Product Development: 10 Fallacies

Jan Bosch

TABLE OF CONTENTS

PRODUCT DEVELOPMENT: 10 FALLACIES

During the last months, several companies connected with me to help them with their product development process. Typically, these are embedded-systems companies with strong stage gate processes dictated by the challenges associated with mechanics and electronics. Their challenges are typically associated with procurement and manufacturing.

The reason that these companies reach out is the increasing awareness that the current ways of working aren't resulting in the business success they used to enjoy. Either the growth isn't there, the margins are under pressure or the market share is shrinking. It becomes obvious that what made them successful in the past is no longer working and they need to find new forms of differentiation. As the saying goes, what got us here won't get us there.

An interesting observation I've made is that those working with mechanics and electronics are of course aware of digitalization as a concept but, similar to the business folks, often fail to recognize that this affects them to a much larger extent than many appreciate. Digitalization fundamentally changes the way we develop and evolve products, leading to a host of fallacies around product development.

Many of the beliefs held by people in these companies may have been true in the past. With the digital transformation, however, these beliefs have become invalid or at least less important. The hard part is that it still is entirely feasible to build products the old way. It's just less and less aligned with the expectations in the market and the cost associated with product development tends to be higher than when using more modern means. I've tried to capture some of the fallacies, each of which I aim to discuss in more detail in the upcoming posts.

1. Product management knows what customers want

Many in R&D are focused on the specification of a new product or a new feature in an existing product. The assumption is that it's the job of product management to interact with the market and customers and to distill these insights into a specification that's the optimal content. Reality shows that in practice, this is incorrect at multiple levels. First, it's based on a generalization of the verbal input received by product managers. Second, it's based on what customers say, rather than on what they actually do.

2. Manufacturing is the hardest challenge

Setting up a factory to build the product under development can be extremely expensive and everything in R&D has traditionally focused on optimizing the manufacturing process, including procurement. Modern practices allow for much more flexibility when manufacturing a portfolio of products, reducing the associated risks. At the same time, building a differentiating product that customers actually want to buy is a much harder challenge in a competitive market.

3. The start of production is the end of R&D

One of the key differences between traditional and digital companies is their view on the start of production (SOP). Traditional companies view SOP as the end of R&D as manufacturing of the product commences and we can move on to other projects. Digital companies view SOP as the real start of R&D as we now have a feedback loop with the customer and deployed systems that we can use to inform our decisions concerning the evolution of the system.

4. Innovation means the latest technology

Many product companies equate innovation with employing the latest technology in their products. The implicit assumption seems to be that if we just have the latest technology, the customers will flock to our products. Of course, innovation is about meeting unidentified customer needs or meeting known ones much better than with existing offerings. Technology is a means to that end, not a goal in itself. There are many more dimensions of innovation, including business model, channel and ecosystem, that have much more impact.

5. Software development is a factory cranking out features

One of the ways I gauge the understanding of digitalization in companies I work with is how they talk about software development. Those that use the "software factory" metaphor have real misconceptions about the role of software development and the creative design activity it really is.

6. Experimentation has no role in product development

The use of minimal viable products and A/B experimentation is typically non-existent and discouraged in traditional companies. The specification is used as a basis for all development activity and there's no reason to question it. This brings us to the notion of knowable versus unknowable. Some things are simply unknowable until we try them out. The response of customers to new products or new features is one of these.

7. Data is only relevant for quality assurance

All companies collect data from products in the field, but traditional companies tend to only collect defect and diagnostic data. Very few of the companies I work with can, for example, answer questions concerning feature usage and the gap between product management predictions about the impact of new features and the actual outcome.

8. The bill of materials has the highest priority

R&D in traditional companies tends to have a strong focus on the bill of materials (BOM). Anything that allows for a BOM reduction is often viewed as justified, even if it means introducing strong dependencies between different subsystems and a significant deviation from the common platform architecture of the product portfolio.

9. Products are static

The focus on the BOM is justified because traditional R&D views products as static. In practice, products evolve continuously in response to, among other things, cost-down initiatives in mechanics,

obsolescence and lack of availability of electronic components and new software features deployed. A stronger focus on preparing for evolution can significantly reduce the cost of evolving a product (or platform) over time.

10. The customer cares about the product

Although there are of course products where customers really care about the physical item itself, they're predominantly concerned with what it allows them to do. For instance, most employees at car companies are deeply passionate about cars, but most of their customers have a mobility need that just happens to be best met by owning a car. And that car should be acceptable from a brand and luxury perspective in the area where the customer happens to live.

Product development in embedded-system companies is often subject to quite some fallacies and shadow beliefs. All of these were or might have been true at some point in the past but are, by and large, no longer true. Product development practices need to evolve for companies to stay competitive. In the end, it's not about mechanics, electronics, software, manufacturing or procurement but about meeting customer needs in a superior way so that we can capture part of the value we provide for them.

PD FALLACY #1: PRODUCT MANAGEMENT KNOWS WHAT CUSTOMERS WANT

As human beings, we all have an innate need for security and safety. Much of the design of modern society is driven by this. We lock up criminals, especially violent ones, punish reckless behavior that might hurt others, design our environment to minimize our risk of getting hurt and enforce rules and regulations on products to ensure that these are safe.

In companies, we see the same behavior albeit expressed differently. In general, companies are organized into functions or departments. Each has its responsibility in the end-to-end value delivery process to customers. We have boundary objects on the interface between these departments such as customer orders, requirements specifications and budget allocations.

People don't try to do the job of another department as that would intrude on their territory. Also, we don't criticize the input we get through these boundary objects, at least not publicly, as it would denigrate the perceived competence of those providing the input. We don't do these things as it would be very easy for others in the company to do the same to us, leading to a vicious cycle that doesn't end well for anyone involved.

When the input we get from others doesn't answer all the questions we might have, we can of course go and ask them, but in practice, that's time consuming and tends to reflect badly on us if we do it too often. So, instead, we do what every engineer does: we fill in the blanks based on our assumptions about what should have been there. advertorial

Many in R&D are focused on the specification of a new product or a new feature in an existing product. The assumption is that it's the job of product management to interact with the market and customers and to distill these insights into a specification that's the optimal content. Reality shows that in practice,

this is incorrect at multiple levels. First, it's based on a generalization of the verbal input received by product managers. Second, it's based on what customers say rather than on what they do.

The conclusion is that product management very often guesses the priority of the highest-impact activities and initiatives. Similar to how engineers often make design decisions based on their best understanding and experience, rather than driven by data. This isn't because people are stupid or inexperienced, but simply because certain things are unknowable. There's simply no way to predict the impact of new functionality or features on customer behavior and the market at large. The only way to find out is to experiment.

In my experience, three principles help address this fallacy: treating requirements as hypotheses, quantification of expected outcomes and continuous deployment. First, many in R&D tend to treat requirements as cast in stone and written in blood: immutable, unquestionable and meeting the requirements is the only thing that counts. In practice, a requirement is nothing but a hypothesis about what might add value to customers. Treating a requirement as a hypothesis and then finding smart, cost-effective ways to validate the hypothesis by iteratively adding confidence is a much more productive approach.

Second, in some of the companies I work with, R&D teams have now learned to immediately ask product managers who come with a feature request what the quantitative, measurable outcome of that feature is expected to be – how system behavior or customer behavior is expected to change in response to the feature. This shifts the nature of the conversation from the requirement to the intended outcome, which then allows for a much more free and open discussion around how to best realize that outcome.

Third, we're looking to minimize investment in new functionality until it has proven itself. The best way to do this is by iteratively building slices of the functionality, releasing each slice and measuring the effect. This of course requires the continuous deployment of software to customers, but also instrumentation so that we can baseline and measure the effect of each slice. In an earlier post, I discussed the Hypex model in more detail, which is a good way to realize this.

Many in R&D tend to use the requirements specification as an absolute, rather than as a list of hypotheses concerning functionality that might add value to customers. This is because the specification is typically used as the boundary object between product management and development, and these boundary objects are typically not questioned. This leads to low effectiveness of R&D as research shows that many features aren't used in practice. Instead, we need to treat each requirement as a hypothesis, quantify the intended effect or outcome and then iteratively develop the requirement to gather evidence that the hypothesis is valid. And, of course, we should kill the development of a feature when the data shows that there's no effect. To quote Peter Drucker, efficiency is doing things right, effectiveness is doing the right things. R&D has traditionally focused on efficiency, but in a digital world, it needs to focus on effectiveness instead.

PD FALLACY #2: MANUFACTURING IS THE HARDEST CHALLENGE

Few things are more satisfying than putting your hands on a new product that you plan to use for the foreseeable future. Whether it's your new mobile phone, a computer, a smart TV or a car, the fact is that we're physical beings in a physical world and we like physical things.

Rationally, we may realize that having access to products as a service may be more economical, and it for sure is more convenient, but western culture is about ownership. Just reflect on the number of mechanisms we have in place to keep track of who owns what. Banks are keeping track of who owns what amounts of money and investments. Government institutions keep track of who owns the title to various pieces of real estate, who owns what vehicles and who owns what animals. These records form the basis for an enormous network of legislation to manage ownership, transfer of ownership and violation of ownership rights.

Manufacturing has, for the longest time, focused on minimizing production and bill of material (BOM) costs. One of the main ways of achieving this is scaling: making as many items per time unit for as long as possible while avoiding changing the manufacturing line. This focus on scaling has fundamentally affected product development in at least two ways.

First, in most cases that I'm aware of, the cost of putting up a manufacturing line far exceeds the R&D expenses for a product. That means that R&D should focus on delivering a product for manufacturing that will not require any changes or defect fixes after the start of production (SOP). So, R&D is about quality assurance and avoiding unnecessary risks. This leads to a start from a frozen requirement specification and to resist any request for changes in the specification once R&D has started. Also, the closer we get to SOP, the more careful we are in updating the product design. All prototyping tools such as 3D printing for physical validation and simulation tools for determining product characteristics are

concerned with ensuring that the product has as high quality, as in meeting the specification, as we can make it.

Second, R&D has traditionally focused a lot on minimizing the BOM of the product, even at the expense of higher R&D costs. In general, the principle is that any R&D effort that results in a BOM reduction, however small, is worth it as it multiplies over the number of product instances we hope to manufacture. As products started to include electronics and software, this resulted in architectures with many dependencies in the electronic and software components purely to reduce the BOM. The implications for quality assurance of the software were viewed as R&D expenses that were compensated by the reduced BOM.

In a digitalizing world, we see two important shifts. First, products are increasingly intended to evolve through regular software updates. It should be possible to have them fully autonomously update their software with minimal risk of system failure.

This has two important implications. First, the architecture of our products needs to be much more modular and decoupled than earlier as quality assurance of a highly interconnected architecture is expensive and we can't afford to do it all the time. Second, we need 'headroom' in the electronics for software updates so that we can deliver these software updates without running out of space. Of course, both of these implications violate the traditional BOM focus. Instead, we need to focus on lifetime value rather than only BOM and manufacturing costs.

The second shift concerns the general pace of evolution. The desire to manufacture the same product as long as possible is replaced with a constant evolution of the product. This constant evolution isn't just concerned with software but also with electronics and mechanics. Many companies already have cost-down initiatives to replace components in products with lower-cost variants. More recently, though, the lack of availability and accelerated obsolescence of electronic components has caused companies to redesign the electronics in the products continuously. Although these examples focus on lower costs for the same product, the next step is of course to deliver new value through redesigns.

The implication is that many product-centric companies need to move toward more of a platform focus where each product is a configuration of the assets available in the platform architecture. In a platform, there's a platform architecture with variation points where product architectures may vary. For each component in the platform architecture, we have multiple variants if there's differentiation that can be provided through this. For instance, many companies use two or three variants where we have a high-end, mid-range and low-end variant of components.

New versions of components may then be introduced as the new high-end variant, demoting the former high-end variant to mid-range and the former mid-range variant to low-end. The former low-end variant is phased out and removed from the inventory. A platform-centric approach, in this way, allows for a constant evolution of products in the portfolio.

The implication for manufacturing is that, while still important, it becomes less central in the process as each product is a configuration of a constantly evolving platform. Research has long focused on smart, fast and low-cost reconfiguration of manufacturing lines with the intent of cost-effectively producing smaller runs.

Traditionally, the main focus of R&D was to optimize manufacturing and bill of material costs. However, digitalization and the expected continuous evolution of products require a shift in focus to modularization and platformization as well as a focus on cost-effective manufacturing to a dynamic product portfolio. Manufacturing is still important, but other aspects have gained importance and these new priorities need to be incorporated at the expense of manufacturing and BOM costs. The focus is on the value that we can deliver throughout the economic life of products.

PD FALLACY #3: THE START OF PRODUCTION IS THE END OF R&D

In the previous post, we focused on the role of platforms to accomplish a constant evolution of the products we manufacture. As such, we accept that the traditional premise of manufacturing – producing as many copies of the same item as possible for as long as possible – is no longer a viable way forward.

However, there's an additional dimension: the evolution of individual product instances over time. One of the key differences between traditional and digital companies is their view on the start of production (SOP). Traditional companies view SOP as the end of R&D since manufacturing of the product starts and we can move on to other projects. Digital companies view SOP as the real start of R&D because we now have a feedback loop with the customer and deployed systems that we can use to inform our decisions concerning system evolution.

The key challenge many companies run into is the discussion on what needs to be present in the product or offering when it leaves the factory. The traditional view is that the product should have as many relevant and differentiating features as possible since many of these will commoditize over time. This means that from the point production starts, the product gets a little less relevant every day as competitors catch up, customers become less interested and markets shift. Consequently, to extend the lifetime of the product as much as possible, we need to push as much differentiation into it as we can during the R&D stage.

The view of digital companies is quite different: at SOP, the product should have all the necessary enablers in place as electronics and mechanics are often expensive to replace. However, in terms of functionality, we're primarily interested in the minimal requirements customers need the day they get the product. From that point on, we can continuously deploy new functionality and features into the systems in the field to get to a continuously improving customer experience.

The challenge with the traditional approach is that we need to guess what will add value to customers. So, pushing for a large number of features without evidence that these indeed add the expected value easily ends up in featuritis and a very complex product. Complexity in products isn't a problem in itself, especially for expert users, but only if all the functionality captured in the complexity is indeed used by the users. A complex product with hundreds or thousands of features where each user only uses a small fraction is likely to fail due to lack of usability.

Digital companies build the known, must-have features before SOP and then use the connected nature of modern products to iteratively add functionality. This lets us measure the actual impact of a slice of a feature and direct our R&D efforts toward the functionality and features that add customer value. It also allows us to not build features that fail to add value as well as remove features that aren't used at all or only by a few customers.

Interestingly, the simple ability to measure feature usage is already incredibly helpful as it directs our R&D effort and avoids all the difficult opinion-based discussions in companies about features. Many people in companies have quite diverse opinions on what really matters to customers and using data instead of the opinions of people who may not even have met a customer during the last year or so provides a significant improvement in the effectiveness of R&D.

The main advantage of connected products is the feedback loop that allows us to experiment with new functionality, such as A/B testing. This facilitates ensuring that we only build features that are really used, but it also helps improve already implemented features to maximize the value delivered through each feature. Rather than adding feature after feature, in many industries, optimizing the features already present actually gives a higher return on investment.

We should view the start of production as the start of a continuous feedback cycle between us and our customers. This allows us to optimize the system functionality to minimize complexity, only add features that deliver value, optimize the way features are realized and remove features that aren't or hardly used. Life starts at SOP, it doesn't end there!

PD FALLACY #4: INNOVATION MEANS LATEST TECHNOLOGY

Few words in industry have such an overloaded meaning as the term “innovation.” Innovation and all its derivatives, like being innovative, in many contexts simply mean “good.” The underlying idea is that we like new and different, and an innovation will provide that.

In product development, innovation almost always refers to introducing a new technology in the product. This might be the next generation of a SoC (system on chip), a new material, a different user interface, and so on. The general belief is that if we simply keep our products up to the latest that our suppliers can provide and we can afford to develop, they’ll be competitive and consequently profitable.

The challenge with the focus on adding the latest technology to our products is threefold. First, in virtually all industries, everyone gets access to new technology at about the same time. All companies get going on adopting this new technology at the same time and proudly present the next generation of their offerings using this new technology at the same time. There’s no differentiation provided by doing this. Of course, it might be necessary to bring up our offering to use the latest technology, but it simply is sustaining innovation.

Second, the focus on including the latest technology causes organizations to ignore all other types of innovation. In addition to improving the product, we can also focus on the entire product system, the way we bring the product to market, our position in the ecosystem, the business model we use, and so on. In fact, research shows that non-product-centric innovations have a much higher impact than product-centric ones. The big breakthrough businesses of the last decades weren’t technology driven. For instance, Uber and Lyft still use cars but fundamentally change the way people get mobility.

Spotify and Netflix didn't come up with fundamental technologies but reinvented the way we consume music and movies.

Third, the product technology focus causes us to typically veer away from solving the customer's problem and instead focus on improving the product for the product's sake. For many in product development, it's hard to believe that customers really don't care about the product and might even consider it a burden to have to buy it. The reason they do so anyway is that the product solves a problem or "does a job" they need done. The moment, however, that a better way to get the job done comes along, they'll prefer that. So, we need to keep our eyes on what's important, ie the job for which our product "is hired" to talk – in Clayton Christensen's words.

It seems that many people in product development, typically being technology focused, tend to mix up the notions of invention and innovation. In my view, innovation is best described as invention + monetization. Not getting paid for adding new technology that's added to the product isn't innovation but "just development" and maybe an invention.

As all companies have only a limited budget for product development and innovation, the goal has to be to maximize the return on investment. In my experience, the best way to accomplish this is by developing empathy with customers and ensuring we have a deep understanding of their needs. Not just the role of our products in their operations but rather in their entire workflow, their support functions and their customer interactions.

Customer empathy and domain understanding can then be used to develop hypotheses about innovations of any type, including business, process, product system, channel and brand, that might deliver value to customers and that we can capture part of the value of. The resulting hypotheses should be tested in experiments that iteratively provide more and more positive evidence. Those that don't pan out should be excluded. The hypothesis generation and experiment-driven hypothesis validation process thus gives a constant flow of proven innovations that can be prioritized for new product development and scaling.

To many, this seems inefficient as we're testing many hypotheses that won't result in any new products or innovations. But what's the alternative? Putting all our energy into one or a small number of big bets that may or may not work? To use the terminology of Jim Collins: you first shoot bullets before you shoot cannon balls.

In product development, innovation is almost always equated with new technology to be incorporated into the product. Although perhaps necessary from a sustaining innovation perspective, it won't provide differentiation that moves the needle for the company from a business perspective. Instead, we have to view innovation as invention + monetization and broaden our perspective on what constitutes innovation to include the product system, the channel and brand, the user experience, our position in the business ecosystem as well as other factors that contribute to business success. That requires deep customer understanding, hypotheses and experimentation before scaling any promising innovations. As management guru Peter Drucker said: "If you want something new, you have to stop doing something old."

PD FALLACY #5: SOFTWARE DEVELOPMENT IS A FACTORY CRANKING OUT FEATURES

One of the popular metaphors for software development that annoys me to no end is the notion of a software factory. Even if the underlying concept is concerned with well-defined processes and structured ways of working, the image it draws up in my mind is that of a production line. And it pictures software engineers as mindless factory workers that repeat the same task over and over again.

Interestingly, mechanical design and electronics design are never described as a factory. Everybody understands that a mechanical or electronics engineer is designing a solution and verifying all its necessary properties. Once the design is finalized, we can start manufacturing. And everyone understands that these engineers need to evaluate alternatives, create prototypes, test these prototypes and then arrive at the preferred solution after having explored the design space.

For some reason, when it comes to software, especially for folks outside of the discipline, software development is equated to the manufacturing process, rather than the design process. The implicit assumption seems to be that a requirement that needs to be realized in software doesn't require a design but is nothing more than an algorithmic translation from human text into code. I can for the life of me not understand why reasonably smart people fall into this fallacy. The only reason I can come up with is that because software doesn't have associated manufacturing, some may think that software development is manufacturing.

There are at least three concerns with treating software development as a factory. First, it fails to recognize the creative nature of software development. It really is a creative activity where the team as well as individual engineers explore a design space, based on their best understanding of the requirement, the customers and the context in which the functionality will be used. It typically requires exploring alternatives, testing things with customers and using empathy with the user to come to a realization that best serves the intended purpose.

Second, it tends to ignore the larger scope in which a feature or requirement needs to operate. The notion of feature interaction is well established in the software engineering literature and basically refers to the fact that features aren't simply lego bricks that can be put together; they almost always interact with already existing functionality. The integration of a new feature into the existing code can be quite challenging and requires careful exploration and design. The whole purpose of software architecture is to minimize feature interaction and, consequently, future maintenance and extension costs, but every architectural decomposition will have some types of functionality that can be easily added and other types that will require changes in many places in the code.

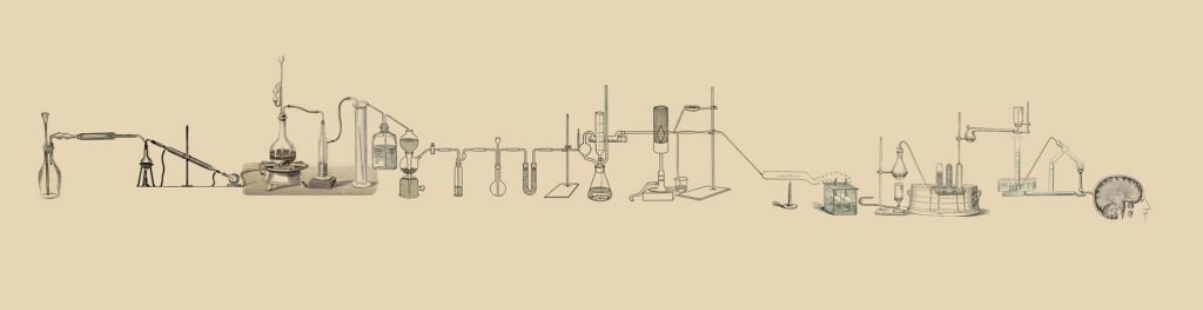
Third, it assumes that all features are worth building and that the specification is the ground truth when it comes to what adds value to customers. With digitalization creating more and more opportunities for DevOps and frequent feedback from the field, the whole notion of building a large set of complete features without frequent, intermediate testing that they actually add value is simply outdated. As we discussed in an earlier post in this series, we need to treat requirements and feature requests as hypotheses that need to be validated. And it's the job of R&D to do the validation.

One way to think about this is to distinguish between efficiency and effectiveness. Efficiency is the notion of doing work at the lowest cost in terms of resources, defects and time. Effectiveness is concerned with creating maximum output against each unit of input. Optimizing the R&D organization for cranking out as many features per time unit as possible assumes that each of these features is actually used and appreciated by customers. Research shows that this is a completely wrong assumption. In fact, less than half of the features in a system are used sufficiently frequently to justify the R&D investment.

We need to shift the focus of R&D from efficiency to effectiveness. This requires us to build a slice of a new feature, get it out there and in the hands of customers, measure its impact and decide on the right course of action after reviewing the data. And this may mean canceling a feature and removing the initial code for it from the system as it doesn't deliver the expected value.

Second, rather than focusing on building new features, we should spend more time on revising and experimenting with features that are used frequently. In many cases, we can achieve a significantly higher return on investment by optimizing existing features than by adding new features that aren't used. For most people with the software factory mindset, that's a complete blindspot.

One of the ways I gauge the understanding of digitalization in companies I work with is how they talk about software development. Those that use the "software factory" metaphor have real misconceptions about the role of software development and the creative design activity that it really is. Instead, focus on fast feedback loops with systems in the field, iteratively build new features to measure their impact and spend much more time optimizing existing features that are heavily used, rather than adding more and more features. It's impossible to not end this post with the famous quote from Peter Drucker: efficiency is doing things right; effectiveness is doing the right things. Let's focus on effectiveness first!



PD FALLACY #6: EXPERIMENTATION HAS NO ROLE IN PRODUCT DEVELOPMENT

In most of the companies I work with, the decision to develop a new product is made based on an assessment of the likely revenue and margin the product will create. This assumes a clear specification of the intended functionality of the product as a basis for the cost estimation associated with the development of the product. Typically, this means product management working hard on creating a case for the product including the differentiating functionality, the position in the market, the intended customer, etc.

Although this all makes sense in theory, in practice this approach is simplistic and fails to deliver on expectations in most situations that I am aware of. One of the main factors that I have written about in several earlier posts is the inability to make accurate long-term predictions. Many claim that their organization can, but in my experience this is mostly concerned with padding the initial estimate to get the estimate to sit at the end of the bell curve and hence the likelihood of staying within the estimate being very high. Informally, this is referred to as the rule of Pi: you ask for an honest estimate, multiply it with Pi (3.14) and use that for planning.

Despite all the best efforts, many product development efforts run over budget, both in terms of time and cost, and that is where R&D bashing easily starts. Armed with the perfect knowledge of hindsight, senior leaders outside of R&D openly wonder why the idiots in R&D are again late and why we keep allowing those monkeys to destroy the profitability of the company.

Of course, R&D gets its chance at revenge when the product is finally done and in the market and sales is not able to generate the revenue that was presented at the beginning of the development initiative. This then leads to sales complaining about lacking features and poor product development practices resulting in a sub-par product. This is where the finger pointing starts and people dig their trenches and start lobbing grenades.

To me, the root cause of all this is a fundamental misconception in the heads of most leaders: the assumption that it is actually possible to predict what a product should contain in terms of functionality in order to be successful. And that brings me to the distinction between what is knowable and what is unknowable.

As the name implies, knowledge that is knowable is that which can be uncovered simply by putting more energy into collecting data and information. When product development fails due to lack of insight into knowable aspects, we have a problem in that someone might not have done as good a job as the person should have.

When it comes to unknowable things, the only answer that we have is experimentation. We have to try it out in the hope that we learn what we need. The idea that some things can not be known before

the start of product development and that require experimentation during product development or after we have shipped the product through DevOps style practices is alien to most companies that I am aware of.

Of course, there are aspects that might be knowable if we would invest large amounts of resources, but where it is prohibitively expensive to do so. In those cases, the use of experimentation may be a much better way to collect the necessary information than to sit and guess.

The challenge is that when people are asked to provide tangible answers to unknowable questions concerning the functionality of a product, the most reasonable approach is to simply guess. The problem with guessing is that something that the individual initially understands is just a guess rapidly becomes a truth that is treated as being cast in stone. People love certainty and hate uncertainty and hence even the most "putting a stake in the ground" guesses rapidly become requirements.

One reason for the reticence toward experimentation is that it introduces uncertainty and risk. If we are unable to determine the exact functionality needed for a new product, we can't do the effort estimation. If we can't accurate predict the required effort, we don't know the expected revenue and margin. All this makes us poor leaders as we decide on product development efforts without proper financial justification.

Still, for all the difficulty of dealing with the uncertainty and risk, it doesn't change anything about the reality of product development. In my view, many of the dysfunctions in product development organizations originate in the inability of leaders to accurately distinguish between knowable and unknowable things. It takes courageous leadership to break out of this conundrum and confront reality as it is, rather than as you wish it was. Even if it is much more comfortable to pretend that you know what you don't, nothing good ever came from ignoring reality.

The best way to address this situation is to treat product development as an iterative process of risk reduction. Risk can be technological in nature, but companies already know how to deal with technology risk. The concept of technology readiness levels (TRLs) was developed as one model to deal with this. The market or customer risk, however, is typically by far more significant but we are much less well equipped to deal with this.

Iterative development breaks up a product development efforts into a series of decision points where the team is asked to clarify and deal with one or a small set of open questions concerning the product. Each iteration receives a small amount of funds, performs tasks to answer the highest priority questions and based on the data that comes back, the governance team decides whether to fund the next iteration or to stop development.

One concern that is often raised when I suggest this approach is that the effort required to even create a simple prototype for testing with customers is so prohibitively expensive that there is no alternative but the traditional product development model. Although I appreciate that this may be the case in a limited number of cases, in the majority of these, my experience is that this is more of a lack of creativity. We saw the same in software companies adopting agile practices where initially teams complained about not being able to break down features such that these fit in a single sprint.

Concluding, the use of minimal viable products and A/B experimentation is typically non-existent and discouraged in traditional companies. The specification is used as a basis for all development activity and there is no desire to question it for a host of different reasons. This brings us to the notion of knowable versus unknowable. Some things are simply unknowable until we try them out and the response of customers to new products or new features is one of these. This requires a different, more iterative, product development process where each iteration is decided upon once the data from the previous one justifies continuation. This fits hand in glove with digitalization as product development continues after the product has reached the market with the continuous deployment of new

functionality. So, in that sense, the iterative approach is a constant throughout the entire product lifecycle and the start of production is just a small blip in the overall process. As Mark Twain said, continuous improvement is better than delayed perfection.

PD FALLACY #7: DATA IS ONLY RELEVANT FOR QUALITY ASSURANCE

One of the human behaviors that never ceases to amaze me is the gap between what people say they do and what they actually do. In management research, this is often referred to as espoused theory versus theory in use.

When it comes to digitalization and working with data, I see a similar pattern in many companies I work with. On the one hand, many consider data to be incredibly important and they don't want anyone else to have access to it. As a consequence, companies are very restrictive about making their data available to suppliers and partners. When asked what they're doing with the data, people often hide behind arguments concerning confidentiality and secrecy.

In practice, on the other hand, many companies aren't using the vast amounts of data they collect at all. The only real use case is quality assurance diagnostics. Recording defects and the system context during which the defect occurred as a means to simplify defect removal is the main use of the data coming back from the field.

Very few of the companies I work with can, for example, answer questions concerning feature usage and the gap between product management predictions about the impact of new features and the actual outcome. This is in many ways surprising as everyone agrees that adding features that aren't used is a significant source of both wasted effort and increased system complexity that reduces productivity when adding or changing other functionality.

We've developed a model capturing four dimensions of evolution when going through a digital transformation. These include the business model, product upgrade and AI. Here, we focus on the data exploitation dimension.

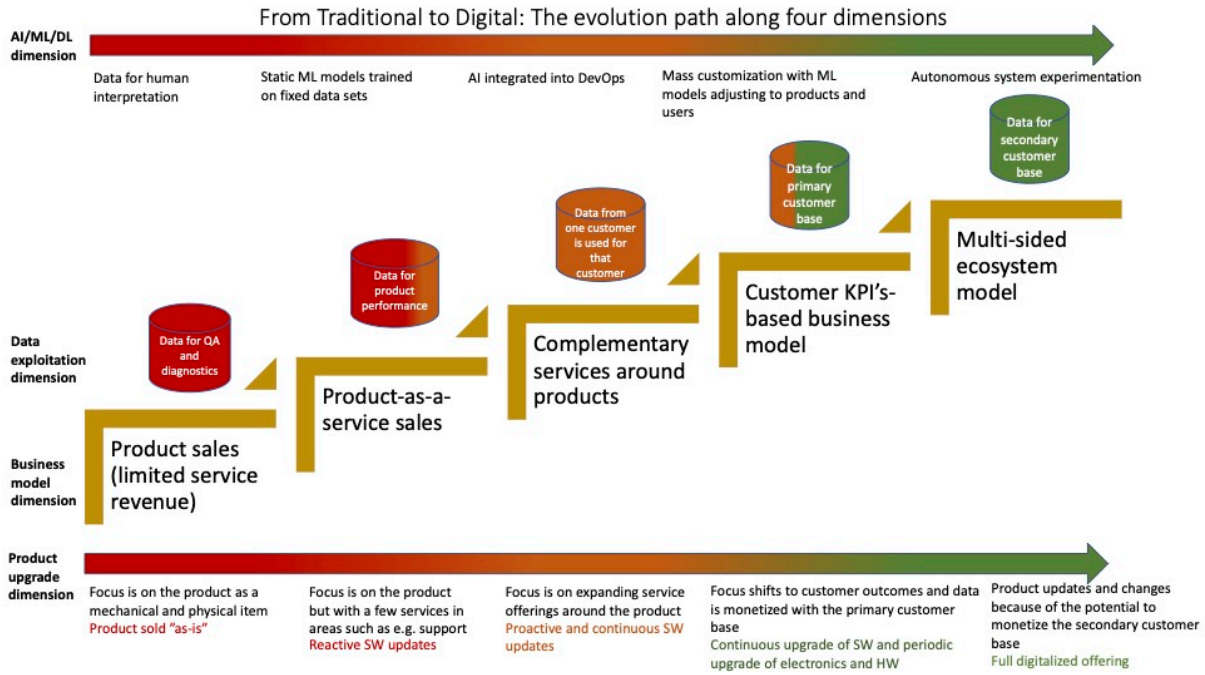


Figure: Four dimensions of a digital transformation

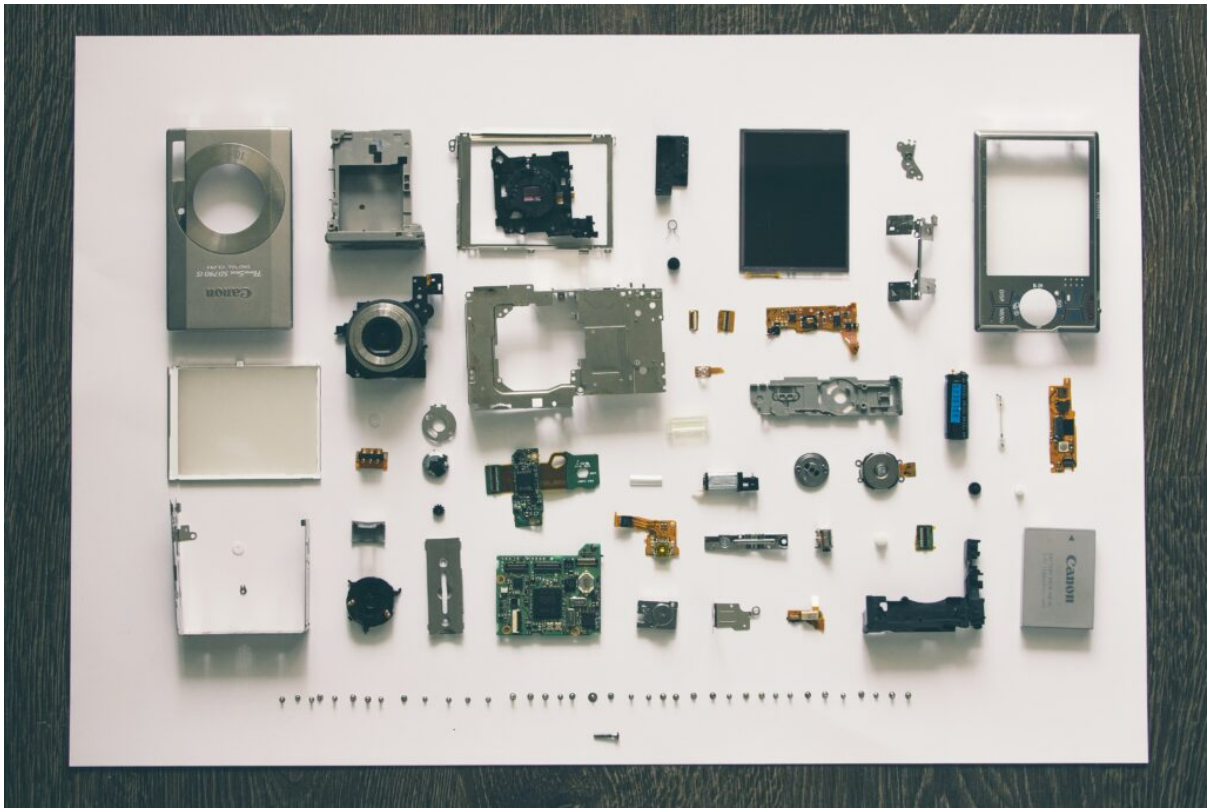
The first step is using data for quality assurance diagnostics, which is the state of many companies. The second step is concerned with product performance. Here, we collect data on feature usage and in general seek to align our R&D efforts with the quantitative outcomes we're looking to achieve for our customers.

The third step is using the data we collect from systems at one customer for that particular customer. The idea is that the data we collect and analyze allows the customer to improve the performance of the business and understand where the challenges in the end-to-end workflow are. Providing a mirror for the business of the customer can be extremely valuable and is something customers often are willing to pay for.

The fourth step is comparative analysis. Here, we use data from all customers to provide insights to individual customers. Often, this data can be used to provide insight into the performance of the customer in comparison to similar businesses. These comparisons can be concerned with cost drivers, eg average salary of employees or operating hours of machinery, but also profit drivers, such as average revenue per machine or employee, and so on.

Finally, a highly controversial topic in many companies is to use the data from the primary customer base and monetize this with a secondary customer base. Typical customers of this type of data are hedge funds, which are always looking for ways to create an edge for themselves, but they can also be stakeholders on the edge of the current business. For instance, most trucks have GPS and accelerometers installed, allowing the company providing these trucks to collect road quality data on all the roads where the trucks are active. The question then becomes who is interested in buying this data.

Most companies I work with use data predominantly for defects and quality assurance and fail to exploit the full potential of the data they collect. Once we have frequent data coming back from the field, we can use it to improve system performance, provide insights to our customers and serve additional stakeholder groups. Data is the new oil, Clive Humby wrote in 2017, but if you don't exploit it, you end up with a bunch of useless storage. Don't just trust your opinions, but rather trust the data.



PD FALLACY #8: THE BILL OF MATERIALS HAS THE HIGHEST PRIORITY

Traditional companies building products including mechanics, electronics and software tend to focus on the bill of materials (BOM), standard unit cost (SUC) or some other KPI tracking the per-product instance cost during manufacturing. When manufacturing large numbers of the same product for extended periods, this makes perfect sense as every penny saved is multiplied over all the manufactured instances.

The challenge arises when companies are undergoing a digital transformation of their product portfolio and aim to deliver value to customers continuously. A typical model is to adopt DevOps as a mechanism to push new software to products in the field frequently. The moment the company transitions to this model, the BOM focus of the mechanical and electronics engineers becomes a significant hurdle as their decisions tend to lead to designs with very little ‘headroom’ in terms of CPU and memory resources.

The consequence is a clash between two paradigms of product development. On the one hand, it obviously is a good idea to minimize the cost of the product as it allows for better margins. On the other hand, not leaving headroom in the product destroys the ability to deliver value over time. Especially during the transition, it’s often very easy to quantify the short-term gains of squeezing the BOM and very hard to quantify the potential gains through continuous value delivery. This leads to companies continuing to focus on the BOM for way longer than what would make sense from a business perspective.

The BOM perspective tends to cause multiple challenges of which I want to discuss the three primary ones. The first, most obvious, is that not leaving headroom in the product destroys the ability of the

company to build a continuous value delivery model. Especially for long-lived products that have years, if not decades, of expected life span, this is a major issue as the current product generations will hamper the company's digital transformation for years to come.

The second challenge, often less well understood, is that an excessive focus on the BOM tends to introduce strong dependencies between different subsystems. Engineers are allowed to introduce any shortcut that results in a reduced BOM and consequently, the system will exhibit high coupling. When adopting DevOps for products, this tends to lead to an increased number of quality issues as it's hard to validate and test all the intricate dependencies in the system.

Finally, especially in platform companies, allowing for BOM optimization tends to result in a significant deviation from the common platform architecture of the product portfolio. This causes what's often referred to as a versioning hell as every product and every generation of every product needs its own specific, unique version of the software. This is acceptable if we have the build the software once, but when we adopt DevOps, it means that we have to generate new software for every configuration every two to four weeks. This gets expensive and labor intensive really fast.

To address these challenges, one of the key actions required is to ensure that you monetize the continuous value delivery through DevOps in some way. If there's no revenue associated with new functionality being delivered to customers, the entire organization will drag its feet as it will only add cost at the promise of some vague 'customer preference' promise.

Of course, we can monetize through maintenance or subscription fees, but an effective approach can be to use the outcome metrics your customer is concerned with as a basis. Assume your system generates 100 units per hour for the customer and you can increase the output to 110 units per hour by deploying new software. In that case, it's entirely reasonable for the customer to share some of the additional revenue with you. This requires, however, a clear understanding of what the key value factors are that your customer cares about. This understanding is surprisingly limited in most of the companies I work with as they focus on the product itself.

The second key action is to focus on bringing your entire offering portfolio into a single platform and make each product a configuration of that single platform. This allows for a vastly improved return on investment for most R&D efforts as much of the newly developed functionality can be shared among all products in the portfolio. Test infrastructure, deployment infrastructure, data collection infrastructure, and so on, can all be shared as well.

Although I've covered this topic before, I still run into people and cases that make me realize that the message hasn't been received everywhere. An excessive focus on the bill of materials leads to significant challenges for companies that are undergoing a digital transformation and adopt continuous value delivery. The lack of headroom, high coupling and versioning hell may easily cause an explosion of R&D expenditure over time. Instead, ensure to monetize continuous value delivery and platformize your entire product portfolio to address these concerns. From a focus on the bill of materials, we need to shift to lifetime value. In the end, we want a continuous relationship with our customers and this is one of the best ways to strengthen that.

PD FALLACY #9: PRODUCTS ARE STATIC

Humans love to think in terms of absolutes. We like to set a target, work really hard toward it and have the illusion that when we reach it all our troubles will be gone and we'll be in a state of permanent bliss. For instance, sports competitions have a very clear target and those of us who have run marathons have all experienced the suffering (especially between 27K and 32K in my case) and the longing for the finish line.

Similarly, many contracts between customers and suppliers tend to be described in targets and goals to be reached. In construction, erecting and completing the building is the typical goal of the contract. In the defense industry, companies often manage a list of thousands of items required by the government purchasing from them and the contract stipulates that the project is complete when all these requirements have been satisfied.

Product development in many companies is organized similarly. A list of requirements is specified, a cost estimation and a revenue prediction are performed and an investment decision is taken to develop the product. The basic notion is that everything is now frozen and cast in stone and it's just a matter of executing properly.

When describing the process as I just did, it's obvious to everyone that this viewpoint is a fallacy. Life doesn't stop after crossing the finish line of a marathon (at least mine didn't). Buildings still need to be maintained and evolved after initial delivery. Product requirements tend to change at a rate of 1 percent per month. The world isn't static but evolves continuously and so do we and the products we're involved in building.

The desire to establish concrete and tangible "anchoring points" is because people seek to avoid having to deal with continuous change. Creating the illusion of an immutable target gives a sense of certainty that allows the alignment of large groups of people without having to deal with the constant overhead of perpetual change.

This approach may have worked in the traditional transactional world where people would buy a product, use it up and buy the next one. In a digital world, however, customers expect continuous value delivery. I expect the product that I got from you to get better every day I use it and I don't want to wait until I have to buy the next generation.

In practice, it isn't just customers who desire continuous improvements. Products evolve continuously in response to a variety of causes. Cost-down initiatives in mechanics are a typical example in many embedded-systems companies. Obsolescence and lack of availability of electronic components is a key challenge in many companies, requiring updates to products. And product management and sales also demand a continuous flow of new software features to drive engagement.

Rather than a focus on a one-time, far-in-the-future target, the view that many in product development would benefit from is one of continuous evolution. This view has several benefits. The first is that continuous evolution allows for less risk in each iteration as not everything that product management or sales can think of needs to be crammed into this release. We can take the releases step by step and manage the risk and new content with each generation.

Second, a stronger focus on preparing for evolution can significantly reduce the cost of evolving a product (or platform) over time. When focusing on the one-time target, we tend to make many decisions that sacrifice the long term over the short term. When we know we'll be continuously evolving the product, we tend to take design decisions that balance short and long-term consequences more optimally.

Third, it allows us to move away from a project mindset that many companies experience where a project with limited resources, time and budget needs to hit a particular goal at some point in the (distant) future, and rather focus on a product or platform mindset where we invest continuously but carefully track the value generated by these investments.

Like everything else in life, products also aren't static. However, many companies aim to create an illusion of stability by stating immutable targets to be pursued by product development. In practice, requirements change continuously, our customers expect continuous improvements, suppliers require us to update to the next generation of parts and economic realities force us to engage in cost-down initiatives. Rather than pretending that everything is static and grudgingly accepting the inevitable changes, we're much better off accepting that change is continuous and working in short iterations on a continuously improving product. As Winston Churchill said: "To improve is to change. To be perfect is to change often."

PD FALLACY #10: THE CUSTOMER CARES ABOUT THE PRODUCT

One of my favorite activities when meeting with the companies I work with is to tell them that their customer doesn't give a flying hoot about their product. This often leads to a storm of protests and objections and meeting participants proudly showing me net promoter scores and quotes from customers where they express their undying love for the company and its products. My typical response is to provide my view on the inconvenience of owning one of their products.

For instance, a truck is an absolute hassle to own: it's expensive to buy, uses a lot of expensive fuel and tires, requires fleet owners to pay drivers, occasionally breaks down, requires a ton of space to park, and so on. The only reason someone in their right mind would own a truck, or any other product, is because of what they can do with it. Companies own trucks to move goods, telecom networks to offer communication services and medical imaging products to get images from inside human bodies.

Although there are of course products where the customer really cares about the physical item itself, in most cases, they're predominantly concerned with what the product allows them to do. For instance, most employees at car companies are deeply passionate about cars, but most of their customers have a mobility need that just happens to be best met by owning a car. And that car should be acceptable from a brand and luxury perspective in the area where the customer happens to live.

When companies forget about the intended use of the product, they typically experience three issues: the "tallest midget" problem, overdelivery on KPIs and missing industry disruptions. The first challenge is that businesses that are overly product focused are concerned with developing product generations where each generation is better than the previous one. This internal focus can easily lead to a situation where the company is very pleased with it delivering a better product with every generation, but it's still falling behind the competition and failing to attract customers. The idea is that the latest product is

better than all earlier products and with that, it's the tallest midget in the village, but it's still a midget. advertorial

The second challenge is caused by companies being much more focused on competition than on customers. It's easy to use the products of competitors as the measuring stick and demand that your products are better. As the metrics used for comparing with competitor products may not be what customers really care about, it's equally easy to develop products that overdeliver on KPIs that customers don't care about but that add significantly to the product cost.

The third challenge is that companies may easily miss significant shifts in customer preference and industry disruptions. History is littered with businesses that missed these disruptions and kept going on a path that eventually lead to their doom. We all know the story of Kodak, where according to some the first digital camera was developed. The city where I live is the home of Hasselblad, a photographic equipment company that's now a shadow of its former self. And, of course, I did work for Nokia at some point in the past.

A confounding factor that receives too little attention is that companies very easily end up as hostages of their business ecosystem. Even if many in the company see the disruption coming, their partners, suppliers and customers don't want them to change as it would require the rest of the ecosystem to change as well. Ensuring you initiate and drive the necessary change despite the pushback from everyone around you requires courage and isn't for the faint of heart. But it's also the responsibility of leaders to initiate change before it's obvious to everyone that it's needed.

The best way of avoiding these challenges is to think in terms coined by the late Clayton Christensen: every product is "hired" by the customer for a "job" that the customer wants to have accomplished. The customer doesn't care about the product, but rather about getting the job done. The focus of any product company needs to be to ensure that their product is the best option for getting the job done as customers will switch without a second thought the moment there's a better way. In fact, industry disruptions are the result of a better alternative becoming available to customers.

Many product companies are in love with their product and that's generally a good thing. If you don't care about your product, why would anyone else? However, failing to realize that your customers actually don't care about your product, but rather the job for which they "hire" it, leads to several challenges that may end up disrupting your business. Instead, continuously focus on the reasons your customers buy your product and ensure that you are and remain the best option customers have to accomplish their desired outcome. As Jay Abraham said: "Sell the benefit, not the product or your company. People buy results, not features."

CONCLUDING THOUGHTS