Practical Product Development Challenges

Ashruf, Colin

DOI
10.1109/EMR.2023.3269326

Publication date
2023

Document Version
Accepted author manuscript

Published in
IEEE Engineering Management Review

Practical Product Development Challenges

Colin Ashruf[1]

Keywords: success factors in product development, management of innovation, management of scientists and engineers, product innovations, project and R&D management, project success factors, implementation methodologies & project management, empowerment

Summary

Despite the many product development techniques available today, manufacturers under pressure to reduce time to market while keeping up with stricter regulatory demands are struggling more than ever with their product development processes. Here I list a selection of the most important and frequently encountered problems and challenges from my consulting practice checked with literature, as well as practical recommendations for improvement. While in specific situations some project management techniques prove better suited than others, overall product development success seems more dependent on the organization's willingness and ability to learn and adapt rather than on the specific technique chosen.

[1] About the author: Dr. Colin Ashruf has been an independent consultant for the high-tech industry for over twenty years, advising established multinational corporations as well as start-ups. He regularly writes about the business of innovation for international media and holds a PhD and MSc in Electrical Engineering from Delft University of Technology.

I Introduction

Manufacturers today have an impressive and nonetheless even growing range of product development methodologies at their disposal, ranging from modern Agile and Lean techniques (such as Scrum, SAFe, Less, Kanban, Lean Development) to older Waterfall methods that have fallen out of fashion lately (such as Prince2, V-Model, Systems Engineering) [1]. Despite this abundance, however, organizations seem to be struggling harder than ever with their product creation processes [2]. Partly, that is because the products themselves along with the technologies on which they are based have grown more complex over the years. Meanwhile, driven by technological advances and increased competition, the public has become more demanding, expecting new product generations at increasing pace. Companies failing to address this demand are overtaken by their peers that do a better job at constantly pushing out new products, and face impending oblivion. Further complicating the creation of new products are stricter regulatory demands, both on the products themselves as well as on the way these products are conceived [3, 4]. Products need not only be smaller, faster, nicer, and cheaper, but also better for the environment and safer to use. A good example is the field of medical devices, which need to comply to some of the strictest safety and environmental compatibility standards in use, with their associated risks of use during the entire product life cycle meticulously assessed and mitigated, and paper evidence of compliance kept on file and imminently provided if so requested [5].

Companies thus facing pressures from the market typically turn to consultants for help with finding and implementing product development processes that are faster than what was used before, while more thorough. Unfortunately, though, despite the many options available, each with its own typical advantages and drawbacks, no clearcut advise as to which technique to select is possible, as there exists none as such that warrants success in every case.

The most suitable technique depends on the specific situation at hand, the products under development, markets served, business dynamics, company history and culture, expertise available either in-house or at development partners, etc. Rather than a quick and convenient how-to instruction, adopting the right technique is a lengthy, laborious, and at

This article has been accepted for publication in IEEE Engineering Management Review. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/EMR.2023.3269326

times painful learning experience the entire organization has to go through; it is a matter of sensibly selecting a starting point and adapting it to the specific needs at hand as one goes, finding out by trial and error what works and what needs further tweaking in any particular case. In this quest for a dedicated development process there are pitfalls, however, problem areas that time and again turn up both in practice as well as in literature that the consultant can help navigate.

Section II lists a selection of problems and challenges I came across in my consultancy practice most often, cross-checked with literature and categorized into the main elements of project management (requirements and scope, team, planning and scheduling, and quality), while section III gives concrete recommendations for improvement derived from practical cases.

II Challenges & problem areas

II.1 Product requirements

II.1.1 Product vision

Product development starts with the product idea, the definition of its features and performance, which proves one of the hardest things to get right [6]. Many companies struggle with this, even those that in the past have managed to come up with one or more successful products. The main reason product definition is so hard is because it requires vision, a thorough understanding of both market demands and technical feasibility -- a combination of technical and business experience and skills that is sparse today. In many companies, successful products can be traced back to the drive and enthusiasm of only one or few visionaries [7]. Without such visionaries, because they have retired or left the company for other reasons, manufacturers tend to turn to marketing consultancies to help them understand their markets and come up with suitable product proposals. The problem with this approach, though, is that these consultancies often rely on industry consensus of what the market needs and on (often) publicly available competitive analysis. Blindly following their advice therefore implies running the risk of putting in the enormous efforts required just to come up with the next bleak copy and get snowed under in the already too crowded marketplace. A more effective approach, albeit one that takes years for it to pay off, is to take on or develop from within own ranks promising candidates that with sufficient practice and training can grow into the next product visionaries.

II.1.2 Requirements management

Once market requirements have been agreed upon and converted into viable product requirements or feature definitions, managing all these requirements, keeping track of them, seeing to it that they are all fulfilled throughout the consecutive stages of the project, proves to be another major challenge [8].

Firstly, because of the increased technological complexity the number of requirements can become quite large, easily growing into the hundreds or more. Meticulously keeping track of how these are woven into the many aspects of the final design is quite cumbersome and easily underestimated by teams new to this [9]. Coming up with a set of unambiguous requirements as well as their decomposition into subsystem-level

specifications or user stories that the respective development teams can work with is therefore typically left to system engineers, requirement managers, or product owners dedicated to this task.

Secondly, owing to the nature of today's product business with its frantic markets and rapidly evolving business landscapes, market and product requirements tend to change throughout the development process. (Often enough, though, these changes result from altered insights, which, if that happens too frequently, can point to a lack of product vision or disconnect with customer needs.)

Whichever the reason, a solid grip on requirements is needed if the impact on both the product and the project of these changes are to be quickly assessed. Using simple spreadsheets or word processors to keep track of a large set of changing requirements quickly leads to an inextricable mess and an unworkable burden on the teams. Fortunately, dedicated workflow and database software tooling allowing for orderly bookkeeping of requirements along with their verification and validation has become more mainstream in recent years [10].

A commonly encountered reflex in organizations struggling with changing requirements involves the enforcing of 'freezes' (concept freezes, design freezes, requirement freezes), leadership teams then insisting on discipline in keeping with past decisions. This rather inflexible, outdated approach, however, as proven repeatedly in practical cases, is ineffective, simply yielding products that fall short of market demands and are destined to fail [11]. Building in flexibility during development, allowing changes to be made to requirements or features along the way, sometimes even last-minute, has become all but unavoidable, and with proper tooling also quite doable.

Especially useful for organizations that have a hard time getting the requirements or features right in the early stages are techniques such as Agile and, more recently, Design Thinking that enable stepwise product increments towards market needs by multiple trial-and-validation development iterations [12, 13]. Instead of relying on the vision of one or few team members, typically from the product management department that

formally caries that responsibility but might not be properly staffed to deliver on this, these techniques elegantly tap into the innovative potential of the entire R&D team.

II.2 Scope

II.2.1 Product & project demarcation

Forgoing explicit listing of product and project scope remains one of frequently named reasons for project failure [14]. Refraining from describing project scope in detail, going over it with the entire team -- contributors as well as stakeholders -- will end up with scope potentially being interpreted in as many ways as there are team members. Absent explicit product boundaries assumptions will be made. Again, this is especially true for development projects carried out with partners and teams spread out over multiple organizations and locations, often in different parts of the world, as is the case with many product development undertakings nowadays.

Describing product boundaries in sufficient detail is key here. It is often in the details of scope demarcation that things that at a higher abstraction level still looked clearcut start to get fuzzy and questions arise. Typically, then, during these detail discussions on scope, the feasibility of the project is for the first time truly tried.

II.2.2 Interface management

After the product demarcation and the set of external interfaces have been clarified, the product will usually be broken down into multiple components or subsystems, each with its own interfaces to the rest of the system (internal interfaces) or outside world (external interfaces), which the development teams can then work out in detail concurrently.

The external interfaces[2], which are after all clearly visible on the outside of the product and define it, usually pose no major issues. It is at boundary of the components or subsystems, the internal interfaces, where things most often go wrong and special

[2] The external interfaces of an electronics device, for example, typically include a mechanical enclosure, a human-machine interface made up of a touchscreen and mechanical buttons, and communication modules to enable wired or wireless communication.

attention is needed, more specifically at the non-physical interfaces in software as well as at the indirect interfaces[3] [15].

Software interfaces can be effectively managed using API's (Application Programming Interfaces) that enable components to communicate with each other using a set of pre-defined parameters and protocols [16]. Indirect interfaces tend to be problematic because they are notoriously easy to miss and might even be entirely unknown, especially to organizations new to the product under development or technologies used [17].

As with the external product interfaces, internal interfaces need to be considered in sufficient detail and explicitly described, and any changes therein carefully managed throughout the project to ensure the final product meeting its requirements. Again, it helps to assign these responsibilities to one or more dedicated team members. As a deep understanding of the many components of the product along with their interactions is essential, this task is best left to senior, scientifically skilled overview engineers often dubbed system engineers or architects.

Product development teams lacking such system architects often find themselves struggling with issues arising during integration of subsystems or components; the subsystems by themselves all seem to work fine, but once they are put together everything starts falling apart. Another symptom of lacking system overview is the inability of the organization to cope with changing requirements; as the component interaction is not entirely clear, the impact of changing one component on the rest of the system is basically unknown. Teams lacking system overview are particularly opposed to changing any part of the product during development. Conversely, for development organizations to become flexible and adept at delivering new products at high pace, system overview is mandatory.

II.3 Team

[3] Couplings through one or more subsystems or components. An example of an indirect interface is the hidden coupling between a mobile device's processor speed and its mechanical enclosure or even battery subsystem through introduction of heat into the enclosure.

II.3.1 Availability of skilled workers

The availability of in-house expertise and experience has generally been of concern to product development organizations, and, following the latest labor statistics, even more so in recent years. Skill and experience are scarcer than ever, both within organizations but also outside, something leadership teams would do well keeping in mind in dealing with employees. Commonly heard suggestions to resolve or circumvent this shortage of qualified engineers by "just" outsourcing part of the work ignore the fact that development partners and suppliers typically face the same issue. Moreover, make or buy decisions should be based on broader analysis also factoring in for example business strategy and competitive edge [18]. Outsourcing core competences crucial to retain the latter, even if you can find suppliers willing and able to jump in, might in the longer run hurt rather than advance the business. If outsourcing is to be considered, it will rather be the generic parts of the product requiring generic skills that qualify for this.

Another approach, albeit one that again requires a longer breath, is to develop internal talent, selecting promising candidates and training them on hard engineering and business skills as well as soft ones like leadership and collaboration [19]. This should be a continuous process. Additionally, to bind core personnel to the organization as well as to lure in new recruits, leadership teams should take care to keep the work sufficiently engaging and attractive.

II.3.2 Team composition

As found in numerous studies in varying fields, best-performing teams consist of a mix of personalities, talents, specializations, backgrounds, gender, etc., as well as skills and experience [20]. A product development team therefore needn't exclusively consist of the brightest, most experienced, ambitious outperformers to render it effective, putting the bleak outlook on labor availability somewhat in perspective. Often, one or two motivated members in the team functioning as role models can notably affect its overall performance [21]. Product development, after all, only partly amounts to creatively or intellectually demanding work; it is mostly painstaking toil requiring above all perseverance and mindset -- or, as Thomas Edison put it, "1% inspiration, 99% perspiration" [22].

II.3.3 Team size & focus

Observational research suggests small teams perform better than larger ones [20]. Consensus for product development points to an optimum team size of seven to ten members [1]. Small teams require less internal alignment and thus less overhead than larger teams, they are leaner and more agile, and typically work faster. Since the average product development project given today's highly specialized industrial organization requires more than eight developers, projects should be split up into multiple teams focusing on one or more subsystems or disciplines.

To speed up development work and reduce time to market, leadership teams are often tempted to expand project teams and add more workers. But blindly adding more workers could quickly become counterproductive. Firstly, employing more people requires more management overhead, coordinating collaboration, aligning deliverables and schedules, etc. [23]. Secondly, and arguably more importantly, there is only so much the teams can do in parallel; many development steps need to be done consecutively as they require the outcome of prior steps as input [6]. Taking on more people in this case only slows down progress. There is, in short, an optimum to project size, and finding it remains a fine balance that can only be gained through sufficient trial-and-error. Rather than simply increasing the number of workers, product organizations in many cases had better add a couple of experts with the right experience to enable the right design decisions early on, keeping the number of needed design iterations to a minimum [20].

Another way to reduce time to market is by increasing focus. Dedicated project teams freed up to focus on one project at the time tend to work faster than project teams making use of shared resources, but they are more costly [1, 24]. Having a team exclusively at the disposal of any one of the running projects implies team members not needed full-time all the time will be idling at least some of the time. Typically, Agile methods favor dedicated teams where Waterfall methods hinging on a step-by-step or phase-by-phase approach with inherently more dead time tend to up efficiency by sharing team members across several projects. It is up to leadership to decide on the relative importance of speed versus efficiency, but these are difficult to optimize at the same time. Speed, alas, does come at a cost.

Prioritizing, focusing on fewer projects running in parallel, tends to increase R&D effectiveness [1]. The main hurdle then quickly becomes the prioritization itself, that is, making a choice, thus saying no to certain proposals, which still proves to be one of the hardest things to do.

II.3.4 Empowerment

In complex projects wielding cutting-edge technologies, involving contributors spread out over the globe, and facing short times to market, which typically is the case in today's product development, it helps to have empowered, self-steering teams. Empowered teams that are committed, motivated, and needn't be told what to do on daily basis; that can handle a little unclarity at the onset of the project and adapt to evolving insights into the customer needs; that within agreed limits can decide themselves on how to best tackle certain challenges and divide the work, are preferrable over subordinate teams lacking initiative and relying on strict hierarchy for their daily functioning. This was the case even before the advent of Agile. Self-steering teams and even agility is not exclusive to Agile techniques; Waterfall methods can be agile as well and likewise work better with empowered teams [25].

That said, in practice as well as in literature, increasingly issues with non-hierarchical teams are reported related to responsibility and accountability [26]. Frequently heard problems from companies that had recently adopted any of the Agile variants include the issue of workers supposedly "hiding behind the rules" [27]. The issue here is that telling a team it is empowered does not make it so. True empowerment, being able to handle (shared) responsibility as well as the accountability that comes with it, be it formally assigned to one role or to the entire team, requires an appropriate level of expertise and professional maturity. It also requires a safe working environment where failure is not punished but used to improve. People hiding behind the rules is often an indication of a lack of either, or both.

II.3.5 External teams

Given the complexity and the many specialized technologies that go into today's products, many companies use some form of co-development, collaborating with partners and suppliers on their offerings. Collaboration can take on many forms. Using for example the

client-supplier model, the latter is responsible for only part of the product, often a subsystem the development and manufacturing of which can be easily separated in place and time [28]. When the client uses many suppliers accordingly, it effectively becomes a system integrator, a model that was common in the automotive industry. At the other end of the spectrum, collaborating companies can opt for erecting a fully integrated project team, whereby all team members work together as if belonging to the same organization. Sometimes, for the duration of a project a new legal entity is established so that all team members indeed do belong to the same company. This model is frequently seen with large-scale infrastructure projects. For high-tech consumer products, which are typically smaller-scale and for which collaboration partners tend to change over time depending on the specific technological needs of the products under consideration, an intermediate model is often used.

Best practices as well as case studies reported in the literature point out that using some form of integration, including external teams in the regular workflow as if they were part of the same organization, benefits both effectiveness and productivity [29]. It also considerably lowers project start-up time and improves flexibility with respect to requirements and design changes. Unless the subsystem under development can be easily isolated from the rest of the system, with interfaces being relatively straightforward and fixed regardless of any changes in the rest of the system, some flexibility regarding product requirements and design changes will be called for. For today's product development then, with its short time to market and its frequently changing requirements to keep up with volatile market dynamics, the pure client-supplier model increasingly fails to deliver.

The rather rigid contractual agreements defining scope and schedule typical to the client-supplier model often obstruct rather than facilitate fruitful collaboration [29, 30]. Personal involvement with projects of this kind has demonstrated its drawbacks in practice: the extensive preparation and start-up time needed to try to get everything set in stone in excruciating detail upfront; the endless scope discussions popping up during project execution; the futile arguments about missed deadlines; the ceaseless escalations with senior management that lead nowhere and only result in growing mistrusts between the teams of the respective parties, to name just a few. What contractual handles over

one's suppliers' deliverables and deadlines one may fear to lose letting go of the strict client-supplier model, one gains in actual productivity and development speed using integrated co-development partners.

11.3.6 Soft aspects

Easily overlooked but demonstrated to be essential for team performance, is team happiness or satisfaction [27, 31]. Increasingly these days, in many an organization, weekly meetings start with a round of how everyone is doing. Next to an obvious social function, understanding team members' state of mind and trying to improve well-being also helps improve productivity. Research has shown unhappy teams to underperform [20]. In evaluative retrospectives causes for unhappiness or distress can be uncovered and mitigations or resolutions decided upon. Typically, the act of monitoring happiness alone, of showing concern for the teams' satisfaction, inherently has a motivating and inspiring effect. People feel heard and appreciated -- provided the concerns prove to be sincere and something is truly done with the findings.

In practice found to be one of the main reasons for team unhappiness is lack of empowerment: not being able to decide oneself how to go best go about one's work, sharing responsibilities conventionally lying with management. Empowerment then not only has a positive effect on team commitment and motivation, and hence productivity and speed, but also on employee turnover and the organization's attractiveness as an employer [19].

II.4 Planning & scheduling

II.4.1 Planning

Note that planning and scheduling are not the same; the plan breaks down what needs to be done to finish the project whereas the schedule describes when these steps are to be implemented and completed [6]. Planning always comes before scheduling.

Plans typically list the required steps (for example work-breakdown structure (WBS) elements, work-packages, features, or user stories) in as much detail as possible to force the teams to consider upfront the challenges ahead as well as their resolution (skills, additional workers or other resources, or budget). If not all details are available from the

on-set of the project, the needed refinement or investigations can be listed as work to be done. At the planning stage, teams should also consider the work sequence and make this explicit in a planning flowchart [6]. Some work-packages, for example, will need input from other work packages (internal dependencies) or input from outside the team or organization (external dependencies) for them to be completed. Only once all this information has been gathered do the teams have a good enough grasp of the work scope to proceed to the next stage: scheduling.

II.4.2 Scheduling

Scheduling involves the mapping of the plan's elements onto the available teams and resources, taking into account sequence and dependencies to arrive at a timeline or Gantt chart. It is important that the teams carrying out the work give input on their work elements' needed effort or lead times.

Agile enthusiast might cringe at the above. Indeed, for pure software development -- from which field Agile techniques originated -- planning and scheduling might be kept to a minimum and figured out on the fly [1]. The question of timing can then be resolved by freezing the schedule in a release cadence while allowing scope to vary. Missed features, after all, can always be delivered in a next update release. But that is a luxury usually not granted the development of physical products characterized by their fixed scope and relatively lengthy and costly design iterations. Some form of planning and scheduling will therefore remain necessary in this case [32].

Problems with scheduling in practice often can be tied to lack of planning: teams jumping to scheduling too soon, without good notion of the work-breakdown, the work sequence, or the dependencies at hand. Without planning, however, project teams not only struggle with the predictability of their deliveries, failing to appreciate workflow lead times, lacking insight in critical paths and in internal and external dependencies, but often also miss opportunities for speeding up the work for example by parallelizing workflows.

Another frequent problem with schedules has to do with the way they were made. Instead of letting the teams that are to implement the work packages give input on their respective effort or lead times, this is done for them (for example to save time or simply

because of the hierarchical nature of the organization). Often, these assumptions miss the mark completely. Worse, consequently, the teams no longer feel committed to the schedule forced upon them nor inclined to put in the extra effort needed to meet any deadline.

Counterproductive are also the many arguments at all levels within the organization that flawed schedules and missed deadlines typically trigger. Leadership teams would do well recognizing schedules in product development are rarely ever met; that they are estimates at best and should be treated as such[4]. Failure to do so, creating an atmosphere where missing deadlines has implications for careers—a "heads must roll" mentality—will motivate team members and project managers to load up the schedules with unrealistic buffers. As the best-guess estimates as they are, schedules should be made as realistic as possible; rather than introducing buffers the associated risks of missing deadlines along with their impact on timelines or budget should be highlighted and acted upon [6]. That way, the teams can better manage expectations and keep unwelcome surprises to a minimum.

This is not to say that schedules hold no value whatsoever and product development teams should no longer be bothered with deadlines. Schedules remain critical for business planning, after all. The point here is that missed deadlines should be analyzed in retrospective sessions and the causes fed back to the teams in a productive way; turning such sessions into blame games where people dodge responsibility for fear of their career should be avoided.

Finally, over the years I have noticed in many product development organizations a persistent attitude towards project planning and scheduling as if they were the exclusive domain of the project manager. This, however, in my view is a misconception, and a rather damaging one at that. Projects are managed by the entire team rather than any one person [21]. Project-based working implies a systematic approach to coming up with functioning and safe products that meet market demands understood and practiced by the entire

[4] Plans, therefore, are typically more valuable than schedules as they contain the hard information on what needs to be done. Discussions had rather center on plan than on schedule, something rarely happening in boardrooms though.

team. No project manager will be able to make a difference with a team that cares nothing for structure and control. It is therefore crucial that the entire product development department be trained on project management techniques rather than merely the management and leadership teams [26].

II.5 Quality

II.5.1 Quality management system

Having explicit business processes in place with defined output helps to improve quality of products delivered, making sure they comply to all requirements applicable [6]. This holds for any field, and certainly for product development, despite the often-heard notion creative processes cannot be captured in a straitjacket and documentation would merely serve the need of pencil pushers.

Most manufacturers have some sort of quality management system in place for developing new products; the problem usually is applying it. Firstly, because the teams that need to use them fail to thoroughly understand the business processes in question [33]. In part, this may be due to the above-described opposition to process thinking and documentation in general, but it often also stems from the fact that the quality management systems in use are outdated and no longer suited for present-day state of business. Such quality systems typically contain large amounts of waste, things that are done a certain way because they have always been done that way. Obviously, to gain buy-in from the product development departments one needs to eliminate such waste.

Secondly, because many quality management systems suffer from poor practicability, further feeding the misconception that product development doesn't lend itself for quality management. A quality management system spanning over 800 pages throughout more than hundred documents (a real-life example) is not workable[5]. Better to keep it short and refrain from over-specification, especially when just setting up quality management: not all processes need to be described and formalized in painstaking detail. The main purpose of process definition for product development is for it to be reproducible and

[5] Matters in this specific case were even worse as part of the hundred or so documents where just empty shells containing nothing but preformatted text.

traceable, so that design iterations, accompanied by unambiguous rationales, can be checked for correctness and compliance, and easily changed if needed.

II.5.2 Staged development

Product quality systems nowadays typically call for any form of staged development, dividing product creation into consecutive stages from market study and feasibility, to concept development and prototyping, industrialization and market introduction, and finally volume production [34].

Staged development processes are frequently applied too rigidly. Arguments then arise about the exact phase the project should be in; gate review meetings turn nasty when expected deliverables turn out to be missing or incomplete and projects therefore held up. Keeping in mind the purpose of staged development, however -- that is, a) to warrant product quality, and b) to be able to identify as early as possible products that will not make it to market[6] -- will help resolve such stand-offs.

Making sure all necessary steps have been taken to check the product's viability at each gate, we find it is not that important at which stage exactly the checking is done, as long as it is done (satisfying purpose a) and done as early as possible (satisfying purpose b). It may well be beneficial for the project on a whole to proceed to the next stage, for example for timing purposes or because of resource availability, even if not all stage goals have been met.

Alternatively, different parts of the product may be allowed to reside in different development stages, each part thus following its own staged development with its own dedicated gate deliverables. Part a and b, for example, may be in prototyping while part c is still in the conceptual phase, preventing the teams working on a and b having to idle

[6] Identifying early product failures and terminating corresponding projects helps increase R&D effectiveness and speed as it frees up time and resources that can then be allocated to more promising endeavors [14]. Because of this, all variations of staged development follow the same rationale, going from technically risky, relatively low-cost activities to low-risk, high-costs ones. Only once all technical risks have been eliminated and commercial viability proven are organizations ready to make the big investments associated with manufacturing and market launch.

until the rest has caught up. In practice, this multi-track staged approach works well and can reduce time to market provided everyone is kept well-informed on the status and the issues at hand along with their dependencies, and the (technical as well as financial and scheduling) risks of advancing only part of the project to the next phase are closely managed [35].

Always to be avoided, however, yet regularly encountered in practice, especially with inexperienced leadership teams, is allowing (parts of) the project to advance to the next phase despite failure to meet milestone deliverables merely to delay painful decisions. Project staging's main purpose is to enable go/no-go decisions early on and free up the resources for likely successes [36]. Giving the go, though, is always easier; it takes courage to terminate a project -- one of the main reasons many dead-on-arrival projects tend to drag on.

II.5.3 Testing

Common sense dictates that before launch products be thoroughly verified and validated. Perhaps less obvious is that verification should be a continuous activity throughout the product development process and even afterwards, during the product's life cycle [37]. Synthesis/verification iterations should start all the way at the beginning of the project with the setting up of the business case, and continue well beyond market introduction, identifying reliability issues well before they occur in the field.

Testing, checking, making sure the product does what it is supposed to do, is such an important aspect of product development that compared to strict design work it likely will take up at least an equally large share of overall development time, budget, and resources [31]. Project stage gates, each further detailing of the product, should call for a full set of verification reports and documented prove of compliance. Quality needs to be tried at each point[7].

[7] Product requirements need to be checked against market demands and non-functional business requirements; the system architecture and specifications need to be checked against the product requirements; next, the detail designs of all subsystems along with their interfaces need to be verified against their specifications; subsequent system integrations need to be tested against system specifications, and so on, all the way up to

Many troubled product development efforts suffer from lack of verification and validation; sometimes verification is viewed as a singular final step in the development process that may even be omitted altogether and replaced by mandatory certification testing at external notified bodies. Such approaches are bound for disaster, leading to poor manufacturing yields and issues in the field requiring recalls and sometimes even resulting in considerable claims leaning heavy on profitability. To turn this around, a quality mind-set needs to be built into the genes of the entire product organization; in an almost legalistic approach product performance, regulatory compliance, etc., should be treated as non-existent until proven otherwise through documented evidence [31].

An easy indicator as to whether verification has been sufficiently accounted for in any development organization is the size of its test department. As a rule of thumb there should be as many or more testers than there are strict developers. (Often, though, developers also do test work, so this indicator might give a clouded view.) Another indicator of quality-mindset is the existence of a test department that focuses on products deployed to the field, for example for checking regression of software updates, product compatibility with other third-party products, or lifetime and reliability issues. The automotive and high-volume consumer electronics sectors typically serve as good benchmarks for quality standards.

II.5.4 Demonstrations and documentation

Use of consecutive prototype demonstrators advocated in modern product development methodologies such as Agile, Lean Development, and, more recently, Continuous Engineering to carve out the path to customer needs helps to speed up time to market, allowing for flexibility and quick, actionable feedback that can be used as input for follow-up design iterations [38]. Demonstrations are also a great way of keeping everyone updated on progress and involved; they are more engaging -- and often more informative -- than progress reports filled with tables and graphs and the obligatory green/orange/red traffic lights. But use of demonstrators is no excuse for circumventing documentation or doing away with requirements traceability, nor does it dismiss the

the verification of the manufacturing and supply chain processes in pilot production and the validation of the offering in field trials.

development teams from truly understanding product behavior and the underlying science. Exclusively relying on feedback gained during demonstrations might quickly turn development into trial-and-error tinkering reflective of pastime rather than of professional engineering.

II.5.5 Risk management

Among the risks to be managed by product development teams are technical risks and project risks. Technical risks (potential weaknesses in the system) can be identified and mitigated using failure mode effect analysis (FMEA), tailored to either the design (D-FMEA) or the manufacturing process (P-FMEA) [39]. In practice, despite the technique being around for quite some time, many organizations still struggle with identifying technical risks. It takes a lot of experience with the process of FMEA's as well as with the subject matter at hand to get usable results. Inexperience typically leads to endless sessions erroneously rendering the entire design and process technically risky. FMEA therefore runs the risk of being considered a mere administrative formality and, alas, of not being used to its full potential.

Typically, the main project risks for product development projects relate to scheduling [6]. Because of the complexity (owing to the many, sometimes surprising dependencies) and the technical uncertainty involved, product development projects prove to be hard to both plan and schedule accurately. One way to reduce scheduling risks is by sticking to proven technology as much as possible in product development projects, steering clear of research (the outcome of which is by definition unsure). Maturing new, unproven technology, after all, takes a long time – typically years or even decades. Admittedly, new products always require at least some amount of research, but one can aim to achieve novelty through the combination of state-of-the-art yet existing technology rather than through invention of entirely new technology. It is thus not the building blocks themselves that need to be new but rather their combination.

That isn't to say research-heavy R&D projects should always be shied away from[8], but they should be handled appropriately, the research aspects along with the technical and

[8] We might have never witnessed some of the most useful products around today otherwise.

scheduling risks clearly highlighted so that expectations can be set right from the start. Many product development endeavors ultimately approved by leadership teams and financiers that are far from the science -- as is often the case in today's high-tech business environment -- reach a dead end specifically because of the failure to appreciate the research involved, resulting in unrealistic, overoptimistic expectations for the required development effort or time to market [40]. Disillusioned once the full extent of the challenges at hand becomes painfully apparent, these leadership teams finally feel compelled to abandon the project -- perhaps justifiably so, but for reasons that could have been clear long ago.

An elegant way to keep technical and project risks under control is by advancing research parts to the early stages of the project, establishing technical and commercial feasibility well before committing to the implementing of the clearcut parts of the product, thereby avoiding wasted effort [41]. Alternatively, the research and development can be done in parallel to save time, but then at every step the risks need to be carefully weighed, and the costs of moving ahead with development of potentially unfeasible parts of the product balanced against the benefits of earlier market introduction.

II.5.6 Knowledge sharing & transfer

With today's prevalent co-development model, using teams distributed over multiple locations in different parts of the world, access to shared data is more critical to success than ever. Shared IT infrastructure (collaboration and workflow tooling) can be of help here, but while software tooling can facilitate the sharing of information -- data, specs, documents, etc. -- it cannot warrant the sharing of knowledge – that is, the true understanding and skill gained through decades of toil [42].

This becomes painfully apparent when knowledge needs to be transferred from one site to another (for example because of a company acquisition or the shutting down of one or more locations). In one practical case in which several sites were shut down and their operations taken over by a newly established office, an attempt was made to capture the knowhow of the soon-to-become redundant workers into an advanced knowledge system. Despite the project running for well over four years, the knowledge system's functioning in the end turned out to be highly controversial. Luckily, some of the highly

experienced employees were willing to move to the new location after all and “knowledge transfer” could in that way be secured.

Unlike information, knowledge proves very difficult to share or transfer. It needs to be built up from the ground up, which is a painstaking process that often takes years rather than months. This should be kept in mind when doing business planning.

III Practical recommendations

III.1 Requirements

- Refrain from overly relying on external consultancies to understand market needs and generate product proposals. Their analysis will typically be based on market consensus, leading to unoriginal products. Recruit and develop in-house product visionaries who can combine technical and business skills to come up with products that are truly ahead of the competition.
- If development teams are bogged down by product requirements changing all the time, failing to reach consensus on what to build, consider using an iterative development approach such as Agile or Design Thinking to arrive at products that seamlessly connect to market demands. Avoid introducing arbitrary design freezes just for the sake of scheduling because that may lead to quick results that nobody is waiting for.
- When starting with requirements management, don't feel discouraged if it starts growing into an untamable multi-headed monster at first. Get rid of the spreadsheet or word processors and start using dedicated database tooling. It takes some practice to be able to find the right balance in what to specify and what to leave open, how to make it sufficiently actionable, etc. Assign dedicated roles such as system engineers, system architects, product owners, or product managers to the task of setting up, managing, and keeping track of the requirements.

III.2 Scope

- Make scope explicit in sufficient detail and go over it with whole team, including stakeholders and co-development partners. Failing to do so will result in people making assumptions, the erroneousness of which often pops up only at a later stage in the project when repairs will be time-consuming and costly.
- Pay extra attention to the boundaries of the individual teams' sub-deliveries as that is where things most often go wrong. Assign the task of managing these interfaces to dedicated team members such as system engineers or architects with a sound overview on all parts of the product. Technical control over the interaction of the product's many parts enables fast and effective handing of requirements and design changes, and thereby shortens time to market.

III.3 Team

- As skilled workers are scarcer than ever, start appreciating and treating them as such. The fight for market share and even survival is very much a fight for talent. Select and develop promising employees and bind them to the organization by keeping the work engaging.
- When considering outsourcing part of the work and engaging development partners, make sure to select the right parts for this. Ideally, it is the generic parts of the product requiring generic engineering skills that qualify. Outsourcing the core of the product which distinguishes it from that of the competition means trusting your competitive edge into the hands of others.
- In configuring the teams, remember that diverse teams consisting of a mix of backgrounds as well as skills and experience perform best. Having one or two motivated workers in a team can significantly boost morale and performance. Also, put enough consideration into putting together the steering committee or project review team, as they have a similarly big impact on project performance. Like all team members, steering committee members should appreciate their role and refrain from acting like directive bosses or as if being part of the project team, doing its work for it, debating content rather than process.
- Use small teams as they are more agile and need less overhead. Don't just add more workers to speed up development as this may result in quite the opposite. More workers means more overhead, more alignment, more communication, and more opportunities for this to go wrong. Additionally, there might be only so much that can be done concurrently. Analyzing workflows and critical paths, we find that product development lead times are often dictated by design-verification iterations leading to consecutive prototypes with increased maturity level. Some overlap is possible, but it makes no sense to start the next prototype without completing necessary verification of the previous one only to push out the fixing of potential problems. To speed up, add expertise and experience instead: one or two experienced experts to enable the right design choices early on and keep the number of required prototypes down will do much more for time to market.
- Alternatively, to speed up the work, add focus and use dedicated teams, which means daring to say no to other projects, however hard that may be. Resource

efficiency and development speed are tradeoffs: one comes at the expense of the other. Decide which is more important to you and organize your R&D department accordingly.

- Develop your R&D organization's engineering and leadership and collaboration skills so that its members can be empowered. Empowered, self-steering teams work faster and are more effective handling requirements and design changes. Beware, though, of empowering teams that are not ready yet and might yield under the weight of increased responsibility. Also make sure to create a safe working environment where errors are treated as ways to improve rather than as motives for punishment. It will benefit people assuming accountability.
- Integrate your co-development partners' teams into your regular workflow, treating them as if they were part of the organization. Rigid contractual agreements minutely specifying deliverables and timelines only yield a false sense of control; they take forever to agree and usually only end up in endless scope discussions and arguments about missed deadlines, deteriorating strategic relationships, and causing mistrust back and forth. Using integrated teams offers more flexibility with respect to changes, smoother collaboration, and less miscommunication. Integrated teams are happier and more motivated, and therefore more productive.
- Monitor teams' happiness. It serves a social function as well as helps you improve productivity. Happy teams are more motivated and productive than unhappy ones. That said, take your teams' dissatisfaction seriously and put in the effort to improve, otherwise you will only aggravate the problem. A frequently heard cause for dissatisfaction in product development teams is lack of empowerment.

III.4 Planning & Scheduling

- Planning comes before scheduling; don't confuse the two. The plan is more important as it specifies what needs to be done; the schedule is a mere estimate. Don't start scheduling before you have a sound plan otherwise your timeline estimates will make even less sense and be even more off.
- In pure software development, planning and scheduling can be done on the fly as scope is allowed to vary. In hardware development yielding physical products, we

cannot do without a plan (and rarely ever without a schedule). Agile and Waterfall techniques can then be combined.

- Let the teams do their own planning and schedule (or at least give their input); don't do this for them. You'll miss the mark and lose the teams' commitment and motivation.
- Treat schedules as the best-guess estimates as they are. Analyze reasons for missing deadlines in retrospective sessions and come up with ways to improve. Don't go looking for people to blame; don't create a "heads must roll" atmosphere in the organization. Schedules will become longer and longer, and loaded up with so many buffers nobody will be able to understand them anymore.
- Contrary to common belief, planning and scheduling are not the exclusive domain of project management. Projects should be managed by the entire team, and so the entire team up to the most junior engineer needs to be trained on project control techniques.

III.5 Quality

- Train your teams on your quality management system and start using it. See what works, change what doesn't – and regularly update, eliminating waste. Check if it is still practicable, and shorten accordingly, avoiding over-specification. Whenever projects feel a need to deviate from the quality management system for good reasons, let them and evolve your quality management system accordingly.
- Document your design, not only the as-built drawings for manufacturing but also how you came to a specific design, including design considerations and decisions, calculations, simulations, verification plans and reports, etc. Don't let iterative development techniques such as Agile be used as excuse to bypass documentation. Firstly, for a growing number of products (including software products) stricter safety and environmental compatibility regulations require the technical file to contain above-mentioned information on the design as well as manufacturing process. (Medical devices are just one example.) Secondly, having access to such design rationale makes the handling of regularly required product changes during the product lifecycle (for example because of parts going obsolete) as well as development of follow-up iterations much easier and quicker.

- Agree with the entire team on the documentation to be delivered at all stages of the project upfront. Don't just find out a week before the stage gate review.
- Use project stage gates wisely. Milestone deliverables are not set in stone, and if beneficial for the project on a whole they may deviate as long as product quality is warranted, and everyone involved agrees on the product's technical and commercial feasibility. (If the latter is not true and the signs are mostly red, don't hesitate to terminate the project.) Allow different parts of the product to be in different stages to speed up time to market, even if it makes management and administration more cumbersome, but do have sound risk management in place then.
- Make sure prototypes are thoroughly tested. Don't believe verbal feedback; insist on written evidence in the form of test reports. Similarly, all design steps need to be verified, starting all the way at the setting up of the business case and product requirements. Quality should be tried at each point. Don't release products to the field that have not been stress-tested extensively. You don't want to find out about reliability problems only once you are in the field.
- Product development needs as much testing as it does strict design work. Test and design teams should be about equal in size. Testing should continue well after product release and should in fact never stop, checking for potential failures of products already in the field, preparing for potentially needed mitigations.
- Iterative development techniques such as Agile and Design Thinking relying on demonstrator feedback for upcoming design iterations shouldn't transform scientifically sound engineering into hobbyist trial and error. Everything said about documentation still holds true. The science behind the product needs to be properly understood to be technically fully in control and guarantee quality.
- Technical risks in any development project can be mitigated using failure mode effect analysis (FMEA). FMEA sessions should include engineers with sufficient experience and system overview for them to be used to their fullest potential and their results to hold value. Otherwise, FMEA's run the risk of being viewed merely as administrative overhead.
- R&D projects are hard to plan and, especially, schedule accurately. Scheduling risks can be reduced by limiting research and keeping to proven technology as

much as possible, arriving at the novelty of a product through new combinations of existing technologies.

- If research cannot be avoided, clearly indicate for which parts of the product fundamental investigations are needed along with the technical and scheduling risks involved. Failure to do so will set wrong expectations and might end up in disillusioned senior leadership teams prematurely terminating the project.
- Keep research risks in R-heavy R&D projects under control by completing all research tasks first (in a separate stage sometimes dubbed “advanced development”) before committing time and resources to the strict development work, thus preventing wasted effort. You can do research and development in parallel but then some form of risk management is needed to be able to make sound go/no-go decisions at every stage.
- Finally, knowledge cannot be transferred, no matter how smart or advanced your knowledge system. Information, data, documents, etc. -- those can be shared, but knowledge has to be built from the ground up. Either transfer the people holding that knowledge or be prepared to spend a long time gaining it. Don’t close any sites before you have the ones taking over almost fully operational, and plan for some overlap.

IV Conclusion

The product development challenges in the sphere of product requirements, scope and interface definition, team composition, planning and scheduling, and quality encountered in practice along with suggestions for their resolution described here were found to be principally unrelated to the specific project management method chosen. Some techniques indeed lend themselves better to some business goals (for example Agile for flexibility with respect to requirements, Lean for development quality, V-model for validation and verification traceability) and might be more prone to specific issues (for example poor quality control with Agile, work sequence rigidity with V-model) but that doesn't mean these goals cannot be achieved or the issues satisfactorily tackled with any of the other available techniques, either through some alteration or combination of techniques. Product development success, therefore, in practice proves to be more dependent on the organization's willingness and capacity to learn and adapt rather than on the specific method chosen.

References

1. Denning Stephen. 2018. The Age of Agile: How Smart Companies Are Transforming the Way Work Gets Done. New York: AMACOM.
2. Kim Dikert, Maria Paasivaara, Casper Lassenius, Challenges and success factors for large-scale agile transformations: A systematic literature review, Journal of Systems and Software, Volume 119, 2016, Pages 87-108.
3. Ruohonen, J. A review of product safety regulations in the European Union. Int. Cybersecur. Law Rev. 3, 345–366 (2022).
4. Hodges, Christopher, European Regulation of Consumer Product Safety, Oxford, 2005; online edn, Oxford Academic, 22 Mar. 2012.
5. Wang Charlene, Teo Wee Eong, Tian Lingling, Ramakrishna Seeram, Liao Susan, Medical Devices: Regulations, Standards and Practices. Netherlands: Elsevier Science, 2015.
6. Karl Ulrich, Steven D. Eppinger. 2007. Product Design and Development. McGraw-Hill Education - Europe; 4th edition.
7. Laura Fish, Scott Kiekbusch. 2020. The Designer's Guide to Product Vision. New Riders; 1st edition.
8. Alexander Kossiakoff, Steven M. Biemer, Samuel J. Seymour, David A. Flanigan. 2020. Systems Engineering Principles and Practice (Wiley Series in Systems Engineering and Management) 3rd Edition
9. Tom Gilb. 2005. Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage Butterworth-Heinemann; 1st edition (August 30, 2005).
10. Juan M. Carrillo de Gea, Joaquín Nicolás, José L. Fernández Alemán, Ambrosio Toval, Christof Ebert, Aurora Vizcaíno, Requirements engineering tools: Capabilities, survey and assessment, Information and Software Technology, Volume 54, Issue 10, 2012, Pages 1142-1157.
11. Schneider, Joan, and Julie Hall. "Why most product launches fail." Harvard Business Review 89, no. 4 (2011): 21-23.
12. Kees Dorst, The core of ‘design thinking’ and its application, Design Studies, Volume 32, Issue 6, 2011, Pages 521-532.

13. Jan Bosch, “Achieving Simplicity with the Three-Layer Product Model,” November 2013, Computer 46 (11): 34-39.
14. Anil Mital, Anoop Desai, and Dhanraj Rewal, 2015, “Product Development: A Structured Approach to Consumer Product Development, Design, and Manufacture,” Elsevier.
15. Benjamin S. Blanchard, John E. Blyler, 2016, System Engineering Management, John Wiley & Sons, Inc.
16. Daniel Jacobson, Greg Brail, Dan Woods, 2011, APIs: A Strategy Guide, O'Reilly Media, Inc.
17. Manuel E. Sosa, Steven D. Eppinger and Craig M. Rowles, The Misalignment of Product Architecture and Organizational Structure in Complex Product Development, Management Science, Vol. 50, No. 12 (2004), pp. 1674-1689
18. Rubén Medina Serrano, María Reyes González Ramírez, José Luis Gascó Gascó, Should we make or buy? An update and review, European Research on Management and Business Economics, Volume 24, Issue 3, 2018, Pages 137-148.
19. Gutierrez-Gutierrez, L.J., Barrales-Molina, V. and Kaynak, H. (2018), "The role of human resource-related quality management practices in new product development: A dynamic capability perspective", International Journal of Operations & Production Management, Vol. 38 No. 1, pp. 43-66.
20. Sivasubramaniam, N., Liebowitz, S.J. and Lackman, C.L. (2012), Meta-analysis of NPD Team Performance. J Prod Innov Manag, 29: 803-820.
21. Edmondson, A.C. and Nembhard, I.M. (2009), Product Development and Learning in Project Teams: The Challenges Are the Benefits. Journal of Product Innovation Management, 26: 123-138.
22. Cohen, Alan. Prototype to product: a practical guide for getting to market. " O'Reilly Media, Inc.", 2015.
23. Sethi, R., Smith, D. C., & Park, C. W. (2001). Cross-Functional Product Development Teams, Creativity, and the Innovativeness of New Consumer Products. Journal of Marketing Research, 38(1), 73–85.
24. B. J. Zirger and J. L. Hartley, "The effect of acceleration techniques on product development time," in *IEEE Transactions on Engineering Management*, vol. 43, no. 2, pp. 143-152, May 1996,

25. Jiyao Chen, Fariborz Damanpour, Richard R. Reilly, Understanding antecedents of new product development speed: A meta-analysis, Journal of Operations Management, Volume 28, Issue 1, 2010, Pages 17-33.
26. Holland, S., Gaston, K. and Gomes, J. (2000), Critical success factors for cross-functional teamwork in new product development. International Journal of Management Reviews, 2: 231-259.
27. Tolfo, C., Wazlawick, R.S., Ferreira, M.G.G. and Forcellini, F.A. (2011), Agile methods and organizational culture: reflections about cultural levels. J. Softw. Maint. Evol.: Res. Pract., 23: 423-441.
28. Sabine Fliess, Urban Becker, Supplier integration—Controlling of co-development processes, Industrial Marketing Management, Volume 35, Issue 1, 2006, Pages 28-44.
29. Henry Chesbrough & Kevin Schwartz (2007) Innovating Business Models with Co-Development Partnerships, Research-Technology Management, 50:1, 55-59.
30. Jassawalla, A.R. and Sashittal, H.C. (1998), An Examination of Collaboration in High-Technology New Product Development Processes. Journal of Product Innovation Management, 15: 237-254.
31. Dahlgaard JJ, Khanji GK, Kristensen K. Fundamentals of total quality management. Routledge; 2008 Jan 28.
32. Salvato, J.J. and Laplume, A.O. (2020), Agile Stage-Gate Management (ASGM) for physical products. R&D Management, 50: 631-647.
33. Nanda, Vivek. *Quality management system handbook for product development companies*. CRC press, 2005.
34. Robert G. Cooper, Scott J. Edgett & Elko J. Kleinschmidt (2002) Optimizing the Stage-Gate Process: What Best-Practice Companies Do—II, Research-Technology Management, 45:6, 43-49
35. D Anita Friis Sommer, Christian Hedegaard, Iskra Dukovska-Popovska & Kenn Steger-Jensen (2015) Improved Product Development Performance through Agile/Stage-Gate Hybrids: The Next-Generation Stage-Gate Process?, Research-Technology Management, 58:1, 34-45.
36. Pilar Carbonell-Foulquié, Jose L Munuera-Alemán, Ana I Rodríguez-Escudero, Criteria employed for go/no-go decisions when developing successful highly

innovative products, Industrial Marketing Management, Volume 33, Issue 4, 2004, Pages 307-316.

37. P.G. Maropoulos, D. Ceglarek, Design verification and validation in product lifecycle, CIRP Annals, Volume 59, Issue 2, 2010, Pages 740-759.
38. Bosch, J. (2014). Continuous Software Engineering: An Introduction. In: Bosch, J. (eds) Continuous Software Engineering. Springer, Cham.
39. Mikulak, Raymond J., Robin McDermott, and Michael Beauregard. The basics of FMEA. CRC press, 2017.
40. Cooper, Robert G., and Elko J. Kleinschmidt. "Winning businesses in product development: The critical success factors." Research-Technology Management 50, no. 3 (2007): 52-66.
41. Ashruf, Colin. "Planning the “Unplannable”–the Innovation Management Paradox." Journal of Innovation Management 10, no. 1 (2022): XI-XVI.
42. Linda Argote, Paul Ingram, John M Levine, Richard L Moreland, Knowledge Transfer in Organizations: Learning from the Experience of Others, Organizational Behavior and Human Decision Processes, Volume 82, Issue 1, 2000, Pages 1-8.

Acknowledgements

The author wishes to thank Delft University of Technology for its support, specifically Prof. Dr. P.J. French for the many useful discussions on this topic.