lateralworks >

FTTM Product Development Best Practices

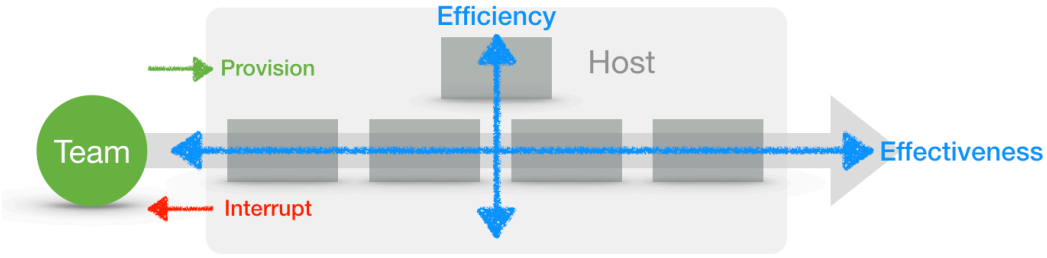
Document Revision, 2018.002

Initiated in Silicon Valley in the early 1990's, lateralworks Fast-Time-to-Market (FTTM) Practices are based on an ongoing worldwide study into the drivers of new product development success.

Table of Contents

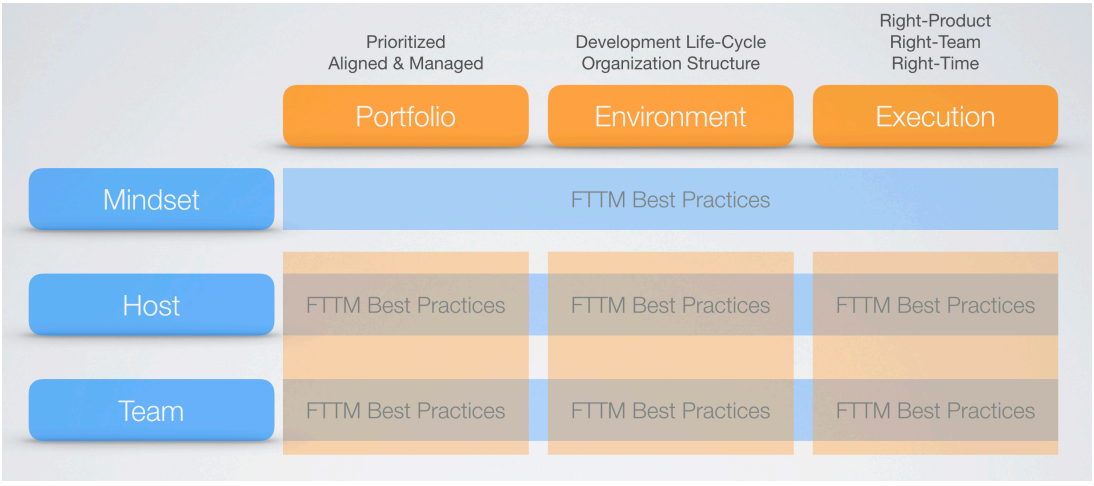
Introduction

This document outlines some of the FTTM Best Practices (Fast-Time-to-Market), first researched by lateralworks in the early 1990's, through an extensive multi-company study of over 500 people involved in fast-to-market projects in Silicon Valley. lateralworks has engaged with hundreds of teams trying to accelerate the delivery of new technology products to market, and continues to study fast team practices through its international consulting practice. Each engagement builds on the research database in terms of both changing practices and deployment tactics.



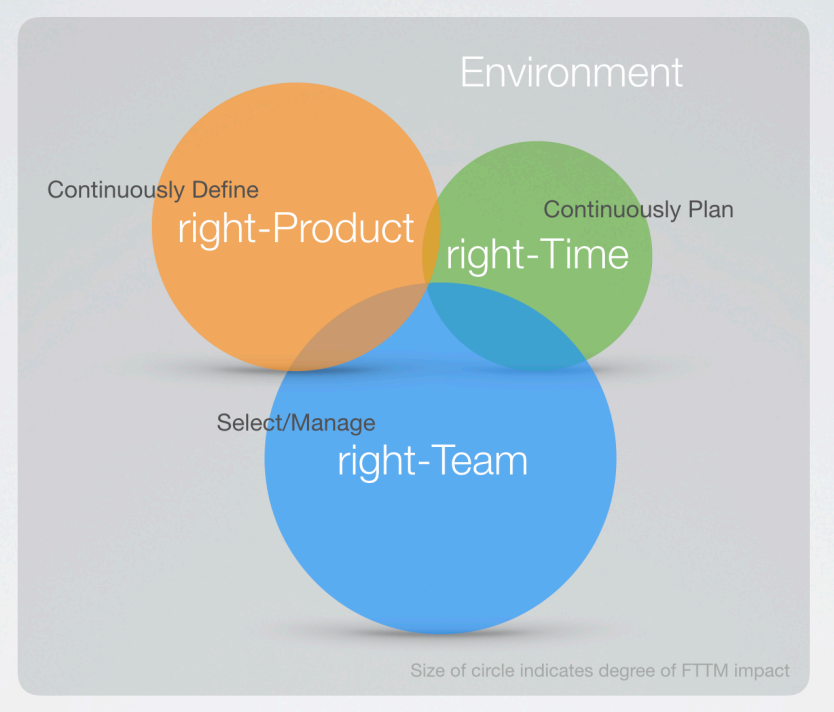
Host...the organization outside of the project team, that provides resources, support and guidance; the "functional management hierarchy." The host can "provision" and "interrupt" teams; it can create the conditions for success or failure of development projects. Fast organizations focus on rapid provisioning and interrupt removal.

FTTM outcomes are a function of two inputs; 1) the behavior of the team and 2) the environment within which the team operates. Individual project teams can achieve fast cycle time results, often at the detriment to other projects (by stealing resources). The portfolio of projects will therefore remain slow if the environment within which the teams operate is not oriented to help teams move faster.



FTTM Best Practices Framework; the provisioning Host, a set of practices organized around Mindset, Environment, and Portfolio

Senior management must create the right environment through specific behaviors and mindset, a fast development framework, lateral organization structures, and fast decision-making systems. It is easy to make one project go fast at the expense of others, so the challenge is to balance the portfolio so the highest value projects get the resources they need in order to get to market quickly.



The "FTTM System" is designed to deliver the right products (what customers want and value) at the right time.

We call this environment the "Host" environment. In fast corporate cultures, the Host empowers heavyweight teams, removes interrupts, and provisions teams so they can be successful. In normal/slow corporate environments the Host tends to interrupt teams by not providing the resources when needed, filling the pipeline with more projects than there are resources to get them done in a timely manner, not developing "bench strength" of skills needed to create the technology, and controlling cost (at all costs) without understanding its economic impact on speed (i.e. arriving to market late).

In short... provision teams for success, eliminate the interrupts.

Framework

	Portfolio	Environment	Execution		
	Prioritized Aligned & Managed	Development Life-Cycle Organization Structure	Right-Product	Right-Team	Right-Time
1.0 Mindset	1.1 Grasp enormous financial impact late products have on profitability and growth				
	1.2 Create fast culture through close-in oversight of product roadmap				
	1.3 Systematically dismantle slow structures				
	1.4 Accelerate the threshold of pain; reward early warning behavior				
	1.5 Fast decision-making system				
	1.6 Cost-of-delay is understood at individual contributor level; used to drive urgency				
2.0 Host	2.1 Fuzzy-front-end is managed	2.6 Fast development framework		2.8 Cross-functional core teams (heavyweight & dedicated)	
	2.2 Projects are prioritized based on business objectives	2.7 Provision teams, eliminate interrupts		2.9 Project-based performance measurement	
	2.3 Projects continuously aligned with resources and skills available			2.10 Freedom Level 1 empowerment	
	2.4 New products are tracked to break-even				
	2.5 Kill projects that fail to meet objectives; early				
3.0 Team	3.1 Co-develop with tier 1 customers; continuously refine requirements	3.2 Schedule as driver vs. reporter	3.3 Translate customer requirements to product definition, reconfirm throughout development	3.8 Cross Functional Core Team ROLES not FUNCTIONS; lateralized	3.10 Aggressive planning
			3.4 Know what they value, when they want it, how much they will pay for it	3.9 Get best people; mix of inside & external SMEs; virtual teams	3.11 Macro-micro planning; micro near, macro far, short interval scheduling
			3.5 Understand customer wants vs hows; prioritize the drivers that maximize customer satisfaction		3.12 Team planning, simulation, continuous scrubbing, breakdown, ownership
			3.6 Start design while refining requirements; never freeze specs		3.13 Market/customer driven versus resource constrained
			3.7 Fail-Fast; integrate/proto early, frequent & many do-it-try-it-fix-it learning cycles		3.14 Know the gap; drives before-the-fact urgency
					3.15 Refresh Planning; continuous schedule pull-in

Best Practices based on lateralworks ongoing research project into the best practices of fast teams. The study was initiated in 1990 and continues today, with new ideas and insights being added through client consulting engagements around the world. Both the practices and the deployment tactics have evolved through continuous improvement, cultural adaptation, and changing collaboration technology.

Best Practices broadly fall into three categories: Mindset, Host, and Team (horizontal axis). Down the vertical columns is a second classification; Portfolio, Environment, and Execution. For example; "2.1 Fuzzy-front-end is managed" is a practice performed by the Host that relates to the portfolio planning and management process. Most of the Portfolio practices are owned by the Host, with the exception of "3.1 Co-develop with tier 1 customers; continuously refine requirements" which belongs to the Team, but it relates to how the Portfolio is managed.

One could argue for a different classification of some of the practices, as they could fall in multiple locations on the matrix. We chose this configuration (above), but what is more important is that there are mindset, host and team practices we observed that enabled the right products to get to market at the right time. Overriding these practices was a corporate "mindset" that speed was the most critical asset of the company, and this mindset was understood by the leadership team and practiced consistently. They didn't just say it, they did it.

In the following pages, we will generally describe the practices that make up each category using a contrasting technique to accentuate the differences in terms of “normal” (or the typical team/corporate environment) and “best” in class. Although it may seem as though we are exaggerating at times, all of the “normal” conditions we write about are observed in real world situations. Each new project generates more live examples.

Few companies follow all the “best” practices, but a lot of them follow many of the practices described as “normal,” and they also considered it normal to behave this way. The difference between “normal” and “best” separated the fast from the slow teams/companies.

When we speak about speed, we always mean “right product delivered at the right time.” FTTM is not just being fast, but also delivering what customers want and value at a given point in time. Balancing this equation is very hard; on time with the right product, not a de-featured or technically compromised (in order to get it out the door on time) product. Going fast is relatively easy when product functionality or quality is compromised (i.e. traded-off), but what the best in class teams do is to deliver what customers want, when they want it. This requires knowing what they want (listening to them), being flexible enough to adapt as their requirements change (when they find out they are not sure about what they want), and then being able to execute on aggressive timelines through continuously pulling-in (accelerating) the schedule.

They did this by continuously changing the product and continuously changing the schedule. This is counterintuitive as one would expect that the way to solve the problem would be to fix as many of the variables as possible. When teams fixed the spec and fixed the date, they almost always failed - missing both by wide margins. Why? Because nothing is fixed and everything is always changing, especially when it comes to technology development.

On the contrary, fast teams kept the product definition fluid and the schedule fluid, always adapting them to the dynamic conditions of ever changing cutting edge technology development. The practices outlined in this document describe how they did/do it - how they execute using counterintuitive thinking. To understand why this is counterintuitive thinking is to really understand the best practices.

How to use this document

This document was designed as a reference guide for teams and organizations implementing new product development improvement programs. Typically, organizations select a few of the practices to implement with multiple project teams. They select the practices that they feel they need to improve.

The first step is to identify what the practice is (in your context) and secondly to determine how it might be adapted to fit the culture and the specific project conditions. These are not plug-n-play practices. They need to be adapted and modified for each environment. Rather, these are more guidelines than how-to instructions.

This document does not have to be read front to back. Rather, the reader can select specific practices based on interest. Each best practice requires a chapter in a book to fully describe. We have simply summarized them in this document.

It turns out that after 25+ years of implementing these best practices the deployment technique tends to be the key factor in determining success. To know and understand a practice versus to actually apply it are two very different actions. The later being much harder.

Further, how it is deployed is as important as understanding the practice itself, because each practice must be adapted to the local culture, team skill levels, and readiness of people and organizations to try new ways of working.

New Product Development Best Practices

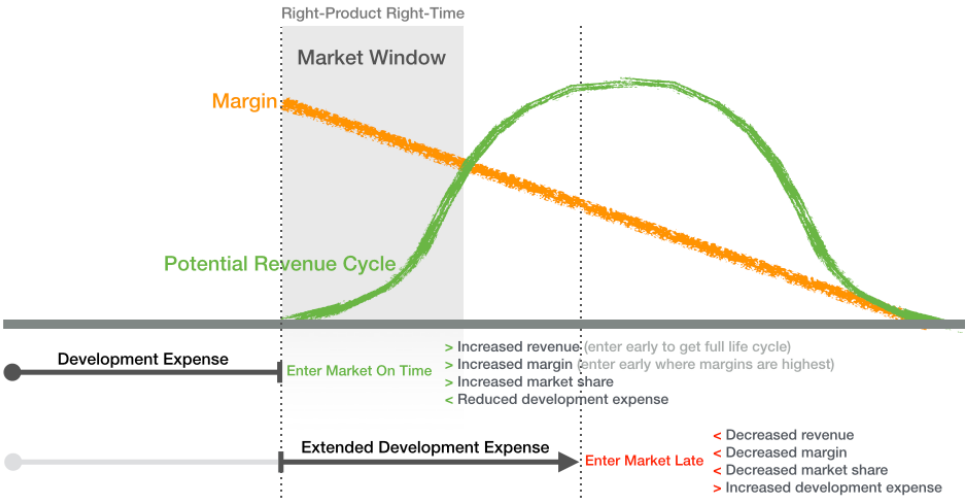
1.0 Mindset

1.1 Grasp enormous financial impact late products have on profitability and growth

Summary

Best product organizations recognize the cost of being late to market, and for every day saved, there is an enormous leverage on profit. Maintaining or pulling in the new product development schedule is therefore the highest priority and has a high profile within the company.

Normally, to "make schedule king" and to get to market on time would require a significant, perhaps unacceptable, trade-off to quality and/or costs.



Normal

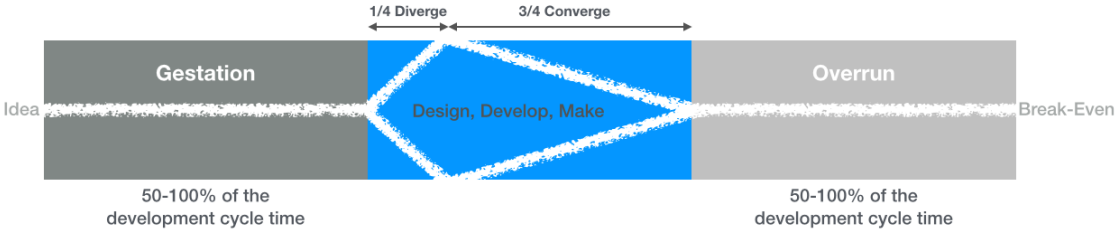
In order to be faster, something has to be sacrificed. "You don't get something for nothing." Five development objectives are involved in possible trade-off decisions: feature quality, defect quality, development cost, product cost and TTM. To go faster, quality or cost must be sacrificed. Making schedule king means building a product with less performance or fewer features, or it means "blowing" the development budget. Management accepts the inevitability of schedule re-plans, and is complicit in permitting the schedule to slip. Management demands schedule reductions to hit target as if they can be achieved by mandate.

Best

Top management articulates the need for speed. They know that reducing time-to-market (TTM) has a huge impact on profitability over the lifetime of the product. They know that for every day

saved, there is an enormous leverage on profit because the revenue to recover the additional development cost is earned sooner. The leverage is so high that you can "pay to save a day" and you will “more than earn it back.” Top management can communicate the economics of delay to the teams, and they know that saving TTM exerts more leverage on ROI than reducing development cost or product cost. Top management leads with "speed"; it makes schedule king. They use schedule compliance (managing the project using the schedule) to lead product teams to find and eliminate “wasted” time.

Making the schedule king means never slipping the schedule, of if they do, they find easy to recover in order to stay “on time”. Since there is so much wasted time in the total TTM cycle, the best organizations continually challenge the product development teams to find and remove “slack". Top management supports their actions, and together they prevent or recover slippage.



Making “schedule king” works because gestation (the period of time between first product concept and full team formation) and overruns (the period of time after the target market entry date for the product is missed) alone involve about 80% of the wasted time in new product development cycles, most of which is waiting.

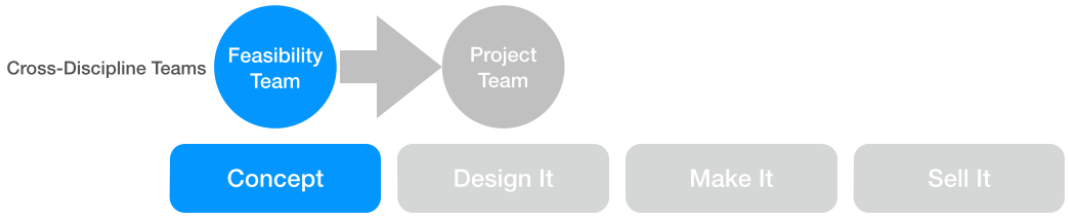
A high proportion of the development schedule is also waiting. Making the schedule king means permitting no "macro" schedule slippage; however the "micro" schedule may move in and out. If there is a trend of macro slips, the project is assessed and killed if a recovery plan cannot be put in place to pull in the schedule, thereby freeing resources for projects with a greater success potential. These decisions are made quickly and before the “point of no return” on the troubled projects. Mistakes are quickly converted to opportunities when discovered in time.

1.2 Create fast culture through close-in oversight of product roadmap

Summary

Best product organizations have a single point of ownership and responsibility for all product planning decisions. They dedicate a cross-discipline team to develop rapid feasibility studies, and use fast prototyping when needed to validate paper assumptions. The process is simple and everyone knows who owns each step of the process, including those who are "Recommenders," those that must "Agree," the "Input" providers, the "Decision-Maker," and who is "Responsible" for performing/executing decisions. The planning cycle is refreshed every month, it is fluid, and it is designed to make quick adjustments based on external conditions. It is driven by customer input and not internal "experts." There is a constant feedback mechanism to understand failure, perform failure analysis, and to make upstream corrections based on this learning.

Normal product organizations have no single person or entity who owns or is responsible for product planning. The responsibility is diffused throughout the organization between marketing and engineering functions, and across levels of business units and management hierarchy. The process (if defined) is complex and time consuming, requiring lots of man-hours to gather the data that few understand or even use in the decision-making process. In the end, no one is responsible, leaving final decisions up to the CEO who becomes the tie breaker. They usually lack sufficient information to make informed decisions and defaults to quick "gut" or intuitive decisions. The people that make the most noise or have the most political influence get their projects approved. People complain that the process is broken, but they secretly like the broken system because when things go bad, it is impossible to isolate blame or the causes of the failures since no single person or group is responsible.



Normal

The new product roadmap (or product portfolio) is shared between various functions in the organization. Marketing "owns" the product planning activity, but tends to operate unilaterally. Engineering's role is to vet the ideas generated by marketing, but rarely are conflicts resolved. For the most part, both organizations operate on their own and in an adversarial environment of negotiation and bargaining.

Marketing has learned to ask for more than they need, while Engineering tends to under-commit in order to preserve a buffer to account for the technically unknown. Operations and Finance are not involved in product planning decisions. Marketing tends to drive roadmap planning with little contact or input from key customers, because they understand customer needs, technology trends, and the competitive landscape better than their customers or competitors. Further, it is believed that customers are not a good source of future innovation ideas, since they don't know what they don't know, which is another reason they're input is somewhat discounted.

When Engineering is involved in planning, they do not push back on the product requirements produced by Marketing because they know that any resistance is a waste of time and in the end, they know that they (Engineering) control what features and functions that eventually make it into the product.

Marketing defines best-case requirements and they increase them to be safe to make the product easier to sell. All features, functions, and use-cases across all segments are expected to be developed (just to be safe), and little if any product functionality is prioritized to determine what is above and below the line of what is really needed. The cost-benefit value of each feature is rarely studied (yet alone understood), so all features are considered critical to the product's success. The conflict and misalignment between Engineering and Marketing is rarely resolved. Both operate with a "different set of books." Rarely does Marketing perform Voice-of-the-Customer (VOC) with the customer to understand what the customer's driving requirements are and why.

Product Planning is done on an annual basis through long drawn out data-gathering efforts by many people with few refreshes or changes between cycles, because the effort to get the data is so difficult and time consuming. The data tends to be dated/inaccurate due to the long gathering and analysis cycle.

Decision-making defaults to "exactly wrong" instead of "roughly right" in an effort to get the "right data," which everyone knows is based on assumptions and imprecise predictions of the future. If changes in market, customer, and/or competitive forces occur between cycles, little can be done to change the roadmap which is set in stone. Any changes have to wait.

Product planning decisions are eventually rolled up through the hierarchy to the executive staff level of the corporation. These are presented as "the plan" and then "signed-off" by senior management with little (if any) input due to the complexity of the presentations and mounds of supporting data. Executive management assumes that the people below them know what they are doing, the people below assume the CEO knows what they are doing.

Products are added, few are removed, no one checks to see if the Engineering organization has the skills and resources to complete the concurrent projects. This bottoms-up process is

complex and full of data steps in terms of MRDs, complex business cases, intricate technology roadmaps and competitive analysis. Hundreds of slides in PowerPoint are created consuming many hours of work. Top management is not able to dig into the details of the planning since the layers of management are not really connected.

Rarely is the product roadmap presented as an aggregate plan, but instead is discussed as individual products or technologies, specific market segments, or by business unit where nobody can see the macro-level implication of the decisions. It is expected that senior management understands the confusing and complex data presented, but most of the time they don't and don't say anything.

The sign-off process is complicated and it is hard to identify a single source/focus for decision-making, and therefore ownership for the product planning decisions are diffused throughout the organization. People complain that the process is complex and lacks lateral synchronization between the functions, yet no one can change it because there is no one person to go to, and they all know that the lack of a single point responsibility makes it easier to avoid criticism if/when product planning decisions result in failed products coming to market, e.g. late to market, low/no margin, not what customers wanted, etc.

The organization lacks a cross-functional product planning body and a single executive who is responsible for product planning across the company. At the end of the day, that responsibility rolls up to the CEO, but is quickly pushed back to the operating business unit executives, and they push it back down into the functions below them. The "hot potato" ownership game continues to cycle up and down the hierarchy.

Best

There is a single executive who owns the product planning decisions. Decision-making is lateral, not hierarchical. Data for making product planning decisions is developed iteratively by a cross-functional body comprised of senior people from Marketing, Engineering, Finance, and Operations. Plans are refreshed monthly and includes new competitive and technology trending information in real time. Plans are fluid and subject to change if the cost-benefit can prove that changes will add value.

Input for planning is primarily generated from customers and strategic partners, while internal subject matter experts carry less weight. The voice-of-the-customer (VOC) process is streamlined so it is easy to get continuous input from the market place.

The decision-making process is simplified. It uses a business planning discipline similar to a new venture start-up model, where the business case, marketing plan, cost-of-delay analysis, prioritized product requirements based on weighted customer requirements, and macro

schedules are developed by dedicated new product concept teams, primarily staffed by Engineering and Marketing leads.

Projects are funded in a manner similar to VC funding of start-ups. Teams compete for funding like start-ups compete in capital markets for equity capital. The winners get funded, the rest don't.

The decision-making process is not complex so all people in the organization understand it. The steps fit on one page and everyone understands who is responsible for making the final product planning decisions, who "Recommends" (gathers input, analyzes and makes proposals), who:

- Must "Agree" - individuals with veto power and yes-no control of recommendations — exercising the veto triggers a debate between those that must agree, and when this does not work, decisions are escalated to the Decision-Maker "D"
- Provides "Input" - the implementation people consulted on a decision
- "Decides" - the person with the "D" is the formal decision-maker who is ultimately accountable for the decisions and who has the authority to resolve any impasse in the decision-making process, and commit the organization to action
- Should "Perform" once the decisions are made - these are the people that are responsible for execution

There is a clear feedback loop to assess failures, so the causes are understood and preventative actions put in place on products earlier in the development life cycle. Failure is seen as a learning opportunity rather than a way to find blame.

Differences between what is wanted (customer driven needs) and what can be delivered in the desired time frame are reconciled *before* the project is funded. If these differences cannot be reconciled, the project is not funded. The resulting product requirements are prioritized based on customer's needs, and their value is clearly understood.

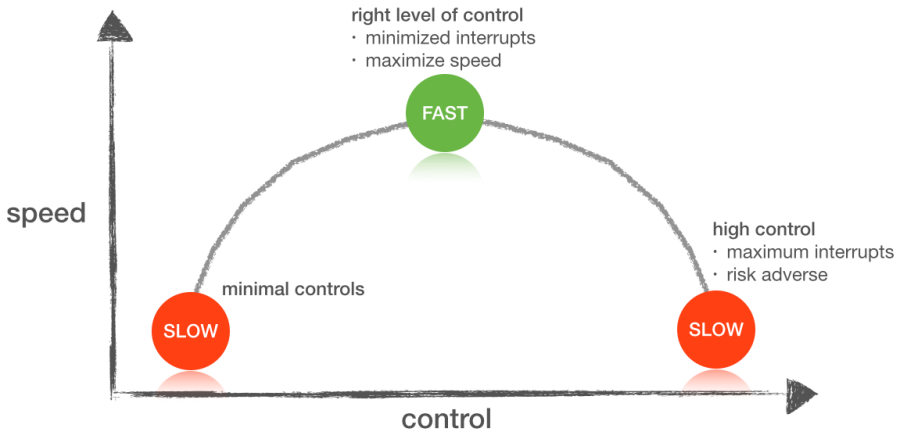
Product planning is continually refreshed and the dynamic roadmap considers both technology trends and product evolution/maturity. Understanding technology directions and product planning are not bifurcated between a Research group and a Development group, but integrated into one iterative system. They understand that one affects the other and they carefully study these interactions to help them predict future product possibilities.

1.3 Systematically dismantle slow structures

Summary

Best product organizations default to speed over control. They know that the cost of delay of a product far exceeds any additional OPEX costs associated with acceleration. Management systems are designed with this in mind; they identify all the interrupts (lack of resources, lack of timely resources, lack of skills, lack of funding, lack of information, constant strategy changes, etc.) that a development team might experience during the project, and they systematically eliminate them.

Normal product organization's dominant behavior is cost control. Decision-making, authority to spend and project flow is closely governed by heavy phase/gate life cycles. Much time and effort is expended to feed these systems, yet they provide little direct value to the team who is forced to use them. In an effort to control costs, most projects are slowed down, but when it is realized they are late, more money than was budgeted is thrown at them to recover, at which point it is generally too late because there was no early warning system in place. No effort is made to identify the source of systemic interrupts to the teams since these are seen as prudent management practices and "value-added" controls which minimize excessive spending. Further, removing the interrupts would eliminate a perfect source of excuses that people use when they are late.



Normal

The default behavior is to control the teams, especially their costs. Onerous Phase/Gate development methodologies are institutionalized to control spending. They are primarily designed to flow information vertically up and down the hierarchy and rarely do these control structures help with the lateral flow of information between functions. The logic is that development projects will be stopped at each gate and funding will not be authorized for the next phase of development until certain criteria is fulfilled. The criteria and requirements are published in lengthy documents that only a few people read and even fewer pay any attention to.

Elaborate PMOs (Program Management Organizations) are formed to be the gate keepers of the gate check lists. Thousands of man-hours are spent feeding information into the system. The more sophisticated companies implement multi-million dollar PLM (Product Line Management) systems to document and control every aspect of product development. Most of the time, the smart engineers find ways to fake out the system or find ways to work-around it to get through the phase gate and to avoid accountability. Top management feels better because on the surface, product development appear more organized and "automated."

For the most part, the phases are not followed rigorously, nor are gates really gates since teams can easily get wavers ("exceptions") to pass around or get through the gates. The wavers are rarely reconciled and there is no follow-up system to check that these waved requirements are ever met.

Instead of limiting the number of active projects in the pipeline, these control systems end up pushing more and more projects into the system without considering the impact of already overloaded resources. When there are time-to-market problems, executives add more layers of sign-off and approval to further "control" the "out-of-control" spending caused by delayed projects, and accelerating their burn rate during the slipped phase of projects where resources are thrown at the project to get it back on schedule. No one at the top sees the cumulative impact of these slow structures except each of the development teams that run into them every day while they try to get their work done.

Best

There is a conscious effort to minimize control structures. Fast organizations default to speed over control. They know the cost of delay (lost profit caused by missing market windows) far exceeds any additional OPEX costs associated with acceleration or relaxed controls. Management systems are designed with this in mind.

An effort is made to identify all the interrupts that a development might experience during the project (lack of timely resources, lack of skills, lack of funding, lack of information, constant strategy changes, etc.). These interrupts are then eliminated, both systemic recurring interrupts as well as situational interrupts that are specific to a given project. They put in place a process to identify these interrupts and continually eliminate them.

The emphasis is on speed and not on control, so approval processes are streamlined and levels of sign off approvals are raised. This permits lower level managers to provision their projects. They have learned that faster sign off and higher levels of spending approval result in faster decisions and end up being more cost effective in the end, since the authority to spend is pushed further down in the organization. People are more accountable and actually spend less.

Reporting is by exception. Projects that are within 25% of the target time and budget goals are left alone. The focus is on those exceptions that have deviated significantly from the norm.

Business Process Management (BPM) is used to map business processes within and across functions. Once mapped, inefficient steps are eliminated. BPM is horizontally focused to consider end-to-end processes such as product development life cycles. Gates are removed and phases are layered to enable the projects to go faster. They know that gates are never followed anyway, so they use the old gates as check points to assess maturity of the technology being developed, but not as a yes/no or a start/stop constraint.

Most controls are pushed down to teams to own rather than pushed up to top management to control. By pushing down the control systems to the teams (e.g. spending), they become more responsible and empowered. They spend less and go faster.

Since control systems are designed to make sure people spend less and go faster, the counterintuitive logic of fast organizations is to use lightweight structures to drive decision-making down to the project teams and then hold them accountable for their actions. It is counterintuitive because more freedom implies less control which, in fact, results if faster development because the teams control themselves.

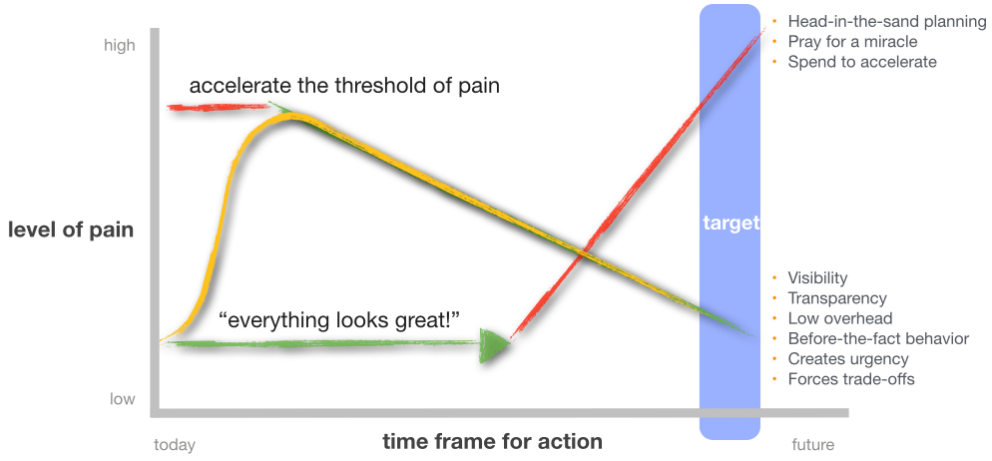
Why are start-ups with fewer resources faster than established companies? Because they lack structure designed to control and govern speed. Large fast organizations achieve the best of both worlds; they have the economy of size and the efficiency of being small because they treat each team like an internally funded start-up with fast management structures, low overhead control systems, and authority to decide and act. In addition, like a start-up, these teams are accountable to the shareholders to produce something on time that meets customer/market requirements.

1.4 Accelerate the threshold of pain; reward early warning behavior

Summary

Best product organizations force the "threshold of pain" to the front-end of development projects. They build management systems that encourage people to identify potential future failures early. They reward them for spotting problems and fixing them before they blow up. The culture is based on failure analysis and learning through this understanding. Teams live by the mantra *"The sooner problems are found, the more time we have to fix them."* Transparency is designed into the way teams are managed. Transparency means honesty, honesty and early warning is encouraged and rewarded.

Normal product organizations, push the pain later, because pushing pain forward is seen as a career limiting exercise reserved for non-team players. Identifying problems that have yet to happen is negative and can become self-fulfilling. Positive energy and aggressive predictions are the key to future success. Set the bar high and people will rise to the challenge. Failure is to to be avoided at all costs, but when things eventually go "off the rails", it is easy to hide from accountability in the details of the organizational machination and complex technology churning. Learning from failure is discouraged because failure means someone "screwed up" and will get "beaten up."



Normal

Development projects are initiated with great hope and optimism. All failures in the past are put aside because this time it will be different because *"we have learned from our mistakes."* Since the atmosphere is positive and buoyant, any dissenting voices are quietly silenced by peer pressure to conform to the prevailing upbeat attitude of top management. People that bring up potential future problems are encouraged to be better "team players" and adopt a "can-do"

sprit. They are told that negative ideas become self-fulfilling, so it is best to see the glass half full and get on with the difficult job at hand.

When potential future problems are spotted well in advance, these are ignored because it is viewed as a planning failure to have not seen these at the beginning. Recognizing the potential problem could force the management team to make difficult trade-off decisions sooner than they really want to.

It is explained to people that these potential problems may not happen and that if everyone was more positive, perhaps they could cause them not to happen, so there is no need to be forced into difficult choices now. Nothing is done. Eventually, the problems that were identified surface late in the development program when there is little or no time to recover. These have now grown into bigger problems, requiring more time and more money to solve, and the schedule slips. No one is held accountable for the failures because they were "unseen" and "unanticipated", and if people had performed as planned they would not have happened in the first place. People start pointing fingers as to who's "fault" it was, trying to locate the blame.

Usually the technical problems are sufficiently complex they are easy to hide behind. Everyone blames the difficulties of trying to developed advanced bleeding edge technology because it hasn't been done before, etc.

The culture does not encourage learning, because this means finding the root causes of failure, so that actions can be taken to mitigate future failures. Failure is seen as an unforeseen event that is out of anyone's individual control. "We are all victims of aggressive schedules and have to accept the problems associated with being on the cutting edge of technology." In addition, learning means learning cycles, and each learning cycle costs money. To save money, the team crosses their fingers and hopes to get it right first time.

Best

The "threshold of pain" is forced forward to the front-end of development. There are systems built in to the development methodology that encourage people to identify potential future failures. When these future problems are identified in advance, they are rewarded. The rewards encourage people to look for more potential problems.

Learning through failure analysis is part of the culture. Like in any engineering innovation, failure is seen as the fastest way to learn. People are encouraged to "fail faster." Accelerated failure/learning is how new technology gets to market faster. Identifying potential future failures and taking actions to mitigate this failure is encouraged by top management. The R&D budget supports the cost of multiple learning cycles.

The culture and management metrics are set up to push the pain forward so it can be dealt with early, while there is still time to fix the problems or take mitigative steps to reduce the failure's impact on the schedule. This positive attitude towards finding future problems sooner than later causes a greater level of transparency and openness in the culture. People are not punished for failure, but rather rewarded for finding and fixing it before it causes too much damage. The learning cycles required are built in to the schedule so it is clear

This mindset creates an open and honest product development environment in which realistic information is shared. This realism and early-warning behavior improves decision-making because teams and top management have much better information with which to make those decisions sooner.

This "learning culture" extends to each level of management, so people are more willing to take risks because the fear of failure is minimized if they are able to identify problems sooner and correct them. In the end, higher risk usually means faster development results. Risk, failure/learn cycles, and early warning are designed into the management system in fast companies.

1.5 Fast decision-making system

Summary

Best product organizations promote and work in an environment where decisions are made quickly. They have an overall understanding of the business direction and technology policy. Fast decision-making procedures or forums are available that will provide fast guidance when issues surface that may impact the team's schedule. The concept of "defaulting to action" is the norm.

Normal product organizations - critical decisions affecting the product team's schedule need approval by upper management. Teams lack an overall understanding of the business direction and technology policy. They don't proactively seek a "default to action" operating mode and assume it would be risky, so the default behavior is to wait for decisions or generate more data, or do both.

Normal

Project gestation (the period between idea and team full core team formation) often takes as long as the development itself. Lack of clarity in business direction and technology policy slows down product positioning. Senior management or staff respond slowly to product team requests for direction or approval. The development budget is over committed by a factor of two or three times, but this is not visible. The lack of limited resources become a reason for delay and slow action.

Best

The product team accepts responsibility for controlling its own destiny. The product team members consider it their task to be responsible for business development ("that's the way the market looks at it and so do they"); product positioning means positioning a complete business relative to the competition and relative to the company's portfolio of product businesses; the team emphasizes overall business success from the start (e.g. Break-Even-Time and/or Return-on-Investment).

Development acts like a start-up and takes the initiative to get a coherent and compelling business concept; gets executive commitment; doesn't wait; actively keeps senior management support alive to keep the strategy consensus together and avoid re-plans.

Executives recognize that TTM delays are primarily caused by product specification changes and slow decision-making by the host organization. Top management contributes most to FTTM by minimizing specification changes while giving fast guidance on business direction, technology and budget approval.

The product team is released from the time constraints of all the host organization's periodic schedules. Business objectives should guide, but never delay products. The team gets the 80-20-business guidance it needs, when it needs it. Fast decision-making procedures are implemented that authorize fast starts and provide fast guidance.

Best teams/organizations know the gestation period offers an inexpensive opportunity to cut TTM by as much as 30-50%. One or more product champions are assigned at product conception.

A fast forum for fast decision-making process is implemented early. They make sure it's always on-line with a short turnaround time (typically <24 hours). If the decision-making process remains informal, product champions default to action after consulting the key players.

Getting the product team the necessary guidance or issue resolution via strategic by-passes (even if imperfect) decisions are expedited. As a rule, best teams know that if a product will be affected by changes in business objectives, then TTM should be accelerated rather than changing its specifications. A percentage of development budget funds are allocated to the pursuit of unplanned opportunities arising during the year.

Best teams/organizations make short but sufficiently clear versions of the business plan (values, vision, mission, strategy, goals, key success factors and key processes) and technology policy available to all. It should provide the quick 80-20 answers to the following questions about business direction:

- What are the market segments for current and future business?
- Who are the driving customers for each market segment?
- Who is our competition in each market segment?
- Will we compete in them by first-to-market, second to market or niche product positioning?
- How aggressively should low cost products be pursued?
- Which portion of the follow-on system makes up the business?

Provide quick 80-20 answers to the following questions about technology policy:

Target product parameters

- What is the relative emphasis on product parameters (features, quality, cost, reliability, etc.) to achieve first-to-market, second-to-market or niche positioning?

Support technologies

- Is investment in related product and process technologies required to support the main product?

Technology leverage

- Is the product family structured to achieve technology leverage (number of lines, system threads, reusability, customer migration path, cannibalization intent)?

Advancement steps

- Will technology advancement be "many small steps" or ”a few big steps?"

Exclusivity

- What competitive barriers are intended based on mix of proprietary and open standards?

Sourcing

- Which subsystems, process technologies or tools will be out-sourced?

Facilities

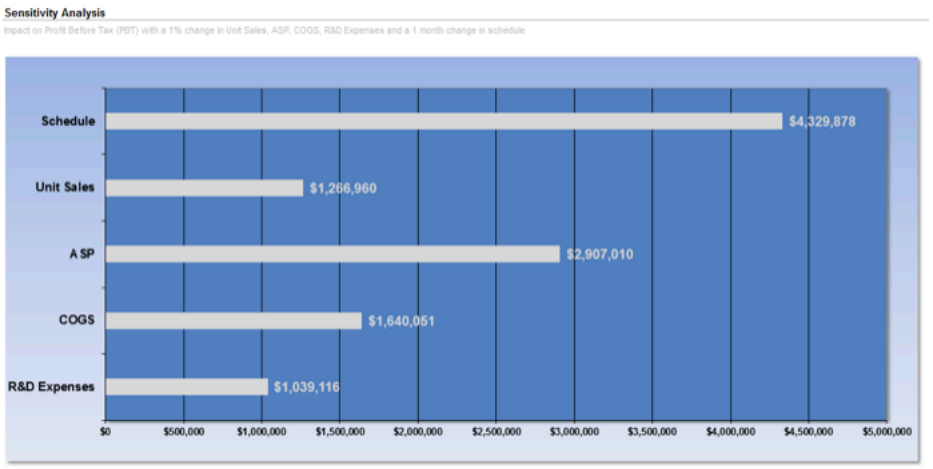
- What facilities will be used to develop product?

1.6 Cost-of-delay is understood at individual contributor level; used to drive urgency

Summary

Best product teams/organizations have calculated the "economics of delay" and have adopted the "pay to save a day" modus operandi. They use this to create a sense of urgency and to accelerate decision-making.

Normally, the true costs of delay have not been calculated or seriously pursued. No one owns this “cost” in the organizational hierarchy.



Cost of Delay Model	
Sales Change	Custom
Delay (months)	12

Cost of Delay	
Per Month	$47.5M
Per Day	$1.6M

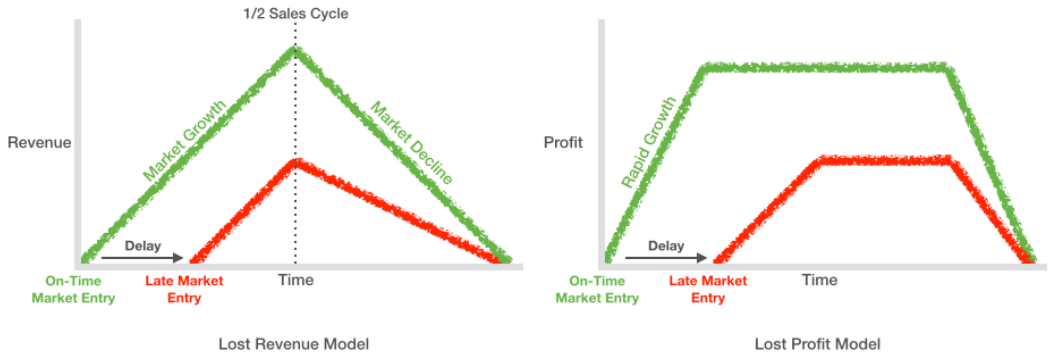
Example of a cost of delay model. Note that schedule is the most sensitive factor in this project’s profit projection. It was $1.6M in lost profit for every day this high volume consumer product was late.

Normal

The impact to delay in terms of “cost” to the company is not known by the development team. Since no one owns the “delay cost” in the functional organization, it is not identified and tracked. Time and money are not connected and are managed by two separate functional organizations. Financial decisions are made based on annualized functional department budgets without regard to the impact these decisions might have on product development cycle time (i.e. One additional person added to a project might accelerate the schedule by 90 days, but they are not hired because the function did not budget for the additional person at the beginning of the year: the lacking resource results in a $100k/day cost-of-delay to the project team, yet no one is accountable for this lost opportunity to save time). The annual new product development budget is set and "that's it." If the company does spend money for speed, they prefer to spend it on expensive productivity tools.

Best

Offer to "pay for a day" is the best way to make management support for the economics of delay visible. Although some money will be spent (to save a day), the offer to pay has more than an offsetting value in changing the mindset.



The best organizations calculate the cost of a "day of delay." They consider these variables:

- Lost margin
- Lost revenue
- Declining average selling price
- Extended run rate of development costs
- Lost market share
- Late entry into sales life
- Lost customers and markets
- For enabling technologies: loss of access to customer product cycles

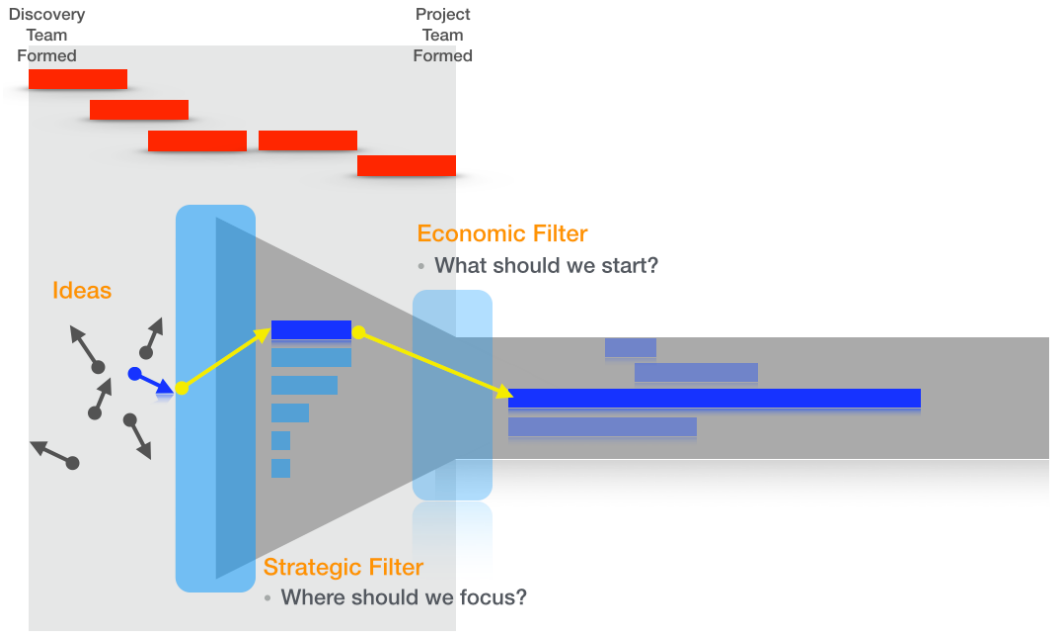
They use the cost-of-delay number to drive rapid decision-making and provisioning of their team with the resources they need to achieve their TTM target.

2.0 Host

2.1 Fuzzy-front-end is managed

Summary

Best product organizations manage the front end of development, from concept to team formation by assigning small multidisciplinary teams of engineers and marketing people to quickly determine the feasibility of new product ideas. They significantly reduce overall development time by managing the front end activity within a fixed time-frame. It is treated as a formal development phase and managed like the rest of the product development process with specific target dates to be met.



Normal product organizations let new ideas drift for long periods of time. Sometimes they are championed by marketing, sometimes by engineering, but rarely by both together. They lack the resources to do in-depth technical and business feasibility analysis, so the new ideas get cursory attention. The real work is left to the development team when it eventually gets formed (usually late and usually without enough of the right people) to do the discovery and investigation work that should have been done earlier. Problems arise later when the goals established in the fuzzy front-end are inherited by the formal core team. These goals were set without enough information, yet they never seem to change, including unrealistic expectations for product performance and project schedule.

Normal

The period of time between from when a new product concept is first discussed to when a formal and fully staffed team is assigned is called the fuzzy front-end of the product development process. It is called “fuzzy” because in most organizations it is an unmanaged/unowned process and there is little control. It happens when it happens.
In the normal environment, this is done exclusively by one or two marketing people and/or it is done by a skunkworks of engineers working in the back room on the “secret” project that is yet unfunded, but draws resources from other projects to conduct the feasibility studies.

The period of time between idea and team formation can be as long as the actual design, development, and manufacturing ramp phases. The reason is because it is unmanaged and usually under-staffed and/or under-funded. It is under-staffed because the resources requited are stuck on other projects that are late, hence they can’t roll-off. In some cases they are rolled-off, leaving the existing projects under-staffed.

The fuzzy front-end meanders along because it is under-funded and often under the radar. Management focus is on those projects that are well into their development life cycles and are experiencing problems. They lack a fully staffed and funded multidisciplinary team to rapidly study feasibility, both from a technical and business perspectives.

Best

Senior management recognizes that up to 100% of the total development time can be lost in the un-managed time from idea to team formation, so they assign a small multidisciplinary team to study feasibility of the idea in order to quickly understand what is required to productize the new idea.

They understand that the work necessary to turn a product concept into a product/project requires a cross-functional team of marketing and engineering to understand the customer requirements, market positioning, competitive/benchmark factors, determine technical feasibility and maybe even develop preliminary prototypes to validate the ideas/concept. If it involves new building block technology, then advanced research need to get involved.

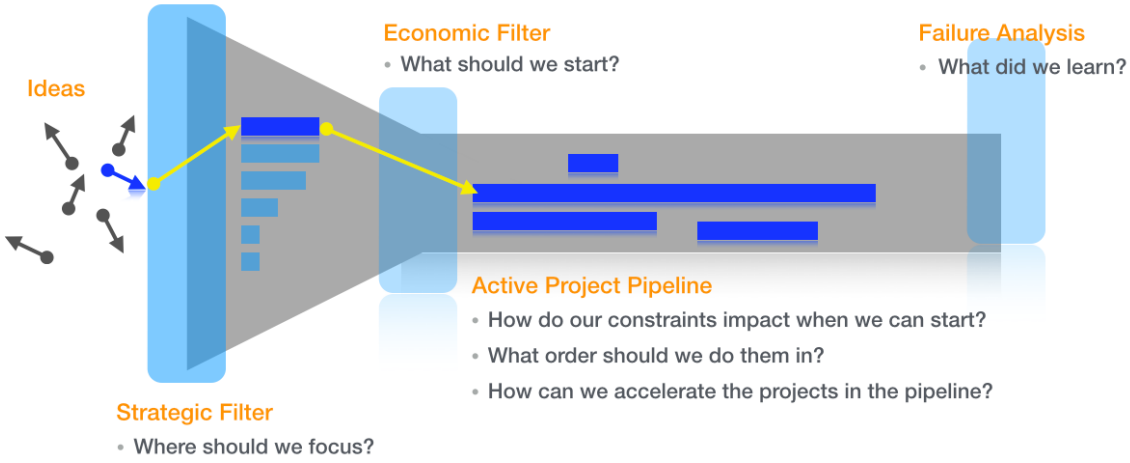
They set narrow windows of time for making these go-no/go decisions (e.g. <2 months). When this “gestation” period is factored into the overall development cycle time, a large acceleration can be achieved in total time to market when gestation time is reduced.

Instead of it being under the radar, these feasibility assessments get immediate attention and focus because they represent one of the major sources of new product projects.

2.2 Projects are prioritized based on business objectives

Summary

Best product organizations have a formal process for prioritizing new product development projects that are linked to the business planning and strategy development. The formal process quantifies the value of each project relative to the weighted business objectives. It is a managed consensus process where all stakeholders from engineering to marketing are involved, and are forced to reconcile what is wanted with what can be delivered given the constraints (resources, budget, risk, etc.)



Normal product organizations have few (if any) formal systems to prioritize new product development. The funding process is driven by business units with the most influence. Projects starts are informal, so funding is erratic in the early stages when stability is most needed. Projects never get killed, and new ones keep getting being added without regard to the impact they have on the current resource allocation.

Normal

There is no formal system for prioritizing new product development projects. The project ideas are generated by customers, marketing, and engineering. Marketing and engineering have different systems for selecting which projects they will fund/initiate. Marketing's is based on business planning; revenue and margin plans that must be achieved. Engineering's is based on what they can do with the available limited resources/skills. The high profile the customer, the more likely that project starts.

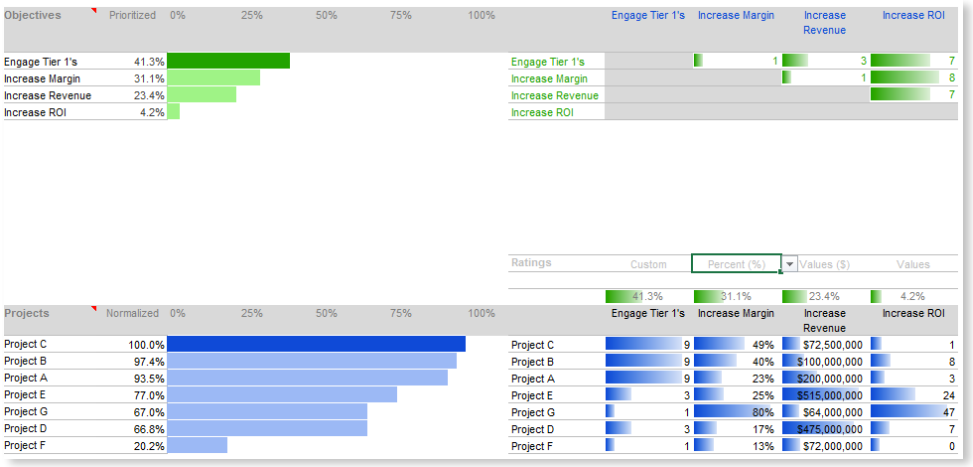
OPEX and CAPEX budget is derived from the business planning done by Marketing. The business planning and budgeting process encourages inflation of expectations and forecasts;

one group wants to show more revenue and higher margins, and the other uses those to justify increasing development budgets.

Failure occurs because the two planning systems are not balanced and aligned around the available resources (skills and quantity). The result is a pipeline that continues to expand to meet the business planning expectations, and the continued over-usage of the budget constrained resource pool. The two perspectives rarely get resolved until there is a crisis where people and budgets get cut across the board. There is no way for anyone to see which projects are more valuable than others, so they all get cut equally (but few ever get killed).

Best

A formal process is in place to model project priorities based on strategic business objectives and economic objectives. Strategy and economic metrics are clearly defined by the business units and weighted based on the objectives which contribute most to fulfilling the business strategy.



Each project is assessed against each business objective to determine which projects contribute most to each objective, which results in a prioritization of projects. It is a zero-sum ranking, so when one project increases in importance, others are decreased in importance. This forces real prioritization, and is used later to assess the impact of constraints (limited budgets, resources and risk) on project priorities.

The process is owned and managed by the product planning function, which is not part of either engineering or marketing. Engineering and Marketing are forced to reconcile what is wanted versus what can be delivered. They don't move forward until both sides reach agreement. They do this in a regular cadence; from once a month to once a quarter, depending on the market and company.

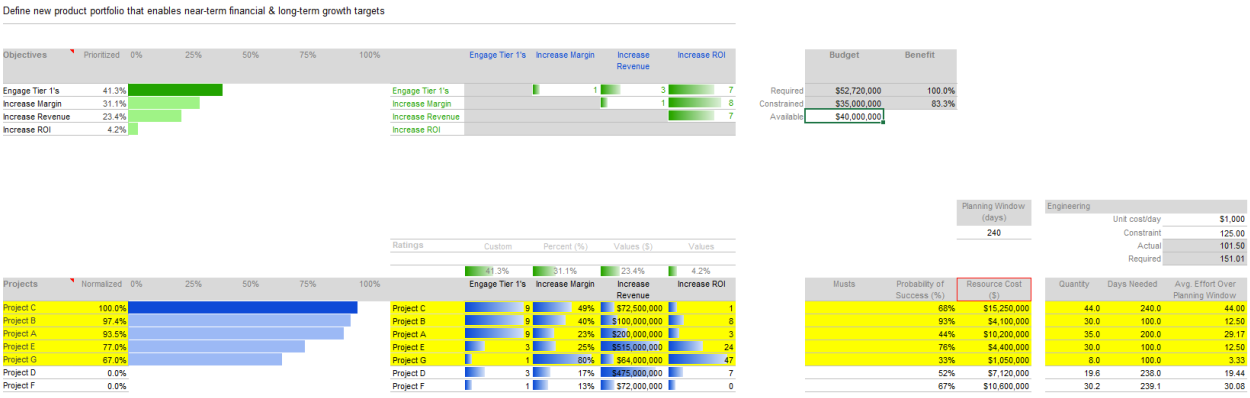
New ideas are pushed through the process to see how they rank against the approved project portfolio. This creates a formal “starts control” process that funds, reduces funds, or kills projects that fail to score high enough against the objectives. This regular portfolio “refresh” also considers changes in business strategy/direction, and how the existing portfolio is impacted.

2.3 Projects continuously aligned with resources and skills available

Summary

Best product organizations have a system to model resource constraints on current development projects, and the impact new projects might have on those projects. All stakeholders participate in the "forced" resource and skills alignment process on a regular refresh cadence.

Normal product organizations have no organized system for balancing resources. Systems that are in place are ad-hoc and are used to show the misalignment, but they don't offer solutions because they only consider one functional perspective. Such systems are cumbersome to use, unpadded infrequently and inaccurate. Meanwhile, more projects get added to the development pipeline because there is no way to see their impact on current projects. Attempts to highlight the problem are career limiting for the up and coming executives, so most are silent, toe the "can do" party line and hope they get promoted before they get caught.



Normal

In addition to no formal project prioritization system (see "Projects are prioritized based on business objectives"), these companies don't have any way to assess the impact new projects have on existing resources already deployed and committed to projects in the pipeline. New projects are added without regard to the impact they have on the existing approved projects. Functions are expected to "get creative" and become more efficient to handle the new projects.

The problems are compound every planning cycle as the existing projects in the pipeline get slower as new projects are added. The efficiencies of scale don't apply when it comes to thinking about people working on difficult technical innovations. When executives complain they don't have enough resources, they are told they are less efficient than their competitors who are able to generate more new products with fewer people.

Best

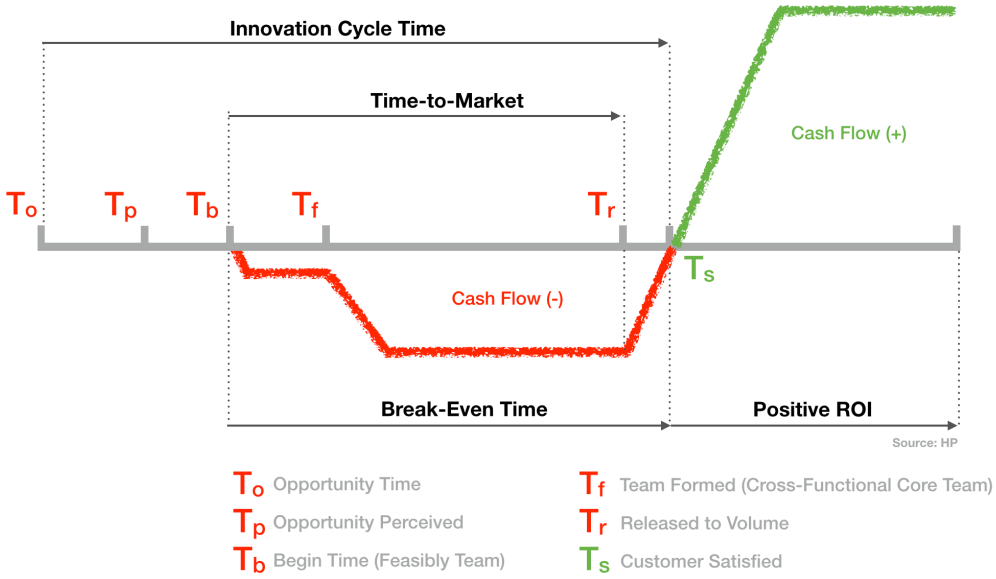
The formal prioritization system (see “Projects are prioritized based on business objectives”), is used as the basis for modeling the impact of constraints (such as budget and resources) on existing the existing project portfolio. When the constraints are applied to the prioritization model, they quickly learn that some of the lower ranked projects should not get funded.

They do a simple cost-benefit analysis to determine which projects get funded and which projects don’t, get killed or are delayed. They know that about 20% of the projects generate 80% of the benefit (based on the prioritized business objectives), so the process is designed to locate the high value 20% of the portfolio that must get maximum resources. The process is refreshed from once a month to once a quarter. The results are based on the multidisciplinary input by the key stakeholders.

2.4 New products are tracked to break-even

Summary

Best product organizations track new products from idea through to break-even. Development teams understand the impact of their work on the company's success/failure. Because they see a product through to break-even, they tend to make different decisions early on in the development cycle than they would if they were just taking the project to the manufacturing handoff point. Being able to make a yielding product cost effectively can change design decisions early when designers are conscious of future market success and product profitability.



Normal product organizations track project cycle-time to the end of development to where the engineering organization hands the product off to operations for them to manufacture it cost-effectively. The clock starts when the full team is formed (not even when a partial team is formed). No one tracks the pre-team feasibility phase. Engineers rarely see/correlate the economic impact of their work with the company's success/failure.

Normal

New product development projects are tracked by function, so marketing tracks the projects they start, engineering tracks the design and development activity, and over the wall it goes to operations (manufacturing) to track the manufacturing ramp up to reach the yield and cost targets. At some point sales starts to track the market performance of the new product and somewhere, way out into the future, a finance person might run an analysis to determine if the project made or lost money.

The engineering team that does the design hardly ever knows if their work makes or loses money for the company, since this is not their concern, as it is someone else's job to worry about much later in time. Once they have finished their job, they move on. Since each function/ silo is partitioned from seeing the complete life cycle, none of them ever see the end result of their work.

They don't identify with the project/product success, rather their focus is exclusively on their function's achievement of their objectives, usually based on cost control metrics.

Best

The fastest teams/environments turn on the clock when the new product idea is first discussed. They measure cycle time (from idea to team formation), they measure design and development time and they measure the transition through the manufacturing ramp. This follows with a measurement of the time it takes to get from high volume through to market distribution and eventually to the time when the project breaks even.

This is not tracked by the functions, but the core team that owns the development project through to the beak-even point. They follow the product all the way to the point that it starts to make money for the company. This creates a sense of ownership, purpose and a business, since they can experience the impact their work has on the company.

2.5 Kill projects that fail to meet objectives; early

Summary

Best product organizations kill projects that fail to meet the business objectives. The decision-making process is agnostic towards position or expert power, since it is based on agreed data and decision modeling techniques.

Normal product organizations rarely kill projects.

	Max Budget	$52,719,333.80										
rojects	Benefit	12.8%	18.7%	31.5%	46.2%	59.0%	64.2%	77.0%	78.2%	83.3%	96.1%	100.0%
	Cost	$1,050,000	$4,100,000	$5,150,000	$9,550,000	$16,670,000	$19,750,000	$26,870,000	$31,920,000	$35,000,000	$42,120,000	$52,720,000
	% of Max Cost	2.0%	7.8%	9.8%	18.1%	31.6%	37.5%	51.0%	60.5%	66.4%	79.9%	100.0%
Project C									Funded	Funded	Funded	Funded
Project B			Funded	Funded	Funded	Funded	Funded	Funded	Funded	Funded	Funded	Funded
Project A							Funded	Funded		Funded	Funded	Funded
Project E					Funded	Funded	Funded	Funded	Funded	Funded	Funded	Funded
Project G		Funded		Funded	Funded	Funded	Funded	Funded	Funded	Funded	Funded	Funded
Project D						Funded		Funded	Funded		Funded	Funded
Project F												Funded

Normal

Projects rarely/never get killed. When they do, they reemerge as new projects with a different name. The business objectives are vague or they change, and there is no way to assess how well each project contributes to those objectives. Most projects that get approved and never killed are those projects put forth by senior management or people inside the organization with expert power. Since that have missed their economic value are kept on life support, since no one wants to admit that they should never have been started or should have been killed a while back before sucking up valuable resources.

Best

Projects are assessed (at least) on a quarterly basis to determine how they score against the business objectives, especially if those objectives have changed. Projects that don't meet a certain level (score) are challenged. Projects that have missed their viable economic window are also challenged, with the aim to clean out the deadwood projects and free up resources for

projects that can most contribute to meeting the business goals. It is a data-based decision, not an emotional-based decision.

2.6 Fast development framework

Summary

Best product organizations - the development life cycle is owned by a single person/entity. Their job is to continually redesign it so it makes development projects move faster. The framework considers the problem from end-to-end, i.e. as one integrated system, starting from product concept through to the product end of life, rather than as a series of organizational deliverables. Systemic interrupts are continually identified and removed so teams get what they need when they need it. These frameworks are simple, one page explanations of the process. Because they are macro and simple, people understand it, which means they use it. Details are left up to the core team to fill in on their specific project. All projects look the same at the macro level, but at the micro level they are unique., i.e. there is flexibility within the framework so the team can adapt as necessary vs. giving the team a rigid of working that never works for all types of project.

Normal product organizations - development frameworks (product life cycles) are slow, complex, and cumbersome processes that are designed to control development spending and to ensure decision-making is concentrated in the top levels of the functional hierarchy. The concept of design for reuse and platforms, spinning rapid derivative products are discussed but not implemented, because it implies an overall orchestration of the creation process which does not exist. Prior attempts to improve the development framework have failed, since the people that really need to do it are the people that are consumed by the daily grind of development problems. Product life cycles are rarely redesigned, but when they are, they are done by people on the sidelines, and they are therefore discounted and never get implemented.

Normal

Product development is structured functionally. There are many owners of the pieces of the development process, and there is no single overall owner of the product from end-to-end as it flows through the various functions that touch it during the development life cycle. The process is complex and few understand it, defaulting to just trying understand their contribution to the process. The development steps are rarely challenged and are hardly ever re-engineered to be more efficient. They tend to be additive systems that become bloated and overbearing, which is why few take the time to understand them. Few people follow the steps because they add no value and just slow down the development work, or they assign NPI people (overhead) to feed information into the tracking systems that few use.

The concept of “design for speed” is not understood, and rather than developing reusable platforms, companies default to reinvent the wheel on every project, because they know it is impossible to reach a consensus. They will either tilt towards platforms that have everything or have no platform/derivatives at all, with each project becoming a new invention each time.

Since no single person/entity owns the overall product development life cycle (since it cuts across all the functions) it becomes impossible to optimize the product design process for speed. The process was designed to control spending and decision-making, and was never conceived as a way to make product development go faster. To fix it means "blowing it up." No one wants to do this, so only incremental changes are made, resulting in little impact.

Best

The team emphasizes overall business success from the start, i.e. through to Break-Even and/or Return-on-Investment. The development life cycle has been optimized for speed. It is owned and managed by a product management function which cuts across the functions, reporting to the executive staff level. The product development framework owner's job is to find ways to reduce friction (i.e. interrupts) and continually improve the process to remove waste and inefficient steps. It considers the complete life cycle, from first concept to end of life.

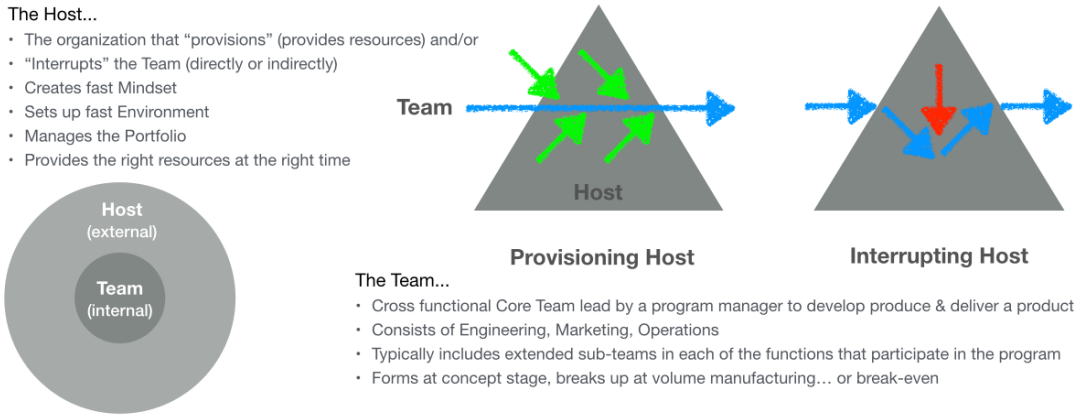
The team knows that up to 100% of the overrun time can be lost during the "fuzzy-front-end" of development, i.e. the time from idea generation to full core team formation/project funding approved, so they carefully manage the front-end feasibility analysis in order to make faster go-no/go decisions about the project. They design speed into the product through the use of reusable parts and common platform architecture where up to 80% of the parts are reused and only 20% is changed for derivative platform spinoffs. Design and speed are seen as one thing, not two separate things. Every effort is made to design so the development effort is accelerated, i.e. "design for speed" rather than design for the most elegant technical solution.

2.7 Provision teams, eliminate interrupts

Summary

Best product teams/organizations promote and create mechanisms to provision the product teams quickly with resources, information and fast guidance where required. The host organization works with the product teams to minimize the schedule impediments (interrupts).

Normal product teams/organizations *"*default to delay" much of the time because they are waiting for the hierarchy to decide/act.



Normal

The design of the development organization structure by the host organization is ad hoc, slow and considers few alternatives beyond what has been done before. New product development work is divided between the functions responsible for the work. The project is coordinated by the functional hierarchy within and between functional groups. Breakdowns occur at the "hand off" points between functions because walls have been built up between the functions. The functions default to "throw it over the wall" method of working with little communication or interaction between functions. Uncertainty of the cross-functional tasks associated with development work forces those doing the work to coordinate the work while they are doing it. This inevitably overloads the vertical information capacity of the hierarchy and delay is inevitable. The entire development organization defaults to delay much of the time because it is waiting for the hierarchy to decide/act. Consequently, the host is slow to provision the development organization but quick to interrupt it.

The development organization is deeply "embedded" into the host and each individual is deeply "embedded" into several projects. The over-head of context shifting and frequent interrupts are high and resources are provided to the team when they are available. They are typically not sufficient, with the wrong skills, and provided late due to the need to have a limited number of

skilled people continue working on the other projects that have higher priority because they are closer to finishing and are late.

In these "slow" environments, team members are typically more allied with their functional position within the host than the new product development team(s), of which they are a part of. They view team work as secondary to the work they are directed to do within the function.

There are adversarial relationships between New Product Development (NPD) teams and the host:

The Team… *"They've never killed a project."*
The Host… *"We've never had a product delivered on time yet."*
The Team… *"Host has to understand that resources aren't infinite."*

Teams takes pride in being able to work around the host and view the host as the major impediment to speed. The underlying assumption is the Host views the team as a "means for between-function hand-offs" (versus the cross-functional owner). The Host controls/limits the team's ability to decide/act on issues central to the team's functioning and/or ability to stay on or beat schedule.

The Host owns and controls the budget, it stretches people resources too thin between projects and/or sustaining work and limits or provides untimely information, decisions, resources, customer information, etc. The Host also limits equipment, tools, facilities (unavailable/shared/dated).

Examples of typical host interrupts that we have seen in our work includes:

- 14 signatures required and 4 months to approve 2 critical engineering requisitions
- Critical project resources redirected away from the team (unknowingly)
- Periodic audits (as a result of low confidence) and frequent requests for information (need to know) virtually shut down productive project work
- Artificial gates (checkpoints) and organizational reviews imposed to maintain overall control of daily decision-making (systematic interrupt)
- Slow to respond on critical project/product decisions
- Team was limited in it's decision-making scope
- Frequent changes to the product strategy and re-prioritization of the organization's "key" projects
- Failure to provide access to test software and the training to use it effectively
- Failure to provide enough test equipment (2 shared between multiple projects)
- Marketing and Sales (the Host) limited the team's direct contact with customer
- Throughout the project, requirements were vague and unclear

Best

It is recognized that delay caused by the host is a primary cause of TTM delay. Cross-functional core teams are formed on time, dedicated, virtually collocated, autonomous and empowered, and lead by heavyweight (vs. lightweight) program managers.

Resources are provided on time and with the right skills based on what the team needs, not what is available. If this can't be done, then the program is not started. Unless the development organization is designed to be a fast organization, the usual results will occur: engineers can be found to spend less than 25% of their time doing engineering work and 75% is spent on interrupts of various kinds. Likewise, the product team itself spends a huge portion of its time being interrupted or delayed by the host organization.

In relation to the host organization, the overall design criterion for the fast organization is to optimize provisioning and minimize interrupts between the team and the host organization. Provisioning includes providing the team resources and information (direction, resolution, etc.) in a timely manner. There are two ways to design around the problem of overloading the hierarchy. The best organizations use a combination of all three: use self-contained teams and establish lateral relations (in other words they make the development team as autonomous as possible), design faster communication systems to the top management, and co-locate the team members for direct contact (or virtually collocate).

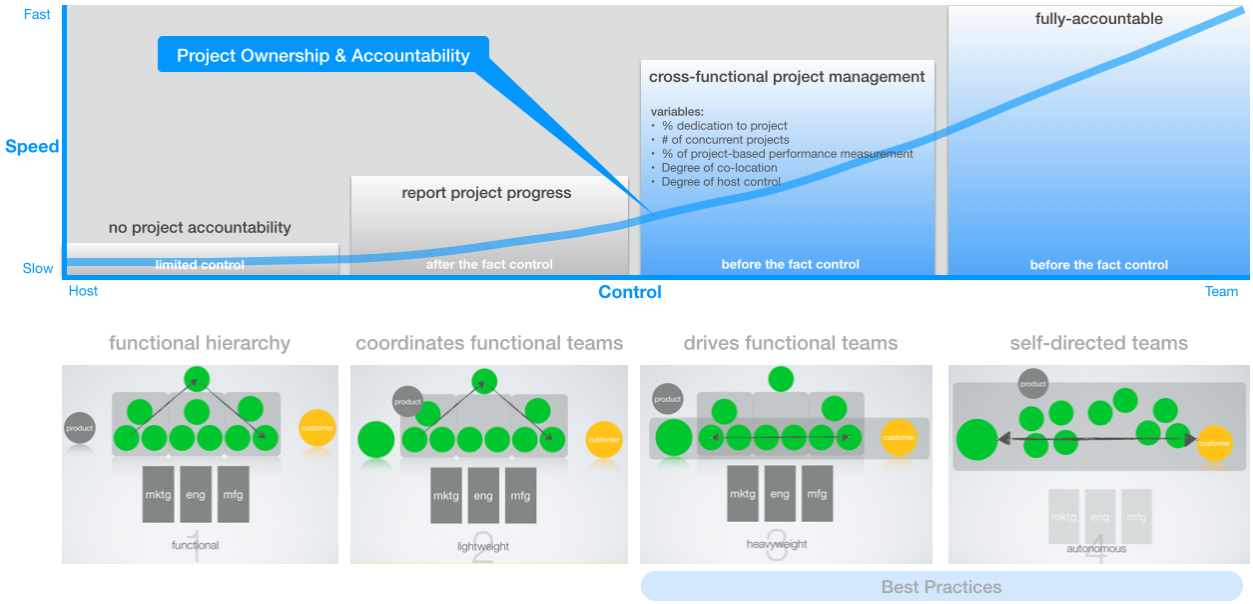
To get such an organization in place, top management holds the teams accountable to the schedule, and reviews it in a rapid decision-making forum. The Host gives true "ownership" of the product (like a start-up) to the core team. They provide sufficient & timely information (market, financial, supplier, inter-company, etc.) to the teams. They provide sufficient budget for the team for the duration of the project (based on what is needed to do the job right vs. what is available or negotiated). They ensure the right skills and number of people are allocated to the project as they become needed on the project. They identify and provide sufficient tools & technologies to teams such that these are not bottlenecks to development and finally, they really do this and not just say they will do it.

The Host insures that appropriate facilities for the team (co-located facility, "war room", etc.) are made available when team needs them. This does not require "begging" on the part of the team, but rather the facilitization of the project is built into the fast design and development flow. Finally, they delegate sufficient authority to decide and act in order to stay ahead of schedule. Systemic interrupts are identified and eliminated. This is a continuous improvement process to find every thing the company does to generate friction, i.e. felt by teams when trying to move fast, then fixes them, from increasing sign-off levels to expanding the scope of decision-making.

2.8 Cross-functional core teams (heavyweight & dedicated)

Summary

Best product organizations form heavy-weight teams early, i.e. during the project feasibility phase, on critical projects. They are truly cross functional, composed of members from engineering, marketing, operations, and finance. They are dedicated 100% to a project and rarely multiplexed between other projects. They are lead by heavy-weight Engineering Project Managers (EPMs) who have general subject matter expertise and also knowledge of the customer and the market. The core teams are run like start-ups and have all the necessary skills to run like an independent business. The host structure within which they operate is designed to provision the team with what they need to be successful. The power is shifted from the functions to the team. Both are critical to the success of the project, but the team is empowered with making a new product which means that they share in the rewards of the product success, but are also responsible for a product's failure.



Normal product organizations teams are "cross-functional" in name only. They are comprised of many part-time, over committed and under resourced functional representatives that take direction and are loyal to their functional hierarchy, not to the team/product. Because there are many functions on the team, the core team becomes bloated with sometimes >20 representatives. They politely attend project meetings, however the actual work on the project is done back in their functions (not in core team meetings). Since core team members are on ~10 teams at the same time, they miss a lot of meetings and rely on electronic communications, which is rarely up to date, often inaccurate or never read because of the hundreds of e-mails that get delivered every day. The teams are lightweight and used for coordinating between the

functions rather than driving laterally through the organization. Nobody is responsible for the integration of functional deliverables that come together to make a product deliverable. Today's cross-functional teams are really just new forms of old hierarchical departmental organization, and as such tend to be good at moving information vertically, and bad at managing horizontally.

Normal

There are many cross-functional teams. One engineer could be on 10-15 at one time. They participate on a part-time volunteer basis, and because they are on so many projects at one time and spend little time in team meetings, they have little if any connection with the team's mission.

Most of the engineering team's time is consumed with technical meetings within their function. They receive performance reviews through their functional hierarchy and naturally focus on satisfying those requirements first. If there it anything left over at the end of the day, sometimes the teams get it. Team members complain of motivational problems caused by not being able to spend sufficient time on any one thing to do it right. Work product quality and motivation suffers.

When asked, "What do you work on?" the response is usually, "I work in X Group and do Y activity." Hardly ever do these core team members identify or associate themselves with a specific product by name. Their focus is exclusively functional.

The core team is formed late in the development cycle. Members join the team at different points in time, making it difficult to develop any team culture or focus. Core team members join the team late due to the delay of a previous project and/or members being assigned to too many teams at one time.

Leadership of the core team is unclear. There is a program manager, but their role is to coordinate meetings, document submittals and to give the "status" presentations when asked by top management. For the most part, it is an unmanaged team coming together regularly for meetings. However, all the members know that the "real work" is done back in their functional sub-teams. The core team is supposed to be a forum to share information and coordinate the hand-offs, but eventually it becomes more of a formality because it is required by the development process procedures. Teams resort to large multi-day all hands meetings to try and get information moving around the team when the core team fails to do this job during its weekly sessions. Even more time is sucked away from developers with these big meetings.

Since there is no strong/authorized leader, the engineering elements of the project are broken down into independent work efforts with little coordination between the pieces. Core teams are bloated with 15-30 people because every engineering element of the product needs to be

represented, and because they lack a technical hierarchy that leads back to a single person responsible for each major product deliverable.

Who is the “Chief Engineer” in charge of making the difficult technical tradeoff decisions that are essential for timely systems development? These people are hard if not impossible to find. The decisions are spread over multiple people so no decisions ever get made.

The lack of technical hierarchy on the team leads to sub-optimization, expanded task durations and downstream handoff problems. The core team members are not dedicated to a single project; most of the time they are on >10 at the same time. It gets worse with individual contributors who are multiplexed across >15 projects at one time.

No one has enough time to do a good job on any of the projects because they are overloaded. This is why the focus is inside the functions and why team members expect the program manager to manage the “white space,” which everyone knows is impossible on complex technology projects today.

The core team and extended teams are not co-located, even when on the same site. They rarely see each other and rely on automated systems to share data or communicate online. During online working sessions, those on the call are doing other things (replying to other e-mails related to the other projects they are on in order to keep up), while they are trying to pay attention to the speaker.

The lack of face-time results in poor communication within and outside of the team. This is even true with people that work on projects at the same site, but don’t sit near each other. Sixty (60) meters within a site is about the distance when physical contact drops off. People default to electronic tools, even when they sit next to each other. They are often used to insulate people from outside contact, which is the opposite of what they were designed to do. If you want a decision to stall, put it into an email and copy 30 people.

Finally, the core and extended teams take direction from their functional hierarchy. They pay lip service to project managers and are polite in team meetings, but they know that the real work is done back in the functions. The functions are protective of information and of making commitments. They add buffer to take into account the organizational inefficiencies. They know the incoming work will be late, so they add buffer to account for the delay and to protect themselves from blame when the project is late.

When these duration buffers are accumulated, the schedule becomes bloated and inflated. The team are then forced to reduce durations. Few really care about the overall program schedule anyway, since they are only being measured on the performance of their function and their own

technical metrics which they are held accountable for by the more powerful, functional hierarchy above them. Career advancement happens when this power structure is satisfied.

The notion of an "autonomous" team structure is seen as wishful thinking and impossible, given the current limited resources and large number of committed projects in the pipeline. The teams are lightweight at best and only serve a minor coordinating role. Top management knows this and defaults to getting status reports from their technical subordinates through normal staff meetings and periodic project meetings that only involve the people inside a given function on a specific project. Everyone is doing this at the same time in their functions resulting in very little flows laterally across the functions. Development is slowed and the schedule slips.

Best

Executives recognize that the success of the business is based on timely release of new products (since these generate the highest margin) and in the technology. They also generate the fastest growth. They know that they must invest in strong teams and team leaders. This includes sufficient numbers of people to staff multiple teams with the right skills needed to develop the technology and the product. They know that to limit both of these means development timing will be impacted.

To this end, they form Heavy-weight teams on their most important projects early in the development life cycle, as early as the feasibility study phase. They know that the fuzzy-front-end of projects is where up to 100% of the time is wasted/lost. They staff these dedicated teams with senior managers.

Many use rotation through "project managing" as a path to executive leadership, because they feel that leading a cross functional team is the best preparation for general management of multiple functions in the future.

The leader is a senior engineer with sufficient technical and managerial experience to make difficult technical and business trade-off decisions quickly, and they are empowered to do this. They have the respect of the team members, and the team members take direction from them. Anywhere from 50-100% of team member's performance reviews are done by these heavy-weight team leaders.

The project has a single engineering leader that all technical issues and decisions roll-up to. They can also take on the role of the system's architect and technical visionary. They are are not experts in every field, but rather an expert generalist in all subjects. They see the complete system being developed end-to-end and can make the trade-offs necessary for the complete system to be successful, i.e. they are able to "connect the dots."

The teams are more autonomous and are collocated, at least the core team sits within a few Steven C. Wheelwright's 4 forms of development team organization, Harvard Business School (adapted by lateralworks).

meters of one another in a common "team space." This common space is where all the meetings about the project take place and where information about the project is posted on walls. When projects have remote teams, a similar physical team room configuration is set up at the remote location. They rotate meetings in each location to maximize face time between members.

The team is the focal point for all decision making about the project/product. The Host functions supporting the project provide teams with what they need to be successful, as provisioners of the right people, the right information, the best quality work, at the right time.

The center of gravity is moved into the team from the host functions. The path to executive leadership/power is through these important team structures. They invest in their teams like they invest in researching new technology, because they understand that a strong team is the best delivery mechanism to rapidly get new products to market.

The four team structures above illustrate how speed and structure are interconnected. The first form is called "functional." This is the slowest way of managing projects; there is virtually no lateral flow of information, and all coordination is done through the functional hierarchy.

The second form is called "lightweight." This is the most common organizing form in technology companies, since it requires no change in the functions where the power resides. Everyone likes it because each function controls their part of the project, yet there is an illusion of lateral control through the project management organization (PMO). These PMO people tend to be note-takers and checklist fulfillment specialists, and always ready with the corporate side pitch outlining the status of the project. In reality, they don't really know what's going on.

The third form is called "heavyweight." This is a difficult structure to implement because it means shifting some of the power from the functions to the team. Since power is a zero-sum game, someone has to give it up to enable a heavyweight team.

It also requires certain management skills to be present on the core team, which is not always the case. Companies usually have only a few good people for this role, but rarely enough good people to staff all the dedicated project core teams. In this heavyweight model, the functions are positioned to provision teams with the right people, with the right skills, and the technology so that the development schedule is accelerated. Decision-making is 80-20 inside the team, with the functions retaining 20% veto power, but the majority of the responsibility falls on the core team to make the right decisions.

Heavyweight is what most aspire to and it represents the second fastest organizing form. They are faster than lightweight teams, but it is clear that all projects should not be set to level 3 - just those strategically critical projects that have the greatest impact on achieving the business plan objectives.

The fourth form is called “autonomous.” Teams are dedicated and collocated in a separate space. They are set up like a company inside a company and have all the advantages of a well funded start-up in terms of focus and rapid decision-making. This form is used to develop breakthrough products or in products where time-to-market success is directly connected with the life and death of a company. Some companies have seen this form of organizing as a way to build new businesses very quickly, and then use it periodically on special projects. Autonomous teams are the fastest and most effective forms of FTTM development.

The autonomous team is analogous to the Navy model. In the Navy model, the Captain (project manager) of a ship leads his/her team to battle. All personnel on the ship report to the Captain while on the ship. The role of the base commander (functional manager) is to provide the training, skills , tools and equipment in preparation for battle. They also support their personnel while at battle, but understand that the Captain ultimately has authority while under the Captain’s command.

The Navy model has worked for more than a hundred years.

2.9 Project-based performance measurement

Summary

Best product organizations measure and reward an individual's performance based on the success of a new product in terms of timeliness to market, customer satisfaction, and overall profitability of the product over time in the marketplace. It is an externally looking product focus versus an internally facing functional emphasis. *Effectiveness* over efficiency.

Normal product organizations measure and reward an individual's performance based functional departmental achievement (i.e. annual departmental goals). *Efficiency* over effectiveness.

Normal

It is very hard to associate an individual's daily activity with the success or failure of new products. The emphasis is on functional or hierarchical achievement vs. new product success.

Managers own functions, but no single person owns the new product. The logic here is that if each function is successful, then therefore new products flowing through these functions will be successful, except it rarely works this way. The function always trumps the product. People behave as they are measured (per W. Edwards Deming). Efficiency wins.

The emphasis is on the function and the hierarchy within the function, because movement up the hierarchy is controlled by the hierarchy. Save money, stay within your budget, achieve specific technical goals, and so on and you move up and make more money in the company.

There is really no emphasis on "product" success, because no one owns the product; it just flows through the functions who work on it, like the car that flows through stationary workers on the assembly line. The thinking is that all I need to do is make sure the door I hang on the car is done right while "someone" else is making sure the complete car is going together correctly at the end of the line. My interest is not in the finished car, but rather than my door is installed correctly. It is manufacturing thinking applied to new product development, except it is with people not parts or machines.

This early 20th century manufacturing thinking dominates how people are measured in development of new products. It is much easier to manage and make people accountable for functional goal achievement, while it is much harder to reward a cross-functional team of hundreds of people for the successful creation of a new technology.

So the default in the normal environment is to measure people they way it has been done since Alfred Sloan developed the modern manufacturing company in the 1920's/30's. The same practices are applied to cross-functional team work that are required to develop a new

technology product. Measure/reward vertically, not horizontally because it is easier to manage and it is a known method. Teams fade to the background, while functions jump to the foreground. This encourages people to pay more attention to the functional manager's demands than the cross-functional project manager's leadership.

Best
Leadership recognizes that new product success determines the rate of growth of the tech-based product company. Profitability and growth comes from releasing new products, on time.

To this end they also understand that cross-functional teams must be measured and rewarded based on the collective success of the team - everyone that touched the product as it flowed through the company from concept and on to high volume production. If the team fails, then the individuals on the project fail, and if the team succeeds then the team shares in the reward.

Traditional functional goals are still in place in these environments, because they know that efficiency of a function is also critical to success. For example, reuse is one way to make a function more efficient. Develop it once and reuse it many times on multiple projects. A project team working on a new product may not take the time to design for reuse on other products, because this may delay their schedule, so they "hard wire" it for their specific application, not thinking of others that might need it in the future. So it is important for the functions to maintain efficiency standards, but these subordinate to the project development goals at the end of the day.

Factors	Coordinating	Semi-Empowered	Empowered
members	part-time	part/full-time	full-time
leaders	part-time	part/full-time	full-time
leader's role	check schedules	coaching, facilitating	empowering members
resource control	limited	great influence	control
mission	set by senior management	set by senior management	set by team
task focus	limited	great	total
performance appraisal	functional manager	functional manager	team/team leader
company policy	followed closely	stretched	work around/ignore
final decisions	recommend	great influence	control

Source: "Cross-Functional Teams", by Glenn M. Parker

There is a 60:40 or 70:30 ratio of functional measurement to project/product measurement and reward. The majority of the measurement system is loaded towards how well a new product meets its goals in terms of its target date for release, its functionality, and how well it performs financially when it is sold (margins, returns, revenue, etc).

The lateral/horizontal emphasis on the product drives teams to more efficient cross-functional behavior. It is not sufficient for my function to succeed if at the end of the day the product is late or missing functionality, so my focus is on how I interact with the inputs and outputs from my function and I always consider how my actions will affect the overall success of the product when it is released. *I am worried about "my car door," but more worried about how the finished car looks when it roles off the assembly line*.

This lateral focus is executed through the degree of "performance review" authority that is given to the functional manager versus the new product development manager (i.e. project manager). Using the same 70:30 measurement ratio, best teams give a majority of the performance input to a team member's lateral manager (the project manager of the new product effort). Both evaluate performance; one from the functional perspective and the other in terms of how well the person is doing to further the project's goals.

This gives the "project manager" a significantly greater role in managing team members, because their future career is based on how well the new product meets its goals when released. Organizations that have embraced this lateral performance measurement system place their best people in those lateral management jobs.

Further, they have identified career paths to the top through project experience, not just through the more traditional functional hierarchical route.

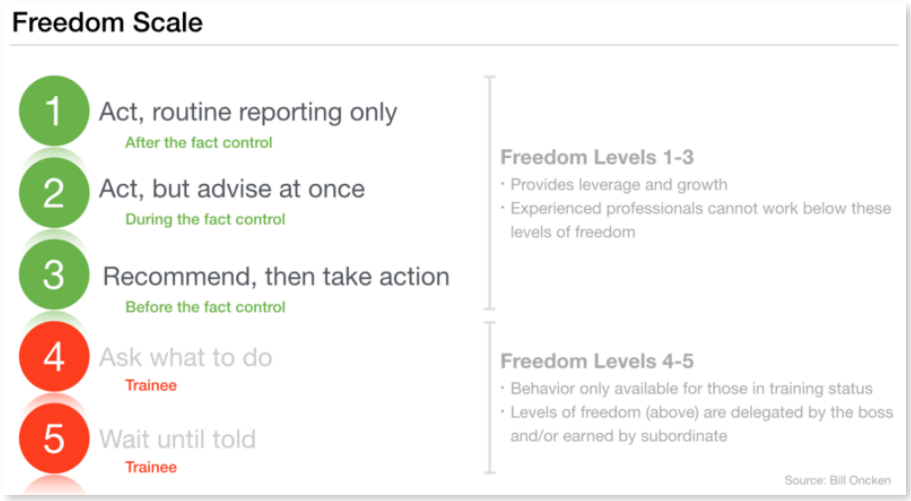
The function's role in these "project-centric" companies is one of rapid provisioning of the best trained and skilled resources to the project teams and to deliver optimized/reused technology across projects in order to reduce aggregate development time. They, in fact, also get rewarded for project success by making sure teams get everything they need and when they need it.

2.10 Freedom Level 1 empowerment

Summary

Best product organizations empower high performers at Freedom Level 1; act, routine reporting only. This provides managers with only "after the fact" control, so they must trust subordinates. This trust enables subordinates to grow faster, take more risks, and get things done faster. They act, then report - which is much faster.

Normal product organizations limit the power and control they assign to project teams/ individuals, believing that people need to be micro-managed in order to optimize their performance. They require individuals, due to a lack of trust in many cases, to permit "before the fact" control over their actions. These teams are slower due to the time required to "check first" before acting.



Description of Freedom Levels

There are five levels of Freedom. The lower the value, the faster decisions are made, actions taken, and the more people feel ownership in the outcome — it's a way to quantify empowerment.

Lets start at the lowest level, in this case it's five (5) "Wait until told." This is a completely passive state, only reserved for Trainee's, yet we've seen executive vice presidents "waiting to be told." The more you wait, the less empowered you'll be and the more you will stay at Level 5.

Level 4 is marginally better than Level 5. In this case the person is a little more proactive and "Asks what to do." The more you ask, the more people will tell you what to do, you will find

yourself quickly back to Level 5. Most of the time people disempower themselves because they continually "ask." You also find this behavior when people are not trusted due to performance problems. The boss does not trust the subordinate, the subordinate does not feel trusted. Its a viscous cycle.

Level 3 moves one out of the "trainee" mode. In this case, you would "Recommend, and then take resulting action." This does provide the manager with some degree of "before the fact control," yet it encourages people to take action and not wait. Perhaps the best example of this from our best practice study was the concept of "UNIDIR." On a very fast development project, people would sometimes start emails with the acronym "UNIDIR," which stands for "Unless otherwise directed, I intend on…" We also called this behavior "default to action," versus "default to delay and discuss."

Level 2, "Act, but advise at once" provides leverage and growth to the individual being empowered/trusted to take actions. This provides the "boss" with "during the fact control" and is a more collaborative form of management. This is also much faster than Level 3, since there is no time waiting for the recommendation to be assessed and a decision made. Level 2 is "real-time" management. Often, we see different decisions/situations getting different levels of empowerment; for example a major strategic business decision such as an acquisition or the termination of a business unit might be Level 3, while technical product roadmap decisions may be granted a Level 1 or 2.

*Level 1, "A*ct, routine reporting only" provides the Boss with after–the–fact control only; maximum level of empowerment/most growth for subordinate; most leverage for boss; this freedom level reserved for those who are "self starters", have demonstrated an ability to consistently produce excellent work, on time within or under budget. Freedom Level #1 may not apply to all elements of subordinates work – depending upon their experience/competence or the relative impact some of the subordinates work may reflect directly on the bosses overall performance.

Normal

Using the Freedom Scale (above) as a reference, development teams/individual contributors operate at Freedom Levels 3-5. This provides functional managers with the ability to "catch" bad decisions before they happen. This is a control-based management system. The control is with the hierarchy above the team. Teams are not trusted with too much authority.

Typically, this behavior is driven by a need from the Host to control financial expenditures. The logic is that if teams are permitted to make spending decisions on their own then they would spend so much money that the company would go bankrupt. This financial control extends to other decision-making topics over time, to include; management decisions, people decisions, and most importantly technical decisions.

In the development of technology, the best engineers and scientists tend to move up the hierarchy. They typically don't have formal management training, they are just good engineers, so the logic goes that an engineer can do anything and especially something as simple as managing people. They default to doing what they know best, solving technical problems and making technical decisions. This means that they jump down into the details, and in doing so, the team is disempowered. At this point the manager owns the project outcomes, the team does not.

Freedom Levels 4-5 are reserved for trainees, not professionals, yet we see many professionals being disempowered by this management behavior. The more concern you have for a subordinate, the more that person is micro managed (Levels 4-5). The problem with this type of management is that people don't learn, they are not permitted to fail, and they have no ownership over the outcomes, because actions are decided by someone else. As a manager you have zero leverage over people at Levels 4-5. It is slow and the bottleneck is the information flow through the functional managers, yet this is the norm in many development organizations.

In normal situations, the best teams and individual contributors can hope for is Level 3. They are permitted to make recommendations and then act on their own. This provides management with the before-the-fact control they feel they need in order to control spending and to make sure *bad* technical decisions are not made by development teams. It is a slow and cumbersome process due to the hierarchical reporting requirements.

Best

Development teams operate with total freedom at levels 1-2. Management trusts the teams and is satisfied with *during* or *after-the-fact* control. The more trust the team gets, the more empowered they feel. They tend to own the project outcomes more in this environment and are better at controlling costs and making sound technical decisions, because they own the outcomes and can be made accountable for their actions/decisions (since they make them).

Levels 1-2 offer the highest level of leverage for management. This is also the fastest mode of working on new product development projects, because people are allowed to make real-time decisions without the need to check first or analyze alternatives for presentation at a later date in which someone outside the team will make the decision.

Best teams are permitted to operate at Levels 1-2 on most activities. The freedom can be granted situationally based on the activity. For example, a team may get Level 1 on technical decisions, but may be asked to work at Level 3 on hiring decisions. These teams are faster and more empowered.

3.0 Team

3.1 Co-develop with tier 1 customers; continuously refine requirements

Summary

Best product teams organizations engage with customers directly to define and then to develop the product. They integrate development organizations so that engineer-to-engineer communication can occur. They use this close-in contact to accelerate decision-making and improve overall system integration.

Normally, product teams determine product requirements from generic market requirements, standards information, competitive data, and/or by increasing technology by a factor of (X) over the previous generation. Customers are not a part of the development process.

Normal

Communication with customers is more like a vendor supplier relationship.

Team does not perceive itself to be credible with the customer, so they attempt to guess/align their schedules and product features with a future perceived need of the customer's – but are always late because the customer's schedule/needs changed in the interim period of time. No contact with customer's development teams, only contact is through low-level product managers and/or marketing/sales.

Perception is that engaging closely with a customer limits your flexibility do create a standard product.

Fear that sharing actual schedule and product performance information with customers may cause the customer to look elsewhere.

Best

The team knows that they need to not only understand their customer, but also their customer's customer requirements. Co-development enables teams to understand the overall system requirements, not only their individual piece. This knowledge causes better systems integration, higher quality, and faster time-to-revenue. Partnering with a customer development team creates customer "pull-through" and a mindset for speed.

Co-development enables incremental releases of partial deliverables and early feedback. Best teams share all schedule and technical information with their customer because they know that trust and credibility can be built on joint problem solving.

3.2 Schedule as a driver vs. reporter

Summary

Best product teams/organizations recognize the cost of being late to market, that for every day saved there is an enormous leverage on profit. Therefore, maintaining or pulling in the new product development schedule is of the highest priority and has a high profile within the company.

Normally, to "make schedule king" would require significant, perhaps unacceptable, trade-offs to quality and/or costs.

Normal

To be faster, something has to be sacrificed. "You don't get something for nothing."
Five development objectives are involved in possible "tradeoff decisions:" feature quality, defect quality, development cost, product cost and TTM. To go faster, quality or cost must be sacrificed. Making schedule king means building a product with less performance or fewer features; or it means "blowing" the development budget.

Management accepts the inevitability of schedule re-plans and is complicit in permitting the schedule to move out. Management demands schedule reductions as if they can be achieved by mandate.

The schedule is used to report progress to people outside the team in the hierarchy, but rarely used by the team to manage day-to-day activities or used to make decisions. When used for reporting, the schedule is “updated” to reflect the particular information needs and project outcomes of the intended audience.

Best

Top management can articulate the need for speed; they know that reducing TTM has a big impact on profitability. They know that for every day saved there is an enormous leverage on profit because revenue to cover the development cost is earned sooner; the leverage is so high that you can "pay to save a day" and "you'll more than earn it back."

Top management can communicate the "economics of delay;" they know that saving TTM exerts more leverage on ROI than reducing development cost or product cost.

Top management leads with "speed"; it makes schedule king. This means they use schedule compliance to lead product teams to find and eliminate wasted time.

Making the schedule king means never moving the schedule to the "right" (i.e., slipping the schedule out). Because there is so much wasted time in the total TTM cycle, the best organizations continually challenge the product teams to find and remove "slack;" top management supports their actions and together they prevent or recover slippage.

Making schedule king (and used as the driver of project activities) works because gestation and overruns alone involve about 80% of the wasted time in new product development cycles, most of which is waiting. Also, a high proportion of the development schedule is also waiting. Making the schedule king means permitting no "macro" schedule slippage; however the "micro" schedule may move in and out.

3.3 Translate customer requirements to product definition, reconfirm throughout development

Summary

Best product teams/organizations involve the customer during each phase of the TTM cycle (not just the development portion) to assure product superiority and a shorter TTM schedule.

Normally, customer information is gathered at the front end of the development cycle with after-the-fact feedback at beta. Information typically comes through the sales account team or the marketing function. The quality of the information may be further eroded, depending upon the quality of the relationship between marketing and development or the project team.

Normal

Customer input is gathered at the front end of the development cycle with only spotty validation from the customer until beta.

Customer contact is often limited to the sales account team or the marketing function. Effectiveness of product requirements/specifications is governed by the quality of the relationship and respect between marketing and development. If they listen to each other, specifications are pretty good; if they don't, specifications suffer accordingly.

Best

The best teams get and use the "right" customer information throughout the entire TTM cycle (gestation, development, overrun); this enables the product to be superior and TTM cycle time to be shorter. The best teams plan and schedule voice-of-the-customer (VOC) involvement, run it in parallel and integrate it with the entire TTM schedule (not just the development portion). The best teams design VOC involvement to fit the TTM phase:

- *Planning* (gain counsel on business strategy)
- *Gestation* (run "skunk-works" with leading customers)
- *Requirements* (create user groups; collaborate in Quality Function Deployment (QFD) modeling)
- *Design* (develop partner labs and allow direct contact with designers)
- *Development* (conduct partner or customer "use-ability" testing)
- *Beta* (do early installation of product for customer business use)
- *Field Support* (get feedback and assist with Root Cause Analysis) and
- *Overrun* (create mutual problem-solving forum to keep the relationship)

The best teams scrub the VOC schedule repeatedly, and review it for quality, timeliness and effectiveness in the rapid decision-making forum. The best teams use customer input to resolve major development decisions throughout the TTM cycle, not just the front end of the development phase. This can save weeks of management and team deliberation.

3.4 Know what they value, when they want it, how much they will pay for it

Summary

Best product teams/organizations distinguish between customer requirements and product specifications. They work hard to get requirements right in terms of the solution the customer is really looking for.

Normally, specifications are considered "right" regardless of customer need. "Customers typically don't know what they want. If they do it usually confirms what we already knew."

Normal

Get specifications "right" regardless of customer need. The attitude is *"Customers don't know what they want; asking them confuses them and us."* When talking to customers, it's difficult to hear anything new. Customers often talk about their business and not about the product, or they just confirm what we already knew about the specifications.

Lots of customer information is available (customer satisfaction, root cause analysis, etc.), but it's not connected to the requirements process.

Best

The best teams distinguish between customer requirements and product specifications.
They get requirements right in terms of customer solution and total customer satisfaction. The best teams evaluate the product's value in terms of a solution on actual customer path and consider the product's role in the entire experience the customer has with your firm. Executives articulate the business logic of customer satisfaction and communicate the message that customer satisfaction relates directly to market share and repeat business.

The best teams link customer satisfaction directly to the requirements-setting process with a closed loop. Use the Teach-Negotiate-Tell (TNT) approach to customer involvement:

- T (Teach): Let customers teach the core team how the product is used to achieve a business solution on the customer path.
- N (Negotiate): Agree on "how much is enough" for key requirement trade-offs.
- T (Tell): Tell them what you are planning to do (what the specs will be and the delivery date).

Find out if they will regard this as a superior product or a "me, too" solution at time of introduction.

3.5 Understand customer wants vs hows; prioritize the drivers that maximize customer satisfaction

Summary

Best product teams/organizations develop product specifications by:

- Systematically studying the customer's priority requirements
- Learning and responding to the total customer satisfaction "formula"

Normally, customer requirements are skipped in favor of going directly to product specifications. Customer input is acknowledged only when large customers complain.

Normal

Skip customer requirements and go right to product specifications. Use complaints of large customers for primary guidance. Quality Function Deployment (QFD) considered too complex; other VOC tools too soft.

Best

Customer requirements and product specifications address two different but related worlds; the best teams always distinguish between the two. Since the customers ultimately determine whether the specifications are right, "you might as well find out up front and fast." The best organizations converge the information from several VOC tools (focus groups, surveys, customer satisfaction database, etc.).

They use QFD or something like it; it's the best way to get high quality and low-cost products. Specification changes is one of the primary causes of TTM delay because it emphasizes do-it-right-the-first-time (DRFT), the best teams use the QFD process as it reduces the total number of specification changes over the life of the project.

If QFD is not used formally, the best teams use an informal tool that accomplishes the same and is mandatory. Otherwise, the product will ship with inferior specifications or there will be delay from rework to correct the specifications later.

Even if QFD is not used to establish specifications, the best organizations establish a small team to run a modified QFD as a fast validity check on the effectiveness of converting the customer's requirements into specifications. The best teams run the following validity check on specifications: take the six specifications derived from the customer's top half dozen requirements; get six customers to "sign off" that; each specification reflects a valuable interpretation of the customer's underlying requirement. Based on their knowledge of likely competitive trends, delivery on the specification would be superior in the marketplace at the anticipated time of the product's introduction. If other relevant specifications are delivered at competitive parity, they would buy the product.

3.6 Start design while refining requirements; never freeze specs

Summary

Best product teams/organizations develop short form ("abbreviated") specifications in order to start design while they are concurrently refining customer requirements and product specifications. They recognize that freezing specifications does not take into account the changing needs of the customer over the development cycle. Instead they:

- Work consciously to minimize the "change volume."
- Attempt to incorporate as many changes as possible early in the development cycle (when the rework penalty is less).

Normally, work on product design does not start until specifications are fully defined. Or, design begins with few, if any, specifications.

Normally, a change control strategy does not exist or is ad hoc resulting in an arbitrary freezing of specifications or a large volume of specifications that tend to rise throughout the development cycle.

Normal

Designers start with few if any specifications Designers are forced to follow the phase review steps and not start their work until specifications are fully defined.

Best

The phase review concept is a convenient simplification used by management to control projects and empower the team. Specification setting and design are by nature interdependent. Feasibility and desirability are inseparable. One without the other makes no business sense. In innovative development projects, both are evolving. The best teams run them concurrently from the start. The best teams get the design started ASAP with fast 80-20 initial specifications. With a few leading customers they use a fast but informal QFD to develop a short package (one page) of specifications expected to meet 80% of the customer's most important requirements and expectations. The best teams coordinate the progress of the design process with the QFD process; they install version control on external and internal specifications.

Ad hoc or no-change control strategies results in divergent or chaotic paths to converge specifications. Normal organizations follow a strategy of Fixed Containment of specifications. They freeze specifications at the front end and insist on no further changes. Because the world changes (e.g., customer needs change, competitive offerings improve or technical difficulties arise).

These almost inevitable events force a major re-plan to another set of "really" frozen specifications. This cycle (freeze specifications, the world changes, and cause major re-plan, re-set specifications) is one of the primary causes of TTM overrun. In normal operations there is a large volume of specification changes (especially around major re-plans); the rate of change rises as development proceeds. When customers are involved normal organizations have systematic procedures for convergence (i.e., the more customers you ask, the more specification changes and rework you get).

However in the best organizations make the minimum number of changes to achieve convergence and adaptation of specifications. Even if the total number of changes remains the same, it's faster when they are front loaded (when the rework penalty is less) and then taper off as development proceeds. They follow a strategy of Flexible containment of specifications: They get a short form set of "fast specifications" early. They minimize the "change volume", but maximize the speed of adaptation to customer or competitive changes through continuous use of VOC and QFD.

They minimize the change volume but maximize the responsiveness to technical feasibility or difficulty through delineating a path of early commits proceeding to late commits and contingency planning for high risk modules. The best teams establish change control guidelines and a Change Control Board (CCB) to implement flexible containment. They expect development teams to govern themselves to the "spirit and letter" of the guidelines; whenever the CCB was needed, they convened it instantly.

The best teams review the change process in the rapid decision-making forum and use the following guidelines:

- Stop the unnecessary changes.
- Sort the early from the late commits.
- Implement all changes quickly.

3.7 Fail-Fast; integrate/proto early, frequent & many do-it-try-it-fix-it learning cycles

Summary

Best product teams/organizations push for early systems integration in order to prototype their design so that they "fail" as fast as possible. They know that with failure comes learning and the faster they learn, the faster they innovate. They accelerate learning cycles (do-it, try-it, and fix-it).

Normally, failure is avoided because it represents poor design thinking. It should be right the first time, so teams take a long time to get to the prototype stage (systems integration). They integrate later in the design cycle, so problems surface close to final design verification milestones, then push past target dates when more time is required because it was not right the first time. The learning cycles are elongated because people want to make sure each step is perfect, causing slower overall learning. Projects take longer.

Normal

The phase-gate waterfall product development life cycle determines behavior. Each function passes the baton to the next in a sequential and methodical manner. The integration phase of the design process is at the end of the life cycle. Each function focuses on making their part perfect. The theory is that if each function does their job (i.e. Unit Testing) then when the final system is integrated it will all work flawlessly - since all the subsystems have been verified, the integrated system should work as designed.

In practice it does not work this way, since problems caused by systems integration can't be predicted or tested at the unit level of the design process. This is even more true for extremely complex systems. In fact, it is the integration process that usually surfaces design failures.

The reason for the late integration is typically caused by functional development practices. The engineering is validated (EVT), then passed to a different group of people to validate the design (DVT) where systems are integrated and prototyped, and then the new product is production validated (PVT) to confirm it can be manufactured within cost targets and conforms to technical specifications.

Each phase is linear. Failure at any stage causes the design to cycle back through each step sequentially (EVT->DVT->PVT->EVT….). Design failures are usually found later in the process using this waterfall method as the product gets thrown over each silo wall to the next group of specialists. This process is slow and can result in integration problems and/or manufacturability issues later in the life cycle. The higher the silo walls, the greater the integration problems later on.

Failure is avoided. The "right the first time" mindset causes people to spend a lot of time making sure their design works. The time accumulates with each design step. Some teams say, *"It is not as important for us to succeed, we just can't fail."* Fear of failure and being blamed for it causes engineers to hold on to each step as long as they can to make sure everything works prior to system integration. When the product is finally integrated, either as a simulated design or a physical product, they find problems that no one anticipated. However, much time was lost in the early stages making sure everything was right, so the team has lost any buffer to iterate the design and prototype builds. Ultimately more design iterations are required and the project slips. It is forced back to the start of the design process for the long journey back through the linear phases and gates.

Best

The product life cycle development process is agile and iterative. Teams are encouraged to fail as fast as possible because they know that failure is the fastest way to learn. Learning is the fastest way to finding technical solutions. With solutions comes innovations.

When teams accelerate their learning, they can accelerate the product's development. They try to "break it" early in their design thinking through early prototyping and/or simulated systems integration depending on the product. They use this failure analysis knowledge to improve the design. It is understood that unit testing will not catch all the problems. They unit test rigorously, but push for early integration to evaluate how the design behaves as a system. The behavior of the system informs future design decision-making at the unit level.

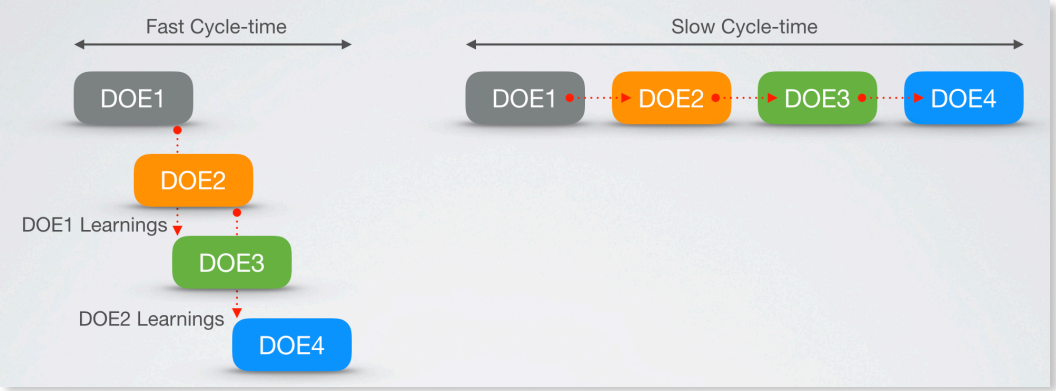
Cross functional teams are involved in developing the product. The functional silos that gate each development milestone are not present in FTTM Teams. FTTM Teams still go through the EVT, DVT, and PVT phases, but they are performed concurrently (i.e. overlapped). There are no gates, but rather checkpoints. The fastest teams have the greatest degree of parallelism in their development process. In the 1970's-80's this was called Concurrent Engineering, now it is referred to as Agile Development. It is the same thing, doing steps in parallel and iteratively, but now with a new name.

They strive for rapid design-test-fail cycles as the path to quicker learning. This means that there are many system integration tests (both simulated and physical depending on the product). Each team member works with the assumption that *"It is better to be roughly right, than exactly wrong."*

This means that roughly right is faster and good enough at the unit level, because they know that the more complex problems will result during systems integration testing or the prototype phase. Spending more time in the *diminishing point of return* part of the curve is a waste of time. The push to get the design to the point it where it can be integrated into some form of a

prototype (physical or simulated), because they know that this will result in identification of design problems earlier that will need to be solved.

Learning is accelerated by getting to the prototype stage faster, then trying to iterate faster by accelerating the time each step takes (design-test-fail-redesign). The high performance teams accelerate learning even faster by running concurrent design-test-fail cycles (i.e. well managed Design of Experiments - DOE).



For example (above), they start DOE1 followed in time later by DOE2, when DOE1 is ready they evaluate the results while still processing DOE2, they start DOE3 to verify what they learned from DOE1 while DOE2 is processing, when DOE2 is finished they start DOE4 with learnings from DOE2… and so on. This concurrent design cycle is key to faster development cycle time. They know that the additional costs associated with concurrent agile development are a fraction of what they return with on-time or early delivery of the finished product to the market.

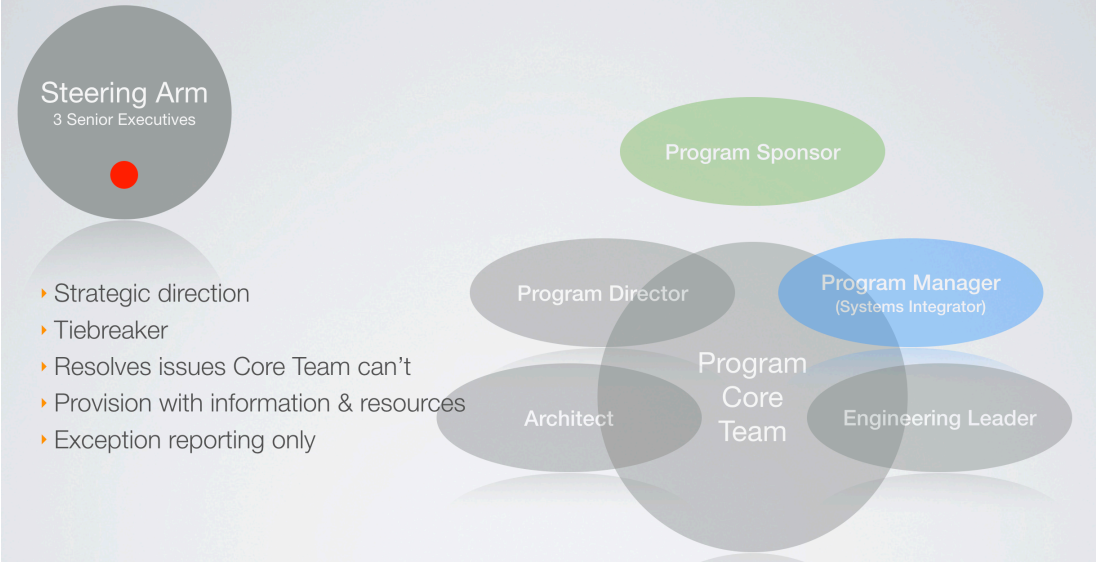
Most importantly, the “fail fast” mindset is the key to unlocking or freeing-up creative thinking about problem solving. Fear of failure usually stifles creativity. Early integration and design validation is the key. It provides a validation of the design thinking early enabling vector changes based on failure analysis. They know that there will be many design validation iterations, not just one big one at the end. This causes more experimentation earlier in the design process and more understanding of the problems that need to be solved. This provides before-the-fact knowledge much earlier in the creation process that can be used to refine design thinking.

Short intervals of iterative learning and incremental vector changes lead to faster solutions. Long intervals, few iterations, and big vector changes usually result in big project delays and/or compromised product design.

3.8 Cross Functional Core Team ROLES not FUNCTIONS; lateralized

Summary

Best product teams/organizations organize their core teams around roles, such as Architect and Chief Engineer. Each team member has a functional responsibility, such as Hardware or Software, but the dominant behavior/bias is towards their program role. These program roles cause a lateral focus from beginning to end of the development project.



Example of a core team's roles, using a Steering Arm to help them remove organizational interrupts.

Normally, teams are organized around functional responsibility, such as Hardware or Software. These functional roles cause a vertical bias within each person's functional specialty. Effectively repeating the functional silos at the core team level, creating more emphasis on the parts rather than the whole.

Definition

A "core team" are typically 5-10 key people that are responsible for developing a new product or executing a project. They represent each of the functions that touch the product from the initial concept through design and engineering, and on into manufacturing and out to distribution of the new product to the customer.

FTTM teams include all functions on the core team on day one, from marketing to manufacturing and even finance, procurement, and HR on larger projects. They also give members project "roles."

Normal teams focus on engineering core teams (i.e. the disciplines represented within the engineering function) and only tangentially involve marketing at the start of the project, then hand over the design to another "core team" in manufacturing that is charged with making the product manufacturable or meeting production cost targets. More typical are multiple functional core teams moving the ball up the field from one function to the next.

Normal

The core team, if there is one, is really an engineering team because it is understood that the engineering phase of the new product creation process is the most critical one. Marketing develops the MRD (marketing requirements document) and hands it off to engineering, so there is no need for them to be on the project core team (of engineers, which is the main focus of development anyway).

The requirements conflicts are rarely resolved at this stage, because each side maintains "plausible deniability" - marketing can blame engineering for not developing what the customer wanted and engineering can blame marketing for not clearly describing that the customer wanted. What is eventually developed is usually what engineering can create, rather than what a customer expected (if anyone really knows what the customer expected). The conflicts in requirements is rarely reconciled due to the "siloed" organizational/teaming structure used to manage the project.

The manufacturing function also does not need to be on the core team because their role is later in the process and there is no need to waste their time on early development issues, because they are focused on downstream problems associated with manufacturability and production of the current products being manufactured. They are not involved up front in the gestation phase of the project where they may have been able to inject "design for manufacturability" (DFM) thinking into the design team.

The core team is a cross functional *engineering* team made up from each of the engineering disciplines. There is for example a hardware team member, a software team member, maybe a test and verification team member, and various design leads who can also be on the team. The team also includes technical subject matter experts (individual contributors). These "teams" can be quite large, sometimes 20-30 people, because no one wants to get left out of the technical decision-making process. So in effect, these are not really core teams, but can contain almost all the people working on a project.

The bigger and more complex the project, the greater the number on the "core team." Meetings tend to be frequent, long, and poorly structured, often degrading into "rat hole" technical debates and discussions with the strongest SME's winning hard fought arguments. The team meetings are one of the few sources of information about the project, which is why they attract more people than are typically on the standing invitation list.

There are usually a few subject technical matter experts and position-power people that tend to drive these teams. The emphasis is on the function and the particular issues/problems in that function at that time. There is less of an emphasis on the product and the overall schedule, unless there is also a project manager on the core team. The project manager on these teams is more of a record keeper and occasionally raises red flags about delays, but few pay too much attention because the emphasis is on technical problem solving and "*the schedule was unrealistic anyway.*"

Since the core team is dominated by functional thinking, discussions tend to drill down into specific areas of the project often losing sight of of the product perspective. Much time is spent on SMEs defending positions or establishing the ground rules for the integration points between functions - *"I deliver X to you and you deliver Y to her...."*

Really, engineering core teams are just a new term for what used to be called engineering department heads that would meet regularly to try to synchronize the development efforts moving through their functions. Then the term "team" became the way to describe any group of more than two people. Every gathering of people became a "team."

Then "cross-functional" teams became the rage. When applied to "engineering" core teams they meant that that you invite all the department heads or technical leads from engineering to be on a "cross-functional" project team. Of course the same people site on every cross-functional team in the company, making this group the information bottleneck in many cases.

Best

The core team is a dedicated (100% to a project if possible) group of 5-7 people who represent each of the functions that the product flows through from concept to volume production. They assume project roles, subordinating their functional roles (which they still maintain). These project roles include; chief engineer, system architect, system integrator, customer integrator (marketing) and so on.

They have another outer layer of core teams (sub-core-teams) that tend to be more focused on each function, such as an engineering core team comprised of each of the various engineering disciplines. This engineering project core team would report into the Chief Engineer on the program core team or the System Architect, for example.

The second layer of core teams can also represent integrated project milestones deliverables. So for example, one of the first milestones is to define the product. This could be a team lead by the marketing person (customer integrator role) on the program core team and include representatives of the engineering and manufacturing sub-teams. The milestone team is formed

for the sole purpose of completing the integrated project milestone and then disbanded to form new “milestone teams” associated with subsequent milestones.

These milestone teams report into the program core team of 5-7 people. There can be hundreds of people working in this flat configuration, with a small group of people in the center and multiple sub-core teams revolving around it, like the moons revolving in concentric circles around a planet.

The emphasis for the program core team is “role” responsibility. Take for example the Systems Architect role. The Architect is responsible for the total product or system and how each element of the system interacts/interfaces. They focus on the interface points and make the trade-offs necessary to optimize the total system (rather than sub-optimizing its components). There are always cost, technical, and resource trade-offs that need to be made to deliver on time. The system architect owns this job.

The Architect is responsible for making these calls so that the end product is not compromised. The Architect may also wear a functional hat on the core team. They may also be, for example, the person who is responsible for the software development function. But their functional role subordinates to the program role. One of the missing “roles” on many failed technical development projects is the systems architect who oversees and makes trade-offs for the betterment of the end product. When products fail technically, it is often due to sub-optimized decision-making which is driven by functional role bias.

3.9 Get best people; mix of inside & external SMEs; virtual teams

Summary

Best product teams/organizations secure the highest technical and skill level people for their development projects combining internal employee resources with external contract resources who function as the virtual project team. No distinction is made between internal and external resources. It is one virtual team focused on a single mission.

Normally, development teams are put together with who is available at the time. Little to no effort is made to secure external resources due to the fear of IP leakage or a belief that the only people that have subject mater expertise are already within the company. Key people are usually oversubscribed due to there being too many projects and not enough people. Resource conflicts go unresolved and the best people are always oversubscribed.

Normal

New projects are initiated with the people that are available at that moment in time. Teams are resource starved as a management technique designed to "optimize" the headcount. Teams are usually understaffed and lack specific skills due to the *"What is available"* method of project staffing. The key people on the project are also working on manny other projects at a time that are further into their development life cycles, so their contribution at the start of the new project (where their time is most critical) is minimal. Firefighting current problems take precedence over preventing future problems from occurring. The best people are stretched thin firefighting. This is a cycle teams find impossible to escape.

The general feeling is that what the team is creating is so special that there can't be anyone outside the company that can help them. This bias has secondary "job security" benefits for the inside people, making them more valuable to the company. Further, the fear that intellectual property (IP) will leak is the other reason external resources are not investigated as possible project contributors. *"We can't get consultants on our project because then everyone will know what we are doing,"* becomes the main reason for doing it all inside the company.

An inability to use external resources exacerbates the resource problem making internal specialized people even more overloaded/overworked. Further, lack of external thinking tends to focus the technical problem solving on what the people inside know and down't know, making it take long to solve difficult problems.

Teams make due with what they have been given. At the end of the day, lack of the right people, with the right amount of time and skills becomes the perfect fallback excuse for non-performance on the project. Management knows this too, so they don't push it. This behavior

often leads to general cross-team morale problems and a defeatist attitude when the best people are not able to be secured for difficult development projects.

Best

The project team is seen as the most critical thing to get right. The wrong team means certain future failure, so every effort is made to secure the right resources on day-one of the project's formation. This includes not just the needed quantity of people, but also to put in place a group of people with the necessary technical skills to achieve the project goals. Tremendous effort is made to secure the most technically competent people in their respective fields, regardless of them being inside or outside of the company.

Careful balancing of the project portfolio enables proper resource planning so that resources from Project A can transition in time to start Project B, when Project B is getting started. The fastest teams have fully dedicated resources to a single project. Medium-fast teams have resources spilt between 1-2 projects. Slow teams are accessing resources that are multiplexed across <2 teams (the typical allocation is 5-10 simultaneous projects in shared functions like "central engineering").

Project teams are resourced, fully, on day one of the project or a careful effort is made to bring key people onboard prior to the time they will be needed on the project with some consideration made for the start-up learning phase required to become productive. The right people delivered to the project at the right time is a major focus for management, HR functions and external recruiters, because they know that the right-team can have the greatest impact on a project's outcomes.

No distinction is made between internal employee resources or external contract resources (contractors, consultants, suppliers, vendors). If the team lacks a skill, then that skill is quickly retained from the outside if that person or resource is not available inside the company.

IP protection is secured through NDAs and everyone recognizes that TTM is the best form of protection, in that if the project is late then the value of that technical knowledge is lower if it were communicated to a competitor (it is now out of date). Who wants "old" or outdated IP? This is a conscious trade-off made to get the right people who can enable the project to get to market faster. Obviously, the most sensitive differentiating IP is worked on by inside people, because it is recognized that this IP has the core value the company provides its customers.

These teams have inside and outside people working such that it is hard to tell who is an employee. They are all focused on the successful project outcome. Further, using external resources is a way to deal with the periodic resource shortfalls experienced in a multi-product development environment when resources from Project A are not available to roll-off onto

Project B. The external resources bring new thinking to teams and help to accelerate problem solving, because they tend to have broader experiences to leverage.

The Host knows that if they provision the team with the best people at the right times during the project, they can eliminate this as a possible excuse for poor performance later on.

3.10 Aggressive planning

Summary

Best product teams/organizations use critical path scheduling systems and involve the whole team in interactively planning the program. These teams spend 3x the time in planning their work than normal teams. They reconcile the schedule to the targets and if the gap is too great, kill it before it starts. When started they aggressively refresh their plans every week and are constantly looking for ways to pull-in the schedule – before it slips.

Normally, when schedules exist they are not reflective of what is actually happening on the program and therefore not used by teams to manage the work effort. Schedules don't represent the "real" time necessary to accomplish the goals, but rather reflect the "not bought-in to" program targets of senior managers. Large slips are typical and unforeseen.

Normal

Schedules, if the exist at all; tend to be conveniently "manufactured" to meet the top-down driven dates that have been "given" to the team. Somehow these schedules always show the project finishing "on time" at the beginning of the project.

These top-down schedules never change, until the day of a milestone review meeting when it is announced that they have slipped the project 6 months.

"Group-think" sets in and schedules are not challenged for their accuracy or completeness. Individuals who challenge the schedule are perceived as not being "team players." Schedules rarely reflect the reality of under-resourcing or over optimistic estimates of what can be done… and are rarely believed by the "workers" who have to complete the tasks. And as a result, are seldom used as a tool for managing daily workflow – only for presenting status to top management.

Since schedules are rarely changed during a project, they quickly go out of date and only get modified prior to major status review meetings. Most team members never see the schedule and work on the things they either; a) want to/have interest in at the time, b) have been told to work on by their functional manager, or c) guess they should be working on based on information they have gleaned from attending many, many project meetings.

Schedules are high-level and don't reflect what the team is actually doing.

Schedules mimic the functional structure and further reinforce the walls between the silos. Schedules for gestation, customer involvement and process improvement activities rarely exist. Development schedule content and design are not improved or reviewed after they are first

built. The prevailing attitudes regarding scheduling are, "Don't tell me my job. -- I'm a competent professional. -- I can schedule my work better than an outsider. Just make sure I get the inputs from others that I need - when I need them." Lots of time and creativity is spent on the "design of the product."

Almost no time is spent on the "design of the work" required to accomplish the new product development schedule. There are design reviews of the product.

There are no "design reviews" of the work content. They resist short interval detail; they fear opening the door to micro management and control by senior managers.

They fear over planning. They do not have sufficient grasp to define activities in short duration detail. Focus is on damage control and after-the-fact recover efforts to deal with major slips in the schedule. Fire fighting and heroics are rewarded when a project is saved in the "eleventh-hour" through overworking (i.e. motivating) people and compromising the product functionality in the interest of time.

Best

They use a rigorous critical path based planning and tracking system, in order to:

- Develop customer focused mission statements and milestones: structured on customer needs rather than on internal functional deliverables. Create work-breakdown-structures with near-term detail.
- Generate market driven schedules based on customer need dates (rather than when they can get it done based on their available resources).
- Get top-down and bottom-up commitment to the schedule.
- Know the difference between a TARGET and a real SCHEDULE. They understand the gap and take before the fact action to accelerate the schedule. Create realistic schedules that are developed by the team, for the team as DRIVERS rather than REPORTERS of status to senior management. If the gap is too great and can't be closed, they kill the project.

REFRESH the schedule every week (sometimes more frequently) with more detailed activity breakdown, actual work completed/in-progress reports, and try and pull-in their critical path in the near-term. Pull-in the schedule before it slips.

They scrub every aspect of the TTM cycle. They pay attention to schedules for the gestation period, the voice-of-the-customer involvement, improvement of processes, as well as the development schedule. They know that scrubbing expands grasp. It increases the cycles of learning about what it takes to produce the desired outcome and it promotes migration of practices throughout and across teams. They recognize that scrubbing replaces the "mistake, fail, punish" cycle with a "try it, fix it, recycle learn" cycle. They know that grasp of the whole

plan in sufficient detail is necessary to control the work and pull in the schedule. They treat scrubbing like a "design review" of how the work is to be done. Most of the opportunities to improve the way work is done involve the dependencies between work activities. They understand that all team members need not grasp the entire plan. Scrubbing should not be done with the whole team; rather triads of knowledgeable and capable individuals get together to scrub small chunks of the schedule.

They move up the planning continuum:

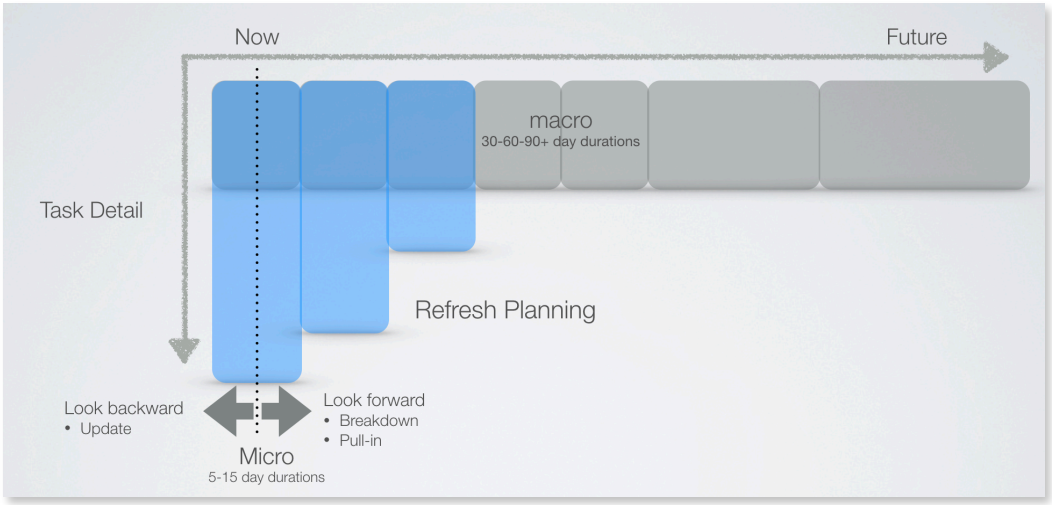
- No planning
- Planning but no refreshing
- Planning with continual refreshing

During the development of the baseline plan they scrub for completeness, accuracy of activities, activity duration's, resource allocations, precedence relationships, potential for concurrence, and preventive and contingent actions. During the execution of the project they scrub for short interval schedule pull in opportunities.

3.11 Macro-micro planning; micro near, macro far, short interval scheduling

Summary

Best product teams/organizations start planning their projects from the macro perspective to include all aspects of the product realization process from concept to release. They breakdown the near-term schedule (<6 weeks out) into smaller micro increments that are used to pull-in/accelerate. They roll the planning window forward every few weeks, breaking down into shorter interval as they go and when more is known about the project.



Normally, detailed plans are generated for the complete project. It is time consuming, so few people work on it so there is little team buy-in. If the project changes scope, then the planning has to be redone or is not done at all due to time limitations. There is a belief that detail equals accuracy, so more micro detail means a more accurate schedule.

Normal

The planning process is a painstaking and laborious process involving just a few people and usually just the project manager compiles detail functional plans from the various functional teams working on the project, such as software, hardware, mechanical, test, and manufacturing schedules. They do their planning on their own, without the project manager being present. They are all separate and detailed schedules that describe each function's work flow. They are usually in different formats and follow different standards which make it difficult to integrate.

The project manager is left to somehow stitch them all together into a cohesive integrated plan. Since they were done separately, the interface points between functions don't match up. This starts the "communications" problems between functions that the project manager is supposed to sort out ("manage").

The detailed schedule is too complex for anyone to understand, so few people ask questions about it or refer to it in their daily activities on the project. It is really just the "project manager's schedule" and quickly goes out of date as things change on the project. Detail is seen by everyone as providing a more accurate estimation of project completion and project status. So more detail planning is always encouraged, which further consumes people's time making them hate the planning process even more.

Detailed schedules are usually complex. The complexity renders them unusable, but they make impressive documents to scare senior management away and give them confidence that the team knows what they are doing. As a management tool for the team, these detailed schedules tend to be of little value. Rarely does the detailed plan provide a reasonable estimate of completion compared to when the project is expected to be completed, due to its complexity, poor structure, and functional bias.

Best

The core team (5-10 people) meet for 1-2 days in a focused offsite workshop to develop their macro plan. They are able to look at the project in the abstract; from a "macro perspective." The macro plan is no more than 50 tasks that represent the total project from concept to some future defined end point (usually some sort of product introduction or manufacturing ramp phase).

The macro plan is constructed using 4-6 major integration points by which the team will measure progress. The outcomes of each integration point are agreed to and differences are reconciled between team members. Expectations are mutually set and confirmed at this point. These are cross-functional milestones. Cross-functional integration is forced top-down through this process. This avoids the interface problems described earlier as "normal" and the separate planning associated with the functional bias.

The macro plan is a strategic statement from the team about where it is going and how it will get there. It is designed to raise and respond to the strategic decisions that must be made about the project in the early stages in order to finish on time.

The macro plan provides the first gap analysis to the core team. This is a measure of time from when the project is needed to be finished to when it really will be finished, based on the critical path macro schedule. Most projects are late by 30-50% of the overall project duration, but rarely do teams know it in the first 1-2 days of starting their projects. The macro planning workshop generates this critical gap information and causes the full team to participate in the planning process, which is the first step in getting cross-team buy-in to the schedule.

The macro plan also permits the core team to do simulation studies together to explore alternate paths to faster project outcomes. These simulations generate the strategic discussion needed at the start of a project in terms of how the project should be structured, what technical decisions need to be made, what resources are required, and so on. The macro plan does not cause the team to get lost in the forest they way detail micro planning does, but rather it provides a birds-eye view of the complete landscape from start to finish.

The gap that emerges from macro planning usually stimulates urgency much sooner for project teams and the supporting management hierarchy. When, for instance, you know that your project is 6 months late on a 12 month project completion window, you know that strategic decisions must be made to close this considerable gap (your 12 month project is taking 18 months). When this information is not known, teams tend to accept the project requirements that they where given and only find out much later into the project that they are missing the target date by a wide margin. By then it is too late to recover. The macro planning process exposes the gap sooner, with very little information, when there is still enough time to take corrective action.

As time moves forward, teams breakdown the macro plan into incremental detail. This involves the next level of team member participation, either team leads and/or individual contributors do the task breakdown. The macro plan provides the framework to decompose long duration macro tasks into smaller 1-15 day tasks. The focus of the breakdown is <6 weeks out, because any further out into the future usually means less clarity. The more cutting edge the technology, the less clarity teams have on future details. So they only breakdown what they know today. The planning window roles forward every few weeks with more tasks being decomposed as more is known. This process avoids major schedule redo’s when things change significantly.

The point of breaking down into detail is to find ways to do the work faster by either doing it differently or doing things in parallel or even eliminating things all together. The near-term detail is used to accelerate the near term schedule or recover time that was lost due to slippage. This is called short-interval scheduling.

There is no sense breaking down future tasks if the details are not known to the team at the time, so future tasks are left as macro placeholders that can be decomposed when more is known in the future and as you get closer to them. Eventually, the project is fully broken-down into detail, about the same time the project is finishing. So planning never ends on FTTM Projects. It is continuous, iterative, and constantly changing as the project changes.

3.12 Team planning, simulation, continuous scrubbing, breakdown, ownership

Summary

Best product teams/organizations involve the complete team in planning and tracking their project through an engagement process involving ongoing schedule scrubbing to gain individual ownership over the project outcomes.

Normally, a single person (project manager) creates the plan. The plan is then shown to the rest of the team who are expected to buy-in and accept it. There is very little group participation in the ongoing maintenance of the schedule as the project changes, normally the solo project manager is driving all changes to the schedule and reselling the new dates to the team.

Normal

The project plan is created by one person, usually the project manager or technical lead for the project. They use their own individual experience or may involve 1-2 others if needed at times, but the basic organization, work-break-down structure, duration estimates, and network logic are made by the project manager. If things don't fit, the project manager reduces durations until they do. These aggressive duration estimates become "challenges" and "opportunities" to think creatively by the people who have to do the work. They tend not to be very enthusiastic about these opportunities.

When finished with his/her detailed plan, the project manager needs to "sell" the schedule to the rest of the team. Because they were not really involved in creating it, they are somewhat confused by the schedule and don't say much for fear of looking ignorant or getting too involved. They know it is not their plan, so have little interest in the details of it in terms of the logic and time estimates that where made for them.

Oddly, the schedule fits inside the time specified for the project's completion. This further disengages team members, because they know that the durations are underestimated and certain risks have not been factored into the planning. It is a best-case, 100% goes right schedule they are seeing for the first time.

After the schedule review meeting with the team, the schedule quickly becomes out of synch with what the team is actually working on. The effort to maintain the schedule (so it reflects what people on the project are actually doing) is so great that the project manager spends less and less time keeping her/his schedule up to date. They default to their management tasks of driving daily activities and communicating project status to higher level management. The schedule slips into the background.

Periodically, the project manager tries to bring the schedule up to date for major milestone reviews or to check off a item at a gate review (*i.e. "Up to date schedule presented, check… it is done, move on to the next gate review line item"*). The schedule and the work on the project tend to drift away from each another. The team rarely sees the schedule and uses their own detail methods for tracking their own work (spreadsheets, cloud team collaboration, email, lots of meetings, and so on). The project manager uses these different information sources to try to estimate current status and when each group will finish.

There is ongoing of discussions about why the team does not buy into the committed dates and why they don't spend more time planning their work, but little changes in the way they plan.

Best

The full team is engaged by the project manager in planning and tracking the project, so that the project initially reflects their committed dates and on an ongoing basis, so that it keeps pace with the changes occurring in real time on the project. This is done using low-time-overhead practices enabling engineers to focus on their work.

The team spends the necessary time to plan and simulate different critical paths to specific end points. They know the planning time invested will pay off over time with better coordination and communication across the team. They understand the schedule gaps and explore, together using the schedule, how to close those gaps. The exploration process brings the team closer and exposes all of them to future challenges, in advanced - like having a project simulator.

Periodically, team members scrub the schedule in smaller groups. Scrubbing means to review detailed sections of the schedule for accuracy and completeness, based on the changes on the project. Constant schedule scrubbing generates ownership in the data and the schedule outcomes across team members.

This is their schedule and the project manager is their planning facilitator, but it is clear the team members own it. The project manager does not touch the schedule without team members present. They use it for daily management of tasks, because it is up to date and reflective of what they are working on. They use it to predict future events and to evaluate project trends, again because it accurately reflects what they are working on. They use it to identify the interface points they have with other groups and what inputs they need to get from them and when they are needed to stay on schedule.

In general, they spend far more time planning than normal teams. They accept the investment of time because they know what kind of return they get from their time investment. Visibility, transparency, early warning, low overhead reporting are all the benefits they get from putting in the time with the project manager to accurately build the plan and then to scrub it continuously, so it remains accurate.

3.13 Market/customer driven versus resource constrained

Summary

Best product teams/organizations plan new product development projects without initially using constraints (time, cost and resources). They focus on the market window or a driving customer's timing. They work out the optimal schedule and then evaluate the gap, then use techniques to close the gap. Then they apply constraints to the optimized schedule. This provides more gap data in terms of resources and cost. Similar techniques are used to close these gaps.

Normally, plans are developed from the inside-out. The plan is a function of the available resources and money. This results in schedules that push past target dates, so the schedule is arbitrarily cut to fit the time available. The resource constraint is never exposed nor resolved. Later in the project it will return.

Normal

The plan is a function of the available resources. Duration is derived using resource driven estimates. If 1 person would take 10 days to complete a task, then 0.5 of a person would take 20 days. This kind of bottom up thinking is used to generate the schedule. It is called "resource driven" scheduling.

Working within the limitation of a constraint is a corporate mindset, *"I only have X therefore I can only do Y."* No one ever challenges the assumptions, they just accept them as is and as fact. This acceptance of constraints translates to other aspects of development. Space, power, cost are all typical design constraints. Engineers wonder why they can't solve problems when they accept what they have as a given and then try to solve within them. They know it can't be done, but have no way of communicating it without sounding like they are complaining, so they just accept it knowing that problems will always resurface later. To challenge the constraint assumptions is to not be a "team player."

All planning is done inside the constraints given to the team. "You have 12 months and $5M dollars and 15 people to do this project, now show me a schedule that gets me there in 12 months for less than your budget..." The team comes back with a 12 month schedule, at less than $5M, and all 15 people are fully utilized.

In the end, the project takes $7M, 18 months to complete, and consumes 30 people. Why the disconnect? Because they worked backward from the constraints. Plans naturally follow this pattern in normal environments. Work expands or also contracts to fill the time available for its completion. The problem started when planning from the constraint.

Teams talk themselves into believing it can be done, because it has to be done. The gap is never exposed. The people on the team know there is a gap so they remain skeptical of the planning and the commitments if forces them to make.

In order to address the market and customer timing demands, the team force fits the schedule using available resources. If they show their original resource-driven schedule, they get shot down by management. They are forced to live with their situation even though they know it has a high probability of failure.

Best

Ignore constraints, with one exception they recognize the market or customer timing requirement and call it the "target date." The target date and the schedule dates are two different things. One is when it is wanted, the other is when you are going to get it. *Want* and *Get* are not automatically the same.

The targets are then turned off and used later as a reference point during schedule gap analysis. They not only ignore the time constraint, but also the resource and financial constraints. They plan assuming they have the resources and skills they need. Most projects just get worse when you add in constraints. Even with no constraints, projects are usually late, so why start planning from them? Resource driven schedules simply show that the project can't be done in time, but this does not resolve the conflict. Why waste time making them?

This is not to say they don't factor in risk, complexity, resource efficiencies, and some degree of resource availability (no one is ever 100% available). These are constraints that make their way into estimating logic. These are factored into duration estimates and the number of contingency tasks they add to their schedules to account for risks and unknowns. They don't use the number of resources they have to plan, they define what work is needed to achieve an end state, determine how long it will take to get there, and then determine how many resources are needed to get the tasks done in that time frame. So it is a *"what is needed?"* versus a *"what do I have?"* estimating process.

What results from this is a plan that more realistically estimates the resources needed to get to market at a given point in time. If there is a gap, they model the schedule acceleration needed to close the gap. What usually happens is that more tasks are done concurrently, more risks are taken, and more things have to happen on time in order to stay on schedule. The additional resources required are modeled in the accelerated scenario.

At this point it becomes a business decision; should we add more resources to accelerate the schedule? What is the ROI? or is there another way to do the project, such as outsourcing part of it that would reduce our resource/financial exposure and still achieve the project goals? This process is a logical decision-making activity based on realistic information about *How long it*

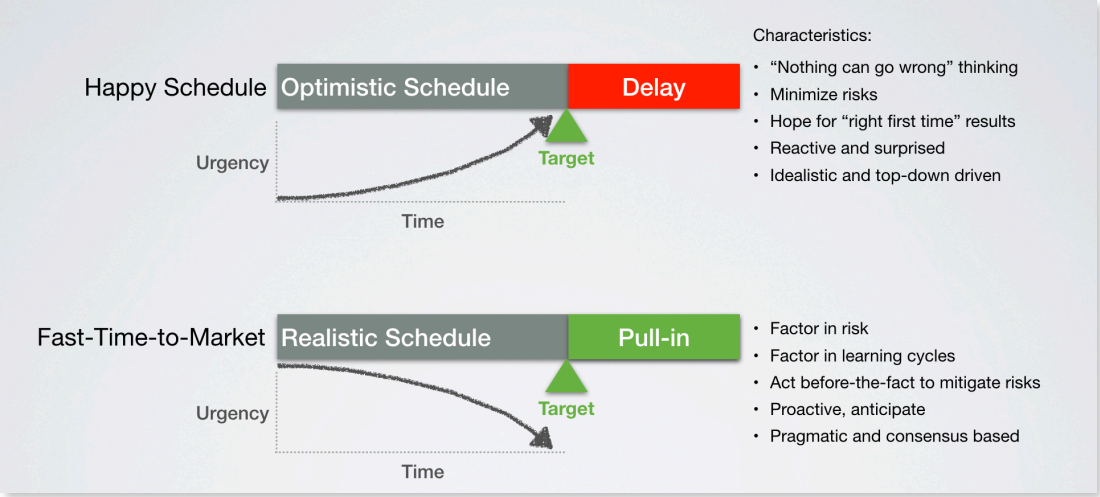
would take without constraints?, How long would it take with constraints? and *What are ways that it could be done differently if constraints were considered?* It is a scientific management process, rather than arbitrarily cutting and then closing ones eyes to reality.

If at the end of the day the gaps in schedule, cost, and resources can't be reconciled, then the project is killed before it starts, thereby freeing up resources to work on projects with a greater likelihood of success. Resources are not wasted on a project that will fail.

3.14 Know the gap; drives before-the-fact urgency

Summary

Best product teams/organizations identify the schedule gap at the start of the project. This is the gap between when a project is expected to be completed and when the schedule tells them it really will be finished. They use the gap to generate urgency today, while there is time to affect change. This drives changes in scope, outcomes, and methods in advance of a failure to improve the project's chances of success.



Normally, schedules are manufactured to meet the demands of the audience. They show the audience what they want to see. These are "happy" schedules, since everyone is happy that the project will finish on time. The sense of urgency is low. This only delays the inevitable when the project naturally slips later on.

Normal

Happy schedules are the norm and can be career enhancing. The corporate culture demands that they be produced, communicated, and religiously bought into by the executing project team. Everyone knows to "get on board" and be team players. It they think positively then good this are bound to happen.

There is no schedule gap in these environments, because if a gap was to be recognized and acknowledged by management, then the slip would be come self fulfilling, so the thinking goes. To avoid having self fulling prophecies becoming reality, teams generate schedule that fit nicely inside the time window they are given. This is the no-stress form of project management. At least no stress today.

It is less stressful to take the position to *not* argue that you may be late. It is seen as negative energy, you that you are a can't-do team player, and someone who can only find faults and not solutions. With enough pressure from above even the most resilient manager will buckle under the pressure to conform to this obfuscating behavior. Managers are encouraged to make aggressive schedules and then drive their teams harder to accomplish them. When they fail, they are replaced in the harsher cultures. People see examples of this and this reinforces the culture of happy schedule creation.

The schedule does not change in these places. Every week the team is on schedule. A fluctuating end date is cause for concern and could bring a load of scrutiny to the team. So the best way to avoid this attention is to report that everything is fine. It is fine at the start and it is fine most of the way through.

The problem with this game is that eventually the truth catches up and reality can't be ignored when a major deliverable is to be demonstrated. When it fails to materialize, the project reviews start and the team is inundated with "help" and lots of questions about how this could have happened and who is at fault for the mistake? A new committed date is published that is slightly later than the original date, because the team knows that if they really tell the truth, the new much later date would not be acceptable (same problem when they tried to tell management what the real date was at the start of the project). This ridiculous schedule game continues over and over again. It is painful and career limited for most involved. These organizations experience a lot of turnover and don't know why. They assume it is caused by "communication problems," so institute more team building off-sites and training classes.

Best

Fast teams and environments know the difference between a target and a schedule. The target is the constraint or goal, while the schedule is what is needed to get there. If there is a gap (usually is) they recognize it and embrace it as a tool for generating urgency at the start of the project, rather than at the end when there is no time to recover.

This is a before-the-fact management system that rewards people/teams who honestly plan by factoring in risks and unknowns. *"Let the truth set you free"* one manager exclaimed to his teams.

The gap is encouraged by senior management as a tool to be used to explore alternative strategies to do the project on time and per customer or market expectations. It forces people to think differently and to make the tradeoffs, they must, in order to fit the schedule inside the time window available.

Could an initial minimum viable version be released in the time available, then quickly followed by a product refresh a few quarters later with full functionality? These are the types of questions people start asking when they are presented with the reality of a schedule gap. The bigger the gap, the more open they are to challenging the base assumptions about the project. When there is no gap and the schedule is a happy one, people are not forced to think about the project differently. They figure they can get everything they want within the time frame they want to get it.

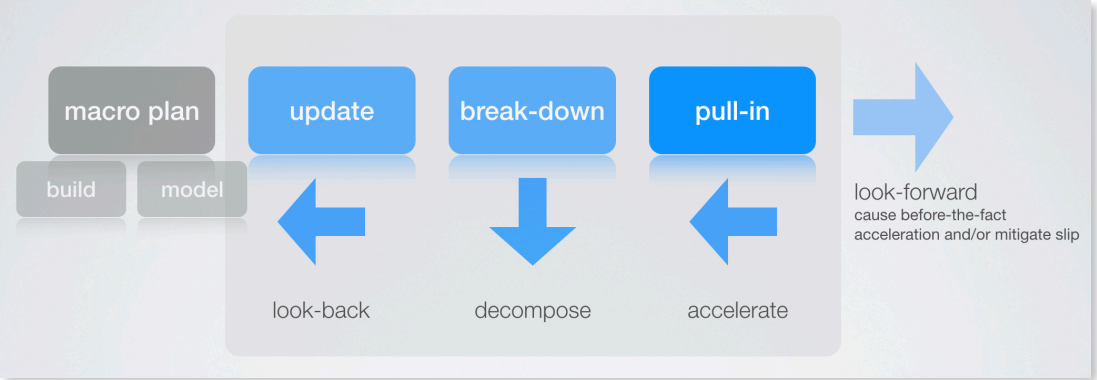
Ten pounds of flour will not fit in a 5 pound sack, so why pretend that it will in order to make people feel good at the start of a project? In the end, those 5 extra pounds are going to come spilling out onto the floor.

Why not use the planning process to expose this problem early, while there is still time to approach the problem differently? This is the culture of best teams who get to market on time. It is critical that this thinking be in place at all levels of management, because the honesty and transparency it creates can quickly be killed by the top manager who slams the team for showing him/her a schedule that does not finish on time. The team will quickly revert back to presenting "fake" schedules to avoid the pain. They also start to separate themselves from the project's outcomes, which over time become the senior manager's outcomes, not theirs.

3.15 Refresh Planning; continuous schedule pull-in

Summary

Best product teams/organizations refresh their schedules every week, sometimes multiple times in a week. They update it with actual task progress, they add changes to the workflow, and they try to pull-in the schedule when it slips to recover lost time and when it is on time, they try to accelerate it even further to bank time for when the unknowns pushes the schedule back out again. This process is relentless and continuous throughout the project. It is the key to being on time.



Normally, schedules are not updated with what is really happening on the project. The schedule falls out of synch with what is happening on the project and can no longer be used to predict deliverables and timing. It is somewhat maintained, but not relevant as a real management tool.

Normal

The schedule is created at the start of the project by the project manager. Very little team input is solicited, because the team members don't have time to plan - they are already late and need to get on with the design effort - when design competes with planning, design always wins.

The project manager makes her/his best attempt to to map out the workflow and estimate the time it will take to do each work package. The good ones with technical subject matter expertise can come close to approximating the project plan on their own, but then it lacks team input.

When done, the plan is presented to the team as "their schedule." The team quickly sees the deficiencies in the schedule, but does not have the time to sit down with the project manager to fix the problems to make the schedule reflect what they are really going to do. Some emails are shared and comments made to the project manager. It is up to her/him to make the corrections. All this takes too much time and besides, the project just changed direction again and the schedule that was made earlier is now even more out of date. No one has time to bring the

schedule up to where project has moved. There is now only a distant relationship between the plan-of-record and the actual project direction.

The schedule is rarely used by the team because it is not up to date. It does not reflect what they are doing so it is of little use to them. Rather, its value is as a management reporting tool to distract management into believing that the project is really complex and that they should not ask too many questions - they are continually told, *"It is on schedule, don't worry."* They do worry, which is why they continue to micro manage the team, but few actually look at the schedule.

Best

The schedule is created by the team, scrubbed, and the gap analyzed. Alternative paths to completion are modeled and decisions made about how to position the project to improve the team's chances of success. It is an open, transparent, and honest scientific process.

The baseline schedule is refreshed weekly, sometimes daily by the core team. They tell the schedule what actually happened and re-estimate tasks that are not complete, based on new information generated by working on the tasks. They bring all tasks up to date to the current status of the project. They always know where they are because every update causes the schedule to match what is happening on the project. This is a low overhead effort, taking 30-60 minutes for each weekly team refresh meeting.

Once the schedule is updated so it reflects the current state of the project, a new critical path is calculated to the end date. This usually reflects a slip in the schedule, because most projects will always move to the right if not pulled-back. The core team models different pull-in scenarios and determines the best actions to recover some of the lost time or even accelerate the project ahead of schedule in order to bank time for the future, knowing they will withdraw from the time-bank when the unexpected happens.

More sophisticated teams will peel back the critical paths to determine the longest path and the paths immediately behind the longest paths. This focuses team attention on reducing multiple paths, not just the longest. They know that any one of these shorter paths just behind the main path could bite them if they don't try to accelerate it. They focus on 10-20 of the paths behind the critical path, attempting to accelerate all of them. Unlike normal teams that assumed and hope that the 10-20 secondary critical paths get longer (they assume the people on the critical path will have problems and their schedule will get longer, thereby giving them more time), fast teams expect the critical paths ahead of them to get shorter which means that the full team is pulling-in. They assume acceleration versus slip.

They call this process Refresh Planning. It generates smaller sub-group meetings of 60 minutes or less during the week to further refine the schedule, breakdown macro tasks to micro tasks, and further pull-in at the micro level.

The Refresh Planning Process connects the team to the plan. It is a living document that at any time can tell an accurate story of the project's history, where it is now, and somewhat predict its future. The more they do it, the more they own it and are familiar with not just their own work, but with the work of other team members and how they all interface with one another.

The schedule is alive, real, and accurate in forecasting future events (of course the degree of accuracy is a function of the risks and unknowns that are factored into the plan).

Appendix

Start of the best practice study project (background)

Current: Since initiating the study, lateralworks has continued studying fast teams around the world. The best practice database is continually being updated with new findings and insights. lateralworks consulting teams have worked on hundreds of projects since the initial study and have applied many of these practices with many clients in a variety of cultures. Each time learning about how best to adapt and deploy the practice with teams and organizations.

From the initial published study

Our study of the best practices of highly successful new product development teams began in the early 1990's. LateralWorks conducted numerous interviews with executives, mid-level managers, and new product development team members representing virtually every line and staff function. The majority of those interviewed were in positions that directly or indirectly affected new product development initiatives.The companies in the study operate in highly competitive niches of the high technology market, where time-to-market (TTM) is perhaps the most essential variable to manage. The research for this project continues to be ongoing.

15 original study participants:
Amdahl Corporation
LSI Logic
Cadence Design Systems
Compression Labs
Quantum Corporation
Sun Microsystems
Cypress Semiconductor
Tektronix
Philips Semiconductors
Tandem Computers
3Com Corporation
Philips Electronics
Network Equipment Technologies
IBM Corporation
Conner Peripherals (Seagate)

The practices that were distilled from the interviews are specific and implementable. The intent was to isolate the specific practices of the most successful organizations and teams, practices

that were believed to have both a direct and indirect impact on the ability to deliver the right product at the right time to maximize profit.

The 15 companies that participated in this first phase of the study all designed, developed and manufactured high-technology products. There was a wide range of project types, from low-volume (one-of-a-kind manufactured products) with long sales lives to high-volume commodity products with very short sales cycles. Systems companies that participated in the study developed products that required forward and backward integration into their current systems, while other companies made stand-alone component products. The practices reflected in this study are common to all.

Framework within which the study findings are organized

In the process of analyzing the information collected during the interviews, it became clear that development speed, while essential, must be "woven" with the continuously changing needs of the customer throughout the development cycle to be successful. The product must not only arrive on time but must also be the "right" product when it arrives. Recognizing that the customer's product requirements will evolve during development, and responding quickly to those changing needs, is a practice of those teams/organizations that highly successful. Those teams that "froze" specifications early and were not in regular contact with the customer’s changing requirements were clearly less successful. We also found that the most successful development teams chose to seek product requirements from the most suitable dominant Tier 1 customer in a particular segment of the market.

As a result, LateralWorks has organized this best practice study according to three concepts: Right Customer, Right Product, and Right Time. New products succeed most when they successfully address all three. In today’s competitive market, two out of three isn’t enough. Giving our clients the tools to hit the “sweet spot” where all three intersect is the prime goal of LateralWorks.

Right Customer

Highly successful companies base their business plans on the market. They study market segments and carefully choose which ones to compete in, looking for the most suitable dominant Tier 1 customer in a particular segment—the Right Customer.

The Right Customer’s product requirements are the ones to zero in on. If the dominant customer buys the product/solution, the Tier II customers are more likely to jump on the bandwagon and buy it too.

Derive product requirements from a Tier II customer, and the product will likely appeal only to the Tier II customer, and end up capturing a small share of the market—if any. LateralWorks has developed methodologies that help companies focus on the Right Customer:

- Engage with customers directly to define and develop the product
- Analyze market segments in-depth to position products precisely
- Put the emphasis on finding and meeting customer requirements, rather than letting the product's potential capabilities drive the process

Right Product

Simply stated, the Right Product is the one that precisely meets the Right Customer's product requirements. But determining those requirements and translating them into product specifications is a complex process, and businesses don't always put in enough effort. The problem: it's possible to wind up spending money and time developing a product that has a fabulous array of features—which the customer doesn't turn out to need. Designing unnecessary features can also lengthen the development cycle, causing the product to miss its window of opportunity. LateralWorks has a proven process for taking the guesswork out of determining the Right Customer's real product requirements:

- Identify the right people in the customer organization to ask
- Develop questions that get to the heart of what they need
- Find out what the customer's own customers actually want
- Obtain the customer's involvement throughout the entire development cycle, to keep up with their changing requirements
- Quickly translate the customer's priority requirements into product specifications
- Convince engineers to buy in to those product specifications

Right Time

Product lifecycles are short and steep, and there's a great deal of competition from Asian and now third world countries--especially in the high technology sector. Therefore, entering the market at the Right Time is critical to a company's survival. This almost always involves being first to market, at the time when demand is just beginning to accelerate. By entering the market at the Right Time, companies can sometimes contract to sell at the highest average selling price for an extended period of time, avoiding the price degradation that inevitably occurs over time. LateralWorks provides tools that turn the schedule from wishful thinking into a dynamic tool for managing the process, reducing time-to-market without compromising other requirements:

- Ensure that the organizational structure supports fast teams
- Know the cost of being late to market, and use that cost-of-delay as the primary driver for speed.
- Implement fast decision-making procedures that authorize fast starts
- Make schedule KING
- Create fast schedules and rigorous tracking systems

Methodology for initial FTTM Best of Practices Study

The study sought actionable factors which discriminate between low performance fast-time-to-market (FTTM) versus high performance.

Several analytic techniques are available to identify factors which could potentially improve performance in a given endeavor. These include task analysis, expert judgment, and more recently, competency analysis. Task analysis involves describing all the tasks associated in doing the work. Expert judgment involves invoking the experts opinion on what factors theoretically should have most impact on job performance.

Competency analysis seeks the factors by analyzing critical incident interviews from high performance cases versus lower performance cases. To make the contrast more graphic the extreme ends of the performance curve are preferred. The interview does not inquire about what a performer would do in general or would recommend, but rather inquires about what the performer actually did in a recent incident.

The strength of competency analysis to identify factors for improving performance in the work place has been shown to be about five times that available through task analysis and expert opinion (R2=.49 versus .09) Recognizing the strength of this tool, many companies and the American Management Association have used competency analysis to find the discriminating factors to improve many aspects of work and management. We believe we have built a first version of a competency model for fast-time-to-market and expect to refine the model in each future data gathering phase.

FTTM was operationally defined as Right Product - Quick. Effectiveness measures were:

- Right product was measured by meeting targeted R.O.I.; and
- Quick was measured by reductions in start versus finish time consisting of two components; gestation time prior to team formation and overruns compared to initial project schedule.
- The data in this study consisted of competency based interviews of about 95 Executives, Managers and key independent contributors involved in new product development projects. The conclusions about factors were based on separating 13 high performance cases from the other 82 using the evaluation criteria of the companies and participants themselves to classify the cases. We have not attempted to define any external or independent criteria to evaluate FTTM performance.

To identify the factors which distinguish the 13 high performance cases from the 82 others, we carried out the conventional non-quantitative ("brute force") discriminate analysis. This resulted in six factors and 30 sub-factors which we call Best Practices.

In addition, quantitative data were gathered from all 95 participants This descriptive data provides supporting information, but it was not used as the basis for factor identification. No quantitative factor analysis nor discriminate analysis was performed with this data. The factors found in this study are based on professional judgment and experience in developing competency factors.

No statistical significance is being asserted in the findings. We believe, however, that competency based analysis is by far the most powerful methodology available for identifying actionable factors and best practices to improve FTTM because it is based on what higher performers are actually doing. This lets us all learn from their best practices which is the value of bench marking.

Each set of discriminating factors is formatted as follows:

- Three factor clusters;
- Six discriminating FTTM factors; and
- Multiple best practices associated with the factors.

Note: Many observers find competency factors and practices "obvious". When readers find them obvious, this is an advantage because it means the factors will be more actionable. Obviousness does not detract from the fact that many equally obvious ideas do not discriminate better FTTM performance from lesser performance. So, the discriminating power plus their obviousness gives the FTTM competency factors and best practices their strength for improving FTTM performance.

"Right" customer information throughout the entire TTM cycle (gestation, development, overrun): Continuous and systematic give and take with the customer throughout the TTM cycle insures access to the "right" information.

Voice-of-the-customer (VOC): The essence of customer input. Understanding that the customer has become so critical to business success that specific technologies and tools have been developed to assess this information. The voice-of-the-customer implies that a new product team understands the degree of value the customer places on specific features and functionality of the product "to be." It also implies that the team can translate these values into specific requirements that delimit the boundaries of acceptability.

VOC schedule: A specific schedule of customer input throughout all TTM phases including planning, requirements, design, development, beta, and qualification. The best teams integrate specific customer involvement steps into their project schedules. The customer's involvement throughout the TTM cycle insures access to the "right" information.

Customer path: The "exact" experiences the customer undergoes in the defined segment/niche at a given moment in time. This also takes into account the customer's environmental context at the time it is being assessed.

Do-it-right-the-first-time (DRFT): The emphasis on Total Quality Management and Zero Defects has caused organizations to re-evaluate old paradigms that assumed first pass perfection was not a realistic expectation, that rework was the norm and DRFT was largely accidental. DRFT requires that many more micro cycles of learning (do-it, try-it, fix-it) occur in an increasingly compressed period of time. DRFT also requires extensive use of benchmark or historical data to assist in predicting outcomes. Where data does not exist, it must be captured during the design-verify-test cycle and fed back to the process.

Install version control on external and internal specifications: A mechanism that delimits changes on existing specifications that are imposed within and outside of the organization. Typically this mechanism takes the form of an automated specification change-order-control system, in which all changes are recorded and are accessible by all members of the product team. Specific version control is required to manage the change authorization process.

Strategy of fixed containment of specifications: Sets absolute limits on the number of changes allowed. Specifications are "frozen" in order to minimize schedule delay caused by changing specifications. Rarely do development efforts that employ specification "freezes" meet eventual customer requirements, because customer requirements evolve during the life

cycle of the development project. The longer the development project, the more chances there are for the customer requirements to be something other than that which was originally "frozen."

Flexible containment of specifications: The flexible containment envelope delimits the extent to which the product team will make changes to the product design and/or process. Unlike the fixed containment process, which seeks to freeze all changes, a flexible containment envelope provides for versatility in dealing with the changing requirements from the customer during the design/development cycle. However, the envelope does have limits, which insures "change" has some boundary.

Delineating a path of early commits proceeding to late commits and contingency planning for high risk modules: Identifying on a time continuum when changes must take effect for a given portion of the product development. Contingency planning for high risk modules means identifying the back-up actions needed for those modules or interfaces that are identified as high risk. For example, during the development of an application software system, there are "early commitments" that must be made such as architectural, language, and database decisions if the project is to proceed to later phases of development. There are also "late commitments" decisions such as user interface and output requirements that can be delayed until later stages of the development effort.

Commit ratio: The relationship between those resources that are committed to approved projects versus those resources that are available for new product development projects.

A stream of incremental derivative products based on a given platform: A series of succeeding product concepts/designs that are derived from the previous design -- all based on a common "platform" or base product architecture.

Rapid incrementalism in technology advancement: Requires that only short intervals of time elapse between the release of succeeding product increments, so as to limit the profit problems associated with cannibalizing your own products (i.e., replacing your own products with a competitive offering).

Project gestation: The period of time from the origination of a new product concept to the time when a core product team is formed to start working development. This time frame offers a major opportunity for reducing the TTM cycle.

Default to action: When faced with a problem or decisions, key players and/or product teams assume responsibilities and take action rather than waiting for top management to make a go/no go decision.

Strategic by pass: An effort to resolve a product-related issue in which the existing hierarchy or information flow was by passed in order to obtain a quick answer and/or decision.

Macro schedule: A broad, high-level schedule that summarized the specific action plan of the project. Cost and resource allocates are also summarized in this plan.

Micro schedule: Detailed schedule. Typically "micro schedules" are derivations of and roll up to a "macro schedule." Micro schedules are defined as having individual activity durations of less than ten days. The micro schedule is constructed using precedence relationships to explain inputs and outputs between given activities. Additionally, each activity contains cost and resource allocation information.

	Portfolio	Environment	Execution		
	Prioritized Aligned & Managed	Development Life-Cycle Organization Structure	Right-Product	Right-Team	Right-Time
1.0 Mindset	1.1 Grasp enormous financial impact late products have on profitability and growth				
	1.2 Create fast culture through close-in oversight of product roadmap				
	1.3 Systematically dismantle slow structures				
	1.4 Accelerate the threshold of pain; reward early warning behavior				
	1.5 Fast decision-making system				
	1.6 Cost-of-delay is understood at individual contributor level; used to drive urgency				
2.0 Host	2.1 Fuzzy-front-end is managed	2.6 Fast development framework		2.8 Cross-functional core teams (heavyweight & dedicated)	
	2.2 Projects are prioritized based on business objectives	2.7 Provision teams, eliminate interrupts		2.9 Project-based performance measurement	
	2.3 Projects continuously aligned with resources and skills available			2.10 Freedom Level 1 empowerment	
	2.4 New products are tracked to break-even				
	2.5 Kill projects that fail to meet objectives; early				
3.0 Team	3.1 Co-develop with tier 1 customers; continuously refine requirements	3.2 Schedule as driver vs. reporter	3.3 Translate customer requirements to product definition, reconfirm throughout development	3.8 Cross Functional Core Team ROLES not FUNCTIONS; lateralized	3.10 Aggressive planning
			3.4 Know what they value, when they want it, how much they will pay for it	3.9 Get best people; mix of inside & external SMEs; virtual teams	3.11 Macro-micro planning; micro near, macro far, short interval scheduling
			3.5 Understand customer wants vs hows; prioritize the drivers that maximize customer satisfaction		3.12 Team planning, simulation, continuous scrubbing, breakdown, ownership
			3.6 Start design while refining requirements; never freeze specs		3.13 Market/customer driven versus resource constrained
			3.7 Fail-Fast; integrate/proto early, frequent & many do-it-try-it-fix-it learning cycles		3.14 Know the gap; drives before-the-fact urgency
					3.15 Refresh Planning; continuous schedule pull-in

Practices (list form)

1.0 Mindset

- 1.1 Grasp enormous financial impact late products have on profitability and growth
- 1.2 Create fast culture through close-in oversight of product roadmap
- 1.3 Systematically dismantle slow structures
- 1.4 Accelerate the threshold of pain; reward early warning behavior
- 1.5 Fast decision-making system
- 1.6 Cost-of-delay is understood at individual contributor level; used to drive urgency

2.0 Host

- 2.1 Fuzzy-front-end is managed
- 2.2 Projects are prioritized based on business objectives
- 2.3 Projects continuously aligned with resources and skills available
- 2.4 New products are tracked to break-even
- 2.5 Kill projects that fail to meet objectives; early
- 2.6 Fast development framework
- 2.7 Provision teams, eliminate interrupts
- 2.8 Cross-functional core teams (heavyweight & dedicated)
- 2.9 Project-based performance measurement
- 2.10 Freedom Level 1 empowerment

3.0 Team

- 3.1 Co-develop with tier 1 customers; continuously refine requirements
- 3.2 Schedule as driver vs. reporter
- 3.3 Translate customer requirements to product definition, reconfirm throughout development
- 3.4 Know what they value, when they want it, how much they will pay for it
- 3.5 Understand customer wants vs hows; prioritize the drivers that maximize customer satisfaction
- 3.6 Start design while refining requirements; never freeze specs
- 3.7 Fail-Fast; integrate/proto early, frequent & many do-it-try-it-fix-it learning cycles
- 3.8 Cross Functional Core Team ROLES not FUNCTIONS; lateralized
- 3.9 Get best people; mix of inside & external SMEs; virtual teams
- 3.10 Aggressive planning
- 3.11 Macro-micro planning; micro near, macro far, short interval scheduling
- 3.12 Team planning, simulation, continuous scrubbing, breakdown, ownership
- 3.13 Market/customer driven versus resource constrained
- 3.14 Know the gap; drives before-the-fact urgency
- 3.15 Refresh Planning; continuous schedule pull-in

Notes: