]https://www.livescience.com/29365-human-brain.html

[30]https://www.healthdirect.gov.au/central-nervous-system#:~:text=The%20central%20nervous%20system%20(CNS,is%20the%20body's%20processing%20centre.

[31]https://www.technologynetworks.com/neuroscience/articles/gray-matter-vs-white-matter-322973



Each region serves a different role. Gray matter is primarily responsible for processing and interpreting information, while white matter transmits that information to other parts of the nervous system.

How does the brain work?

The brain sends and receives chemical and electrical signals throughout the body. Different signals control different processes, and your brain interprets each. Some make you feel tired, for example, while others feel pain.[32] Some messages are kept within the brain, while others are relayed through the spine and across the body's vast network of nerves to distant extremities. The central nervous system relies on billions of neurons (nerve cells) to do this.

[32]https://www.ncbi.nlm.nih.gov/books/NBK279302/

Human Problem Solving

In fact, Allen Newell and Herbert Simon stated how we solve complex problems, what we think about as we work, and the factors that affect whether we find an answer or stay stuck with the problem forever in their classic book "Human Problem Solving". Their work and books have had a tremendous impact on psychology, artificial intelligence, and economics. Allen Newell and Herbert Simon recommend the following strategies in their book to obtain reliable data for seemingly complex problems;

- Categorize the problem you want to solve or find the equivalent in the existing categories.
- Write a computer program that solves the problem in its simplest form.
- Ensure that all stakeholders related to the problem can verbally express their ideas about the solution and include them in the solution.
- Find the differences or similarities of your program with the verbal solution.

Remember that their solution may not be perfect. The most important advantage of this theory is that it tests the link between objective and subjective answers. We know how computers work, but we don't have a clear understanding of how minds work. When we use the computer program, an objective model, to explain the mysterious phenomenon of thought, we actually eliminate a problem. They also argue that human thought, a controversial thesis, is an information processing system like computers. Frankly, it makes solid and exciting predictions about how we think.

Problem Solving is Searching a Problem Space

Newell and Simon argue that problem-solving is essentially a search through an abstract problem space. We navigate through this space using operators, and those operators transform our current information state into a new one. We evaluate this state, and if it matches our answer, the problem is solved. We can liken this to finding our way in a physical space. Compare problem-solving to finding the exit of a maze:

- The problem space is the physical space in the maze. You have some current location, and you want to be at the exit. Solving the problem means finding your way out.

- Operators are the physical movements you can make. You can go left, right, forward, or backward. After each activity, you're in a different place. You evaluate your new state and decide if you have found the solution or if you should move again.

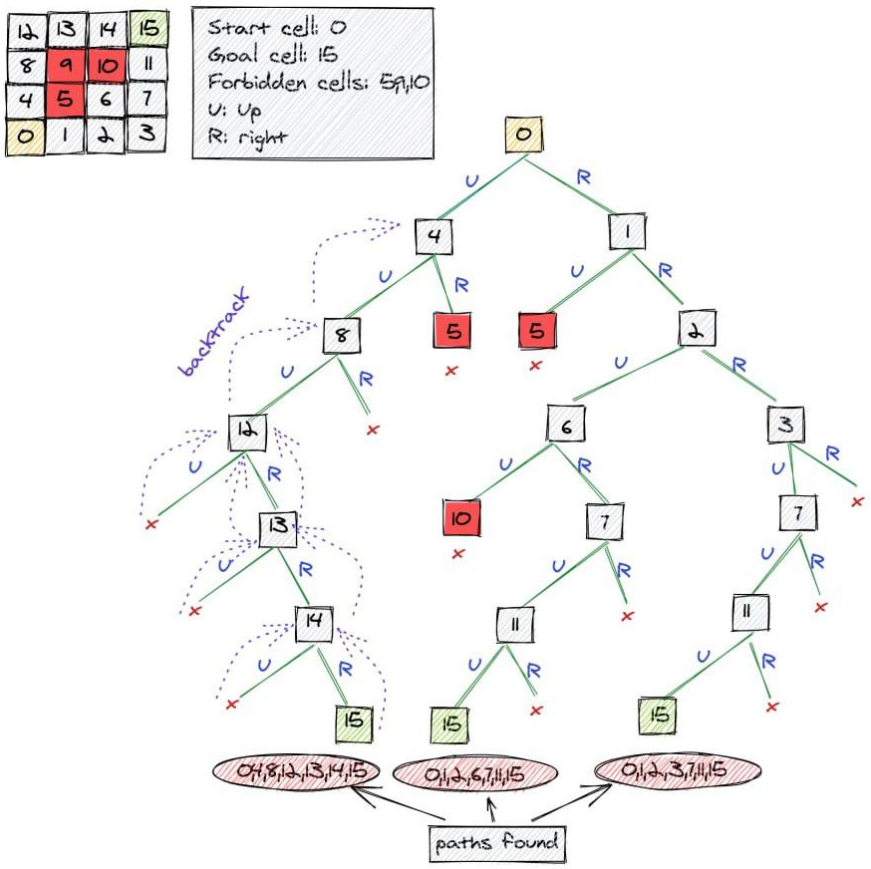
Now consider solving a Rubik's Cube[33]. How does this perspective on problem-solving apply?

- The problem space is all the possible configurations of the cube. Given there are over 43 quintillion possibilities, the space is enormous.
- The operators are your ability to rotate the cube in various directions. Even though the space is vast, the operators available at each moment are pretty limited.

[33]https://www.popularmechanics.com/science/math/a30244043/solve-rubiks-cube/#:~:text=A%20Rubik's%20Cube%20has%20one,cubies%2C%20and%2012%20edge%20cubies.&text=The%20immediate%20math%20to%20be,21212!)%2F12.

- Solving the puzzle involves moving through this abstract problem space, ending in a configuration where the colours are adequately segregated to each side of the cube.

In a Rubik's Cube, the operators on the problem space are physical, but they need not be.

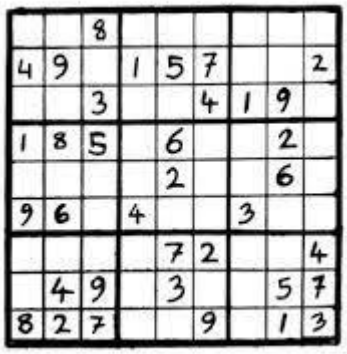


Consider Sudoku[34], where there might be other ways of conceptualizing the problem, resulting in different problem spaces:

- A virtual space might just be the set of all possible numbers assignments to squares. Most of these would fail to fit the constraints of the numbers 1-9 being used uniquely in each subgrid, row, and column. Search in this space might look like trying out a random combination and seeing if it is correct.

[34]http://pi.math.cornell.edu/~mec/Summer2009/Mahmood/Intro.html

A better space would be augmented. Instead of allowing only fixed numbers at each square, you might have information about sets of “possible” numbers. Operators would consist of specifying a particular square and eliminating possibilities from those that remain via other constraints. This is closer to how experts solve Sudoku puzzles, as the bare problem space is unwieldy.



The difficulty of solving a problem isn’t always in searching the problem space. Sometimes, the hard part is choosing the correct space to work in in the first place. Insight-based puzzles, such as the nine-dot puzzle, fit this pattern. In this puzzle, you must cross all nine dots using four straight lines, drawn without lifting your pencil from the paper.

Problem Solving Difficulty

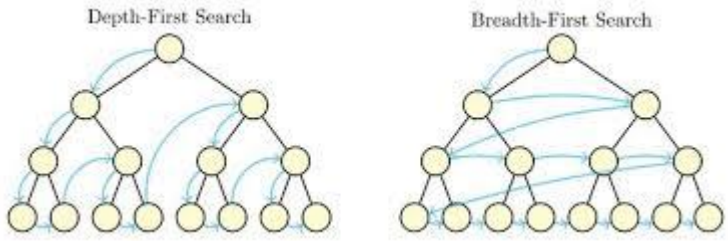
I’ve already mentioned two factors that influence the difficulty of problems: the size of the problem space and how strongly the task itself suggests the best space. Newell and Simon found others in their research.

A simple one is the role memory has on problem-solving. Human cognition[35] depends far more on memory than most of us realize. Because working memory is limited, we lean heavily on past experience to solve new problems.

For instance, subjects universally prefer evaluating chess positions one sequence of moves at a time rather than pursuing multiple lines simultaneously. The difference

[35]https://www.cambridgecognition.com/blog/entry/what-is-cognition#:~:text=Cognition%20is%20defined%20as%20'the,used%20to%20guide%20your%20behavior.

between breadth-first[36] and depth-first[37] search for computers is a technical choice. For humans, depth-first is necessary because we don't have the working memory capacity to hold multiple intermediate positions in our mind's eye.



Conversely, many problems cease to be problems once we have the correct procedure in memory. As we learn new things, we develop memorized answers and algorithms that eliminate the need for problem-solving altogether. Tic-tac-toe is a fun puzzle when you're a kid, but it's boring as an adult because the game always leads to a stalemate.

Seeing something as a problem must occupy a strange middle ground. It must be unfamiliar enough so that the correct answer is not routine, yet not so vast and impenetrable that searching the problem space feels pointless.

What sorts of programs fit best?

Given Newell and Simon frame their theories of human cognition in terms of computer programs, this raises a question: what sorts of programs fit best?

Newell and Simon argue in favour of production systems. A production system is a collection of IF-THEN patterns, independent of one another. The collection of productions fires when the "IF" part of the observed pattern matches the contents of short-term memory. The "THEN" part corresponds to an operator—you do something to move through the problem space.

This remains a popular choice. ACT-R theory[38], which continues to be influential in psychological research, is also based on the production system framework.

[36]https://en.wikipedia.org/wiki/Breadth-first_search

[37]https://en.wikipedia.org/wiki/Depth-first_search

[38]http://act-r.psy.cmu.edu/about/

Productions have a few characteristics that make them plausible for modelling aspects of human thought:

- Their modularity means that parts of what has been learned can transfer to new skills. While transfer research has often been pessimistic, it's clear that humans share acquired skills much better than most computer programs.
- They extend the basic behaviourist notions of stimulus-response. Productions are like habits except, because they can operate on both internal and external states, they are far more powerful. They can incorporate goals, desires and memories.
- They force serial order on human thinking. The brain's underlying architecture is massively parallel—billions of independently firing neurons. But human thought is remarkably serial. Productions, processed in parallel but acted on in sequence, suggest a resolution to the paradox.

How Can We Solve Problems Better?

Human Problem Solving articulates a theory of cognition[39], not practical advice. Yet it has implications for the kinds of problems we face in life:

The Power of Prior Knowledge

Prior knowledge[40] exerts an enormous influence on problem-solving. While raw intelligence—often construed as processing speed or working memory capacity—does play a role, it is often far less important than having essential knowledge.

Consider the ways prior knowledge influences your thinking:

[39]https://www.sciencedirect.com/topics/psychology/cognitive-theory#:~:text=Cognitive%20theory%20posits%20that%20an,and%20deep%20structures%20(schemas).

[40]https://www.researchgate.net/publication/334291100_Prior_Knowledge_Its_Role_in_Learning

```
   ABCD          CRACK
 x    4        +  HACK
 ------        -------
   DCBA          ERROR

    SUN          SEND
 +  FUN       +  MORE
 -------      -------
   SWIM         MONEY
```

- Prior knowledge determines your choice of problem space. This is clear in the cryptarithmetic puzzles[41] used by Newell and Simon. Subjects who already knew a lot about multi-digit addition were able to form a problem space consisting of letter values, odd-even parities and carries. In contrast, less-knowledgeable subjects struggled. Some worked in a more fundamental problem space, trying out random combinations before giving up. Others attempted dozens of different problem spaces, none of which were particularly suited to the task.

- Prior knowledge determines which operators are available to you. A sophisticated library of operators can make the problem much easier to solve. In some cases, it can eliminate the problem entirely as search is no longer required—you simply proceed with an algorithm that gets the answer directly. Much of what we do in life is routine action, not problem-solving.

- Prior knowledge creates memories of specific patterns, reducing the analysis required. In chess, for instance, dynamic ways need a player to simulate how the play will unfold over time. This difficult-to-process task can be replaced by learning static patterns whose outcomes are understood just by looking at them.

[41]https://acrogenesis.com/or-tools/documentation/user_manual/manual/first_steps/cryptarithmetic.html

> Consider a way such as a “forking attack” where a knight attacks two pieces simultaneously. While this pattern can be discovered through searching the possible future moves of the pieces in play, good players recognize it visually on the board. Simple recognition eliminates the need to formally analyze the implications of each piece, sparing precious working memory capacity.

All of this suggests that knowledge is more important than intelligence for particular classes of problems. Of course, the two factors are often correlated. Intelligence speeds learning, which allows you to have more knowledge. However, the intelligence-as-accelerated-knowledge-acquisition picture suggests different implications than the intelligence-as-raw-insight picture we associate with genius. Geniuses are visionary, in large part, because they know more things.

The Only Solvable Problems are Tractable Ones

This is an area where I’ve changed my thinking. Previously, I had written about what I called “tractability bias”.[42] We tend to work on solvable, less critical problems rather than more challenging problems that don’t suggest any solutions.

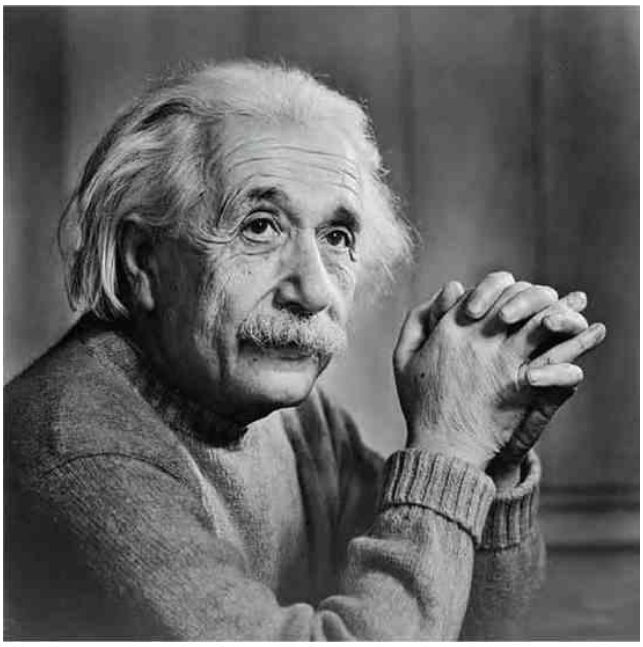
There is some truth to this account: we tend to avoid impossible-seeming problems, even if they’re more worthy of our efforts. Yet HPS points to an apparent difficulty: the importance of a problem has nothing to do with our ability to solve it. Even for well-defined challenges, problem spaces can be impossibly vast. Finding a solution, even something that is “good enough,” can be impractical for many problems.

I suspect our emotional aversion to challenging problems comes from this place. Without reasonable confidence, our problem-solving search will arrive at an answer we don’t invest any effort into. Because problem spaces can often be enormous, this is shrewd, not lazy.

To be successful, we need to work on significant problems. But, we also need to find ways to make those problems tractable. The intersection of these two requirements makes much of life so intriguing—and challenging.

[42]https://www.scotthyoung.com/blog/2021/06/21/beware-tractability-bias/

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. Albert Einstein's experience 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."

In fact, most of us start dealing with any problem with a solution. To be honest, we act hastily. Unfortunately, we believe wholeheartedly trust all our assumptions. The only reason for this is hidden in our forager genes[43]. Our decision-making[44] abilities have been working with our assumptions and instincts for millions of years. We have learned

[43]https://www.frontiersin.org/articles/10.3389/fpsyg.2013.00150/full

[44]https://harappa.education/harappa-diaries/herbert-simons-decision-making-theory/

to leave aside our assumptions and intuitions in the last few centuries of humanity and analyse the problem with scientific methods.

We first need to analyse and understand the problem in-depth to reach an absolute conclusion. Understanding the situation allows you to address the risks, fix things that are broken and seize 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 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. This section 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 result. There may be many different stakeholders, internal and external customers, or other organisation members within the problem space. The only thing that matters is to define the functional and non-functional requirements in the problem space as precisely as possible. Many different methods are used in problem space analysis, but we first need to know what ways 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[45] is one of the simplest ways to understand it clearly. According to psychologists, when trying to rephrase a problem, even the smallest parts need to be rethought. 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, 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?

[45]http://www.infernalramblings.com/articles/Economics/527/

Expose and Challenge Assumptions

‘It’s not that I’m so smart, it’s just that I stay with problems longer’ – Albert Einstein [46]

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. No matter how apparently simple it may be, every problem comes with a long list of assumptions attached. These assumptions may be inaccurate and could make your problem statement inadequate or even misguided. The first step to getting rid of flawed assumptions is to make them explicit. Write a list and expose as many assumptions as possible, 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 might not be valid and their consequences. What you will find may surprise you: many of those flawed assumptions are self-imposed — with just a bit of scrutiny, you can safely drop them.

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

[46]https://www.creativedgetraining.co.uk/2018/02/problem-solving-einstein-way-2/

Chunking

Making the problem more minor is one way to examine a problem statement. So is making it more prominent. When solving messy issues (also known as ill-defined or wicked problems), the problem solver should use every trick to craft the most helpful problem statement. However, Messy issues tend to be vague, their goals fuzzy, and the way to a solution is murky.

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

Making a problem more abstract means identifying the problem's core concept(s) 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. When you group related chunks into a higher category, chunking down refers to moving down to specifics.

In psychology, chunking is a technique to memorise long lists of items. For example, you break the number into meaningful chunks such as country and area code to remember a telephone number. Objects can be grouped in whatever way makes sense to you—colour, shape, use, etc. Because they are meaningful to you, such groups 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 choose, one object can remind you of another.

Chunking can also refer to how text is broken into meaningful chunks, such as chapters, sections, paragraphs, and lists.

Each problem is a small piece of a more significant 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 overwhelmed with details or looking at a problem too narrowly, look at it from a more general perspective. 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?”.

To explain 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 have a broader meaning than the given the word. (For example, a hypernym of ‘car’ is ‘vehicle’). WordNet is a great, free tool for finding hypernyms for a given the word (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 into many smaller problems — each more specific than the original — can also provide greater insights into 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 valid words here are hyponyms: terms that are stricter than the given one. (E.g. two

hyponyms of ‘car’ are ‘minivan’ and ‘limousine’). WordNet can also help you find 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 look at it from different perspectives. Looking at it with different eyes is a great way to have instant insight into 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, adding features to your product may be a matter that one would be willing to pay more for.

Rewrite your problem statement many times 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 in how the different roles would deal with your problem.

Use Effective Language Constructs

There isn't a one-size-fits-all formula for adequately crafting the perfect problem statement, but there are some language constructs that always help make 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 are many solutions, not just one — or maybe none. As simple as this sounds, the feeling of expectancy helps your brain find answers.
- 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 'increasing your energy, 'living longer, and others are 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 unaware.

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 essential to develop a problem statement that genuinely excites you. Hence, you're in the best frame of mind for creatively tackling the problem. If the situation looks too dull for you, invest the time adding vigour 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), and another 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[47] is a creative thinking technique 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 more creative solutions. Changing the order of the words in your problem statement doesn't matter much, as long as the keywords are reversed. For instance, imagine you are responsible for limiting the traffic congestion in your area. Your challenge is 'How do we ensure that fewer people take the car to their work?'. Swapping the keywords, you could rephrase this challenge as 'How do we ensure 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 feel 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 exciting 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, making laundry or cooking dinner in exchange for a small supplement to their pensions.

[47]https://hatrabbits.com/en/reversal/

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

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 focused on a problem for a long time to think about it entirely differently - and come up with a broad range of new ideas that might help solve it.

Gather Facts

Investigate the causes and circumstances of the problem[48]. 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 apparent. However, if the statement is "You don't make enough eye contact when I'm talking to you," 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.

[48]https://www.creducation.net/resources/problem_solving_skills/gather_facts.html

Clarify the Solution Space

After fully understanding the problem, we need to clarify the solution space. In fact, the solution space is used to determine the borders of the complete 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 face 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 big or small actions that will accomplish those goals. Including it as part of the development process ensures that your product roadmap is grounded in your significant objectives.

As it begins with the intended goal and extends out from there, all identified features directly impact 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.

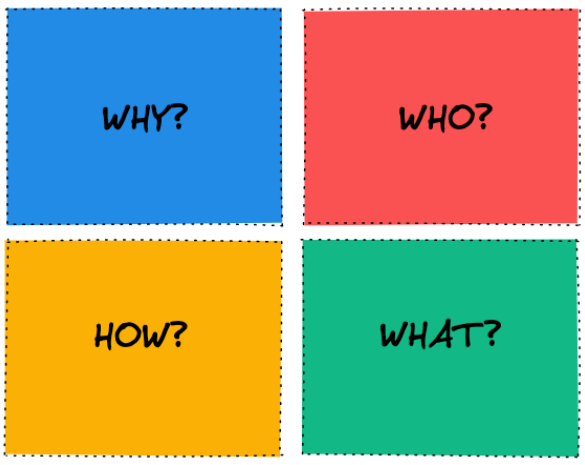
With similar origins and fundamental principles to story and mind mapping, impact mapping is a visual method for feature identification and prioritisation. It quickly illustrates the path from the primary goal to a specific feature by identifying the relevant actors, how they can help achieve the intended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

With the goal at the centre of impact mapping, more time can be spent identifying and defining the goal than anything else. The rest of the exercise could lead the project in the wrong direction without the proper purpose. If you can answer the following SMART goal questions, 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 numerous impact mapping exercises with different groups, you can deduce where there are overlaps and divergence of impact deliverables based on the biases of the other cohorts.

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

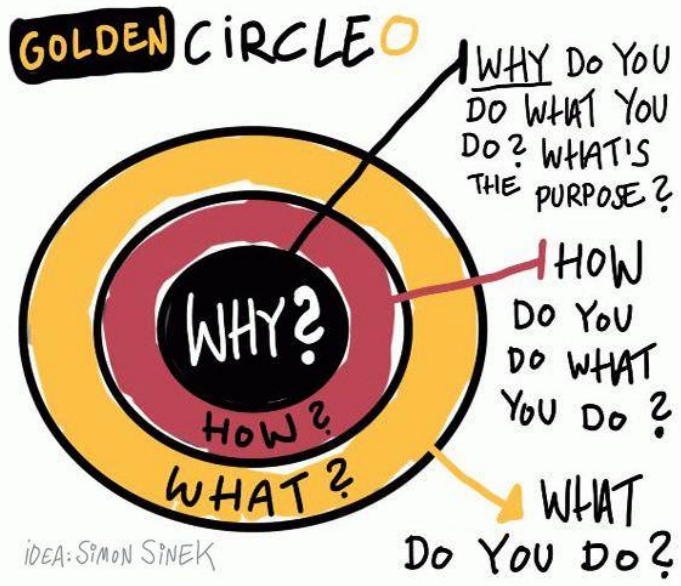


The Goal: Why?

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 clearly explain why achieving this goal will be helpful for the organisation. 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 Golder Circle theory in his book "Start with a Why"[49].



The Golden Circle[50] consists of three layers;

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.

---

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

50 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.

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."[51]

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.[52]

51
https://www.methodsandtools.com/archive/impactdrivenscrum.php

52
https://www.agileconnection.com/article/problem-solving-impact-mapping?page=0

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.

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 reach 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. 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?

We find our " deliverables " 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. Suppose we use impact mapping instead as a general problem-solving tool to design business objectives or improvements. In that case, the "deliverables" are small experiments we will try to achieve the impact that will help us solve our problem.

In an HBR article[53], 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:

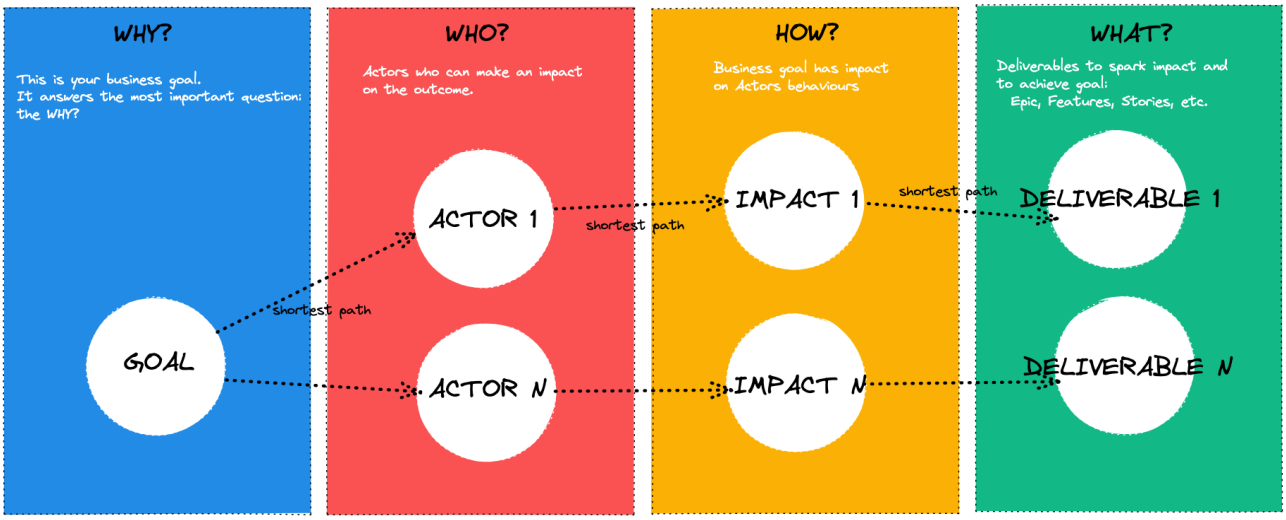
> Outputs are important products, services, profits, and revenues: the What. Outcomes create meanings, relationships, and differences: the Why. Outputs, such as revenue and profit, enable us to fund outcomes; but without outcomes, there is no need for outputs.

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

---

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

Therefore, impact mapping is a great way to create a more explicit link between deliverables and their associated impact and goals. This is useful for reevaluating critical business decisions and prioritising the product roadmap over time. You can see a plain format of this framework below

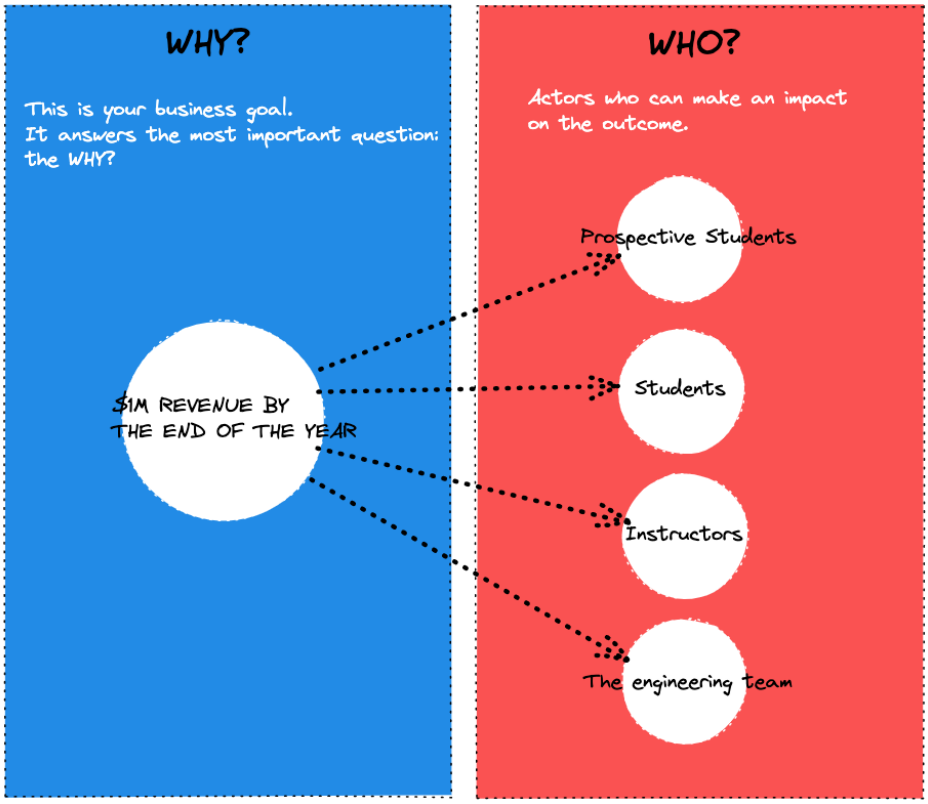


Use case: $1M Revenue by the end of the year

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

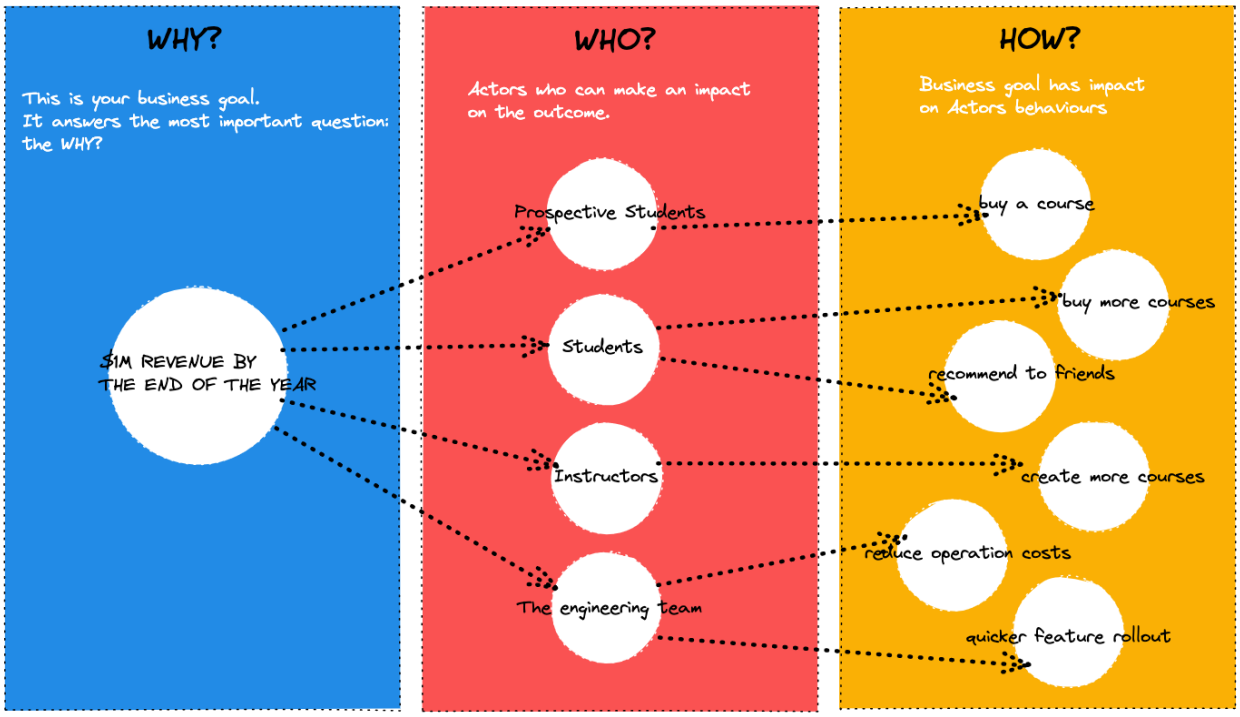
Why represents your main goal. This goal should be obvious. You should avoid confusion when defining your goal so that all team members should understand this regardless of their level of knowledge. Your resources will likely fail to resolve if your goal is not understood, no matter how professional and competent your resources are. Therefore, You have to prefer simple and straightforward sentences instead of using general patterns and sentences that are difficult to understand. You should use more specific words instead of "everyone", which contains a vast mass.

You have defined your WHY. Congratulations, you've taken the first step. Now, ask yourself, WHO is needed to achieve your goal? We need to find which actors we can use to reach this goal. We can define prospective students, students, instructors, and our engineering team as 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. Unfortunately, a solution space consisting of actors that you do not have yet is far from reality. WHO's answer may include actors you don't have yet, but targeting them alone won't make you successful.

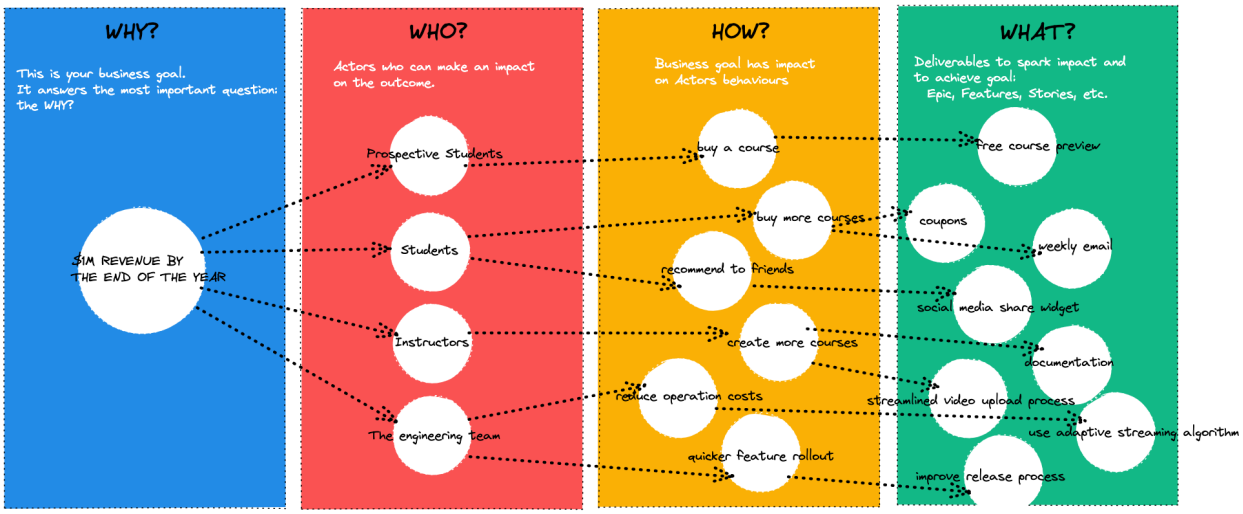
After determining our actors, we need to specify which impacts these actors can contribute to our goal.



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 make the impossible impact for them to do.

You can never expect an elephant to fit into a small jam jar. 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 because 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 specify only deliverables you think are of great benefit to you. First, you can define all your deliverables and then prioritize them based on their benefits. After this stage, the only way to go is to identify and focus on the shortest paths to reach the final goal.



How can you determine which one will provide the most significant 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 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 quickly 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”[54] 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 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 an assortment 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 business place. But, as an organisation grows in scale and the dollar amounts at stake become larger, relying on gut instinct becomes very

[54] https://hbr.org/2020/02/10-steps-to-creating-a-data-driven-culture

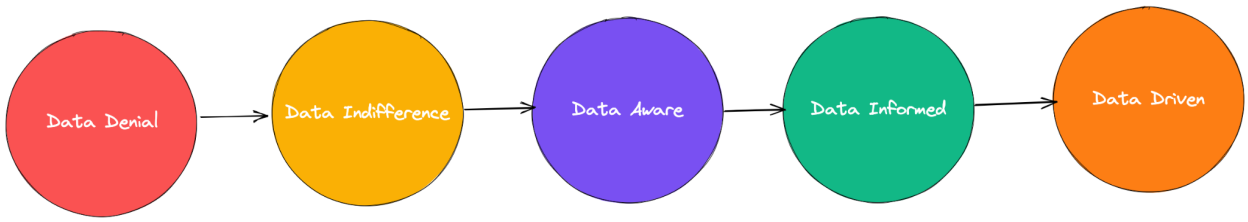
risky. As mentioned above, with data-driven decision-making, the data collection and analysis processes become methodical and rigorous so that the information yields a far more accurate reflection of reality.

The Benefits of Data-Driven Decision Making

Data-driven decision-making[55] 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 specific 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.

55
https://online.hbs.edu/blog/post/data-driven-decision-making

The next step is to become Data-Aware. The business collects 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. After all these steps, data plays a central role in as many decisions as possible across the organisation. 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

Based on the data, you will probably choose the shortest and most beneficial way if you decide which path to proceed in your solution space. Consider our 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.

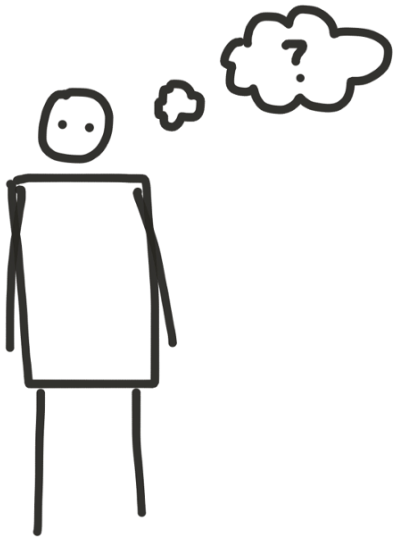
You have to rely on your data to find cost-efficient solutions when deciding which deliverables to crunch in your solution space.

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[56]. 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 apparent, you can be one hundred per cent sure that they will take the risk.

It may seem strange to you, but there is a valid reason why they do this. Product-minded engineers 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 deceptive.

They constantly answer WHY and keep doing this until they realize that there are no different WHYs. You should know that; if your solution's scope 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 software industry's vast problems is never linear. The exact solution to any multivariate problem

56

https://www.google.com/search?q=start+with+why+goodreads&oq=Start+with+WHY+good&aqs=chrome.0.0i512j69i57j0i15i22i30j0i22i30l4.1815j0j4&sourceid=chrome&ie=UTF-8

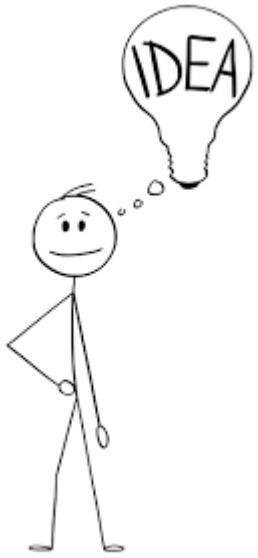
is when the variables are most accurately defined. They aim to clarify the variables to get the most precise solution to the problems you share with them.

Product-minded engineers 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

Product-minded engineers 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.

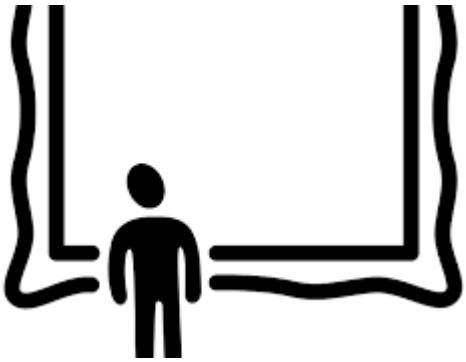
Think Twice



Product-minded engineers often have well-thought-out ideas. They take the time to understand how the business works, how the product fits in, and its goals. They really care about the feelings of your customers. . They often dive straight into 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 speak in vain. They always think twice before sharing their thoughts with their teammates, product persons, managers or other stakeholders 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 in their ideas. If they genuinely believe in their ideas for the product, they are brave to eliminate all gaps and share their ideas again.

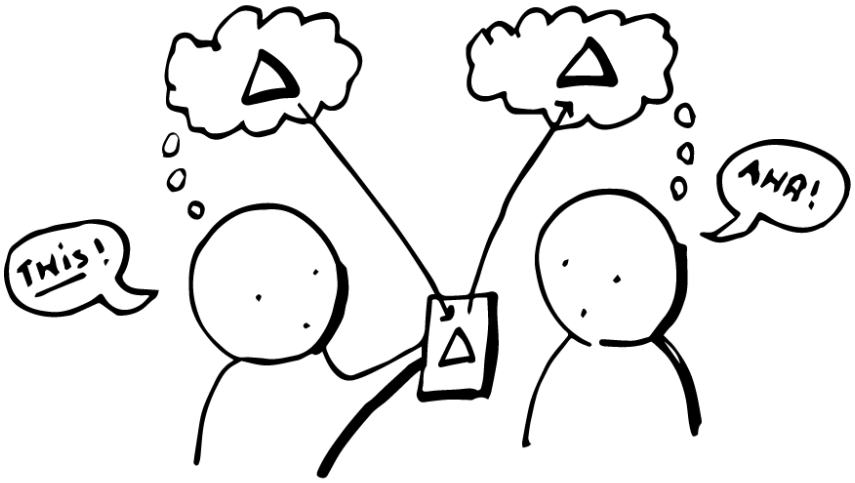
Think outside the box



Product-minded engineers are adept at seeing the big picture. They never focus on a single point. They do not hesitate to use their instincts to know 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 market factors, and competitors' reactions.

They address all the open points and share them with all stakeholders. They always wonder about gaps 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



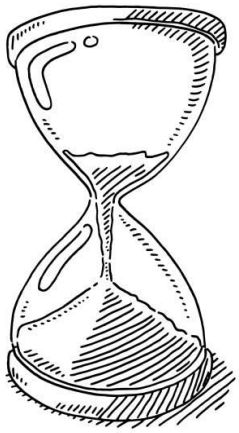
Product-minded engineers are almost always in solid communication with their non-engineered stakeholders. They are also smooth communicators, making it clear they're interested in learning more about how other disciplines work.

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 view can be obtained due to collaborative thinking. For this reason, their communication is vital. And generally, everyone accepts them as part of the solution.

Trade-off Juggler

One of the most unique strengths of product-oriented engineers 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, they get to valuable conclusions quickly using their engineering and product insights.

Always Patient



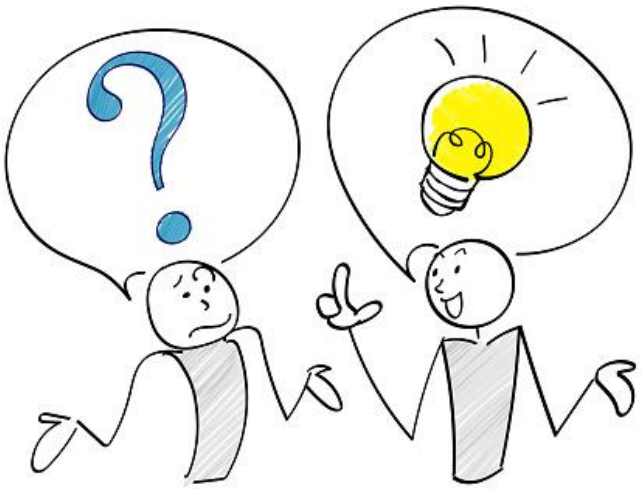
They realize that Rome was not built in a day. They constantly strive to improve their ideas, but they know that the right idea can only make a significant impact at the right time. They wait patiently for the right time to come. While doing this, they smell the market and follow the competitors.

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 schedule.

Edge-Case Master

They can predict many edge cases and address them to the right points. Sometimes, you need 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 entirely with good middle-ground suggestions.

Strike Home with Customers

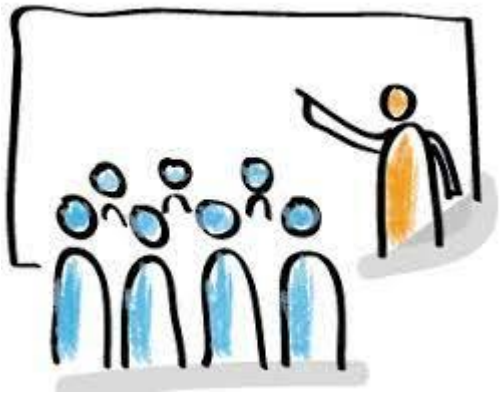


They constantly worry about their users not understanding their intentions clearly. "How can we be sure that our users will be able to use the features we have developed in the way we intended?" They constantly ask themselves the question. While the features are still on the anvil, they are worried about 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 constantly 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 more clever than the customer. They learn from every case that they cannot present 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 storied crunchers. On the contrary, they want to measure the impact of even the most negligible work on their customers. After the jobs 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 behaviour 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, 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 to 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 acquiring these habits. In time, you will realize that you have all the competencies of an excellent product-minded engineer.

Discover your company in time

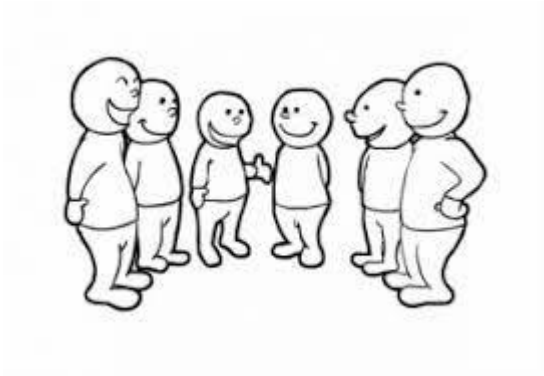
You can start by learning about your business model, how your company made money, or where it spends its money. I'm definitely not talking about learning confidential. Your intention should be to grasp the company's business model. You can research your company's most profitable products, departments, or parts. You can note 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 excellent insights. In fact, product persons love engineers 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 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.

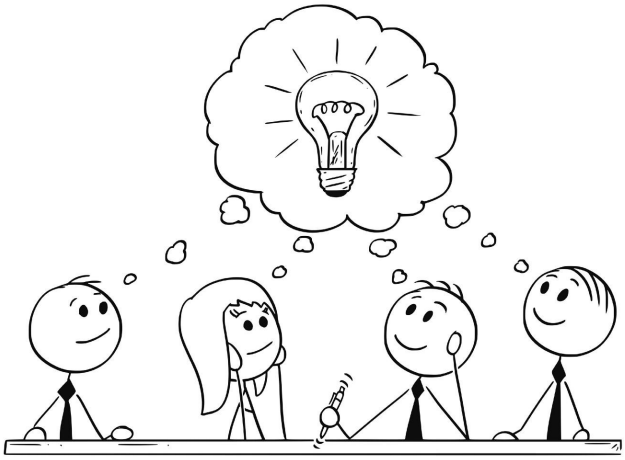
Find who frequently interacts with the customer



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 pair with UI designers, UX people, data scientists, and customers who frequently interact with users.

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

Don't be afraid to suggest new ideas

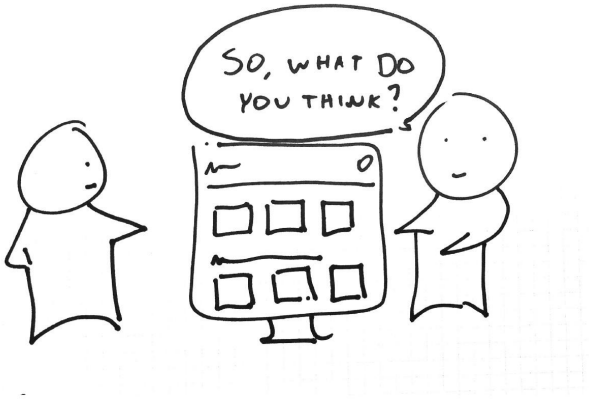


You can organize brainstorming sessions. You can bring well-backed product suggestions to the table in these sessions. Never be afraid to take the initiative. If you really know your business, product, and other stakeholders, it's worth the risk! You can start with small suggestions for the project you are working on. Make sure that your first suggestions do not have enormous efforts. Because massive 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 you work with, you can quickly realize your weaknesses and 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 try 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. Choosing the proper techniques, programming languages, tools, etc., is challenging for your technical needs. 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 generalise 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 essential 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 start; how it works is just a detail for them. The product is 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!!!

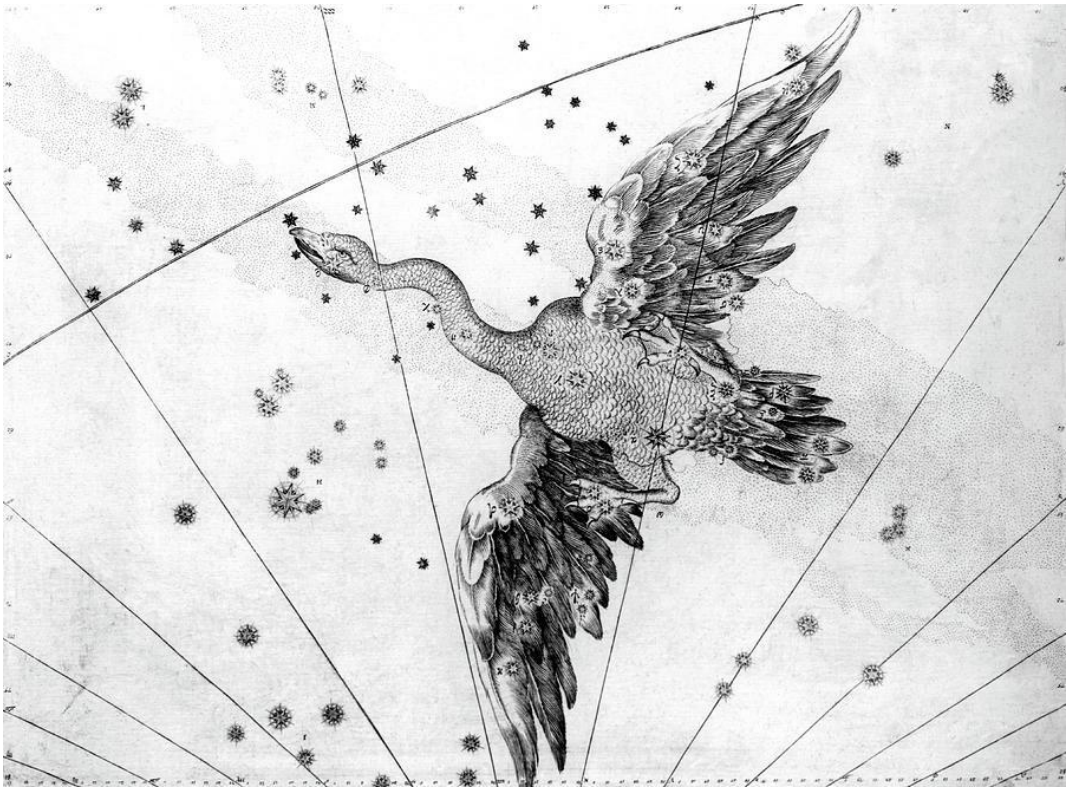
I hope these conversations are familiar to you as well. These conversations end as they started most of the time, and both parties continue to protect their ideas. Code-first engineers are missing something very fundamental despite their good intentions. Programming is about building products that solve 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 many fancy terms that are really meaningless to them. They only focus on how the product works. Of course, this doesn't mean writing lousy code or developing non-scalable products. These are really important because excellent engineering decisions will 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 balancing both sides.

Chapter IV

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

The Cygnus method



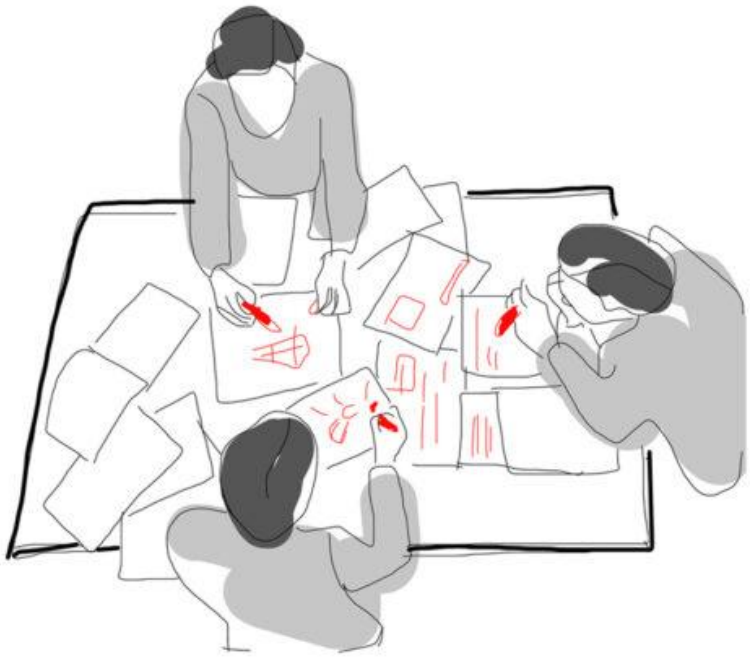
Cygnus constellation[57] is the cover photo of this book. It is the eponym of this method. The Cygnus method is a set of rules that includes the basic principles of eight different approaches that I have applied in my engineering career and takes its name from the same constellation.

Cygnus 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 is to be a good product minded engineer.

57 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 get a conclusion. Therefore, they do not contain stress factors.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 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

To contribute to something, you do not need to be one hundred per cent sure of it. In fact, it is a naive intention based on the idea of contribution, and the result is often positive. 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 primary 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 aspirations 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.

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 perfect 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 issues 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"[58] 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 knowledge from different backgrounds by reading.

58

https://www.google.com/search?q=humble+programmer&oq=humble+programmer&aqs=chrome.0.69i59j0i22i30l4j69i61l2j69i60.159j0j4&sourceid=chrome&ie=UTF-8

Develop your Engineering Perspective

There is no silver bullet for this. Problem exploration includes identifying, framing, 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 understanding 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 their experience level.

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

Pragmatic Thinking and Learning

One of the most precious books I've ever read is "Pragmatic Thinking and Learning: Refactor Your Wetware"[59] 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[60]*.*

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 to think pragmatically. In fact, the key characteristics that the pragmatic programmer should possess are summarized by Andrew Hunt and Dave Thomas;

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

59 https://www.goodreads.com/book/show/3063393-pragmatic-thinking-and-learning

60 https://www.goodreads.com/book/show/4099.The_Pragmatic_Programmer

- 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 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, you tend to think in terms of the practical or logical rather than the ideal situation.

The term pragmatics[61] 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 and beauty have no meaning in and of themselves. Instead, someone who looks at pragmatics would try to understand how they are 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.

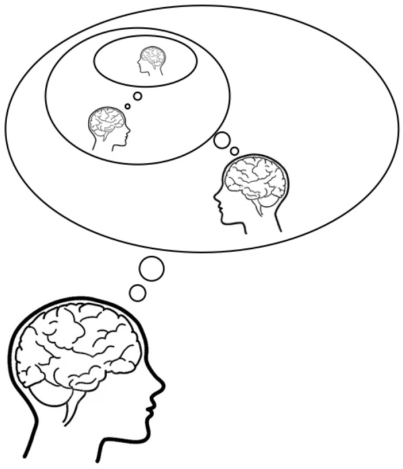
61 https://all-about-linguistics.group.shef.ac.uk/branches-of-linguistics/pragmatics/what-is-pragmatics/

*f 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 more extensive system. When you focus on small system pieces, you cannot see the big picture. You probably won't be a part of the solution if you take things separately.

Use rules for novices, and intuition for experts.

This is one of the core principles of the Dreyfus skill model[62]. As you become an expert, you must learn to use your intuition and trust your intuition. Your intuition is based on your experience. Therefore, they will help you be part of the solution rather than a hindrance in many cases.

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 experience of the whole picture.

62 https://en.wikipedia.org/wiki/Dreyfus_model_of_skill_acquisition

You can never know everything by yourself; the good percentage of the human brain has been measured. In other words, you have a hard limit. Therefore, do not make yourself believe the lie 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.

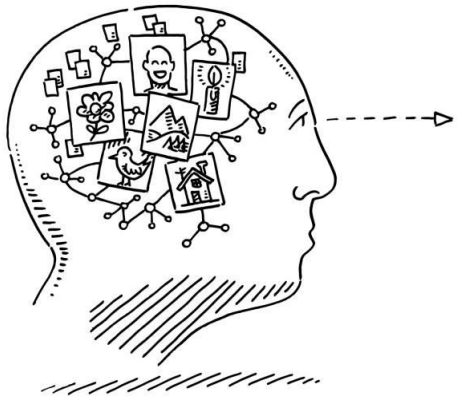
Keep practicing in order to remain an expert

You have to practice constantly to sharpen your skills. The sector we are in is continuously expanding like space. The harder it is to follow an ever-expanding area, the more you have to practice following it as closely. 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. Actually, you don't need a 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[63] concentrate on the significance of how learning is delivered. There are many different ways of learning[64], 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 other 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.

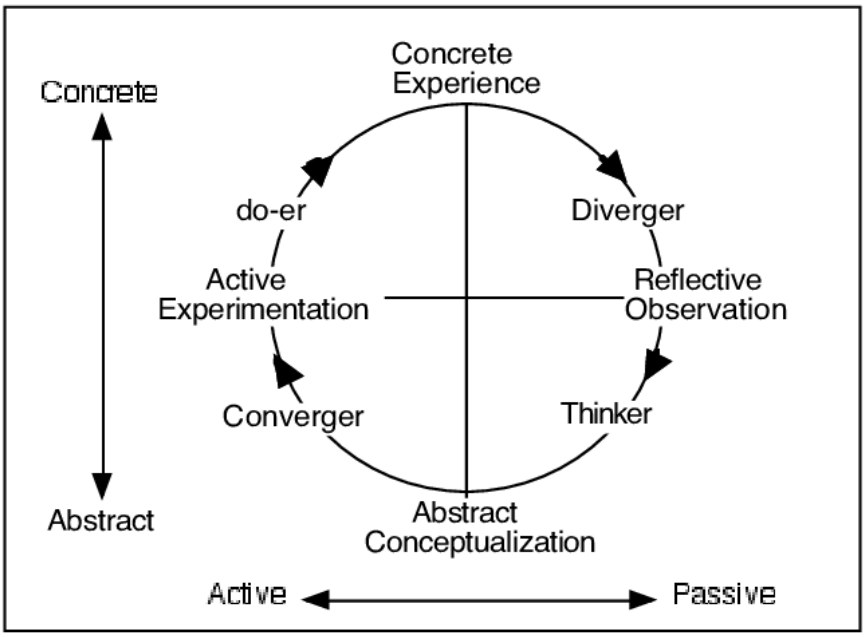
63
https://www.wgu.edu/blog/five-educational-learning-theories2005.html

64
http://www.ibe.unesco.org/en/geqaf/annexes/technical-notes/most-influential-theories-learning

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.[65]



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

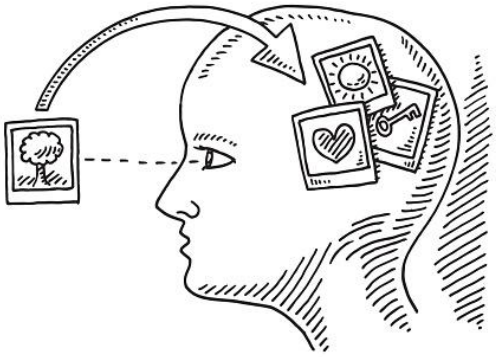
65

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

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.

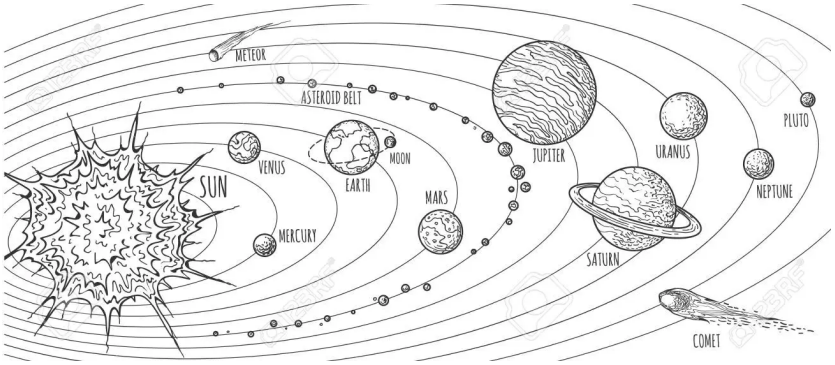
In training terms, therefore, learning is facilitated if the course content and process key into participants' existing experience and are so designed to encourage reflection and the formation of new concepts.

Capture all of your ideas



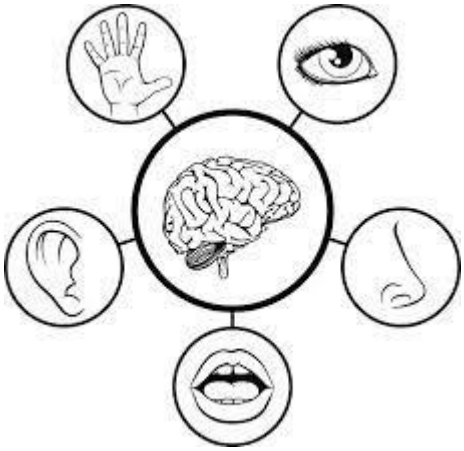
Please consider your ideas. You cannot keep everything in your mind. As I said before, you have a hard limit. For this reason, get into the habit of taking down all the ideas that come to your mind. When you have time, test your ideas and challenge them with others.

Strive for good design



In this image, you see our entire solar system. It would be bizarre to expect you to design the solar system. Just focus on how perfect it is. As a matter of fact, Humankind always tries to develop the best they can. You should try to design extraordinary things. It doesn't have to be divine.

Use your all senses

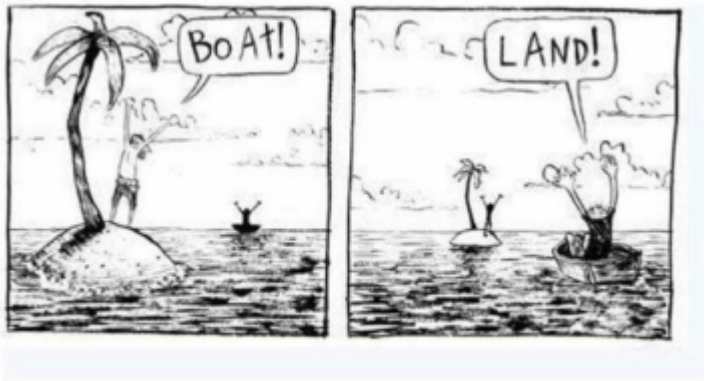


The more senses you engage in a task, the more involved and focused your brain will be. Fiddling, music, walking, etc. Try to use all of your senses as much as possible and engage them with the problem. This will cause your feelings to become sensitive over time. As your senses become more sensitive,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

Step away from the keyboard

Sometimes you have to get away from the problem 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

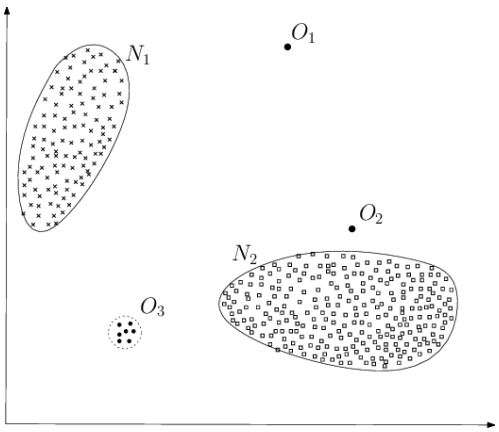


Change your viewpoint to solve the problem:

- Look at it in reverse.
- Exaggerate it to the extreme.
- Change your point of reference.

So you can clarify the solution. Remember, the solution is not linear.

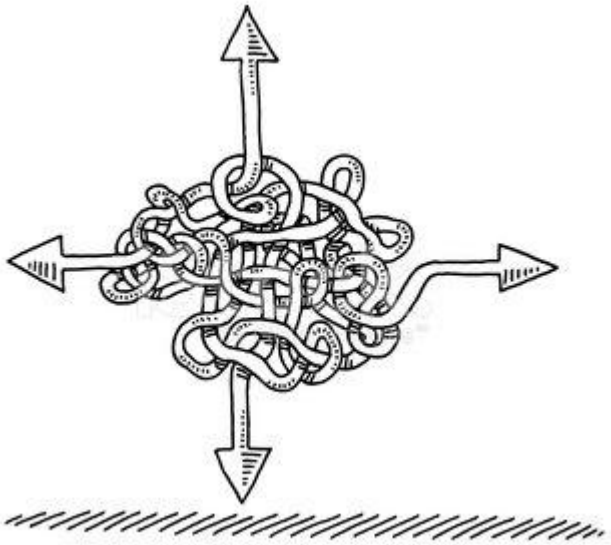
Watch for outliers



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 is seen as impossible as a percentage can create unexpected costs for you. - when they happened.

Be comfortable with uncertainty

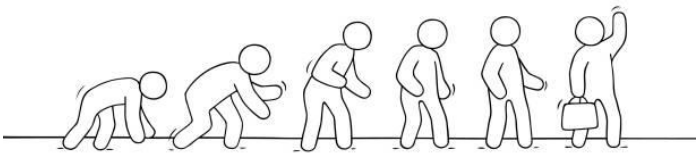


There will always be uncertainties. Uncertainty is a phenomenon in life itself. You cannot ignore this. Try to clarify the uncertainties as much as possible, but do not spend too much time with them. Don't worry about the rest if you really think you've eliminated many uncertainties. In time they will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

Hedge your bets with diversity

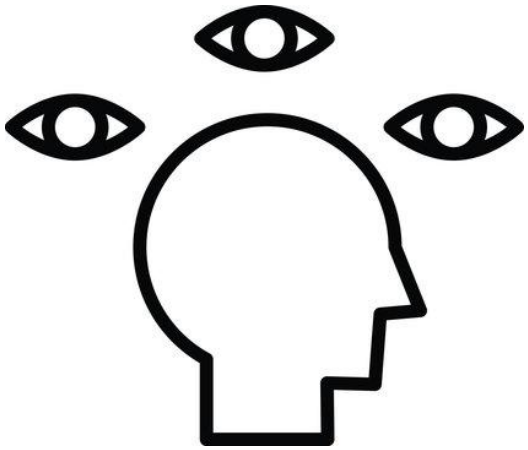
You may not always have accurate ideas. Remember that you are playing a bet when you have to trust your assumptions or instincts. To win the chance, you must rely on the cards. If you don't trust your cards, don't play this bet.

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



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 the abilities that we have developed over billions of years and do not behave like other 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 more unambiguous 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 the truth, 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 what 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.

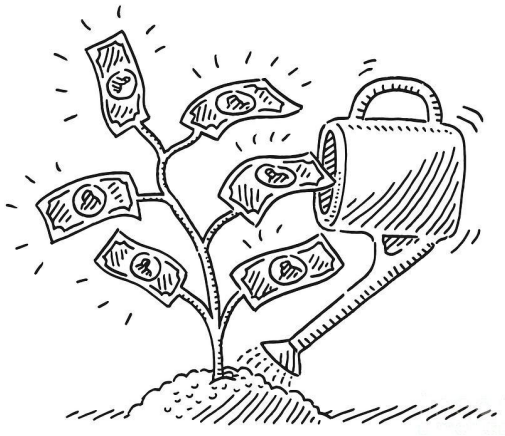
A high goal is frequently easier to reach than a low one because a low goal exerts low motivational force. Some of the hardest jobs you ever accomplished actually seem easy simply because they were a labor of love.

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 precious 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.

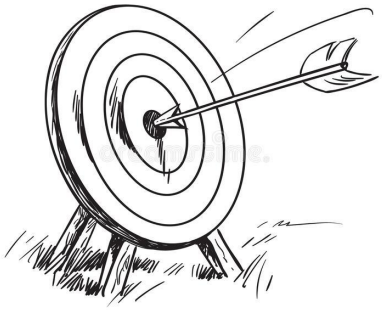
Learn from similarities, unlearn from differences

The similarities actually offer a lot of opportunities to learn something. You can compare similar things 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 tendencies when making your decisions will likely cause you to make subjective decisions. Whatever causes tendencies 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 have no other choice.

Fail fast and learn fast is a suitable method. You can fail fast—and learn quickly—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 distract you in a quiet environment. Although thought is an abstract concept, its results are tangible. If you don't have time to think, 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 crucial reason behind bad choices 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 vessel to someone else or trust the autopilot. For this reason, you should constantly follow the work 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 to not lose the trust in yourself. 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 professionally and in our daily lives. We deal with issues for a substantial part of our time and try to apply the solutions we find to these problems.

Research shows that the most significant reason behind almost 60% of software projects failing in the last ten years[66]; is that the software teams cannot 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 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 specific 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 how we can gain the product-first approach. I have shared methods and experiences that I sincerely believe may benefit you. I hope this book will enable you to include product-first intent in your code-first perspective.

66

https://www.projectsmart.co.uk/it-project-management/the-curious-case-of-the-chaos-report-2009.php