The Digital Transformation

A Holistic Perspective For Business Leaders

With the increasing prevalence of software, data and AI, the key elements of digitalization, companies are reaching an inflection point. Although the need for change in business model, management approach and technology strategy has been building slowly for decades, we now are reaching a point where transformation will be sudden and disruptive.

One of the more profound indicators of increasing industry disruption is the increasingly rapid turnover in the Fortune 500 list of companies. According to research, the average tenure of companies on the list is now down to 10 years from 35 years in the 1990s. This illustrates the difficulty that many companies experience when confronted with the digital transformation.

So, why is it so complex to address the digital transformation? One of the reasons is that it initially tends to hide as a new technology. Whether it is software, data or AI, from a general management perspective, it looks like yet another technology that R&D has to contest with but that does not affect the rest of the organization in any meaningful way.

Of course, today we know that the assumption that technology can be incorporated in isolation is entirely incorrect. Virtually all innovation starts with the creation and broad availability of a new technology. Over time, this new technology then changes business models as well as business strategy, the architecture of products, systems and services, ways of working, processes and tools as well as the organizational structure of the company.

According to researchers studying the diffusion and adoption of new technologies, the introduction of the electric motor in manufacturing and product initially replaced the central steam engine with an equivalent electric motor that generated all energy needed in factory in a central location and then the energy was distributed using wheels, chains and bands. It took several decades for factory owners to realize that electric engines can be built in all kinds of sizes and that a much smarter factory design is to have small(er) electric engines for each machine in the factory. Then, rather than complicated distribution mechanisms, only the electric power needs to be distributed.

The point is not that digital technologies are so disruptive in nature. These technologies are, but there have been several disruptive technologies in the past that affected industry and society. The key challenge today is that society as well as industry as a whole is able to absorb new, disruptive technologies at a significantly accelerated rate and traditional, typically hierarchically organized companies, are experiencing significant challenges in absorbing these changes proactively and before they are disrupted by competitors.

The second force is that because of a variety of technologies that have surfaced over the last decades, such as the internet, cloud computing, Moore's law and others, it is faster, cheaper and easier than ever to start a new company. Even if most of these startups fail to survive or at least thrive and break through, the sheer number of these companies causes a situation where virtually every industry is under attack by new entrants that are able to exploit the opportunities offered by digital technologies much faster than the incumbents.

Thus, as a business leader, your challenge is to identify new opportunities, experiment with these to learn whether the opportunities are real and to decide on timing for adoption of these experiments. Once you decide that the time for company wide adoption has come, the real leadership challenge starts in transitioning the entire organization, your customers and your ecosystem.

Driving the digital transformation is especially challenging as it brings with it the rapid sequence of three technologies that build on each other, i.e. software, data and artificial intelligence. Software has been around for more than 50 years, but it is only during the last decade that the embedded systems industry is truly affected by the emergence of software. Initially software was treated as a technology supporting the mechanics and electronics and its lifecycle was managed in the same way as the atom-based technologies. During recent years, the adoption of continuous deployment has caused a separation of the mechanics and electronics lifecycle in products and the software lifecycle. We can see this all around us, from mobile phones receiving updates frequently to modern cars receiving new software during visits to the service station.

The expectation is that every product worth more than a few dollars or euros will have a microprocessor, sensors, connectivity and the ability to replace software frequently after the product leaves the factory. The immediate consequence of this is that every product will and has started to generate data about its performance, user behaviour, errors and defects, the context in which it operates as well as many other data points that the product has access to. This has lead to the age of "Big Data" where human data analysts, using a variety of statistical tools, derive insights from the data, instantiate dashboards for management and R&D teams and continuously search for deviations, segmentation and opportunities for personalization in the data.

The availability of large data sets, combined with major improvements in certain types of microprocessors, especially graphical processing units (GPUs), has lead to the society wide adoption of machine learning and deep learning as techniques to automate tasks earlier thought to be uniquely human. These tasks include image recognition and real-time speech translation, but also many, many other application areas that may seem less obvious such as house price prediction, process control in factories and recognition of fraud in financial transactions.

So, we see that digitalization is a complex transformation where multiple waves of technology build on each other to create opportunities that could never even have existed mere years ago.

The first companies to identify and capitalize on these opportunities are the ones that will win this battle. Although the first mover advantage has not always materialized, in this case incumbents are, by and large, disrupted because the organization was unable to muster the momentum to overcome the resistance against change.

Looking at the history of large and successful companies such as Kodak and Nokia, it is important to remember that these companies were completely aware of the new technologies and what the capabilities of these new technologies were. In fact, Kodak is credited with creating the first digital camera in its research lab. The challenge was not that these companies were unaware of the digital technologies. The challenge was that these companies were unable to instigate the change and overcome the resistance and inertia that are common in large groups of humans. As Morpheus tells Neo in the movie The Matrix: There is a difference between knowing the path and walking the path.

The Challenge of Digital Transformation

The digital transformation is one of those Copernicum revolutions where in some ways nothing changes and at the same time everything does. Especially for embedded systems companies, the company still develops similar physical products so in many ways nothing has changed. We have the same R&D departments. We still need factories to produce our products. We still need sales, marketing and finance. Etcetera.

At the same time, everything changes as the value that the company delivers to its customers often is now realized through continuous models where software and data allow the company to monetize its contributions through services and continuous value delivery rather than through the transactional model of selling the customer a new physical product every couple of years.

The continuous model of value delivery creates the opportunity of continuous improvement. Rather than designing and building a product that then slowly deteriorates over time, we can now offer our customers solutions where the relevant KPIs improve all the time. A situation where life gets better all the time rather than once every couple of years when I upgrade to the next product.

The notion of continuous improvement requires continuous data flows, careful analysis and optimization of processes through the use of this data. The role of artificial intelligence techniques, such as machine learning and the subfield deep learning, is precisely here: using continuous data streams to train models that, when put in operation, can solve problems that were unsolvable before or can do a significantly better job than traditional approaches.

In the table below, we summarize the main differences between a traditional business and a digitalized one. The key difference is that because of fast feedback loops and availability of data, a digitalized business can operate in fundamentally different ways.

	Traditional	Digitalized
Business	Transactional model where customers buy products periodically (typically every couple of years)	Continuous value delivery model based on services; monetization through KPIs and expectation of continuous improvement
Ecosystem	One dimensional value network from suppliers to product company to customer.	Multi-dimensional business network with multiple avenues for monetization using products, data and other assets.
Architecture	Deeply integrated architecture optimized for minimal bill of material cost. Focus is on freezing the architecture after design and a “big bang” release.	Modularized architecture separating parts that evolve at different frequencies through APIs (mechanics, electronics, software). Focus is on facilitating continuous evolution and release.
Process	Process dictated by mechanical design and manufacturing constraints. Focus on planning and prediction in order to minimize cost due to late changes and quality issues.	Process focused on fast feedback loops facilitated by continuous deployment and data streams. Focus on experimentation and continuous learning.
Organization	Hierarchical organization with functionally organized departments. HIPPO-based escalation paths to iron out conflicts due to local optimization.	Empowered, cross functional teams responsible for different aspects of value delivery. Cross-team coordination through architecture evolution.
Culture	Atoms-over-bits mindset; tendency for local over global optimization. “I’m responsible for doing my job well; nothing else”	Bits-over-atoms mindset. Deliver on the company mission and take on responsibilities based on what is needed rather than job description.

Table 1. Comparing traditional and digitalized organizations

The table indicates the key challenge that companies undergoing a digital transformation have to contend with: the change literally affects every aspect of the company. Every function, every department and every individual is affected by the shift from a traditional to a digitalized business. Not only that: it also requires that every part of the organization changes with a certain or significant level of alignment as as the intended changes will be impossible to realize. A typical example in this context is that the R&D organization prepares a service offering to complement the product offerings that the sales organization is completely unprepared for and incapable of selling.

In earlier books, I have outlined how these changes should be sequenced and which steps should be taken first and which ones later. See for instance, the books on speed, data and ecosystems [Bosch 17a], on using data to build better products [Bosch 17b] and impactful software [Bosch 18].

This book is concerned with providing a variety of perspectives with the intent of providing business leaders a holistic perspective on digitalization and its implications on the company, its people and its ecosystem. Traditional ways of thinking assume that specialization is the key skill to accomplish, but when it comes to leadership, one has to have a holistic understanding in order to avoid local optimization and bad decision making processes.

Themes

As a leader for a company looking to successfully navigate the digital transformation, a heavy responsibility rests on your shoulders. The challenge often involves breaking down structures that have proven to be successful in the past and that are still providing revenue and profit today, just to replace these with novel, unproven ways of conducting business often based on little more than early experiments.

Although Facebook's Mark Zuckerberg is often quoted with his belief that the biggest risk any business has is to not take any risk at all, in practice it is incredibly difficult to jump for the safe, secure and proven to the novel, risky and unproven. And yet, not embracing digitalization is a recipe for disaster when the business ecosystem that you're part of suddenly tips.

It is important to realize that all changes start slowly and then reach a tipping point where the pace of the change accelerates exponentially. Similar to how Hemmingway responded when asked how he went bankrupt: first slowly, then suddenly, companies are disrupted in a similar way. Often, the changes are small blips on the radar for years before turning into an unstoppable tsunami of change. When you wait for the tsunami to appear, you have set yourself up for disruption.

Although digitalization is not necessarily unique in the nature of the change, it is the main challenge that virtually every industry faces these days. It is difficult to contend with for at least

three reasons. First, digitalization is a multi-faceted problem that has many dimensions. In the table in the previous section, we already identified business strategy, ecosystem, architectural, process, organizational and cultural changes that are required to successfully transition. Second, the digital transformation has a fractal-like quality in that it affects individuals, teams, departments, companies and entire business ecosystems. At each level, there are similar, though not identical, challenges that require attention in order to successfully transition. Finally, third, the digital transformation requires a broad skill set and understanding from leaders. It is not sufficient to be an expert in your own area, but rather you need a sufficient understanding of every field of relevance in your organization as well as the creativity, curiosity and innovative mindset to imagine the opportunities that adoption of digital business models, technologies, organizational setup and culture can create and facilitate.

One of the well-known parables is the story of the four blind men and the elephant. As each man touches a different part of the elephant, each develops a completely different understanding of the object that they're touching. The digital transformation is analogous to this in that it is basically impossible to imagine the ecosystem and your company after the digital transformation at a level of detail that includes all relevant dimensions. Instead, we have to study the challenge from many different angles and, from that, generalize and create a holistic understanding of the problem.

This basic conception defines the structure of the book. Rather than providing a structured, step-by-step set of recipes forming a cookbook, the book presents reflections organized in 10 themes. As such, it aspires to satisfy the MECE principle (Mutually Exclusive; Collectively Exhaustive), but due to the nature of the problem of course fails to include all relevant aspects. It, however, covers the key aspects of a digital transformation and, more generically, provides an illustrative instance of any larger technology shift with cross-cutting implications for the organization.

Below, each theme is briefly introduced and I explain why this theme is important for successfully traversing the digital transformation.

Business strategy

For any company that aims to be in charge of its own destiny, the starting point needs to be its business strategy. The strategy defines where you would like to go and how you think you can get there. Strategy is a notoriously large topic and it is both blamed and unjustly held responsible for things that don't go well in companies. However, the fact is that unless you know where you want to go, it's really hard to get there and this is where business strategy comes in.

The short chapters under this theme cover topics ranging from culture to innovation and from agility to organizational deadlock, but the main tenor is concerned with a transition from an old, traditional business operating system to a digitalized business operating system.

Leadership

For all the talk about empowerment and flat organizations, individuals and teams look for leaders to set direction. The main difference between traditional managers and modern leaders is that managers tell people what to do and then use coercion to make them comply. Leaders create a positive environment where people want to follow the direction pointed out by the leader because it aligns with their beliefs, norms and values. When organizations are becoming increasingly flat and driven by empowered, self-managed teams, leadership is more important than ever. It is about engaging with teams to set jointly agreed and committed direction and then getting out of their way as the team executes.

Under this theme, the chapters discuss the distinction between (micro-)management and leadership, questions traditional coordination mechanisms such as meetings, the risks of falling into the trap of busyness and firefighting and, of course, the nature of leadership.

Innovation

Although strategy and leadership have been buzzwords for decades, during recent years few have reached the buzzword status as innovation has. Innovation is viewed as the silver bullet in many companies and there are more innovation café's, garages and what not going on around the industry than one can shake a stick at. In many ways, innovation is a poorly understood word that often gets mixed up with invention (novel, but without monetization) as well as mixed up between sustaining innovations (novelty in existing solutions and for existing customers) and disruptive innovations (new solutions for existing and/or new customers). Although there are many ways to skin this cat, at least these two types of innovation require fundamentally different approaches in companies.

For all the confusion around the notion of innovation, any company that does not want to sink into the morass of commoditization needs to continuously innovate in order to maintain relevant differentiation in the market. Under this theme, we explore the confusion around the term, relate innovation to agility and explore the notion of failure in the context of innovation.

Personal Development

For leaders and individual contributors alike, your ability to deliver value in any organization is founded in the mindset and resulting behaviors that you bring to work. As none of us is perfect, we all have responsibility for developing ourselves continuously with the intent of self actualization. As Ghandi said: "Your beliefs become your thoughts, Your thoughts become your words, Your words become your actions, Your actions become your habits, Your habits become your values, Your values become your destiny."

Consequently, you have the responsibility to work on the core of what defines you as a person, meaning your deepest set of beliefs. Personal development is concerned with identifying the limiting and incorrect beliefs that you harbor as a person and then working on changing and improving these with the intent of allowing yourself to grow as an individual and, ultimately, as a formal or informal leader.

R&D Strategy

Whereas business strategy is concerned with positioning the company in its business ecosystem with the intent of maximizing revenue, margin and growth, R&D strategy is concerned with defining the best approach to allocate and organize the R&D resources of the company with the intent of optimally supporting the business strategy.

The uncomfortable fact of R&D strategy is that, in most companies, it actually comes before business strategy and sets business strategy. The R&D strategy often is more long-term than the business strategy, resulting in a situation where choices made in R&D in previous years define what is possible or impossible in the business strategy today.

Systems Engineering

As digitalization is concerned with software, data and artificial intelligence, it’s easy to forget that all this software and data needs to run in a physical context. Although pure software systems run on general purpose hardware and don’t need to worry much about the physical infrastructure, the more interesting systems are embedded systems where software interacts with electronics and mechanics. From automotive to medical imaging products, there is a need for systems engineering that combines the various technologies in an integrated package that provides a reliable solution.

Traditionally, systems engineering focused on the mechanics and electronics, but with the digital transformation taking place, we need to shift the focus to software and data and ensure that that the mechanical and electronics parts optimally support a digitalized system. One of the key challenges is to support continuous deployment of software without jeopardizing the functionality, safety and security of the system.

Speed

All analysis shows that in many ways society is speeding up. One of the measures, mass market adoption of new technologies, clearly shows that the time from the introduction of a technology until it reaching 50 million users in the US, is shortening for every new technology introduced. Although it is easy to think of a train going faster and faster on a straight track, the real challenge is business agility. I define business agility as the ability of a company to respond to changes in the market place rapidly and faster than the competition.

In this part of the book, the main focus is on feedback cycles and accelerating these cycles. Fast feedback cycles allow companies to shift from opinion based decision making to data-driven decision making which is an enormous benefit as it allows for a significant increase of the effectiveness of the company.

Data and Artificial Intelligence

Digitalization is concerned with software, data and artificial intelligence (AI). Although traditionally the focus has been on software, increasingly the latter topics are becoming more and more important. It is data that allows us to generate insights that are far from obvious to the human mind and to detect patterns in large amounts of data that are elusive otherwise. AI, and especially machine learning and deep learning, provide technology to identify these patterns in data and then act on these patterns to deliver solutions that were unfeasible before or that function much better with the use of AI.

Ecosystems

No company is an island and we all operate in a business ecosystem. Although much research exists that focuses on inside the walls of the organization, research focussing on the business ecosystem is much less prevalent. With the digital transformation affecting virtually every company, though, it will affect the existing ecosystems as well as introduce new stakeholders and avenues for value creation. A better understanding of ecosystems is required to take intentional and proactive steps to optimally position yourself in a situation where the business ecosystem is shifting rapidly and often unpredictably.

Organization and Change Management

Although all the topics discussed so far are concerned with change, actually realizing this change is challenging as, in the end, individuals, teams and organizations need to start to do things differently from how they worked in the past. In parallel, the way we organize people is shifting from more traditional hierarchical organizations to flatter organizations in which individuals and teams are more empowered to operate autonomously. These two developments are seemingly at odds with each other as a central change agent has less steering ability with empowered teams than with hierarchically organized organizations.

In this part of the book we explore this area and discuss how to effectively drive change even when the way we organize changes. In addition, we discuss the importance and relevance of the move towards empowerment and why this improves the competitiveness of organizations.

Who Should Read This Book

This book is organized in ten themes that are largely, though not completely, orthogonal. Each theme covers a series of shorter, one to three page, chapters that provide a reflection on an aspect associated with the theme. The intended audience consists of three groups, i.e. those interested in digitalization in general, non-technical formal and informal leaders involved in digitalization efforts in their organization and, finally, formal and informal technical leaders.

For those interested in digitalization in general the first theme, business strategy, is likely of primary interest as it covers the key drivers of the changes that digitalization causes in organizations and ecosystems.

For non-technical leaders, most themes in the book will be of relevance, except for the more technical themes. These technical themes include systems engineering, speed and data & AI. The other themes should both be of interest and sufficiently understandable also for those without a technical background.

Finally, technical leaders should benefit from reading the material in all themes as their responsibility concerns the entire chain of understanding the business strategy implications of digitalization to the technical and organizational implications.

About The Structure Of The Book

Each chapter under each theme is intended as an independent piece and is not necessarily related to any other chapter. Also there is no specific sequence of chapters, although the themes are intended to build on each other in terms of content. The chapters started their lives as part of a weekly blog post series. Each chapter is based on the original post, but is frequently edited, sometimes significantly.

And with all of that out of the way, let’s get right into it!

Business strategy

1. Digitalization: 5 Insights
2. Why Digitalization Will Kill Your Company Too
3. On Organizational Deadlock: Why We Need A New Business Operating System
4. The End of Order
5. Why R&D Sets Business Strategy
6. The End of Control
7. It's Not Technology That Is Holding Us Back
8. The End Of Product Companies
9. Where Is This IoT World That I was Promised?
10. Structure Eats Strategy
11. Why Your Strategy Is A Paper Tiger
12. Business Agility: What Got Us Here …
13. Nobody Cares About Your Product
14. Projects Suck; Do Products Instead!
15. Enough Efficiency Already! Focus on Effectiveness!
16. Are You Ready To Burn The Ships?
17. Your Product Is No Longer The Product

Leadership

1. On Hierarchy, Micro-Management and Leadership in R&D
2. Leading through Disruption (or not)
3. Leading in the Digital Age
4. Meetings Are A Waste Of Time (And What To Do About It)
5. Why Firefighting Ruins Your Company
6. Activity Is Not The Same As Progress
7. Why Trust Is The Foundation Of Success
8. Don't get stuck in your company's echo chamber

Innovation

1. Innovation Is Hard Work
2. The End of Innovation (as we know it)
3. When Did You Last Talk To A Customer?
4. Why Failure Is The Only Option
5. Does Agile Kill Innovation?

Personal Development

1. Why You Will Ignore Opportunity Cost in 2018 Too

2. Why Your Job Will Also Suck in 2018
3. To All You Unsung Heroes
4. Stop Thinking It's Not Your Problem
5. Do You Have Skin In The Game?
6. Why Do You Do What You Do?

R&D strategy

1. Effective R&D in Complex Systems
2. The End of Requirements
3. 9 Out Of 10 in R&D Work On Commodity
4. Half The Features You Build Are Waste
5. On the Role of Software Architecture
6. How To Double Your R&D Effectiveness - Part I
7. The End of Planning
8. The End Of Product Teams
9. On Functional Safety in the Age of Continuous Deployment
10. Why Bad Business Habits Kill R&D Effectiveness
11. Why Variability Management Still Is An Unsolved Problem
12. Stop Wasting Resources And Do Platforms Already!
13. Stop Customizing Your System. Configure It Instead!

Systems Engineering

1. Six Practices Transforming Systems Engineering
2. The End Of System Architects
3. Systems Engineering in a Service-Driven World
4. The End Of Architecting Systems

Speed

1. Five Reasons Why Speed Matters …
2. Speed versus Quality
3. Towards Testing After Deployment!
4. Why Fast Feedback Cycles Matter
5. How Agile Are You Really: Let's Find Out!
6. Stop Complaining About Agile

Data & AI

1. How To Stop Building Useless Features
2. Minimize Investment Between Validation Points
3. Towards Data-Driven Decision Making
4. On Big Data, Profiling, Fake News and Political Interference

5. Towards Dataism
6. Five Reasons Why You Are Not Data-Driven
7. What Are You Optimizing For?
8. Become Data-Driven In Five Steps
9. Will Computers Program Themselves?
10. Engineering Deep Learning Systems is Hard!

Ecosystems

1. Orchestrate Your Ecosystem (Or Be Ruled By It)
2. The Five Traps of Software Ecosystems
3. Why Your Customers Are Slowing You Down
4. Why One Customer Is No Longer Enough
5. Are You a Product, Platform or Ecosystem?
6. Don’t Be Left Behind By Your Business Ecosystem
7. Your Ecosystem Has 50 Shades Of Gray

Organization & Change Management

1. Why Cross-Functional Teams Should Be The Norm
2. The End of Reorgs
3. On The Scope Of Feature teams
4. Too Many Caesars; Too Few Romans
5. Autonomy, Empowerment, Alignment and Coercion
6. Overcoming Inertia in Organizational Change
7. Stop Trying to Change Your Company!
8. Why Organizing Into Functions Kills Change

I. Business Strategy

Business is moving towards a new operating system. Although it can easily be viewed as hyperbole, the key elements of the traditional way of running a business are fundamentally shifting. As we showed in the introduction, business is shifting from selling products in a transactional business model to selling services in a continuous business model, decision making from opinion-based to data-driven decision making, the business ecosystem from one dimensional to multi-dimensional, organizational structure from hierarchical to empowered, cross-functional teams, etc.

In a world that is complex and changing quickly, one of the most important things to do is to establish a clear sense of direction. Even if the strategy is not optimal, it sure is a heck of a lot better than not having a strategy at all or continuing with the old strategy that made the company successful in the past.

It is a basic and normal human trait, when faced with a confusing context, to freeze and end up in paralysis. Analysis paralysis is a cliché because it is common in many organizations to end up in a state of paralysis because the confusion is causing everyone to have different, more or less informed, opinions on the next steps that the company should take. With many formal and informal leaders pulling in different directions, the resulting state easily becomes a deadlock where nothing changes.

Obviously, a state of paralysis is the worst possible outcome when business agility and proactively change are the key elements of success. The only way that an organization can break out of this paralysis is to set a business strategy that directs all resources in the company in one direction.

The chapters below offer several perspectives on the changes driven by the digital transformation. These perspectives focus on the things that no longer are true after the digital transformation, the things that become true after the transformation as well as aspects of the transformation itself.

Digitalization: 5 Insights

This week was all about digitalization for me. I spent most days this week at three companies in different parts of Europe and spoke at the excellent Digitalization seminar at Chalmers. Two of the companies that I worked with this week are traditional “metal bending” companies that are in the midst of the transformation towards becoming digital companies whereas the third is a younger startup working hard at riding the digitalization wave in its industry and capitalizing on it. Between the time at these companies and all the hours on the plane, I spent some time reflecting on what I had learned and I thought I’d share my insights.

#1 Digitalization is about business development and transformation: Digitalization has major implications for the R&D organization. This is especially true in companies where mechanical and electrical engineering were the central disciplines and software was only a small part up to recently. However, at most companies that I meet, the folks outside R&D often believe that digitalization is a R&D challenge and fail to see that the transformation affects *everything* in the company. This includes business models, sales and marketing, delivery and support organizations, manufacturing and supply chain management and even internal functions such as HR and finance. I still meet many senior leaders outside R&D that worry and complain about the R&D organization, but fail to see that they need to step up to the challenge as much if not more than R&D. Digitalization is about business development (exploit the opportunities) and transformation (align with the new reality).

#2 Speeding up the heartbeat of R&D is still a challenge: Most companies have traditionally had a slow release cycle of yearly releases of new products and product upgrades. Software and digitalization allow for much faster deployment cycles and several embedded systems companies have adopted continuous deployment of software where new releases are pushed out every couple of weeks. However, for the majority of companies that I work with, speeding up the heartbeat of R&D is still a major challenge. The typical arguments include (1) our customer is not asking for it (or even actively resists it), (2) we can’t guarantee quality or (3) we want to but our ecosystem partners are unable to comply. All these arguments are real, but it doesn’t absolve you from the responsibility to still implement continuous deployment!

#3 Thinking outside the product portfolio is hard: During the discussions, the focus of the companies was almost exclusively on the effects of digitalization on the current product portfolio and occasionally on new products in the same category that could benefit from the new technologies. However, the opportunities around data-driven and digital services or pure software products were hardly ever mentioned. It’s clear that thinking outside the current product portfolio is very difficult for many in industry, even in the cases where it is obvious that great promises exist.

#4 It’s about data *and* about software: Another observation this week is that most of the people that I talk to either talk about data or they talk about software, but the understanding that

it is about both is less appreciated. The idea that continuous deployment of software also allows us to constantly evolve the data streams that we collect. Or the fact that the data from deployed products allows us to significantly improve the effectiveness of our development practices. As well as that there is significant synergy between the two ideas. The understanding of the relationship between these two perspectives is limited across the industry and when it exists, the conclusions are often not internalized to the point that these materialize in the company.

#5 Irrational fears and unvalidated beliefs: Finally, many harbour irrational fears about the impact of digitalization and have beliefs about the expected impact that are not based on any serious intellectual thinking. It seems like the uncertainty associated with such a major transformation causes so much uncertainty that people jump to conclusions in an attempt to create a false sense of certainty, even if these conclusions have little to do with reality. Accepting that we don't know how things will play out while acting based on our best understanding and adjusting when new information becomes available is a skill that we all need.

My main reflection is, though, that digitalization is the latest technology that, as most of the transitions before it, will materially improve life for all of us. The risk is, though, for companies as the risk of disruption, as evidenced by the constantly decreasing tenure of companies on the Fortune 500, has never been greater. Accelerating the pace of transformation is the key insurance against disruption and the highest priority for senior leaders.

Why Digitalization Will Kill Your Company Too

The benefit of spending a lot of time on airplanes is that you get time to think. As I work with quite a few companies and the topic of digitalization comes up on a regular basis, I thought I'd reflect a bit on the role of digitalization and the disruption that it causes in industry.

The first thing to realize is that digitalization offers great opportunities, but also severe risks for companies. The Fortune 500 presents an illustrative example of this: the average tenure of a company on the Fortune 500 list is, according to recent research, now reduced to 10 years (from 95 years when the list was first assembled). One of the key reasons for companies to fall off the list is that they fail to build new core capabilities and stay in their comfort zone of technologies and capabilities that made these companies successful in the past. If we look at the technology and capability that currently is disrupting companies and industries, of course it is digitalization.

Just to be clear on definitions, digitalization is concerned with using software and data to offer significantly improved products, pure software products as well as digital and data-driven services. The intent is, obviously, to create new revenue opportunities as well as replace traditional business models with new business models.

In my experience, most of the companies that I have the opportunity to observe in different ways are fundamentally missing the boat and are at significant risk of disruption due to digitalization; hence the title of this article. Although there are, of course, several causes for this, there are at least four patterns that seem to be recurring.

First, the vast majority of companies have no or very limited software skills in their senior leadership. Many leaders have grown up inside the company through traditional technology paths, such as mechanical or electrical engineering, and are firmly stuck in an "atoms-centric" world view. Alternatively, leaders have a financial or management background without any fundamental product and technology understanding. Management by numbers is a very effective model when you're in a stable, mature and slowly evolving industry, but it is destructive during times of transformation because the need for change will not show in the numbers until it too late by far. Change is based on a core belief that the envisioned future in inevitable or at least that the current model is obsolete and needs to change. No spreadsheet will ever indicate the need for change.

Second, in some ways a derivative of the first pattern, most companies are incapable of "ambidexterity". Building the capability to deliver on today's challenges while also preparing for the future needs of the company is extremely hard to do. This requires sacrificing resources that could be used to deliver on today's revenue generators and have these people spend time on topics that might become relevant at some point in the future, but whether these benefits will come to pass if viewed as highly uncertain. Most managers prefer certain short term smaller

success over uncertain, long term success of uncertain magnitude. Although the organizations tend to talk about preparing for the future, in reality this “vision” is never translated into an actionable strategy and tends to lead to the company “dying a death of a thousand cuts”.

Third, the most surprising pattern that I see if that many leaders believe that digitalization is an R&D problem that only affects the software R&D unit. For some reason, it is more convenient to believe that today’s business models, products and business ecosystem will stay the same and just be slightly adapted because of software and data. The problem is of course that nothing could be further from the truth. Digitalization allows for business models that are fundamentally different from today and digitalization starts with understanding the business implications, both in terms of opportunities and risks. Failing to recognize this is a recipe for disaster.

Fourth, many companies justify their lack of initiative by referring to the lack of desire for change from their most valuable customers. When reflecting on this, it is obvious that your most valuable customers are no more eager to change than what you are as they are on top of their industry and digitalization is as much a risk to them as it is to your company. The danger is of course that an entire business ecosystem is replaced by a new one and that entire value networks get disrupted. Waiting for customers to change first is comfortable and allows you to deliver on yet another quarter, but obviously is mismanagement from a longevity perspective.

Concluding, my current observations of many of the companies that I spend time with have led to my prediction that most of these companies will be disrupted and killed by digitalization. If you are interested in avoiding that fate, my recommendations are fourfold:

1. Get software expertise in your senior leadership and board as soon as possible and listen. The world is changing rapidly and all your experience, collected over the last decades, is a liability rather than an asset.
2. Preparing for the future does not happen just by saying it. Put systems in place that ensure that sufficient resources are allocated into exploring opportunities created by digitalization and make sure that the success metrics are different from the mature legacy businesses.
3. Involve all functions in the company in the work around digitalization and embrace the uncertainty that comes with radical innovations.
4. Focus on your fringe customers and experiment with these customers at the edges of your business around new opportunities and innovations enabled by digitalization.

Most companies will be killed by the digitalization transformation and the odds are stacked against most companies. However, it’s still better to go down fighting than as a victim and, who knows, you might come out victorious!

On Organizational Deadlock: Why We Need A New Business Operating System

In almost every meeting that I have these days, the notion of organizational change comes up. Everyone is concerned with changing too slow and the risk of being disrupted by more agile and innovative competitors they haven't even heard of. My role gives me the opportunity to talk to people at all levels in typical organizations, ranging from front line engineers to C-suite executives. Interestingly, everyone complains about the same thing: they want to change, but they can't get the rest of the organization with them. Front line engineers are waiting for manager approval to change things and senior leaders complain that they can't get the rest of the organization to "follow" them. The result is that we end up on organizational deadlock: everyone is waiting for everyone else to make the first step.

For the last decades, organizations have focused on operational efficiency. To achieve this, the focus was on building deeply interconnected organizational architectures where every function and team is dependent on other functions and teams and delivers to others. The analogy is that of a finely tuned machine where all the cogs and gears fit as closely together as possible. This works great as long as the organization is in a predictable and stable business environment. However, two things are happening: first all work, blue-collar and white-collar, that is repeatable is being automated. For years, the focus of IT was to support people in doing their job more efficiently. Now the focus has shifted to automating the job completely. Second, companies no longer operate in stable environments, but rather in highly unpredictable environments where the risk of disruption has never been higher.

The fundamental business operating system of the last decades of building the highly integrated organizations that are highly optimized for one purpose and are not much use for anything else is what is causing the organizational deadlock that many organizations experience. Sadly, this operating mechanism not only affects industry, it as much, if not more so, affects government institutions. And as one of the people that I talked to this week reminded me of, it is the "big five" management consulting firms that share a lot of responsibility for large, traditional successful organizations to end up in this place. So, relying on them to get us out of this mess is a really bad idea - we need inspiration from elsewhere.

One "knee-jerk" reaction that several people and organizations look for is a strong leader. This strong leader is then expected to set a strategy, cut through the organizational deadlock and get everyone aligned around one direction. The logical consequence of that strategy is that this "strong leader" also gets the power and authority to decide and overrule everyone. And of course, in this approach, dissenting voices are not allowed and shut down aggressively. We see this in politics as well as in many companies, but, to paraphrase Albert Einstein, for every problem, there is a solution that is simple, elegant and wrong.

Centralizing power and relying on a strong leader is wrong at two levels. First, it puts the fate of the organization (or even a country) in the hands of a single individual and assumes that this person will be right in the direction that he or she sets. Consequently, it does not make use of the collective intelligence of the organization. Second, it's "the easy way out" for everyone else. By abdicating power to the strong leader, everyone else feels that they have no responsibility for the future success of the organization. So they can just lean back and wait for their marching orders.

In my experience, the right direction is the exact opposite of centralization. The reason for organization deadlock is that everyone is waiting for permission from elsewhere. So, the central theme of a new business operating mechanism has to be empowerment of individuals and teams. This gives not only authority but also an expectation to decide, experiment and act to teams and individuals. This does not mean that every team will be successful. In all likelihood, the majority of things that are tried will not be successful. But the fact is that rather than running one experiment, set by the "strong leader" and betting the company on its success, we're running hundreds or even thousands of small experiments, learn and find a direction forward that helps the company transition to a successful future.

We need a new business operating mechanism that is based on, at least, these three principles:

- **Empowerment**: First, everyone in the company needs to be empowered to make decisions based on their best understanding without having to ask for permission. This is a critical element as the very act of having to ask for permission already reduces or removes initiative. In addition, hierarchical organizations are notoriously slow in decision making, especially when things need to go up and down the line and we can't afford these delays.
- **Data-driven decision making**: The organization needs a much higher level of transparency and access to data should be free for everyone. Members of the organization want to make good decisions and one important ingredient is access to information.
- **Relentless innovation**: Finally, the focus of everyone should be relentless innovation in all aspects of the products, solutions and services, but also internal processes and ecosystem engagements. Of course, all these innovation efforts should be data-driven so that there is as quick feedback as possible as to what is successful. Repetitive work should be aggressively automated as humans are too expensive and quite poor at boring work.

Transitioning to this model can not be done in one step, so we developed a model to move towards this new business operating mechanism in an iterative fashion. As shown in the figure below, we view agile practices as the first step where teams get autonomy. This sets the direction for allowing for increasing empowerment in larger and larger sections of the

organization.

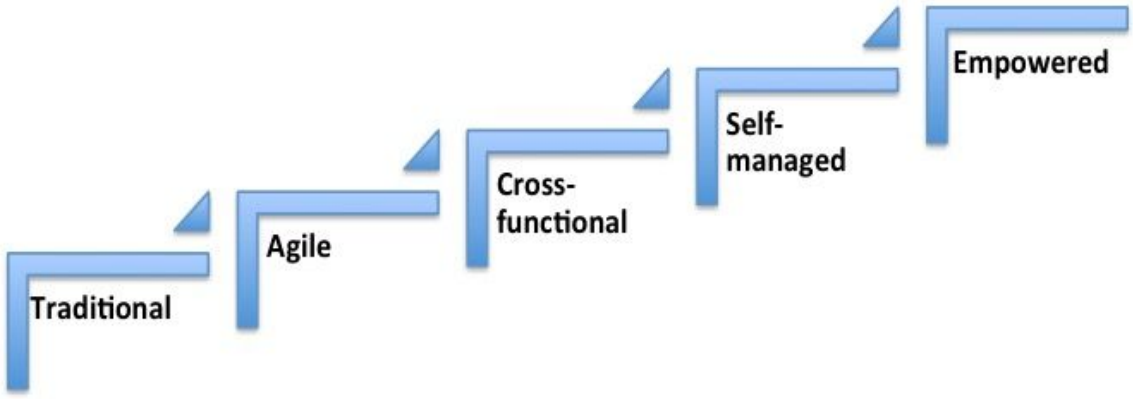


Figure: five steps to become an empowered organization

The End of Order

Everyone loves order. Order gives structure, predictability, safety and comfort. Our brains are wired to seek order, either by shaping the world around us to maximize it or by avoiding situations where we might lose it. There is nothing more comfortable than sitting smack in the middle of our comfort zone. For the entire history of mankind, our sole driver has been to create order in the environment we live in. Moving from a hunter-gatherer society to an agricultural society offers a major increase in order, structure and predictability and this has lead to huge societal benefits. The Dutch did not wait for high seas to drown their fields and villages, but instead built dikes and pumps to keep the water out.

Organizations, basically being groups of individuals, are the same: managers design systems to create order. Employees, though sometimes grumbling about it, accept and even like these systems as they offer structure and predictability. Over the last hundreds of years, organizations of all types have built processes, procedures and other mechanisms with only one purpose: order and predictability resulting in safety and comfort. For instance, when new senior leaders enter the organization, the first step often is a reorg. Not because the organization needs it, but because it isn't until the reorg is done that the new leader feels that he or she operates in an ordered environment. Similarly, for most organizations, where resources are allocated this year is based on where the revenue was generated last year. We act in this way because we seek order and our default assumption is that this year will like last year with a few small deviations.

The problem is that life and business are inherently unpredictable. We live in a world where the pace of change is ever faster. The large incumbents in numerous industries are threatened and disrupted by new entrants that operate in different ways and often can't even be considered to be part of the same industry as they redefine the borders. Is Uber just another taxi company? Is AirBnB in the hotel business? Of course not! Business models that worked for the last century or more suddenly don't work anymore as innovations allow for different monetization strategies or allow a new entrant to operate at an order of magnitude lower cost level. Customers "hire" different technologies, products and services for the jobs they're looking to accomplish. They never cared about your product; they cared about getting the job done.

All these developments have one thing in common: the existing order gets disrupted. Many complain that the changes and challenges in their industry are chaotic and that there is little one can do to defend against disruption. I do not believe that this is actually true: there is a place in between order and chaos and this is ***complexity***. In a complex world, it is not that there is no order. It's just that it is difficult to understand, often non-linear and morphing constantly. Individuals and companies alike deal with complexity by seeking to simplify to the point that they feel in control again. To structure internally to the point of comfort and predictability, even when there is no predictability out there. To create plans and then following the plan, because it creates a (false) sense of control.

To quote Mencken: for every complex problem, there is a solution that is clear, simple and wrong. Successful companies need to go against their natural instinct and turn their back on seeking to constantly create order. Instead, we need to lean into the complexity of the real environment in which we operate. We need to embrace complexity and relish in it as only those that do will be able to respond intentionally to the forces around them. How does one realize this? Based on my research, consulting assignments and work with startups, I believe that there are at least some principles to use as a basis:

- **Focus on the customer**: The first principle is obvious to the point of being painful, but the fact of the matter is that I see time and again that in most organizations, the vast majority of employees is so far removed from the customer that their understanding is little more than a hazy set of shadow beliefs. This leads to many decisions being suboptimal or plain bad.
- **Embrace the complexity of your ecosystem**: Your business ecosystem is complex and there are many players and forces to consider. Although it is easy to generalize and simplify, it is critical to embrace the complexity. Not only does it allow organizations to identify shifts and risks earlier, it also allows organizations to proactively model the desired ecosystem structure and realize strategic shifts in their ecosystem that serve their interests. In my research and consulting, we're explicitly working with this, so contact me if you want to know more.
- **Don't plan for more than 6 months**: Life is inherently unpredictable and we should constrain our desire for predictability through long-term plans. Instead, insist on continuous planning or, at most, 6 month plans. One main advantage is that all processes and mechanisms in the company that are unable to operate at this frequency become visible. This allows the company to identify where it enters the "wishful thinking" stage, rather than stay connected to reality.
- **Decentralize responsibility to cross-functional teams**: Rather than rely on centralized and hierarchical decision making, give teams the authority to make decisions. Even if those decisions may affect many in and outside the company. This requires these teams to be cross-functional to ensure that all relevant aspects are considered, but at the same time, the team has to carry the final responsibility for the decision.
- **Defer decision making to the latest possible point**: Although this may seem counterintuitive to many, especially decisions that are hard or impossible to reverse should be taken as late as possible. Especially in embedded systems, early decisions are required concerning the mechanics and electronics of the system. However, in this and other domains, it is almost always possible to take a series of decisions at different points in time, where the early decisions set foundations and later decisions add more details. The advantage is that this allows you to take decisions with as much timely information as possible.
- **Treat every decision as an experiment**: Although we tend to think about decisions as one-time commitments, the fact is that many decisions are reversible with limited consequences. In traditional organizations, this is often viewed as embarrassing for the leader that took the decision. We need to change our perspective as in complex systems, the effects of decisions often are hard to predict. Consequently, we need to aim

to treat decisions as experiments. The decision reflects the hypothesis that we aim to validate. Once the decision has been taken, we can evaluate the consequences and results. If the outcomes are not what we expected, we revert the decision and try something else.

- **Learn twice as fast as the competition**: Above all, we need to learn faster than the competition and the above mechanisms allow us to learn. This means that rather than organizational paralysis in response to the complex environment, we need action and lots of it to experiment and learn what works and what doesn't. And of course, don't hesitate to learn from the mistakes of others in your industry and elsewhere.

Concluding, mankind has imposed its order and structure on the world and this has had great societal benefit in terms of quality of life. Companies have built on that approach and constantly seek to increase structure, stability and predictability. The unfortunate consequence is that organizations get out of touch with the complex reality in which they're operating and instead live in a logical, structured and simple hallucination in which they can easily be disrupted by competitors and new entrants. Instead, using the presented principles, we can embrace the complexity of our business environment and relish in it!

Why R&D Sets Business Strategy

After running my business over the last years, I've started to see a pattern in the engagements with large companies. It often starts with senior R&D management, one or more lead architects and perhaps a couple of experienced engineers. There is some major transformation, platform effort or other topic that starts off as an R&D initiative. As our work progresses, however, it becomes increasingly clear that the project basically affects all aspects of the company, including business, processes and the organization. Everyone then gets a little fidgety and realizes that perhaps it's necessary to bring in the business side for a discussion. After some time, typically my next visit, the meeting between the team from R&D and some general managers and a sprinkling of sales, marketing and other non-R&D roles takes place and the weirdest thing happens: the business side actually thinks that they are setting the business strategy! And funnily enough even the team from R&D believes that this is the case! The result typically is a kafkaesque scene where everyone is really polite, but at the same time the business folks are expounding on their vision and strategy without much connection to the technical feasibility or even market desirability and R&D desperately tries to derive some insight into how the business side is thinking that they can bring into their work.

So, let's do away with this illusion once and for all: ***in all software-intensive companies R&D sets the real business strategy***. And as all companies are software companies these days, this means that R&D sets the business strategy everywhere. With "real" business strategy, I mean the one that gets executed and not the powerpoints. The reason is that the architects and R&D leaders take strategic decisions concerning the software and IT architecture that make certain business directions easy, fast and shoe-ins and other directions virtually impossible without major investment, re-architecting and writing off major parts of the infrastructure and existing functionality. Despite all the agile methods, continuous integration and deployment and A/B testing, the fact of the matter is that when you have a couple million lines of code of functionality, you simply can't turn on a dime. Changes in response to business strategy will take significant amounts of time. In fact, these changes will take much more time than the business side needs to make up their mind and tell everyone what they want.

The consequence is that architects and R&D leaders think long and hard about the likely business directions that the company will take in the coming years. Based on their best bets, they take architecture and R&D investment decisions to enable the most likely business directions. These decisions may happen years before the business side is ready to go there, but that's fine as it may take the R&D organization all that time to get ready. Not because they're slow and inefficient, but because there is a lot of code affected by the changes. And there is the sprint heartbeat to get new functionality out every couple of weeks, so the bandwidth is limited.

In fact, my belief is that most cases where the business side is dissatisfied or even flabbergasted at the lack of R&D capability are caused by the R&D leaders and architects having made the wrong decisions some or several years earlier. They simply misjudged the

business directions that the company ended up taking. Of course, it's easy to blame the R&D folks, but my experience is that when there is too little interaction between the business side and R&D; if the business side treats R&D as a place where you order functionality, rather than engaging in a strategic partnership, the likelihood of R&D setting the wrong direction goes up dramatically. Similarly, R&D leaders and architects that fail to spend significant time understanding what business the company is in and how this business is likely to evolve, also own responsibilities when R&D and IT go off the reservation.

The main reason for me writing my "Speed, Data and Ecosystems" book was to offer a source that both business leaders and R&D leaders can use to obtain an improved understanding of the societal and industry trends that will likely set direction for the business as well as to offer a rich set of techniques to understand, model and execute on the consequences of these trends. Similar to most of my engagements with large, successful companies, my book offers a bridge where business and R&D jointly develop a common understanding and language to collaborate on preparing all of the company for a bright future, including business and R&D strategy.

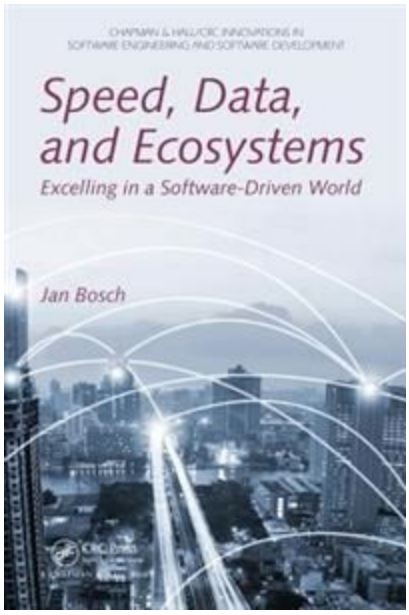


Figure: Speed, Data and Ecosystems book

Finally, since the book came out, I realized that many want more than just a book. Consequently, I am preparing for a two day course later in the spring where business and R&D leaders can participate to understand and apply the trends and techniques and to explore the consequences for their company and business. If you're interested in learning more, send me an email at jan@janbosch.com.

The End of Control

This week I spent in Buenos AIres (Argentina) at the International Conference on Software Engineering (ICSE 2017), which is the premier event in the field. As usual, I enjoyed the pre-events such as workshops, smaller co-located conferences and panels more than the main conference itself. Although many will disagree with me, I feel that the new thinking and insights are found in these smaller pre-events and that the main event is more conservative.

During the days, there was one theme that kept on reappearing (or at least it felt that way to me): with especially large companies viewing themselves as ecosystems within the walls of the organization, the broad interest in and adoption of software ecosystems and the attention to systems-of-systems, the traditional process- and architecture-centric mechanisms for exhibiting control are no longer effective. And it is unclear what replaces these mechanisms. One thing is clear though: we're moving from centralized control to decentralized alignment and mutual influencing.

When analysing alignment, we can identify four dimensions of alignment. First, there is process-driven alignment, such as traditional CMMi processes. These processes worked well inside of companies, but as companies increasingly move toward ecosystems and partnering approaches, approaches such as release trains, independent deployment of components by different partners as well as automated deployment of new versions of components by subsystem providers will increasingly become the norm, even in safety- and security-critical, regulated systems.

The second dimension of alignment is through architecture. By defining interfaces, traditional platform companies provided extension points that third parties could use for customer-specific or domain-specific extensions. Through software product lines, companies applied these principles inside the boundaries of the organization and with the emergence of software ecosystems, the same mechanisms were offered to third parties. Going forward, however, strict architecture control will not be feasible in decentralized contexts as it puts too much power in the hands of a single player. Hence, we need new mechanisms such as architecture principles and underspecified interfaces agreed among communities.

The third dimension is alignment through the business model. Traditional organizations used typical customer-supplier models where suppliers were reimbursed for delivering subsystems with specified functionality. Using this approach, large platform companies seek to develop control points in the ecosystems that allow them be gatekeepers to third party solution providers. Going forward, other business models such as revenue sharing, free use of the platform for a fee, indirect monetisation mechanisms, such as through advertising and data sharing or the complete absence of a commercial relationship seem likely models in a decentralized world.

The final dimension companies use for alignment has been the legal model through contracts. Especially large companies have used legal agreements as mechanisms to achieve control, exclusivity and an (un)balance of power which by many would be considered to be unfair. Going forward, we are likely to see different forms of SLAs with harder or softer terms in the context of service deployments, license agreements of various forms that support aforementioned business models as well as Free/Libre OSS (FLOSS) legal terms that allow the users of the software or service to use it in any way it wants (depending on the OSS license used).

In global software engineering, ecosystems and systems of systems, the alignment between partners is less and less control driven and other mechanisms for governance and alignment are required. Above, I discussed the four dimensions of alignment that companies can use to decide (1) how they would like to govern and align with partners (inside or outside the organization) and (2) what would be acceptable or not acceptable for them in terms of governance imposed by their (potential) partners.

In a world that is moving towards decentralization, finding the right mechanisms for alignment and governance are going to be key for companies to be successful going forward. Insist on centralized and absolute control and the ecosystem will bypass you. Maintaining too little leverage in the ecosystem, however, will limit your ability to monetize. As usual, the right balance will depend on the situation and likely change over time as well. A software driven world is an exciting world!

It’s Not Technology That Is Holding Us Back

During another week on the road (Zurich, Paris and Stuttgart this time) and after discussions with dozens of people at the conferences and meetings, I noticed a pattern that repeats itself again and again in technology driven companies. For the latest major digitalization technologies, such as big data, IoT and deep learning, I have observed the large technology companies respond. Interestingly, the incorporation and adoption of these technologies is treated as a technology problem. So, initiatives are kicked off to build infrastructure, build up competencies around the technology, architectures are designed, scalability is confirmed, etc.

The moment the first activities are underway, plans are requested detailing when which part of the technology stack will be available and ready for deployment, budget are estimated and, after the normal political hassles, resource allocations are approved and made available. Once this is in place, the normal project management follow-up takes place with project meetings, steering board meetings, etc. that are all tracking progress, following up on resources and finances and admiring the shiny demos of the technology that are shown by project members. And, at the end of each meeting, everyone basks in the glow of the bright future that the company can expect to have with all these exciting technologies becoming available.

In all this, there is one big glaring party missing: the customer. With all the focus on the technology, there is very little attention to the customer and the ways in which this technology is expected to add value to customers. Of course, there are a couple of dreamed up use cases, but these have never really been confirmed with customers. And as everyone keeps telling each other about these fabulous use cases that customers are just going to love, the use cases start to lead a life of their own and everyone in the company just knows that these use cases are the ones that are going to allow the company to disrupt the market.

Then, when the first early deployments with customers are getting ready to be put in operation, we notice the next stage: customers are not interested, the solution does not provide the promised benefits or the technology solves a non-existent problem. Initially, the lack of maturity of the solution is blamed and the company doubles down on its investments as the technology is viewed as strategic and critical for its future success. As the solution matures and the market adoption still is lackluster and fails to deliver on the inflated expectations, the belief in the technology in the organization starts to wane and the technology falls into the “trough of disillusionment”. The good news, however, is that by that time there will be some other cool, exciting technology on the horizon and the attention of everyone in the organization shifts to the next promising, cool thing. And before you know it, similar to a dog chasing shiny bumpers and getting distracted every time a new car with a shiny bumper appears, the organization runs off after the next thing.

The key lesson is that technology adoption is not about technology. Adopting a new technology is about innovation. It’s about finding the use cases where the new technology adds real,

tangible value for customers. It's about trying out dozens of different use cases and accepting that in the end, the majority of these will fail to deliver on the promise. And it's about realizing that even though customers *say* they love the innovation and can't wait to use it and promise to pay you good money for it, it's about what customers *do* that really matters. The gap between espoused theory and theory in use is large and this is even more true when it comes to adopting new, exciting technologies.

The problem is that in many of the technology driven organizations that I work with, everyone has an engineering background and to a large extent believes in the mantra of "build it and they will come". Consequently, the adoption and evolution of new technologies is viewed and treated as an engineering problem rather than an innovation and business development problem. Companies invest millions upon millions to build infrastructure, architectures and competencies without a single truly validated use case. We build fantastically strong foundations for businesses that may never realize.

Instead, we have to accept that it's not about technology. It's about the benefits we can offer to customers through the use of these new technologies. And it's our customers that are the judges of what is beneficial to them. So, instead of building the foundations, we should build dozens of use cases with the ricketiest, scrappiest, duct tape and bailing wire technology solutions below them. Then we verify whether customers actually do with our solutions what they said they would do. And only in the minority of cases where this turns out to be the case do we strengthen, productify and scale the technology basis below.

Concluding, adopting new technologies is not about technology, infrastructure, architectures or competencies. Every technology oriented company that I know and work with is able to solve the most complex and exciting technology problems. It is the innovation processes that are the weak spot: going out with a rickety prototype to a customer to verify that it indeed solves a real problem is something that technology companies find incredibly difficult. However, if a solution solves a real problem, customers will use it no matter how scrappy it is. And if it doesn't solve a real problem, customers will not use it no matter how glossy and shiny the solution. So, accept that it's not technology that is holding us back; it's the lack of an experimentation and innovation culture that is the real problem.

The End Of Product Companies

Most of the software-intensive systems companies that I work with are in the process of converting from pure product businesses to product + service or pure services businesses. This is a huge change for any company and means changes at several levels and all functions of the company.

Over the last months, I have been trying to understand what this means for companies and why it is happening. If we start with the implications, it is clear that the first major change is the relationship with the customer. This relationship changes from a transactional relationship, where the customer only interacts with the company when he or she needs a new instance of the product, to a continuous one where the customer and the product deployed at the customer constantly or frequently interact with the company. This is a consequence of the normal service delivery because identifying when the deployed product starts to exhibit issues and fixing it before the customer even notices a problem obviously results in a higher quality of service.

The second change is that continuous deployment of software to products out in the field now has an associated business case. Deploying new software may extend the economic life of products at customers, reducing the cost of delivering the service as the physical product needs to be replaced less frequently. Alternatively, if the service fee is dependent on performance, new software releases can improve performance and in that way increase revenue.

The third major change is financial. Product companies typically get paid at the time of product delivery and that creates a cash flow where expenses for manufacturing and to some extent R&D are aligned with the revenue generated from product sales. In a services business, however, you're asked to deliver the product in order to start the service delivery, but the recouping of those expenses occurs over a much longer period of time and fundamentally changes the cash flow. In fact, you are financing the product for your customer. Of course, with proper pricing, a services business can be very profitable, but not getting paid for the initial investment is inherent in the shift.

Fourth, everything in the company runs at a higher heartbeat and shifts from waterfall-style to continuous processes. Of course, this is the case in the service monetization where monthly service fees are typical and in software where frequent deployments are increasingly the norm, but even for the physical parts of the products we often see that companies start to provide a constant stream of improvements and cost measures. There is no real incentive anymore to create a big hoopla around a new product model or type and consequently companies are less inclined to accumulate an extensive set of improvements only available to the new product as a mechanism to drive upgrade and replacement decisions.

Of course, there are many other implications of the transition to a services company, but those go beyond the scope of this post. The next question is why companies are moving from product

to services businesses. The answer to that question lies in the ownership of the customer relationship. For decades, B2B and B2C customers have moved from CAPEX to OPEX. Owning expensive items has a high risk associated with it when you don't know how long you will need the item. In addition, quality issues become very binary. If you own one car, it is very unfortunate when it breaks and it leaves you with an unpredictable cost at unexpected times. A leasing company owns thousands of cars and they can calculate and manage their risk because they know that a certain percentage of cars will experience issues. It become a manageable cost of doing business. In addition, especially younger generations have shifted their behaviour from ownership to access.

As customers increasingly want access rather than ownership, they automatically turn to services rather than products. And once you use a service, do you really care about the product that is used to deliver a service? When ordering an uber, do you think about what car the driver will have? Or do you only care about getting from A to B quickly and efficiently? This means that the traditional differentiators of product companies, such as brand, reliability, lifetime cost, etc., are no longer relevant. As a customer, whether the service provider uses a cheap, crappy product to deliver the service or an expensive, reliable product does not really matter to me as I only pay for the service. This means that service providers have all the incentive to treat product companies as cost-driven suppliers. As a product company, the only way you can make a living in an industry moving to services is by being the cheapest. All your differentiation is irrelevant as it never makes in the RFQ.

The only way for product companies to survive in such a business context is by owning the customer relationship. And that, in turn, means not just providing the customer with a product, but to manage the entire set of tasks that the customer wants to outsource. Which then turns the product company into a services provider. And as a services provider, the company needs to orchestrate its ecosystem. If your product needs to be installed at the customer, you are now responsible for getting it installed, meaning that you need to orchestrate the selection of the installer, ensuring that the installer does a good job and financing the installation. If your product needs regular maintenance, similarly you will be on the hook for ensuring that this happens, either by your own people or by partners.

Software and connectivity are the root causes of a fundamental shift from products to services in a wide variety of industries. I am calling the end of product companies as the only way a product company can survive in an industry that has shifted to services is by being the cheapest. And that's no fun way to run a company. So, instead product companies are forced to forward integrate in their value network and take ownership of the relationship with the end-customer (B2B or B2C). And that requires them to become service companies.

Where Is This IoT World That I was Promised?

During a conference in Vienna, I listened to a keynote from a researcher from a large industrial company. The talk was full of the promise of IoT, connectivity, cloud, edge computing and apps and the business benefits that could be realized. The presenter did a good job, but I have heard these presentations on the fabulous new world of IoT many times before as well as read perhaps hundreds of articles.

The challenge is that everyone is working on the easy part: sensors, connectivity, places to store data, security, etc. It's all about technology enablers and platforms, but everyone seems to ignore the elephant in the room: every company, large or small, has solutions to provide vertical integrations where their devices, their edge computing solutions and their cloud solutions. And for all kinds of reasons, they provide little if any opportunities for others to connect to their devices, either for collecting data or for control. There is no viable and convincing approach that allows for the first steps of horizontalization of the IoT market.

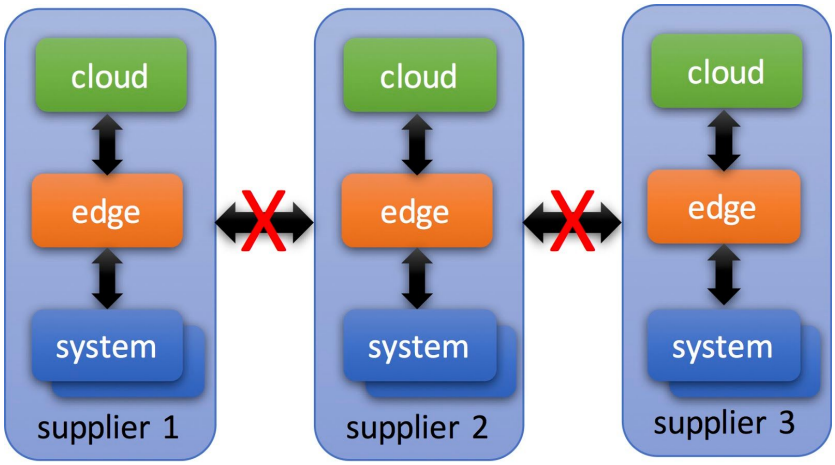


Figure: Illustrating the problem in IoT

One of the companies that I work with described the challenge as outlined in the figure above. Each supplier provides a vertical solution consisting of the systems, edge computing solutions and cloud solution but goes out of their way to avoid cross connections to other vertical solutions. Similarly, I was involved in a research project with a home security company and an energy provider. Both installed sensors and actuators in the homes of customers and neither was willing to explore the potential for accessing and controlling each other's devices.

In many ways, the IoT industry today is similar to the computer industry in the 1970s (see the image from Andy Grove below): vertically integrated businesses that provide their own stack, from top to bottom, and that force customers to buy bait, hook and sinker into the vision provided by the company offering the stack. Once you choose a certain ecosystem, you're locked in and changing to another ecosystem will prove to be an expensive endeavour.

The difference is that in the 1970s, the business benefits of automating administrative processes were obvious and the companies were less aware of the lock-in problem. In modern days, many companies that I work with are wary to become overly dependent on a single supplier as they have paid a huge price for that mistake multiple times in the past. And also these large suppliers realize the risk that they expose themselves to if they just work as product providers. So, some companies worry about their suppliers forward integrating into the value network and replacing them altogether.

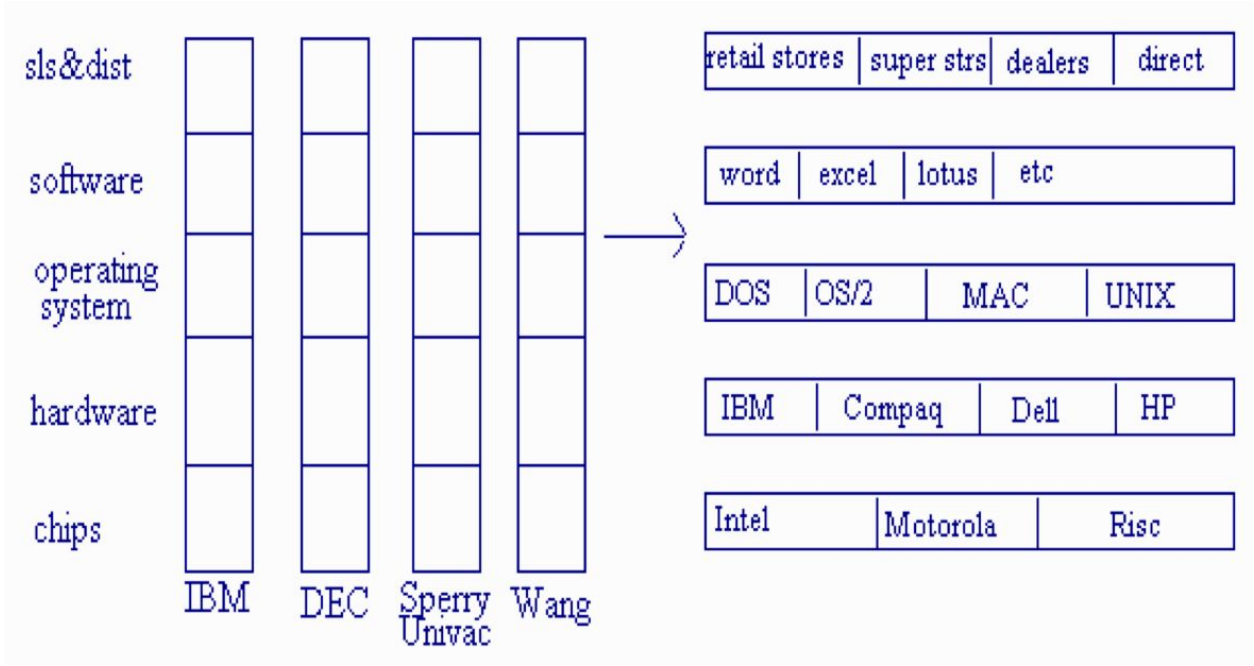


Figure: Image from Andrew S. Grove

That does not mean that there is no value being created in the IoT space. Vertical integrations allow for all kinds of benefits related to preventive maintenance, more close-loop optimization and increased agility. However, it is obvious that the true potential of IoT will not be delivered until we find an effective way to horizontalize the IoT domain, similar to the transformation of the computer industry. However, for the current industry players to go through that transition, there needs to be a fundamental disruption of business models and industry structures that the incumbents will be loathe to undertake. And as IoT typically addresses industries with high barriers to entry, because of regulation, security, safety and customer brand creation concerns, this transformation may become a slow and dragged out affair.

Now, I will not claim that I have the solution, beyond identifying the need for the IoT space to horizontalize. However, I do believe that blockchain technologies, allowing for fully distributed establishment of trust without reliance on a central trusted authority, will be a central component. In a recent paper (see reference below), we discussed the different alternative blockchain architectures and their relative benefits. The reason is that none of the large and

small incumbents will want to give up autonomy and place themselves under the control of one of their competitors. Blockchain style technologies allow ways to control secure access, safety related concerns and other aspects of interoperability across vertical IoT stacks without a single, central trusted party.

Second, the end-customers (B2C and B2B) need to push much harder for secure interoperability. Unless the suppliers are forced to open up, they have little business incentive to do so. In industries such as telecommunications, the operators partner up to dictate industry terms to their suppliers. Although other industries such as manufacturing, automotive and logistics, may not have the same structures in place, these industries will have to create solutions to accomplish the same outcome.

Concluding, IoT is delivering on some of its potential but major societal and industrial benefits are left unrealized as we are unable to break through the boundaries put up by the aforementioned vertical stacks. This really needs to be addressed by the end-customers in the various industries potentially benefiting from IoT. Preferably this happens soon as a fully connected IoT world is a better one than the one we live in today and we owe it to humanity to get us there sooner rather than later.

Reference: Xu, X., Weber, I., Staples, M., Zhu, L., Bosch, J., Bass, L., ... & Rimba, P. (2017, April). A taxonomy of blockchain-based systems for architecture design. In *Software Architecture (ICSA), 2017 IEEE International Conference on* (pp. 243-252). IEEE.

Structure Eats Strategy

Again and again, I run into situations in which change agents in organizations that I work with run into organizational boundaries that stop them from moving forward in a way that makes perfect sense from a business and innovation perspective. For various reasons, within the current organization agreements have been reached where some functions have responsibilities that others can not intrude upon without causing major hassle and disturbance in the company. The result is a situation where the current organization dictates what the company looks like going forward, no matter if this is the right setup or not.

So, even though I am risking sounding like a broken record that have me seen present or have read some of my work, I am afraid that I need to once again present the BAPO model. In my experience, BAPO offers one of the few antidotes towards the current organizational setup dictating the future of the company.

The BAPO model has a very simple premise: the starting point for any organization is the business strategy and the innovations that define the future of the company. The way we generate revenue and how we generate value for our customers needs to be the starting point for any subsequent activities or decisions. So, the BAPO model defines the “B” as the starting point.

Based on the “B”, the next step is to define the architecture, structure and technology choices. This refers to as the “A”, which is in turn provides the starting point for the process, ways of working and tools, i.e. the “P” of the model. Finally, and only based on the B, A and P, should the organization (“O”) be defined.

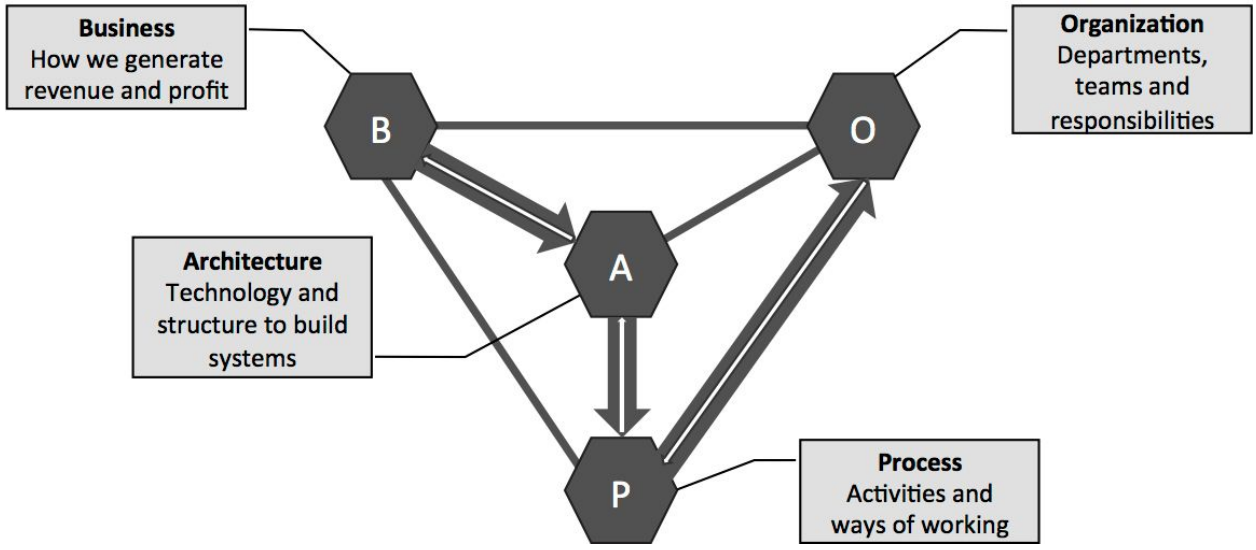


Figure: The BAPO model

The interesting situation is that most companies are not BAPO but instead they are OPAB: the existing organization is used as a basis for the definition of processes, which in turn leads to an accidental architecture. This restrictive architecture, driven by the past of the company, rather than its future, then offers a highly limited set of business strategy options.

In company after company, whenever I present the BAPO model, the first reaction is: Yes, of course! And then, after a few seconds, some or several people in the audience realize that this is exactly the opposite of how their company really works. And the realization (and associated embarrassment) that the company actually uses OPAB sets in. The result often is a lively discussion across the participants on where the company has falling into the trap in the past and where it is at risk of falling into this trap again right now.

My call to action: be vigilant against cases of OPAB in your company and bring up the BAPO model whenever the organization is at risk. Ensure to first discuss the business strategy and value to customers that we're looking to accomplish and only then allow for the discussion to turn to what this means internally in the organization. Start from the B; not the O!

Why Your Strategy Is A Paper Tiger

All companies have a strategy. Often captured in a nicely designed PowerPoint deck and summarized on a piece of plastified paper so that everyone can look at it and share how much they are in support of the business strategy. These strategies allow everyone to sing kumbaya, sitting around the campfire, even if they secretly, or not so secretly, are at war with each other.

The problem with most strategies is that they have been watered down to something that everyone in the organization can agree to and expressed at a level of abstraction that makes it trivial for everyone in the organization to concoct a story as to how they are supporting and contributing to the strategy. This frequently extends to teams that are operating completely at odds with each other.

The second problem with many corporate strategies is that they are inherently inconsistent. For instance, we want to be technology leaders and have a high degree of customer intimacy. We want to be a cost leader and cheapest in the market while being viewed as the most innovative company in our space. We guarantee our customer's their privacy and we use data we collect from our deployed products and customer behavior to generate additional revenue.

The third problem is that most strategies fail to outline the prioritization of items. The hardest part of strategy development is to decide what to NOT do. What will we stop doing so that we create opportunities for these new topics to receive the bandwidth that they need to have even a remote chance of having real impact on our bottom line in a year's time, or at least in a couple of years.

Even if the strategy makes perfect sense and is viewed as internally consistent and relevant at the headquarters, folks in the front lines, the regions, the sales organizations and the service providers that actually meet the customer and are the real face of the company fail to see how this applies to them and conveniently ignore anything specified in the strategy. And when asked, they come up with a story on urgent topics in combination with an interpretation of the strategy that suits their needs.

The only way to make progress is to ensure that the strategy is translated into real, quantitative and measurable objectives at all levels in the organization. In traditional organizations, one of the lean practices, Hoshin Kanri, is mentioned as a technique to accomplish this, but there are other approaches as well. The notion here is that the strategy is recursively broken down into quantitative targets at sufficiently short timelines that it creates an operating system where the entire organization is operating in line with a single organizational goal.

The alternative is to give up on the idea of a central, top-down strategy and to adopt a bottom-up strategy, driven by the teams across the organization. In many organizations, this is

the reality anyway and it might be quite liberating to align the organizational strategy processes with the reality of the company.

Whatever approach you take for your company, at least ensure that the strategy is actionable and leads to real, tangible effects in your organization. It is too easy to create a circus around strategy development that doesn't result in any business benefit. Let's agree that we do not allow for anymore paper tiger strategies!

Business Agility: What Got Us Here ...

During the last months, I have run into several situations where folks outside of R&D considered this concept of agile a nice little toy for the propellerheads in software, rather than something that is in service of the company. In many cases, these people are stuck in a waterfall mindset and feel that it's perfectly fine to release products once per year, if that often, and to keep selling what they've always been selling. The basic perception seems to be that the slow, planning-economy based approach has worked well so far and there really is no reason to change. All this talk about agile, speed, data and disruption is just noise and it's best to ignore it and carry on.

At these occasions, I explain that the main reason why we care about agile in software is because it is an enabler for business agility. Business agility is concerned with the ability of the company to rapidly respond to changes in its business environment. This means rapidly stopping to do something that we've been doing for a long time when it becomes clear that it's the wrong thing to do going forward. And it means rapidly starting to do something when an opportunity presents itself. Being able to constantly direct the resources of a company to the highest priority and most valuable activity at a moments notice is critical in an increasingly competitive world.

Traditional companies use plans as a tool to align the different functions in the organization. In the yearly planning cycles, the various functions prepare their plans based on the company strategy and then spend lots of time in alignment meetings to make sure that the plans of the different functions result in a successful execution of the overall company plan. Once the plans are approved and budgets allocated, everyone is focused on executing their respective plans. Incentives and rewards are provided based on the ability of everyone to deliver on their respective plan.

Each plan is based on a best guess of the state of the outside world at the point when the plan was created. However, the world is a dynamic entity and the guess might not have been very good to begin with. To paraphrase my favorite definition of "theory", a plan is a beautiful thing killed by a gang of ugly facts. The reaction by those in favors of plans is the quote from <X>: failing to plan is planning to fail. But here's the thing: I am totally in favor of planning. It is the blind, static execution of plans without constant adaptation to the reality that I care about. No plan survives first contact with the enemy, as the military like to say. As one of the product managers in one of the Software Center companies said: I love agile because it allows me to change my mind every three weeks (the length of their sprint).

Imagine the organizational ability to redirect resources every couple of weeks. Assuming you have the right instrumentation to decide what to adjust the resource allocation to, everyone in the organization would always work on the most important goal. The challenge in traditional, hierarchical companies is that to enable this level of responsiveness in the organization requires

empowering the rank-and-file in your organization. It means that individual teams can decide what to work on next based the information that they have collected during the previous sprint. In principle without any manager approval, which of course disempowers many managers in the organization and, in fact, makes their roles superfluous.

In addition, as most work requires competencies from different functions, we need to organize in cross-functional teams. Within the team we can adjust the direction on a continuous basis. Between several teams, adjustment can still be achieved quickly. Between different functions in large organizations, responsiveness is all but impossible. The only times where it has worked to some extent is where the leaders of different functions have a very good relationship and have asked their staff to operate in informal, de-facto cross-functional teams.

Concluding, although in many organizations, the adoption of agile practices starts in software R&D, the principles apply to every part of the organization. The goal is to achieve *business agility* and for this we need cross-functional teams that are empowered to constantly adjust their activities based on their interpretation of the market, customers, technology developments, competitors, partners and others. As the saying goes: what got us here, won't get us there. It's true for business agility too.

Nobody Cares About Your Product

This week was about travel and keynote presentations. Although the traveling can be a bit of a drag, I love meeting people and hear them sell their stories after they have listened to my talk. The interesting thing to hear is that many people talk about the product that they are contributing to and how much their efforts are focused on supporting the system engineering efforts. Although I have written about this in an earlier post, I feel it warrants revisiting: nobody cares about your (physical) product.

During one of the flights I listed to a podcast with Stephen Jurvetson (a very successful VC) and his prediction is that in the near future everything physical will cost 1$ per pound. Everything! Cars, mobile phones, medication, refrigerators, etc. Everything you touch throughout the day will cost 1$ per pound. The progress on additive manufacturing, low cost transport and other technologies is such that you will basically pay a nominal fee for the weight of the product.

The only thing that you can do to get customers to pay is to create intellectual property that adds value beyond the atoms. This obviously includes product design and luxury products. Anyone who has bought brand products at a premium has paid for design and brand. Even though there often is an implicit promise of better quality and trustworthiness, the main reason we pay for it is social signaling. Younger generations are adopting an approach where luxury and frugality are mixed in interesting new ways and the transition from products to services removes, or at least fundamentally changes, the role of design. When ordering an Uber, do you really care about the brand of the car?

The second source of intellectual property is software. As differentiation through atoms is becoming harder and harder, it increasingly is the software that constitutes the real value for customers. In earlier posts, I have written about the shift in value from atoms to bits, but I still run into way too many people, especially here in Europe, who are constrained by the systems engineering types and the broadly held belief that the software is in service of the physical system, instead of the other way around.

The third and maybe newest form of intellectual property is data. This is still early days and monetizing data is still an emerging market. However, it is clear that both high quality datasets that can be used for training and validation as well as live data streams that can be used for real-time value creation are incredibly valuable assuming we can find the right business cases. Especially with the emergence of machine learning and deep learning, this trend is only accelerating.

One of the implications is that the business ecosystems in which we operate are changing rapidly. Recently, I talked to a company that had just lost a bid with a customer to a new, though by now established, entrant in its industry. The analysis of the company showed that the new competitor had sold its product below cost, so at a loss. However, the competitor was doing fine

financially. The conclusion was that the competitor had found a way to monetize data coming from the product and had agreed with the customer to sell below cost in return for the ownership rights of the data.

Concluding, nobody cares about your (physical) product. Because of breakthroughs in manufacturing, low cost transportation and the ability to reach very high economies of scale, atoms will cost a dollar per pound. We will only pay for intellectual property, be it design, software or data. That, in turn, changes the business ecosystems from simple, one-dimensional ones (you and your customer) to complex, multi-dimensional ones where data from one customer category can be monetized by selling it to another customer category. How will your company survive in a world where everything physical is, basically, for free?

Projects Suck; Do Products Instead!

With the vacation in the Nordics largely over, I have had the opportunity again to spend time with several companies over the last week. I noticed a pattern that I had seen before but didn't really reflect upon: whenever a company that I work with is experiencing difficulty in changing to where they want to be, preparing for the future in some way or otherwise drive improvements, its main organizing principle was to structure work around projects. Even if the result of a project is a product, the project mindset was prevalent.

A project mindset implies at least three things. First, the project team's goal and mission is to successfully deliver on the goal of the project on the expense of everything else. Second, any improvement that does not provide benefit to the project itself is not implemented. Third, any risk to the project that can be mitigated and removed without negatively affecting the outcome of the project is avoided.

The result of the project mindset is a situation where change, transformation and improvements come to a complete standstill. Each project ongoing inside the company applies the project mindset and refuses to take a long term perspective or accept risks that come with doing things different from how we have been doing them before. And the more successful the company, the more projects are ongoing and the more resource constraints we have, meaning that the short-termism only gets worse. Improvements that clearly benefit the company overall, but that can't be amortized within the context of an individual project don't happen!

Interestingly, the companies where I observe this project mindset actually deliver products to their customers as a result of their projects. However, there is little ownership of the long term evolution of the product portfolio. The solution, obviously, is to adopt a product or rather product line mindset. Ideally each product is "just" a side effect of the constant evolution of the product line. The project to create a specific product for a specific customer (or customer segment) ideally consists of selecting the components from the product line that are optimal for the specific set of requirements, preferably configuring these components and minimizing the customization, verifying the result and, finally, delivering the result.

A product (line) mindset is ambidextrous in that it balances short-term project needs with a long-term company needs. It can afford to invest in improvements that benefit all projects, even if the cost can't be carried by a single project. And it can afford to take risks as the worst case situation is that a next version of a component in the product line is delayed or fails to materialize. In that case, the projects will have to select among the existing components, but at least there is an alternative.

Concluding, although the title of this article blames projects, it really is the project mindset that is causing surprisingly much damage. The product (line) mindset is superior, being ambidextrous, constantly taking on improvements as well as necessary risks. Even if the project mindset often

feel good due to the dopamine kick in our mammal brains when finishing something, this is one of those cases where perception is not equal to reality. When it comes to long-term success, you need a product (line) mindset.

Enough Efficiency Already! Focus on Effectiveness!

Since the summer, I have worked with several companies that are starting to see continuous deployment on their horizon. This is great progress and brings many advantages such as fast feedback on quality issues in the field as well as the ability to quickly fix any issues that customers experience. Internally, more frequent deployment often triggers a wave of automation of the activities surrounding the release processes.

The realization that escapes many of the companies that I work with is that continuous deployment also allows for some fundamental changes to your development processes. Our research, as well as that by others, shows that somewhere between half and two-thirds of all new features being developed are never used or used so seldomly that the R&D investment in the feature was not justified. And yet, most R&D organizations are steered on cranking out as many features per time unit as possible. That means that the focus is on efficiency, but at the price of building many useless features that might even prove to be harmful as they clutter the system and user interface and result in higher maintenance cost than strictly necessary.

Once a company reaches continuous deployment - which I define as releasing at least once per agile sprint - we can adopt a different development process where we do not build the entire feature top to bottom and including all bells and whistles. Instead, we can afford to build only a slice of a feature, deploy it together with the necessary instrumentation and measure how it performs in practice. If it performs as expected, we can decide to build the next slice. However, in case the data does not match expectations, but we may also decide to cancel the feature or reimplement it in a different way that we hypothesize performs better. This of course focuses the R&D resources on things that actually add value to customers because we have a feedback loop that gives us immediate data on the extent to which we are delivering on customer value.

Most R&D organizations tend to pride themselves on their efficiency rather than their ability to maximize delivered customer value. However, R&D of course expected to deliver value to customers as that’s the only way we have to monetize our products, systems and services. A quick example to illustrate my point is the following. Most companies express R&D budget as a percentage of revenue, say 10%. For an agile team of 5 people using 3 week sprints and on average costing 1K€/day/engineer, this means that a sprint costs around 75K€. Using the aforementioned numbers, this means that this team needs to generate 750K€ in business value in order to justify their investment. In my experience, very few agile teams think in these terms.

Actually, when lifting our perspective to the company level, we often do know the value of products for our company as this is typically captured in accounting systems in the form of revenue. However, we often have a much less good understanding of the value we are delivering to customers. And when we go down to the feature level, it is often entirely unclear what the value of an individual feature is, either for the customer or for the company. This leads to a situation where half or more of features are never used or used so seldomly that the R&D

investment was not justified. With continuous deployment, however, it is possible to measure, at the feature level, whether the intended benefits of the feature, in terms of user behaviour or system behaviour, are being realized or not. That allows us to minimize the percentage of features that are not or insufficiently used by customers.

Concluding, we need to shift our perspective from efficiency (delivering as many features as possible) to effectiveness (maximizing the amount of business value per unit of time or resource). Continuous deployment and the associated feedback loops allow us to fundamentally change our ways of working in R&D and through that significantly improve the effectiveness of R&D. So, stop focusing on efficiency already and start focusing on effectiveness!

Are You Ready To Burn The Ships?

This week I gave a talk at a meetup on business agility. The, admittedly controversial, title of my talk was “Why Digitalization Will Kill Your Company Too”, a topic that I discussed in a previous blog post. The interesting thing during the meetup was that lots of participants asked me why companies are not changing faster. A variety of hypotheses were mentioned ranging from blaming senior management and the customer to referring to human nature and our inborn resistance to change.

My answer to the above question is that there is no silver bullet. There is no one thing that you can do that will successfully change your company. Instead, based on our research, I claim that successfully traversing the digital transformation requires the adoption of a fundamentally new business operating system and a systemic transformation that changes everything in the company.

Similar to computers, organizations have operating systems too. The traditional businesses use hierarchical structures, waterfall-style processes, opinion-based decision making and business models based on transactions and physical products. The new, digital business operating system is based on empowered, cross-functional teams, agile processes, data-driven decision making and business models based on subscriptions and digital services.

In most companies that I work with, the awareness of the new, digital business operating system exists and virtually everyone agrees to some or all of the aforementioned principles. At the same time, though, the operating system that is actually running the business is the traditional one. So, everyone says one thing and does something else.

Almost always, there are experiments going on with the implications of the new digital business operating system concerning business models, customer engagement, team formation, agile work practices, etc. And, not surprisingly, in the vast majority of cases, the experiments are quite successful and provide ample evidence that this is the right direction.

And, yet, frustratingly, moving from experimentation to actually changing and deploying the new way of working does not happen or annoyingly slow. In many cases, there is a war going on in the company between the traditionalists and the avant-garde and it’s not atypical for the old-timers to win and to see the company to slide back into the old ways of doing things.

My conclusion is that this is a lack of leadership. When it’s clear that change needs to happen and the first experiments have been conducted with promising results, there comes a point where you need to jump from the old way of doing things to the new way of doing things. As a leader, you need to decide that we are done with the old way of doing things. That the change is no longer optional and that the old ways are no longer available. You need to burn the ships, take the revenue and margin hit as you stop serving your customers in the old so that you can

build up the new way of serving your customers. It will hurt in the short term, but it saves you from dying a slow, painful death by a thousand cuts as the old way of doing things slowly goes away.

Many companies stay with the old way of doing things too long and only when the revenue and margins have eroded is the momentum for change in sufficient to push it through. The problem is that by that time, the company does not have the resources to do what it needs to get done. Nor does it have the time as it is already way behind more nimble competitors.

One reason why all this happens is that leaders offer their staff the option to use either the old or the new way. Change is treated as optional rather than as mandatory. And virtually everyone, when offered the opportunity to choose between changing or staying the same will avoid change. The only way that I can see is to decide, well before you're with your back against the wall, to burn the ships and make sure that the only alternative is to move forward. Are you ready to burn the ships?

Your Product Is No Longer The Product

Recently, when interviewing (for our research) someone at a company in the food space, I learned something that I hadn't realized: these days, cheese is a by-product. For thousands of years, farmers milked cows and then, as a way to preserve the milk, they made cheese out of it. And that cheese, or at least the surplus, was then sold to customers. Of course, over the last hundred years or so, that process has been industrialized, but the business model has not changed.

Over the last decades, however, because of the fitness boom, weight lifting, body building and other activities, protein powder has become an important and valued product. And it turns out that whey, originally a by-product of cheese making, is rich in protein and little else and is used as a basis for protein powder. This changed things around in the cheese industry as the thing we used to throw away suddenly is the most important and valuable part of the business.

There is an important parallel to what happens in response to digitalization in many companies making physical systems. Over the last decades, we have added electronics and software to our, originally mechanical, systems. And these new technologies have allowed us to collect data about the quality and usage patterns of our systems and the people that use our systems. The data generated by our systems has, over the last years, started to become increasingly important and valuable. And, one could imagine, the physical systems may become the by-product needed to get the data.

The confusing thing is that this in many ways is a Copernican revolution: in some ways, everything changes as the focus is now elsewhere and the functions and roles that used to be the most important are now relegated to the sidelines. On the other hand, nothing changes as we're still building systems similar to what we were building in the past. There are, however, at least three main changes that companies need to implement in order to remain successful when such a change happens.

First, as there now is a new asset to monetize, data, we need to develop a new, second business ecosystem of stakeholders. Currently, in the industry we see new companies appear that look to help traditional companies to monetize their data. Although it's early days still, at least for embedded systems companies, it's important to start to explore that new ecosystem and look for ways to shape it to your advantage, rather than being forced into a role in that ecosystem that may not be optimal.

Second, the new asset requires a new business model. This obviously requires the basics such as ensuring the rights to the data as many embedded systems companies abdicate ownership of the data and implicitly assume that their customers own the data generated by the systems they bought. Although this may be viewed as "obvious", it is anything but obvious and there are many arguments why your company should retain the rights to the data (warranty, regulatory

reasons, quality assurance, SLAs, etc.). In addition, the company also needs to figure out how to price the data and what the “unit of value” is, especially as we’ll increasingly see continuous data streams rather than discrete data sets as the offered products. Finally, once you have a multi-sided business ecosystem, you can decide to potentially subsidize one side of the market. For instance, you could sell your physical systems at or below cost and make up for it by monetizing the data generated by these systems. This is valuable for at least two reasons. First, it allows you to gain market share as you’ll be cheaper than your more traditional competitors. Second, since you’ll have more data from more systems available, your data will be more valuable to those customers.

Third, the design criteria for the systems change as there now is a need to optimize these systems to generate as much of the new value as possible. This means that the company has a set of hypotheses on the types of data that would be valuable and, typically, it means developing solutions to allow for the collection of new types of data even after the system has been put in operation. This often leads to tension in the organization as the new requirements frequently require compromises on the system properties, such as cost, reliability, safety, etc. that were prioritized and that remain important going forward as well.

Concluding, with the digital transformation (software, data and AI) affecting industry, many companies are experiencing a situation where what used to be their primary value proposition takes a secondary role and new value propositions, driven by data, become more important. This has significant implications of which I discussed three in this article. The question that I hope to leave you with is whether your cheese is at risk of becoming a by-product also?

I. Leadership

Few topics are as elusive to define as leadership, but yet we all know it when we experience it, either as leaders or by being exposed to leaders. Leaders define a compelling vision, invite you to join, facilitate when the team runs into difficulties and treat everyone with respect without the organization falling apart.

Reams of paper have been filled with reflections on leadership, but a surprisingly large amount of material on leadership really concerns management. Management relies on the traditional command and control structure that forms the backbone of traditional hierarchical organizations. In this view, individuals are interchangeable cogs in the machine. The cogs are viewed as in support of letting the machine run as smoothly as possible.

In a world where all work that can be automated is automated, the humans in the organization are less and less concerned with repeatable and repetitive tasks. Consequently, rather than organizing work for humans in repeatable and standardized processes, each work item will be unique and require significant creativity. And as a consequence, we need to organize work in cross-functional teams that focus on achieving a defined outcome but that by and large have the ability to use whatever means available to achieve that outcome.

When the focus shifts to multi-disciplinary, cross-functional teams, the role of leadership is to define the right outcomes to aim for, support teams with overcoming difficulties, help individuals grow and develop skills and provide the context in which these teams can be optimally productive.

In this part of the book, we discuss a number of aspects of leadership, including the contrast with management, the importance of empowerment, avoiding firefighting, the difference between activity and progress as well as allowing for and fueling diversity as a means to drive innovation and creativity.

On Hierarchy, Micro-Management and Leadership in R&D

Over the last years, I periodically visited a company that I have really admired for a long time for their focus on autonomous agile teams. These teams have a lot of freedom within their area of responsibility and can choose virtually anything from the platforms to use, the tools to use, when to release, what functionality to build, etc. The company is data driven and the teams get a lot of energy out of knowing that they were contributing to the revenue of the company as there were metrics in place that allowed teams to estimate their contribution.

Recently I visited them again and learned that several of the people that I had talked to over the last year either had left, were in the process of leaving or were planning to leave. So, what happened? Why this sudden change of heart among the R&D staff? It turned out that new senior R&D management had come in. They had analysed the situation, determined that freedom that the R&D teams experienced was causing major inefficiencies as there was no sharing of commonly needed assets, no focus on cost and lots of duplication. In response to this conclusion, they centralized decision making and used the reporting hierarchy to enforce a new set of decisions and consequent behaviours. The result was that attrition went through the roof as many felt that the implicit contract between the company and its employees was broken. And in a pretty hot market for engineers, finding another exciting opportunity is not difficult at all.

So, what can we learn from this? The wrong conclusion is that R&D management should have left the teams alone. Especially in fast growing companies, there are points where a course correction in the way that R&D is conducted is required. Continuing with the “old” way of doing things even though the company needs a “new” way of doing things is as risky, if not more so, than not changing. So, the question is how to realize this course correction without all the negative consequences described above.

The R&D management at the aforementioned company fell back on the only mechanism they new: traditional, hierarchical management, centralization of power and micro-management to enforce the new behaviour. Of course, this means disempowering teams and individuals and treating them like replaceable cogs in a machine. This is extremely demotivating for engineers and this causes those who have a clear idea of how they want to work and that have alternatives available to seek greener pastures.

One of the main characteristics of agile development is that it empowers teams and individuals to perform their work in the way that the team considers best. As such, agile development is at the forefront of a major transition in industry towards self-managed teams and organizations. In our research (see reference below), we have studied how organizations adopt empowerment and self-management. As shown in the figure below, organizations transition from traditional to empowered organizations through a number of steps.

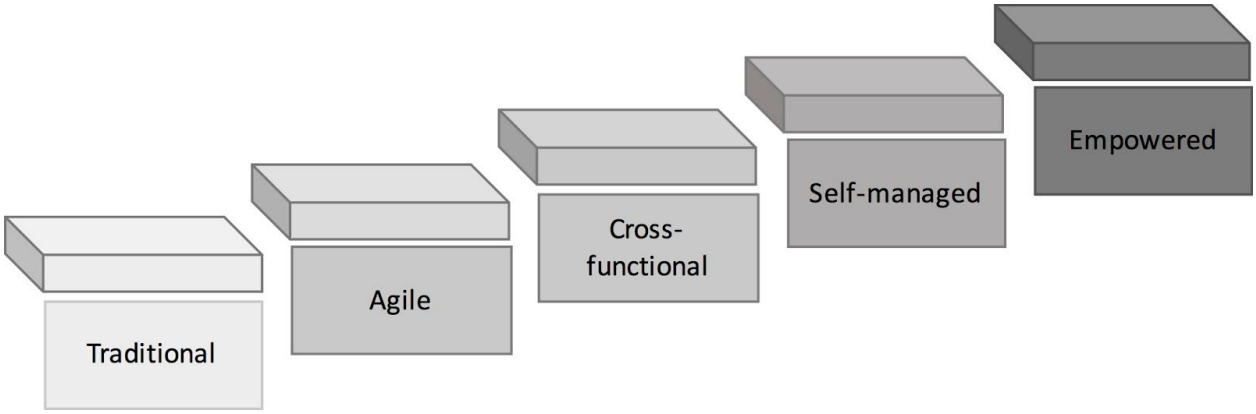


Figure: evolution of empowerment in companies

The table shows how self-management and empowerment starts in teams and then spreads throughout the organization until even the culture in the company is infused with this kind of thinking

	Traditional	Agile	Cross-functional	Self-managed	Empowered
Culture	Hierarchical	Hierarchical	Hierarchical	Hierarchical	Empowered
General Mgmt.	Hierarchical	Hierarchical	Hierarchical	Empowered	Empowered
Inter-team (PdM/R&D)	Hierarchical	Hierarchical	Empowered	Empowered	Empowered
Local (R&D)	Hierarchical	Empowered	Empowered	Empowered	Empowered

Table: relating functions and empowerment level

In this new world of empowerment and self-management, the use of hierarchy and micro-management is not longer the strategy of choice. If you fall back to that, your people will just leave you. Instead we need *leadership.* We need leaders who can explain to and convince teams why changes are necessary and what the road is that we need to follow to get there. And once the goal and path are clear, the teams need to be empowered to figure out how to get there and the leaders become coaches and mentors. We need to achieve self-managed and empowered organizational change.

Reference:
Olsson, H.H., and Bosch, J. (2016). No More Bosses? A multi-case study on the emerging use of non-hierarchical principles in large-scale software development. In Proceedings of the 17th International Conference on Product-Focused Software Process Improvement (PROFES), November 22nd-24th, Trondheim, Norway.

Leading through Disruption (or not)

One of the ongoing questions in my mind is why large companies have such a difficult time managing disruptions and, as a consequence, why the average tenure of companies on the Fortune 500 has now decreased to 10-12 years. My thinking around this question got triggered recently, when I received an email in response to a talk I gave at a large company. Historically, this company has been extremely successful, but more recently it has fallen on hard times. My talk was intended to reenergize the employees and shift their focus from the recent not-so-bright past to a promising future.

Below, you'll find the email (anonymized and shortened to protect the individual and the company). I am sharing it because I think it summarizes quite eloquently the trouble that many of the large companies that I work with struggle with.

Hi Jan,

Thank you so much for your talk today at our company! It was very inspiring to see what direction your thoughts and research have taken. I can relate to many of the points you made. You put words to many of the thoughts that has been in my head the past year. If you have time, it would be very interesting to have chat about how to create an innovation culture in general and at our company in particular. My apologies for the lengthy email. I realized I had a lot of thoughts in this area. I am very interested in your view, but I fully understand if there is no time to read it through.

The past year has been a journey for me, where I have had the opportunity to think a lot about strategy, innovation and leadership. I started a business program and at the same time, I was very lucky to meet some very inspiring leaders. The more I have learnt the more lost I have felt. I look at my work surrounding and I compare what we do with the surrounding industry. I keep hearing "We have managed many crises before, we will handle this one too" and then we do as we always do. Don't misunderstand me, I really feel for my organization, it is an amazing company and has achieved great things and I can fully understand that we are in a red alert mode and that things might get wrong when you are panicking. No matter how much I think about, how much I read, how much I discuss with others, I cannot change the uneasy feeling inside of me saying that we are doing things wrong.

You are an evangelist for new ways of thinking and you are a very important authority at my company. Your words are echoed many times in internal discussions. However, I feel that the very core that needs to be changed is rather untouched - our leadership culture. I am a strong supporter of servant and transformational leadership. I can see that very many first line managers are trying to be that too. However, middle management and top management seems to be more transactional leaders. This creates a large conflict in the organization, with a notion that we need to innovate but we are not allowed to fail. Somewhere, the basics and

fundamentals seem to have been lost. If we want an innovating organization, we need to be allowed to fail. If we want our engineers to come up with crazy ideas, they need to understand the problems and the bigger picture. For me it becomes so incredibly clear that the view of our employees needs to be changed. We can no longer regard our employees as a cost on an excel sheet and instead see them for what they are, our most important asset. We need to live our core values and show them the respect they deserve. It might sound judgmental, and that is not my intention at all. For me this is the consequences of the decisions that has been taken. Everything, such as the choice of organizational structure, incentive programs, different roles, leadership trainings, C-level nominations, etc., promotes certain behaviors and builds the culture. Given what I see, I feel a considerable uneasiness. I sense a large mismatch between what we say we want and what we actually do. The more I think about it the more alienated I feel. Do I see things that others don't see? I am not particularly intelligent or very experienced, so I start doubting my own conclusions/insight. Why don't I hear these kind of discussions more often?

First line managers and developers seem to agree with me that innovation is important and it calls for another leadership style. However, at second line management level the agreement with that statement seems much weaker. During the last re-organizations it seemed that managers considered too radical were removed, especially those having the most progressed thoughts about leadership and how to encourage empowerment and innovation. What I see, is a strong force of conformity. What is different seems scary. A manager sticking out too much is not conforming to the expectations as a manager. A small variant of the classical "Jonathan Livingston Seagull" syndrome. In times of distress and panic like now, this behavior is enforced even further. Embracing the different is seen as a risk and the current focus is to minimize risks. From a sociological perspective this is very natural, we tend to like and choose persons that are similar to us, persons that do not deviate too much from how we are. For me this becomes very myopic and somewhat a vicious circle. I hope I am missing something. We seem to have a culture and structure that try to counteract cultural changes and maintain the status quo.

I feel that in order to create an innovation culture at my company, we cannot do that bottom-up. It needs to come top-down, with an enormous buy-in from corporate management. Having Peter Drucker's "Culture eats strategy for lunch" in mind, I realize how incredibly difficult that is. At the same time, the overall principles of human nature, human organizational and leadership behavior is not unique to my company. As far as I know, we do not belong to another species ;). Is it at all possible to transform a company from a traditional form into an innovation company within reasonable time? Or are traditional companies doomed to succumb to disruption?

Do you think it is possible to shift my company's prospects without fundamentally addressing our corporate culture and financial strategies? Are we right now just scratching the surface? Or are we actually doing something big without me realizing it? What are your thoughts? What are your view on other companies? Are other large corporations more innovative than us? If so, what differs them from my company?

Best regards,
<name removed>

This email summarizes the key challenges that I see in many large companies. First, companies want to innovate, but innovations are not allowed to fail and individuals are punished for trying things that don't work out. Obviously, the vast majority (90%+) of innovations fail in the companies that I have worked with, so not accepting failed innovations is the fastest way of killing initiatives by your people. Second, for all the talk about embracing diversity, many companies, especially when things are difficult, seem to become increasingly homogeneous. People, especially in management positions, become more concerned with mimicking the behaviour of the most powerful managers rather than following their own beliefs and insights. The hierarchy becomes a structure that reinforces the qualities and weaknesses of the top dog. Third, as the margins fall, the focus increasingly shifts towards delivering to customers and cost reduction efforts. In part this is a, largely subconscious, desire to return to the past where the company was successful and to reduce uncertainty, but also by economic realities: to keep the stock price up and the owners happy, the company has to generate cash. Finally, many large companies are run by bean counters and numbers people. That works fine in a stable environment where the focus should be on optimizing the last bits out of the business model and R&D process. However, it works terribly in fast changing and disruption-heavy markets. There we need product people in the lead that understand the various things that engineers try out, understand that the majority will not work out, but that still have the benefit of educating the company. And that have the vision to identify those new innovations and product ideas that will drive future revenue.

The traditional model of innovation was that the R&D organization gets optimized for efficiency and the innovations come out of the research lab. That model is now fundamentally flawed. In my experience, the research labs are only concerned with filing patents and all the simple, repeatable R&D work is automated or outsourced to low-wage countries. The role of the R&D organization is to innovate and build new businesses. No matter what the general management of the company may believe and controversial as it may sound, the real business strategy of any company is developed by R&D.

The interesting thing is that I still have to meet the first person who disagrees with the points above on a conceptual level. The challenge is that operationalizing this when the company is in trouble is surprisingly hard. All the decisions taken by managers, from the most senior levels all the way down, are individually correct or at least defendable. The problem is that it puts the company in a vicious cycle where things get worse continuously.

To get out of the vicious cycle and into the virtuous cycle requires **leadership**. The kind of leadership that shows the non-obvious direction, that explains why doing things the way we did is wrong and that keeps the company from clinging to its comfortable past and gets it to embrace the uncertain future so that the company can create its own destiny, rather than waiting for others to disrupt it.

Leading in the Digital Age

During the week before Easter, I decided to take some time off work and escape into the Swedish wilderness (Hallandåsen in northern Skåne, for those who care about the local geography). Although I regularly read books of various kinds, this week I read several books about self-managed or empowered organizations (including Turn This Ship Around by David Marquet, Remote by Jason Fried and The Seven Day Weekend by Ricardo Semler). Although I have written about the topic of empowerment in earlier blog posts, I realized that I haven't written about the implications for leadership.

As we're going through the digitalization transformation, there are a number of fundamental shifts to how organizations operate or can operate. The first is digital technologies allow for unprecedented levels of transparency. Sharing data across the entire organization as well as processing data to tease out specific insights has never been easier. As traditional organizational structures often are based on information asymmetry between managers and frontline, increased transparency reduces need for traditional manager roles.

Second, digitalization and automation specifically has largely removed humans from repetitive, easy to standardize work. Whereas IT for decades has focused on supporting humans in doing their job, now the focus has shifted to full automation and removal of humans from the process altogether. This means that the only work involving humans is work that is complex and hard to standardize. As a consequence, decision making is highly contextualized, meaning that the context plays a large role in determining the best way forward. Obviously, this means that the people best suited to make decisions are those closest to the specific situation. In other words, the team rather than some distant manager.

Third, education levels across society have never been higher, meaning that everyone in the organization has the knowledge and, with time, the experience to make decisions. Traditional organizations stress the notion of experts and specialists that know everything about a narrow area, but very little about anything else. As these people can't make decisions that incorporate all relevant dimensions, you can't rely on experts to make decisions. This creates a necessity for managers to bring together the different views and perspective and make decisions. Empowered organizations insist on everyone understanding the business that they're in so that individuals and teams can take decisions locally that align with the principles of the organization.

These shifts due to digitalization, automation and empowerment, leading in the digital age requires different skills and behaviours from leaders. Below, I describe some of the key differences.

Principles over orders: Leaders set the principles based on which the members of the organization define their behaviour. Leaders do not give orders or tell people what to do. Instead, everyone in the organization is expected and trusted to apply their best judgement,

based on principles, to specific situations. Although peers may (and likely will) provide their perspective, leaders don't second-guess and punish if decisions get made that turn out to be less than optimal.

Personal leadership over leader-follower: The second difference is that everyone has to exhibit personal leadership. This means that everyone knows what he or she wants to accomplish, understands the context in which they operate and takes decisions and execute without being told what to do. Out of personal leadership, in teams natural leadership can evolve where team members take temporary leadership roles during certain periods and then relinquishing leadership when the situation calls for it.

Trust over audits: The moment an organization decides to audit, review, inspect or in any other way checks the way in which employees do their job, employees will stop taking full ownership and responsibility for their actions. In empowered organizations, everyone needs to know that the buck stops with them. Feedback, support and learning needs to come from interaction with peers, rather than from hierarchical managers.

Customer over organizational structure: All successful digital companies that I work with have put the customer at the center of everything they do. Most companies say that they do this, but in practice the internal organizational structure, in terms of rules, regulations, etc. take over and customers lose out. In complex ecosystems, there may be multiple customers. In that case, it is important to choose who is the customer that the whole organization will rally around.

One blatant example of failing to focus on the customer was this week's United Airlines debacle where a customer ended up in the hospital after being forcibly removed from the plane because employees of United Airlines needed to get on the flight.

Team-appointed managers over manager-appointed teams: If your organization needs managers for whatever reason, leading in the digital age requires an inverted hierarchy: teams choose their managers and provide frequent feedback on how managers are doing. And managers step down if the feedback is not satisfactory.

Diversity over homogeneity: Traditional organizations tend to value homogeneity and many employees feel pressured into talking the same, dressing the same and agreeing with the same. The risk is that companies turn stale and start to believe in their own hallucinations. Instead, organizations need and should encourage diversity. Different people have alternative opinions and, sometimes heated, debates are necessary for the company to continuously reinvent itself, find new markets and grow.

Agility over long-term planning: Long-term plans provide false predictability and an illusion of control over an uncertain, fast changing and unpredictable reality. In many organizations, adherence to the plan wins over adjustment to reality and the only way to avoid it is to not plan

for the long-term. Of course, some planning will be required, but limit any plans to maximally 6 months out.

Emergent over top-down strategy: A direct consequence of the previous point is that the organization should avoid long-term, top-down strategies. Strategy should emerge from the teams. As part of their mission setting and (6 month) planning, strategy should be one of the topics considered by teams.

Concluding, leading in the digital age requires a fundamental shift. It requires empowered, democratic organizations where leaders set principles, listen rather than tell and, where necessary, take temporary, meritocratic leadership roles. This is a major departure from traditional, MBA-style, hierarchical management, but judging from the constantly increasing rate of disruption of traditionally managed companies, the old model isn't working too well.

Meetings Are A Waste Of Time (And What To Do About It)

The traditional form of coordination in organizations is meetings. The challenge is that most meetings are incredibly ineffective and a waste of time for everyone involved. All of us have perfected the behavior of pretending to pay attention while working on other things at the same time. The problem is of course that during meetings, it is virtually impossible to do deep, hard work and work is constrained to the easy administrative tasks. Like many others, during my time in corporate, many of my days consisted of 8 hours of meetings, some hours of email and then, finally some real work in the rest of the time.

On the subject of coordination, I believe that many organizations can learn quite a lot from the architecture and development of software systems. In software engineering, there is a taxonomy of coordination mechanisms that teams use that are preferable over meetings. These mechanisms are, in order of preference, architecture-driven, continuous-integration centric, social media centric and, finally, meetings. Below, I'll describe each of these approaches in more detail.

First, architecture-centric coordination decouples teams working at different sides of architectural interfaces. As long as both teams meet the agreed specifications on their side of the interface, there is little need to coordinate. Of course, no interface is nor should be static and architects periodically evolve the architecture and the interfaces between different parts. At these times, there will be a need for other coordination mechanism between the architect and the teams. However, with proper interface management and the use of deprecated interfaces, the coordination cost is minimal. Although this works well for development teams, it also works in other context where teams work at different sides at an organizational interface. The same requirement needs to be satisfied as with development though: the primary activities should mesh with this architecture. Some companies have perfected this model into a service-oriented organizational architecture where two-pizza teams provide services to internal and external customers through predefined interfaces and have instrumentation in place to measure each team's performance.

In software development, a second mechanism for coordination is through continuous integration. Here, the development organization uses the build and test system as a way to coordinate between teams. When two teams touch the same component and create incompatible changes, the team that checks in its code last will run into the incompatibilities as the test system rejects their code and this team then has to fix the problem. Even though this may seem effort consuming, organizing a meeting every time we might experience an alignment problem, which is the practice in many companies, wastes lots of time by bringing two or more teams together to discuss alignment. Outside of software development, the same mechanism can be used between teams, especially when IT systems are involved. By putting clear acceptance criteria on the contributions of different teams, preferably algorithms but otherwise humans can accept or reject the contribution.

Third, although surprising to many and not the least me, social media tools such as Slack offer a powerful mechanism for teams to coordinate internally as well as for cross-team coordination. Even though it is an approach that requires time and attention from humans, rather than using automation, it often offers a better, real-time and lower effort way to maintain alignment. Although tools like Slack are used by many agile development teams, we see very broad adoption in the industry at large outside of software development.

Finally, for all the issues associated with meetings, there are times where it is the best approach to achieve alignment. There is solid research on how to run effective meetings, including the need to only invite the key stakeholders, have a clear agenda, time box items as well as record and communicate decisions. I won't go into the details as there is so much material online, but the first step has to be that those in the meeting actually need to be there and have something to contribute. Just remember that Peter Drucker (the famous management professor) said: "meetings are a symptom of bad organization".

Concluding, coordination is an important, but expensive activity in companies. Strategically choosing the best mechanism for achieving the required level of alignment with the right and most cost effective tool is important any organization. Just remember that in the vast majority of cases, organizing a meeting is not the right mechanism.

Why Firefighting Ruins Your Company

There are few sessions at companies that I run or participate in where there isn't some form of firefighting that makes it into the discussion. In virtually every company, there are situations where a critical bug is found in the field, the CEO of a customer company escalates to senior management, a critical project is severely behind schedule or something else happens that causes the whole company to go into "fight or flight" mode. Often, a crisis team is formed, members, typically those considered as the most competent, are pulled out of their day job, normal work is suspended for them and everyone supporting the crisis team and the long march towards resolution of the crisis starts. In automotive, these crisis teams are often called "task forces", but every industry has their own version of it.

What is positive is that the approach works, at least when looking at the crisis itself. By bringing a multi-disciplinary team of experts together, giving them a crystal clear goal and full freedom to execute in any way they want and giving freedom to ignore any standard processes around work, expenses, customer contact and other topics, the team often is amazingly effective at addressing and resolving the issue. And as humans, we hardly ever get a larger dopamine kick then when working brutally hard towards a goal and then achieving it. From a social perspective, the people that address the crisis gain status in the social hierarchy, get rewarded and often are publicly celebrated in the organization.

The challenge is that firefighting leads to disastrous consequences for the rest of the organization that often are ignored. There are at least three main issues that can directly be associated with firefighting, i.e. hero culture, lack of commitment and loss of competitiveness.

The first challenge is that the company often develops a hero culture. The heroes are those that take on a project that is going to hell in a handbasket and they "save the bacon" for the company. The heroes get rewarded and the message that is sent in the company that being a hero is something to aspire to. It easily leads to a culture where the normal, day-to-day work is not taken quite as serious as it should and people are almost waiting for things to be escalated, the task force formed and fun starting. Rather than focusing on the root causes of escalations and making changes to architectures, processes, ways of working and the culture to avoid escalations in the future, the company spirals down a path where escalations, crises and task forces become the normal way of operating.

Second, in the organizations where escalations and crisis teams are common, the accountability and commitment of individuals and teams suffers. As the crisis team can demand work from individuals and teams on a moment's notice, the normal sprint content comes under pressure and it becomes acceptable to fail on delivering the committed sprint content with the excuse that too much time has been consumed by responding to urgent requests and demands from firefighting teams. Although this often starts as an exception for all the right reasons, it rapidly becomes the norm and this easily leads to finger pointing and politicking to protect one's brand

in the organization. The consequence is that teams no longer dare to depend on promises from other teams. This, in turn, leads to a situation where each team, being responsible for some project, seeks to bring everything under its own control, which then in turn leads to significant duplication of effort, clone-and-own branching and other behaviours that destroy R&D effectiveness.

Finally, when customers of the organization realize that the company tends to respond strongly to escalations, the power balance between the customers and the company changes in favor of the customers and this will be exploited. Most customers will increasingly rapidly escalate whenever they feel the need, resulting in the company spending an ever larger portion of its R&D resources on customer-specific work. As these resources are not building functionality that benefits all customers, but rather fix the issues of only one customer at a time, the company easily starts to slide down the consulting path. However, the business model in the industry often assumes that R&D efforts can be amortized over multiple, if not many, customers. So, the companies that most easily respond to escalations invest the least in differentiating product functionality. And this naive interpretation of focusing on the customer causes exactly the opposite: starting from the customers that are less willing to escalate, the company will lose its customer base to competitors that manage to avoid ending up in firefighting.

Concluding, I am by no means saying that escalations can always be avoided and that there never is a need to take immediate and drastic action. My point is that this tactic should be used sparingly and only in the most extreme of cases. And that leaders should thank the crisis team for getting the company out of a pickle, but at the same time clearly communicate that the intent is to never end up in this place to begin with and take the necessary actions to reduce the need for firefighting situations in the future. Failure to do so will, over time, destroy your company. You have been warned!

Activity Is Not The Same As Progress

After another week on the road, I am typing these words late Friday evening on a plane home. I spent time with three different companies (who shall remain unnamed) this week and on the way back I was reflecting on the commonalities of what I learned this week. The main lesson was: lots of activity; not nearly as much progress.

In each of the companies, I saw lots of activity. Different people were beating on their respective chests, looking to impress the people around them with all the energy and activity that they manage to initiate and drive themselves. And often with a very convincing story to associate all the waves that they're creating. However, when looking into the actual impact of all that activity, the picture turns much less rosy: much of the activity does not actually lead to any progress for the business.

The reason for people starting activities is clear and typically well intended. Especially in times of transition, it typically is impossible to get a clear, strategic direction from top leadership. There is a lot of confusion and forces pulling in different directions. For a forward leaning individual, it often feels impossible to sit and wait for the murky waters to settle and then it's better to start swimming in what is perceived as the right direction. It may well be better to start moving, rather than wait for clarity, but it is far from clear that the activity will lead to a positive impact on the company.

Second, when it comes to R&D, there is the important dimension of time: although business can change it's mind on a dime, the reality is that all the software resulting from hundreds, if not thousands, of person years of effort can not really change that quickly. In one of the companies that I visited this week, the R&D organization had started to refactor their system more than 3 years ago to enable a business strategy that had materialized only in the last months.

Third, many organizations suffer from a constant creation of smaller and bigger kingdoms, organized by business line, function, country, site or, in the worst case, even individuals. Although these kingdoms often are surprisingly resilient and powerful, the kings and queens reign based on the rights provided by top management. A well known saying in management is that "perception is reality". Consequently, creating a perception of activity is beneficial for the kings and queens interested in extending their reign.

Although the aforementioned and many other reasons are compelling, the fact remains that we are interested in driving real, tangible and sustainable progress in the company. Much of the activity does not necessarily contribute to that desired progress. So, next time you're initiating some activity, project, task force, action or any other work item, ask yourself: is this activity or will this result in progress.

Why Trust Is The Foundation Of Success

Many understand the importance of trust in a society. The ability to park your car somewhere and know it will be there, undamaged when you get back. Booking and paying for a travel ticket or hotel room before arriving and knowing that it will be there. Working for an employer and knowing that there will be a paycheck at the end of the month. Walking home late at night and not having to worry about being robbed, stabbed or shot. There are numerous examples where citizens act based on trust and the more your trust is rewarded, the more trust in society you have.

A well functioning society is a trusting society where everyone feels a basic level of safety for themselves and for their loved ones. This is one of the reasons why I dislike all the "apocaholics" (those constantly predicting the next apocalypse) going around like Chicken Little shouting "the sky is falling, the sky is falling" as much of their argument is concerned with reducing the level of trust that exists in a society. Of course, there are many things in society that could be improved, but playing the fear and distrust card often is unnecessary at best and destructive at worst.

What many fail to recognize, in my experience, is that trust is equally important in organizations. In one of my earlier blog posts, I described a pattern that I often see in companies: a team has learned the hard way to not rely on promises from or the work of other teams. As a consequence, the team works hard to bring everything under its own control. This tends to cause an enormous duplication of effort, a significant reduction of specialization-based productivity improvement and a politicization of the company as everyone is constantly fighting to maintain their independence.

As human beings, we go through three stages in our lives: dependence when we rely on our parents, independence when we learn to stand on our own and, finally, interdependence when we enter a long-term relationship where we become dependent on each other. Companies can be viewed as going through the same process, but those that have low trust levels never reach the interdependent state and remain in a kind of adolescent culture where every individual and team aims to remain as independent as possible.

It is often hard to quantify how destructive a low level of trust and a pervasive lack of interdependence in organizations is. When every individual and team in the organization meets others with a basic distrust and a "guilty unless proven innocent" attitude, all the benefits of operating as one company disappear. It leads to finger pointing, factions, the building of kingdoms and a generally internal, rather than external, focus that sets up a company for disruption.

In the end, much of this boils down to culture and this is where leaders need to role model. Although many parents say to their kids "do as I say; not as I do", both in families and in

companies, it is the behavior that gets copied, not what one says. So, as leaders, do you constantly control and micromanage or do you trust your people? Do you escalate and form task forces or do you trust the team that is in trouble to sort things out and ask for help when they need it?

The final point is that the ecosystem surrounding your organization is becoming increasingly important. I have seen many examples where the culture inside the organization is carried outward into the interactions with your ecosystem partners. And, different from your own people, those partners have a choice of who they want to work with. Typically they prefer to work with companies where there is mutual respect and trust and people avoid companies that are based on distrust, blame cultures and politics.

At the other end of trust, of course, is accountability and conscientiousness. It is not enough to trust others; you also need to be a trustworthy person and your team needs to be reliable. One of the simple adages is that to be successful, you need to say what you do and do what you say. Sometimes it is as simple as that, even if executing according to this principle can be incredibly hard.

Concluding, trust forms the basis for successful organizations as it is the fundament for collaboration and mankind, western society, families and your company are based on collaboration. As a leader, you need to role-model trust and trustworthiness. Companies that fail to build this culture fail and are setting themselves up for disruption, especially in a rapidly digitalizing world.

Don’t get stuck in your company’s echo chamber

The last weeks, I have attended several conferences and events and I have noticed a remarkable difference between public and company internal conferences. The internal conferences have a tendency to have a set of themes that everyone talks about in the same way. Whether it’s ecosystems, blockchain, artificial intelligence or any other popular topic, everyone in the company seems to be talking about the topic and its implications for the company in the same way.

Successful companies tend to have a strong culture, sometimes bordering on a cult, and the definition of a culture of course is a common set of norms, values and behaviours. Although the company culture is an important element, there are several risks associated with it. Here, we’ll discuss three.

First, some of the companies that I work with have a tendency to create a set of stories about themselves, their ecosystem and their expectations about the future that in some way become self-fulfilling prophecies internally. The danger is that it may lead to companies investing significant resources based on a set of beliefs in the company that are not aligned with reality. The problem is that, sometimes after years of investment, the product or service is released and is met with a deafening silence because the internal beliefs are not aligned with market reality.

Second, especially companies that have been very successful at some point tend to get stuck with what made them successful in the past. So the prioritizations, the ways of working and view of the customers and their needs stay with what has made the company successful in the past and fail to evolve. This is especially seductive because successful companies became that way because they went against the grain of the industry and acted on a counter-intuitive belief that turned out to be successful. It’s very easy to stay with that belief, even when reality has passed you by.

Third, the strong culture has a tendency to kill new innovations and ideas in the company as there is no or very little space for dissenting voices. So, any new concept gets killed in the cradle before it has a chance to prove (or disprove) itself. And the interesting thing is that the discussion tends to circle around why a new idea can never work and the opinions of people at the company reinforce each other.

The common denominator of the risks discussed above is that the company has become an echo chamber where people tell each other what they expect to hear and repeat the same things that other have shared with them. This is very natural human behaviour that we all tend to gravitate to as we want to be liked by those in our community and the easiest way to be liked is to agree, share common opinions, norms and values and repeat what others likely want to hear.

How do we break out of this echo chamber and avoid getting stuck in it? Well, the key is connect with people in your “weak network” or even people completely outside your network. To observe, learn and start to understand other points of view with the intent of bringing these concepts back to your own organizations. In many internal conferences I am the only outsider and then it’s my role to bring new viewpoints, ideas and concepts to the participants.

A second mechanism is to attend industry conferences, but again ensuring that you don’t get stuck in the same echo chamber, but now at the industry level. One conference that aims to offer novel, fresh insights and that I am very much looking forward to is the upcoming Software-Centric Systems conference SC2. Of course, I may be biased as I am involved in the organization of it, but if you happen to have some time on October 10 and are able to take yourself to Eindhoven, there are few better ways to spend your day than attending SC2!

Concluding, successful companies tend to have strong cultures as the culture is instrumental in aligning the efforts of many people. However, this strength has its inherent risks, including creating self-fulfilling prophecies, getting stuck in the past and killing promising new innovations. Don’t get stuck in the echo chamber of your own company but instead make sure you get inspired from many and very diverse sources. It’s by far the best insurance for staying relevant!

Innovation

Few words inspire as much wishful thinking as the term “innovation”. Every company wants to be innovative, people dream of being innovators and governments stress that economic growth and development originate from innovation. Interestingly, every company, individual or government focusses on the outcomes of successful innovation and wants more of that. The societal impact, economic benefits and social accolades that come with having been at the birth of a major innovation.

The first distinction that we have to make is to differentiate between sustaining innovations and radical innovations. Sustaining innovations make new versions of products and services more attractive for existing customers or at least maintain the relevance of offerings when functionality in products tends to rapidly commoditize. This requires a continuous addition of differentiating functionality. Although it is hard to predict whether these sustaining innovations will be successful, the return on investment tends to follow a Gaussian distribution curve. This means that sustaining innovations on average return a similar amount.

Radical innovation, on the other hand, is concerned with new offerings (products or services) for new or existing customers. The reality of radical innovation is that it is hard work of the kind that makes the return of ones efforts quite uncertain. Radical innovations may seem very promising when interviewing potential customers, but it turns out that what people say and what people do in reality differs quite significantly. The challenge with radical innovation is that the return on investment tends to follow a power function, meaning that a few innovations have far outsized returns whereas most innovations do not provide any return. In mature organizations, accepting that most innovations will be waste is very difficult and people leading those innovations often experience negative implications for their career.

Although I will not even try to present an approach for successful innovation, there are two factors that are, in my experience, important. The first is that any successful innovation needs a customer, preferably many, and to provide anything novel, the perceived benefit has to be significant in order to overcome the resistance against change. This requires deep customer understanding and empathy. It is not enough to talk to customers as no customer is able to verbalize the need for a breakthrough innovation. Instead the innovation team has develop hypotheses concerning concepts that might provide value and then test these.

The second factor is that the success rate of innovation is, or rather should be, rather low. This means that the number of innovation experiments should be quite significant. When there is space for only a few innovation experiments, the tendency is to reduce the risk and go for the easy wins that tend to result in small returns. Any breakthrough innovation was, at the point of introduction, laughed at by the people in the industry, viewed has highly risky, or both.

In this part of the book, we discuss a number of aspects of innovation, including the ones above, but also other topics such the relationship of innovation with agile and the importance of failure.

Innovation Is Hard Work

This week I spent a few days at the OOP 2017 conference in Munich, Germany. Before, during and after my talk there the topic of innovation came up frequently. It seems like every company has its own garage, lab and open innovation initiative. They all have read the lean startup book, know about unstructured time and doing away with hierarchy so that the intrapreneurs have the chance to come up with great, breakthrough innovations.

Now, don't get me wrong: companies in virtually any industry need a major shift from focusing on efficiency to focusing on innovation. Everything that can be made more efficient should be automated already and if it hasn't happened yet, it will soon. My concern is the companies put their understanding of the innovation infrastructure in place, create some space for people to work on innovation and expect that innovation will somehow magically happen. Many fail to realize that innovation is really hard work.

The biggest misconception is that ideas matter. In a way they do, but ideas are a dime a dozen. It's really easy to have ideas and to have lots of them. Just run a brainstorming workshop and if you really get the team going, you'll end up with more ideas than you can handle for the rest of your life. The value is not in ideas, but in *validated* concepts. Concepts that have been tested with customers and ecosystem partners and for which the feedback has been positive.

Validation of innovative concepts is not a single step, but rather an iterative process where confidence and investment in the concept go up with every iteration. In the early stages, one can identify three stages of validation. First, it's often a good start to ask around inside the company itself and get feedback. This can be from colleagues, experts as well as people in different functions and parts of the organization. The next step is to ask customers and ecosystem partners outside the company to provide their perspective and their interest in the concept. The challenge with this step is that we only learn what customers and partners say they will do. Lots of research confirms that there is a significant gap between what people say they do (espoused theory) and what they actually do (theory in use). Consequently, the third iteration is where through the deployment of a minimal viable product (MVP) or minimal viable feature (MVF), we are able to measure actual customer and system behaviour.

In our research (see for instance the reference below), we have worked on this approach in various contexts. The figure below shows the process of idea generation, the use of an adapted business model canvas and then a funnel of four steps, i.e. validate problem, validate solution, validate small scale MVP and, finally, validate large scale MVP.

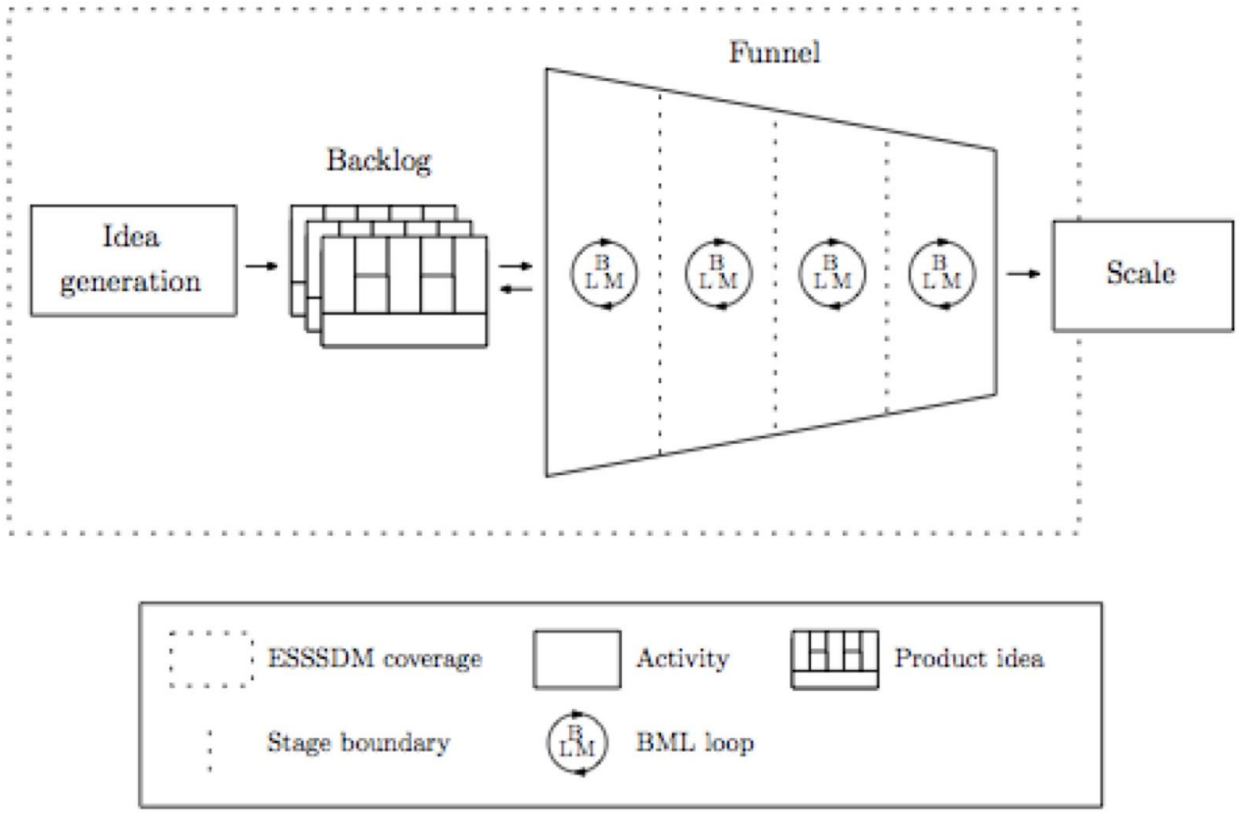


Figure: An instance of an innovation funnel

The problem that most companies and individuals run into is that it is really hard work to leave the safe, comfortable office, find potential customers, interview them, figure out that the assumptions that we had are completely wrong, going back to the drawing board to iterate and figuring out that most of the “great” ideas turned out to be complete duds. It is much more convenient to stay within the ivory tower of the innovation lab, work on technical solutions and give demos to the senior executives to show how innovative we all are.

As Thomas Edison already said, many moons ago, opportunity is missed by most people because it is dressed in overalls and looks like work. Innovation is hard work and requires you to go out, expose yourself and your ideas, receive unsalted feedback, get your darlings killed, even after the initial feedback was really positive, but when the actual behaviour of customers does not match what they told you.

No matter what innovation method, approach, process or silver bullet you subscribe to, realize that innovation does not just happen by putting some infrastructure in place. Your organization needs a culture where everyone constantly test and validate their assumptions, use data instead of opinions, minimize the investment between validation points and iterate as fast as possible to learn faster than the competition. A culture where failure is not defined as a potential innovation not delivering on the expected outcome, but one where failure means not have

explored the concept at all. Where surprises, often ones that you would rather have avoided, are celebrated as these are the ones where the biggest learnings happen. But also a culture when everyone realizes that innovation means revenue and growth, even though it is really, really hard work.

Reference

Bosch, Jan; Olsson, Helena Holmström; Björk, Jens; Ljungblad, Jens; The early stage software startup development model: A framework for operationalizing lean principles in software startups, Lean Enterprise Software and Systems, pp. 1-15, 2013, Springer Berlin.

The End of Innovation (As We Know It)

If there is one buzzword that is getting a lot attention in the circles I operate in, it is INNOVATION. As I spent another week on the road spending time with different companies, I realized that I probably haven't had a meeting at a single company over the last year where the "I" word was not used. Almost every company I meet has some innovation initiative, an innovation garage, a lighthouse project, some special team working on something disruptive, a co-creation initiative, some open innovation activity, etc.

Unfortunately, when I ask about the results of these initiatives, there often is a lot of fluff and very little in terms of real and tangible results that materially drive the bottom line for the company. Instead, these innovation initiatives seem to be more directed towards the needs to the C-suite team and their desire to show the board that they're doing everything they can to secure a future for the company. I guess most of the C-suite executives like their jobs and spending some company resources on creating the right impression with the board as well as the customer base may well be the right decision. But you don't build new revenue streams this way.

Cynicism aside, the fact remains that driving successful innovation activities that lead to material outcomes is hard, as many startups can attest to. And in many ways it's even harder, and in my experience very ineffective, in large companies. When I reflected on why this is the case, I identified four reasons that are at the root of this challenge.

First, most companies are organized functionally. This means that the people who meet and understand the customer (product management, sales and marketing) may have all kinds of novel insights and innovative ideas. However, because of their background and skill set, they are unable to build anything like a prototype or mock-up. Similarly, the folks that would be able to build prototypes and innovative solutions, such as the staff in the R&D department, are typically shielded from the customer. So, they don't get these novel and innovative insights and consequently don't understand the customer. Even in the cases where the Kiplingesque divide between "business" and "R&D" is crossed, as the two sides typically don't form a single innovation team, the result is very long feedback loops that kill innovation.

The second main challenge is that affects innovation is that many companies pride themselves on predictability and reliable return on investment. For sustaining innovations, such as the constant flow of new features and products in already established business areas for the company, the return on investment is calculated beforehand and the results typically follow a Gaussian bell curve in terms of predictable outcome. However, for innovations that are intended to build new business opportunities and revenue sources for the company, the outcome tends to follow a power curve. This means that most innovations fail and are waste, but the few that are successful make up for all the failed ones. I once heard a VC explain that his most successful investment had returned more than all his other investments combined. And his second most

successful investment had returned more than all others except the first one. And so on. The power curve of return, however, meshes very poorly with almost every process in companies as these are all based on predictability and high likelihood of successful outcomes. Failing to meet a successful outcome is inherently considered a failure and everyone shies away from that. Hence we see that innovative ideas that could be disruptive are scaled back and the energy is focused on the initiatives that have the highest likelihood of success, and consequently the lowest risk and limited returns.

Third, because of the aforementioned, companies seek to kill off innovative ideas as early as possible to minimize wasteful investments and typically before an idea has had a chance to prove itself. And in many companies, decisions concerning innovative ideas are taken by senior leaders based on their opinions, rather than based on actual data from customers. As innovations are by definition breaking the rules that are established in a specific industry and senior leaders have typically grown up in that industry and implicitly believe that those rules are the only ones that could possibly exist, decisions concerning innovative ideas are made by the worst possible individuals.

Fourth, in many organizations, different units, functions and teams have established a certain turf for which they, and they only, have responsibility and the right to operate in. Unfortunately, innovative ideas tend to have a lack of respect for organizational boundaries. I have experienced several cases where innovation initiatives were slowed down or killed altogether because of the organizational structures that existed in the company. The structure of the organization can easily become the very thing that makes you fail.

Finally, as I shared in earlier blog posts (such as here), in a typical company, 80-90% of all staff in R&D (including product management and other functions related to R&D) work on commodity functionality. However, for a variety of reasons, these people believe that the functionality that they work on is differentiating or even innovative. Consequently, many companies tend to innovate in areas that customers don't care about, but that are considered very important within the company. These shadow beliefs tend to lead to a lot of waste in innovation efforts.

These are five reasons why innovation at large companies almost always fails. People don't talk about it and only sing the praises of the few things that do work out, but the fact is that innovation processes at most large companies that I meet are very ineffective and wasteful. Rather than claiming that I have the silver bullet, I do want to share what I have found to be the five characteristics of successful innovation systems at companies.

First, innovation takes unstructured time. Employees need time not managed by bosses or other individuals, but unstructured and entirely up to the individual on how to spend it. Google famously gives their employees 20% of their time, but during my time at Intuit (a great, great company to work for, BTW), every employee could spend 10% of his or her time on innovation initiatives.

Second, innovation needs teams. Very few individuals can take an innovative idea and go through the entire lifecycle of the idea without help from others. Consequently, the organization needs to establish tools and mechanisms to help employees build teams across functions and sites. In that way, individuals passionate about the same innovation idea can band together, evolve the original idea, build mock-ups and prototypes, collect feedback from customers and build the case for a larger investment from the company.

Third, innovation needs customer feedback. Most companies are extremely careful about who gets to interact with customers. However, innovation requires almost constant customer feedback and teams need to be able to connect with customers throughout the innovation lifecycle to learn, collect feedback and test alternatives.

Fourth, innovation needs funding. Once an innovation team, using their unstructured time, has developed a promising concept, has collected positive feedback from customers and has tested as many of the prerequisites required for the innovation to scale, the company should have mechanisms to provide the funding needed to allow all or part of the team to work full time on their innovation.

Fifth, and finally, innovation needs scale. As I discussed earlier, innovation follows a power function in terms of success, meaning that most ideas fail but the few that succeed can materially move the company. This means that we should aim for dozens or even hundreds of ideas to move through the innovation funnel in order to get to the very few that knock the ball out of the park.

Concluding, innovation as it is practiced at most companies is very inefficient at best and often counterproductive. We need new innovation systems and in this post, I highlight the five characteristics, i.e. unstructured time, teams, customer feedback, funding and scale, that I consider to be necessary ingredients. Please share your experiences and insights in the comments!

When Did You Last Talk To A Customer?

Several of the companies that I work with have change initiatives ongoing. Some of these initiatives are more technology oriented, others affect the business strategy and yet others seek to reposition the company in the ecosystem it is part of. However, all these initiatives have one thing in common: customers will be affected by the changes that are under consideration.

Any change initiative, innovation initiative or cost reduction initiative is based on so called “leaps of faith”, i.e. the beliefs that underlie and justify the investment in the initiative by the company. My concern is that many companies spend vast amounts of resources on building some product or preparing some change in the engagement with customers before actually testing with these stakeholders whether this is something that they support and are willing to engage on. The leaps of faith are gargantuan jumps with a high likelihood of being wrong.

Whenever I bring up my concern and the suggestion that it might be helpful to get early feedback on the concept under development, I get a variety of responses that never fail to surprise me. These are some of the main ones:

- We know what the customer wants
- We don’t want to disrupt our current relationship with our customer
- Our brand will be damaged or affected negatively if we show unfinished products or solutions
- Sales does not allow anyone to talk to customers
- My boss is afraid that I make promises the company can not keep when I talk to customers
- My solution is not yet good enough to show to customers
- We need the following X more features and then we can show it to customers
- Our customers don’t want to talk to us
- This solution targets a new market and we don’t have any customers yet

The responses basically fall into three main categories: beliefs about the customer, internal customer engagement processes and beliefs about the quality of the solution. Let’s examine each of these categories in more detail.

Beliefs about the customer: If you feel you know your customer really well, I will challenge you: whatever you’re proposing, the reaction of customers is going to surprise you. If you think your customer doesn’t want to talk to you or think less of you for discussing early stage ideas or concepts that are under development, you have a much bigger problem: you’ve reached the point where the customer views you as an interchangeable supplier and you’ll lose the customer the moment a competitor comes along with a better value proposition. Finally, if you are targeting a new market and customer base, you are flying blind as you have no idea whether the concept resonates with these new customers. Obviously, it’s even more important in this

case to learn what is important to build empathy and understand what resonates with customers.

Internal customer engagement processes: In my experience, this comes down to two patterns. First, the sales and marketing organization is afraid that others from the company interfering with the ongoing relationship will affect future sales. As their bonus is directly connected to sales targets, the folks in sales and marketing will fight tooth to nail to resist others upsetting their carefully orchestrated engagement process, no matter for what reason. Second, for a variety of reasons, senior leaders are extremely concerned about the brand of the company and the perception of the company by customers and prefer to reduce the risk by minimizing interactions between employees and customers. Clearly these leaders do not trust their own employees to act with sufficient common sense to avoid upsetting the customer.

Quality of the solution: Especially in organizations that have a long history of delivering products and solutions to customers, everyone, from product managers to engineers, has a very high bar of what an acceptable product looks like. Consequently, for new products and solutions, the tendency is to wait until the quality and feature richness of the new solution is on par with mature products. In addition, the team is often concerned about the customer reaction and worries that it might be negative. It's then easier to avoid this risk by delaying a little further.

The commonality between these three categories is that in every case, the right behavior is exactly the opposite of what happens in many of the companies that I work with. Everyone (or at least almost everyone) should talk to customers on a very regular basis. The more uncertainty exists around customer preferences and product functionality, the earlier and more frequent customer interactions should take place. And if you're solving a problem that the customer cares about, he or she will be happy to spend time to talk to you and to help you solve it in a way that is best for his or her company. Obviously, the inverse is true as well: if you're working on something the customer doesn't care about, it doesn't matter how glossy or feature rich the solution is. The customer won't touch it and doesn't want to talk to you.

Software-intensive companies and the industry at large are undergoing several major transformations. Digitalization, servitization, big data, artificial intelligence and ecosystems are only some of the topics modern companies struggle with. As a consequence, business-as-usual and the associated comforting, repeatable processes are rapidly disappearing.

Instead, we live in a stage of constant uncertainty. In this state, close and frequent interaction with customers is absolutely critical. The principles for startup companies as preached by the likes of Steve Blank and Eric Riess now apply to all companies. Because we're all back to running startups and this means we have to start acting like entrepreneurs. And entrepreneurs focus on what the customer wants, act on their best understanding, constantly validate their beliefs with customers, learn and adjust and find ways to deliver value so that they can capture some of that value as revenue.

So, after reading this article, ask yourself this: when did you last talk to a customer? And what can you do to talk to one today? Start acting like an entrepreneur because soon, that will be the only role that matters.

Why Failure Is The Only Option

During the last weeks, I've spent time with or close to the innovation units of otherwise traditional and conventional companies. As I wrote in another post earlier this year, it seems that every company has some kind of innovation initiative ongoing. As I had a bit of time to discuss with the people responsible for the incubators and innovation teams, it became that, though they were hiding it well, everyone was very concerned that the innovation team may fail or that the incubator or garage will not result in any success for the company.

It made me reflect on the reasons why everyone so afraid to engage in initiatives that may have a less than 100% success rate. Of course, this has its roots in the traditional ways in which companies operate. For mature product lines, any investment in sustaining innovations or cost-down initiatives is expected to have a guaranteed return that typically follows a Gaussian distribution in terms of the actual outcome. And those that manage to consistently deliver on this or even manage to exceed expectations are rewarded with promotions, bonuses and intangible benefits by the company.

The challenge with innovation initiatives is that the returns follow a power curve: most innovations will not provide any return but the few that do will pay for the investment in all innovation initiatives combined. The concerns and fears of the people that I met during the last weeks are based on the mismatch between the implicit expectation of a guaranteed result, originating from sustaining innovation efforts, and the reality of radical innovation initiatives, where the majority of ideas will simply not pan out.

This fundamental conflict in perception is detrimental to innovation as it causes one to hedge the bets. Rather than focusing on high risk/high reward innovations, teams will focus on low risk/low reward innovations as an innovation that does not deliver on expectations automatically is viewed as a failure of the team. A consequence is that teams will often look to delay taking their innovation to market as they, as long as there is no negative feedback from the market, they haven't failed yet. Another consequence is that team as well as the organization as a whole fails to learn from the market and from the things it tries out with customers.

Although I have written about innovation before, I want to stress the importance of so-called failure in innovation. Innovative ideas by necessity break the rules and the consequences of that are very difficult to predict. The inability of the team to predict a certain outcome from an experiment is not because they're a bunch of ignorati. For truly innovative ideas, the result of experiments testing the idea in market is fundamentally unknowable. Being a complex system, innovation is similar to the butterfly - hurricane effect. There is not necessarily a direct and proportional relation between the trigger and the effect.

Although many books have been filled on innovation, in my experience with various companies there are at least three elements that are critical:

- **Wide, rapidly narrowing funnel**: To find the few “killer app” ideas, we need many ideas to enter the innovation funnel. Rather than focusing all the organizational energy on the top three ideas, we need dozens, if not hundreds, of ideas early in the funnel. However, before an idea can get further in the funnel it needs to collect the first evidence on its value and relevance. Ideas that fail this test are removed from the process. This early culling of ideas is not failure but a necessary element of the process.
- **Selection based on customer feedback** : It is critical that the feedback on innovation concepts comes from customers and is not driven by internal opinions. Many of the companies I meet have innovation boards populated with senior managers and the live and death of innovative concepts is based on their opinion, rather than market and customer data. In hierarchical organizations, many have honed the skill to convince managers and key decision makers based on masterfully crafted powerpoint presentations and storytelling. Your selection process has to actively discourage this type of behaviour and focus on feedback from real customers.
- **Learning, not revenue**: Finally, the focus, especially in the early stages, needs to be on learning as much as as fast as possible. Although testing for monetizability as part of the innovation process is perfectly, i.e. are customers willing to pay, early on revenue should not be one of the targets. Quoting the title of one of the well-known books on innovation: Nail it; then scale it. First make sure that you’ve explored the entire design space around an innovation concept and have ensured maximum traction. Only then invest in scaling and growing of revenue.

Concluding, innovation concept that do not turn out as expected are a normal and necessary part of the innovation process. Building a culture where the team isn’t viewed as having failed, but rather as a team that learned a lot in the process. The companies that are most at risk of disruption are those where all innovation efforts are successful and generate revenue. These companies are staying in their comfort zone and fail to reinvent themselves. So, as an executive or any other type of leader, remember this: failure is the only option. When you last fail and what did you learn from it?

Does Agile Kill Innovation?

During a recent conversation with a journalist, the downsides of agile came up in the interview. The questions were centered around stress levels of team members, the frustration with not being able to do a proper design before building features, the perceived reduction in innovation and other factors. During the discussion I realized that there still are many people that seem to have missed an important aspect of agile: the transformation of tasks from being implicitly performed to explicitly scheduled.

When companies employ more traditional, waterfall style approaches, there typically is a certain amount of slack in the weeks after a release. Of course, there are issues from the field that need to be resolved and some are busy with that, but a significant number of people has time for activities that we didn't get around to when pushing for the release. These activities include managing technical debt, experimenting with new, innovative ways of doing things in the system, becoming familiar with potentially relevant new technologies, trying out new product or feature ideas, etc.

The interesting thing is that these activities occur organically and provide a valuable source for innovation, both customer- and technology-driven, as well as a natural way to manage maintainability of the system by paying off technical debt. However, as these activities occur organically, are implicit and consequently not actively managed, the company learns to rely on these activities taking place.

When adopting agile, the sprint structure causes a situation where teams are constantly driving for feature development. Well functioning teams want to do right by the company, which means that if teams are asked to build features prioritized in a seemingly infinite backlog then teams will do so. The process of sprint planning provides a much higher degree of transparency in resource allocation and, as a consequence, there is no slack in the system.

The consequence of the lack of slack is that the company needs to explicitly allocate resources to activities that used to take place implicitly and organically before the adoption of agile. This means that if you want to have a maintainable system over time, resources need to be explicitly allocated to managing technical debt. If the company is relying on a constant flow of bottom-up innovations from its R&D staff, then time for innovation activities needs to be explicitly scheduled.

Once leaders in the company realize the need to explicitly schedule time for results that used to come "for free", this typically leads to a situation where the relationship between R&D and the rest of the company is made explicit. In those organizations that view R&D just as a supplier from which you order features, the willingness to provide time for managing technical debt or for innovation is typically non-existent. In those companies where there is a true partnership between R&D and the rest of the company, this leads a situation where explicit resourcing

decisions are taken not just concerning feature and product development, but also around quality, technical debt and innovation activities.

Concluding, agile does not kill innovation, but as with many other aspects of software intensive businesses, agile makes the problems so blatantly visible and explicit that the organization has no choice but to address these concerns. There is no such thing as a free lunch! Quality, maintainability and innovation all require resources and not allocating these resources will give you exactly what you asked for.

Personal Development

It is very easy to point at the company, the business ecosystem and customers and complain about everything that is wrong with the world. However, as a Dutch saying claims: whenever you point at someone, four fingers are pointing at yourself. In many contexts, it is actually the maturity of individuals and teams that limit the development of the company, the ability of the organization to go through the digital transformation and to reinvent itself. Consequently, one of the themes for a successful transformation has to be the personal development of the individuals involved.

Most people have a dual relationship with the notion of personal development. On the one hand, the notion of developing oneself and becoming “better” is of course a positive thing in and of itself. On the other hand, development requires changes to mental models, habits and behaviours which typically is diffcult and unpleasant for the people undergoing the change, even if it is of their own volition.

The main driver for personal development typically is that the pain of non-chaning is more significant than the pain of changing. The worst situation is where people are so stuck in their current situation that they suffer through rather than make the necessary changes. This notion of learned selfhelplessness is of course incredibly limiting but still not uncommon.

In this part of the book, there are some reflections on the use of introspection and reflection in order to develop an accurate view of ones current situation and the benefits and challenges. Once there is understanding, the next step is of course the formulation of suitable actions to address the challenges. Finally, even if actions are defined, the next step is then concerned with actually executing on the actions.

A final item concerning personal development is the notion of a purpose. As the saying goes, the purpose of life is to live a life of purpose. So, the first question one has to ask is what your purpose entails and how your work and actions align with those values. Especially this later one is a very big and difficult topic that many people shy away from thinking about. In this part of the book, you are encouraged to think about it anyway.

Why You Will Ignore Opportunity Cost in 2018 Too

As the saying goes, change is the only constant. All the companies that I work with want the change, either because of fear of being disrupted or because of the opportunities that disrupting their industry bring. This is at least what companies and their employees tell me. In practice, however, I run into situation after situation where people are trying to slow change down. Many try to keep today as it has always been with the promise of changing tomorrow.

Psychology, of course, provides explanations of why this is the case: humans, on average, fear losing something they have today about three times as much as they appreciate gaining something. So, as a company, losing out on today's revenue based on historical business models because of some high-risk experiment that would inform the future and perhaps create new revenue opportunities is very difficult to realize. In practice, we give up "guaranteed" revenue now for the promise of new revenue in the future.

Once you take that perspective, you see examples of this problem everywhere. Especially in regulatory frameworks and certification processes, the current state is only allowed to change very slowly and after very deliberate and extensive evaluation of the intended change. The argument always is concerned with avoiding the risks that come with implementing changes. Three quick examples:

- In automotive, many express concern about the risks of self-driving, autonomous cars. Everyone is scared of an autonomous car causing some horrific accident because of lack of testing and improvement. However, interestingly very few people are concerned with the traffic deaths caused by human drivers. According to wikipedia, in 2016 alone close to 40 thousand people died in traffic accidents, just in the US. Assuming that autonomous cars can reduce this number with 75% (a conservative estimate, I believe), delaying the introduction of autonomous cars with three years because of conservative attitudes in OEMs and certification institutes will result in almost 100 thousand unnecessary deaths (again, only the US!).
- In the medical space, certification institutes like the FDA are notoriously slow in approving new medicines and treatments. It can literally take decades for approval processes to be completed. And I know that in medicine, there is a lot of controversy around the data generated by medical trials and there are many in the industry that view patients and their insurance companies from the perspective of revenue maximization. However, I am also convinced that the vast majority of people working in healthcare do so from a genuine desire to do good. Similar to autonomous cars, I can't help wonder how many people have died unnecessarily because treatments being approved too late to help them. And how does this compare to people that have died because of novel treatments that proved to have severe negative side effects? Unless I have data to the contrary, I believe that the number of unnecessary deaths due to slow approval processes outweighs the number of people that died because of risky treatments with orders of magnitude. And let's not forget that only in the US, every year more than

400.000 people die because of preventable medical errors. And this is an industry that has been notoriously slow and conservative in adopting new technology and approaches.

- In all industries building safety-critical systems, safety certification is a process that is entirely based on waterfall processes. You finish the development of a new system, establish safety through a combination of testing and hazard analysis, convince the certification institute that you've done a good job, get the stamp of approval and start manufacturing. The problem is that any change to the system triggers, in principle, a complete redo of the safety assessment. The problem is that that the cost of recertifying is so large that companies leave known safety hazards in the system. In general, many other areas of improvement are ignored. That means that in automotive, medical, aeronautics, trains and shipping as well as many other industries, the systems are performing suboptimally and causing preventable damage and deaths due to slow, expensive certification processes.

Although these examples are concerned with certification and safety-critical systems, it is the underlying mindset that I care about. Whenever confronted with risk, humans have a tendency to glorify the current state and exaggerate the risks associated with change. We have a tendency to ignore the opportunity cost of our inaction.

In industry, some companies have realized this problem and put systems in place to counterbalance this basic human tendency. For instance, combining the 70/20/10 innovation model with the three horizons model offers a system to ensure that new innovations receive sufficient funding. The three horizons model, as the name indicates, categorizes the businesses that the company is concerned with into three buckets. Horizon 1 are the businesses that are large, successful and the cash cows of the company. Horizon 2 businesses are proven, but still smaller and growing rapidly. Horizon 3 businesses are experiments that may become future horizon 2 and horizon 1 businesses. The 70/20/10 model then allocates resources to the three horizons as indicated: 70% of resources are allocated to horizon 1 businesses, 20% to horizon 2 and 10% to horizon 3. That system ensures that companies invest enough in creating new businesses that, typically, capitalize on the new opportunities that the organization otherwise might have ignored.

The 70/20/10 model is a mechanism explicitly intended to avoid falling into the opportunity cost trap, but of course it is rigid and not concerned with context and opportunity per se. It just focuses on effort allocation and as I wrote about in earlier blog posts (like here), it may easily result in low risk/low reward innovations receiving the 10% of resources.

As individuals, we need to build systems for ourselves as well to ensure that we don't fall into the opportunity cost trap. Although it's tempting to focus on willpower as a way to ensure that we proactively embrace what we experience as a risky future, most of us have learned that willpower is a limited resource that you run out of rapidly. Instead, building systems that, over time, rely on our habits is a much better mechanism to avoid the opportunity cost trap.

Concretely, there are at least three mechanisms that I have used with, I think, some success:

- **Remember that you're going to die**: In his 2005 commencement speech at Stanford University, Steve Jobs reminds us that our time on this planet is limited and that we need to use it to live our own lives to the fullest. In his words: "You are already naked; there is no reason not to follow your heart."
- **Imagine the worst that can happen**: The Stoics employ a daily practice where you are asked to imagine the worst scenarios that could happen to you. This prepares you for the loss that undoubtedly will happen. Using this practice for the changes that you're looking to make in your life helps you visualize the fears associated with the change. And by making them more concrete and assessing how difficult it would be to recover from these worst case scenarios, you can get a handle on your instinctive fear of change. In his TED talk, Tim Ferris introduces the notion of fear setting as a tool to manage irrational fears of change.
- **Visualize your life in a decade**: Finally, try to visualize what you want your life to look like 10 years from now. Once you have established this vision, the next step is to analyse what you need to now, in a quarter and in a year to bring that future about. This can help overcome the emotional resistance by using the rational mind.

Concluding, it's the start of 2018 and we have another fantastic year ahead of us. Rather than staying stuck in our day to day motions, I challenge all of you to follow your dreams and become cognizant of the opportunity cost associated with inaction. Both companies and individuals suffer from ignoring opportunity cost. I believe that making opportunity cost explicit and quantifying what the risks and costs of not changing will make you take the right action. And that, in the end, will lead to a better you, better companies and a better society. God forsättning!

Why Your Job Will Also Suck in 2018

While I am blessed to work with many people that really enjoy their jobs, I still meet way too many people in positions that they are really not enjoying anymore. If they ever did enjoy their job. For a long time, I thought that these folks were just exceptions and that most enjoyed their jobs. However, when looking deeper into the question, I started to see a different picture appear.

According to Gallup (see here for link), worldwide only 13% percent of people are engaged in their work. The rest is not engaged (the majority) or actively disengaged, meaning that they really hate their job. The study is a few years old (2012), but in more recent articles, it seems the numbers haven't changed much.

Even in the Nordic countries, often considered to be outstanding from a "workers" perspective by other regions in the world, the numbers are dismal with engagements levels of 16% for Sweden, 21% for Denmark, 16% for Norway and 11% for Finland. How can it be that countries that are consistently ranked as the best places in the world to live, build companies and raise families (which I can personally attest to be being the case, having lived in Sweden, in total, for more than 12 years), still see such low numbers in employee engagement?

After having thought about this for quite a bit, my current belief is that, for the vast majority of people, this is a lack of taking responsibility for your own life. It is very easy to blame the government, industrial-financial complex or some other vague, abstract concept for your situation. Even more typical, it is easy to blame the company or organization you work for. However, it is important to remember that there is only one person responsible for your happiness, satisfaction and contentment: YOU!

Of course, there is a whole misinformed culture of "follow your passion". I say misinformed, because the blind belief that just by being passionate about something will make sure things will turn out fine is plain naive and stupid. I am convinced that at least some of the many non-engaged and actively disengaged folks are those who tried to follow their passion and got smacked in the face with the results.

Instead, the best path forward is to build a rare, valuable skill that you care enough about. Once you build your skill, you will notice that you actually become increasingly passionate about the skill and associated topics. As you chose a valuable skill, you will be able to monetize it and increase your income over time. However, don't spend this newfound financial stability on buying things, mortgages, expensive cars and other external validation as you'll end up wearing golden handcuffs and you'll end up in financial trouble if for whatever reason your life is disrupted. Instead, to use your increasing financial stability to continuously increase your autonomy. If you're valuable, your employer will be much more willing to let you work part-time and/or remotely. It will be easier to become an independent consultant and avoid the office

politics and the boss you hate so much. Perhaps you want to take sabbaticals or mini-retirements? All this is feasible, but only if you have something rare and valuable to offer to the world.

2018 is looking to become a totally amazing year. Are you going to stay in this job you think sucks, using the excuses that you've always used of why you are stuck there? Or are you taking control of your own destiny and start working your way towards gaining control and autonomy? It's really scare to do this, because you might fail. In fact, you're likely to fail many times before you have found the path towards a meaningful life that gives you the control, autonomy and even passion for your life that you are looking for. But what is the alternative? Staying where you are and some day figuring out that your life is over without you never even having lived? Choosing the meaning of your life and then actively working towards it is critical for a living good life. Remember that it's all up to you and that blaming anyone else is just a cowardly excuse. Take charge already!

To All You Unsung Heroes

The coming week, Gothenburg hosts the Internal Conference on Software Engineering (ICSE 2018) where close to 2000 people come together. That same week also the Lindholmen Software Development Day is organized that brings together around 700 people interested in software, data, digitalization, artificial intelligence and related topics. At these events, there is of course a host of presentations by high-profile people in the various communities and many are in awe of the successful and respected “special snowflakes” that will be present at these events. We all like to be inspired by the ideas, viewpoints and success stories from other companies and industries.

The last weeks, I spent time at several of the companies that I work with and the pattern that I noticed is that, despite the pervasive frustration about the speed at which the changes that we’re looking for are realized, there is progress in all these companies. This progress often comes hard-won and while facing all kinds of incredible resistance from all parts of the company. But still the teams I work with are slowly, steadily and stubbornly “pushing the boulder up the hill”, accepting that it sometimes rolls back a bit and never losing faith that the top will be reached from where the change will become easy because the entire organization has bought in and now starts to act in accordance to the, now obvious, idea and concept.

During the workshops, I was reminded of a quote attributed to Thomas Edison that innovation (and changing processes, ways of working, architectures and tooling is innovation too) is 1% inspiration and 99% perspiration. It’s not about having ideas; it’s about making ideas reality. That’s where the real progress to your team, your company and your community is accomplished.

So, this blogpost for all those people in organizations around the world that take on these challenging change initiatives against all odds. Who experience the scorn of their peers, receive negative feedback from their managers, advice from their friends to stop fighting. Who have been side-tracked, passed over for promotion or even fired because they kept driving the change that they wholeheartedly believed in. Not to gain the recognition of their peers, not to become rich in the process, but for themselves. Because when you believe you’re doing something that is right, doing it even when it’s hard is what you need to do in order to be able to look yourself in the mirror and know that you did what you believed in.

Real, tangible progress is not created at conferences, during presentations or in workshops run by consultants or academics. The real progress that matters is created by people who day in, day out keep straining against the harness provided by the process, incentives and cultures of companies. Those who manage to deliver on today’s needs while securing a future for the company, often against the will of those in company only looking at today’s needs. It is all those individuals and teams who are the real heroes. Keep fighting! I am rooting for you!

Stop Thinking It's Not Your Problem

Engineers form the core of a product company. Although I have the greatest respect for sales people and know how hard a job that can be, it's the engineers that design and develop the product that in the end is needed to have anything to sell.

Building a product that sells of course requires that customers actually want it and are willing to pay for. Although this may seem obvious, this is where the problem starts in many companies as the process of figuring out what customers want as well as the process of monetizing the product and the associated business model tends to get separated over multiple functions and roles in the organization. Among others, we have sales & marketing, product management, engineering and user experience/usability experts. Each of these functions and roles owns part of the process of building and selling the product or service.

In most companies that I work with, the responsibilities between the roles has resulted in engineers basically asking for requirements and declaring success when they've delivered a solution that passes all test cases and satisfies the various edge cases that might exist. Although this attitude is considered to be the right one in most engineering cultures, it has several problems associated with it.

The first problem is that many engineers build whatever feature they are asked to build in the same way: basically a solid implementation that covers all exceptional cases, minimizes interactions with other features and deals with those where necessary. In our research, however, we see that companies build at least four different types of features. As shown in the figure below, we recognize duty, checkbox, wow and flow features. Each of these features requires a different implementation as well as a different level of coverage. A "wow" feature should show one use-case really well so that marketing and sales can use it, but doesn't have to cover all alternative use cases. A checkbox or duty feature should be built to just enough to satisfy the regulator or to allow the company to claim that they have feature parity. It's only "flow" features that require a full "gold-plated" realization and the best possible user experience. Engineers not understanding the difference between the categories of features tend to waste a significant amount of resources for the company.

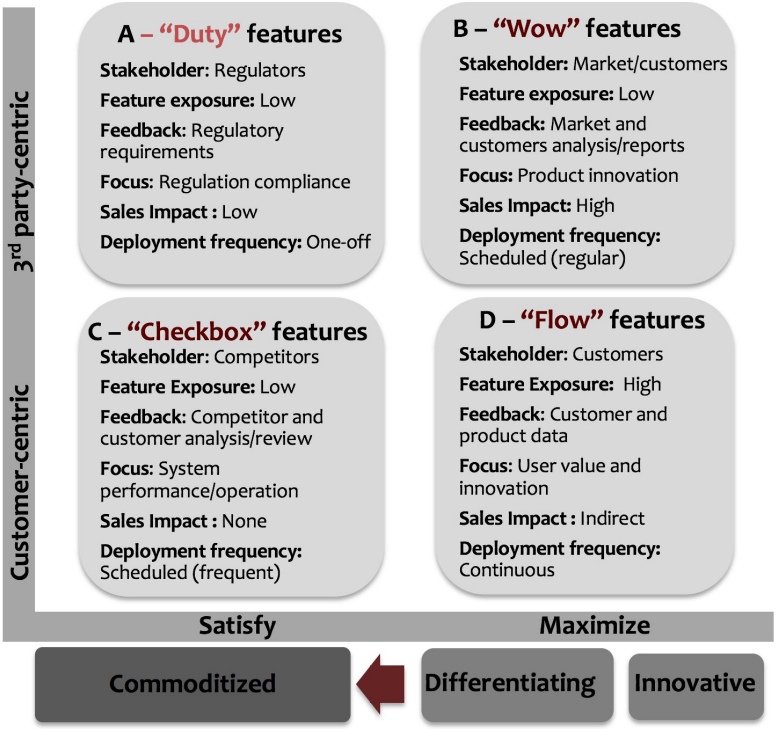


Figure: Feature Types model

The second challenge is that engineers run into is picking items from the backlog that do not actually generate revenue for the company. In a company that I visited recently, engineers tended to focus on improving a mature product for which the customers paid a yearly maintenance fee. The company was also building new products to offer to existing and new customers, but the R&D organization complained about lack of capacity. In practice, however, engineers tended to build features that did not generate any additional revenue for the company and ignored spending time on the new products that would generate additional revenue. This lack of understanding of the business that the company is in leads to significant lost revenue and opportunity cost.

The third challenge that I see several R&D teams end up in is a highly limited and distorted understanding of what customer success looks like. Especially when the agreed division of work is one where product management throws requirements and specifications over the wall and R&D teams just build to spec, the realization of functionality tends to leave a lot to be desired from the user perspective. Customer empathy is incredibly important when building a product that delivers on customer needs and expectations, but I still meet many teams that have not realized this yet.

Concluding, my dear, esteemed engineers, business is too important to leave to the business folks. You have to get involved and deeply understand what business we're in, how revenue is generated, what matters for customers and what sales needs to convince customers to go into business with us. You can't ignore this and think it's not your problem. Especially in a world that

is changing ever faster, engineers need to be in tune with the market and act accordingly. It is your job, after all!

Do You Have Skin In The Game?

Over the last month or so, I have been reading a book titled “Skin in the game” by Taleb. He doesn’t like academics very much and scientists get roasted quite severely in several chapters. Being a professor myself, that stings a little but the problem is that I agree with him. His main point, severely abbreviated, is that there are two groups of people. Those that operate with skin in the game and those that don’t. Operating with skin in the game means that your decisions and prioritizations have real implications in the world and affect you, potentially causing you harm of some kind, e.g. financially, physically, etc.

Those operating without skin in the game can take all kinds of decisions, make recommendations, propose theories and frameworks, but the consequences of these decisions, recommendations and models do not affect them but rather others get hurt. For those operating without skin in the game, the validation of what they do is not measured in real-world impact, but by the opinion of their peers. Of course, academia is the prototypical “lack of skin in the game” industry in that literally everything is based on evaluation by peers. Journal and conference papers are peer reviewed. Promotions are decided by evaluation committees. University rankings are based on the opinions by other academics at other universities. Etcetera.

Although I am far from pleased about the state of practice in academia, I did realize that the lack of skin in the game is by no means limited to academia. We see the same in politics where the persons introducing laws and regulations seldomly experience the consequences themselves. We see it in government functions where many hide behind an overly conservative interpretation of rules and regulations in order to avoid bringing any skin of themselves into the game. The “big five consultancies” advising companies on strategies that cause these companies to be caught completely anware of and unprepared for market and technology disruptions.

However, we also see it in industry. Most companies I work with have many decision processes where the individuals making the decisions are in no way affected by the consequences of their decisions. C-suite executives putting new financial targets in place that hurt the most productive and value adding teams. Product managers prioritizing features and functionality that never gets used after deployment. Systems engineers taking decisions that cause the company enormous maintenance and software development cost.

The main reason for this situation is twofold. First, the feedback loop between the decision being taken and the consequences of the decision becoming apparent is often very long. By the time the consequences became obvious, the original decision maker has moved on or has a convincing story as to why the decision was deemed the right one. Second, when the negative consequences do appear, there is no negative consequence for the decision maker. A financial planner making decisions for his or her customer, causing major harm to the portfolio of the customer, will still charge the fees. C-suite executives mismanaging the company and causing

bankrupty still got paid during their tenure and do not experience (in the vast majority of cases) any negative consequences.

In most companies that I work with, the employees do not operate in the real world, but rather operate in a “virtual” reality that is largely separated from the real world and where actions are judged and planned based on the perception that this will create with peers and bosses. As the old adage says, in management, perception is reality. Ensure you manage the perception and you’re golden.

As a nice anecdote concerning how to bring skin in the game that might otherwise be missing: in some of the old greek city states, if someone accused a peer of some crime and the accused was found not guilty, the accuser would receive the punishment that would otherwise have gone to the accused. That approach most certainly takes care of all the frivolous lawsuits that exist in some countries. I am not recommending capital punishment for anyone who makes a mistake here, but use this as an example of the fact that there are mechanisms that can be employed to ensure that people have skin in games where it otherwise might be missing.

Concluding, I feel we all have a responsibility to operate as much as possible with skin in the game. To take responsibility for our actions and decisions. To build systems where those that are empowered to make decisions are also experiencing the consequences of these decisions, also over time. So, as you read this, reflect on the following question: Am I living in the real world with skin in the game or am I living in a “make believe” world driven by the opinions of my peers?

Why Do You Do What You Do?

The last days, I had some time to reflect and one of the things that consistently surfaces is the amount of irritation and frustration that many people experience in their jobs. Now, to be clear, the people that I work with and that I meet are no complainypants at all, but generally serious, hard-working people that are looking to make a difference. And yet, lots of frustration and stress.

The key factor differentiating the companies where I meet the more severe cases, though, is concerned with purpose. The companies that have little purpose beyond earning some money seem to be the most afflicted with internal tension, strife and politics whereas companies that have a very clear sense of purpose, though far from perfect, seem to be doing much better. Paraphrasing the former CEO of Whole Foods: We need to make money in the same way that our bodies needs to make red blood cells; we need red blood cells to live, but the purpose of our lives is not to make red blood cells.

Interestingly, in several of the startups that I am involved in, there is a clear sense of purpose and an openly expressed reason for the existence of the company. The employees of these young companies rally around this purpose and often the primary reason for being at the startup often is a desire to make a positive impact in the world through the activities of the company. For larger, more established companies, the sense of purpose tends to be subdued or even missing altogether. And even if there is a mission and set of values on the corporate website, it fails to be connected to the day to day work and interactions in the company.

At the individual level, I often experience a similar confusion or lack of purpose for many people that I meet. When it’s not clear what the purpose of your professional life is, then other, extrinsic factors take over like winning the competition with a colleague or competitor, the job title, size of the pay package and associated leasing car, etc. The problem, as we all know, is that none of these things give a lasting sense of fulfillment and accomplishment.

Now, you may ask, what is my purpose then? For me, my starting point is that I am a hardcore modernist and I believe that technology and its use in society to the largest extent has positive implications for mankind. At this point, digitalization, which I define as “software, data and AI” is the most important technological transformation. Hence, my professional purpose is to help the software-intensive industry to accelerate their digital transformation in order to make the benefits available to society as a whole. This is why I run Software Center, consult with dozens of companies and am involved in several startups. Now, this purpose undoubtedly does not work for you, but it most certainly works for me.

I have noticed that many people have a vague, unarticulated sense of what they would like to accomplish in their lives. This easily leads to bad decisions that are not in line with your purpose. When that happens, often, you feel in your gut that things are not right, but you push

through anyway because your rational mind tells you it all makes sense. The result is a constant tension between what you're doing and what you should be doing, causing unhealthy behaviours and outcomes.

Concluding, as you're reading this, my suggestion is that you spend a moment to reflect on why you do what you do. To make your purpose explicit and express to yourself, but preferably in written down form, what the meaning of your life is. And then to reflect on the extent to which your current life is aligned with your purpose. And, finally, to make the changes required. Because the world deserves you acting in accordance with your purpose!

R&D Strategy

Whereas we in previous sections focused on the overall business strategy, the different functions in the company require strategies as well. In software-intensive systems companies, one of the key areas that requires a clear and well-defined strategy is research and development (R&D).

As Peter Drucker is famous for saying: the goal of any company is to create customers. Customers buy products, solutions and services. These offerings are created by the R&D organization and consequently, R&D is central in the establishment of a digitalized company.

One of the key realizations that I had over the last years is that, in practice, it typically is R&D that sets the overall strategy for the company. The reason is that the response time of R&D to direct its resources to create the right architectures, solutions and products often is much slower than the ability of business leaders to change course. This means that R&D needs to predict the strategic direction of the company years ahead of the business. If R&D made the right choices, the optimal business strategy options are easy to realize and deploy as the R&D organization already started preparing for these options years ahead of time. If R&D made the wrong choices, the consequence is that the optimal business strategy options can not be exploited as the company is unable to field the right products, solutions and services. Hence R&D is critically important to ensure the long term success of the company.

One of the key challenges that R&D organizations experience includes the high investment in commodity functionality. As one of the later sections describes in more detail, our research shows that 8 or 9 out of every 10 people in R&D work on commodity functionality. One of the key drivers of work on commodity functionality is customization of products and systems for specific customers. Although the customer typically appreciates the customization, the long-term consequences area highly problematic in that the evolution of customized products is extremely costly and the R&D organization needs to repeat the adaptation to each customer’s version of the product.

A second key challenge is the focus on efficiency that most R&D organizations exhibit. Generally, R&D organizations focus on building and deploying as many features, products, services and solutions as possible, without establishing a clear understanding of the business value generated by these efforts. As our research shows that, typically, more than half of the features in a typical system are never used, we need to adopt approaches that allow companies to shift the focus in R&D from efficiency to effectiveness.

Concluding, business strategy and R&D strategy have a complex, intertwined and bi-directional dependency on each other and it is critically important for business leaders to understand

implications of decisions concerning R&D as these decisions safeguard or destroy the long term future of the company.

Effective R&D in Complex Systems

Several of the companies that I work with build very complex systems that are hard to break down in largely independent parts. Instead the components in these architectures are internally complex and have elaborate dependencies between them. Although this is an architectural challenge, it also leads to an organizational challenge.

Traditional R&D organizations have a preference for component teams where the team owns a component and coordinates with other teams over the interfaces between this component and other components. Assuming the BAPO model, this is a model where the organization follows the architecture which generally should be considered a good thing. A component team has a solid understanding of the component that it is responsible for and consequently can limit design erosion inside the component.

The disadvantage of organizing R&D as component teams is threefold. The first is that it leads to coordination overhead as most features and other R&D work items affect multiple components. With component teams, the functionality needs to be assigned to the different components and the teams need to agree on how to extend or change the interfaces between them in response to the new functionality. This requires teams to spend significant amounts of time together as the initial division and interface specification never works as intended. Second, the architecture of the overall system is difficult to protect as each component team will optimize locally, but feel little responsibility for the overall system. Finally, component teams have a tendency to ensure that they have enough work. So, teams and their product owners/managers will fill their backlog to ensure that they have work, rather than prioritize the R&D effort based on what is most important for the company overall.

In recent years, I have been a proponent of feature teams. A feature team is responsible for realizing a feature and can make changes in any component that is involved in the feature. The major advantage is twofold. First, the inter-team coordination overhead is removed and the team only needs to align internally. Second, the realization of the feature tends to be much more homogenous as all functionality is built by one team.

The disadvantages of feature teams are mostly concerned with protecting the architecture, ensuring quality and competence. In organizations successfully using the feature team model, these disadvantages are counteracted by several means. First, these organizations appoint architects responsibility for protecting and explaining the part of the architecture they are responsible for. Second, continuous integration allows for rapid and frequent feedback on work performed by feature teams, ensuring that teams breaking functionality get immediate feedback. The main challenge that I see in companies that I work with, though, is concerns competence: these systems have parts and interactions that are so complicated and that take so long to become proficient in that it is unreasonable to expect that a feature team can cover this all by itself. Even when using feature teams in this context, these teams spend enormous amounts of

time with other teams, experts and architects in order to understand what needs to be build and how it is best realized.

The above leaves us with a bit of a conundrum because both models lead to significant coordination overhead in the context of complex systems. When basic models fail, as a leader you need to return to first principles. For me, there are four basic principles that should be followed when organizing R&D in complex systems.

First, **autonomy is king**: To the largest extent possible, teams should be organized such that their autonomy if the highest possible. Coordination cost, as expressed in meetings, documents, emails back and forth, etc. are simply the death of productivity. Coordination should be managed through architecture and automation, such as the continuous integration infrastructure, and not through human processes.

Second, **less is more**: Many R&D leaders always seek to add more resources to the organization, leading to more and larger teams. In my experience, adding resources can easily lead to so much additional effort that the productivity of the organization sinks below that of the earlier, smaller organization. The aim should always be minimize amount of R&D resources involved as productivity asymptotically drops when growing the organization.

Third, **hybrid works**: In several companies that I have worked with, homogeneity is the highest goal. Everywhere in the R&D organization, the same team setup, processes, ways of working and tooling are used. The underlying assumption is that it simplifies things for everyone involved and it allows for greater mobility of staff. Especially in complex systems, however, this approach breaks the Einstein principle of making everything as simple as possible, but not simpler. In reality, it is often much better tobe flexible and allow for different models within the R&D organization responsible for a system.

Finally, **break architectural complacency**: Due to their complexity, the companies building these systems tend to have been around for a long time. Due to resource constraints and limits in computing power years or decades back, the system architecture frequently became so complex as deep cross-architecture dependencies were required to meet challenging quality attributes. The organization then easily develops a shadow belief that this highly complex architecture is a necessity and a fact of life, resulting in architecture complacency. The final principle is that the R&D organization needs to frequently reevaluate the need for the elaborate dependencies between different parts of the architecture. Simplifying dependencies and adopting highly decoupled approaches such as service-oriented and micro-service architectures not only decouple components, but also increase the autonomy of teams. The miracle of Moore’s law helps immensely address these issues.

Concluding, as an R&D leader, it is tempting to grasp for models that have worked in the past or that work well at other companies. It’s so easy and comforting to have a template and recipe to bring to the meeting and to use as an anchor. Over the years, however, I have become

increasingly convinced that each organization is unique; similar, but not the same. Consequently, every organization needs to find its own path and go through its own journey to achieve the desired outcomes. This requires leaders to start from first principles and reason through and experiment with the implications of these principles. Don’t be the fool that looks at the finger, but rather look at the moon the finger is pointing at. Teams follow leaders that set high bars for themselves and their organization, so start from principles and reach for the stars!

The End of Requirements

The time has come to eradicate requirements as a mechanism for communicating between different groups inside and between organizations. Although requirements have been used as the key mechanism to describe the functionality desired from the system since the beginning of software engineering as a field, over recent years the limitations have become increasingly clear to me. Some of the problems include:

- **Whispering game**: Especially in large organizations, the distance between the customer and the R&D team is very far. This means that requirements are transferred and transformed between different functions (sales, marketing, product management, etc.) and what reaches the R&D team deviates significantly from the original need.
- **Jumping to solutions**: It is very difficult for people describing requirements to avoid jumping to solutions instead of describing needs. This constrains the team in realizing the optimal solution and fails to employ the full creativity of the team.
- **Tacit knowledge**: Every non-trivial system has intricate aspects that are difficult to communicate. In my experience, requirements only communicate the obvious 10% of expected functionality or outcomes and fail to capture the 90% that is hard.
- **Open for multiple interpretations**: No matter how precise, many requirements are fuzzy and open for different interpretations. This leads to the classical situation where the team delivers something that is far from what the customer desired.
- **Distance to the customer**: Requirements are used as a solution for keeping the teams and customers apart and for the team to avoid having to learn the domain. Agile software development has taught us that this simply doesn't work: teams need to meet the customer, learn the domain and build empathy. This is the only way to build software that meets the needs of customers.
- **Conflicting interests**: Many problems in software engineering originate from conflicts within the customer's organization where different roles and functions contradict each other. Requirements tend to obfuscate these conflicting interests between different parts of the client organization instead of help resolve these. This easily leads to a situation where the team is blamed for dysfunctions at the customer side.

Instead of requirements, we need instead to start focusing on outcomes: what does the customer seek to accomplish with a new system or new functionality in the existing system. For instance, when requesting new functionality, the focus should not be on the requirements on the functionality, but rather on the measurable changes in customer and system behaviour that we're looking to accomplish. For instance, for a free online game showing ads as a revenue generator, three factors would be important: new users, recurring users and revenue. Whatever the customer requests, the intended outcome will be to drive up one or more of these factors without (too much) negative effect on the other factors. If at all possible, these desired outcomes should be expressed quantitatively.

As most functionality added to a system will affect some factors positively and others negatively, we need to describe what the relative priority is. This requires us to assign a weight to each factor and to describe the expected outcome as a value function. For our fictive online game, the value function may look as follows:

Vf = 0.3 * % New Users + 0.3 * % Recurring Users + 0.4 * % Revenue

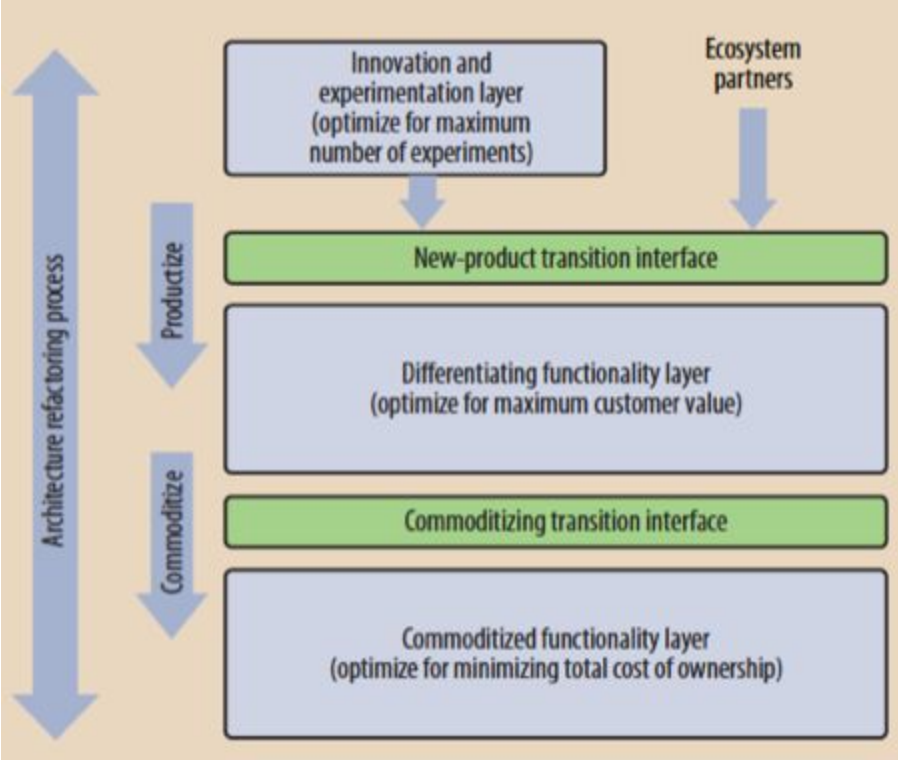
When the team receives some suggestions for new functionality and the intended outcome, development shifts from a “meet the requirements” model to an experimental mode of development where teams try out different solutions in order to positively affect the value function for a system or a feature. This, in turn, helps the organization to shift from an opinion driven to a data-driven organization.

For those organizations that struggle to get away from requirements, there is an intermediate step: Rather than describing requirements, insist on capturing the desired functionality as test cases. This means that the customer or product manager at the beginning of every sprint meets with the team and together with the team agrees on the test cases that need to pass for the desired functionality to be realized. In fact, we use the test cases as requirements. Although only a step on the way, it does address several of the aforementioned challenges of traditional requirements.

Concluding, it’s time to let go of requirements as the mechanism to capture the functionality desired from the system.Instead, focus on outcomes, adopting an outcome-driven development model and moving towards a much more experimental approach. This blog post only scratches the surface of this topic. In the short book titled “Using Data To Build Better Products”, I provide much more detail on how to move from requirements-driven to outcome-driven development, so do take a look [Bosch 17b].

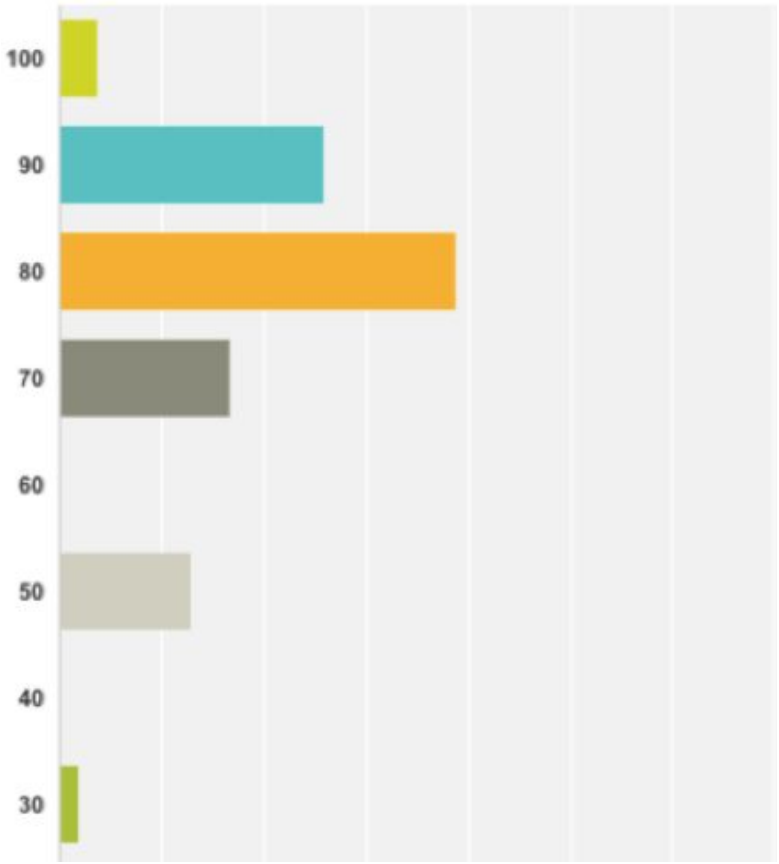
9 Out Of 10 in R&D Work On Commodity

This week I ran a workshop at a large well-known company applying the three layer product model (3LPM). The 3LPM is a conceptual approach to categorizing the functionality in your product, platform or portfolio. As the name suggests, the functionality in the system is put into three boxes, i.e. commodity, differentiating or innovative. The model is shown in the figure below.



Commodity functionality refers to functionality that competitors offer as well. Consequently, this functionality needs to work, but doesn't help the company distinguish itself from its competitors and through that drive sales. Differentiating functionality concerns proven features that your company provides and the competition does not or that you provide in a significantly better way than others. As customers care about it, it helps you drive sales and consequently it matters. Finally, innovative functionality is concerned with unproven features that may provide future differentiation, but it's unvalidated. The process is, of course, that functionality starts of as innovative and when it proves successful, it is productized and becomes part of the differentiating functionality layer. Once the competition catches up, the functionality becomes commodity.

As shown in the figure below, our research shows that 80-90% of all R&D effort is allocated to commodity functionality. This means that out of every 10 people in R&D, 8 or 9 work on functionality that no customer cares about. It just has to work and, if it does, nobody cares.



The question of course is how a company stays competitive when the vast majority of its R&D staff works on things no customer cares about. The answer is quite blunt: you don't stay competitive this way. You have to work very hard on freeing up resources in the commodity functionality using a variety of strategies, ranging from replacing proprietary functionality with open-source or commercial components to outsourcing commodity functionality to suppliers in low-wage countries.

If we had only seen one or two companies in this situation, it would have been an exception. However, I have now asked well over 50 companies for their self-assessed estimation of R&D effort into commodity and the answer averages around the 80-90%. Just reflect on this for a moment: 9 out of 10 people in your R&D organization work on things no customer gives a flying hoot about!

So, what to do about this? Well, going back to the company that I visited this week, the first step is to assess where you are. This means systematically decomposing your system into smaller and smaller elements until each can be uniquely classified as commodity, differentiating or innovative. Next is to assess inter-mingling and the dependencies between these elements and finally, it is estimating the resources allocated to each element on an ongoing basis. This provides a current state assessment.

The next is to develop a vision of a desired state. This is not a situation where there are no R&D resources spent on commodity - some will always be required - but rather a resource allocation state that is aligned with the business strategy and materially better than the current state. Finally, we need a transition plan that focuses on moving to the desired state such that the earliest actions bring the largest benefits.

Concluding, organizations routinely spend 80-90% of their R&D resources on commodity functionality. Although many view this as a necessary evil and assume that there is nothing that can be done to address this, there are many strategies that companies can employ to free up resources in commodity so that these can be applied to differentiation and innovation. Similar to many situations that leaders in R&D experience, don’t accept the status quo as the best or only way to do things. Challenge, develop alternatives, experiment and improve!

Half The Features You Build Are Waste

Over the last two decades or so, I have worked with companies on their R&D efficiency. R&D efficiency, to me, is concerned with creating as much value as possible for every unit of R&D effort (person hours, currency, etc.). Value can be created in many ways, but a pretty good approximation is the frequency of use of a feature across the user base.

Measuring feature usage is pretty much the norm in the Web 2.0 and SaaS world, but in many of the other companies R&D has little or no understanding of the actual use of features in their systems. In our research we have studied feature usage in different companies and software systems and in general our research shows that around half of the features are hardly ever if ever used.

So, not only do 9 out of 10 in R&D work on commodity functionality, as I shared in last week's blogpost. Of the small amount of resources that is allocated to adding new functionality, which obviously intended to be differentiating, half of that functionality turns out to be a waste. Half of the features are used very little and do not provide the expected value. For instance, in the picture below, we captured frequence of feature use and it's clear that the majority of features are used very infrequently.

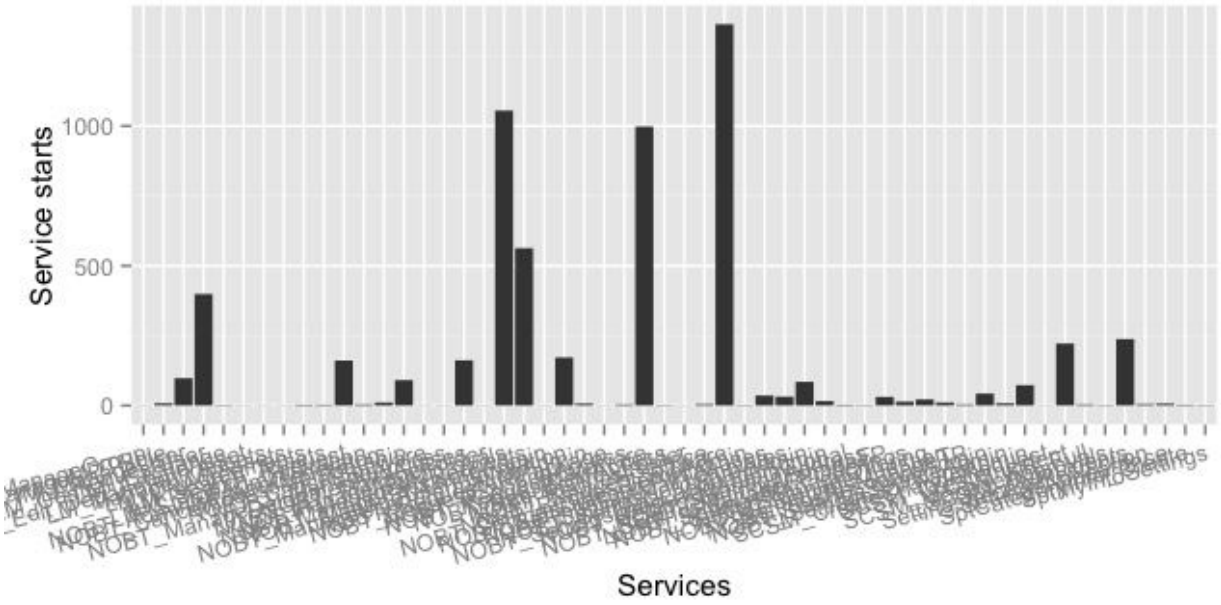


Figure 1: overview of feature usage for a software system

In many companies, product management is a separate organization that communicates with the R&D organization through requirement specifications. In response, the R&D organization just builds the functionality according to the specification. This process is based on the,

frequently qualitative, input that product managers receive from customers. This means that R&D builds according to the specifications with very little understanding of how the functionality will be used by customers.

A second challenge is that many organizations rely on what customers say they want, rather than base decisions on the actual behaviour of customers. Especially in B2B markets, customers have clear opinions about what they want to see in new product releases. However, the actual behaviour of customers and the effects of new functionality on system behaviour is frequently very different from the expectations. There are many examples of the gap between espoused theory and theory in use.

The root cause for these challenges is the lack of data and use of data in the end-to-end product development process. Although companies typically collect vast amounts of data, this data is not useful for tracking feature usage. And when I work with R&D teams, the awareness and understanding of using data for anything but quality assurance and troubleshooting is highly limited. We have done research on this topic for years now and have several interesting models and other results. One example is, for instance, the HYPEX model shown in the figure below.

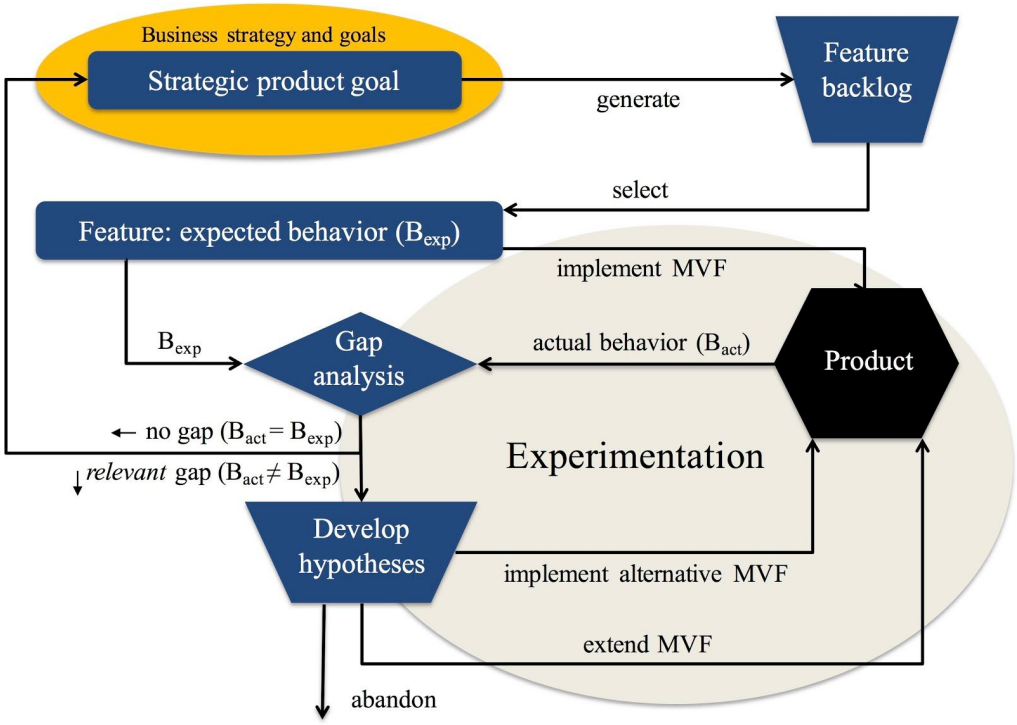


Figure 2: The HYPEX model

To address this, over the last months I've written a short book [Bosch 17b] on using data to build better products. The book describes the basics of working with data in R&D and uses a fictive startup team working with data for their embedded product. The book focuses on three steps. The first step is understanding feature usage. The second step is concerned with

optimizing features to deliver the intended outcome. The third step is about modeling the intended value of a feature and then tracking the relevant indicators during development and after deployment. The book illustrates these steps with hands-on examples and a running example.

On the Role of Software Architecture

This week was all about software architecture as we hosted the international conference on software architecture in Gothenburg (ICSA 2017). At the same time, I hosted a company get-together between, among others, Booking, Spotify and Klarna. During the get-together, the role of architecture also came up a number of times. I realized that the research community (and the traditional systems engineering companies) has a very different view on the role of software architecture as compared to the online Web 2.0/SaaS companies. When comparing these two different perspectives on software architecture, four fundamental differences jumped out at me.

Just-in-time architecture: Traditional software architecture thinking is focused on doing architecture work early and before others start. The thinking is that you need to put a structure together for everyone to operate within before the rest of development can start. The online companies focus much more on just-in-time architecture work: just before a more structural change to the system is needed, e.g. due to scaling, work is kicked off in a bottom-up, “concerned engineers” style that addresses the concerns at hand.

For online companies, the focus seems to be on doing as little architecture as possible as it puts structures in place that constrain development. Which leads to the next observation...

Decouple teams: Traditional systems engineering companies and the academic community at large seems to design architectures to satisfy multiple, conflicting quality attributes and focuses on the performance of the system in operation. The online companies focus their energy on architecting their system such that teams can operate with as few dependencies on each other as possible.

This may sound like traditional component teams, but it really isn’t. Component teams carry responsibility for a specific component in the system but requirements tend to affect multiple components and consequently, the component teams have very high coordination cost. The alternative organizational structure, feature teams remove this coordination cost, but have other areas where coordination cost may be substantial. The online companies architect such that teams can truly work independently with as simple interfaces between the different systems as possible and allow teams to (almost) run their own products.

No architects: Traditional companies tend to be big on job titles and specialization including developers, software architects, systems engineers, systems architects, development managers, etc. In the online companies, it seems to be the opposite. One of the companies explicitly states that they don’t have architects and this is very intentional. The traditional idea behind the architect is often concerned with taking the decisions and then policing the teams to ensure compliance. This goes straight against the notion of empowered teams that can operate without coordination overhead and that feel both the freedom and the obligation to protect the architecture.

Development inefficiencies: Finally, traditional companies treat any development inefficiencies, e.g. two teams building similar functionality, like the plague as this is viewed as the worst use of precious development resources. Consequently, vast amounts of process, oversight, management, governance and otherwise is put in place to increase the efficiency of development.

The online companies, on the other hand, tend to be very much focused on data-driven development, such as measuring feature usage, and experimentation, such as A/B testing, and know that most of the R&D work never leads to a significant contribution to the business. That makes the R&D in a way waste, but at the same time, you need to try things out to figure out what works. The awareness of the lack of effectiveness of R&D efforts causes online companies to focus on enabling teams to try as many different hypotheses as possible and if this means doing double effort in different teams, so be it. The cost of controlling what teams do is much higher than the occasional double effort.

Concluding, although every system has an architecture, accidental or intentional, and this architecture needs to be governed and carefully evolved, there are fundamentally different ways of doing this. There are significant differences between traditional embedded systems companies and online Web 2.0/SaaS companies. Although some of these differences are driven by the domains in which these companies operate, I believe that there is a lot to learn from how online companies work with architecture as it applies across many industries.

How To Double Your R&D Effectiveness

The vast majority of R&D leaders and especially consultants focus on the efficiency of R&D: how to convert requirements into software against the lowest possible cost and resource needs. Although anyone from agile and scrum experts to continuous integration professionals focus on this challenge, very few people focus on the effectiveness of R&D. Effectiveness refers to maximizing the business value generated by the R&D investment.

In earlier posts, I have shared that research by us and others shows that somewhere between half and two-thirds of all features added to a system are waste. The frequency of use is so low that, if the feature is used at all, that it should not have been built in the first place. This means that even if the efficiency of building the feature may have been very high, the R&D effectiveness was very low as it did not deliver any business value.

Similarly, I have shared in earlier posts that in most companies the majority of R&D resources are spent on commodity functionality. In fact, 80-90% of R&D resources at many companies is allocated to commodity functionality. Although commodity functionality is needed for your system or product to work properly, the fact of the matter is that it is differentiating functionality that drives sales and consequently generates business value. Again, the efficiency of R&D in commodity may be very high, but the business value generated is very low.

Assuming that this is the case for your organization, how can you increase the effectiveness of your R&D? In the way that I work with companies in order to accomplish this, we focus on three activities. First, starting from the architecture of the system, we classify each component or subsystem as either being commodity, differentiating or innovative. If it is not possible to classify a subsystem into one of these categories, we decompose the subsystem into its constituent parts and repeat the exercise. Please note that for any component to be marked diffferent from commodity, there has to be real evidence from customers that the functionality exhibited by the component indeed is differentiating or innovative and consequently delivers business value.

Once the components that make up the system have been labeled, the next step is to estimate resource allocation for each component. There are different ways to do this, but the details are beyond the scope of this article. Once the current resource allocation has been established, we make a decision that commodity components will only receive R&D resources for bug fixing and for staying compatible with external software assets. For instance, when the company upgrades the operating system on top of which its software runs, also legacy components need to be adjusted. However, except for bug fixing, no R&D resources can be allocated to commodity components. If conducted properly this allows companies to reduce the allocation of R&D resources to commodity with 50-75%. This in itself will already double R&D effectiveness.

In the next article, I will discuss the other activities that we apply for improving R&D effectiveness. These activities are concerned with measuring the value of already realized

features in deployed systems as well as new features. Concluding, very few R&D leaders and consultants focus on effectiveness. I believe that this is a huge missed opportunity for most companies. In our research and my engagement with a variety of companies, I have developed and applied techniques that allow companies to double the effectiveness of their R&D investments.

The End of Planning

Over the last years, I have time and again run into the conflict that many companies experience between long term planning and continuous (agile) practices. In an embedded systems context, where the system contains mechanics and electronics in addition to software, the organization needs long term planning for “the atoms” because of manufacturing processes, ordering of parts, etc. As the company seeks to ship a system that also includes the software and consequently all R&D units, including software, are asked to commit to delivering a large amount of functionality at specified date at some point in the (far) future.

Software R&D, on the other hand, is concerned with agile practices and sprint-based prioritization of the most important things to take on next. Being asked to commit to a delivery of the software part of a product 12 to 18 months out is simply a bad idea from a business perspective. First, because all engineering disciplines, including software engineering, have great difficulty with long term plans, those responsible for creating these plans exercise extreme caution. Often the rule of pi is used: first the effort and time is estimated and subsequently all numbers are multiplied with pi (3.14) in order to maximize the likelihood of delivering on time. However, engineering involves humans and once they know they have 18 weeks for a task that was first estimated to take 6 weeks, engineers will find ways to fill the 18 weeks with work on the specific task. Second, in a fast moving world, from a business agility perspective the company wants the ability to decide as late as possible which functionality to put into the product. And it wants to change the prioritization continuously as new information becomes available, competitors make surprising moves or new technologies reach suitable price points.

When analysing why this pattern keeps reoccurring across industries and companies, I realized that the notion of long term planning for software is based on several misconceptions. First, product managers, system architects as well as sales professionals who grew up in a mechanics and electronics driven world assume that the functionality delivered at the start of manufacturing is frozen and can’t be changed anymore. Because of this, software needs to commit early and deliver on the exact date. In an age where virtually all products are connected and most products move to continuous deployment, the software enabled features can be deployed throughout the life of the product.

A second misconception is the, often implicit, belief that post-deployment issues will be incredibly expensive to fix. While this is undoubtedly true for the mechanics and hardware of products, it is not the case for the software in connected products. When properly architected, the cost of deploying new versions of software rapidly approaches zero.

The third misconception is that deep dependencies and connections between the different parts of the system is not a concern but actually a good thing because it allows engineers to squeeze the last bit of performance out of a system that is dimensioned as close to the minimum as possible. Instead, proper decoupling should be placed at those interfaces in the system

architecture where the rate of evolution is fundamentally different. In practice, mechanics and electronics tend to evolve in yearly (or even slower) cycles whereas software may be released multiple times per week. This leads to two orders of magnitude difference in evolution speed and obviously the interface between both parts needs to be as clean, stable and decoupled as possible. Unfortunately, in many companies I have seen situations where the specifics of sensors or actuators used in the lowest levels of the system percolate throughout the entire system right up to the user interface so that even the user is exposed to the details deep down in the bowels of the system.

The fourth misconception is that bill-of-materials (BOM) is the only thing that matters. Especially companies that mass-produce products tend to hold an implicit belief that anything that lowers the BOM is worth any engineering effort. This belief is incorrect for two reasons. First, with the size of the software increasing with an order of magnitude (10x) every 5-10 years, the software related R&D expenses go up as well. For many companies, the per-unit software cost is now on par, if not exceeding, the cost of hardware. It's just that many companies use separate budgets of BOM and R&D and consequently this is not obvious until one analyses the financials. Second, dimensioning the hardware capacity at the very minimum removes the ability to deploy new functionality to products in the field because there is no space, either from a storage or a computational perspective, for the additional functionality.

Although there are other incorrect beliefs around this topic, the aforementioned ones are the primary concerns. The solution is, of course, to embrace the fact that bits are different from atoms and need to be treated as such as well. So, for anything atoms, we keep the existing planning processes as these, for better or worse, are the best ones available. However, for anything bits, we adopt a continuous planning process where every sprint the most important work gets prioritized and put in the backlog. As there will undoubtedly be a few items that are non-negotiable for the product as a whole, the software R&D team may be asked or forced to make a few long-term commitments. However, as the volume of long-term commitments stays low, say below 30% or 50%, it allows the company to enjoy the business agility that it needs to stay competitive.

Concluding, forcing software R&D teams to make long term plans and commitments is not only bad for the software teams, it is a suboptimal approach for the company as a whole. Embracing the unique and different nature of software and capitalizing on the business agility that continuous, sprint-based prioritization offers is the superior way of working. And, of course, this does not just concern products under development but also, in the age of continuous deployment, the products out in the field.

The End Of Product Teams

In the age of continuous deployment, B2B and B2C customers expect that the software in their product or system is updated frequently throughout its economic life. As a minimum, security updates need to be deployed, but also bug fixes and improvements or even entirely new functionality is expected to be added to products and systems out in the field.

An important implication of this is that the traditional way of building the software for embedded systems is no longer feasible. Traditionally, for each product development initiative a product team is created. The team is responsible for getting commitment on the requirements for the product, designing the product and developing it. Towards the end of the process, the product is tested and validated against the requirements. Finally, the product is released for manufacturing and shipped to the customer. Once this is accomplished, the team disbands and moves on to a new product initiative.

Of course, in most companies, products are not built from scratch, but rather are built on top of a platform. At least for the software in the system, the typical process is that the product team clones the platform code so that it owns its own copy that it can make the changes in that are required to meet the product specific code.

In fact, most companies evolve through a pattern as shown in the figure below. Starting from independent products, companies standardize infrastructure, introduce platforms and then adopt full fledged software product lines. However, most companies stay short of the final stage.

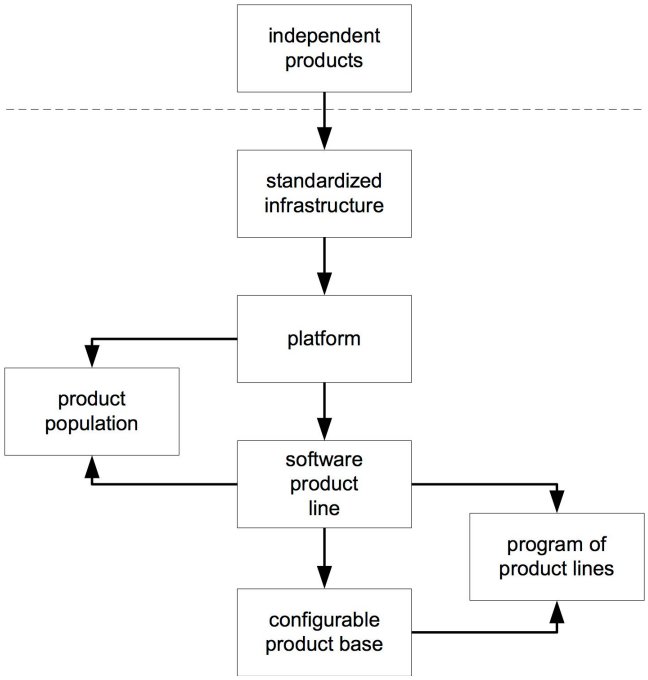


Figure: typical evolution of software-intensive systems companies

The challenge is that product teams up to now disbanded and moved on to the next project when the product or system enters the market. Maybe some maintenance and bug fixing would be required, but for that a separate maintenance organization was in place. When the company employs continuous deployment, the product team can not disband as the software in the product needs to continue to evolve as long as there are products operational in the field. And customers will want to receive new functionality and features that expand the capabilities of the product. Finally, in case the product or system is deployed as a service, the company itself may benefit from deploying new software to improve efficiency.

So, if each product created by the company needs a product team for as long as there are products deployed in the field, the number of product teams needs to grow continuously. For instance, assume a company that introduces 10 new products to market per year and each product has an economic life of 10 years. In the old model, assuming the R&D for a new product would take a year or less, the company could keep 10 product teams around that constantly worked on new products. In the case of continuous deployment, however, over the years, the number of product teams will need to grow to 100. For most organizations, this is an entirely unacceptable situation as the economics of their R&D would never support this many product teams.

It is clear that we need a fundamentally new setup: we need to move to the final stage of the evolution model: the configurable product base. As shown in the figure below, rather than creating product teams for each product, the teams build features, fix defects and refactor one single code base. Each individual product is automatically derived from this single code base.

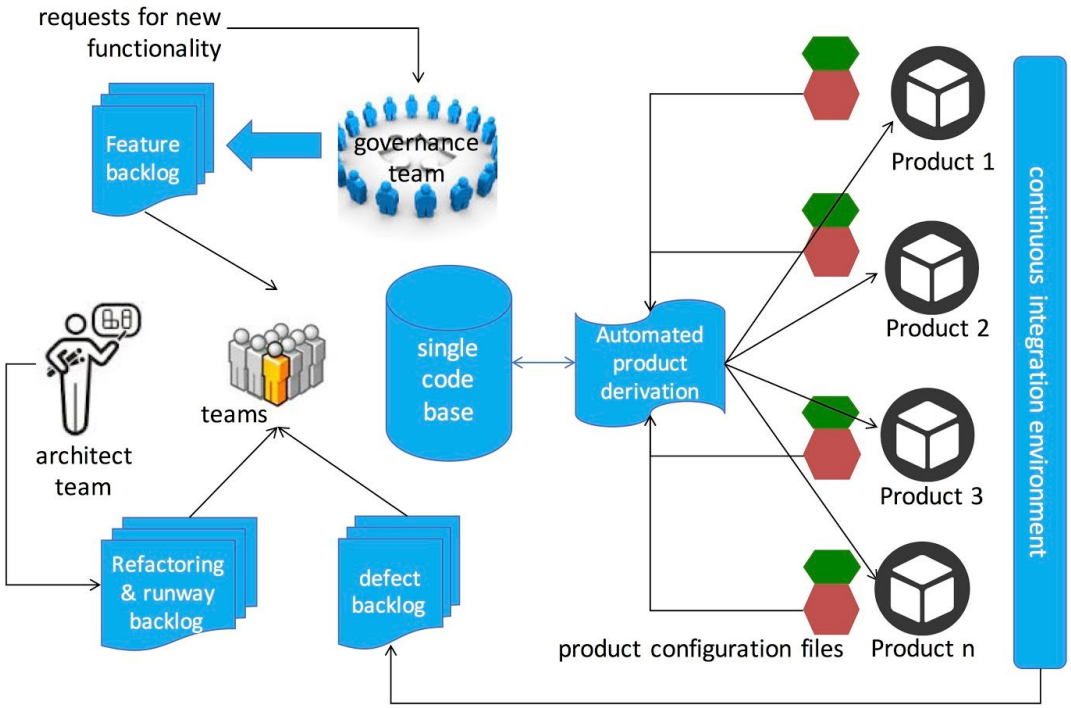


Figure: configurable product base

The approach presented in the above figure allows the number of products to scale easily as the generation of product specific code as well as the quality assurance of this product specific code are automated. The engineers and architects work on the single code base and their efforts positively affect the entire product portfolio.

Although the approach requires a significant level of discipline and maturity of the organization, the benefits are enormous. In a world where we are moving towards continuous deployment, there really isn't any alternative to this approach that addresses the evolution of functionality throughout the lifetime of the product except for scaling the R&D organization linearly with the number of products out in the field.

Concluding, we're reaching the end of product teams, for all the reasons outlined in this article. Instead we're moving towards portfolio teams that use automation to generate and continuous validate each and every instance of product specific software. Currently, I work with several companies to realize this transition in their organization as there are several aspects to this model that may not initially be obvious. Please do reach out in case you would like to learn more!

On Functional Safety in the Age of Continuous Deployment

This week I hosted a workshop on continuous deployment of software subject to functional safety standards. We agreed to keep it low profile on who participated, but several of the large companies in automotive, aeronautics, industry and defense were present, including OEMs and tier 1 suppliers. It was a good group that was dominated by functional safety experts with a smaller number of agile experts sprinkled in between for good measure.

From a business perspective. It is obvious why we want continuous deployment. It's all about shortening the feedback loop between the customer and the company providing the product. By measuring how the product performs in the field and how customers use the product, we can use that data to improve the functionality in the product. With new software deployed every few weeks, the improvements, though small in every sprint, lead to a large cumulative effect. Then, customers of course increasingly expect their product to get better over time. And, purely practically, the cost of product recalls disappears if we can fix the issue with a software update that is distributed, basically for free, to products in the field.

At first sight, the two topics of continuous deployment and functional safety seem to be at complete odds with each other. Agile is all about small, incremental steps on a sprint based iteration. Functional safety is about testing the heck out of the complete system, collecting all the necessary evidence that the system is safe, getting it certified by an external assessor and then preferably never touching it again as every change requires redoing the safety certification.

During the workshop, however, it became clear that there are concrete ways to integrate functional safety into agile development practices. Continuous, or at least sprint-based releases of software, can be accomplished even if it requires functional safety assessment and certification for every sprint. However, work is needed in several areas. For the companies building these systems changes are needed in system architecture, development processes and automation. For assessor, the way assessment and certification is performed needs to changed, but we leave this out of the scope of this article.

The system architecture of software intensive systems tends to be highly interconnected with numerous dependencies throughout the system. The reason for this is twofold. First, traditionally the primary driver for these systems was the bill of materials (BOM). The belief was that any small saving we can achieve in the BOM justifies any R&D investment that we need to make to accomplish it. And especially for products produced at high volume this was true for many years. However, in many companies the cost of software R&D is becoming so high that it tends to be on par with the BOM cost and consequently, we need to make more intelligent decisions. Second, for some reason, system, mechanics and hardware architects seem to have put complexity at a much lower priority than software architects. However, if we want to deploy software continuously throughout the lifetime of the product, we need to put excess resources in the system from the start. Otherwise we can't even provide the first upgrade. Second, however,

we need to modularize our architecture so that safety related functionality can be assigned to independent subsystems allowing safety assessment to be simplified.

The development process, especially the sprint activities, is the second area that needs extension. Work on functional safety, such as hazard analysis and evidence that shows that known hazards have been addressed appropriately, needs to take place every sprint and need to be added to the backlog. The real change is, however, even more fundamental: the work on functionality safety needs to satisfy two characteristics, i.e. it needs to be iterative and it needs to be cumulative. This requires that for each item on the backlog, iterative hazard analysis takes place that identifies new hazards created by this item (if any) and the potential implications on the already identified hazards. In addition, the activities required for functional safety need to be cumulative in that I don't want to be required to redo all the unaffected work.

Finally, one area that is undoubtedly required is automation. The current functional safety certification process requires vast amounts of documents that to a large extent are manually created. It is clear that for a sprint based approach this is unfeasible and we need to automate the generation of these documents. However, the focus should be on creating a database where the data required for these documents is stored. The documents can then be generated whenever necessary.

Concluding, also software that is subject to functional safety standards can be deployed continuously and we've started to work our way towards making this a reality. During the workshop, we agreed to organize a follow up workshop later this year. If you are interested to learn more and potentially join, please send me an email on jan@janbosch.com.

Why Bad Business Habits Kill R&D Effectiveness

Since the summer, I've been working with well over a dozen companies in different capacities and in several of these companies, there is a challenge on the R&D front that actually is caused by bad business habits. When I'm asked to spend some time with a company, it very often is because there is some challenge on the R&D side such as lack of reuse, quality issues, predictability and time to market concerns, etc. Frequently, R&D is then viewed as the problem child of the company by the other functions and R&D leadership sometimes has even started to believe the story and frantically looks for solutions inside the confines of the R&D organization. When analysing the challenges faced by the organization, though, it turns out that although there are opportunities for improvement inside R&D, the root cause of much of what ails R&D actually originates on the business side of the house. There are at least three problems that I have seen recurring time and again: overly tactical sales, unhealthy customer focus and too strong reliance on physical products.

One of the biggest challenges that I see is that the sales organization is ready to sell customers exactly they're asking for without understanding or, in fact, caring about the downstream consequences. This leads to significant overhead for R&D due to the lifetime cost of the additional complexity, due to variants, customer specific code branches and customer specific requests for "their" version of the product. The problem here is **not** that R&D shouldn't built functionality that provides value for customers. The problem is that the little revenue from selling something unique to a specific customer doesn't even come close to the lifetime cost of maintaining this functionality for years or sometimes even decades to come. Nor does sales take into account the complexity, performance or reliability consequences for other customers. Even if customer specific functionality can be disabled for other customers, it still will affect the other customers in some ways.

The second challenge that I see companies struggle with is the pattern where the customer that screams the loudest is prioritized over other customers. In some of the cases that I saw, the functionality built for a "loud" customer is commodity or so specific for that customer that it does not provide any business value beyond this customer. Although many companies pride themselves on their customer orientation, the fact is that spending R&D resources on the specific needs of one customer comes with an opportunity cost: those resources can not be used for building functionality that serves many or all customers.

The third challenge that many business struggle with is that their senior leadership, even on the business side, has a background in mechanics or electronics and, as a consequence, has a strong focus on physical products and views software as secondary. This exhibits itself in at least two ways. First, whenever faced with a choice to invest in "atoms" or in "bits", the choice almost falls towards "atoms" at the expense of "bits". Second, the focus of everyone in the

company is on minimizing bill of materials for products independent of the implications on the software.

So, having identified these challenges, the question is what we can do about it. In my engagements with different companies, I typically focus on three main activities: reinforcing the business strategy and its implications, establishment of a governance mechanism with proper representation and adopting the three layer product model as a template for reasoning about innovation, differentiation and commodity.

First, I still find it surprising that in many companies the business strategy is treated as some vague, irrelevant document that doesn't really affect day to day operations. When developing the business strategy, the company tends to prioritize software and the establishment of digital services, data-driven services and software products in addition to its physical product portfolio. Starting from the business strategy and reasoning with the team through all the consequences and what this requires in terms of changes right here and now often leads to more intentional and less habitual behaviour in the organization. It is amazing how often different leaders will interpret the business strategy in a way that is logically incoherent but that allows them to keep doing what they've always done.

Second, decision making concerning the prioritization of R&D activities is surprisingly often decentralized and a highly politicized process where project managers, product managers, team leads and engineers constantly engage in a process of "cow trading" and exchanging favours to get what they need for their project or their customer without any regard for the overall company interests. Establishing a governance mechanism where key leaders meet on a frequent basis, preferably every agile sprint, to decide on the relative priority of all the various requests from customers, the roadmap activities, major defects, architecture refactoring and infrastructure needs removes the local optimization and allows for the company to optimize for the overall needs of the company.

Third, in earlier articles, I have introduced the three layer product model (3LPM). The model, as shown in the figure below, distinguishes between three types of functionality: innovative, differentiation and commodity. Our research shows that typical organizations spend 80-90% of their R&D resources on commodity functionality. One of the key reasons is that there often is an incorrect understanding in the organization about what functionality is considered to be innovative or differentiating rather than commodity by customers. This results in a situation that companies tend to invest heavily in commodity functionality that customers tend to have little interest in. Using the 3LPM, one can get a conversation going in the organization about what functionality actually is commodity and should be deprioritized for R&D investments.

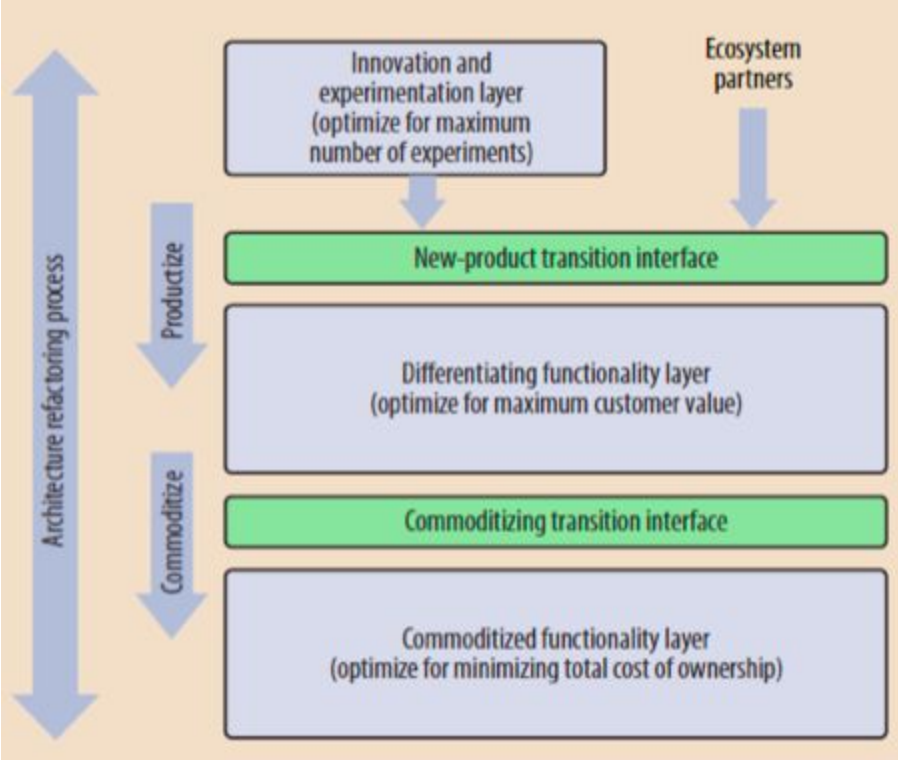


Figure: The Three Layer Product Model

Concluding, the business side of many companies causes fundamental inefficiencies in R&D that offer no business benefits to customers or to the company itself. These inefficiences are caused by overly tactical sales, unhealthy customer focus and too strong reliance on physical products. By reinforcing the business strategy and its implications, establishing a governance mechanism with proper representation and adopting the three layer product model for reasoning about innovation, differentiation and commodity, we can significantly increase the effectiveness of R&D and business success.

Why Variability Management Still Is An Unsolved Problem

This week I was in Madrid to present a keynote at an workshop around variability management (VAMOS 2018). It was a fun event and to some extent a trip down memory lane as I was part of the early research community that systematically started to work on software variability management in the early 2000s. We organized workshops, made it an increasingly important topic in the Software Product Line Conference series (this year it's in Gothenburg), I had PhD students working on this (email me if you want to read their thesis), edited books on the topic (e.g. this one), companies were founded (not by me!) to provide tools to address this problem, etc. In short, it was a next, big upcoming topic in software engineering.

As I was preparing the keynote and presented it at the event, however, I realized that all the problems that we were studying in the early 2000s still are problems today. And upon reflection, I realized that the companies that I work with today are experiencing the same challenges as the companies that I worked with almost two decades ago did. How, in heaven's name, have we failed to make progress in an area that is notoriously costly, complex and frustrating to handle?

In principle, variability management is simple. There are two sides, i.e. the "problem domain", where we outline what features and functions our system can offer, and the "solution domain" where we have the technical realization of the system, including variation points and variants associated with each variation point. On top of this, we express constraints, both in the problem domain and in the solution domain. For instance, some features cannot be combined with other features whereas in other cases selecting one feature requires that another feature is present as well. In the solution domain, some features are implemented such that they can only be selected or de-selected together. This may lead to a situation that the final "problem domain" variability model reflects solution domain constraints as well.

In my experience there are three main reasons that cause this simple model to fall apart in industrial contexts. First, the solutions that are available often fail to support scale. Research prototypes show the principles using a dozen or so variation points. Industrial practice, however, shows that many systems contain thousands if not TENS of thousands of variation points. Conceptually and practically our methods, frameworks and tools are unable to handle that scale.

Second, in engineering but especially in product management, sales and general management there is a gross underestimation of and lack of respect for the complexity of variability management. One consequence, among many, is that it often results in situation where engineers are forced to quickly add some customer specific functionality to a system that is otherwise set up for proper variability management. As the customer specific functionality should not be available for others, it naturally should be modeled in variation points and associated variants. Under time pressure, engineers then easily opt to create a separate customer-specific branch of the software. This is of course the first step on the highway to hell

as before you know it, every customer will have its own branch and you'll slide into a consulting business rather than a product business.

Third, as eager as organizations are to add variants and variation points, as reticent they are about removing variants and variation points when these no longer serve a business purpose. A variation point or variant, once introduced, is hardly ever removed. This is interesting as in embedded systems companies, the systems architects often work hard to restrict and reduce variation for physical components but ignore or demand more software variability. Also, as many companies fail to collect configuration data for their customers, it is often unknown if there still is a customer out there using a certain variant. As management often does not realize the cost of maintaining a variation point and its associated variants, the number of variation points only grows.

With all these challenges, how do companies still get products out the door? In my experience, the most advanced companies often use homegrown solutions (often called some variation of "configurator"). In combination with process discipline around managing variation points and variants and significant amounts of automated testing in a continuous integration chain, this allows these companies to support a broad configuration space from a single code base. The most immature companies tend to end up with customer branches, sometimes lots of them, and vast amounts of effort to replicate functionality that is required by all customers. However, although these problems solve the pre-deployment variability, these do not solve the post-deployment variability.

For a more general solution, however, in my experience there are four topics that provide material benefit. First, the Three Layer Product Model (3LPM) is a very effective basis for managing the introduction, evolution and removal of variability. As shown in figure 1 below, the notion of 3LPM is to organize functionality in three layers, i.e. a commodity functionality layer, a differentiating functionality layer and a layer for innovation and experimentation.

For variability management, the innovation layer should not be allowed to add variants and variation points. The reason is that most innovations fail and we do not want our software to be littered with useless variation points and variants. For the differentiation layer, we invest in adding and evolving variation points as it allows us to serve different customer segments and geographies better, support multiple external platforms, etc. Finally, in the commodity layer, our focus should be on removing variation points as the business value proposition for maintaining variation points and variants for commodity functionality is no longer valid.

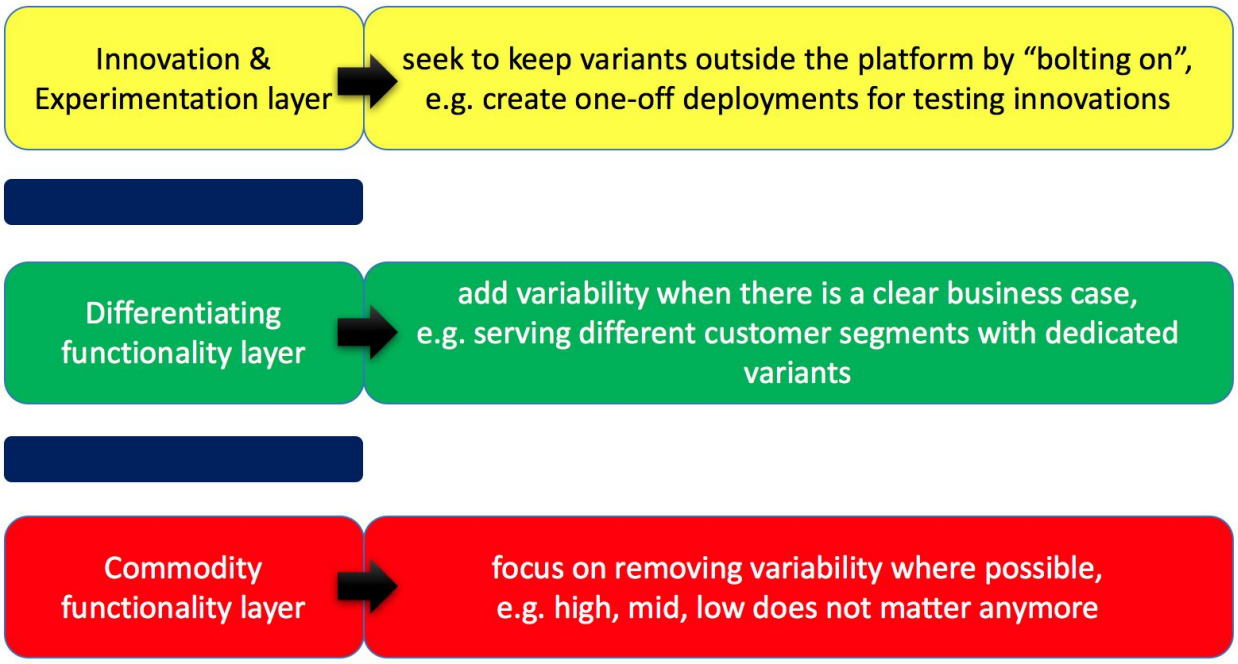


Figure 1: Variability management focus in each of the 3LPM layers

Second, in an earlier blog post, I outlined an approach for automatically generating and testing software for all products automatically as prerequisite for continuous deployment. Even if this requires the development of an inhouse tool, it will remove many of the inefficiencies associated with variability management. As shown in figure 2, this approach will ensure that the organization does not fall back into building customer-specific solutions.

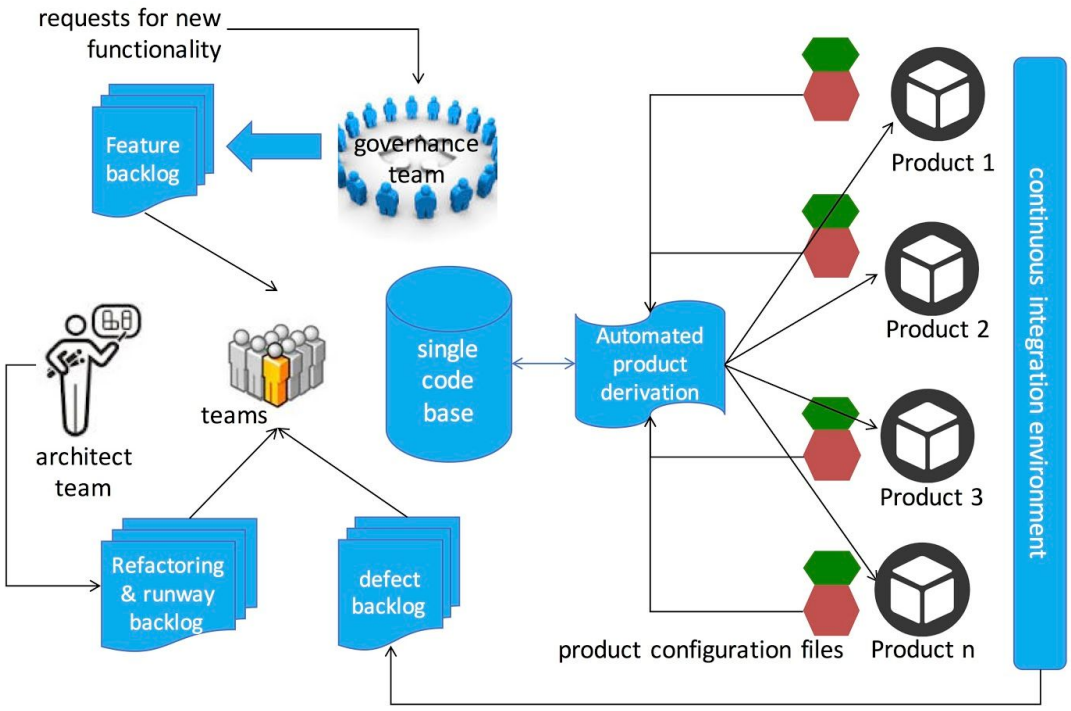


Figure 2. Managing variability and automated product derivation

Third, carefully design a configuration interface that customers will get access to. In several cases, I have experienced a situation where a company was completely unable to control their variability model as its customers had built their configuration tools around its, often poorly designed, variability model. This meant the company, for years and years, had to carry the cost of a poorly designed configuration interface. Instead, design a customer facing configuration interface that is as much as possible expressed in “problem domain” concepts, so that you maintain the flexibility to change the solution domain variability.

Finally, as a fourth strategy, convincing the non-technologists in the company as well as the traditionalists in engineering, often it is required to build some kind of economic model to outline the cost of variability management. Although I have not yet managed to create a generic model, creating a company specific model that estimates the lifetime cost of introducing a variation point, introducing a variant, the running cost of maintaining and evolving variation points and variants and finally the cost of removal should be explicitly available as this is the only counterweight that one can offer against a spur-of-the-moment, let's make the customer happy kind of decision.

Concluding, software variability management is still hard, complex and expensive. It is a challenge often severely underestimated by business and engineers alike. However, there are solution approaches that one can take to address the problem and, if not solve it, at least make it significantly easier to handle. Unfortunately, it often does not allow for a standard recipe to be

applied, but rather it needs a dedicated, company-specific approach to address it. Of course, feel free to reach out if you feel I can help. But get going on this challenge as soon as feasible as it is hard and not only technical in nature. It requires significant behavioral changes from the organization as well. Enjoy your variance!

Stop Wasting Resources And Do Platforms Already!

This week we hosted the 22nd International System and Software Product Line Conference in Gothenburg and I had the honor of being the general chair for the conference. Software product line research is concerned with the challenges associated with building a family of products from a shared platform. These challenges include managing variants, balancing platform development and product development, feature models, etc.

In a world where the size of software for most companies goes up with an order of magnitude every 5 to 10 years, one would assume that the need for reusing software is only increasing. And, with companies adopting continuous deployment, the cost of manually pushing out software for every product that your company has in market at the end of every agile sprint is easily becomes unmanageable. Obviously, the best solution for this is automatic product software derivation from a single (product line) code base.

There is an interesting dichotomy when it comes to software reuse. Software built outside the company is reused to a very large extent. Open source software ranging from Linux to MySQL and from Wordpress to Apache enjoys enormously broad distributions. Commercial software, ranging from the Windows operating system to AUTOSAR base software components, is also broadly used and when companies mention that they have tens of millions of lines of code in their products, this is obviously counted as well.

When it comes to sharing software built inside the company, however, the story is fundamentally different. Although some companies do successfully develop and use platforms within the company, in the majority of companies, for a variety of reasons, the amount of reuse between product teams is highly limited and significant duplication of effort takes place. The result is that, according to our research, 80-90% of all R&D resources is spent on commodity functionality. That means that 8 or 9 out of every 10 people in R&D are working on things that no customer gives a flying hoot about. Stuff that should just be there and work, but that the product team should not spend any time on.

Many moons ago, I published an evolution model as shown in the figure below. The model suggests that starting from a set of independently developed products, the first step should be to standardize on the externally sourced software. Then, the focus should be on developing a platform that includes only features that all products need. The next step is a full-fledged product line where the shared software assets contain functionality used by a subset of products, meaning that variability management becomes a challenge. The final step is the configurable product base where each product or customer configuration is automatically derived from a common code base that contains the superset of all features and functionality.

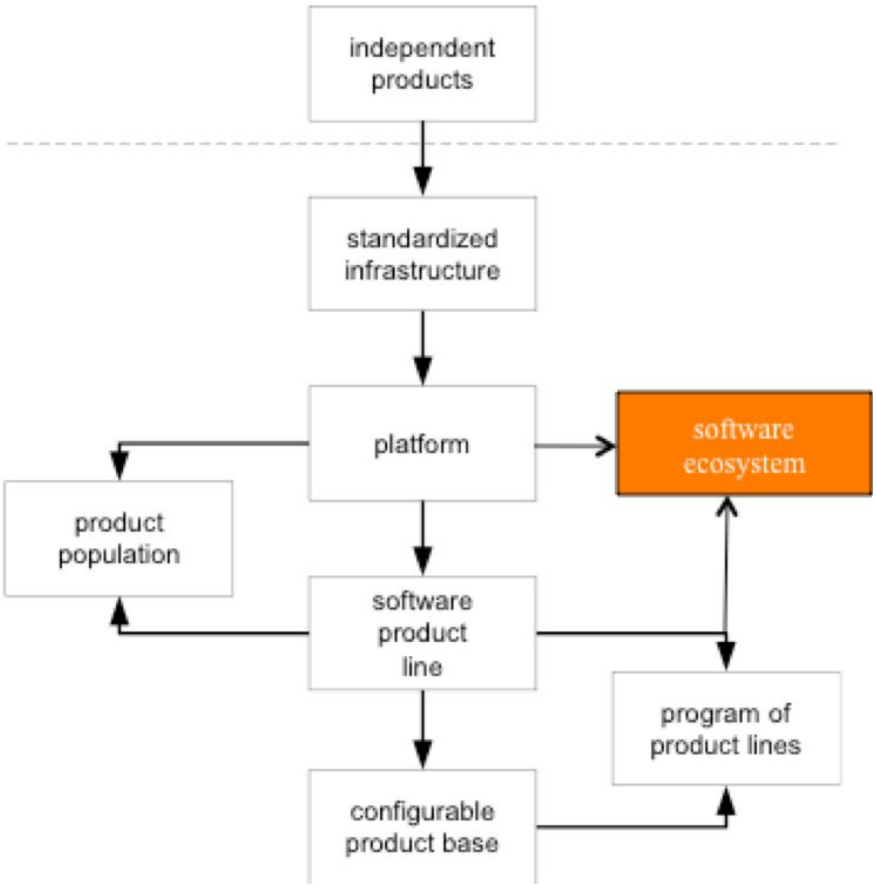


Figure: Model of evolution of reuse

Concluding, I realize that reuse, platforms and product lines easily are viewed as waterfall-ish, focused on requirements, planning and architecture work and consequently seen as non-agile. The point is I am trying to make here is that it's very hard to be agile when you're carrying around the anchor of tons of commodity software. Building on top of a solid platform providing the commodity functionality and focusing your energy on the differentiating functionality is a much smarter way to achieve agility, time to market and a high degree of effectiveness. So, stop wasting resources and do platforms already!

Stop Customizing Your System. Configure It Instead!

Although mass-market companies have figured this out long ago, companies offering their software-intensive systems to a smaller group of powerful customers are often under significant pressure to customize their systems for individual customers.

There are at least three reasons why customizing your system for each individual customer used to be a good idea. First, when closing the initial deal with the customer, the sales process typically requires a back and forth in terms of concessions and pricing. Offering customization of the software can be an effective mechanism to maintain pricing levels while giving the customer something that makes her or him feel special and unique.

Second, the customization itself often needs to be conducted by the company selling the system and is, of course, not free. Especially in situations where the company is organized in a central organization and market companies, providing the customization is a very effective approach for market companies to generate revenue.

Third, whenever there is a need to update the software in the system, the market company will need to repeat the customization for the new version of the software. In that sense, customization is a gift that keeps on giving from a revenue perspective.

When stepping back, it is obvious that from a narrow, short-term perspective, customization may make a lot of sense, but from a broader, long-term perspective it is a bad idea. Although this has been true in almost all cases over the years, more recently, the adoption of continuous deployment makes it even more important to stop customizing. Again at least three reasons why we should stop customizing.

First, it doesn't make sense from a financial perspective. In most companies, customization monetized through a markup on the salary cost of the engineers doing the work. Even if you manage to have a 100% margin, the revenue driven by these engineers pales in comparison to product R&D. In a company with an R&D budget that is 5% of the company's revenue, it means that every € spent in R&D has to result in 20€ of revenue for the company. Even if it may look good on paper to sell the engineer hours for a factor 2 of the cost, it is much less valuable than investments in product R&D.

Second, customization severely complicates continuous deployment as it requires the customization to be repeated for every release of the software. In most cases, this is prohibitively expensive for your company and your customers, resulting in a situation where customer will do everything they can to avoid upgrading their software. This, in turn, results in you being required to maintain many releases of the software, sometimes ranging over a decade or more.

Third, as I described in last week's blog post, R&D needs to shift from focusing on efficiency to focusing on effectiveness. That requires continuous deployment and a constant flow of data from the systems in the field to determine that the system as well as features are generating the value for customers. Realizing this fast feedback loop is impossible when every customer has a unique customization of the system.

Instead of customization, we need to adopt configuration. This requires us to architect the system in such a way that features that are used by some customers, though not all, are configuration items in the system. Customer unique functionality should be built at the other end of a stable product interface, so that the product can evolve continuously while the customer unique functionality can stay the same and work against the stable interface.

Concluding, if you customize your products and systems for each customer, it is time to stop this practice. Instead, re-architect your systems to provide the main points of variation as configuration settings and introduce a stable interface for customer unique functionality (the true customer-specific customization). Failing to do so keeps you stuck in the old ways of working, which simply is not competitive in the long run. You need continuous deployment and a quantitative data channel back from your systems in the field in order to maximize the effectiveness of your R&D.

Systems Engineering

Digitalization is concerned with software, data and artificial intelligence. However, it would be naive to assume that digital technologies have no relationship to analog technologies such as mechanical engineering. This is where the topic of systems engineering becomes relevant. Software intensive systems may contain significant amounts of software, but there also are other parts, built of atoms, that need to be taken into account.

The challenge in most companies is that traditionally software was considered to be supporting the mechanical and electronic parts of the system. This resulted in a number of consequences that hinder a digital transformation. First, in most systems, the development and release process follows the slowest part of the engineering cycle, which typically is the mechanical part of the system. This means that software, being able to evolve much faster, is slowed down to an unnaturally slow development and evolution cycle.

A second consequence, in my experience most system architects have a tendency to take a project-centric approach where it is the requirements of the current project that take the front seat and everything else is secondary. This tends to result in a situation where backward compatibility and consistency between different products and projects is highly limited and potentially vast amounts of non-value adding variability is introduced in the mechanical and electronic parts of systems.

Vast amounts of variability need to be incorporated into the software as most companies prefer a single version of the software for their portfolio of products. Although variability is fine if it adds value for customers, introducing variability that does not add value results in a significant increase in complexity and consequently cost without the company being reimbursed for these expenses. When not managed well, the result tends to be large, complex software that is difficult to test, leading to quality issues, and that is difficult to evolve, leading to unplanned delays in the development of systems.

The way to address this is to move to what I refer to as “systems engineering 2.0” where software is placed in the driver seat and where mechanics and electronics are subordinate to the needs of software. This requires, among others, the application of a principle that sometimes is referred to as “architectonics” where the layers of the system that evolve and change at different frequencies are architecturally decoupled through clearly defined interfaces. This requires that also the mechanical and electronics parts are designed with a platform approach in time. This means that also these parts are designed according to an architecture and adhere to interface definitions and backward compatibility.

In the sections below, the notion of systems engineering 2.0 and the implications of this transformation are discussed in more detail. The digital transformation affects everything in the company, not just the software folks!

Six Practices Transforming Systems Engineering

Many of the companies that I work with are embedded systems companies, meaning that the products, systems and solutions sold by these companies consist of mechanical, electronic and software parts. Traditionally, software was only a small part of these systems, which resulted in the system engineering process to be driven by the mechanical and to some extent the electronics. As mechanics and electronics are driven by atoms, rather than bits, this causes systems engineering to be slow and following a waterfall-style development process. As digitalization and software play an increasingly central role in all (interesting) systems, systems engineering needs to go through a fundamental transformation.

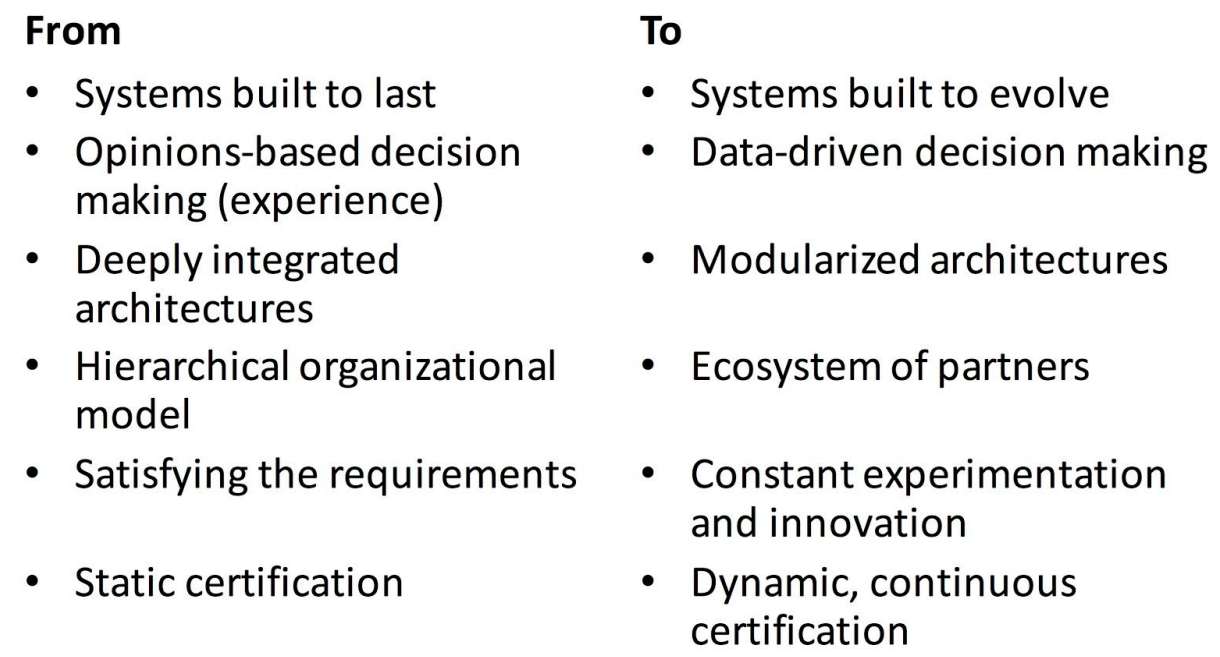


Figure: Six practices

As shown in the figure above, digitalization and software are driving major transformations affecting at least six practices or approaches. Below we discuss these in more detail.

#1 From systems built to last to systems built to evolve: Traditional systems engineering is slow and waterfall because the engineers assume that they're building a system that is required to last a lifetime. In fact, traditional systems in military, healthcare, transport, telecommunications, etc. are expected to last for decades with minimal R&D during the lifetime of the system. The first transformation in systems engineering is that we need to shift our focus to systems that are built to evolve, rather than built to last. This means that the systems architecture needs to allow for a constant flow of new software, electronics and even mechanical parts, resulting in a system that is continuously getting better.

#2 From opinion-based to data-driven decision making: In many organizations, there is an institutional memory and experience is valued. The problem is that the experience that senior leaders have accumulated tends to be years and sometimes decades old. In a fast changing world, experience easily becomes a double edged sword. It can help teams make faster decisions and avoid mistakes. At the same time, it can cause entire organizations get stuck in an outdated way of working and decision making. The best antidote to this is the use of data for decision making. In the words of Edwards Deming: In God we trust; all others must bring data.

#3: From deeply integrated to modularized architectures: When architecting a system that will remain static after deployment, the primary focus is on resource efficiency. Optimization of resources can often be accomplished by allowing for intricate dependencies between disparate parts of the system. Systems that are designed to evolve throughout their lifetime can not afford the high degree of dependency and needs to be modularized. This typically leads to some increase in resource usage, but especially electronics needs to be overdimensioned for the requirements at initial deployment anyway in order to allow for evolution of functionality. The electronics needs to be designed for lifetime cost and not for bill of materials.

#4: From hierarchical organization to an ecosystem of partners: Hierarchical organizations allow for top-down management of systems engineering and control from a central position. This leads to slow decision making and easily lets an organization get bogged down in the complexity of the system. Especially for evolving systems, the organizational model should be that of an ecosystem of partners, both internal and external to the organization, that operate in relative autonomy while governed by system architects whose only job is to avoid negative system implications due to local decisions by partners.

#5: From satisfying requirements to constant experimentation and innovation: Traditional systems engineering tends to be requirement centric and assumes a predefined specification of sufficient detail, internally consistent and without conflict. Of course, few realistic sets of requirements satisfy these criteria: if only because it is really difficult to envision a system before you have experienced it. Systems built to evolve allow some or most of the functionality to be developed after the initial deployment of the system, which allows for experimentation with customers and systems in the field, which facilitates innovation throughout the lifetime of the system.

#6: From static to dynamic, continuous certification: When systems constantly evolve after initial deployment, we need certification approaches that allow us to apply these principles also for the certified parts of the system. The current approach to limit the safety critical or certified part of the system to the minimum and interfacing this core with continuously evolving functionality deployed on top of it will only get us so far.

Finally, the main transformation is how functionality is allocated to different technologies. Traditionally, system engineers avoided software as it was poorly understood by many and

often viewed as a problem, rather than a solution. With the increasing role of software, system engineers need to adopt the principles as shown in the figure below.

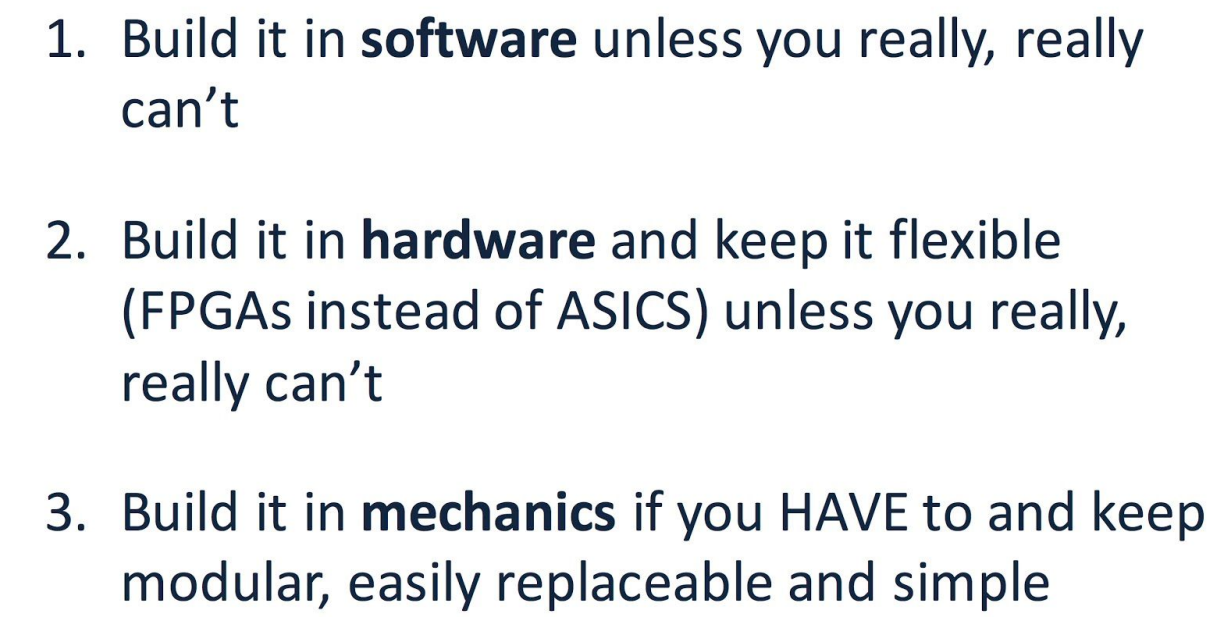


Figure: The implications for systems engineering

Concluding, in the digitalization age, systems engineering needs to transform to prioritize bits over atoms, evolution over static designs, modularity over deep dependencies and ecosystems over hierarchy. In many ways, this represents a fundamental shift in the way systems engineers operate, but it is a necessary one as the companies not adopting these principles are setting themselves up for disruption. If you want more, my INCOSE talk on youtube might be interesting to watch.

The End Of System Architects

The majority of companies that I work with provide products, systems and solutions to the market that include mechanical, electronic (hardware) and software parts. In order to design these systems, companies appoint system architects that are supposed to manage the system design, realization and evolution. Over the last month or so, I have run into the same problem at multiple companies: system architects don't understand software! Most system architects have a background in mechanics or electronics and try to treat software as if it's made of atoms instead of bits.

In fact, I suspect that many system architects have a dislike of software, developers and software architects and try to spend as little time with the pesky subject of software. The consequence of this attitude is that system architects tend to make really poor choices concerning mechanical and electronics architecture that cause negative and sometimes disastrous effects on software.

Some of the problems that I have observed in the various companies that I work with have to do with meaningless mechanical and hardware variants without business value, system architectures with high coupling, a complete disregards of backward compatibility during system evolution, lack of abstraction mechanisms and a general tendency to blame software for everything that is wrong with the system. Below I describe some of these problems in more detail.

Most companies support dozens if not more variants of mechanical parts and hardware architectures that have no or minimal business or customer value. Many system architects seem to prefer to design unique solutions for every physical product that they get to design. This leads to a large number of variants that all need to be supported by the software. There is significant cost associated with new physical parts and hardware architectures in and off themselves, but the resulting software complexity due to variant handling and the complex dependencies tends to be on par in terms of cost and time to market implications. We need to be much more business driven in our decisions and seek to limit the set of mechanical and hardware variants to the absolutely minimal set that facilitates the delivery of customer value.

Deeply integrated and highly coupled architectures cause numerous of ripple effects when changes occur. Many system architects are hell bent on minimizing the bill of materials for the physical part of the product and design lots of cross-system dependencies into the architecture as it allows for maximizing the performance of the hardware components. The consequence is that any change to any part of the hardware immediately has ripple effects throughout the architecture and software, depending on the interfaces provided by the hardware, breaks frequently and the maintenance cost are much higher than in the case of a system design that would incorporate the needs of software into its constraints.

The problems are exacerbated by the lack of understanding of the importance of backward compatibility. Rather than introducing a new interface while maintaining the previous version for a period of time, the introduction of a new version of hardware brings several breaking changes with it. And as the software will be expected to support all the versions of the hardware as well as be reusable across the product portfolio, the set of variants to support only grows.

The highly coupled, deeply integrated architectures also cause a lack of abstraction layers, forcing the software to compensate for the exact sensor types, actuators types, etc. in the specific product version. In some companies that I work with, the exact sensor, actuator and other hardware part needs to be exposed all the way up to the user interface! As if the user cares about the specific hardware part version and its capabilities!

Finally, several system architects have a tendency to blame software for everything that goes wrong in a project. After a development cycle where first the mechanical part of the system is designed and runs late, it is followed by a design of the hardware which then runs late as well. Then the software organization knows what to design for, but of course has very little time left due to the time overruns by the other groups. Still trying to do the best they can, the teams skimp on features and quality assurance in order to help the company meet its deadlines. The result is that software is always late and has poor quality. However, rather than blaming the software organization, it would be good to realize that the root causes of these problems originate upstream.

The only solution that I have been able to come up with is that we have to get rid of system architects. At least, we need to get rid of the system architects that have a mechanical or electronics background, rather than a software background. Instead, we need to promote software architects to the role of system architect. Software architects learn to deal with complexity through the use of abstraction and decoupling from day one in their career. In addition, software architects, in my experience, have a better understanding of the business than other types of architects. Finally, in most companies, the mechanics and electronics (the “atoms”) have commoditized and the differentiation has shifted to software (the “bits”). Hence, to maximize the value that we deliver to customers, we need to put the architect in charge whose expertise is closest to the things customers actually care about. Taking this perspective will allow us to reduce variants, freeze investment in mechanics and electronics that are commodity, find ways to outsource those parts of the systems engineering and to focus our resources on the differentiation provided by software.

Finally, the point I make in this article is rather strongly worded. The reason is that I see many companies and their employees agree with my line of reasoning, but failing to do something about it. Product managers feel that they victims of the system architects, the software folks have reached a state of “learned self helplessness” and business folks try to monetize the systems and products they get from R&D. The time has come to make a material shift in the way we think about systems engineering, who we appoint as systems engineers and how we

align systems architecture with the business strategy. We live in a software driven world and it's time we start acting like it!

Summarizing, in my view the solution to the challenge outlined here is fourfold:

- Promote software architects to the role of system architect
- Asses the system from a 3LPM perspective and treat commodity mechanics and electronics as such (reduce variants, freeze investment, outsource, etc.)
- Install business driven system engineering governance
- Decoupling interfaces between HW and HW close SW and all other software. Build for continuous deployment of all other software

Systems Engineering in a Service-Driven World

In an earlier post, I have brought up some of my frustrations with traditional approaches to systems engineering. That led to quite a bit of discussion and emailing back and forth with various folks. For those that did not read the post, my problem with traditional system engineering include at least the following concerns: (1) meaningless mechanical and hardware variants without business value, (2) system architectures with high coupling, (3) a complete disregard of backward compatibility during system evolution, (4) lack of abstraction mechanisms and (5) a general tendency to blame software for everything that is wrong with the system.

During the last months, in my engagement with various companies, I have been testing and discussing an alternative perspective on systems engineering, intended for a world in which systems are deployed as services and monetized through subscription. One of the main differences between a traditional product business and a services business is that the products created by the company turn from revenue and profit drivers to cost factors. For instance, when a car company converts itself into a mobility services company, the cars that used to drive revenue and profit now are costs that the company will seek to minimize.

When a company offers its products as services, the expectation of customers is that the offering gets better over time. Similarly, the company will seek to decrease its cost by deploying new software into the product frequently and measuring how the product is used as this helps to align internal R&D investments with the key differentiators for customers. And, of course, the longer I can keep a physical product out in the field while it delivers acceptable service for customers, the lower my cost will be.

One approach to systems engineering is to separate the mechanics and electronics from the software and operate both as two independent, though related, release processes. So, everything atoms is engineered in the traditional product-centric, waterfall style and everything bits is built iteratively and deployed frequently. This requires architecting a stable interface between the atoms and the bits, but once in place, the two engineering processes can largely operate in isolation.

Although this works and some of the companies that I work with operate in this way, the challenge is that some features require both atoms and bits in order to deliver value to customers. In many of my posts I take the position that in this new world it's the bits that are differentiating and the atoms are commodity. However, this is of course an oversimplification. The fact is that customers have "jobs" that they are looking for having realized by our product or service. As the saying goes: Nobody buys a drill for its own sake; The job is to get a hole in one's wall and the drill is bought to get that job done. To deliver solutions to the "jobs" that our customers have, it's not about bits versus atoms, but about the best combination of bits and atoms to optimally meet customer needs.

So, a second perspective on systems engineering is to think about a stream-based systems engineering approach where software, electronics and mechanics all are system elements that just happen to evolve and deploy on different timelines. In practice, software may deploy every day or every agile sprint. Electronics may evolve in a quarterly or half-yearly heartbeat, whereas mechanics may evolve yearly or every two years. In the figure below, I tried to graphically illustrate this.

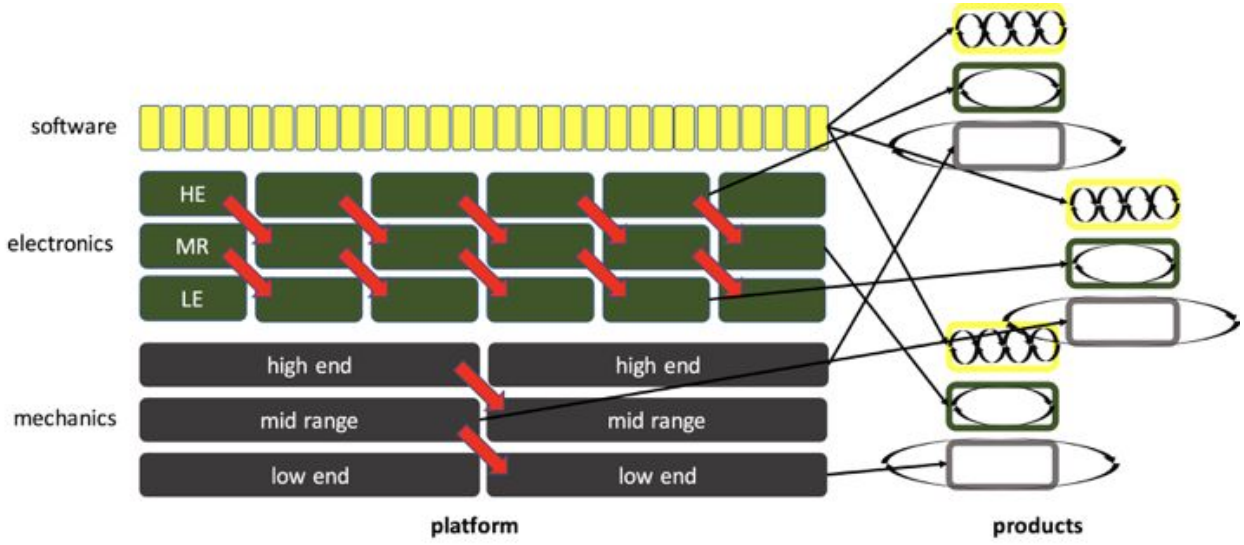


Figure: illustrating Systems Engineering 2.0

Products are initially composed by picking from available versions of the mechanical, electronic and software components available. Once deployed in the field, the software in these products is deployed constantly. Whenever the electronics in the product runs out of computing power or storage space, the electronics is replaced. This perhaps happens every 6-12 months. Finally, mechanical parts of the system may be upgraded every couple of years.

Although the model that I describe here is highly conceptual and needs a lot more details for a company to implement, it addresses the concerns that I have experienced with traditional systems engineering: mechanical and electronics variants are only created when there is business value and need to adhere to the system architecture interfaces and, consequently, are freely exchangeable. This focus on interchangeability forces a high degree of architecture decoupling at the system level. Also, it forces a culture where only the highest priority cross component dependencies are allowed, which requires a significant increase in abstraction levels on interfaces. Finally, as mechanics, electronics and software evolve in parallel and largely independent from each other, the blame culture can be replaced with a collaborative one where new functionality can be realized by mechanics and electronics and then gradually and iteratively be exposed to customers as a mechanism to make the service better continuously.

Concluding, the old perspective on systems engineering suffers from significant problems, especially in a world where products are increasingly provided as services. An alternative,

integrated perspective on systems engineering where each discipline operates in an iterative process, but with different heartbeats, provides a much better approach to align the different technologies in a system. Here, I only scratch the surface of this alternative approach, but shared at least some of the key characteristics. As always, let me know what you think? What are your experiences?

The End Of Architecting Systems

This week, I spent time at companies building fabulously complex systems operating in even more complex ecosystems. It is clear that the business leaders as well as the technologists in these companies are struggling with managing the complexity of the systems that they are selling to customers. Whether it is telecommunications, automotive, healthcare or any other industry, we clearly are reaching the limits of what humans can manage from an intellectual capability perspective.

At the same time, during this week, I also got to spend time with my friends at Peltarion, which provides platform for what we call “operational AI”. When exploring the outstanding progress that we’re making using machine learning systems in general and deep learning systems specifically, it is clear that these systems are not “built” or “architected”, but rather “grown”. The problems these systems solve are so complex that solutions that “designing” is not possible or prohibitively expensive.

As one approach, the field of complex systems and chaos theory has been around for a long time and everyone knows the story of the butterfly in Australia causing a hurricane off the coast of Florida. Rather than claiming that it is impossible to make any statements about complex systems, my main concern is that it is difficult to make accurate predictive statements about the behaviour of complex systems because of nonlinear relations between inputs and outputs, feedback loops and emergent behaviour in systems.

The interesting observation is that almost all our processes in companies, in industries and society as a whole are organized based on the principle that we can design and architect systems in a top-down fashion and that we are able to predict the implications of design decisions. For instance, in the certification community, the use of deep learning systems are considered to be highly suspect precisely because we can’t explain and predict the behaviour of the system, especially for input data that is outside the scope of the training data. Similarly, many laws proposed by governments often provide a specific, prescriptive solution to a specific issue but frequently fail to accomplish the desired outcome due to complex system behaviours.

The solution to this challenge is to focus on the desired outcomes rather than specific solutions. Once we’re clear on what we’re looking to accomplish in precise, measurable terms, it becomes much easier to experiment with different solutions in order to evaluate which of the proposed approaches result in the desired outcomes and which fail to do so. And this experimentation can be done by humans, as is the case with A/B/n experiments in, for instance, SaaS companies or through the use of machine learning algorithms when the data is available and training and validation can be conducted without negative side effects.

The final concern that I want to raise is that many companies lack a clear, explicit description of the desired outcomes. Many companies talk about system performance and customer value,

but few provide the necessary specificity for it to mean something. For instance, how does the average throughput for the entire customer base and the response time for individual customers compare. Or what is the right balance between security and usability? For us to grow systems through experimentation and trial-and-error, we need to know what constitutes better. And we need to know what are the guardrail metrics that can't be violated.

Concluding, in many industries we are reaching the limits of the complexity that humans can intellectually manage and it is important to change our perspective from the top-down architected and designed system to the complex (eco-)system where the consequences of our decisions are not obvious and we need to experiment our way forward through, potentially, multiple trials to achieve the outcomes that we look for. It is time to explore growing rather than building systems.

Speed

Few concepts are talked about so much and understood so little as the notion of speed in software-intensive systems. Many define speed as the ability of an organization to build as many features as possible in the shortest amount of time. As we have discussed earlier, we refer to this as an efficiency focus. The challenge with efficiency is that it is surprisingly easy to be very efficient at things that don't matter at all.

In this book, we focus the notion of speed on the length of the feedback loops that companies, their systems and their customers experience. As we will discuss in one of the later sections, we identify several feedback loops (at least six) in the context of software intensive systems. These feedback loops range from the agile development cycle to the value creation validation cycle.

Fast feedback loops (or speed) are important not for their own sake but because it improves the quality of our decision making. Rather than relying on opinions, it allows teams and leaders to make data-driven decisions. In the next chapter, we go deeper into data-driven decision making, but the key is to understand that speed is a critical enabler for any software intensive company.

The criticism towards speed and fast feedback loops often is concerned with an incorrect belief that speed and quality are mutually exclusive and that increasing one will automatically hurt the other factor. Although it is obvious that a company that employs continuous deployment will need a system for continuous integration and test to establish quality. However, a factor often ignored is that continuous deployment facilitates a very low cost of deploying new versions of software. This, in turn, allows the quality processes to seamlessly include post-deployment testing of systems in the field. Assuming that no business critical errors manage to slip through, for many types of errors can actually be detected post deployment against significantly lower cost and without causing problems for the customer.

Concluding, for basically any company, fast feedback loops and speed provide a critical competitive advantage. Adopting agile software development practices provide a first step on that road, but it is only a start and more is needed. In the following sections we explore the concept in more detail.

Five Reasons Why Speed Matters …

The term “agile” has received vast amounts of attention and adoption in the software industry and the term is now used in business contexts as well. Everyone understands the notion of sprints, but I notice that many do not really think about or understand why speed is so important. Although there are many reasons why speed matters, I wanted to share what to me are the top five.

First, the ability to adjust development priorities every sprint. The notion of working with sprints means that many of the activities around product development are conducted every two or three weeks. This means that product management or whoever sets the priorities for agile teams has the freedom to change these priorities every sprint. This does not mean that the direction can change completely every couple of weeks, but the relative priority of features and other work on the backlog can change every sprint and new work can be added.

Second, speed allows for accelerating the innovation cycle. The innovation cycle has many different incarnations, but at its simplest is concerned with the build - measure - learn (BML) loop. Originating in the lean startup community, the BML loop, as shown in the figure below, is concerned with building ideas, measuring the effect and learning as a means to generate new ideas.

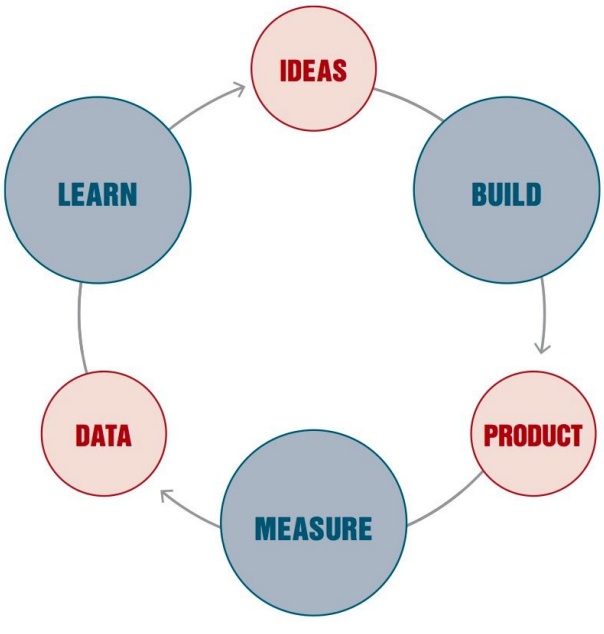


Figure: The Build - Measure - Learn loop

The importance of speed is concerned with iterating through the innovation cycle as fast as possible. To paraphrase the CEO of a wildly successful company: if we can learn twice as fast as the competition what works for customers, there is no way we can lose.

Shortening innovation cycles is also why we see R&D effort shift from mechanics and hardware to software. For instance, in the automotive industry, a car platform has a lifecycle of 10-15 years. That means that a typical OEM goes through a BML loop once every decade or so when it comes to platforms. A typical car project takes around 36 months, which means that the OEM goes through a BML loop once every three years or so. However, with the adoption of continuous deployment, the software in typical cars is deployed every couple of days. This is two orders of magnitude faster than any other technology and consequently vastly superior to anything else.

Third, speed gives teams confidence and satisfaction in that they're able to deliver customer value on a continuous basis. Being able to finish each sprint with functionality that delivers value in and of itself to the various stakeholders is of great benefit. Engineers like to build things, for sure, but they like to deliver value even more.

Fourth, speed allows for continuous feedback on the delivery of value. Frequent delivery of value allows the teams to collect feedback on the actual value created by the delivered functionality. When getting functionality out, we can also collect data about how the software is used by customers. This gives the team input about whether the features and the way the team realized the features is delivering on the expectations. It helps teams build understanding of the actual behaviour of customers.

Fifth, speed forces the organization to maintain quality and avoids the integration and bug fixing hell. In traditional development approaches, functionality first gets built and then the R&D organization starts to test the system in order to get the functionality back to the "production quality" levels that are required for customers. In agile development practices, especially when delivering frequently, we can't afford to drop quality as we won't have the time to get back to production quality.

The challenge, for many companies, is how to get to speed in place. For this, we have introduced the Stairway to Heaven (StH). Although the StH has three dimensions, the speed dimension captures the typical evolution path that companies evolve through in order to shorten their innovation cycles. In the figure below, the journey that most companies go through over time is shown. Starting with traditional development, the company adopts first agile practices, followed by continuous integration, continuous deployment and finally to allow for R&D to embody the innovation system for a company.

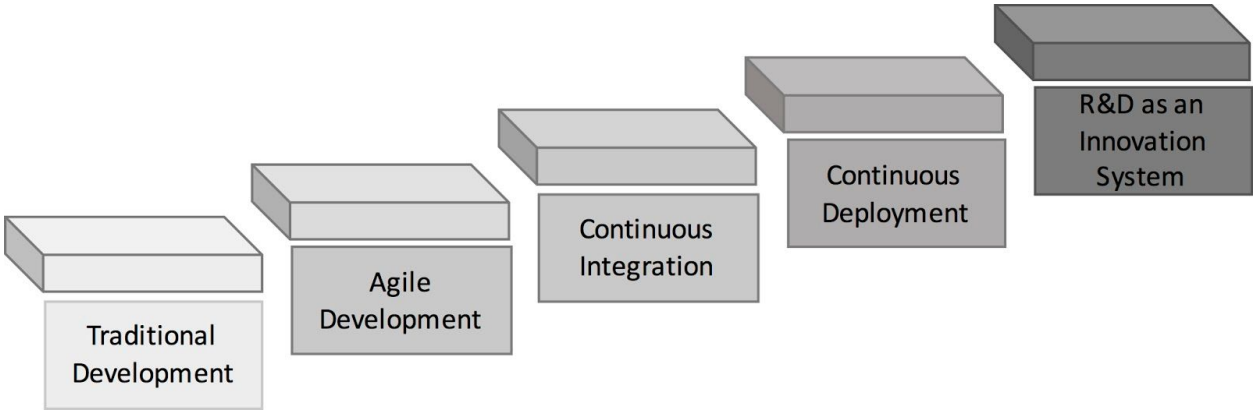


Figure: Speed dimension of the Stairway to Heaven

Concluding, speed matters because it allows us to innovate and deliver much faster than with traditional technologies and ways of working. Faster innovation is a key differentiator for companies and consequently it allows us to improve competitiveness. Finally, it also is an enabler for increasing the effectiveness of R&D and effectiveness as discussed in earlier posts (here and here) has significant room for improvement.

Speed versus Quality

This week I gave two keynote presentations at two different conferences (ICSOB 2017 in Essen and EASE 2017 in Karlskrona - please don't ask me how many days on the road I log every year). As part of my keynotes, I bring up our "Stairway to Heaven" model (see figure in the previous section) and the adoption of continuous deployment in almost every industry that I work with. Interestingly, at the end of both talks the same question came from the audience: how do we protect the quality of the systems deployed in the field when we adopt continuous deployment.

The interesting aspect of this question is that there is an underlying assumption in this question: if testing and validation is done slow and manual, the quality will be higher than if the testing is done automatically and fast. The second assumption is that quality is higher when I build the software for a system, freeze it, test the heck out of it and then put it in products during manufacturing with the intent of never touching it again.

When making the assumptions explicit, it is obvious that both assumptions are wrong. No amount of testing pre-deployment will be able to find each and every issue in each and every deployment of the system. Customers use systems in different contexts, configure it differently and use it differently. Setting up a test and validation environment that covers all permutations will be prohibitively expensive. This is also the case for safety critical systems, even if we set higher pre-deployment testing standards for these types of systems. However, any quality assurance system that incorporates the post-deployment phase inherently leads to higher quality than one that does not.

Concerning the second assumption, it is interesting to see the difference in approach between the software & system safety community and the security community. The latter will actively patch security holes through new software deployments when security issues are identified. The safety community will do everything it can to avoid deploying new software versions as the risk of introducing new safety issues is considered to outweigh the benefits of fixing a know safety concern and the cost of re-certifying are prohibitively high.

It is obvious that a validation and deployment infrastructure that collects information from deployments in the field, that identifies concerns before customers may even notice them and that deploys new software whenever quality concerns of any type are identified is superior to a system that fails to incorporate some or all of the aforementioned characteristics. I would even go as far as to say that we have a moral and ethical obligation to adopt this approach, especially in safety and security critical systems.

Certification institutes are in many contexts, ranging from medical to automotive and aeronautics, hindering innovation by causing major delays in the adoption of new technologies, approaches and methods. Their standard argument is that they protect human lives by doing so.

However, we fail to account for all the lives could have been saved if innovations would have been adopted earlier. Who is accountable for the lives lost due to delays in adoption of new innovations? How do we measure those?

So, next time you the need to grasp for the good old days when everything was milk and honey, where men were men and women were women and a product would be optimal at the time of acquisition and then slowly deteriorate over time as as it aged, remember that continuous deployment allows us to enjoy products that become better, safer and more secure throughout their lifetime. That any system that uses continuous integration, continuous deployment, monitoring of systems in the field and identification of post-deployment issues is inherently higher quality. And, finally, that speed drives quality, rather than being detrimental to it.

Towards Testing After Deployment!

During many of my presentations as well as during meetings with companies, the topic of quality comes up. As I stress the importance of speed, continuous integration and continuous deployment, a general unease settles over the group until someone brings up the topic of ensuring quality. Frequently this is followed by a couple of anecdotes about things that went horribly wrong at some customer. And then, finally, there are some people that bring up the topic of certification and use this as an excuse why none of the concepts apply to them as they are different.

In my work on the CIVIT model (see figure below) as well as other models together with industrial PhD candidates, it has become more and more clear to me that continuous integration goes a long way towards addressing quality issues. The notion of always being at production level quality and never letting the quality drop allows for a different way of operating in software development. In many embedded systems companies, software is viewed as the problem child because it is always late and experiences quality issues. Of course, in most of those companies, senior leaders conveniently forget that the delays often started in mechanical design and hardware and that it is impossible for the software to be properly tested until the actual system is available. Software is expected to eat the delays of the other disciplines and then make sure that the delivery date is kept. That leads to quality issues as the quality assurance phase gets squeezed the most. With continuous integration, assuming sufficient access of the system, you can at least maintain the quality for the current feature set.

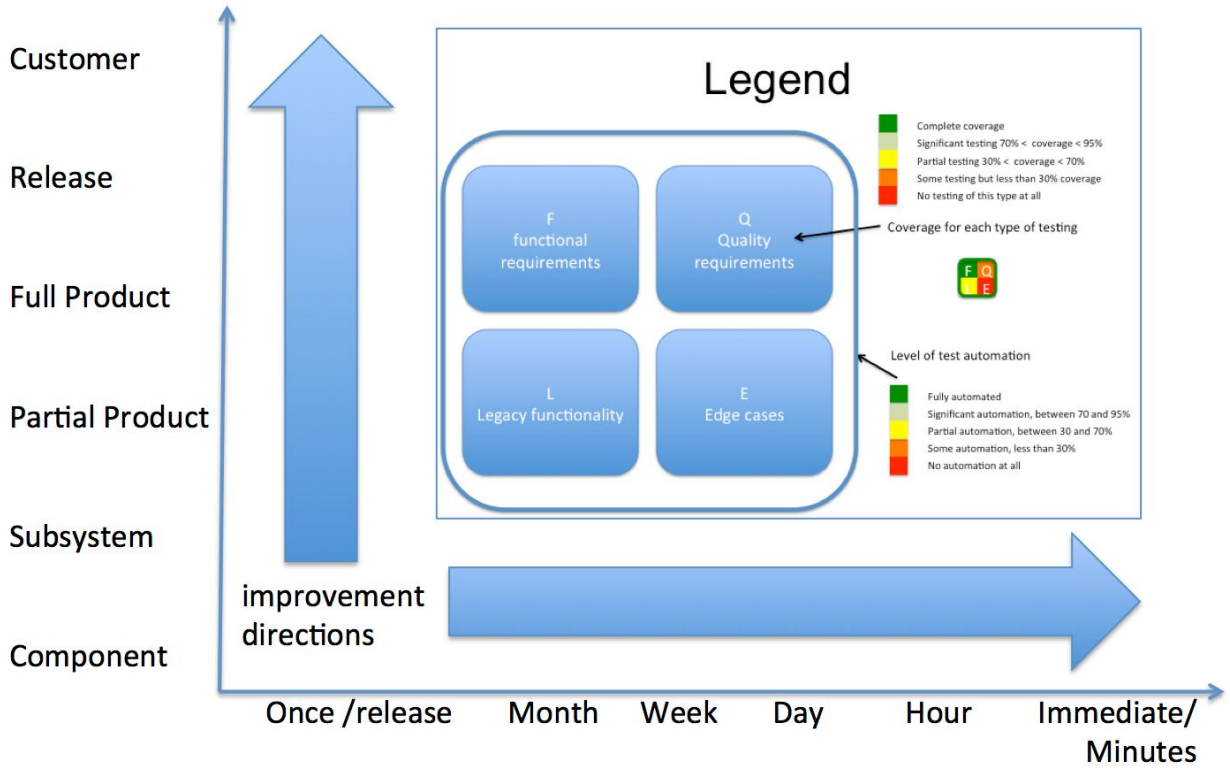


Figure: The CIVIT model

Even in embedded systems companies that are very good at continuous integration and potentially even continuous (or at least frequent) deployment, there still is a lingering culture where not a single issue is allowed to slip through to customers. In my experience, with a continuous integration environment in place, the equation actually changes: it becomes an economic decision to allocate a certain level of resources to pre-deployment testing. The organization can decide what an acceptable level of issues in the field is, considering the economic implications. Especially when a system can be deployed in a wide variety of different configurations, the goal of setting up testing environments for all configurations is an extremely costly ambition. This is not an artificial example - the companies that I work with typically have hundreds to millions of configurations for their successful systems.

A second challenge is that, typically, R&D departments feel that their responsibility ends with the release of the software and in many cases, teams have little insight into how a version of the software is performing in the field. There are no incentives to look at the data and R&D only looks at post-deployment performance when customer service rings the alarm bells. Of course, in true continuous deployment contexts, the concept of DevOps addresses this, but when releases are less frequent and there are separate release and customer service organizations, R&D teams tend to focus on the next release and its functionality rather than today's hassles. Especially as it's certainly more fun, according to most!

Even in the embedded systems world this has to change due to the new context in which we're building and deploying software. One of the consequences of continuous deployment is that there typically is a rollback mechanism in place as well as mechanisms to track the behaviour of the system. Consequently, it is becoming more and more feasible to push certain testing activities to the post-deployment stage. Testing the software after it has been deployed at customers is a big no-no in many organizations, but it's a logical consequence of continuous deployment! It is much more cost effective to try new software out in a concrete configuration in the field and rollback in case of issues, than it is to test every configuration in the lab. And with the ability to detect the need to rollback before the user even notices a problem, it does not affect the customer's perception of system quality!

So, my call to action is to start considering post-deployment testing as an integral part of your testing infrastructure and to make careful decisions about when to test what aspects of the system behaviour. Even if it seems like sacrificing quality, it is in fact an approach to improve quality. Quality is experienced in the field and not in test labs. Consequently, the best place to determine quality is in the field. Don't be afraid of post-deployment testing; rather embrace it!

Why Fast Feedback Cycles Matter

When studying software-intensive systems companies, one of the interesting observations is that they all evolve in the same way. In earlier research, we have referred to this as the "Stairway to Heaven". In figure <earlier in the chapter>, the speed dimension of the Stairway to Heaven model is shown. This model is a descriptive model based on our research with dozens of companies and it follows a logical sequence of activities. First, starting as a classical company, the organization adopts agile practices. Subsequently, continuous integration is adopted and rolled out in the company. Once continuous integration and test have established a situation where all software is always at production quality, the company will move towards continuous deployment. Finally, it will start to use its deployed products for innovation purposes, for instance through A/B experimentation.

Although one can easily argue that companies evolve through these levels, the question one should ask is WHY this happens in the first place. Our research has shown that this is because companies benefit from shortening the duration between decision and the feedback on the outcome of the decision. In general, the shorter the feedback cycle, the more accurate the decisions can be as a shorter feedback cycle allows for a more rapid learning loop. In the case of software-intensive systems companies, we have identified six feedback cycles that are sped up when climbing the Stairway to Heaven:

- **Development**: When adopting agile development practices, the first principle is that the team works in sprints of four weeks or less. This means that at the end of every sprint there will be working software. This leads a development cycle that is much shorter than during waterfall development as all work in progress is wrapped up into complete code every sprint.

- **Requirement**: The second practice when adopting agile development is that teams conduct sprint planning. This means that for every sprint, there is an opportunity to reschedule requirements and to change the content for the next sprint. This is a fundamental difference from traditional release planning where release content is defined and agreed upon upon freezing it until the release of the system.
- **Quality assurance**: Continuous integration and test allows for much faster feedback on the quality of the system under development as compared to traditional testing approaches. In addition, once the company adopts continuous deployment, internal quality assurance is complemented by quality assurance from the field.
- **Governance**: Every R&D organization has at least three types of activities, I.e. feature development, defect management and refactoring. Even with the most advanced CI system, there will be defects that slip through and that need to be managed and repaired afterwards. Similarly, the design of the system will erode over time and require investment to maintain an acceptable design quality. Operating in short cycles allows the R&D organization to dynamically reallocate resources on short notice and for short(er) periods of time.
- **Deployment**: When adopting continuous deployment, the basic benefit is that our software is rolled out frequently, which provides feedback on the quality of the software. It also provides a solution to the problem where finished features are kept “on the shelf” for far too long without providing benefit for the customer as well as the company that built them.
- **Value creation**: The final cycle is concerned with confirming that the value that we predicted would be delivered by new functionality indeed is created. For instance, a new feature may improve some quality attribute or change user behavior in a desireable way. When using a traditional release model, this feedback becomes available often years after the decision to prioritize the development of the feature has been taken. When using continuous deployment, we have access to this feedback in a small number of weeks.

In the table below, we related the stage of the Stairway to Heaven model to the feedback cycles that are shortened at that stage. The key reason why shorter feedback cycles are so important is that companies take *opinion-based* decisions when the feedback cycles are long as the relation between a decision and its effects is too far separate in time. When feedback cycles are short (such as one sprint), decision processes also become data-driven. In earlier articles, I stressed that for a typical software-intensive company, half of all the features built are waste in that the provided value does not justify the R&D effort that was needed to create the feature. Shorter feedback cycles provide a very powerful solution to reducing that waste.

	Traditional	Agile	CI	CD	Inno System
Development	Long	Sprint	Sprint	Sprint	Sprint
Requirements	Long	Sprint	Sprint	Sprint	Sprint
Quality assurance	Long	Long	Sprint (internal)	Sprint (external)	Sprint (external)
Governance	Long	Long	Sprint	Sprint	Sprint
Deployment	Long	Long	Long	Sprint	Sprint
Value creation	Long	Long	Long	Long	Sprint

Concluding, shorter feedback cycles allow companies to transition from opinion-based to data-driven decision making. That allows allows for a step-function change in the quality of the decisions that are made which, in turn, improves the competitiveness of the company significantly versus competitors that are stuck in traditional ways of working and opinion-based decision making. Climbing the Stairway to Heaven is not a “nice to have” for the R&D organization but rather a critical factor for maintaining the competitiveness of the company.

How Agile Are You Really: Let's Find Out!

In an earlier blog post, I raised the question of how agile organizations really are. This was based on my experience with a variety of companies that claim to be agile, but that, when taking a closer look, were not agile at all. Or that were using some agile practices, but failed to reap the real benefits that employing an agile way of working might bring.

The same is true when it comes to continuous integration and testing practices. Or when it comes to architecture practices, deployment practices, customer involvement, data-driven development, ecosystem engagement, etc. There often is a significant gap between how companies claim they are operating and the actual reality in the R&D teams. And in many companies, there is a pattern that by saying something often enough, people start to believe it.

The challenge of course is how one gets an objective and accurate overview of the state of R&D. Performing code analysis is beneficial for some purposes, but it doesn't allow for these types of analysis. An alternative is to get an external consulting team or expert in who interviews some of folks in the organization, studies how work is performed and then provides an "expert opinion". The problem with the latter is that this tends to be very expensive, uses only a thin slice of the organization as a basis for the analysis and often still contains significant bias.

What we would really want is an approach that allows us to measure objectively and accurately, gives everyone in the company a voice and is based on decades of research. Well, guess what? That is exactly what I have been working on with my team over the last years.

Based on the research around R&D practices that we published both in books, such as these, as well as in research articles, over the last years we have developed as configurable survey that allows you to assess the relevant aspects of R&D in your organization. This gives you a quantitative insight into the R&D topics that you care about. In addition, by running the survey again at a later point in time, you can track progress. Third, it allows us to compare the different sites within the company. Finally, we offer the opportunity to compare yourself to similar companies as we have data from several companies already available. And of course, you don't just get the numbers, but also concrete suggestions on how to improve your R&D organization.

Some examples of the results of the survey include graphs like the one below. Of course, the survey also provides more detailed insights such as insight into the actual deployment frequency as shown in the second figure below.

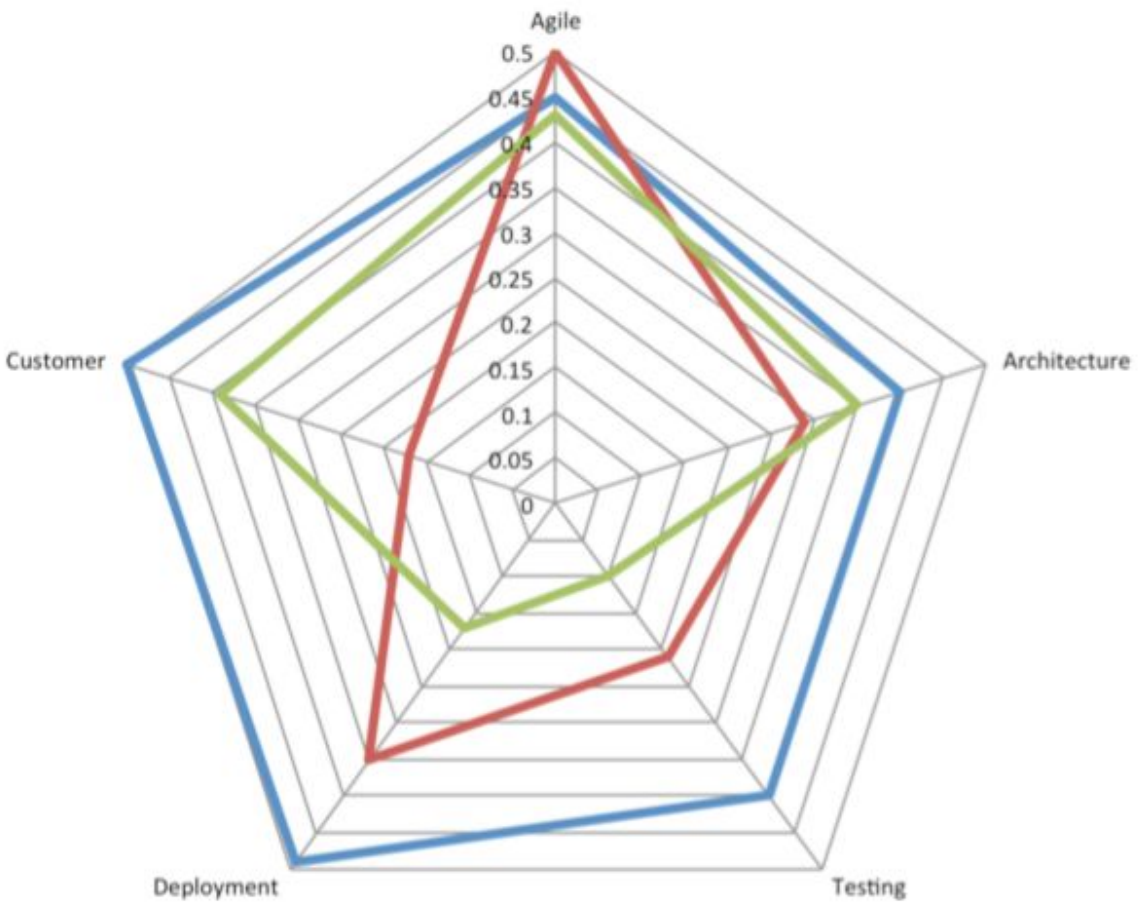


Figure: Assessment of five R&D practices

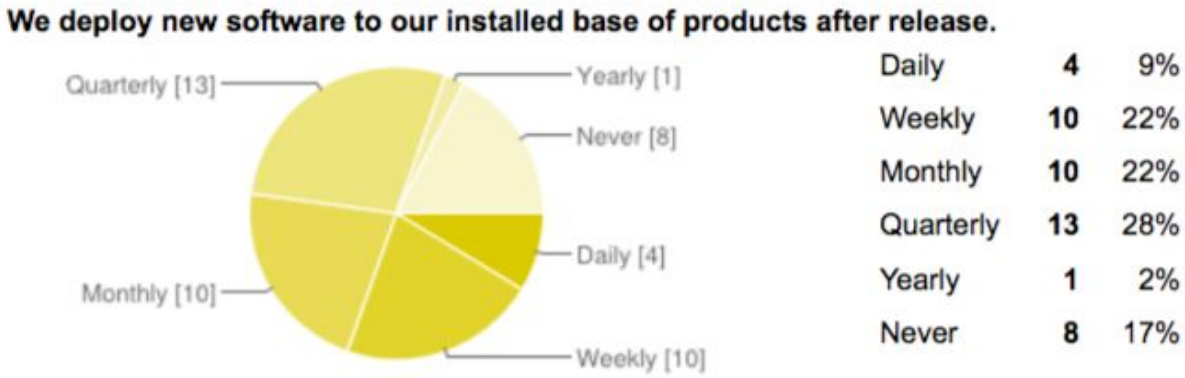


Figure: Assessment of deployment frequency

Concluding, many companies talk about their advanced R&D practices, such as agile, but when studying the cases in more detail, it often turns out that reality does not meet perception. We're offering you a configurable survey where you can select some or all of the following R&D practices for evaluation:

- Agile practices
- Architecture practices – e.g. technical debt management
- CI and testing practices
- (Continuous) Deployment practices
- Customer involvement practices
- Data-driven development practices
- Ecosystem engagement practices
- Organizational empowerment practices

This allows you to get ***quantitative*** feedback insight into the state of your R&D practices as well as tangible recommendations for improvement. Sounds good? Send me an email for more information.

Stop Complaining About Agile

The last months, I have started to see an increasing number of articles complaining about agile software development. Many of the articles have a similar tenure. On the one hand, they stress that agile is causing teams to be really stressed and constantly focusing on the next feature to deliver. And on the other hand, these articles sing the praises of traditional waterfall development and highlight how teams and individuals had time to think, properly architect and test systems and life was much easier.

First of all, the depiction of waterfall development in these articles is far from my experience with a host of companies. The final integration stage of a waterfall project where everything comes together, nothing works and stress levels go through the roof is an experience that almost everyone who has been in the industry for a while will recognize. It's just that we tend to conveniently forget about the bad memories from the past.

Second, most people forget the continuous growth of the size of software in our industries. According to research, the size of the software in systems goes up with an order of magnitude every five to 10 years. That means that if you were building a 1MLOC system a few years ago, you'll soon be building a 10MLOC system. And the architectures, ways of working, tooling and organizations that were already struggling with building the 1MLOC system are unable to build a 10MLOC system. Hence we need to constantly reinvent ourselves. And maybe you had a good experience with waterfall development in the past, but this concerned a system that is a fraction of the size of the system you're building currently.

Third, and the most important thing that almost everyone forgets: it's not about agile software development. Agile software development was never the goal to begin with. The real goal is *business agility*. Although I have written about this in an earlier article, it bears repeating: the goal is to achieve a situation where the business can pursue new business opportunities instantly. Stop investing in ongoing projects that no longer are as relevant instantly. And to use fast feedback loops to know what customers use and gives value and what does not.

In a fast changing world, responding slower than the market is a recipe for failure. As Jack Welch at some point said: If the rate of change on the outside exceeds the rate of change on the inside, the end is near. We need agile software development. We need continuous integration. We need continuous deployment. But we also need agile systems engineering. Agile product management. Agile business leadership. We need the entire company to be agile as learning from and responding to the market and customers faster than the competition is THE critical differentiator that companies have.

Concluding, stop complaining about agile. It is the best approach we have to maintain and improve competitiveness. Longing for the good old days when engineers were happy and had the time to do a proper engineering job is useless at best and counterproductive at worst. If

only, because the good old days never existed in the first place. Instead focus on true business agility where software R&D may lead the way, but where the entire company needs to organize itself around sprints, fast feedback loops and cross-functional teams in order to deliver value to customers continuously.

Data & Artificial Intelligence

With the ever increasing connection speeds, ability to instrument software and, according to some companies, soon 50 billion connected devices on the planet, the recording, analysis, storage, processing and use of data will continue increase exponentially even if we, in many ways, already now are drowning in data.

In our research, we basically see at least three uses for data. First, it offers the opportunity to significantly increase the effectiveness of R&D organizations. Rather than building the features requested by product management top to bottom, we can adopt an iterative development approach that allows us to release features a slice at a time and then measure the impact of each slice in order to ensure that we are allocating our R&D resources to the activities that result in the highest return on investment. Also techniques such as A/B or split testing use this data concerning customer or system behavior to determine the desirability of alternatives.

The second use is concerned with artificial intelligence. Machine learning and deep learning models require significant, if not enormous, amounts of data for training and validation purposes. Depending on the technologies deployed, the data should be labeled in order to provide opportunities for training or the systems should allow for measuring the result of decisions, recommendations or classifications made by the ML/DL solution in order to allow for reinforcement learning. In any case, the notion of data is a critical component in this use.

The third use of data is concerned with empowerment of teams. When data becomes available in real-time, in a quantitative form and it is of high quality, rather than telling teams what to do, we can give teams quantitative outcome metrics to achieve, such as improved conversion, reduced number of unnecessary alarms raised by the system, increased throughput of the system, etc. When teams gain responsibility for output metrics and the teams take this responsibility, the teams can become significantly more empowered and much less driven by managerial oversight and micromanagement.

Concluding, we define digitalization as software, data and AI. The sequence that most companies take is that they start with getting good at software. This means adopting agile practices, continuous integration and over time continuous deployment. Once the latter is in place, the next step is to collect quantitative data in real-time from the field with the intent of, initially, taking data-driven rather than open-driven decisions. However, with the increasing size of data and the consequent difficulty of gaining value through human analytics, companies seek to use automated solutions to learn from the data and the most effective solutions we have are machine learning and deep learning.

How To Stop Building Useless Features

Although there are still companies out there that are going through the agile transformation, most of the companies that I work with have adopted agile practices at least at the team level. Agile teams are more empowered and because of that, among others, more productive and more motivated. However, if we lift our focus up from the team level to the R&D organization and we include product management, architects, R&D departments, verification and validation departments, release organizations and customer support, the adoption of agility at that level is sorely lacking in many companies. At that level of R&D, yearly release cycles, aligning software release with the release of mechanical and electronic products and other “aim for a goal in the far future” waterfall-ish approaches are still the norm.

For all the chest beating that many companies do concerning their agile teams, in many ways very little has changed if we look at the end to end product development process. The reason for this is that most companies are focused on efficiency. They are concerned with getting as many features built per time unit, measure and increase flow, etc. In many companies, R&D is still view as a machine that we throw requirements into and out come products or at least the software that makes it into embedded products and systems. And if teams feel that by working agile, they can crank out products faster or with more features, the rest of the organization is willing to indulge them.

The problem with only adopting agile at the team level and focusing on efficiency is that it is based on a false premise: it is based on the belief that more features will make the product more valuable for the customer. In an earlier **blog post**, I have discussed research by us and others that shows that half or more of the features in a typical product are hardly if ever used. As a consequence, in another **blog post**, I share that in many companies 80% (or more) of R&D resources are spent on commodity functionality. The challenge for most companies is not to get more features out, but to get the minimal set of most valuable functionality embedded in the product. In the words of Antoine de Saint-Exupery: “A designer knows he has achieved perfection not when there is nothing left to add, but when there is nothing left to take away.”

Especially in B2B companies, the standard argument against my line of reasoning is that the customer asked for the feature. This is often followed by the argument that although perhaps some customers are not using it, other customers did ask for it. The fallacy in this argument is that although it is important to listen to what customers say, in the end the real proof is in the actual user behaviour: do users actually use the feature that they said they wanted. In most cases, there is a major, if not huge, gap between “espoused theory” (what people say they do) and “theory in use” (what people really do).

For most of the 20th century, companies did not have the means to measure user behaviour, but with every product that has some inherent value becoming connected to the internet, our ability to measure customer and system behaviour has become orders of magnitude easier. The

ability to measure what customers do instead of what they say they do allows companies to finally shift from a focus on **efficiency** to a focus on **effectiveness**.

Effectiveness is concerned with maximizing the amount of customer value created for each unit of R&D resources. Obviously, building features that are hardly ever used by customers is not a very effective use of R&D resources. Instead, we need a process where we first collect qualitative customer feedback that makes it likely that a new feature will deliver value. Based on that we build a slice (maybe 10%) of the feature and get it out to customers and measure system and customer behaviour. If the behaviour is not in line with expectations, meaning that it is not providing value, the feature can and should be cancelled and removed from the system. If the feature does deliver value, the team can iteratively add more functionality until the delivered value plateaus and then end development of that feature and move on to the next. In a research paper (reference below), we described this process that we called the HYPEX model.

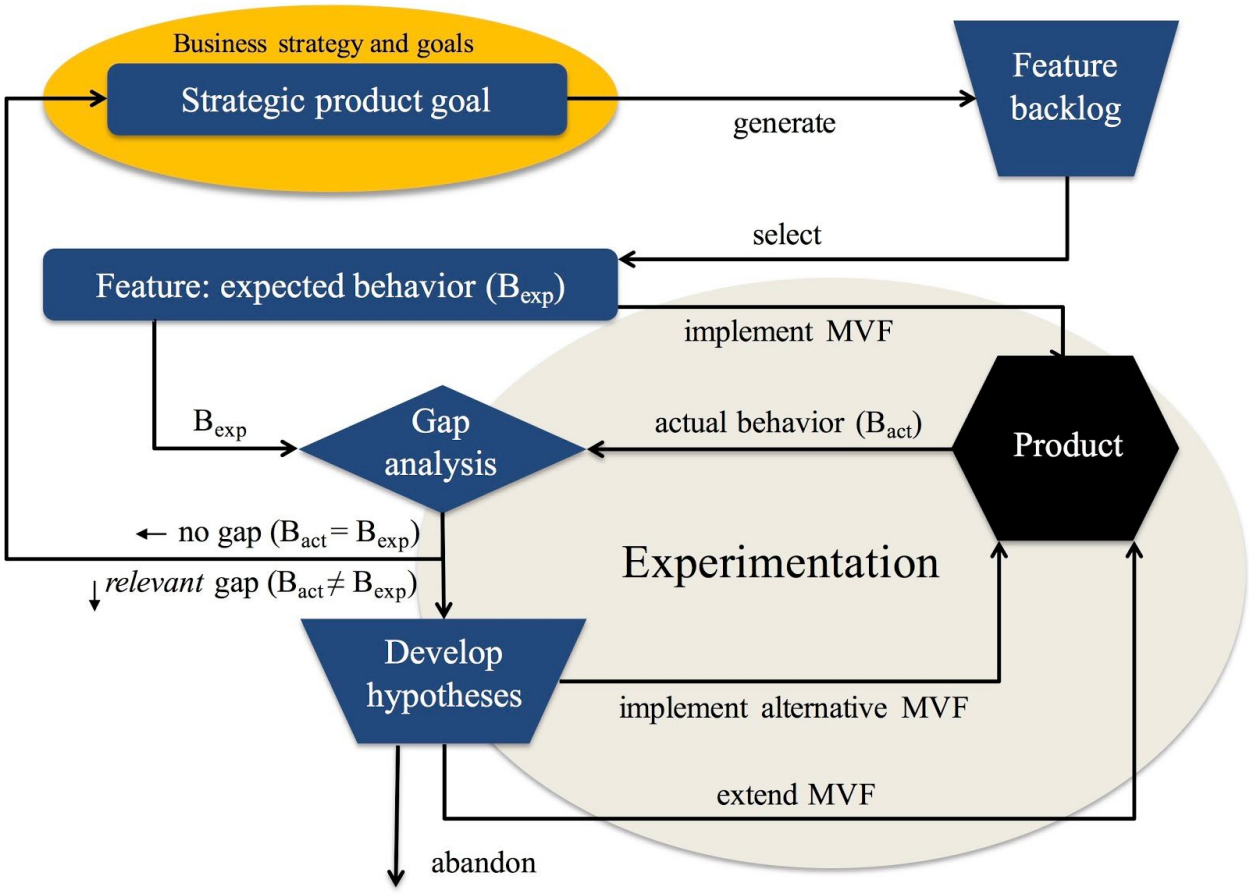


Figure: The HYPEX model

Concluding, for organizations to become truly agile and to transition from a focus on efficiency to a focus on effectiveness, it is not sufficient to have agile teams. Instead, the entire organization needs to operate in an agile fashion, support continuous deployment and enforce data-driven decision making. Once we have data concerning customer and system behaviour,

we can measure whether we are delivering value or not. This, in turn, allows us to stop development of features that do not, remove features that are never used and double down on the features that add most value to customers. To stay competitive, your best bet is to adopt agile across your organization, not just in teams, and to focus on effectiveness, rather than efficiency.

Reference: Olsson, Helena Holmström, and Jan Bosch. "From opinions to data-driven software R&D: a multi-case study on how to close the'open loop'problem." *Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on*. IEEE, 2014.

Minimize Investment Between Validation Points

Reflecting on software intensive companies during the summer break made me think about the way investment decisions are made and what new approaches, such as continuous integration and deployment as well as data analytics, mean for these investment decisions. A key principle that I learned while working in Silicon Valley is to minimize investment between validation points. So, what does this mean? It means that whenever an investment decision is taken, the focus should be to invest as little as possible until you can validate that the investment is really resulting in the desired outcome.

This principle has a number of components. The first is that you are crystal clear on what the desired outcome is. Not just as an individual, but for the organization as a whole. I have been involved in many feature prioritization sessions as well as new product initiatives and the process very easily turns political. When it turns political, people defend or attack features or new products based on their implicit beliefs, but these beliefs often are not actually aligned with others in the organization. The result being the formation of coalitions that seek to successfully get a feature or new product approved without any internal agreement within the coalition as to the expected and desired result of the initiative.

Associated with the first component is the demand that whenever initiating a proposal for an R&D investment (or any investment for that matter) is to specify, in detail and quantitatively, what the expected outcome is. If a new feature is expected to increase system usage, that should be measured. If it is concerned with increasing system performance, than this should be measured. We refer to this as the value function: an equation consisting of a weighted set of factors that clearly expresses the value expected from the feature or product.

The second component of the principle is that we need a mechanism for fast deployment of partially realized functionality, including the instrumentation to measure the factors identified in the value function. Interestingly, in many companies, there is a natural hesitation and even active resistance towards continuous deployment and the collection of quantitative data. One of the predominant reasons is that, in many cases, it will show that the emperor has no clothes. All kinds of beliefs held in the organization will be shown to be invalid. As a consequence, people will look to avoid being exposed and their position affected negatively.

The action associated with this component is that the organization needs to undertake is to adopt full transparency of all the data collected from its systems and products. Only by full transparency, careful analysis of results and an agreed interpretation of the results will the organization become truly data driven. Once this is the case, the company can become a true learning organization where people formulate hypotheses and then test these in order to learn more about their customers and the systems that they provide to these customers.

The final component is a willingness to change course based on the data collected. To proceed taking small steps and adjusting based on the feedback after each step. For traditional, hierarchical organizations, this is notoriously difficult. Achieving alignment between the different functions and hierarchies in the organization tends to be extremely effort consuming and the thought of going through the process every couple of weeks feels like an impossibility. In many organizations, the yearly plan will be executed come hell or high water.

The final action is for the organization to decentralize decision making concerning R&D investment and to assign this to cross functional teams that have a clear and agreed definition of the desired outcome. When a team knows what value function it is optimizing for and it can get frequent, quantitative and accurate feedback on its efforts, it does not need a management hierarchy to take the decisions to move it closer to the desired outcome.

Concluding, organizations have traditionally taken investment decisions where many resources were allocated per decision. Instead, we need to move towards small investments and frequent validation of the impact of these allocated resources. One company I worked with had a yearly release process and allocated hundreds of person years based on the yearly plan and roadmap. After adopting this principle, the company would never invest more than 10-20 person weeks of effort before validating with customers that the direction that each team was working on was relevant. Adopting the principle outlined here will make your R&D investments at least twice as effective. Imagine what you could accomplish if suddenly half of your resources were available to do useful work! So, adopt the principle of minimizing investment between validation points.

Towards Data-Driven Decision Making

This week I worked with two very different companies on what started out as very different topics, but that ended up at the same place: bad decision making due to the reliance on experience, beliefs, customer insights sometimes decades old, etc. So, the discussion turned towards decision making and how to develop processes where data is an integral part of the decision making process.

The interesting thing is that decision making is perhaps the key activity that people get paid for and that justifies their presence in the organization. From the CEO setting strategy for the company to the developer deciding the prefered design for part of the system, we take decisions all the time. The question of course is what these decisions are based upon. For all that we like to think of ourselves as rational and conscientious people, the fact is that most decisions are habitual and are taken subconsciously.

According to Charles Duhigg's book titled "The Power of Habit", we spend the majority of our days in purely habitual behaviours and take very few conscious decisions. As humans, we are of course able to change our habits, but as many who have tried to do can attest to, this change not without a significant amount of effort. Organizations, consisting of several if not many humans, are even more habit driven and change is even harder as it requires everyone involved to agree to breaking a habit. Nevertheless, the goal should be to apply rational, conscious decision making to as many decisions as possible and the question remains: what do we base decisions on?

In my experience, there are broadly four sources for decision making. First, there are shadow beliefs. These are beliefs that everyone in the organization holds and these beliefs make certain decisions extremely easy and defendable as everyone in the organization "knows" that the beliefs underlying the decision are true. The problem, of course, is that shadow beliefs held by organizations may have been true at some point in the past, but most certainly are no longer true today. So, basing decisions on shadow beliefs is a recipe for failure. Decisions get made based on a fundamentally incorrect set of beliefs.

The second basis for decision making is your opinion. Your opinion is formed based on your background, education, interaction with customers, experience and the set of beliefs that you have accumulated while walking the face of the earth. Basing a decision on your opinion is superior to using shadow beliefs as you own the decision and spend at least some time consciously considering the decision.

The third basis for decision making is qualitative data. Qualitative data tends to be characterised by small sample sizes where each case is studied in significant detail. This is the domain of ethnographic researchers and design thinking experts, but everyone who spends time with customers builds up, at least informally, a semantically rich set of qualitative data. Qualitative

data is more objective than opinions and shadow beliefs, but is still based on significant amounts of interpretation. However, especially in cases where you're making future oriented decisions, extrapolating from deep, qualitative insights might be your best option.

The final foundation for decision making is quantitative data. Although this type of data comes in many forms, the basic model is that we can derive information from the relationship between two variables. The data provides the basis for deriving validating insights from your customer base and often provides the ability to predict or at least validate the effect of decisions. Of course, in the online world, one of the best known applications of the quantitative data is A/B/n testing where two or more alternatives are evaluated by exposing each to a subset of customers or systems in the field and measuring the behaviour of customers or deployed systems in order to determine the superior alternative.

It is clear that as a basis for decision making, quantitative data is superior to qualitative data. And both types are superior to opinions and shadow beliefs. However, in practice many decisions that get made in the companies that I work with are based on shadow beliefs and opinions. Based on my experience, there are at least three reasons for that:

- **Lack of data**: Even though it has never been easier to collect data, for a variety of reasons, many companies are still reticent to do so. The excuses differ widely including concerns about EU regulation on personal data, overly zealous data security officers who prefer not to collect anything, the perceived lack of a business case, etc. However, if data is not collected, it can not be used for decision making.
- **Conflict with shadow beliefs**: Whenever there is a mismatch between data and the beliefs held by the organization, there will be attempts to rationalize why the data is wrong. For a human, it is always easier to reinterpret the data rather than change your belief system.
- **Long feedback loops**: Even if the organization has data, a typical situation is that the deployment cycle is so slow that whatever data is available, it is based on systems that were deployed perhaps years ago and consequently the available data is no longer relevant. Similarly, in cases where companies have not adopted continuous deployment, the organization has no mechanisms to experiment with or test decisions before making them final. In these cases, there is no opportunity to make data-driven decisions.

The conclusion and take-away from this article is twofold. First, decisions based on data, preferably quantitative, are superior to those made on opinions and shadow beliefs. And preferably, the decisions to be made should be small, reversible and collectively lead to an impactful outcome. Second, many decisions that historically had to be made based on opinions can now be made based on data, but organizations tend to get stuck in their old way of doing things. So, as a call to action: review the decision processes that you are involved in and evaluate where you could increase the use of data in each decision process.

On Big Data, Profiling, Fake News and Political Interference

This week I gave a keynote at the SANER 2018 conference in Campobasso, Italy. After the talk, some journalist from RAI asked me to do an interview with him about the topics above and I realized that although I have opinions about these topics, it's pretty hard to quickly pull all that together on the spot for a video clip of a few minutes. So, I guess this post is to atone for the interview where I managed to only get a small part of my views expressed.

First of all, what is important to realize is that there is nothing new under the sun. Profiling, fake news and political interference using large efforts to accumulate data have been part of human history. We don't have to go back further than the totalitarian regimes of the 20th century to see examples of every citizen being profiled, spied upon and tracked. Similarly, the news in those countries was not exactly objective, to say the least. And similarly, in the west the people selecting what to select for sharing in the news still use their biases, political leanings, government preferences, etc. And although fake news, just blatantly claiming untruths, is simply wrong, it sits at the end of a scale where also interpretation of events, creative or truthful, occurs. Determining "the truth" in areas outside of the hard sciences is not trivial when you think about it. Realizing this, I gave up watching the news more than a decade ago. I don't watch television. Period. I only occasionally read news papers, online or otherwise. Instead, I try to think for myself and to speak the things that I feel are true, rather than parrot what I've heard some pretty face say on a big screen. So far, that has worked pretty well for me.

Second, the indignance that people express about the use of data by companies like Google and Facebook continues to surprise me. It must be pretty obvious that if you're getting something for free, the company that gives this free thing to you must make money somewhere else. To paraphrase a popular quote: if you're not paying, you are the product. Now, understand me well, I do think that regulation is needed and I am glad that GDPR is being put in place. But for most people, it will mean that they accept the terms of the service that they have been using and move on with their lives. Personally, I active DNT (Do Not Track) where available, use ad blockers and other tools to minimize my digital trail, but I do accept that using the free online services and tools means that I am paying with my data.

Third, there are many, many uses of data that are to your advantage. Most companies collect data not to profile you, personally. In fact, they don't really care about you as an individual. Instead, they care about optimizing their product or service by experimenting with different ways of conducting certain workflows, user interfaces and other aspects of the product. Their use of the data that you generate when you use their product or service leads to a better product that improves your life. Similarly, together with that of millions of others, your medical data, among the most personal and private of information, can be used to identify diseases earlier, link lifestyle choices with medical risks more clearly, etc. It may save your life or that of your children. And with the increasing prevalence of AI, machine- and deep-learning, these benefits will only continue to compound.

Fourth, there is a huge advantage to personalization of a product or service. When you use Amazon, the suggestions and recommendations are concerned with your interests. Spotify suggests new music to you that you would have never found otherwise based on what it knows about your music tastes. Personalized medicine is a relatively novel area where treatments are selected and configured based on your DNA and other relevant factors. However, for a company to personalize its service to you, it needs to create a profile of you, a process known as profiling. Again, if done well, this creates real, tangible benefits to you that improve the quality of your life. Possibly a lot.

Fifth, the societal benefit that the internet, computers, big data and personalization have brought to the world should not be underestimated. Especially in the western world, we live in THE BEST TIME in the ENTIRE history of mankind. Life literally has never been better. It's easy to forget that, but on every metric concerning the human condition, progress has been phenomenal and it's only accelerating. All this progress has come from technology and, during the last decades, from information technology. Remember that we all walk around with a small supercomputer in our pocket that gives us access to a "tree of knowledge" that is unmatched. That we can use that same device to talk to anyone on the planet right here and now. That in a few years, self-driving cars will largely remove the terrible suffering because of traffic victims (around 40.000 a year just in the US). The list goes on and on. And many of these benefits come for free! Free!

Concluding, I do not mean to claim that we're living in a perfect world. There is a lot of unnecessary suffering in the world and for all our efforts, I do not see that disappear. However, much of the physical suffering now takes place much later in life and is caused by natural causes rather than by famine, war or oppression. Mental issues seems to be a larger source of suffering during modern times due, in part I think, to the fact that many have chosen a life of enlightened hedonism rather than put their lives in service of something that is meaningful to them.

Big data and profiling are tools without any inherent associated morality. Similarly to using a knife or a car, we can use the tool to provide societal and personal benefit or to harm others. The current public conversation seems to only focus on the negatives and this threatens to throw out the baby with the bathwater. Rather than instituting rules and blindly following these, I would like us to start thinking again. To go back to the first principles that we built western society upon and to ensure that all these fantastic tools that keep being added to our toolbox are used for beneficial purposes while minimizing the detrimental consequences.

Towards Dataism

Over the last months, one of the books that I've been reading is Homo Deus by Yuval Noah Harari. It's a really worthwhile read and I can warmly recommend it. One of the key ideas that he brings up in the last part of the book is the evolution of religions that mankind seems to go through. From the traditional religions where power is put in mystic outside sources, e.g. God, we currently largely follow humanism as a religion. Here the source of reference or the anchoring point is your deepest, truest feelings, resulting in a model where if something feels right for you and you're not hurting anyone else, you should be free to pursue whatever you want.

Humanism is the religion du jour in most of the western world, but it it based on a fundamental assumption: that there is such a thing as a self. We all feel as if we're individuals, but more recent psychological research currently seems to confirm what Buddhism has employed as a starting premise: there is no such thing as a self. The human psyche may best be described as a set of competing "subpersonalities" and our consciousness attaches itself to whatever subpersonality that happens to be on top right now.

As a response to the above, Harari claims that we're moving towards a new religion: dataism. Rather than making decisions based on what the gods tell us or based on what feels right, we take decisions based on what the data tells us. Although data seems objective and outside of the subjective human experience, make no mistake: this really is a religious viewpoint that we're taking. Anyone who has worked with data knows how easy it is to select and correlate data to one's biases and preferences and then to interpret it in a way that confirms whatever initial beliefs we started with.

Still, for all its challenges, dataism still is a much better starting point that relying on the contemporary interpretation of old books or on whatever feels right. What feels right is typically based on our experiences to date and, in a fast changing world, seldom is the best starting point for decision making. As long as we take a scientific approach and actively seek to limit biases and preconceived beliefs, data is superior to other foundations for decision making.

Now the important point here is that although the above mostly concerns individuals, companies operate in the same way: although we often pretend that there is a hierarchy and that the company speaks with a single voice, it is obvious, even to the casual observer, that there are multiple "subpersonalities" inside companies as well. Informal networks of people that believe the same about the best way forward for the company, how "we" work and should work, what is important and what can be safely ignored, etc.

Similarly, decision making based on humanistic principles, i.e. whatever feels right, is prevalent in companies as well and although many companies use data for decision making. Often the data is "massaged" to have it say what the company wants it to say or there is no data available

and companies take huge and costly decisions based on beliefs, rather than running small experiments to gather the data for further decision making.

Concluding, individuals and companies alike tend to make decisions mostly based on humanistic principles, i.e. what feels right. This tends to lead to low-quality decisions, especially in situations where past experiences and learnings are a detriment, rather than asset, because of the fast changes happening in ones industry. The superior alternative is to adopt dataism as one's starting point for life. However, this requires significant discipline to avoid falling into biases and preconceived beliefs. Finally, although data can be used to a much more significant extent than today, we should remember that data is not available for everything. As Einstein said, not everything that can be counted counts and not everything that counts can be counted.

Five Reasons Why You Are Not Data-Driven

This week I spent a lot of time with companies talking about data and using data for a variety of purposes, ranging from improved decision making to machine learning and deep learning systems. All companies I talk to have tons of data in their archives and often generate a lot of data in real-time or through batched uploads. However, although all companies claim to be data-driven in their decision processes and ways of working, practice often shows a different reality. When reflecting on my experiences with a variety of companies, I realized that there are five patterns or reasons why companies are not as data-driven as they think.

First, **wrong data**: Although enormous amounts of data are collected, in many cases the data that we would need to reliably answer specific questions is not available. Many embedded systems companies collect performance and quality data to track reliability, uptime and system crashes. Although this data offers solid insight into quality, it often fails to answer questions about the value products deliver for customers.

Second, **wrong interpretation**: Often data is interpreted using less than reliable analytics approaches. Ensuring that certain correlations in data exist requires a solid understanding of statistics and data analysis that is not always present. In many cases, statistical techniques are used for analysis without an understanding of the scope in which these techniques apply and this easily leads to violations of the underlying assumptions of these techniques and consequently an analysis that cannot be trusted.

Third, **find what you expect**: Many companies have strong biases about what they believe to be true about the market, their products and their customers. This means that if an analysis seems to confirm what we are expecting to find anyway, companies are much more likely to accept the analysis without due diligence of the mathematical accuracy of the conclusion.

Fourth, **creative reinterpretation**: In some cases that I have seen, data that is inconvenient for leaders is discredited and creatively interpreted. Often, in this context several hypotheses are formulated that provide alternative interpretations, but these are never followed up on. Unless the level of discipline is very high in the company, beliefs tend to trump data.

Fifth, **set up for failure**: Finally, when a company starts with a data-centric initiative, the initial focus is on a big, strategically important topic that has many contributing variables. The topic is selected in order to garner the necessary support. However, the first initiatives on using data to influence the strategic goal have too little power to move the needle on the measured output data. The effect of the input variables is too little push the output variables outside the noise range. Therefore, the initiative is easily categorized as a failure as the effects were too small to measure on the selected output variable. In effect, the initiative was set up for failure from the beginning.

Concluding, becoming data-driven requires discipline to follow data even (or especially) when it flies in the face of the beliefs in the company, technical understanding of the underlying mathematics and prioritization of initiatives that have the opportunity to make a difference. After all, in my experience, all successful companies follow the motto formulated by Edwards Deming: In God we trust; all others bring data.

What Are You Optimizing For?

Over the last weeks, I have been in several group conversations where we had to agree on the relative priority of multiple factors. For instance, are we optimizing for the total number of users or are we optimizing for maximizing revenue per user? Should we prioritize fuel efficiency or is minimizing exhaust waste more important? Should we seek to get as much value from the initial sale of the system or should we aim to optimize the service and maintenance fees during the lifetime of the system? In all these situations, it rapidly became obvious that there were very different opinions in the room and that even individual participants were far from clear themselves.

Not being entirely clear on the relative priority of different relevant factors used to be fine as R&D was mostly driven by requirements provided by product management. And the product managers would have, sometimes extensive, discussions on the relative importance of different requirements and prioritize the backlog accordingly.

More recently, however, we're increasingly moving towards outcome-driven and AI-driven development (as discussed in a blog post from earlier this year). Outcome-driven development uses techniques such as A/B/n testing to determine which alternative for achieving a certain outcome performs better. However, in that case you need to be crystal clear to define in quantitative terms what the desired outcome is. Similarly, AI-driven development typically uses large (often labeled) data-sets to train a machine- or deep-learning model to classify, predict, etc. new data based on a training data-set. However, again it is critically important to precisely define what constitutes success.

The point I am trying to make is that most companies that are new to data-driven R&D practices fail to conduct the exercise to clearly express quantitatively what they are optimizing for. There are several reasons why this is the case. It requires a significant amount of confrontation and transparency to call out team members who seem to be optimizing for other factors than what we agreed upon. Deciding on the relative priority of factors requires the organization to make hard choices. In my experience, when decision makers are asked to choose between alternative A and B, the typical response is "I want both".

Failing to get clear on relative priorities and making real choices, however, results in several dysfunctions and inefficiencies in organizations. For example, in one company, one team worked on improving the security of the product and another team was looking to improve performance. The actions of both teams cancelled out each others efforts and the result was a lot of activity but no progress whatsoever.

Concluding, as we're increasingly adopt data-driven practices such as outcome- and AI-driven development, we can no longer afford to be vague and imprecise on what we're optimizing for. Instead, we need to engage in the hard discussions in the company to define clear relative

priorities for the various factors that matter for the company and then engage in experiments to accomplish quantitatively validated improvements. In short, know what you're optimizing for or pay the price!

Become Data-Driven In Five Steps

Although most of the online (SaaS) companies that I work with are heavily data-driven, the embedded systems companies have had more difficulty to achieve that. With connectivity becoming more and more the norm for embedded systems, though, these companies are now also starting to become data driven. This is great as replacing opinion-driven decision making with data-driven practices allows for an enormous improvement in the effectiveness of R&D.

After going through the first steps of adopting data-driven ways of working with several companies, I think we can see that the transformation proceeds through (at least) five steps. Below I discuss these steps in more detail.

The first step is that a team in the company starts with **modeling the expected value** of a new or already built feature in terms of measurable factors or metrics. At this point, the company often has a release process that is shorter than, say, 6 months for at least some group of users so that the team has the possibility to measure the effect of the feature on the system and user behaviour.

The second step is to **start building the data collection and analysis infrastructure** as well as convincing relevant stakeholders in the organization that the collection of data from the field is an important activity that justifies the internal effort, discussions with the customer and system connectivity cost.

The third step is to adopt an **iterative feature development approach** (like our HYPEX model) where rather than building new features completely in one step, features are developed a slice at a time. Each slice is deployed in the field and the quantitative effects of each slice can be measured. This allows the team to adjust their development in response to the effect of their iterations.

The fourth step is to **accelerate the feedback loop** as shorter feedback loops allow the team to significantly increase the effectiveness of their development. If a team has to wait several months until data from the field becomes available after they finish development, they can progress much slower than when the data from the field becomes available right after the end of the sprint.

Finally, the fifth step is to **build a hierarchical value model** that links business level metrics (that often are lagging indicators that change slowly) to lower level system and feature level metrics (that often are fast-changing leading indicators). By connecting team-level feature metrics to the top level business KPIs, the entire organization can be aligned along the same quantitative value model.

Concluding, adopting data-driven practices allows companies to improve the effectiveness of their R&D (the amount of business value created by each unit of R&D resources) enormously. Although this is contemporary practice in most B2C SaaS companies, we now see that embedded systems companies as well as B2B companies in general are adopting these practices as well. The adoption tends to follow the five steps that I outline here. Of course, your organization may benefit from an alternative path, but it's important to realize that staying with the traditional, opinion-based approach to allocating R&D resources is no longer an option. Become data-driven or go the way of the dinosaurs ...

Will Computers Program Themselves?

With artificial intelligence, machine learning and deep learning on the top of Gartner's hype cycle, it is easy to assume that computers will program themselves going forward and that we've reached the end of software engineering as a discipline. Although one might naively think that we can just feed a lot of data into a neural network and that magic will happen by itself, in reality industry-strength machine learning and deep learning solutions experience a similar set of challenges as traditional software. Ranging from versioning and traceability to organizational and process issues, the engineering challenges associated with large software systems remain present and need to be addressed.

In our work with the companies in Software Center, our research shows that software engineering in the future will be concerned with three development approaches:

- **Requirement driven development** where software is built to specification in the way that software has been constructed for the last decades. Especially in cases where the system needs to meet regulatory demands or achieve parity with systems built by competitors, this type of development will be the most effective.
- **Outcome driven development** where development teams receive a quantitative target to realize, such as conversion rate for part of an e-commerce website or throughput for a telecom system, and are asked to experiment with different solutions to improve the metric.
- **Data driven development** where the company has a large data set available and use artificial intelligence techniques such as machine learning and deep learning to create components that use input data to act based on earlier learnings.

The challenge is that rather than treating these three approaches to development as separate, it is important to realize that the results from development need to be integrated into the same system. The result will be similar to the structure outlined in the figure below.

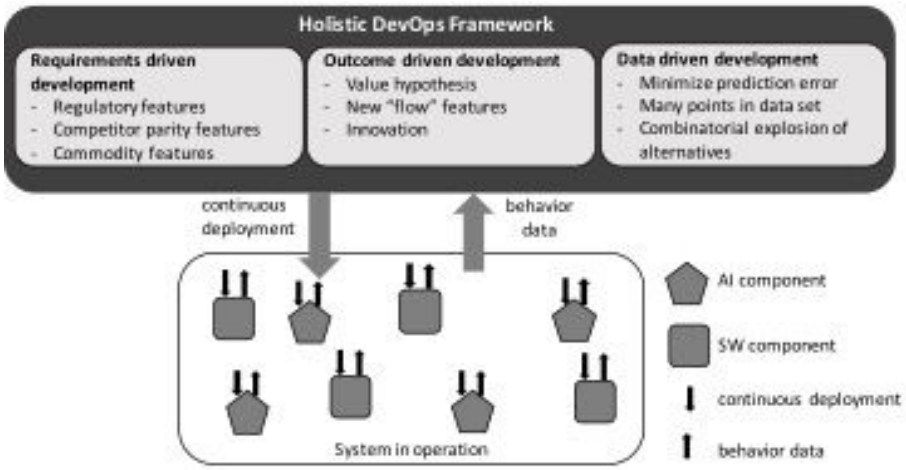


Figure: Holistic DevOps Framework

The important takeaway here is that we build systems where the various components are created by different development approaches but still are subject to continuous deployment and the continuous collection of behavioral data about the system, its users and its context. That feedback cycle can then be used to continuously improve the performance of the system and adjust it to changing circumstances.

Concluding, although one might naively expect software engineering to be reaching end of life, because of the emergence of AI/ML/DL, in practice the field will only become more important as the engineering challenges traditionally associated with software engineering only become more pressing when including additional development approaches. To me, this is good news because after more than 20 years as a software engineering professor, I am hoping for a few more decades contributing to the field!

Engineering Deep Learning Systems is Hard!

This week I spent in the lovely city of Prague, attending the SEAA 2018 conference. The main reason for attending was that I had the opportunity to present a paper that I co-authored with colleagues from Peltarion about the software engineering challenges of deep learning. Peltarion offers an amazing platform for building deep learning systems and if you're interested in AI/ML/DL, you should really check out their platform.

The team from Peltarion that I wrote the paper with has several decades of experience in building deep learning systems and come from successful (and now large) startups in Sweden and have worked for a variety of high-profile customers from around the world. Their experiences and insights provided an excellent basis for a paper that identifies the key software engineering challenges associated with building mission-critical, production-quality deep learning systems.

Surprisingly, although many companies are experimenting with deep learning and build prototypes using tools like TensorFlow and PyTorch, few realize the challenges associated with taking such a prototype and productifying it are many and are likely to cause the whole initiative to fail. This is not an empty threat as we have several examples of companies trying to go it alone without a proper platform. The software engineering challenges of deep learning are real and hard.

Although you can download the slides here and the paper here, I thought I'd provide a brief overview of the topics that I presented. We identify three categories of challenges, associated with development, production and organization.

We discuss five development challenges:

- **Experiment Management**: how to keep track of all the contextual factors (e.g. HW, platform, etc.) when running many experiments.
- **Limited Transparency**: neural networks give little insight how why and how they work.
- **Troubleshooting**: large libraries, lazy execution and lack of tooling complicate solving defects enormously.
- **Resource Limitations**: large data sets and complex models cause the use of distributed architectures that further complicate experiment management and troubleshooting and increase cost, required knowledge and time.
- **Testing**: testing of data is very complicated and there is little tooling. Also testing models and the underlying infrastructure is very complex.

The production challenges we identified include:

- **Dependency Management**: GPU hardware and SW platforms are improving and evolving very rapidly and software often encodes contextual knowledge for performance reasons.

- **Monitoring and Logging**: models are often retrained frequently and behaviour may change. Distinguishing between bugs and improvements is difficult.
- **Unintended Feedback Loops**: real world may adopt to the model, rather than the other way around.

And finally, the organizational challenges we discuss are the following:

- **Effort Estimation**: we fundamentally don't know how long it will take to develop an acceptable model (and if it will ever happen).
- **Privacy and Data Safety**: as we often don't know how neural networks work, it is difficult to guarantee that no personal or private data is stored in the network.
- **Cultural Differences**: data scientists and software engineers often have very different drivers and ways of working.

Concluding, current tools, such as TensorFlow and PyTorch, make it very easy to build deep learning prototypes. Building mission-critical, production quality systems out of these prototypes proves to be a significant software engineering challenge and in the paper and presentation my colleagues at Peltarion and I present the key development, production and organization challenges. Of course, it goes without saying that Peltarion offers a platform that solves many, if not all of these challenges, and offers - what we call - Operational AI. However, the purpose of this article is not to be a sales pitch, to raise awareness. Engineering deep learning systems is hard!

Ecosystems

Most literature in professional and academic contexts takes the perspective of the individual company and focuses inwards from the boundaries of the company. The literature focuses on business models, system architectures, ways of working, organizational structures, company culture, etc. However, the common denominator is that company is the unit of study.

The reality is that, of course, no company is an island. Instead, each company operates in an ecosystem, or rather multiple ecosystems, and managing the relationship to these ecosystems is of critical importance for the competitiveness of the organization.

One of the challenges that many companies experience is that conservative, but highly profitable, customers are unwilling to engage with new innovations developed by the company. This may cause the company experiencing a higher risk of disruption as new entrants may more easily engage with progressive customers, build market share and become the new market leaders. Of course, Clayton Christensen captured this model in his innovator's dilemma, but the fact remains that this challenge concerns the interface of the company to its business ecosystem, rather than the internals of the company.

A second challenge that many companies experience is that they are unclear about the role they look to take in the ecosystem. Fundamentally, a software-intensive systems company can take a role as a product, a platform or an ecosystem company. Each of these strategies requires a different behavior from the company, but many a company has a lack of internal clarity on its desired position, resulting in conflicting and confusing messages to the market and its partners in its ecosystem.

Third, once a company decides on a strategy to follow and becomes conscious of its business ecosystem, there is a desire to reposition itself to a role and location in the ecosystem that is more desirable compared to its current position. As we'll discuss in the next sections, companies fall into several possible traps when repositioning and being aware of these traps can help circumvent them.

Concluding, every company operates in a business ecosystem, or rather multiple business ecosystems, and needs to be strategic and intentional about choosing its position, transitioning towards that position and then defending that position against others. In the sections below, we describe some of the challenges that companies that we work with have experienced as well as strategies to overcome these challenges and thrive in your ecosystem.

Orchestrate Your Ecosystem (Or Be Ruled By It)

Every week or so, I end up in a discussion about ecosystems, be it business ecosystems or software ecosystems. Interestingly, almost everyone I talk to takes a “descriptive” approach to their ecosystem of choice. They describe the ecosystem as a fact of life or a force of nature, like gravity, and often I get explained to me why the ecosystem had to turn out as it did.

The good news is that ecosystems are increasingly getting attention. Companies and the people working for these companies more and more recognize that they don’t live in a simple value chain, buying from suppliers and selling to customers. Instead, companies increasingly shift attention to the fact that they operate in multi-sided markets where influencers, saboteurs, complementors and other roles influence how the company operates.

The bad news is that many seem to assume that ecosystems are either immutable or only change by random acts of nature. My main message is: ecosystems are and can be orchestrated by putting the right incentives in place for all roles in the ecosystem. Similarly, ecosystems can be disrupted by understanding unmet needs of roles in the ecosystem and finding more effective ways to meet those needs.

So, how does one orchestrate or disrupt a business or software ecosystem? This is one of those questions that is conceptually simple and incredibly hard from an operational and execution perspective. However, it consists of the following steps:

- Model and understand the current ecosystem, the players and the drivers, both positive and negative, of the players. The focus should be on the value exchange between the ecosystem partners.
- Develop a tangible and clear view on and model of the desired ecosystem, in terms of participants, their specific role and the value exchange between them.
- Identify which roles need to materially shift between the current state and desired state.
- Select the role that is most likely to shift if offered a better position in the desired ecosystem. Even though the desired ecosystem will likely require shifts in several many ecosystem partners, we are unable to move everyone at the same time and consequently need to take one (or a few) roles over to the desired state in every state. In most ecosystems, the effects are cumulative, the benefits for the first ecosystem partners need to be sufficiently attractive without the cumulative effects.
- Experiment with each role to ensure that what you perceive as a sufficient benefit for the ecosystem partner to adopt the new role. Those that I meet that actually believe that they can create a new ecosystem or make material changes to an existing one live in some kind of la-la land. They tend to believe that people will act as they want just because they want it to be true. Although I am a big fan of reality distortion fields a la Steve Jobs, I just don’t see these fields be successful happen that often.
- Iterate until the current ecosystem role has a sufficiently appealing value proposition and then move on to the next ecosystem role and set of partners.

All this sounds like a simple recipe from a cookbook, but in practice this requires fantastic amounts of energy, time and resources as well as people that are really skilled at convincing others to at least give it a shot. In other words, it's not easy and likely to fail, but it can be done.

The reason for discussing this topic now is that digitalisation is causing major ecosystem disruptions in many of the software-intensive systems industries. Many that I discuss this with view the shifts in the ecosystem as something unavoidable, a force majeure that just has to be weathered with the hope of survival. These people tend to take a tactical approach of small steps in response to what happens around them.

Too few view this as an opportunity to upset the status quo, design the ecosystem best for them in a strategic fashion. But the fact is that a major upheaval in the ecosystem is of course the only real opportunity that one has for a material shift in position, size and profitability. In stable markets, growth tends to match the growth of GDP.

How to go about this is a topic too long for a blog post, but there are three thoughts that I'd like to leave you with:

- Find areas where a small set of ecosystem partners has significant control. This tends to translate into significant pricing power, architectural control, etc. In this area, seek ways to adopt new responsibilities that would shift this power to you and allows for many more partners to compete. That allows you to increase your control and value capture in the ecosystem and gives you more options.
- Look for areas where customers are underserved with standard solutions and identify ways to open up for 3rd party developers to offer more variety. When opening up, however, ensure a control point and a mechanism for revenue share.
- Whenever attempting to shift around in the ecosystem, make sure to experiment in non-material markets first as most experiments may have irreversible effects in the ecosystem.

Finally, in my Speed, Data and Ecosystems book [ref] I discuss these topics in more detail and show how to move from being an internally focused organization to a company that strategically engages with its ecosystems (as shown in the picture below).

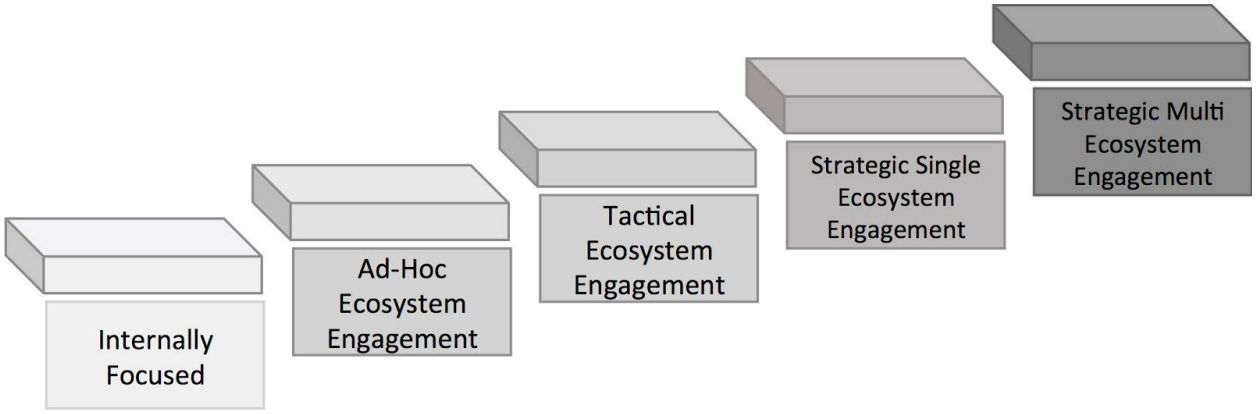


Figure: From internal focus to strategic ecosystem engagement

The Five Traps of Software Ecosystems

Software ecosystems and platform businesses are the new black and several of the companies that I work with are, in some form or another, working on more strategic engagements with their ecosystems. Although the ambition to position oneself as a keystone partner in a software ecosystems and to provide a platform for others in the ecosystem is a laudable one, I see companies fall into a number of traps. Five of these traps are the most common and below I discuss these in more detail.

Descriptive versus prescriptive trap: The first trap that several companies fall into is assuming that the ecosystem as it exists today is “cast in stone”. The assumption is that the current boundaries between different ecosystem partners and partner types are a given and immutable.

Of course, static ecosystems are a misconception as everything changes all the time. One has to understand that an ecosystem is balanced by forces. These forces cause boundaries and power and control positions between different ecosystem partners to balance out to a certain equilibrium. When the forces in the ecosystem change, the boundaries and power positions of ecosystem partners change as well. Consequently, companies should experiment with changing the forces in their ecosystems in order to proactively bring shifts in the ecosystem

For example, imagine a company in a regulated industry that relies on suppliers that provide pre-certified subsystems. This tends to cause for more power to reside with the suppliers because the amount of knowledge, expertise and resources required is quite significant. This results in a small selection of suppliers that have significant pricing power. If the company would incorporate the certification of subsystems as part of its system certification process, the requirements on suppliers would be significantly reduced, allowing for more suppliers being able to offer solutions, resulting in less pricing power for suppliers and a lower cost for the company.

Assumptions trap: Especially keystone partner companies often assume they understand the needs and wants of their ecosystem partners and all ecosystem partner types. Based on this incorrect understanding, companies take decisions concerning the business strategy, architecture, ways of working, etc. that are fundamentally at odds with the real interests of the majority of their partners.

To address this issue, it often helps to explicitly model your ecosystem in terms of the relationships between ecosystem partner and partner types (and not just to your company). As part of the modeling, the value exchange, both pecuniary and otherwise, should be expressed explicitly as well. This will surface the assumptions about partners that your company holds. Once these assumptions have been made explicit, it often is quite feasible to find mechanisms to test and validate these assumptions.

For example, many ecosystem evangelists stress the ease of deploying apps or extensions to their platform to third party developers and believe that this is a key differentiator. Apple has clearly shown that this is incorrect. Their submission process is complex and a pain for developers, but developers still prioritize iOS over other mobile ecosystems. The reason for this is simple: iPhone users spend much more on apps than Android users. Developers accept the complicated app submission process as they know that the likelihood of making money is much higher.

Keeping it too simple trap: Several companies that I work with treat an ecosystem partner, such as another company, as a single entity. Consequently, they miss that even within a single company, there are multiple roles inside the organization that affect decision making and behaviour within the ecosystem.

The solution is to identify and model the various “interfaces” the ecosystem partner has to the ecosystem. For instance, the procurement function, the security officer and the R&D organization will have fundamentally different drivers to support or refuse to participate in an ecosystem. Especially keystone partners need to understand the different roles and functions within the partners they look to engage. Based on this, one can develop hypotheses of the wants and needs of each “interface” and subsequently test and validate these hypotheses.

For instance, many banks are engaging with FinTech startups to bring new, exciting functionality to their customers. These FinTech startups demand access to the financial information of a bank’s customers. The information security and compliance functions of the bank as well as several standards in the banking industry significantly complicate this ambition. Failing to satisfy the needs of these functions will break the ecosystem before it even started.

Doing it all at once trap: When seeking to shift or disrupt an ecosystem, for some reason most companies assume that all partners can and will move at the same time just because the company asks them to. It’s almost as if the traditional reorganization model applied within the company is also applied to the ecosystem. Obviously, without the hierarchical control enjoyed within organizations, partners will only change and shift their position in the ecosystem if there is a significant benefit that overcomes the always present inertia.

Although one does need an understanding of the desired state, the transition process itself should take small steps where, first, the first partner type(s) to move are identified. Second, the company needs to ensure that there is a sufficiently powerful value proposition for these partners. Third, one needs to validate that the partners actually do shift when presented with the opportunity. Once this is concluded for the first partners or partner types, this process needs to be repeated until entire ecosystem has shifted.

For example, Amazon broadened its offering and the set of partners that it involved in a gradual fashion. It didn’t start its current ecosystem as it looks today fully designed and modeled and just pushed the button. This required continuous experimentation and iteration.

Planning trap: Many companies are hesitant to engage with their ecosystems. Their assumption seems to be that it is necessary to fully model and design the desired ecosystem before starting to engage partners. Of course, full modeling and design of what you may want to see is just wishful thinking and things will never materialize in this way.

The best thing to do is to just get started. First, set a high-level direction of what you're looking to accomplish. Then, plan the first step in the general direction and experiment your way towards the first milestone. And while you're iterating and experimenting, constantly adjust according to the emerging feedback.

As an example, especially startups seeking to build a two-sided (or multi-sided) marketplace need to build sufficient participation from both sides before their ecosystem reaches "ignition". It is often much easier to first build up a user base for one side of the market by offering something of value in and of itself, even without the other side of the market being present. Once this is accomplished, one can expand to include third parties. For instance, Amazon, Facebook and similar companies first built a user base by offering a valuable service. Only then did they open for third party developers.

Concluding, positioning your organization as the platform provided and keystone partner in an ecosystem is a highly desirable position that many companies aspire to. In my work with a variety of companies, however, I have noticed that companies tend to fall into a number of traps, some of which I outlined in this post. Please share your comments and feedback and let me know if you have seen other traps that companies experience!

Why Your Customers Are Slowing You Down

Every company is part of an business ecosystem. In the ecosystem, we find the partners, the suppliers, the customers, competitors, complementors as well as other stakeholders. A business ecosystem was more formally defined in the early 1990s by Moore as an economic community with three key characteristics. First, there is a symbiotic relationship between the members of the community and the survival of these members implies survival of ecosystem.
Second, there is coevolution meaning that partners co-evolve capabilities around new innovations. Third, the ecosystem is organized around a platform that is defined as tools, services, and technologies that members of the ecosystem can use to enhance their own performance. The basic notion of a business ecosystem (and a software ecosystem is a special type of business ecosystem) is that everyone is better off within the ecosystem than outside of it.

As is clear from the description of a business ecosystem, there are numerous virtuous, as well as competitive, dependencies between the ecosystem partners. It is these dependencies that allows the member of the ecosystem to be better off by being inside it. The backside of this, however, is that any changes need to be negotiated and agreed between ecosystem partners. And this easily slows down the evolution of the ecosystem and, if not managed properly, it can destroy the competitiveness of the entire ecosystem and risk its disruption by other companies and ecosystems.

Within the ecosystem of a company there are customers - the people that provide you with the revenue to stay, and preferably thrive, in business. The challenge is that in several of the cases where I have been involved, the challenge is that the most important customers are the ones least interested in changing and evolving the engagement model. Why this happens can easily be understood using the Three Layer Product Model that I shared in an earlier blog post (see picture below). Even though the functionality and value that you provide for your customers is differentiating for you, it typically is considered commodity by your customers. If your customers viewed it as differentiating, often they would be more interested in having more control of the functionality; for instance, by managing it inhouse. As it is commodity for them, there is little incentive to evolve as the it will bring with it cost but very little value. Hence, the tendency is to delay making changes.

The risk is of course that this presents a possible source of disruption for your company: as your customers are reluctant to change, you don’t change and as a consequence you may get left behind and disrupted by the competition. This challenge is often exacerbated by product managers and others who wait for customers to ask for changes before prioritizing the work items needed to evolve and innovate. The key challenge, however, is that your customers can and often will slow you down and hurt your competitiveness.

Addressing this challenge is difficult and requires a culture in the company that identifies the importance of being ahead of the customer. There are three mechanisms that I have found helpful in avoiding this situation, i.e. analysing customer requests, horizon-based resource allocation and working with fringe customers. I'll describe each of these mechanisms in a bit more detail below.

First, one of the rules that I often use with the companies that I work with is: if your customer has to ask you for some functionality, you have already failed. Your inability to predict customer needs before the customer is able to verbalize the need and is ready to ask for it is a critical failure that needs to be carefully analysed. This is not to say that every customer request should be honored! Rather that each request should be analysed from the preemptive perspective of predicting customer needs before customers ask for it. This analysis can then be used to analyse whether there are any current topics that are not being prioritized but are likely to become future customer requests. This can then be used to preempt and prioritize work.

Second, in another blog post I introduced the three horizon model where resource allocation is divided over three horizons of work (see here). Ensuring that 10% of resources are allocated for building new differentiation and preferably with other customers or engagement models than the ones slowing you down provides at least some hedge against key customers that are slowing you down.

Third, as Clayton Christensen analysed to nicely in his book on the innovator's dilemma, most companies have a tendency to focus on the most profitable and typically most demanding customers. From a short term revenue generation perspective, this is entirely the correct strategy, but from a long-term perspective this tends to cause you to sacrifice more and more of the rest of the market to competitors. To avoid this, it is important to work with fringe customers using the new, innovative engagement models. This allows the company to innovate with less critical customers and it builds confidence within the company as well as with its key customers that the innovations are actually providing the expected business value.

Concluding, if not managed properly your most important customers are likely to be the ones that slow your rate of innovation down to the point that your innovativeness is negatively affected and you may risk disruption. Business ecosystems are wonderful when they work, but these can also be anchors that slow you down and increase the risk of disruption. To avoid this, I have worked with three mechanisms to avoid this, i.e. analysing customer requests, horizon-based resource allocation and working with fringe customers. In short, enjoy your business ecosystem, but be vigilant against the risks!

Why One Customer Is No Longer Enough

This week, I came across a concept from Karl Popper, the great philosopher of science, where he distinguishes human beings from other creatures in that we can create an idea in the theatre of our imagination. We test the idea against other ideas that we have or others have or even against the world. And if it doesn't hold, we can let it go. We let the idea idea die in our stead. That allows us to continue, a little wiser maybe, instead of dying with the idea.

I realized that this concept is true for companies too. Companies are built around an idea of how the world is put together and how the company can find a value adding role in that world. Good and novel ideas result in extremely valuable, large and successful companies. And make no mistake: Management of such companies is needed, but a company with great management but a poor idea at its core will fail whereas the opposite is not necessarily true.

In a world where the tenure of companies on the Fortune 500 is ever shortening, it is clear that companies get disrupted because they cling to an old idea without letting go and moving on to a new idea. Companies have a tendency to die with their core idea, rather than managing to let the idea die in their stead. One of the reasons is that it is incredibly hard to align hundreds or thousands of people around a core idea to begin with and exploring a set of new ideas, selecting the most promising one and realigning everyone around this new idea is even harder. But if, as a company, you fail at it, you die.

This week we organized the second Digitalization workshop with senior leaders in the Software Center companies and I realized that most European companies at heart still are living an idea that may be reaching the point where it needs to be replaced. This idea is that, as a company, we sell to a customer and that by providing a good product at an attractive price is sufficient for us to be successful going forward.

If we look at the most successful technology companies today, they all have a common element at heart: they serve a multi-sided market. There is not one type of customer, but multiple types of customers and by balancing the needs and wants of these customer categories and providing a platform with low transaction cost, companies are able to build incredibly valuable businesses.

The power of a serving a multi-sided market is that it allows you to subsidize one side of the market and provide a product or service at cost of even for free for one category of customer while monetizing from another customer category. The most obvious example is of course advertising on sites like Google and Facebook but also on millions upon millions of niche websites. The reason that this is happens is of course that advertising is one of the most inefficient industries that exists. Bombarding thousands or millions of people, trying to reach the few that actually have an interest in buying what you have to offer is dreadfully inefficient. Any data that gives me a way to target my marketing spend on a group of potential customers that is more likely to be interested is worth a lot.

The challenge for a company in a traditional single-sided market (where you simply sell to a customer) is that you need to earn a profit from that customer base. When a competitor with a multi-sided business model enters your market and is able to provide the same solution to your customers at cost or even for free, your ability to stay in business is severely hampered. This is exactly what happened to online maps in the 1990s and early 2000s where companies like MapQuest were looking to sell their map data directly to customers. With the introduction of Google Maps and later Apple Maps, for the majority of users this service was free and reduced the business opportunity for the incumbents to niche markets with specific needs not met by the big players.

In my experience, most European companies are still operating in a business model that assumes a single-sided market. Based on the discussion so far, it must be obvious that it's time to start developing multi-sided business models that allow us to monetize our existing customer base in additional ways and to add new customer categories to our business models.

Most of the new multi-sided market opportunities are driven by data. By collecting data from our existing customer base and providing that data to other interested parties, new revenue streams can be developed and the creation of data-driven and digital services is of course at the heart of digitalization. Most of these services are not intended for our existing customers, but rather for other categories of customers.

As a side note: With the introduction of GDPR in May, I understand that I am going against the stream here. But for all the good intentions underlying the regulation, I can't help wonder if it doesn't hurt the already fragile competitive position of the European industry more than it restricts some of the non-European companies affected by it. In my experience, most European companies stay far from the boundaries of the regulation and often opt to not collect any data at all whereas many foreign companies push the boundaries as hard as they can and opt to pay fines rather than to play it safe.

Concluding, the idea that it is enough to sell to a single type of customer has outlived it usefulness and it will die in the coming years and decades. Instead, successful companies will serve multi-sided markets and provide platforms where multiple types of customers can transact. This is not a new idea per se, but most European companies are still under the impression that they can survive in a single-sided market. That is no longer true and it sets you up for disruption. It's time to start developing those other market opportunities before a new entrant in your market makes your customers an offer they can't refuse. Don't let your company die with the idea of a single-sided market.

Are You a Product, Platform or Ecosystem?

Another week; another ridiculous amount of hours on an airplane, so this post is coming from Bangalore. During last week, I noticed an interesting pattern that I had seen several times over the last year: many companies and their leaders are not clear on what business they're actually in, especially as it comes to being a product, platform or ecosystem-centric company.

The distinction is important as each type of company comes with its own set of characteristics and criteria. A product company is concerned with selling a, typically standalone, product. The goal of the company is to sell the product to as many customers as possible against the best price possible in order to maximize margins and revenue. Although the product may have to be integrated with other systems at the customer, the company only needs to focus on a single stakeholder.

A platform company, on the other hand, seeks to commercialize a multi-sided market where, in the simplest case, suppliers and consumers are meeting to exchange value on top of the platform. The platform may require installation on end-user equipment, such as your computer's ecosystem, or it may only provide the marketplace, such as an online hotel booking site. The key characteristic is that the platform is concerned with offering the best possible solution for different stakeholders exchanging value. This means, it needs to balance the interests of different stakeholders as well as its own interests in a way that leads to the best global optimum.

Finally, a company organized around an ecosystem has many similarities with a platform company, especially if it is the keystone player in the ecosystem, but there are some differences in that functionality provided by the platform, the functionality provided by complementors and others as well as customer-specific functionality built by customers are all in the same domain and the dynamics around the evolution of functionality. For instance, the ecosystem platform needs to continuously incorporate new functionality in order to stay relevant and avoid commoditization. In this context, the keystone partner needs to be very careful to ensure a fair context for all participants in the ecosystem.

The challenge when not being clear on what kind of company you are is that the characteristics and success criteria are different for the different types of companies. And trying to execute on multiple strategies easily leads to conflicts, both internally and towards the market and rest of the business ecosystem surrounding the company. For instance, a product company seeks to maximize value capture by avoiding others interfering in the relationship. An ecosystem company needs to ensure that others can generate revenue by delivering value to others in the ecosystem.

In my experience, in many companies there is one espoused strategy and one strategy in use. In other words, companies say one thing and behave in a different way. In a world that is increasingly driven by meaning and purpose and continuous relationships rather than

transactional customer engagements, a discrepancy between words and actions is easily interpreted as hypocritical, resulting in reduced trust and, in the long run, customers moving on to others.

Concluding, be crystal clear on the position you are taking, be a product, a platform or an ecosystem company and then act in accordance with your selected strategy. Failure to do so results in internal confusion and contradictory behaviors as well as reduced trust and engagement in the market.

Don’t Be Left Behind By Your Business Ecosystem

This week, the Software Center (SC) organized the Sprint 14 reporting workshop for the partner companies and universities. A great event with over 100 people participating and learning about the progress that we’ve made during the last 6 month sprint. As we have expanded the mission of SC to *accelerate the digitalization* of the software intensive industry, we are now kicking off new communities, including the product management community. One of the key concerns raised by the product managers in the kick-off meeting is repositioning your company in your ecosystem without alienating the existing partners.

No company is an island. We all operate in a business ecosystem where other stakeholders in our ecosystem have certain expectations on our behavior, power relationships between the stakeholders exist and are used, business models have evolved over the decades and the roles are clear and defined. With the transition towards a digitalized society and industry, however, there is a significant need to change your company and your engagement with and position in the ecosystem.

In many of the companies that I work with, their business approach is basically dictated by the ecosystem and this ecosystem is viewed as static and immutable. Although I am the first to admit that it is far from trivial to change an ecosystem, I am at the same time not defeatist about actually accomplishing a material shift. It’s mostly that it takes time, risk and experimentation. And rather than the ad-hoc, by the seat of my pants approach that I see many companies take, it requires a systematic and iterative approach to realizing this.

In an earlier post, I outlined the traps that companies looking to shift their ecosystem or reposition themselves tend to fall into. These traps include the assumptions trap, the keeping-it-too-simple trap, the doing-it-all-at-once trap as well as a number of others. The underlying common denominator, however, is the lack of empathy with the stakeholders in the ecosystem. There often is an amazing lack of understanding of the drivers of others in the ecosystem.

In recent research, we have started to outline an approach to systematically evolving the ecosystem or repositioning oneself. This approach is a bit too elaborate to explain in a short blog post, but it consists four main activities, i.e. identify the key stakeholders that need to be changed or repositioned (both inside and outside the company), define the current and desired state for each stakeholder and, third, define the transition points for each stakeholders and finally define the experiments, initially for the highest priority stakeholders and first transition points, that may move the stakeholder one step from current towards desired state. Let’s look a bit more deeply into each of the activities in more detail.

The stakeholder identification, as the name implies, is concerned with identifying those individuals and groups that need to be transitioned from today’s state to a new state. We

recognize stakeholders inside and outside the company. Inside the company, stakeholders can be individuals, such as key decision makers, or functions, such as sales. Outside the company, stakeholders mostly are different groups, such as customers, partners and complementors. It is important to limit the set of stakeholders to only those that have a real, material impact or role in the transformation.

The next step is to describe the desired state for each identified stakeholder, i.e. in an ideal world, what would the behavior and role of this stakeholder be. This is followed by an honest assessment of the current behavior and role of the stakeholder. Although teams often have a tendency to sugarcoat especially internal stakeholders and customers, being explicit and frank about the current situation is critical for the success.

The third step identifies the key transition points that each stakeholder needs to go through to transition from current to desired state. Many organizations and change teams assume that a stakeholder has to transition in one, huge leap, but this is unrealistic. Stakeholders need to be taken through several steps and identifying the key transition points is important.

Finally, starting from the first transition point for the most important stakeholder, the team develops hypotheses on what might result in a stakeholder going through a change and reaching the next transition point. Based on these hypotheses, the team can run experiments to test the hypotheses in order to determine if their beliefs are correct. Based on that, the team can work through the entire change journey using a structured, systematic approach.

Obviously, there are many risks, obstacles and setbacks, but a process like I describe here provides a much more systematic approach than the ad-hoc approaches that tend to fall into the aforementioned traps. In the figure below, an anonymized intermediate result illustrates what this could look like.

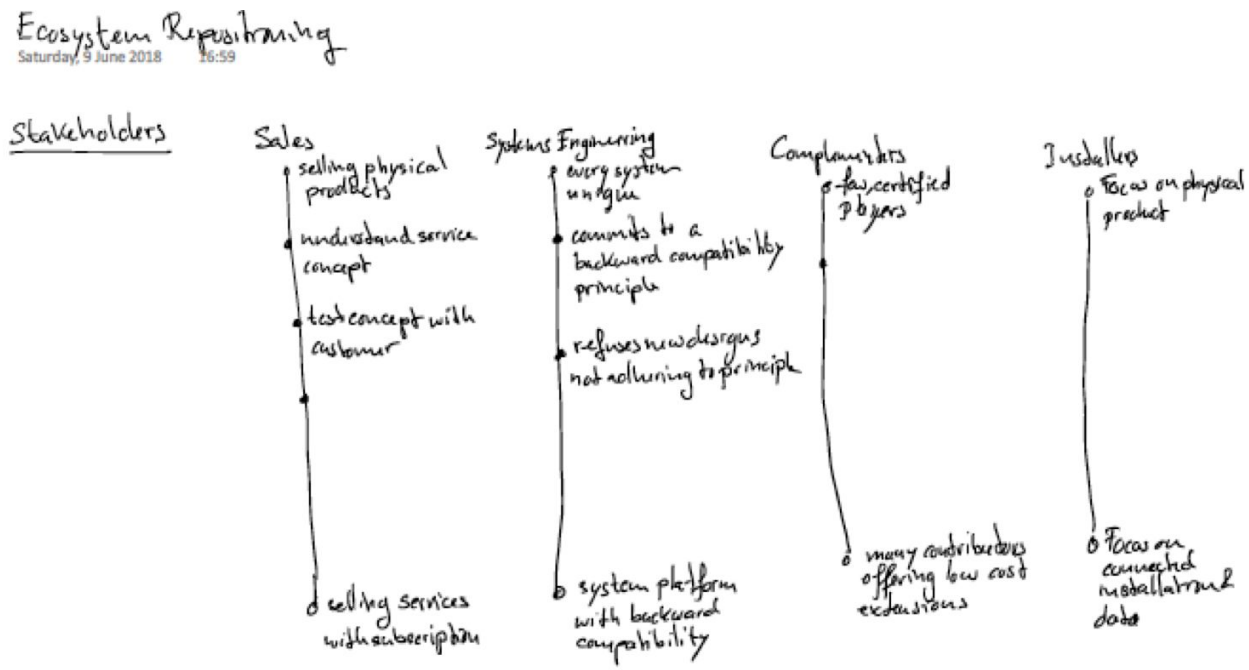


Figure: intermediate example of an ecosystem repositioning planning session

Concluding, the business ecosystem which you operate is the result of a set of balancing forces and digitalization, increasing speed, changing business models and changing customer preferences cause those forces to shift, resulting in evolution or revolution in your ecosystem. Rather than being a victim of circumstances, you need to take ownership of the position and role of your company in the ecosystem. Doing so, however, requires a systematic and structured approach and in this article I outline how we, with some of the companies in the Software Center as well as outside it, apply the four steps described above to help companies proactively reposition themselves in their business ecosystem. The above was very brief, but feel free to reach out to me in case you would like to know more.

Your Ecosystem Has 50 Shades Of Gray

Another week on the road and another set of discussions around business and software ecosystems. Although it is clear that the ability of a company to build an ecosystem around its business is replacing company size as a, or perhaps the, key differentiator, I notice that there is a lot of confusion about what being an ecosystem-centric company actually means in practice.

The confusion around the practicalities of being an ecosystem company is fourfold. First, the proponents of an ecosystem in the company seem to have a strong preference for a collaborative approach. Second, typically confusion exists about competition between the ecosystem partners and the keystone company. Third, it is often unclear how the platform underlying the ecosystem can evolve and what functionality is suitable to be included in the platform. Finally, the strategic business rationale for being in the ecosystem is frequently unclear and unaligned in the organization. Below we discuss each of these topics in more detail.

The notion that an ecosystem is inherently collaborative seems to originate from the open-source community as well as other collaborative communities such as the one around wikipedia. However, although my prefered definition of business ecosystems mentions topics such as symbiotic relationships and co-evolution, the fact is that parts of ecosystems can be highly competitive. This is true in natural ecosystems as well as in software ecosystems. For instance, on the appstore for your phone, there are dozens, if not hundreds, of apps for task management that compete with each other. The key decision as the keystone player is to decide where to use collaboration and where to use competition.

Second, some seem to assume that the keystone company, i.e. the company providing the platform for the ecosystem, can not compete with others in its ecosystem. The argument is that this would discourage partners to join the ecosystem. In practice, it is about clear rules rather than about competition. Most companies will end up with three areas of functionality on top of the platform that they provide for the ecosystem. The first area is where the company provides functionality on top of the platform and it does not allow for competition from others. For instance, for a long time, Apple did not allow other browsers on its iPhones.

The second area is where the keystone company competes with ecosystem partners. For instance, Microsoft allows other document, spreadsheet and presentation applications on Windows, but does compete directly with these applications with its own Office suite. Third, there are areas where the company decides to not compete with its own solutions on top of the platform. Here the ecosystem partners have the playing field to themselves and only compete with each other. Clearly defining, and then communicating, what functionality belongs in each of these three “buckets” is a critical strategic decision.

The third topic is a common misconception that the keystone company can never incorporate functionality into the platform that currently is provided by ecosystem partners. It is important to

remember that any platform is constantly sinking into the morass of commoditization and needs to continuously incorporate new functionality in order to stay competitive. The best way to do so is to identify functionality currently built on top of the platform that is broadly used and that is commoditizing at the application level. Once identified, this functionality needs to be incorporate into the platform and this will likely affect ecosystem partners as well as the app teams working at the company itself. Not incorporating this new functionality will cause the platform to become irrelevant over time.

Finally, one of the topics that never fails to surprise me is the frequently total lack of alignment on the expected purpose of adopting an ecosystem approach. Smaller companies often look to build an ecosystem as a competitive response towards larger companies that offer a broad suite of functionality that the smaller company can not afford to build by itself. A second argument frequently used is that a successful ecosystem will drive sales of the core platform and thus generate indirect revenue. A third argument is direct monetization through app stores or the like and capturing a certain percentage, typically 30%, of the revenue generated by ecosystem partners. And finally, companies under attack by competitors with successful ecosystems adopt an ecosystem strategy as a defensive strategy. The lack of clarity concerning the strategic purpose of the ecosystem initiative will cause significant amounts of discussion, confusion and disagreement.

Concluding, as the title alludes to, ecosystem strategies can be very diverse in their realization and it requires significant effort, in my experience, to achieve clarity and alignment on the strategic intent of the ecosystem and the operationalization of this strategic intent. Your ability to build successful ecosystems around your key products and platforms is increasingly the differentiator that is replacing company size. Small companies will successful ecosystems can disrupt large companies, but of course large companies with successful ecosystems will have a significant moat around their business. So, if you’re not good at this already, you better start building the capability to build and evolve business ecosystems

Organization & Change Management

The final section of the book is concerned with a generic topic that comes up very frequently: organization and management of changes in the organization. Although these topics are not specific to digitalization, these play a critical role in any digital transformation. There are many companies out there that rationally know where they need to go as an organization and at the same time experience paralysis and a complete inability to transition towards the desired state.

The key factor severely complicating change management is the hierarchical, functional way in which companies are organized. Although hierarchical and functionally organized companies have advantages, the very fabric of these companies is resisting change. For most companies, any change requires agreement from all stakeholders and a single “no” will basically torpedo any change initiative. This often leads to a situation that by the time the company is ready to implement change, the need for it is so blatantly obvious that the competition has bypassed the company. This results in a no-win situation: even if the company successfully implements the change it’s “too little; too late” and the change does not offer any competitive benefit to the organization.

In many companies, the answer to almost anything that is perceived to ail the company is a reorganization. For some reason, many belief that a reorganization will solve all ails and lead to a wonderful future. In practice, however, reorganizations that fail to follow the BAPO principle (Business -> Architecture -> Process -> Organization) do little more than reshuffle cards in the deck and kick the can down the road for the organization as whole.

In response to the challenges of hierarchical organizations, several alternative approaches have been developed. The primary mechanism to realize this is empowerment of cross functional teams that operate towards a quantitatively defined goal that supports global, rather than local optimization. In our research, we have learned that empowerment works very well in engaging and energizing teams and them taking ownership for the way they deliver on the set goal. However, if the goal is not sufficiently precise in its definition, it is very easy for teams to do their own thing and tell a story as to how their efforts contribute ot the overall good of the business ecosystem. Hence, teams need clear, pricise and quantitative outcome metrics to strive for as the alternative may well be vast amount of activity, but very little progress.

Concluding, in the next sections, we discuss several challenges that we have experienced in the companies that we collaborate with and some avenues to pursue to reach better outcomes than what the traditional approaches offer, based on the experience from us and many others.

Why Cross-Functional Teams Should Be The Norm

Another week of travel behind me working with great teams at really wonderful European companies. As I typically get involved to work with organizations to introduce new platforms or new businesses or to realize fundamental transformations, I work with cross-functional teams where individuals from different parts of the organization are put together in order to accomplish our goal.

As in these teams, we work on changing some part of the organization, the discussions almost always start with sharing war stories describing things that went horribly wrong due to systemic issues in the current organization. Many of these issues are due to a functional setup of the organization where teams are organized according to their respective functional skills or responsibilities and communicate through documents and periodic meetings.

When reflecting on these systemic issues, I realized that I see three syndromes of functionally organized companies: symptoms caused by the development life, the product hierarchy and business/R&D dichotomy.

To start with the development life cycle, many organizations (still) have separate requirements engineering, development and testing teams. In these organizations, it is very typical to experience high degrees of rework, i.e. late changes to the system under development. This rework is often caused by different interpretations of the same requirement by these three different groups and it's quite typical to see testers, developers and requirements engineers arguing about the right interpretation of the written requirement. The root problem is, of course, that written requirements only capture the 10% that are easy to write down and conveniently ignore the 90% that is hard to document.

The product hierarchy issues that I have experienced are typically concerned with lack of alignment between systems engineering and software engineering. As many companies that I work with are traditionally metal bending companies, the systems engineers often have a mechanical background and have difficulties understanding software. For instance, in one company, the system requirements were written in such a way that it was impossible for the software of the system to specify its own requirements and perform its own tests. Failing to recognize that software is on par or even more important than mechanics and electronics is highly detrimental. Software allows for continuous deployment of new functionality, real-time collection of performance data from systems in the field and enables fundamentally new business models.

Finally, the dichotomy between the business side and the R&D side of companies is still quite prevalent in many organizations. The idea is that the business side knows what customers want and can order functionality from the R&D organization. The problem is of course that these

customer needs are communicated through documents that have the same concerns as the requirement specifications described above.

The root of the issue is that people just need to get in the same room and wrestle an issue to the ground. To discuss until everyone in the room has the same understanding of what we're looking to accomplish. And rather than going back to the respective functions once done with the meeting, the team should stay together and work to accomplish the desired outcome. This is of course the recipe for cross functional teams.

Functional organizations became the norm in the 20th century because they allow for easy scalability, efficiency of the functional tasks and, as a consequence, lower cost for delivering a defined system or product. The problem is that this only works in an extremely stable, predictable context, such as manufacturing lines where the same product is churned out year after year. In these contexts, the product is a commodity and the only differentiator is cost. Consequently, squeezing the last drop of blood from the stone is the only viable path.

The problem is of course that everything that is stable and predictable is or will soon be automated. The vast amount of work in organizations performed by humans is that which is not predictable, highly dynamic and requires coordination across many disciplines in the organization. When humans are involved, the focus is on innovation and delivering (and measuring) customer value. And this is not a question of local optimization inside a single function, but rather requires contributions across all functions.

The final issue is of course one of scale: how does a large company move away from a functional structure. To be honest, I think we can already see the first answers to this question: many companies are moving towards network structures where different nodes, inside and outside the organization, provide specific systems and solutions and operate in a market place. The nodes themselves are not overly large and can work with sets of cross-functional teams.

Concluding, the syndromes associated with functional organizations are widely spread and complicated. Many accept the symptoms as unavoidable and as facts of life, just like gravity and aging. However, this is too easy and a cop-out. We need innovation in the way we organize work as much as we need it in the products and services we deliver to our customers. Don't accept the frustrating and inefficient ways of work. Don't just complain and whine about to to your colleagues. Analyse the symptoms, brainstorm solutions, try out these solutions and iterate. Remember that the only constant is change and that the only static companies are dead ones.

The End of Reorgs

The CEO of one company that I work with lamented that his leadership team was discussing a reorganization again. And that although he was not again reorganizing in itself, the last reorganization had not really solved any of the issues the company struggled with. Consequently, he was concerned that this new reorg would again throw up a lot of dust, but again not solve any of the core issues.

Reflecting on his words, I realized that this is my experience in every reorganization that I have been involved in, either as an employee or as a consultant. The process is obvious for anyone who has been part of leadership teams: someone brings up a problem and then builds the case that the problem is not unique, but actually the symptom of a systemic root cause. The next step is to show that not addressing this systemic problem will cause the company to go to hell in a handbasket. Subsequently, the protagonist shows that this systemic problem is due to the organizational structure and alignment of roles and responsibilities and that organizing differently will solve all the concerns, has many additional benefits and will solve world hunger, global warming, etc. After some debate, the others in the team feel that being against the reorg will make them stand out like a stuck-in-the-mud traditionalist, which would be bad for their careers, and start to sing the praises of the reorg as well. The end result is that the reorg is approved, everyone in the management teams tells publicly how important it is to get this done quickly and that they need the support from everyone in the organization and the change is initiated.

What happens in practice? In reality, dozens, hundreds or sometimes thousands of people's professional lives are affected. Ongoing priorities get disrupted. The organization falls into a period of paralysis as it's unclear who actually is responsible for things. People leave as they don't want to be part of a dysfunctional organization. Etcetera. After some time, the organization recovers, often largely due to the informal organization and people taking responsibility despite, rather than thanks to, the formal organization and the output recovers. Management takes this as a sign of success and beats themselves on the chest and congratulate themselves on having saved the company from certain doom. Then, after some quarters or at most a few years, the same problems that were present before the last reorg start to reappear again and the whole process repeats itself. Like the movie Groundhog Day.

So, it is time to stop this madness. Reorganizations, in and of themselves, do not solve ***anything*** and are harmful. Not a little, but very harmful. In too many organizations, they are used to show activity and decisiveness of key individuals that seek to protect or improve their position over the back of company. Although this is a bit of an oversimplification, there are at least three problems that I see occurring in practice: the pendulum problem, the culture problem and the inconsistent business strategy problem.

The pendulum problem is the situation where the company swings back and forth between two extremes in terms of organizational structure. This can be between centralization and decentralization, between separating and combining business and R&D, between being platform centric or being customer or product centric, etc. The core of the problem is that each extreme has a set of benefits and a set of disadvantages. The company is unable to accept the advantages and to put mechanisms in place to deal with the disadvantages of the organizational setup. Hence it swings back and forth.

The culture problem is where the culture in the company is stronger than the official strategy and structure. A typical example is the hero worship culture. Here, the official strategy of the company is to ensure quality and reliability through processes, consistent ways of working and other mechanisms. However, whenever there is an issue with customers in the field, a product that is late or some other unplanned situation, the hero culture takes over and a key individual or core team gets complete free reigns to solve the issue at hand. And of course get applauded when they to wrestle the issue to the ground. Rather than addressing the key issue at an architectural, process and organizational level to ensure that these problems can't occur, the organization relies on heros to pull them out of the fire. For those working in the automotive industry, the notion of "task forces" represent an illustrative example of this pattern.

The third type of problem is concerned with inconsistent business strategies. This problem occurs when the organization sets a business strategy that is inherently conflicting. For instance, some companies desire to be deeply customer centric and achieve high levels of reuse in the solutions they offer. Although this looks great at the powerpoint level, the fact is that you can't combine these strategies and expect to deliver on both. High levels of reuse require standardization and productification of functionality. Deep customer intimacy requires significant amounts of customization and customer specific development. Trying to solve this contradictory situation causes the organization to focus on platforms on the one hand and customer teams on the other hand without clear guidance on who leads and who follows. It then results into a highly politicized organization where the personalities of individuals determine the balance achieved in practice. Companies tend to then go through a series of reorganizations to resolve this inherent conflict without realizing that the problem can not be solved at that level.

Although there are several other symptoms and problems that I could have brought up, I would like to spend some attention on resolving these issues. The first tool that I have found incredibly useful (undoubtedly as I was part of developing it in the first place) is the BAPO model. As shown in the figure below, the BAPO model suggests that the best approach to design an organization is to follow four steps:

- **Business and business strategy**: Clearly define the business and business strategy that you're in and want to be in. Be explicit about what you define as the strategy to make you successful in the future.
- **Architecture and technology choices**: Based on the understanding of the business strategy, define the desired architecture for the products, solutions and services that you

are offering as well as the technology choices that allow you to optimally deliver on the business strategy.

- **Process, ways of working and tools**: The next step is to design the processes that the company should use to deliver. This also includes the ways of working and the tooling that helps the company automate the repetitive and standardized tasks.
- **Organization**: Based on the above and ONLY after having defined the first three elements should one even think about organization. And rather than going deep on management and reporting hierarchies, the focus should be on empowerment, autonomy of teams and quantitative goals that directly align with business success in terms of revenue and margins.

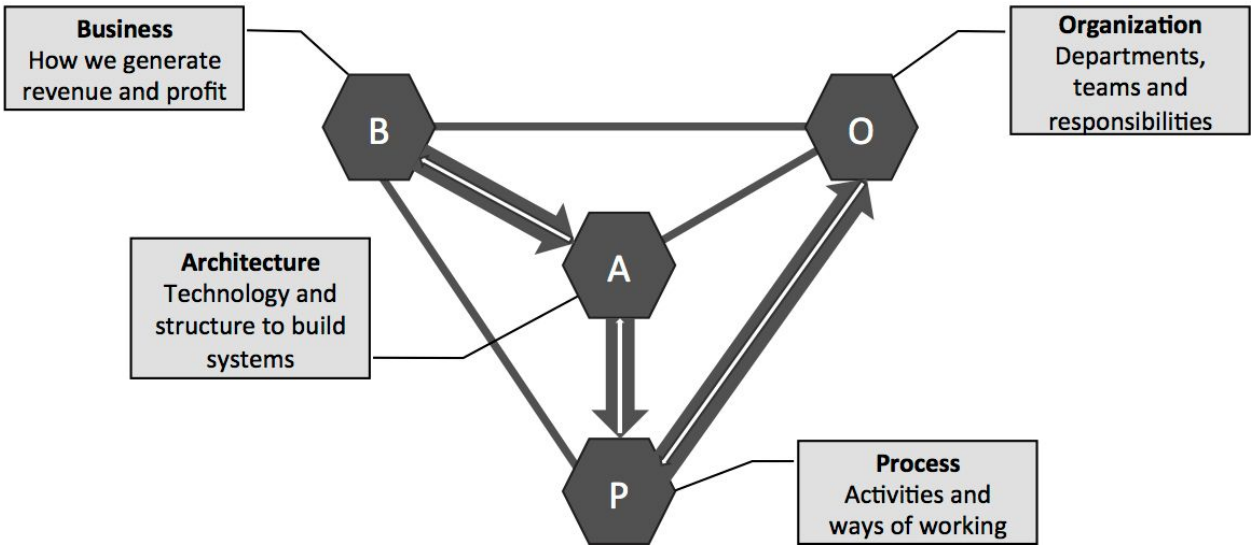


Figure: The BAPO model

The second topic that deserves mentioning is culture. As the saying goes, culture eats strategy for breakfast (and lunch and dinner). Companies with a strong culture are fabulous to work in as long as the culture aligns with the business reality in the business ecosystem. The strong culture requires very few formal governance mechanisms as people automatically do the right thing as the culture dictates what the right behaviours are. So, everyone in the organization feels empowered and free to do the right thing as many do not even realize that the culture is what is causing them to behave this way.

The moment the culture turns orthogonal to, or at least less than optimal for the business reality, the strong culture becomes a liability. In fact, the strong culture can become the very thing that sinks a company. Driving cultural change is notoriously hard, but part of the keys is to drive for self-reflection in the culture. This means that everyone frequently reflects on why the organization behaves in certain ways and whether this still is the right approach in the current situation. A self-reflective culture allows the culture to evolve in accordance with the changes in the business context in which the company operates. In fact, empowered organization without

managers and high levels of autonomy for individuals and teams use self-reflection as a mechanism to drive evolution and continued relevance in the marketplace.

Concluding, the vast majority of reorganizations are absolute and utter waste that serve the needs of a few at the expense of the many. We should stop this idiocy and be highly skeptical of any proposal for reorganization. Instead, getting crystal clear on the BAPO for your company and aligning the organization around that is critical for prolonged business success. In addition, ensuring a company culture that is self-reflective and that continuously evaluates itself for alignment with the business reality in the ecosystem is critical. In most cases, a reorg is the easy way out that allows leaders to avoid the real, hard work that is required to keep the company competitive. Don’t fall into that trap and focus on what really matters!

On The Scope Of Feature teams

One of the discussions that comes up in many of the companies that I work with is the scope of feature teams. The traditional view of a feature team is that it can make changes and add code to any component that needs to be changed in order to build the feature that my team and I are responsible for. Using feature teams has many advantages, not the least to avoid the situation where component teams working on commodity components find ways to keep themselves busy even if their functionality does not provide any business value.

At the same time, there is a limit to the scope a feature team can cover in terms of the breadth and depth of functionality of a large and complex system. Some organizations associate a “skills needs” profile to items on the backlog and only teams that have the right skill profile can pick those items from the backlog. Similarly, I often advice companies to maintain part time component experts that can coach feature teams to make changes in the component in a sustainable fashion. These approaches allow for better scalability of feature teams across a large and complex system.

The challenge still is that when we scale to complex and large systems, there is a point where we need to draw a boundary and interface that indicate the end of the scope of these feature teams. The question is how to determine at which point this boundary is best established. There are at least five approaches that one can take, I.e. architectural, deployment-driven, skills-driven, organizational and 3LPM-driven. Below we discuss each of these in more detail.

The system architecture breaks the system in high level components and subsystems. Here these components and subsystems provide a first indication of where to draw the line. Many architectures exhibit a layered architecture and depending on the content of typical work items on the backlog, it may make sense to constrain a feature team to one layer of the architecture. This has to be carefully evaluated, though, as I have experienced many architectures where virtually every new feature of functionality cross-cut the architecture and required changes across multiple layers. If it matches the work, though, following the architecture may help to reduce the amount of hidden and complex dependencies between components and subsystems.

The deployment-driven approach assumes that different parts of a system deploy at different time scales. In a typical embedded system, the mechanical parts of a system evolve on a multi-year time scale, hardware and hardware-close software on a yearly or half-yearly time scale and more independent software releases every sprint, monthly or at most quarterly. In this case, teams operate only in the part that deploy with the same deployment frequency.

The skills-driven approach is especially used in situations where the company offers highly specialized functionality in one area and the rest of the system provides the necessary support functionality that is not (nearly) as specialized. The subsystem that contains the expert

knowledge is managed by one set of teams and the rest of the system by another set of teams. It is important to realize that in most organizations, there is a overestimation of the complexity of software and an underestimation of the ability of feature teams to operate across the architecture.

Especially in distributed organizations where development teams are located in different parts of the world and work in different time zones, there often is a desire to allow for as much local autonomy as possible. Teams from different locations touching the same code can easily lead to complications, especially if the continuous integration environment is not very mature. Although this approach breaks the BAPO principles (see here for a description of BAPO), if the allocation of responsibilities can be organized this way, it may positively affect the productivity of the organization.

Finally, in an earlier blog post, we introduced the Three Layer Product Model (3LPM) where we organize functionality into innovative, differentiating and commodity functionality. Each type of functionality is assigned to a layer and each layer has its own focus. Teams working on the commodity layer focus on reducing the total cost of ownership for the commodity functionality by standardizing, removing variants, replacing bespoke functionality with COTS or OSS components, etc. Teams working on differentiating functionality focus their energy on maximizing customer value through experimenting with alternative implementations, offering variants for specific customer segments, etc. Finally, teams working on innovation are expected to test as many innovative ideas with customers as possible against the lowest cost and effort per experiment in order to find the future differentiation. In this case, constraining teams to one layer of the 3LPM allows these teams to focus on their own target.

Concluding, determining the scope of feature teams in R&D is, for most companies, a non-trivial problem. In this post, we identified five different approaches to defining the scope and scale of feature teams. However, although I work with dozens of companies, the number of cases that I have access to is still limited. Please share your experiences and learnings in the comments!

Too Many Caesars; Too Few Romans

One of the most interesting patterns that I have observed again and again is that any organization tends to create layer upon layer of bureaucracy over time. Although I can't find the reference, I remember reading somewhere that any organization increases bureaucracy with 0.7% per year. In my experience, this is a gross underestimation.

Often the increase in bureaucracy is caused by the most innocent of reasons, such as a lack of coordination, perceived lack of control by senior management, a crisis that should be avoided in the future by more oversight, etc. The result however almost always is another management layer being introduced, more administration, less value adding activities. On top of that, a further increase in the rigidity of the organization. The more organizational structure and ingrained work processes, the harder any transition and change will be to realize.

One of the positive examples of organizations being able to reduce management hierarchy is the transition to agile ways of working in software R&D. After the transition, rhe result often is that there are much fewer manager positions and the percentage of people doing value adding work goes up significantly. In fact, I once heard a story about a bank that had, in an attempt to bring the IT cost under control, fired everyone in IT that did not write code. Although this might be overly extreme, the bank didn't seem to suffer much…

The problem described here, often referred to as "too many chiefs; too few (American) Indians", doesn't constrain itself to companies. Government institutions and universities often are even worse offenders of the principle. Returning to academia after having worked in industry for a decade hammered home that point quite nicely.

So, what to do about it? Well, My experience is that every organization needs a solid "spring cleaning" once every couple of years. Basically, a redesign of the organization with as few layers and as wide spans of control one can stomach. Allow front line staff to take significantly more responsibility. Have everyone in the old organization apply for jobs in the new organization and find a way to "retool" those without a role in the new organization for the remaining jobs. In practice, this often means that people in management positions need to return to doing value adding work. Once the new organization is in place, one can slowly introduce the few missing elements of coordination that are really needed.

Concluding, almost every organization that I work with (or for) suffers from the "too many caesars; too few romans" problem. It results in all kinds of undesired consequences, ranging from too high overhead, reduced competitiveness and employee satisfaction to organizational rigidity and resistance to change. Instead, focus on resisting the calls for new overhead inducing activities and center all work around the customer and creating, capturing and delivering value. And when necessary, do some serious spring cleaning to clear out the accumulated dust.

Autonomy, Empowerment, Alignment and Coercion

The greatest accomplishments of mankind have been achieved by groups, often large groups, of people. From the pyramids and gorgeous cathedrals built in the past to the modern accomplishments such as putting a man on the moon or the creation of the internet, none of this could have been created by an individual but required large groups of people.

To accomplish these big goals, it is not sufficient to bring together a group of people and wait for magic to happen. Instead, the actions of these people need to be aligned and coordinated with each other in order to create an outcome that is larger than the sum of the parts. Although alignment is a polite word, it inherently means that individuals will have to subject themselves to the will of the group, or more specifically, to the leaders emerging within the group. It implies a reduction of autonomy and freedom.

Over the millenia, humans have used a variety of mechanisms to align themselves. The earliest tribes defined clear expectations on each individual and would expel those that did not comply. And in those days, being alone was an almost certain death sentence. Later on, slavery was used a a mechanism to align the actions of many people. Even if it is unclear whether the pyramids were built by slaves, there are many examples of slaves accomplishing great things. And even if slavery has been abolished in most of the western world for centuries, in many agrarian communities, workers were virtually enslaved by the farming landowner as they either had to work for him or starve.

Although the aforementioned alignment mechanisms are based on one human controlling another, there are more subtle mechanisms employed as well. Organized religion has been used as an incredibly powerful tool to steer the actions of people. And of course, totalitarian regimes use their “religion” as a mechanism to control the population.

In these modern times, at least here in the west, we are of course expected to, in a certain way, enslave ourselves, and live in accordance with our own rational decisions and ambitions, rather than being driven by our instincts, emotions and habits. Although we may feel that these are our choices, of course we are driven to a large extent by our upbringing and the culture of the society we live in.

The reason to explore this topic is because I spend a lot of time thinking about organizations and about the challenge of organizing groups of people. During the last 100+ years, the answer has been hierarchical organizations. And make no mistake: hierarchies work! According to Jordan Peterson, lobsters organize themselves in hierarchies and use the same hormonal system as humans, such as serotonin, to steer behaviour in hierarchies. Humans and lobsters separated something like 300 million years ago. There were not even trees on the earth at that time apparently. So, saying that hierarchies are inherent in our nature is entirely correct.

The problem with hierarchical organizations in that hierarchies are static and that power is driven by position rather than competence. So, rather than creating hierarchies that are meritocratic and temporary in nature, we are stuck with permanent fixtures that maybe were good for the business that we once were in, but most certainly no longer suitable for today's reality. Especially in European companies, many of the senior managers are enlightened in that they understand that going against the desires and expectations of their staff tends to result in a severely shortened career. The end result is, of course, an even more static structure as the underlings are unable to change because of lack of power and managers are unable to change because of their organization resisting.

On a weekly basis, I experience exactly this problem and I see people waste up to 90% of their working hours to just manage the internal organization as they're trying to accomplish outcomes that are desirable for the organization as a whole. That everyone even says they want! I think it is hard to underestimate the fantastic amounts of waste in large, hierarchical organizations. As an example: in an embedded systems company that I visited earlier in the year, a software development organization in that company employed around 700 people of which less than 70 actually had a compiler installed on their computer. This means that more than 90% of people were doing other things than building software!

The answer to this challenge that several startups, consultancy companies and even larger companies are exploring is, of course, autonomy and empowerment. If it is the boss of people that is holding them back from doing the right things, then let's get rid of the boss. Then everyone can take the decisions they feel they need to make and act on these. Although I am very much in favor of providing people with more, rather than less freedom, in a part of their life where many spend most of their waking hours, the point missed by many is that the basic problem does not go away: we STILL need to align the work of everyone if we're hoping to accomplish the outcomes that can only be achieved by groups of people.

When studying organizations that pride themselves on being empowered and non-hierarchical, it quickly becomes clear that alignment still takes place, but in different ways. The least intrusive way is through architecture. Teams are assigned to a specific part of the architecture and are empowered to operate within their part of the system as they see fit. The second model, often combined with the first one, is through data. Teams receive quantitative targets to achieve within their part of the system, such as a specific increase in conversion on an ecommerce website, and are at liberty to decide how they achieve these targets. In this case, all the alignment takes place inside the team.

In situations where it is necessary for multiple teams, or even the entire organization, to align we end up in other mechanisms. Of course, there is the coordinator - the person appointed to coordinate a certain topic across the organization. Depending on this person being a police agent or a coach, this tends to reduce the empowerment of teams somewhat or significantly. In fact, in some of the companies that I work with, some teams have to contend with up to 10 "coordinators" that often have conflicting requirements. Examples include security, safety,

export restrictions, legal constraints, region representatives, customer advocates, corporate social responsibility folks, environmental regulation experts, etc. If not managed carefully, teams become the victims of these different coordinators, are not empowered at all and productivity is reduced to a crawl.

The second mechanism is the use of communities. Here, every team or unit has a representative in a community that is targeted to address one dimension of alignment that needs to be accomplished inside the organization. Although many think of communities as empowering, this is in fact not the case. The main difference is that instead of a manager deciding, it is now the community that decides. Individuals and teams are still expected to follow whatever decision was taken in the community.

The third mechanism, especially in startups, is the use of a strong culture to align people around. The culture basically expresses a set of beliefs and behaviors that capture “what we do around here and how we do it”. A strong culture is incredibly powerful, but the challenge is similar to hierarchies: changing the culture is incredibly difficult (in fact, harder than changing hierarchies) and especially for long-lived companies, evolving the culture to stay competitive is a necessity.

Fourth, there are organizations that embody a mission that is considered by their employees as well as society at large as meaningful and relevant. Of course, many social entrepreneurs use this mechanism to build their company around, often using a triple bottom line of social, environmental and financial factors. However, also for-profit companies can use this strategy assuming they are sincere about it. For instance, Tesla making its patent portfolio available to other companies looking to build electric cars is a good example of a company delivering on its mission to rid the world of cars propelled by fossil fuels.

My goal with this article was to outline the various mechanisms that exist for aligning people in large groups. There is a need for alignment, but whenever mechanisms for alignment are put in operation, the autonomy of individuals and of teams will be constrained. The high degree of interest in empowered organizations is driven by the experience of many that traditional alignment mechanisms easily degenerate into a form of coercion. Coercion damages the integrity of individuals and the result is an organization where its members are not engaged (see earlier blogpost), only do what they’re told and show up only for the paycheck.

In my experience, mission-driven organizations tend to have the easiest time aligning the actions of their members and require the least amount of force to accomplish alignment. These organizations tend to spend some or significant resources on translating the, often qualitative, mission into concrete structures and quantitative goals. Although this is a form of alignment, it allows for a broad engagement of everyone in the organization which limits the negative impact of imposed alignment.

Concluding, when working with your organization as a leader, there are three main take-aways. First, ensure that your organization has a clear and powerful mission that you and everyone in the organization is willing to put their lives in service to. This is critical to express and, even more important, to act in accordance to. We all know the value statements of traditional companies that are all too frequent a farce. Second, whenever possible, rely on meritocratic, temporary hierarchies instead of permanent structures. This allows for a much simpler approach to changing the organization in response to changes in the market. Organizational changes should be small and continuous rather than episodic and large. Third, for every point where you believe alignment is required, carefully examine whether the benefits of autonomy for individuals and teams possibly outweigh the benefits of alignment. In many cases, allowing for divergence increases engagement as people feel in control and also allows for experimentation with alternative ways of doing things that the organization as a whole might learn and benefit from. The aforementioned continuous evolution of the company is more easily achieved if more diversity within the organization is facilitated.

Alignment is a necessary mechanism, but it needs to be wielded carefully and with restraint if one seeks full engagement from everyone in the organization. The notion of the company as a carefully tuned machine is really outdated as all the work that can be “tuned” is repetitive and should have been automated if it is still conducted by humans. The remaining humans in the organization are then by definition focused on innovation and change and consequently need to be empowered and autonomous to deliver on their and the organization’s potential.

Overcoming Inertia in Organizational Change

This week we organized the reporting workshop of the Software Center that I am director for. It's always a great event with lots of participants from the ten companies participating the center and researchers from the five universities that are members. We present a variety of technologies, techniques, methods and frameworks that have been developed based on the research that we conduct together with the companies. The areas range widely but include topics such as software architecture debt management, data-driven development and experimentation, software ecosystems, continuous integration and deployment, organizational empowerment, etc.

The common theme across the participating companies is that they want to implement many of the results in their organization. And the results are concrete and tangible, so there is no need to conduct large amounts of additional work. It's just a matter of implementing the change in the organization. And of course this is where the fun starts, because organizational change is never easy.

So, why is organizational change so hard? Although reels of paper have been filled on this topic in various research communities, my perspective on this, formed by two decades of driving change both as an employee or as consultant, is that it comes down to three main factors: **habits**, **loss aversion** and the **unintended consequence of hierarchy**.

Most of us view ourselves as rational human beings who take conscious decisions throughout the day. Research, however, shows that up to 95% of the day we are habit driven creatures who run on automatic pilot for most of the time. For instance, most of us have experienced a situation where we drove to work or another well known route, arrived and could not remember how we got there. *Changing habits is really difficult and many ways of working that would need to change as part of the organizational change are habits that are deeply embedded in the organization*.

Change always means stopping to do things in a certain way and doing them in a new way. It may have organizational implications and it often changes relationships between individuals, teams and organizational units. Research shows that we experience loss about three times are severely as gaining something. That means that losing €100 stings three times as much as winning €100 in a bet. In organizations, everyone evaluates, often unconsciously, the risk of losing something valuable as much as the gains that are associated with a change. *Change resistance occurs when the perceived risk of loss outweighs the perceived gains of the change*.

The funny thing in hierarchical organizations is that the perception is that the HIPPO can just decide and the organization below the HIPPO will just execute. In practice, senior leaders are reluctant to take radical steps if their teams are not supporting the change. There are many reasons for this, but at its core the pattern is that organizations like stability and predictability

and consequently people who are the least likely to cause trouble, surprises or issues are the ones that get promoted. This means that in most organizations, the higher up in the organization one gets, the more risk averse behaviors are. As a result, changes require support from everyone affected and a single “no” can torpedo the entire change initiative. *Hierarchical organizations are exceptionally good at maintaining status quo and equally bad at realizing change*.

If you had started to read this article with the hope of me presenting a silver bullet, I’m afraid that I will disappoint. However, in my experience there are three things that help. These are **focusing on your circle of control**, **experimentation** and **strategic use of outsiders**.

Stephen Covey often refers to the three circles. The circle of control, the circle of influence and the circle of concern. Any change of any relevance will cut across all three circles. I’ve seen many trying to drive change in their circles of concern and influence and fail as the initiative is easily viewed as lacking authenticity. Why? Because the person instigating change told others to change but didn’t do anything him or herself. So, **any authentic change needs to start from your circle of control**: first change yourself and those in your team and only then ask others. In the words of Ghandi: be the change you want to see.

Second, I’ve frequently seen change presented as a cast-in-stone “this is how it’s going to be” proposition. This gives people great opportunities to identify parts in the proposition that they like that they can then use as a reason to not support the change. Instead, a focus on the desired outcome, that presumably everyone can agree on, and *opening up for experimentation creates much less resistance*. It’s much easier to get others to agree to trying something out to find out if it works.

Finally, especially in my engagements with many companies, I often get the feedback something that they have failed on several times before now finally was successfully realized with my involvement. Reflecting on this made me realize that the strategic use of outsiders in a change process frequently is an incredibly important success factor. The existing relationships in the organization are subtly disrupted by the presence of an outsider and everyone is more focused on protecting the interest of the company as a whole than their own little kingdom. In many cases, *strategic use of outsiders can lead to breakthroughs that significantly accelerate the rate of change*.

Concluding, organizational change is hard and often well-intended employees that believe the change is critically needed shy away from taking action. Understanding the reasons why change is hard, including habits, loss aversion and the unintended consequence of hierarchy, and employing a mental model to accelerate change with elements such as focusing on your circle of control, experimentation instead of “cast in stone” plans and strategic use of outsiders, can help organizations, teams and individuals overcome the inertia of organizational change.

Stop Trying to Change Your Company!

As I work a lot with driving change in software-intensive companies, the people that I work with typically have already an idea of what needs to happen. However, many of them struggle with organizations that are incredibly resistant to change. In fact, someone recently shared his view with me that organizations are designed to be resilient in the face of different forces that are exerted and that the resistance to change is a feature.

One of the reasons why change is so difficult is that the leaders in the company want to have their cake and eat it too. On the one hand, they seek to implement a change, e.g. an agile transformation, adoption of continuous deployment, new data-driven services, etc. and they want this to positively affect the entire organization in one go. On the other hand, of course, these leaders want the benefits associated with the change but without needing to change anything that is the norm today. The Germans have a great saying for this that translates as “wash me, but don’t make me wet”.

The result of all this is that the people that I work with and that are trying to realize change in their organization often are frustrated and tired because of their constant battling of what often feels like windmills a la Don Quichotte. Driving change in organizations is an area where reams of paper have been filled with wonderful conceptual notions, but where my favorite definition of “theory” applies: A theory is beautiful thing killed by a gang of ugly facts. For all the books, theories and experts, in my experience, change management in practice is messy, inefficient and frustrating for everyone involved.

So, what is going on here? To me, there are at least three issues that are at the root of this problem, i.e. risk averse leaders, the gap between responsibility and authority and the mismatch in incentives. Below, I discuss each of these issues in more detail.

First, in most hierarchical organizations, individuals get promoted to higher positions because of their predictability and reliability. Imagine you are a higher level leader who has an open position in your leadership team. Who will you pick to fill the spot? A noisy troublemaker or a reliable employee who always dots the “i”s and crosses the “t”s? In practice, most of us will pick someone who will not rock the boat and can be relied upon. As a consequence, however, the result is that the higher up you go in the Christmas tree of the company’s hierarchy, the more risk averse the individuals tend to be.

Second, because of the risk averse nature of leaders, these leaders tend to avoid taking personal responsibility for implementing relevant changes. The arguments are often very convincing and leaders have learned to rely on delegation as an effective mechanism to increase reach. However, when it comes to driving change, the model falters as there are no existing processes, ways of working and tools to fall back on. Most leaders appoint an individual, typically the person that has been clamoring the loudest, to drive a transformation

and assume that this person can magically drive the necessary changes by sheer belief and personality, but without the necessary authority. The gap between responsibility and authority tends to lead to a stalemate in the organization or, as one of the people that I work with calls it, a guerilla war.

Finally, because the individual or team looking to drive the change is different from the teams that actually need to change their behavior, ways of working, tooling and often even their norms and values. As the organization at large is responsible for delivering on the current business priorities, their incentives are focused on short-term business results. The change team's incentives are concerned with long-term effects of the changes that they are looking to realize. The conflicting incentives will only slow down the change. The operational teams are made up of well-intended people that often want to realize the required changes, but are incentivized to maintain the status quo. And as humans, according to research, experience the pain of loss about three times as strong as they experience the joy of gains, changes that are not obviously improvements are generally resisted by our basic, human traits.

Realizing effective change requires, in my experience, at least three key elements: charismatic leadership, focus and time constraints. As long as we have hierarchical organizations, leaders will model behavior that is emulated by the rank and file. If the leader interested in accomplishing the change does not take personal responsibility to drive this and makes this his or her most important task, the rest of the organization will be even less inclined to get into motion. And the leadership has to be charismatic as there is a significant need to paint the bright new future to overcome the fear of loss in people.

Second, in many organizations there are many change initiatives ongoing in parallel and each of those have very long running periods. The typical argument is that change is hard and we need to give it time. In practice, however, many change initiatives means that the organization is unable to set priorities and because there is no real, short-term deadline and these change initiatives may actually negatively affect each other, the end result is often no change whatsoever.

My recommendation is that organizations, at the relevant scope (team, business unit, etc.), should have one and only one change initiative ongoing and the time period for this initiative should not be longer than 12 months and preferably significantly less, such as three or six months. The change initiative should be carefully planned, committed and executed as a project with full backing of management. Often, the period before kicking off the initiative can be used to create awareness and build understanding in the organization. However, once the initiative is set in motion, the execution will happen come hell or high water.

Concluding, in many organizations, everyone talks about change and has their own pet change project. Many are trying to influence the organization, but the end result in most cases is exactly nothing. We need to stop *trying* to change the organization and start to *really* change the organization. As Yoda famously said: Do or do not. There is no try!

Why Organizing Into Functions Kills Change

This week I spent in the lovely Dagstuhl buildings, a location gracefully funded by the German government to allow for exchange between German and international academics. Our goal was to, in reality, to create a new academic field of study around “software intensive business”. The interesting thing is that everyone understand that the world is digitalizing and that value creation and capture are increasingly driven by software, there is no research community that actually studies the field. Instead, we have researchers from software engineering, information systems and business schools who study aspects of the field from the perspective of their “core discipline”.

The result of researchers from different fields studying aspects of what, to me, clearly is an independent field of study is that the new field does not receive the attention and recognition that it deserves. In fact, the way we are organized into academic fields of study is clearly showing down the emergence of new, more relevant, fields of study because of inertia and conservatism in the academic community. (Contrary to popular belief, my experience is that that the academic community is most resistant to new ideas of all communities that I have been part of.)

During the days in Dagstuhl, I realized that there is a wonderful parallel to most companies that I work with: we’re organized into functions that deliver part of a known end-to-end process and the interfaces between the different teams need to remain constant for the process to work. Of course, in practice change in business is relentless and there is a need to continuously adjust to the changes in the environment. However, the existing functions in the company are actively resisting change as it violates the attempts at local optimization that each function seeks to achieve.

The alternative, cross-functional, multi-disciplinary teams, is much better to adjust to changes in the environment. However, this requires a fundamental departure from the traditional functions that most organizations and individuals are loath to undertake. There are risks associated with one’s career and ability to develop as a professional, it requires a healthy willingness to engage with topics that are far from one’s profession which feels uncomfortable for anyone who has been rewarded one’s entire life on what one knows and, finally, it requires a level of accountability as teams own responsibility for often highly visible outcomes that affect the company as a whole.

In a fast changing world, however, I can’t come to any other conclusion than that it is by far preferable to organize cross-functionally as a primary principle and then to find ways to allow people to also develop in their functional skills as a secondary principle.

Conclusion

The digital transformation is so challenging precisely because it is so transformative at all levels. It affects every function, role and individual in the company as well as the ecosystem in which the company operates. In this book, we explored many aspects of the elephant, specifically organized in the ten themes that make up the chapters.

Fundamentally, however, the digital transformation is concerned with replacing one business operating system with another one. The traditional business operating system is concerned with transactional business models, a relatively simple one-dimensional value network, deeply integrated system architectures focused on minimizing bill of materials, a waterfall style process optimized for mechanical engineering, hierarchical organizational structures favoring the establishment of HIPPOs and a culture in which “atoms” are prioritized over “bits”.

We and our companies need to transition to a new digital business operating system. In that operating system, business is focused on continuous value delivery, a multi-dimensional business ecosystem where monetization takes place in multiple ways, system architectures that allow each type of technology to evolve at its own pace and supports continuous deployment as a mechanism for continuous value delivery, operates its business processes based on the principle of fast feedback loops, organizes itself in empowered, cross-functional teams and embraces a culture where “bits” are prioritized over “atoms”.

Although installing a new operating system on a computer is a relatively trivial task, unfortunately the same is not true for organizations. As humans are involved in the process, the basic beliefs, norms and values of us as employees drive our behaviours in ways that are hard to externalize. We all have habits that are very hard to break and company habits are, in my experience, even harder to break than individual habits.

In addition, the wide adoption of IT solutions, which has been great for productivity improvements, has also codified and embedded work processes deep into the bowels of companies. This means that even if we have convinced our colleagues to change with us, the result may still be that the systems, procedures and processes of the organization are inhibiting the changes that we’d look to establish.

Another challenge for most companies is that even if everyone agrees that the changes are needed at a high level of abstraction, developing the concrete and specific solutions for the company requires experimentation and trial and error in order to figure out the best way forward. Organizations easily end up in deadlocks when there is fundamental disagreement concerning the specific way forward that is broadly agreed upon at the abstract level.

What Should I Do?

As we discussed in the section above, replacing one operating system with another is a challenging task that will take significant, concerted effort across the company and likely even throughout the business ecosystems that your company is part of. Based on my experiences of working with dozens and dozens of companies, the best approach to accomplish this transformation is by combining a holistic, end-to-end perspective with a laser focused, narrow slice of end-to-end change and maintaining a clear, preferably quantitative, dashboard on the state of the transformation.

In my engagements with companies, we work following a specific process consisting of several steps, i.e. define desired state, establish current state, identify the thinnest possible slice where we can realize the change end-to-end, evangelizing the change and broadening and deepening the level of awareness and following this process iteratively. We initially focus on a small core group for implementing the change and in parallel we create awareness for the larger circle around the core group that needs to be aware of what we're trying to do so they don't block the change. Or at least don't block the change unintentionally. Below each step is briefly described.

Define desired state

As Alice experiences in Wonderland when talking to the Cheshire cat, if you don't know where you're going, any road is fine. Digital transformation is a large, complex subject area and as an organization we need to be precise and concrete in what we mean in terms of our digital transformation. Of course, as the future is uncertain, it is hard to be precise, but it is better to accept the unknowns and make a precise best effort definition of desired state than to fall in the trap of analysis paralysis.

In my collaboration with companies, this first step requires at least a one-day workshop and frequently multiple workshops. As we will be reiterating this workshop periodically, it is important to reach the necessary level of detail, but not more than that.

Establish current state

Once you know where to go, the next step is to determine where you are. Interestingly, there often is surprisingly little agreement on current state within the company and different individuals have widely differing opinions. Although we need to establish a qualitatively described current state, it can be very helpful to use surveys, data analysis and other techniques to establish the current state quantitatively as it provides a common ground as well as a basis for measuring progress.

Thinnest possible end-to-end change slice

As the saying goes, talking only gets you so far. At some point, we need to actually start doing what we have been talking about all this time. Successful organizations have a penchant for action and the best way to test whether the path toward the desired state is the right one is to try things out.

Based on dozens upon dozens of engagements, the best approach is to find the thinnest possible slice that realizes a change in an end-to-end capacity. What this looks like depends quite a lot on the specific company, but an approach where the team works on business, architecture, process and customer engagement questions leads to ensure that the holistic perspective is maintained and helps us avoid local optimization that will not scale to the entire company.

Evangelizing the change

Change management is, in many ways, a sales process and this requires that (senior) leaders in the organization explicitly present and discuss the results for change initiatives in an effort to convince the next circle of individuals in the organization to initiate a chance initiative as a next step in the process.

Repeat for next slice

In many ways, we are looking to start with a small circle of change agents and a wider circle of people aware of the change around it. Over time, we then seek to broaden the scope of the inner circle until everyone in the organization has been included. This means that once the first initiative or initiatives are well underway, we explore the possibilities of kicking off the next end-to-end slice of change.

As a final word of caution, the above process works well if there at least is some basic willingness and curiosity concerning the intended change in response to the digital transformation. If the organization is in deadlock and it is impossible to drive progress, often the only way forward is to form a new organization next to the existing one that operates according to the digital business operating system principles. Individuals can be moved from the old organization to the new one, but not without an invasive “sheep dipping” process to ensure that the old viewpoints, beliefs, norms and values are left behind and the new belief system adopted.

Feel free to reach out to me in case you are interested in exploring this for your own organization. My professional goal in life is to help companies accelerate their pace of change and I would love to have the chance to work with you.

What's Next?

As Yoggi Berra famously said, prediction is hard, especially about the future, so I will refrain from doing so. Having said that, there are some overall trends that are hard to ignore. First, we are in a transformation where software-intensive systems transition from being reaction-based, i.e. responding to actions from the users, to being proactive and taking actions that likely cause humans to be reacting. For instance, an Uber driver works for the software algorithm, not the other way around.

A second consequence of the above is that we see software-intensive systems transition from supporting humans in doing their job to replacing humans altogether and fully automating fast swaths of processes in organizations. We see this in manufacturing where the most advanced factories basically operate in the dark because there are no humans around.

Third, to bring this future to bear, it is clear that we need a level of intelligence in our software that goes beyond the traditionally programmed solutions. Consequently, it is to be expected that the progress that artificial intelligence, machine learning and deep learning have made over the last decade will continue to accelerate for the foreseeable future.

Fourth, it is likely that all systems will become self-optimizing in that these systems constantly measure their level of performance, experiment in various ways to improve this performance and then institutionalize the new way of doing things once a better way has been found.

Finally, although the above may seem scary and worrying to some, it is a logical progression from the fabulous progress that mankind has made over the last century. Technology has improved the state of mankind in all measurable ways, including poverty, child mortality, life expectancy, war deaths and discrimination against women and minorities. Although every technology casts a shadow where the negative consequences of a new technology develop, we have so far been able to harvest the good and manage the bad. There is no reason to assume that this will not continue during this century as well.

Further reading

The topic of digitalization is so extensively discussed in the media that I decided to refrain from recommending work by others. Most work focuses on only a slice of the digital transformation challenge and tends to results in local optimization that I do not find helpful. This book is an attempt to paint a broader, more integrated picture of the digital transformation and to offer a logical, objective framework for leaders to operate within.

In the references below, you'll find some earlier writing by me as well as the link to my blog that is updated weekly with new content.

References

[Bosch 17a] Bosch, Jan. *Speed, data, and ecosystems: Excelling in a software-driven world*. CRC Press, 2017.
[Bosch 17b] Bosch, Jan. *Using data to build better products*. Amazon, 2017. https://www.amazon.com/gp/product/1541210808/
[Bosch 18] Bosch, Jan. Impactful Software. Amazon, 2018. https://www.amazon.com/Impactful-Software-Business-driven-Refactoring-Ecosystems-ebook/
[Bosch] www.janbosch.com