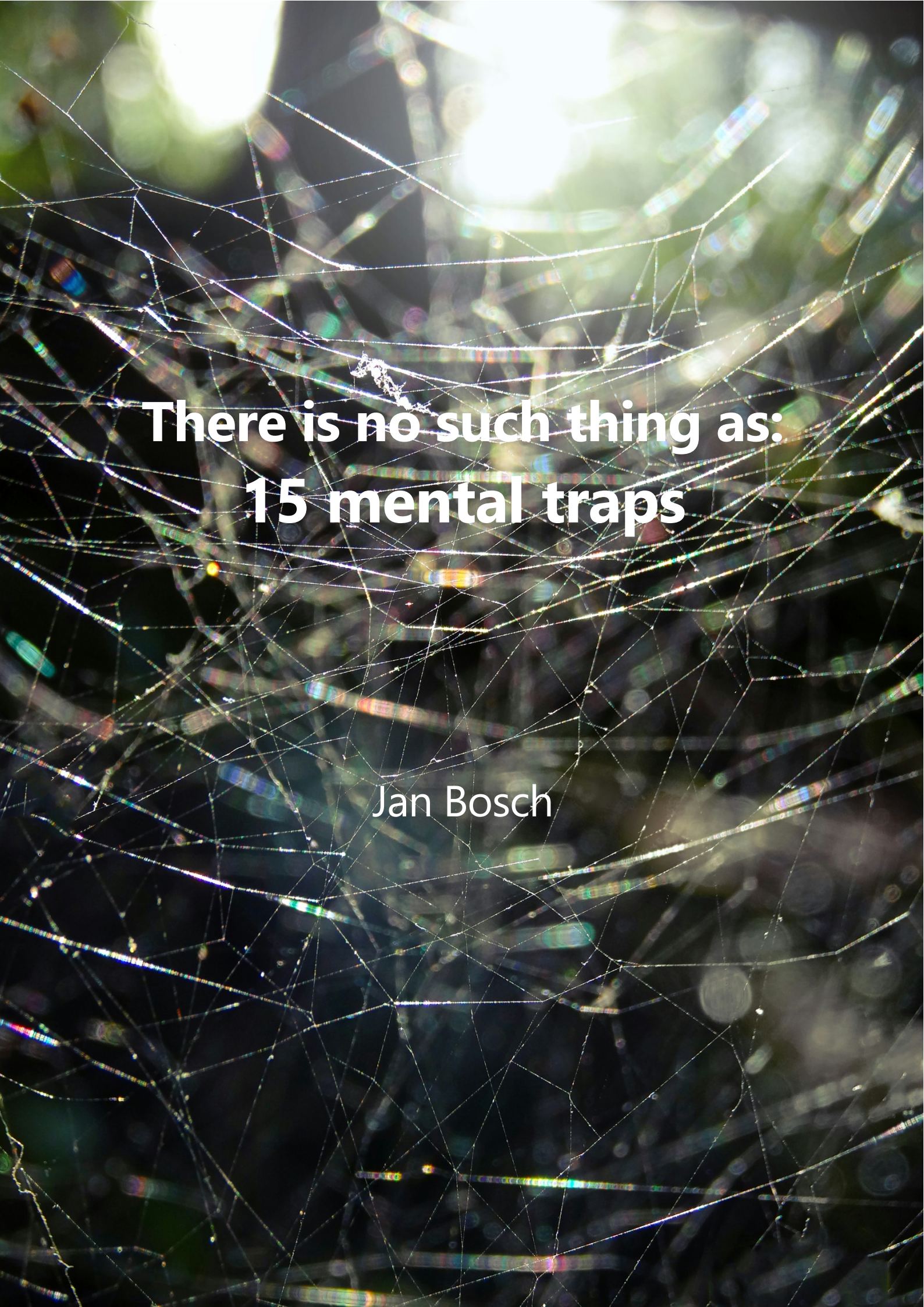


There is no such thing as: 15 mental traps

Jan Bosch

THERE IS NO SUCH THING AS: 15 MENTAL TRAPS

Human intelligence is amazing and has led to everything that we experience in our day to day lives. Every thing we use, every idea we communicate around with our peers, every abstraction we use to deal with a complex problem has been created by a human. Nothing exists until it has first been imagined in somebody's mind.

The human mind, however, is limited in its information processing capacity. In addition, it is extremely energy hungry and uses around 20% of the total energy the body uses. This means that this less than 1.5 kg organ consumes an order of magnitude more energy than any other organ in your body. As we have had bouts of food scarcity for most of human evolution, this means that we have quite a few mechanisms to reduce the "power consumption" of the brain.

These power saving mechanisms include habits, shortcuts, instincts as well as several others, but the main one I want to focus on here is concepts. A concept is a label we associate with an idea or related set of ideas. We have very concrete concepts like a chair, a plate or a jacket as well as much more abstract concepts such as gravity, democracy and friendship. By using such a concept, we can discard all the detail that sits under it and use one label to refer to potentially infinitely complex notions.

The very essence of humanity is our ability to create, use and believe in concepts, independent of these having a physical reality. For instance, the notion of a family or a tribe, the way we identify as members of a nation state or the way military units are molded into a team is entirely based on getting us to believe in a non-physical, artificial concept. At the extreme, people will sacrifice their lives for their unit, tribe or nation state. In the normal course of life, we use concepts all the time to communicate, solve problems and validate solutions. In fact, in many ways, innovation and research are concerned with the creation of new concepts.

Nothing this powerful comes without a backside and concepts are no different. Similar to a map being an approximation of a real-world area, abstracted to only show a highly limited number of aspects about that area, concepts abstract much more complex underlying structures and show only those aspects that are considered relevant when the concept was first developed.

As we as humans ignore well over 99% of all information that reaches us through our senses, we use concepts to filter and quickly map what we are experiencing through our senses into the preconceived notions and concepts that we have internalized in the past. As an illustrative example, think about how

we teach young kids the shape and sounds of animals using picture books. Although the animals are mapped to the same picture, each country has created its own set of sounds that animals make. When I left the Netherlands, I was really surprised that cows, roosters and pigs sound completely different in Sweden and the USA than in the Netherlands.

The mental traps that I want to explore in this post are concerned with mixing up the map with the real world. In information technology or software engineering, we use a whole lot of concepts and especially in a digital context, there is not really a real-world "map" to remind us that the concept is a model of reality and not reality itself. So, here are 15 concepts that I believe often are misused or taken to literally and we do so to our detriment. So, my claim is that the following concepts do not really exist. There is no such thing as the:

1. **System**: Any system we refer to is typically part of a larger context. Hence the term systems-of-systems and often has difficulty to operate on its own without the context being present. Similarly, the parts that make up the system often can be considered independent systems themselves. Considering a system as a unique, standalone, independent entity often is conceptually wrong.

2. **Architecture**: Numerous times I have heard that the architecture of a system is the root of all problems a company experiences. I think that this is a gross oversimplification and one can argue that there is no such thing as a software architecture. Instead, it helps to think of architecture as a continuously evolving set of design decisions.

3. **Process**: During the eclipse of CMMi, many organizations strived to reach level 5 as it would solve all problems. The challenge was considered to be "the process". Reality is of course that people run many activities in parallel and that processes intertwine and are much less clear cut than what one might think.

4. **Team**: Especially in agile, the notion of a team with a high degree of autonomy and high cohesion is often viewed as one of the building blocks of success. In practice, many teams have members that are only tangentially involved and not really involved in the day to day operations. Also, interpersonal issues can really affect morale and effectiveness of teams and make the concept much less powerful than what many claim.

5. **Data**: We have tons of data, many companies I work with say. As if data is some goldmine that we can just start to exploit. In practice, the "data" often is completely useless for anything and we need to be much more specific in what kind of data we collect and the context in which it is collected.

6. **Artificial Intelligence**: I have been flabbergasted by the bifurcation in society around AI. One group believes it is akin to the second coming of Christ whereas another group thinks we are on the way to Skynet and the extermination of humanity. Although there are some strong thinkers on both sides, many have very little clue of what they are talking about and use the concept completely inaccurately.

7. **Ecosystem**: No company operates in a single business ecosystem or software ecosystem. There are many overlapping, interconnected ecosystems that every company operates in and focusing too much on "the ecosystem" is a gross simplification that can easily cause one to lose sight of the true complexity.

8. **Customer**: The customer is one of the most powerful concepts in business. In practice, this individual often does not exist, especially in B2B contexts. Typically there are multiple stakeholders in the customer organization that all influence buying decisions and that use the

offering in different ways. Simply talking about “the customer” often causes more harm than good.

9. **Product**: Companies have a tendency to talk about “the product” a clearly identifiable entity, but especially in software, there often is quite a bit of configuration and customization for different customers. Also, similar to a system, it is often hard to draw boundaries around the product, for instance due to integration with systems on the customer’s end.

10. **Business model**: The business model is not as simple as it looks. Sales very often is quite open to discussing different ways of monetizing. Especially in software, where you are not selling a widget, but rather an initial system, a promise for the future and integration, there are multiple dimensions and the business model is not as clear cut.

11. **Supplier**: It is easy to talk about a supplier that simply delivers parts to us when we need it at a price we agreed upon. In practice, suppliers of software are also continuously evolving their offering and easily become innovation partners rather than simple suppliers. And, not uncommon, suppliers may easily become competitors when they grow the functionality in their offerings.

12. **Company**: Especially large organizations tend to be conglomerates of smaller communities that compete with each other (sometimes more than with outside parties), where outside parties are involved in ways that go quite deep and where ownership relationships of legal entities are extremely complex and messy.

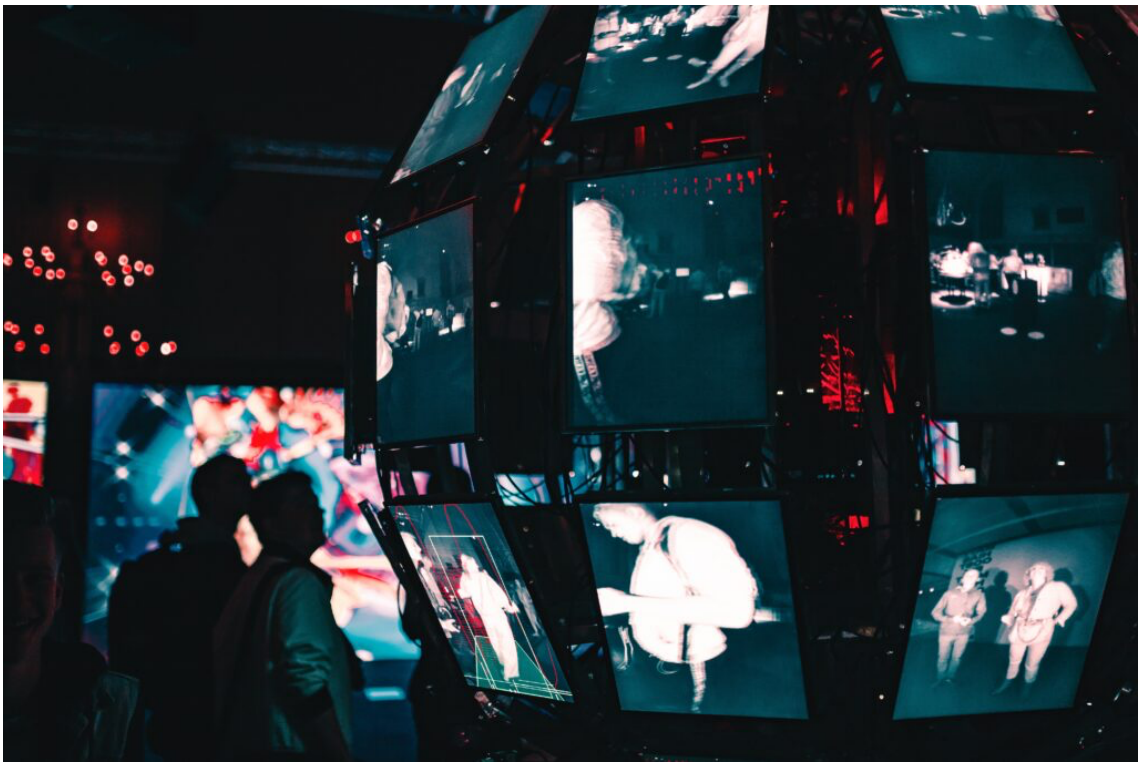
13. **Innovation**: Few terms in business are as overloaded as innovation. It is thrown around everywhere and in general means “good”, but is very poorly understood. In fact, I often feel it does more harm than good in conversations and banning the term would force people to say what they actually mean.

14. **Customer value**: For the last decade or so, I have conducted research on value, together with other researchers. My conclusion is that in the majority of companies there is no agreement at all as to what constitutes customer value. This leads to enormous inefficiencies in most organizations as people focus on different, conflicting priorities in the name of customer value.

15. **Platform**: Finally, platform is, once again, a poorly understood concept that is severely overused in most companies. I know of at least five different interpretations of the term and frequently run into discussions where people talk about the platform using different interpretations of the word.

Concluding, in industry and society, we tend to use words and concepts that represent, often highly complex, ideas. However, we tend to mix the map with the underlying landscape and ignore the limits of the abstractions of the concepts we use. In this series, I hope to increase awareness of the limitations of the concepts we use with the intent of both allowing for more careful use but also for the development of new, more accurate concepts that capture more of the relevant aspects in the abstraction and perhaps ignore aspects that were relevant earlier.

TABLE OF CONTENTS



THERE'S NO SUCH THING AS "THE SYSTEM"

As humans, we like to label things and put boundaries around them. As we have limited intellectual and information-processing capabilities, this helps us abstract otherwise complex concepts and use them in building even higher-level concepts that we can use to explain, predict or create things. Of course, the nature of this process is that it helps us ignore the vast number of aspects associated with a concept and focus on a very small number of remaining aspects that are considered to be integral to the notion itself.

When we work in teams, this is even stronger as we tend to build a team or organizational vocabulary that helps team members communicate efficiently. We all know of the TLAs (three-letter abbreviations) most companies use. Of course, it doesn't only concern efficiency but also points to an insider-versus-outsider perspective as the comfortable use of TLAs signals that you're 'in the know.'

The challenge with referring to "the system" is threefold. First, it can easily lead to misinterpretation. The term "system" has quite a few interpretations. Of course, the basic one is where we refer to "a regularly interacting or interdependent group of items forming a unified whole." However, there are many other definitions, including the capitalist system, the Newtonian system of mechanics, the system of species or a systematic process for accomplishing a particular task.

In addition, drawing the boundary around a system is often particularly hard. Any system we refer to is typically part of a larger context. It usually has difficulty operating on its own without the context being present. Similarly, the parts that make up the system often can be considered independent systems themselves. Considering a system as a unique, stand-alone, independent entity generally is conceptually wrong. Hence the term "systems-of-systems," which appears more and more frequently in the last decade.

Even within the same organization, the actual interpretation can easily differ in non-trivial ways. This can easily lead to confusion or even conflict when people operate under different definitions of the same term. In the embedded-systems industry, a typical definition referred to the physical product being sold

to customers. However, with digitalization and connectivity, for most companies, the system encompasses the physical product as well as the mobile application to interface with the product and the cloud solution to store data and functionality not incorporated in the physical item.

A second problem is that, in many contexts, the system can become a fixation for the organization, meaning that everyone is considering it to be the source of all problems as well as where the solution has to originate from. As Einstein famously said, no problem can be solved by the same level of consciousness that created it. This is also true for systems where many problems are created by outside forces.

For example, I've worked with many companies on platforms and I often am brought in when a platform fails to deliver the benefits everyone expected when it was first created. A typical pattern is that requests for new functionality and new configurations of the platform take too much R&D effort and time. Management wants to know what the problem is and often leans toward outsourcing and getting rid of the idiots in R&D. Often, though, the situation is that the platform hasn't received any resources for architecture refactoring and technical debt management for years. Instead, numerous customer requests have been forced in with hacks instead of proper engineering as the business side prioritized closing the next deal over maintaining the long-term integrity of the platform. In this case, the platform and R&D organization aren't the problem, but rather the challenge is outside of the system's scope.

Third, focusing on the system very often becomes an excuse for not making the necessary changes in the organization. By simply stating that the system is the problem but that we have no way of changing it because of "reasons," it becomes easy to keep going forward with the anchor holding you back. The challenge in this case is that each individual is correct in that from their perspective, it's impossible to change the system. It requires a coordinated, cross-organization effort to actually drive the necessary changes in a system-wide fashion. This, however, calls for coordination and action outside the scope of the system itself.

An illustrative example at the business ecosystem level is where companies are held hostage by others in the same business ecosystem. In a value chain, especially upstream partners are explicitly told not to change and adopt new technology. The threat is that this partner will be ostracized and kicked out of the system and replaced with someone else who will play ball. This is of course a great way to maintain the status quo in the short term, but it's devastating in the long run. If individual companies in an ecosystem don't change continuously, it will cause an ecosystem-wide disruption, replacing all companies with a new set of players serving the end customer in a better way.

There's no such thing as "the system." Of course, we can use the term in our work, but we have to be very careful to ensure that everyone uses the same meaning. Also, we need to avoid getting fixated on the system itself and realize that there are layers below and above to consider. Finally, we should never use the system as an excuse to avoid or delay change. As Edwards Deming so eloquently said, your systems are perfectly designed to give you exactly what you're getting today. If what you're getting isn't what you're looking for, it's time to redesign your system.

THERE'S NO SUCH THING AS "THE ARCHITECTURE"

It's highly controversial to claim that there's no such thing as "the architecture" of a system. On the one hand, folks in R&D use the term to indicate that the technical debt in the system has reached a point where more resources need to be dedicated to architecture refactoring. On the other hand, those on the business side who have at least a modicum of technology understanding use it as an argument toward their peers to explain why things take longer than expected or why we need to divert resources to address architectural issues.

Another situation where the architecture is used is when a faction in the company wants to start a project to replace the existing system or platform. The argument tends to be that the architecture of the old system is so old and poorly suited for today's needs that it can't be fixed. Hence, we need to start from scratch.

In my view, this use of the term "architecture" is driven by a poor understanding and some misconceptions about the nature of software. There are at least three main ones I believe are worth discussing.

First, the use of the term "architecture" suggests that there's a clear boundary between the architecture and the rest of the system. This is of course incorrect. Even if the system is broken down into a small number of large subsystems, this isn't really what the notion of architecture is about. Instead, it helps to think of architecture as a continuously evolving set of design decisions. These design decisions may affect the breakdown of the system into large components but also low-level aspects such as task scheduling. Architects concern themselves with design decisions that have a system-wide impact – the latter example around scheduling affects the real-time behavior of the system and is hence an architectural concern.

Second, many view the architecture of a software system as a static structure. This is in part because we borrowed the term from the construction industry. As soon as we build things out of atoms, it becomes difficult or at least expensive to change the structure.

Interestingly, this isn't even true for buildings. Instead, the concept of architectonics structures buildings into multiple levels where each higher level is more expensive to change. For instance, non-

load-bearing walls can quite easily be moved whereas this is more difficult for the load-bearing ones. And although we often think that the location of a building is permanent and immutable, there are several examples of older, wooden buildings in Sweden that have been moved.

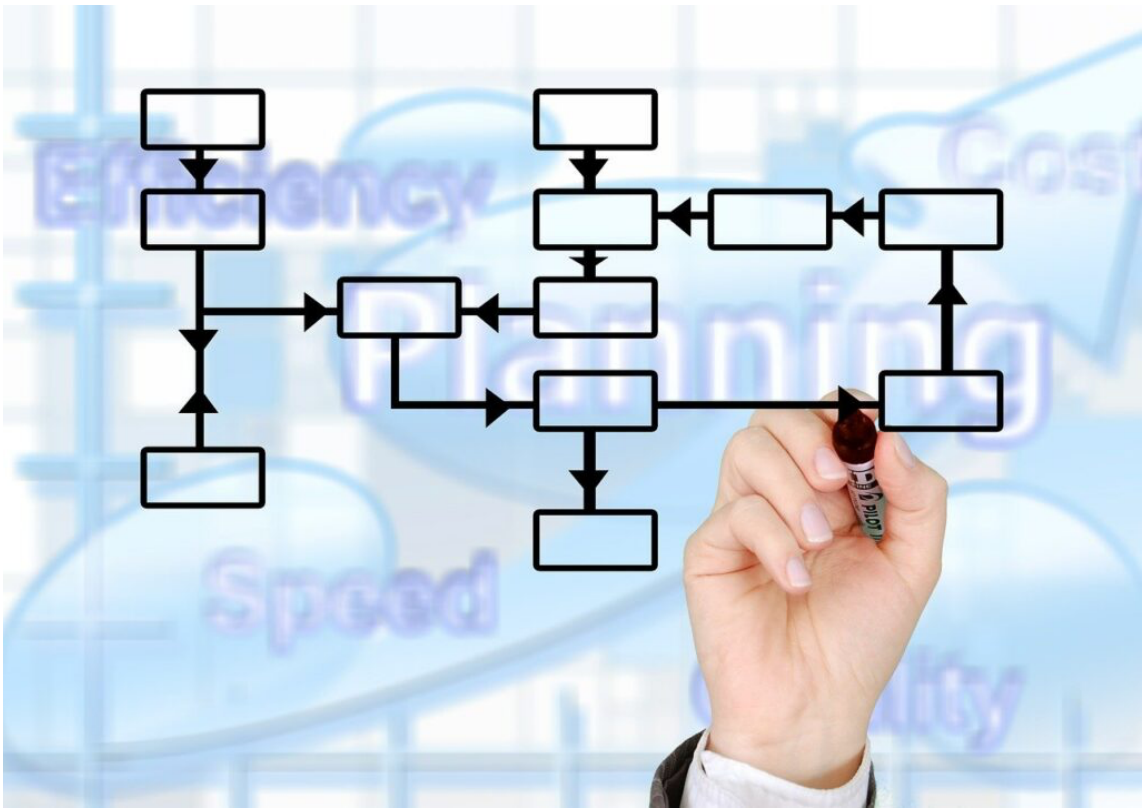
For software, changing the architecture of the system is a continuous process in response to the requirements put on the system, the evolution of technology and other factors. Earlier, we thought of software systems as we think of humans in that aging is unavoidable. However, we now know that the aging of software systems is basically the consequence of underinvestment in architecture refactoring. I claim that no software system that has received sufficient resources for managing its evolution will ever have to be retired as long as there's a need for its functionality.

A third, important factor is that, especially outside of R&D, many fail to understand the connection between the business and business strategy of the company and the architecture of the systems aiming to realize that strategy. In many ways, the software architecture is an embodiment of the business strategy and a change in business strategy will, typically, necessitate a change in software architecture as well.

A contemporary example is the transition from one-off customer projects in many companies to adopting DevOps where all deployments of systems in the field need to receive software updates every couple of weeks. A company using a platform for one-off customer projects can afford to use a clone-and-own strategy and create a unique copy of the software for each customer project. When adopting DevOps, this approach becomes prohibitively expensive and the platform needs to become configurable so that each customer-specific version is simply a configuration of the platform. This has significant implications for the architecture of the platform as well as the build system, test infrastructure and deployment process.

Of course, this is an incarnation of the BAPO model, which states that the architecture and technology choices are or should be a consequence of the business and business strategy choices. When leaders lack the understanding that the design of a system is spurred by business drivers, it easily leads to a situation where R&D gets blamed for being slow and unresponsive while there's often a fundamental disconnect between business and technology.

Numerous times I've heard that the architecture of a system is the root of all problems a company experiences. I think this is a gross oversimplification and it can be argued that there's no such thing as a software architecture. At least not in the way that most people tend to use the term. There's no clear boundary between the architecture and the rest of a software system. The architecture is continuously evolving in response to new design decisions and the evolution of existing ones. Finally, the architecture is the embodiment of the company's business strategy and any change to the strategy affects the architecture as well. By all means, use the term architecture, but make sure that you know what you're talking about. In the end, whatever good and beautiful things we build, these end up building us.



THERE'S NO SUCH THING AS "THE PROCESS"

Although the Agile aficionados tried really hard to kill the notion of process in their efforts to remove the heritage of the capability maturity model, the fact is that every team, company and business ecosystem uses processes. These can be formally defined, to the point of individuals or companies being legally liable if the process isn't followed, like in the medical space. Or, at the other end of the spectrum, they can be highly informal and grown over time as teams figured out how to do things.

As organizations grow and more and more people become involved in the key value-delivery activities, the need for coordination goes up exponentially. The preferred approach to coordination is to use architecture. In this case, the architecture defines the interfaces between components and teams working on different components can operate independently as long as the interfaces aren't affected. However, for every architecture, there's a point where we need to adopt processes to manage coordination between teams and along the organizational boundaries.

We tend to use the term "process" quite lightly as "a series of actions or operations conducing to an end." In practice, however, any process has quite a few interfaces as well as individuals who are expected to behave in certain ways and generate specific outcomes for the process to keep moving forward. Also, once a primary process takes hold in an organization, secondary processes seek to hook into it with the intent of piggybacking and simplifying their own coordination.

In many ways, this is a positive thing as it provides structure and scaffolding for the company to operate in as well as providing a heartbeat for the organization as many processes are periodic. In a lot of companies, however, over time the process grows into an unwieldy set of activities, interfaces, dependencies and expectations that become part of the walls and fabric of the company. As long as no change is needed outside of individual activities, this will allow the organization to deliver value to its customers and optimize each step. The challenge starts when change is required and everyone in the

organization realizes that they live in a straightjacket that's impossible to change unless everyone changes at the same time – which of course is impossible to do.

This is where everyone starts to blame "the process" as being the root of all evil, for several reasons. First, there's of course sheer frustration about the inability to change where it really is necessary to do so. Second, it provides an excuse to not change and keep things as they are. This victimhood approach is especially dangerous as the company easily becomes more and more out of touch with the market and its customers, rapidly increasing the risk of disruption.

The principle is that if you can't change at one level, you attempt to change one level up. This continues iteratively until you reach the ecosystem level. There have been situations where entire business ecosystems have been disrupted as they were unable to respond to the changes around them, such as the shipyard industry in Europe in the 1970s and 1980s.

The misconception held by many is that there indeed is such a thing as "the process." Of course, there are documented and certified documents of how companies, teams and individuals are supposed to work. But there are at least three main concerns with using the term as many do: definition, completeness and boundaries.

First, as with the other concepts we've discussed so far, the notion of "the process" is interpreted very differently by people inside and outside the organization. Similar to the elephant in the Indian fable, everyone has a different experience and interface to the process and another conceptual model of what everyone assumes is the same. This tends to lead to confusion, misinterpretation and tension between individuals and teams.

Second, although very intelligent people have spent vast amounts of time defining, modeling and describing processes, the fact is that any formally defined process tends to describe a fraction of all activities, artifacts and dependencies involved. In many ways, the description only captures the tip of the iceberg and not the 90 percent below the waterline. One illustrative example is when workers in a company go on an alternative strike where they only conduct the activities formally specified in the process descriptions and refuse to do anything not described. As one would expect, things rapidly grind to a halt.

Finally, there's the challenge of boundaries. Formal process models have a clear beginning and end, with steps, dependencies, artifacts, and so on. In practice, no process is truly independent and has significant dependencies and relations to other activities. In that sense, what we refer to as a process may easily be a randomly selected set of activities in a cloud of many more actions, artifacts and dependencies. Even if we model a set of processes and dependencies between them, the way the interconnecting lines get drawn is highly debatable in most contexts.

The notion of "the process" is highly elusive in most contexts and can easily be abused as an excuse to avoid or delay change. This significantly increases the likelihood of disruption of the offering, company or ecosystem. Instead, we have to treat the concept with healthy skepticism, realizing that definitions, level of completeness and perceived boundaries are highly arbitrary and strongly depend on the viewpoint of the defining individual. As one of the key people in process thinking, William Edwards Deming, said: "We need to work on our process, not the outcomes of these processes."

THERE'S NO SUCH THING AS "THE TEAM"

There's a powerful story in the narrative of Western civilization around the team. It comes up in the context of the military, especially special operations, in the startup community, especially in the early phases of a company, in sports as well as in the leadership literature. The idea is that a group of people can be merged into what's almost an organism where everyone is aligned around the same goal, committed to putting all their energy towards that goal and operating in perfect harmony, without dissent and complaint. Team members act fluidly without having been asked as they know what needs to happen and altruistically sacrifice their own needs and wants for the greater good of the team.

As you read the above, I think everyone understands that the notion of a team is an illusion or, at best, an unattainable platonic ideal. As human beings, we all have our personal goals and ambitions. Of course, part of those ambitions is to feel part of a community and be recognized as a valuable member of that community, but few people will sacrifice everything to achieve that. We all have our own lives to live and need to (or should) spend time on our self-actualization.

In the software development context, our story is the notion of an Agile team: a group of people operating in sprints and scrums. Like in rugby, where some of the terminology originated, the idea is that the team acts as one with every member contributing his or her best while supporting the others where they need it.

There are three challenges with teams that I see in practice. First, especially in multi-disciplinary teams, different disciplines approach challenges in very different ways, which easily leads to conflicts. A typical example is between user experience (UX) folks and engineers. UX is concerned with providing the best possible user experience for each of the features provided by the system, which often involves experimentation with users. Engineers typically want to build a new function and move on to the next thing.

Second, even for homogeneous engineering teams, there often is a conflict between those who are more inclined toward architecture and have a long-term focus and those who are more concerned with simply getting functionality implemented, even if it means incurring some technical debt. As there's no absolute and objective measure to balance long-term needs such as minimizing and paying off technical debt and short-term needs of delivering functionality, significant debate can ensue. This can be viewed as creative conflict within the team, but it does create tension and negatively affects team harmony.

Third, as humans, we have different temperaments, but one of the common behaviors is to seek to climb higher up in the hierarchy. In engineering teams, there's often limited opportunity to ascend in the formal management hierarchy, but the reputational, informal hierarchy is just as relevant for most of us. And this leads to tensions and conflict as different people are vying for the same position.

As a consequence, the notion of a team is often not living up to the platonic ideal that many make it out to be. Of course, there's a team as in a group of individuals under a common label, but the level of performance and the amount of tension and difficulty experienced may still be significant. In the end, a team is a collection of individuals with their own plans, ambitions and goals.

For leaders, one of the key challenges is overcoming the limitations of teams to maximize performance. To accomplish this, it's important to be realistic about the drivers of the individuals making up the team. Although the leadership literature is infinitely long, it's my experience that three activities are key for leaders to engage in.

First, it's critical to identify what makes each team member tick. This requires significant empathy and reflection as nobody will just throw out their goals, ambitions and quirks. Typically, what people say is not what's really going on in their heads, even if they truly believe what they're saying. A leader needs to uncover team members' actual drivers and use this insight to optimize their performance and that of the team as a whole.

Second, key aspects of self-actualization are growth and development. For a leader, one of the mechanisms to help team members is identifying opportunity areas for development. All people have blind spots that are limiting their ability to grow and develop and they need constructive feedback from others to start seeing them. This helps them develop and grow and can significantly improve harmony in the team.

Third, conflicts in teams are unavoidable for all kinds of reasons. These can be open conflicts with people talking loudly with each other or passive-aggressive conflicts where people ignore each other and avoid collaborating. It's the job of a leader to find ways to use conflict in the team as a source of energy and development.

Especially in the Western world, the notion of a team is a very strong concept and has a compelling story around it. In practice, many realize it's a bit of a myth as people are different, members from different disciplines in cross-functional teams approach challenges differently and we often compete and jockey for positions. Leaders are responsible for making teams as productive as possible. This requires understanding the drivers of team members, providing constructive feedback to make people grow and converting conflicts into positive growth and development opportunities for the team. As John C. Maxwell so eloquently said, "Teamwork makes the dream work, but a vision becomes a nightmare if the leader has a big dream but a bad team."

THERE'S NO SUCH THING AS "THE DATA"

One of the typical patterns I experience a lot when working with companies on digitalization is the claim that they have all the data in the world. They can just pick what they want from the candy store to get the data-driven insights they're looking for. When inspecting the data in detail, it rapidly becomes obvious that the amount of data available is indeed humongous in volume but highly limited in terms of usefulness, if not entirely useless.

There are many reasons why data can't be used for the purpose we had imagined, but three very exemplary ones include lack of context, variants and confounding variables. Many companies, especially in the embedded-systems space, have collected data for decades, but the data collection was typically focused on quality assurance. Outlier data was collected as it often was indicative of specific defects that could then be more easily remedied by service staff. The problem is that for quality assurance, it's usually sufficient to record that an event has occurred and it's less relevant to know under what circumstances it happened. So, the context of the event isn't recorded, which makes the use of this data for any other purpose basically impossible.

For example, in one of the automotive companies I work with, the vehicle recorded events where the engine temperature became too high. The initial purpose was that service staff could check these events when the vehicle was in the workshop as it would often indicate particular problems with the engine that the technician could then more easily fix by replacing specific parts. The engine design team became aware of this data being available and wanted to use it to optimize the design of future engines to avoid the overheating problem. At this point, it became clear that the lack of context, ie what happened to the vehicle that caused the engine to overheat, was lacking completely and hence the team couldn't use the data.

Second, many SaaS companies have one version of their product and use DevOps to ensure that all servers are running the same version of the software. In the embedded-systems space, there tend to be

several, if not many, variants of the product out in the field. Customers also configure the product according to their specific context and purpose and that of their company. Finally, as the product is owned by the customers, access to the data generated by the product isn't automatic and needs to be agreed upon with them. Consequently, many companies have limited data for each specific variant and configuration of the system.

For example, in the automotive industry, a typical company will ship between 500,000 and a million vehicles per year. That seems like a very rich source of generated data. However, each region in the world puts its own requirements on these vehicles and consequently, a vehicle sold in the US can't be used in the same data pool as one sold in Europe. At least not unless the data is verified to indeed be comparable. In addition, vehicles come in different models, each of which will certainly not be comparable to the others. On top of this, each model for each region has several variants and each variant has numerous configuration options available. Consequently, even if shipping a million vehicles per year, many companies only have a smaller number, eg a few tens of thousands, of vehicles for each region-model-variant combination.

Third, the typical case when working with data is that we're looking for a specific factor that can't be measured directly. Instead, we measure proxies with the intent of using them to determine or estimate the value of the factor we're really interested in. The challenge is that in many cases, numerous confounding variables cause the relation between the proxy and that factor to be tenuous at best.

For example, in telecommunications, one of the key factors operators care about is customer satisfaction. The idea is that high customer satisfaction leads to lower churn, which is a major contributor to profitability as the cost of customer acquisition is quite high for operators. Although it's possible to measure customer satisfaction by asking people (assuming you can trust what they say), if we want to have a real-time and continuous assessment, we need to measure how the network is being used. Proxies for this are the data volume consumed and the bandwidth experienced. Although these are perfectly measurable, the question is what the actual relation to customer satisfaction is. It turns out that the use of a mobile network is highly influenced by numerous other factors, including the weather (people mostly use Wi-Fi at home), large events (like football world cups), vacation periods, and so on. None of these variables have a bearing on customer satisfaction, but they do influence the proxy actually measured.

In my experience with quite a few companies, the usefulness of historical data is often highly limited. Instead, with the increasing adoption of DevOps, the better way tends to be to extend the software in the systems out in the field with the data collection functionality for the specific variable or factor you're looking to measure. This allows you to evaluate the usefulness of the collected data and change the code if it turns out that things aren't optimal.

Second, rather than trying to determine correlations and causations between factors and variables using historical data, use A/B testing instead. When done properly, A/B testing allows for quantitative, statistically relevant conclusions concerning causation and relations between things that can be measured and things we care to know about. Of course, there's the never-ending debate between frequentist and Bayesian statistics aficionados, but in my view, it's more important to simply have the data and be able to analyze it.

Third, although this is relevant for other contexts as well, reducing the number of variants of systems in the field and ensuring the data generated by each system is legally available to the company is critical for easier analysis as a higher number of comparable system instances automatically leads to higher statistical relevance and shorter testing periods.

There's no such thing as "the data." Instead, each question we seek to answer and each variable we look to track over time needs specific data to be collected from systems in the field. In practice, historical data often lacks context and is highly limited in volume. In addition, confounding variables complicate

analysis to a significant extent. To address this, use DevOps to update software to systems in the field to generate data when you need it. Use A/B testing to establish statistically relevant causations between measurable factors and limit the variants and configurations of deployed systems. As Mark Twain so beautifully said: “Data is like garbage. You better know what you’re going to do with it before you collect it.”

THERE'S NO SUCH THING AS "ARTIFICIAL INTELLIGENCE"

After two AI winters, artificial intelligence is now as hot as any topic can become. Over the last decade, entire research groups have been bought up by the large tech companies and the investment in all things AI has been phenomenal. The alignment of three major forces, ie the availability of data thanks to the big-data era, the GPU compute infrastructure thanks to the computer gaming industry and several breakthroughs in models and algorithms, has allowed for AI to become an increasingly critical component in many businesses.

ChatGPT, GPT4 and other large language models (LLMs) are fundamentally changing how we can interact with machines in the context of text as well as coding. Together with many others, I expect the productivity of software developers to go up 10x with the use of this technology. The same is true for other forms of generative AI, such as image generation, audio generation, video generation, and so on.

The challenge I see is that AI has led to quite a fundamental bifurcation in society. One group believes it's akin to the second coming of Christ whereas another group thinks we're on the way to Skynet and the extermination of humanity. Although there are some strong thinkers on both sides, many have very little clue of what they're talking about and use the concept completely inaccurately.

There are many ways to provide some structure to the overall notion of AI, but one way to think about it is to break it into three main approaches: classical inference, generative AI and reinforcement learning. Classical inference is concerned with training a machine learning model with a data set for classification, prediction or recommendations. From a conceptual perspective, the model is a statistical model that has been trained by the training data and validated with a second data set.

Generative AI is concerned with machine learning models that generate text, images, videos, music, product designs, logos or whatever based on a prompt. These models tend to be extremely large and trained on vast amounts of data, but at its core, a generative AI model for text is a statistical model to generate the next word in a piece of text based on what other words were there already. The results are amazing, but in my view, we're still in the "narrow AI" context rather than the artificial general intelligence (AGI) that many of the critics of AI are concerned about.

Reinforcement learning is concerned with systems experimenting with their own behavior to optimize their performance. It's concerned with a state space, an action space and a reward function that's learned

over time. This area of AI is where I'm very excited about and where one of our PhD students is actively conducting research.

One of my big frustrations has been that many software-intensive systems have been static and have failed to evolve even if data shows their performance to be subpar. Of course, with the introduction of DevOps, we've reached a state where systems are becoming better over time, but the progress is often slow and applies equally to all products in the field. I want my system to learn about my behavior and adjust itself to how I want it to perform.

My favorite example is the adaptive cruise control in my car. It doesn't work very well, in my opinion, so over the year or so that I've owned the car, I've learned how to interact with it so that the car behaves as I want it to. With reinforcement learning, we can get to a situation where the system learns over time based on the rewards you give it and you get your unique behavioral preference for your product.

My concern with the use of the term "AI" in the general public is threefold. First, many don't really know what they mean with it, how it applies and in what way it is or isn't relevant. This leads to very confused and misinformed discussions in society and the media.

Second, the only response governments, especially the EU, have seems to be regulation. Rather than using technology, the lawmakers seem akin to people with hammers: all they look for is nails. For instance, we can already use AI to determine if text was generated by an LLM or a human.

In addition, we're looking to institute laws for characteristics that we as humans are not at all agreeing on. It's easy to use terms like bias or fairness, but if we're honest, it must be clear to everyone that there's no consensus at all on what they mean in specific contexts. That leads to high degrees of uncertainty in companies looking to employ AI, which then causes many leaders to err on the side of caution. Especially in Europe, this results in adopting new technologies much later than other parts of the world, causing us to lose our competitive edge in yet another area.

Third, in the financial industry, one of the sayings is that the most dangerous words in investing are "this time it's different." That's how bubbles are started and how many people lose fabulous amounts of money. AI is treated in this way whereas, in my view, it simply is another major technology shift that humanity has experienced before. Starting with the adoption of agriculture 12,000 years ago, followed by the industrial revolution in the 18th and 19th centuries, the adoption of computers in the 20th century and now the emergence of powerful AI solutions in the 21st century, humankind has always lived in times of change. Every one of these revolutions brings major shifts to society, but I think we can agree that as a species, humans have benefited tremendously from each of these and I have no reason to believe that this time it will be any different.

Finally, many are looking to predict what might go wrong with AI and then try to preempt the negative effects by regulation, protests, public opinion pieces and the like. The problem is that humans are terrible at prediction. One of my favorite examples is the horse manure crisis in New York in the late 1800s. As the city was growing rapidly, also the number of horses increased and the amount of manure produced by them was becoming a problem. There was a prediction that in fifty years, New York would be covered with more than two meters of horse shit based on the data. Of course, the automobile was introduced and the problem disappeared.

Humans are great at solving problems once these have occurred but terrible at predicting them from the outset. The world isn't linear but rather a complex system where the effect of actions is very hard to foretell but often easy, or at least much easier, to address once challenges materialize. So, let's stop trying to regulate and strangle innovation in AI and instead monitor whether things that we fear actually materialize and then, if they do, use technology instead of regulation to address them. As Rodney Brooks said: "AI is a tool, not a threat."

THERE'S NO SUCH THING AS "THE ECOSYSTEM"

Similar to living creatures in a biological ecosystem, companies live in a business ecosystem. According to James Moore, a business ecosystem is an economic community supported by a foundation of interacting organizations and individuals – the organisms of the business world.

A business ecosystem is characterized by three aspects. First, there's a symbiotic relationship between the organizations and individuals. This means that everyone in the ecosystem is better off inside than outside. It's the responsibility of the keystone players to make sure of this as companies and individuals will leave otherwise.

The second characteristic is co-evolution. All parties need to continuously evolve to stay relevant for the ecosystem's end customers. However, this evolution needs to be coordinated to ensure compatibility between all involved. Of course, we can define decoupling interfaces that allow some parties to evolve independently of others, but that tends to only allow for some decoupling in time. Eventually, evolution needs to occur on both sides of such an interface.

Third, business ecosystems tend to be organized around some kind of platform. This platform may consist of tools, services and technology used in the ecosystem to enhance performance. It's the platform that allows the business ecosystem to create an "unfair advantage" that results in superior performance inside the ecosystem as compared to outside it.

One often discussed ecosystem is the Apple iOS ecosystem with all the app providers monetizing their apps there. Since end customers with iPhones spend significantly more money on apps as compared to competing ecosystems such as Android, there's a clear benefit for app developers to be in the Apple ecosystem. As iOS evolves and new phone types become available, app developers need to continuously evolve their apps to stay compatible. And, of course, iOS is literally the platform that allows for the unfair advantage with millions upon millions of people using iPhones.

Although a clean, easy-to-understand narrative, it's also misleading in that it's way too simplistic. Each player in the iOS ecosystem, including Apple, is not only part of that ecosystem but also of several other ecosystems. Each of the apps provides functionality that relates to another ecosystem. For

instance, your banking app connects to your bank and all the investment, payment and saving solutions offered by it. Your Linkedin app connects you with all the players in the Linkedin ecosystem ranging from recruiters to sellers and peers. Your parking app relates to the physical space in your immediate surroundings and the places you can park and pay for parking your vehicle.

Those ecosystems are, in turn, connected to several other ecosystems. For instance, your banking app connects also to the financial investing ecosystem where numerous players are running index funds, ETFs, stock market exchanges, and so on. And, of course, the financial investing ecosystem is, in turn, connected to real-world companies.

Many people I talk to tend to oversimplify the notion of their business ecosystem and treat it as an independent entity that can be understood and reasoned about in isolation. This leads to several ecosystem traps that many fall into, including the "descriptive versus prescriptive" trap, the "assumptions" trap and the "doing it all at once" trap.

The first trap is to assume the ecosystem is cast in stone and immutable. When you assume the interfaces between the partners in the ecosystem are static and a given, there's no need to prepare for changing interfaces nor do you even try to change your position in the ecosystem. Instead, an ecosystem represents an equilibrium of forces and when there's a shift in those forces or you can achieve one, the interfaces in the ecosystem will shift too.

The "assumptions" trap is where keystone players, in particular, make overly simplistic assumptions about partners in the ecosystem. A typical example is where a partner is viewed as a single entity instead of realizing that sales, operations, R&D and other roles in the partner organization may all have different drivers and needs. Ignoring this may easily cause suboptimal relationships with these partners and unexpected failure of initiatives as some key stakeholders on the partner side effectively block progress.

Finally, the "doing it all at once" trap is concerned with the deadlock that many fall into when they try to make changes to their business ecosystem. As everything is connected to everything else, it easily results in a perception that everything has to change at the same time. As that's impossible, the consequence is a deadlock and a belief that things are immutable, resulting in the first trap.

For businesses looking to become platform companies with a two-sided or multi-sided ecosystem on top of their offering, the challenge becomes even more interesting as both sides of the ecosystem need to be built at the same time. In general, research shows that it's much easier to build up one side of the ecosystem first and then start adding other sides.

No company operates in a single business ecosystem or software ecosystem; every company operates in many overlapping, interconnected ecosystems. Focusing too much on "the ecosystem" is a gross simplification that can easily cause you to lose sight of the true complexity. There are several traps that companies may fall into when dealing with their business ecosystems. Instead, aim to understand the true complexity, focus on small-scale experiments to evaluate various strategies and take small steps to continuously improve your position in the ecosystem. To quote Pearl Zhu: "A business ecosystem is just like the natural ecosystem; first, needs to be understood, then, needs to be well planned, and also needs to be thoughtfully renewed."

THERE'S NO SUCH THING AS "THE CUSTOMER"

If there's one person I'd love to meet, it's this "customer" virtually all the companies I work with refer to. In many businesses, the customer is this mythical individual who has all the answers, who is the only reason why we can't or must do certain things and whom everyone must appease for the company to be successful.

It's obvious that for any company to exist, there need to be customers to pay for the work. Also, most research suggests that customer-centric companies are more successful. The challenge in most cases is that no individual can be considered "the customer." In B2C companies, the number of customers is typically quite significant and tends to be in the thousands or millions. Talking about "the customer" is quite irrelevant as there are so many of them.

In such cases, companies need to resort to other techniques to determine the functionality they should prioritize. Some use personas in an attempt to put a face on "the customer." Others use customer segmentation and use these customer segments for prioritizing functionality that best serves the most valuable segments. Often, various forms of statistical analysis are employed to make sense of the oceans of data most B2C companies have available to determine what "the customer wants."

In B2B companies, the typical number of customer businesses tends to be much lower. For a company building airplanes, it's a few hundred. The same is true for telecom operators and many other industries. The challenge for B2B companies is that their customer businesses tend to have many employees and functions. Several of these functions interact with the company and within these functions, there typically are multiple individuals interacting with the company.

For B2B companies, it's tempting to treat "the customer" as a single entity with a single voice, but this is far from the truth in virtually all cases. Instead, we need to understand the drivers and key priorities for each of these functions, the internal politics and the strategic priorities that the C-suite has

established. In most companies, for a deal to go through, it requires all the key functions and key decision-makers to approve or at least not veto the decision. A single no or veto can break the deal.

This is why so many companies seek to establish strategic partnerships, invest in collaborative innovation efforts, wine and dine key decision-makers and use many other tactics to overcome the inertia caused by those in the company not in favor of the value proposition made. A typical tactic is to build close relationships with a key champion in the department that will apply the solution offered by the company and use this individual to overcome the hurdles the procurement department tends to put in place. Having someone on the inside negotiating on your behalf is an incredibly powerful asset.

In my discussions with companies, I use, among others, three strategies to address the challenges when talking about "the customer": clarification, experimentation and confrontation.

First, whenever someone claims the customer demands something, I start to ask clarifying questions about the specifics. These include questions about the role of the person making the claim, the context in which it was made, what the individual's decision authority is, how this might play into the politics at the customer company, to what extent socially desirable behavior might have been a factor, and so on. This allows me, the others in the group as well as the individual making the statement to reflect on the accuracy and validity of the statement.

Of course, as I'm typically in the role of an outsider, either as a consultant or a researcher, it's easier for me to challenge someone in the company than for an employee, especially when the individual making the claim has a senior role. However, in my experience, most company cultures allow for genuine, authentic clarification questions by anyone.

Second, whenever there's an opportunity, I look for ways to translate claims and statements by individuals in the workshop or meeting that I'm part of into testable hypotheses. Once I've succeeded in doing so, the next step is to formulate one or more experiments to quantitatively test the hypotheses. The challenge is that what customers say and what they actually do in practice can be, and often is, quite different. As we should base our decisions on what they do rather than what they say they do, this allows us to overcome an important hurdle in improving the effectiveness of our R&D.

In practice, experiments that answer hypotheses in a complete, statistically validated fashion can be quite expensive. Alternatively, we can run a series of experiments where we increase the validity and trustworthiness of the hypothesis with increasingly expensive experiments. That allows us to disprove the least successful ideas early and spend little resources on them.

Third, especially in companies that consider themselves particularly customer focused, there's very little in terms of ideas and hypotheses of what customers might want that doesn't originate from a customer asking for it. The challenge is that it's our job to identify new customer needs before the customers are even able to verbalize them. It's between the need appearing and customers being able to verbalize them that companies get disrupted. By the time an incumbent realizes what customers really want, a new entrant may already have built up a lead that's no longer feasible to catch up with.

To quote Henry T. Ford, if I'd asked my customers what they wanted, they'd have asked for a faster horse. Rather than asking what customers want, focus on what you believe will add the most value. This requires an opinionated product management team that follows the principle of "strong opinions, loosely held" to create a creative tension in the team that allows for ideation and hypothesis development based on the company's domain knowledge.

In this context, it's important to note that in many companies, "the customer" is used as an excuse to avoid the creative conflict that's necessary for companies to identify novel and innovative concepts to better serve customers. It feels uncomfortable and difficult to express that you disagree with your colleagues and believe that there are specific ideas that need to be tested and experimented with. This

is where leadership needs to be proactive and create a culture where teams can have constructive disagreements and maximize their learning together.

In virtually all companies I work with, there's a frequent reference to "the customer," a mythical being that needs to be appeased at all cost to ensure the continued success of the company. No matter whether we're in B2C or B2B, there's no "customer" as a single, identifiable being. To address this, three tactics are helpful: clarification, experimentation and confrontation. Although we're not the customer, it's our job to build empathy and understanding to the point that we can identify the needs of customers before they can formulate these themselves. Or, as Steve Jobs said, "Get closer than ever to your customer. So close, in fact, that you tell them what they need before they realize it themselves."

THERE'S NO SUCH THING AS "THE PRODUCT"

Of the words that get misused or oversimplified, "product" is one of my favorites. It creates such an easy image in people's heads that often is completely inaccurate, especially in the high-tech domain.

When we hear the word "product," we tend to think of this widget that passes hands from one of our salespeople to the customer. We get paid for it and move on with our lives. The product is simple, the sale is transactional and the relationship with the customer ceases to exist once the transaction has been completed.

In reality, this is of course completely inaccurate for, especially, software-intensive products. There are at least three reasons why the notion of "the product" is difficult to define in this context: boundaries, integrations and DevOps.

First, even if it may seem easy to define what the product is and what it isn't, in practice, there tend to be multiple elements that are part of the overall product. Typically, there's of course the physical product, consisting of mechanical parts, electronics and software, potentially including machine learning components. In addition, however, there often is some kind of cloud solution to complement the physical product. This cloud solution often provides a set of additional capabilities, including configuration, data-driven insights, predictive maintenance and recommendations. Finally, there might be a mobile app providing an additional interface to the physical system. All these parts, when combined, make up the offering to the customer. Many forget about everything except the physical product when referring to "the product."

Second, even in the cases where it's easy to put boundaries around the product by the organization providing it, customers use the product in a context together with other systems from other providers and/or internally developed solutions. This causes the product to be integrated with, potentially a plethora of, systems on the customer's end. In those cases, even if we seek to keep boundaries clear,

the customer will force us to lean over those boundaries and take responsibility for the integration. And, of course, it becomes significantly less clear what the product actually is as we're likely getting paid for taking that responsibility.

Third, virtually all companies I work with are adopting DevOps in some way, shape or form. DevOps results in a continuous relationship with the customer, rather than a transactional one. As you push new software, data collection code and ML/DL models to systems in the field on a continuous basis, you also need to ensure that the product continues to function as intended. Not just in isolation in the test lab, but also in the specific configuration and integration context at the customer. In many companies, this has led to the business model changing from transactional to continuous as the latter aligns better with the product's cost structure.

As a lack of clarity around the product, in terms of boundaries, integrations and continuous improvement, is counterproductive, we need to address this. The most helpful mechanism I know of is Clayton Christensen's model of the jobs to be done. In this model, a product is 'hired' to do a job that the customer wants to get done.

The "jobs to be done" model reinforces a point that I think is critical to remember: no customer gives a flying hoot about your product. Except for a very small number of luxury products that are solely used for social signaling, customers acquire a product to accomplish a certain outcome. They don't care about the product itself. In fact, many would prefer to not get the product if there was another way to achieve the same outcome.

As an example, I own a car, but I thoroughly dislike it. It takes up a lot of space, services are expensive, it costs quite a bit to drive, parking in the city is a nightmare, and so on. I only own it because I have a job that needs to get done, I have mobility needs and the car proves to be the best way to meet my needs.

For mature, stable markets, products have clear, well-defined interfaces and most of what I shared so far doesn't really apply. For technologically advanced products, however, the interfaces are much less clear because the 'industry-dominant design' isn't developed to the same extent or it's changing at a much higher rate than for commodity products. Consequently, deeply understanding the continuously evolving "job to be done" by the product is critically important.

In the case of radical innovation where the company is seeking to introduce a fundamentally new product, the customer is often not even clear on the job to be done and what product to hire for it. So, in that situation, closely working with the customer to achieve clarity is most likely the most effective use of our time.

Although many refer to "the product," in practice, it often is unclear what the boundaries of the product are, how it's integrated with other systems at customers and how it evolves. To address this, companies should focus on the job to be done for which the product is 'hired' by customers. Starting from that understanding can significantly improve the way companies work with their products and it leads to much more clarity as to what the product actually constitutes. As Jay Abraham says, "Sell the benefit, not your company or the product. People buy results, not features."

THERE'S NO SUCH THING AS "THE BUSINESS MODEL"

Although not everyone likes to talk about making money or generating revenue for the company, the fact is that every organization needs funds to survive, grow and develop. For this, we need customers who actually pay us and that's where the business model comes in.

Before digitalization became such an important development, many companies used a transactional business model where the customer paid for the widget that the company sold, at which point ownership was transferred from the company to the customer. With digitalization, however, the relationship between the customer and the company often becomes continuous, as a consequence of which it's less obvious how to monetize the offering.

In my experience, there are at least three challenges or complications many are unaware of when assuming a simple (or simplistic) business model: the monetization model, the elements included in the model and the flow of funds through the business ecosystem.

First, for the monetization model, especially with continuous models, there tend to be three main alternatives: subscription, usage and performance. The challenge is that even if you sign a contract with a large company, the number of seats in the subscription or the amount of usage may be quite limited, making it difficult to generate sufficient revenue to cover the expenses of creating and evolving the solution in the first place.

Second, especially for companies that used to sell predominantly mechanical products, it's surprising to see how digitalization adds components to the offering. In addition to the continuous updating of the software in the product itself, there's often a complementing cloud solution and mobile application that are part of the entire customer experience as well. Depending on the setup, the business model is a bundled one where everything is included in one fee, a freemium one where some features in the cloud and on the mobile app (and sometimes even in the product itself) need to be paid for separately or an unbundled offering where each part needs to be paid for separately.

Especially when it's feasible to collect data from products in the field, many companies also offer their customers data-driven services that are separate from the product itself. The use cases can vary from predictive maintenance to fleet management to comparative services where customers can compare their performance KPIs to those of others. As these services aren't narrowly connected to the product itself, they're often easier to monetize. In fact, some companies are using the revenue from data-driven

services to subsidize the price of the physical product and seek to achieve a virtuous cycle where more products get sold due to the more attractive price, generating more data to be monetized, allowing for lowering the cost on the physical products, and so on.

Third, even when the company is generating significant revenue from its offerings, the margins might be quite low or even negative. One reason might be that customers have demanded service-level agreements where significant clawbacks can occur due to underperformance of the offering. Also, partners and suppliers in the business ecosystem (or value network) may demand a significant part of the generated revenue.

A typical pattern is a setup where we need significant scale to become profitable as the fixed costs are considerable and need to be amortized over a large volume. Especially in companies that are used to scaling their traditional business in terms of manufacturing and procurement capacity, there's often a tendency to scale well before product-market fit has been established for the new digital offerings. And, of course, the nature of digital technologies is that these tend to scale at very low cost, but the initial R&D investment can be quite significant.

I have an engineering background and haven't bothered much with sales for most of my professional life, but I have an increasing respect for those who work with the issues around the business model and sales. Although not an expert, I feel three areas are important to discuss.

First, deciding on the business model is a hard and involved process that can typically not be done in an ivory tower inside the company. Especially when adding new digital elements to the offering to customers, experimentation with different models is often required to figure out what generates sufficient revenue at relevant margins. Frequently, this is concerned with the perception of customers of what you're trying to get them to pay for. As an example, in most cases, charging a fee for services that are very close to the product itself isn't accepted as customers feel this should be for free since they already paid for the product.

Second, there often is a need to balance short-term revenue versus long-term benefits. The term "lifetime value" of a customer is often used in this context. Finding the right balance between charging in the short term and not losing the customer the moment a cheaper alternative presents itself is incredibly difficult and often requires experimentation. Many companies I work with have learned the hard way that it's much more expensive to close a new customer than to keep an existing one.

Third, especially in continuous models with significant digital components, there are network effects that can be exploited. This is often referred to as "growth hacking" where the team seeks to balance incentives for different stakeholders in the business ecosystem to reach the "ignition point" where the scaling of the business fuels itself, rather than requiring continuous investment and effort from the company. In my experience, everyone loves to have a network business, but getting there is incredibly difficult and the path is littered with companies that didn't succeed with this.

Even if business models were easy at some point in the past when we were selling widgets and had transactional sales, they're most certainly no longer simple in a digitalizing world. It's easy to miss that the monetization model can include different mechanisms that have major implications on the revenue generated. There are also elements that are included in the business model that didn't exist before, like cloud, mobile and data-driven services. And the flow of funds through the business ecosystem may result in low margins despite significant revenues. Instead, focus on experimentation when creating new business models, optimize the lifetime value of customers with careful balancing and seek out and exploit network effects when these present themselves. As Alexander Osterwalder said, "Once you understand business models, you can then start prototyping business models just like you prototype products."

THERE'S NO SUCH THING AS "THE SUPPLIER"

No company is an island. Instead, we're part of a value network or business ecosystem where we co-exist with customers, partners, suppliers, competitors and so on. The ecosystem is continuously evolving and changing and occasionally, there are large disruptions.

Many feel that suppliers are the ecosystem's most stable and trustworthy members. They have a vested interest in building trust relationships with their customers, i.e. us, as it offers a moat against the continuous price pressure and risk of replacement. However, that doesn't mean that we can trust them to never go against our business interests if it would be better for their business.

The typical pattern between companies and their suppliers follows the three-layer product model discussed in earlier posts. Whenever functionality that's differentiating for us is commoditizing, it becomes relevant to explore whether a supplier could take over this functionality and own its maintenance and evolution.

The complaint many have is that suppliers will immediately seek to sell the same functionality to our competitors. It's a natural tendency for suppliers to amortize their R&D investments over as many customers as possible. Of course, we could force them to only offer the functionality to us, but then they become simply an external R&D unit and there's no or at least limited benefit from a cost perspective.

Obviously, the notion of a supplier where we buy widgets and pay for these without much consideration for anything besides the price is simplistic in the high-tech industry. The offerings there tend to have a high degree of complexity and the components and subsystems that come from suppliers need to be integrated into the overall system. This integration can be quite deep and require a similarly deep relationship with the supplier, turning it from transactional to continuous.

When working with suppliers, there are at least three main areas to consider: DevOps, being held hostage and competition from suppliers. First, with the adoption of digitalization, many companies

adopt DevOps. The software in the offering is released frequently, both by internal developers and suppliers. This means that the relationship with not only the customer but also the supplier becomes a continuous one, making it much more difficult to keep the supplier at arm's length and maintain a cost-centric approach. Typically, the switching cost when changing suppliers goes up significantly and traditional contracts need to be replaced with agile contracts where the company and the supplier can easily make changes without significant overhead.

Second, especially as the integration with suppliers, partners and customers becomes much deeper due to DevOps, the company can easily become a hostage of its business ecosystem. Even if it's obvious that significant change is required, the suppliers, partners and others in the value network loudly communicate that anyone who upsets the applecart will be ostracized and kicked out of the network. As companies need time to transition from the old to the new business model, few can afford to sacrifice their current revenue streams for a future-oriented value stream that often is much smaller and quite uncertain initially. Hence, companies risk getting stuck in an increasingly outdated ecosystem structure until external players disrupt the ecosystem as a whole.

Finally, suppliers are always looking to work their way upmarket, which means taking over functionality that initially was provided by our company. If we're not careful and incorporate sufficient new and differentiating functionality, the layer of functionality we're providing on top of what's coming from suppliers gets exceedingly thin. When this happens, the suppliers can decide to displace us altogether and work with our customers directly. For example, companies exploring autonomous driving offerings were quite easily able to get all the major parts of a car sourced from traditional tier-1 suppliers without an OEM in the middle.

Suppliers have no interest in protecting our business interests but rather are responsible for their own revenue. They'll work with us as long as their interests align with ours. And, to be honest, I've seen many situations where suppliers were treated rather roughly by their customers and I can't really blame them for not offering the loyalty they're asked for. If we're not developing and defending our niche, others will ensure we have no place in the ecosystem.

It's easy to talk about a supplier that simply delivers parts to us when we need them at a price we agreed upon. In practice, suppliers in the high-tech industry are also continuously evolving their offerings and easily become innovation partners rather than simple suppliers. There are three main areas to focus on, including the impact of DevOps on the relationship with suppliers, avoiding being held hostage by the ecosystem and avoiding being displaced by our suppliers in the ecosystem. These areas apply to everyone as, to quote William Edwards Deming, everyone is a customer for somebody or a supplier to somebody.

THERE'S NO SUCH THING AS "THE COMPANY"

During my time in industry, I learned that every company has two cohorts of employees: the transients and the lifers. The transients would change company every couple of years whereas the lifers were on track to spend their entire professional life at one company. These lifers may have tried some other companies early in their careers but had found something that made the company and themselves click, or they lacked the initiative to keep looking. Both groups, however, often referred to "the company" as a unique, identifiable entity.

We see the same in brand marketing where commercials seek to create an image in people's minds of a company with a set of norms and values as well as a culture that's positive, principled and standing for something. When reflecting on this, it's surprising how effective this is. Think of the last time you thought of buying something. Whether it's running shoes, a car, a new kitchen or even something as simple as underwear. I bet you thought about the brand, what it communicated to the community around you and your perception of the quality of the product.

Most products, however, have very comparable quality and there's little difference in brand and off-brand products. In groceries, this is particularly prevalent. Most food factories produce the same stuff continuously and simply put different packaging around it based on the price point that the product should sell at. And don't even get me started on advertising for dog and cat food!

The notion of a company is an illusion. It may exist from a legal perspective, but in practice, a company is a, potentially large, group of people with their own agendas, priorities, beliefs and ambitions. Large organizations tend to be conglomerates of smaller communities that compete with each other, where outside parties are involved in ways that can go quite deep and where ownership relationships of legal entities can be extremely complex and messy.

There are multiple reasons why considering a company as a single entity can easily lead to problems, but here I'll discuss three primary traps that I've experienced and seen people fall into. First, there's the

assumption that your colleagues in the company have the same goal. As companies typically have a defined strategy, it's easy to assume that everyone has the same understanding of what that strategy means and they'll do their utmost to realize it. In practice, a strategy can typically be interpreted and implemented in a variety of ways, which can lead to conflicts in the organization as different groups disagree on the realization.

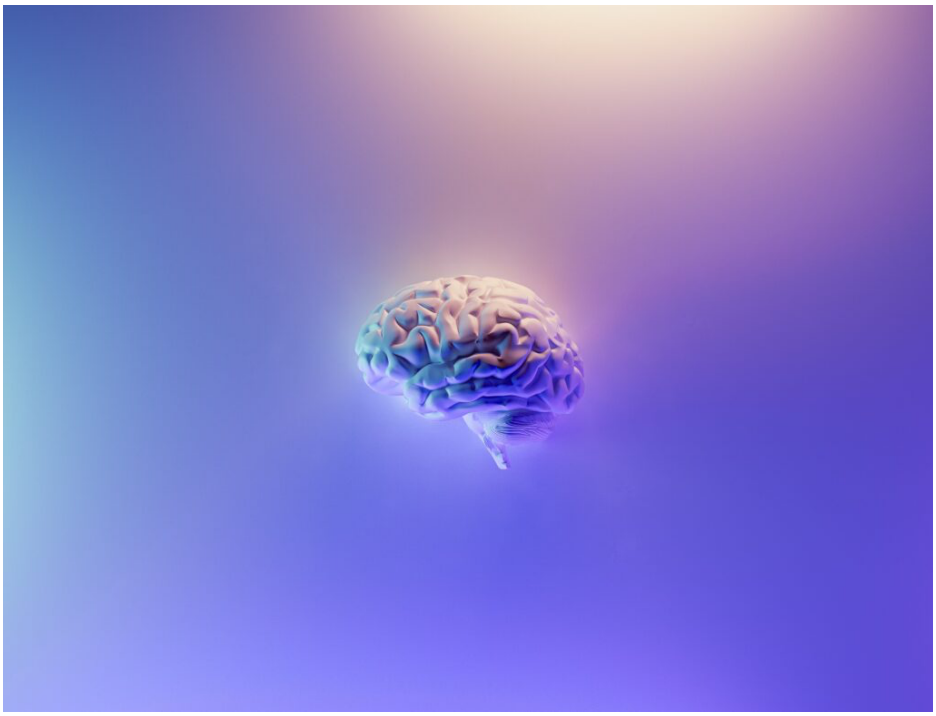
In addition, even though there may be a company strategy, different factions in the company may well disagree with it and decide to coast (not do anything to realize it) or even actively resist it. Often there will be verbal support of the proclaimed strategy, but no action. As humans are storytelling machines, there typically are convincing stories about why the team isn't acting on the strategy right now.

Second, for people outside the company, it's easy to treat every representative of the company as speaking with one voice. In practice, of course, every individual and function in the company has their own drivers and priorities. Startups often fall into this trap and suffer from many false positives from large companies that never go anywhere. They assume that the word of one individual is a clear indication of intent and they fail to take socially desirable behavior into account or do not notice that the person in question has no decision-making authority. Experienced folks in sales, typically working at larger companies, often first decide who the decision-makers are and then 'work the reporting chain' to get to a positive outcome.

Third, especially in global companies, there tends to exist quite strong competition between sites in different locations and countries. In my experience, the headquarters does what it can to maintain control of the company. However, HQ is also furthest from the markets where the revenue is generated and many sites tend to exploit information asymmetry as a mechanism to limit the ability of HQ and other sites to influence local affairs. Of course, the standard story is that kingdoms get formed unless you remove the dust from the organization through regular reorganizations, but the informal power networks tend to be quite strong.

In some of my consulting engagements, I was asked by HQ to visit local sites and perform assessments of various kinds. Every time this happened, I ran into significant resistance and power plays where locals were looking to either stonewall me or to use my presence as a way to jockey for position in the local site or toward HQ.

For all the talk about "the company," the fact is that companies consist of groups of individuals and factions that aren't (fully) aligned and all have their own drivers, priorities and ambitions. Assuming that every individual is a mouthpiece of the official company strategy and failing to understand the politics and undercurrents in an organization can easily lead to poor or subpar outcomes. The saying goes that company culture is the backbone of any successful organization, but beware that the proclaimed culture, norms, values and strategy often only have a passing relationship with reality.



THERE'S NO SUCH THING AS "INNOVATION"

Everyone loves innovation. It communicates that things will be better. That we get more of what we want. That customers will love us more. That we'll stand out from the crowd and be recognized for the great work we do. That the people we work with are motivated and excited. That everyone will beat a path to our door simply to throw money at us to get our innovation.

It's pretty obvious that this is a bit of a sarcastic way to start. Few terms in business are as overloaded as "innovation." It's thrown around everywhere and in general means "good," but it's very poorly understood. In fact, I often feel it does more harm than good in conversations and banning the term would force people to say what they actually mean.

Although there are many concerns around the use of "innovation," I want to highlight three that I think are especially important to be aware of. The first is the fundamental difference between sustaining and radical innovation. The former is concerned with adding new features to existing products, updating existing products with technology upgrades and otherwise maintaining the differentiation in already successful products. Radical innovation, on the other hand, is concerned with creating new products entirely. This is where concepts such as lean startup thinking, experimentation and fail-fast come in.

The difference between these two approaches is very significant in terms of expected return on investment. For sustaining innovation, the majority of initiatives will have a positive return and the rate of return follows a traditional Gaussian curve in terms of likelihood. The return on investment in radical innovation follows a power function. This means that the vast majority of initiatives fail to result in significant business and are shut down. However, the few that are successful generate so much new business and revenue that the investment in all the failed initiatives is paid back many times over.

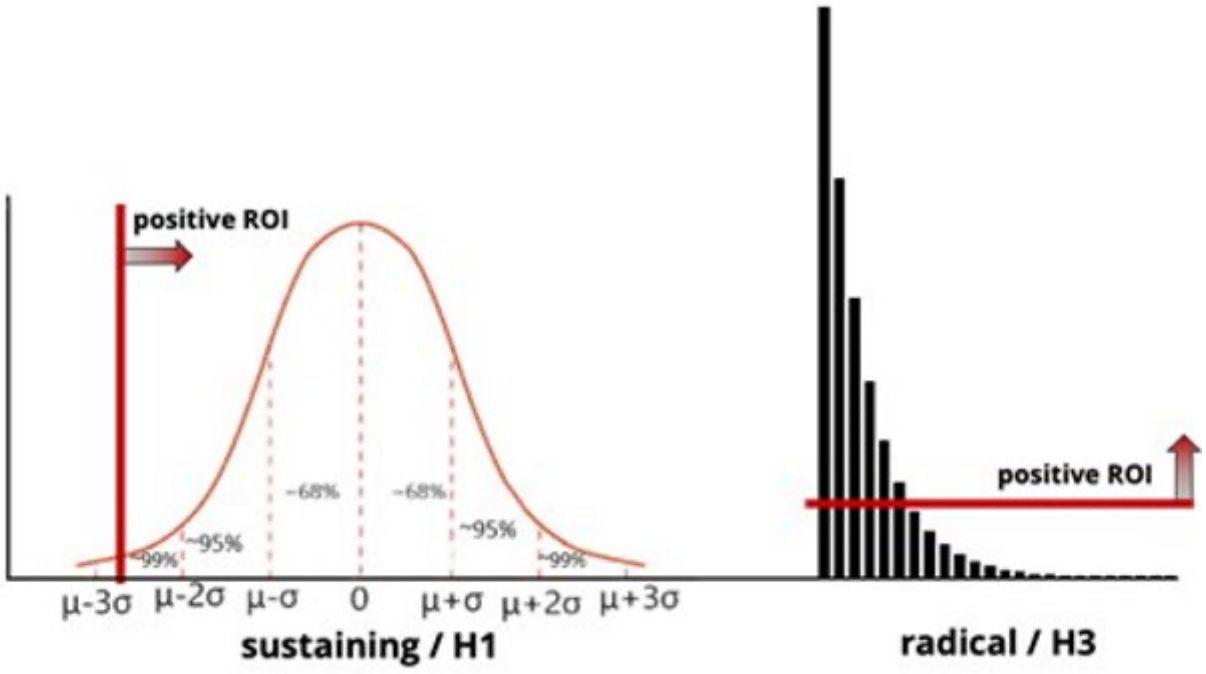


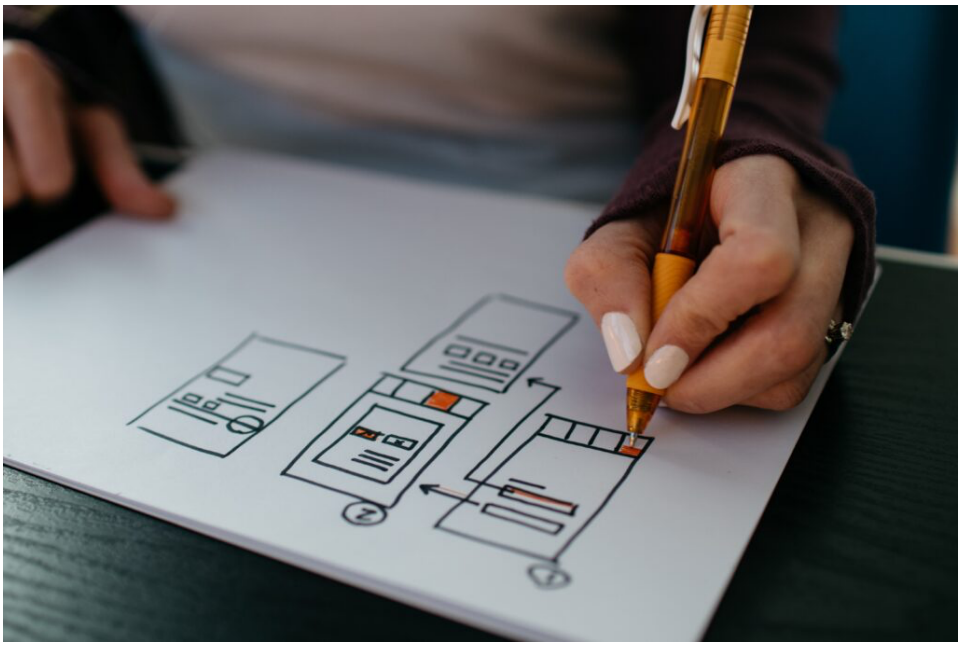
Figure: Sustaining versus radical innovation

The problem is that most people approach radical innovation with the sustaining innovation mindset and consequently punish the radical innovation initiatives by shutting things down after the first few initiatives that don't pan out.

The second challenge is that most view innovation as improving the product's capability. There are several approaches to this and in earlier posts, I've brought up the ten types of innovation model by Doblin (now part of Deloitte). It's important to understand that the financial models around offerings, the processes, the channel, the business ecosystem, the service and all other product aspects are subject to innovation too and tend to have much higher returns on investment. In fact, the research shows that investment in improving product performance has by far the lowest return whereas investment in the business model and ecosystem tends to have an outsized return.

The third challenge is that many companies fail to build effective innovation systems in the organization. As I discussed here, companies need a system around innovation that includes time for employees, virtual and physical meeting places for people to exchange ideas (like hackathons), active support from senior managers (to counterbalance the resistance from the middle management layer), methods to gather customer feedback (including feedback on prototypes and MVPs) and a mechanism to get the most promising initiatives properly funded and part of the overall company resource allocation process. Many companies have one or two elements in place but fail to build flywheels that really establish a repeatable innovation machine.

The word "innovation" is thrown around all the time in industry and its meaning has basically been diluted to "good." Unfortunately, many have a very poor understanding of the actual implications of innovation and most focus on sustaining innovations that are intended to improve the performance of already successful products. In addition, many treat innovation as either being highly predictable or completely random. Instead, our experience shows that companies need to identify that there are several types of innovation, that there's a fundamental difference between sustaining and radical innovations and that a successful innovation culture requires an elaborate system of multiple elements to be built up. It's not an accident but rather a repeatable innovation engine. As Simon Sinek writes, "Innovation isn't born from the dream, innovation is born from the struggle."



THERE'S NO SUCH THING AS "CUSTOMER VALUE"

As alluded to earlier, I work with quite a few companies in a consulting, advisory, board, research collaboration or other capacity. In many of these engagements, the discussion turns to what to build or how to extend the product with new functionality. When asked why we're prioritizing certain types of functionality, the typical answer is that the prioritized work will maximize customer value.

The notion of customer value is very hard to disagree with. Of course, we all want to deliver value to customers and improve their lives, their business or any other aspect that our offering is concerned with. The problem is that the term "customer value" mostly is a 'feel good' concept. When we start to drill into it and try to define what constitutes customer value, the glossiness rapidly crumbles. In practice, there's a severe lack of agreement in companies and teams as to what constitutes customer value. And even if there's a qualitative agreement in terms of storytelling, when seeking to quantify this, things rapidly fall apart.

For the last decade or so, I have, together with some colleagues, conducted research on value. My conclusion is that in the majority of companies, there's no agreement at all as to what constitutes customer value. This leads to significant inefficiencies for at least three reasons.

First, when we lack a proper understanding of what adds value to customers, it's very easy for the organization to prioritize development work and other activities, based on the beliefs of product managers in the company as well as others, that turn out to not add value. We have quite a bit of evidence from our research that half or more of the features in a typical system are never or hardly ever used and shouldn't have been built in the first place. This is an enormous problem as it indicates that half of all R&D effort is wasted.

Second, a lack of clear understanding of what constitutes customer value quickly results in teams building different narratives as to what adds value. This can easily lead to conflicting work across teams. For example, the team working on improving user experience or performance may have its contributions undone by the security team adding functionality that improves security but deteriorates user experience or performance.

Third, even if the organization has quantitatively defined what KPIs are to be tracked and optimized for, development initiatives will frequently improve some of these KPIs and deteriorate others. Often, a clear prioritization is lacking as to the relative importance of different KPIs, resulting in deadlocks or different prioritizations in different teams.

Our research shows that there are at least three strategies that can be used to overcome these challenges. First, apply value modeling. This is the process where the team identifies the measurable factors constituting the value delivered to the customer and then defines the relative priority of each of these factors. Although this seems obvious and easy, it proves to be very difficult for most teams as it forces an agreement on priorities and relevant KPIs.

One of the key challenges during value modeling is that groups tend to include too many factors because they want to be inclusive to the various participants. This leads to the "vital few versus worthwhile many" trap. When facilitating value modeling workshops, we work hard on reducing the number of factors to optimize for to the smallest possible number, preferably one. This is sometimes referred to as the Northstar Metric.

Second, we need to adopt short development cycles and experimental approaches. Each decision to develop some functionality or take some action is based on a hypothesis as to what the impact of the initiative will be. However, as we all know, hypotheses have a significant likelihood of being incorrect. Consequently, the less we invest until we can collect evidence concerning the hypothesis, the more effective we'll be.

Third, collect quantitative evidence to correlate feature and system KPIs to higher-level system and business KPIs. It's often hard to translate the value of adding or improving a feature in terms of the top-level impact on the company. In the age of DevOps, however, this can be accomplished by continuously collecting lower-level and higher-level data over multiple releases and conducting multivariate analyses. When doing so, we can make more evidence-based statements about the impact of individual features and teams.

The notion of customer value is a feel-good term that in most companies doesn't mean much beyond the opinion of the individual using it. It's easy to get everyone to adhere to customer value as a goal but all have their own interpretation of what this means. This leads to significant inefficiencies in organizations, including wasted development efforts, conflicting work conducted by teams and poor prioritization of work. To remedy this, we encourage companies to adopt value modeling, short development cycles and experimental approaches and to track both lower and higher-level KPIs to establish quantitative relationships between these KPIs that can be used for prioritizing development work. As Jeff Bezos so beautifully said, "We see our customers as invited guests to a party, and we're the hosts. It's our job every day to make every important aspect of the customer experience a little bit better."

THERE'S NO SUCH THING AS "THE PLATFORM"

Everyone loves to own a platform and use it to their advantage, either internally or externally. The notion of a platform communicates a perspective where there's a fundament that we and others can build on top of. Without having to do all the hard, boring work ourselves. Instead, we can focus on that which is differentiating.

As many have experienced in their professional lives, the platform notion is severely overused in most companies. In software alone, there are at least three different interpretations: the commonality platform, the superset platform and the ecosystem platform.

The commonality platform relates to the basic notion of software product lines. The idea is that it captures the common functionality between the products in a product portfolio. Each product can then use the platform and build the product-specific functionality on top of that.

Of course, in most product portfolios, there are features and functions that are present in some but not all products. When these are incorporated into the platform, it requires the introduction of variation points that allow specific features to be activated or deactivated depending on the product using the platform. Many traditional platforms suffer from a very large number of variation points that can easily become unwieldy and hard to manage.

As the name indicates, the superset platform includes the superset of all functionality by all products in the portfolio. Each product simply becomes a configuration of the functionality present in the platform. Similar to the commonality platform, the superset platform contains variation points to allow for the configuration of each product.

With the growing adoption of DevOps, the superset platform is increasingly becoming the optimal approach for companies as, compared to alternatives, it provides the most efficient way of creating each product-specific software configuration. Of course, this requires configuration tools to automate the derivation of products but also automated tests to ensure that each product is working as it should, also after extensions to the superset platform.

With the enormous success of companies like Apple with the iOS ecosystem and Google with Android, everyone and their brother are now looking for ways to have others build solutions on top of their platform. In practice, the ecosystem platform is by far the hardest to accomplish as it requires us to create a two or multi-sided market.

In creating these multi-sided markets, there's the notion of the ignition point where the number of players on each side is sufficient to generate enough activity to keep the market going without active involvement of the platform provider. Reaching the ignition point, however, is a hard slog for many companies that easily consumes large amounts of resources and funds.

To avoid wasting vast amounts of resources on a failed attempt to build this market, my recommendation to the companies I work with is to focus on one stakeholder, typically the customer, and do everything we can to maximize their number. When that's successful, rather than us having to recruit complementors, they'll start to knock on our door asking to be let in. At that point, the time has come to put some attention on that stakeholder group as well.

In addition to the above interpretations of the platform in a software engineering context, the term is also used in other contexts. For instance, companies may use customer support platforms where customers can answer each other's questions rather than relying solely on customer service staff. In marketing, many companies seek to establish a platform where prospective customers meet existing customers around specific problems and challenges that our offerings provide an answer for. And the list goes on.

As with the earlier posts in this series, my point isn't to claim that platforms don't exist but rather to encourage you, dear reader, to ensure that everyone in the conversation has the same understanding of the concept to avoid misconceptions, inefficiencies and tensions. Each type of platform has its own characteristics, in terms of investment needs, risks, challenges and mitigation strategies.

For companies working with commonality platforms, the highest-priority activities are the following three. First, you should start moving to a superset platform in preparation for adopting DevOps. Even if your customers are currently not interested in getting your software more frequently and claim that they're happy with your yearly release, this will change sooner rather than later and it will take quite a bit of time to transition the platform from a technical perspective.

In addition, many companies that use commonality platforms also support various customer branches. The cost and inefficiency of maintaining all these branches for both products and specific customers are vastly underestimated. It's only once you've brought everything into one superset platform that you realize how much waste was created by the old approach.

Second, surprisingly many companies still rely to a significant extent on manual testing for quality assurance. This may have been acceptable when software releases were infrequent, but with DevOps becoming an industry best practice, it's critical to reduce and remove the dependency on manual testing and build up the infrastructure for automated testing. The ambition has to be to always be at production quality.

Third, when the company and the product portfolio are successful, there will be external parties interested in adding specific extensions to some of your products. This development has numerous implications that go beyond the scope of this post, but when it happens, you need to have an agreed-upon strategy for opening up. One strategy is, of course, to decline opening up, but you'll be surprised how creative external developers can be. I know of cases where third-party developers inserted themselves between the software and the operating system layer to intercept calls and add functionality at that point.

As the last post in the series, the notion of a platform also is a poorly understood concept that has many associated interpretations. Consequently, any discussion on platforms easily leads to confusion, frustration and tension unless we're exceedingly clear on the definition we're using. Due to digitalization and the increasing use of DevOps, we need to change the way we work with platforms. This includes transitioning to a superset platform approach, establishing an automated test infrastructure that covers all products in the portfolio and defining a clear strategy for when others come to us, asking for us to open up our platform for their purposes. And, to end with a quote from Mitchell Baker, "The web as a platform is the most powerful platform we've ever seen."