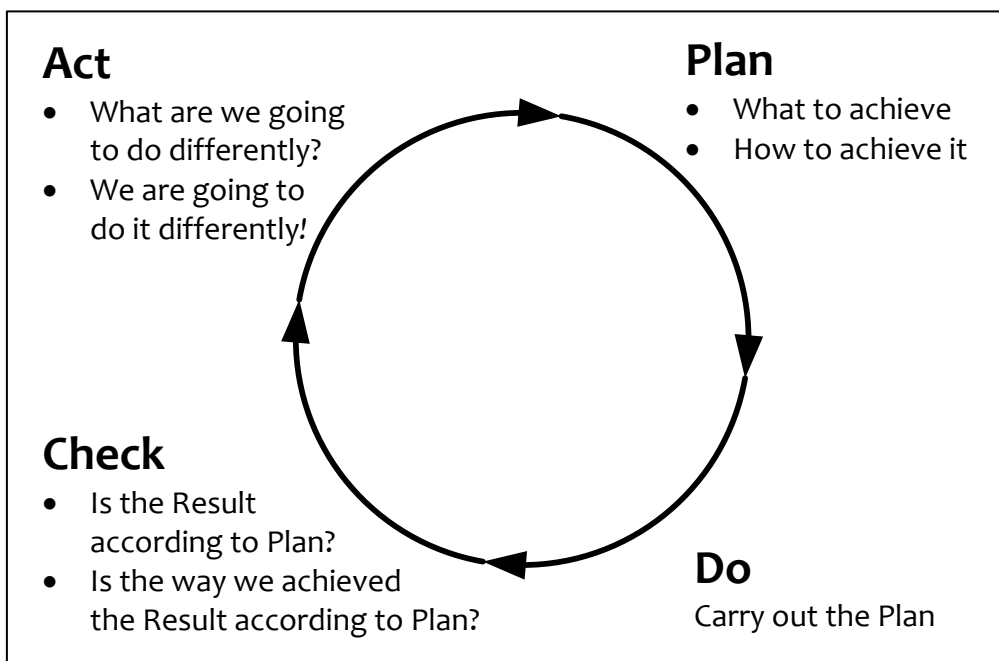


Niels Malotaux

Predictable Projects

How to deliver the Right Results at the Right Time



Predictable Projects

How to deliver the Right Results at the Right Time

1 Can the result and the delivery time of projects be predicted?

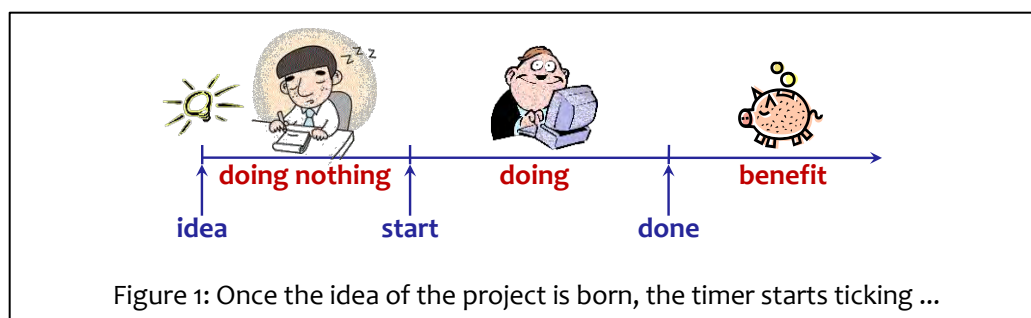
This text describes a very pragmatic approach that can quickly make your projects more successful in significantly shorter time, without bad stress, at the same time being able to predict what will be done when, with quite remarkable accuracy.

Does this sound impossible? That's what many people thought before they tried. Does it sound incredible? After all, you have been working in projects for many years. You have all the diplomas and experience. What can it be that you missed, that will suddenly make your projects delivering better results much faster, while everybody knows that projects easily take longer than hoped for? I almost always hear reserves like: "It can't be done", "We've tried it before", "I've heard that before", "It won't work here", "Our projects are different", "We build laaaaaarge systems", or "I'm doing these things already". These arguments are quite understandable, because people simply cannot imagine that there are techniques to deliver better results in substantially less time. Otherwise, they would be using these techniques already, wouldn't they?

There are many theoretical processes that promise great results, however, only practice shows whether it works in the real world. Based on decades of research and a lot of experience, the approach as presented here has been tested and honed in the practice of small and large projects, in several different countries, cultures, and disciplines. Even if you don't believe it, if those people who did it became some 30% more productive within *several weeks*, isn't that a good reason at least to try? The proof of the pudding is in the eating. Before you taste the pudding, it's not easy to explain the taste. However, the taste can be quickly tested. If your team becomes some 30% more productive, it means that they can produce some 50% more value in the same time, with the same people.

We'll first describe some background, then the different elements of the Evolutionary (Evo) Planning approach, followed by a more elaborated description of the first element of Evo Planning. Finally, we invite you to do a five-week exercise to get hands-on experience, because, believe it or not, there is much more detail in it than you can imagine when you just read this text. Further background and elaboration can also be found at www.malotaux.eu, while at www.malotaux.eu/downloads several booklets and presentations, as well as videos at www.malotaux.eu/conferences, can be freely downloaded. The author invites you to discuss, to agree, to disagree, or even try to get some taste yourself by some coaching. Email: niels@malotaux.eu

2 The importance of time and predictability



Every day we start a project later, we will finish a day later (Figure 1), depriving us and/or the customer from the revenues which by definition are much higher than the cost of the project, otherwise we shouldn't even start the project. The only good reason why we don't run this project yet is that we currently are spending our limited resources on more rewarding projects.

Initially, every day of the project adds potential value, but gradually we arrive in the area of diminishing returns where every extra day adds less than the return we should expect. This means that delivery time should be appropriately *designed* for optimum benefit. After all, the Project Manager is *responsible* for delivering the right result at the right time, but all people in the team *determine* the time it takes to deliver the appropriate system or service.

Engineers often think that the project isn't finished until "all" requirements have been delivered. Well, if delivery time is also a requirement, and often the most important requirement, why are all other requirements treated as being more important than the most important one? The requirements from all stakeholders are often contradictory, so the design process is there to find an *optimum compromise* for the conflicting requirements.

A product manager at a telecom company told me that 2 people had been analysing a project to take 4 people 10 months to complete (Figure 2). They said, however, that the development time could be shortened by 4 months if they would invest in a system costing €200k, but that the business was not prepared to invest this amount of money. After all, the 4 months savings would cost the project only €160k. So why spend €200k to save €160k?

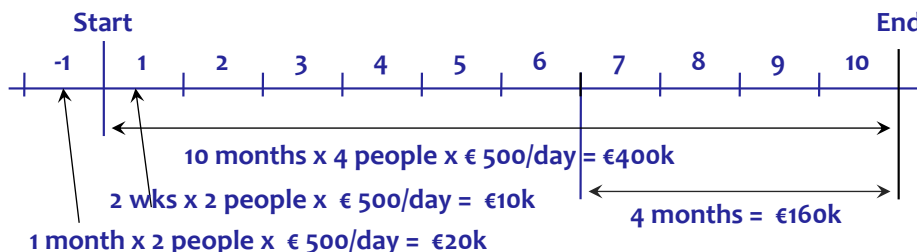


Figure 2: The importance of Time

The analysts said: "Give us two more weeks to investigate further to explain better why we should invest this €200k." The product manager complained that he felt that the project should start, and that no more time should be wasted on additional analysis.

I asked him: "How much do these people cost per hour?" Like most people in and around projects, he didn't know. I suggested: "How about €500 per day?" He nodded. "That means they spent some €20k (2 people, 20 days, €500), and they want to spend some €10k more to investigate more?" He said: "Yes, but I don't want them to waste more time!" Then I asked: "What will be the benefit once the project is finished?" The answer was: "The expected benefit will be €16M per year!" (Telecom must be like gold, but of course these companies don't tell us). "Aha! So, the extra 2 weeks don't cost €10k but rather some half million euro, and the investment to save 4 months doesn't cost €200k, but would rather secure some €5M extra benefit!" The product manager ran to the business people: "Invest those €200k, and start the project now!"

When I start coaching a project, one of the first things I ask is:

- "What is the cost of one day of (unnecessary) delay?" People usually don't know
- "What is the cost of one day of the project?" People usually don't know
- "What is your cost per day?" Most people don't even know that
(Note: what you cost is not what you get!)

Now, how can people make design decisions if they don't know the cost of time? The exact figure isn't even important. Any reasonable figure will do to make *better decisions*.

Once I was in a project asking these questions when I just saw, through the small window of the door, the boss passing by. I opened the door, and said to the boss: "Boss, these people don't know their cost, the cost of the project, nor the cost of one day of delay. How can these people make design decisions?" He said: "I don't know their cost, but I'll find out!"

An hour later he returned, saying: "€400 per person per day".

The benefit of the project should be huge, otherwise we should do another project. If people don't know the benefit, I usually suggest to assume that the benefit is about 10 times the cost of the project. Using this figure we calculated the cost of one day of delay: 5 people x €400 x 10 per day is €20 000. This is usually a lot more than anybody imagined, and it is a good basis to start making much better-founded design decisions.

¹ See also "The Fallacy of 'all' Requirements", www.malotaux.eu/fallacyofrequirements

3 Seven options we (seem to) have to be on time

As a lot of projects deliver late, there could be some interest to know how one can deliver earlier. There are several ways we see people use to try to finish a project earlier, most of which are intuitively right, but don't work. This contradiction causes people to think that we have to accept late projects as a fact of life. After all, they did their best (most people do!), even took measures (correct measures according to their intuition), and it didn't work out.

Deming said: "Doing your best is not enough." First you should know what to do, have an approach how to do it best, have a mechanism to continuously improve that approach, and then you do your best. There are, of course, also options that do work.

Deceptive options

Let's first do away with the 4 deceptive options. Deceptive options we see applied in almost every project, but they don't work, *on the contrary*: usually, they make things worse. It's surprising that people don't learn, and keep applying them.

1. Hoping for the best - fatalistic

If your past project took longer than expected, what makes you think that this time it will be different? If you don't change something in the way you run the project, the outcome won't be different, let alone better. Just hoping that this time your project will be on time won't help. We call this ostriching: putting your head into the sand waiting until Murphy² strikes again.

2. Going for it - macho

We don't have enough time, but it *has* to be done: "Let's go for it!" If nothing goes wrong (as if that ever is the case), and if we work a bit harder (as if we don't already work hard)... Well, forget it. If the time really is insufficient, it won't happen.

3. Working Overtime - fooling yourself, your customer, and your boss

40 hours of work per week is already quite exhausting. If you put in more hours, you'll get more tired, making more mistakes, having to spend extra time to find and "fix" the mistakes, half of which you won't. You think you are working hard, but you aren't working smart, and the result is less, at lower quality. After a while, you will be working overtime only to rectify the errors you created by working overtime. As a rule, never work overtime, so that you have the energy to do it once or twice a year, when it's really necessary.

4. Adding time - moving the deadline

Moving the deadline further away is also not a good idea: the further the deadline, the more danger of relaxing the pace of the project. This is due to Parkinson's Law³ and the Student Syndrome⁴. At the new deadline we probably hardly have done more, getting the project deliver even later. Not a good idea, unless we really are in the *nine mothers' area*⁵, where nobody can do it, even with all the optimization techniques available. It's better to optimize what we *can* do in the available time before the deadline. The only way a deadline may move is towards us. The earlier the deadline, the longer our future afterwards, in which we can decide what the next best thing there is to do.

Halfway a student project I asked the students whether they would conclude the project successfully, with a good working application. I got the four deceptive options within one minute: "I've a good feeling about it!" said one of the students (hope). "Besides, we have to be successful, so we'll make it happen!" (macho). "I'm working through nights and weekends to make it happen" (working overtime). "Sorry, what we promised last week is not yet working properly, we'll deliver some of it next week" (moving deadlines). Result in the end: useless application, customer not happy = failure!

² See "Murphy's Law for Engineers" www.malotaux.eu/murphy

³ 'Work expands so as to fill the time available for its completion' or, my translation: 'People use the time given'. Parkinson observed: "Granted that work is elastic in its demands on time, it is manifest that there need be little or no relationship between the work to be done, and the size of the staff to which it may be assigned."

⁴ 'Student Syndrome': Starting as late as possible, only when the pressure of the Fatal Date is really felt (attributed to E. Goldratt). When you had to study for passing an exam, did you work hard three weeks before, or mostly the night before? Be honest.

⁵ Some people think that if you take nine mothers, you can make a baby in one month.

5. Adding people - a risky option...

A typical move is to throw more people at the project, in order to get things done in less time. Intuitively, we feel that we can trade time with people, and finish a 12 person-month project in 6 months with 2 people, in 3 months with 4 people, or in 2 months with 6 people, as shown in Figure 3. In his essay *The Mythical Man-Month*, Brooks shows that this is a fallacy, a fairy tale, a myth, defining

Brooks' Law (1975):
Adding people to a late project makes it later

When I first heard about Brooks' Law, I assumed that he meant that we shouldn't add people at the end of a project, when time is running out, as many projects seem to only find out that they are late by the end of the project.

If time is running out, the added people have to learn about the project, and we have to explain to them, spending valuable time, which we cannot spend of the project itself.

The effect is, however, much trickier: if in the first several weeks of a project we find that the speed is slower than expected, and thus have to assume that the project will be late, even then, adding people can make the project later. The reason is a combination of effects. More people means more lines of communication and more people to manage, while the Project Manager and the Systems Engineer/Architect can oversee only a limited number of people, before becoming a bottleneck themselves. Therefore, adding people is not automatically a solution that works. It can be very risky.

In some project, people found out that the actual work was taking some 4 times the time they estimated originally. Apparently, while estimating, they forgot a lot of elements of work, like learning and testing. They tried to solve this by increasing the number of people from 5 to 20, to ramp up productivity. The productivity increased by only 50%. It took project management some 10 months to decide to decrease (against their intuition!) the number of people from 20 to 10. When they finally did, the net productivity of the remaining 10 people was the same as with 20 people. They have been paying 10 extra people for 10 months with no contribution to the productivity! This was an expensive exercise, but it will probably happen again many times at many places.

Putnam confirms Brooks' Law with measurements of some 500 projects (Figure 3). He found that if the project is done by 2 or 3 people, the project-cost is minimized, while 5 to 7 people achieve the shortest project duration at premium cost, because the project is only 20% shorter with double the amount of people. Because Time to Market is of often of huge economic value, this is probably still an economic optimum.

Adding even more people makes the project take longer at excessive cost. Apparently, the project duration cannot arbitrarily be shortened, because there is a critical path of things that cannot be parallelized. We call the time in which nobody can finish the project the *nine mothers' area*, which is the area where nine mothers produce a baby in one month. How can those mega-projects, where 100's or even 1000's of people work together, be successful? Well, in many cases they are not. They deliver less and later than expected, and many projects simply fail. There are only few companies left in the world who can design airplanes, at huge cost of time and money, and usually with huge delays. If you think Brooks' Law won't bite you, you better beware: it will! The only way to try to circumvent Brooks' Law is to work with many small teams, who can work in parallel, and who synchronize their results from time to time. We shouldn't ignore the Law. We have to learn to cope with it.

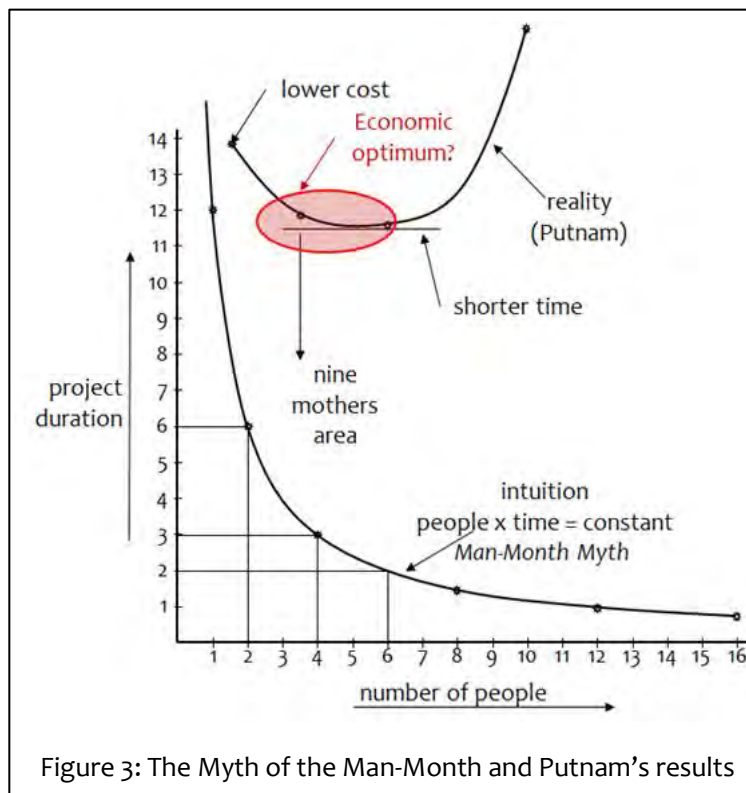


Figure 3: The Myth of the Man-Month and Putnam's results

6. **Saving Time** - The Measure that always works - The low hanging fruit

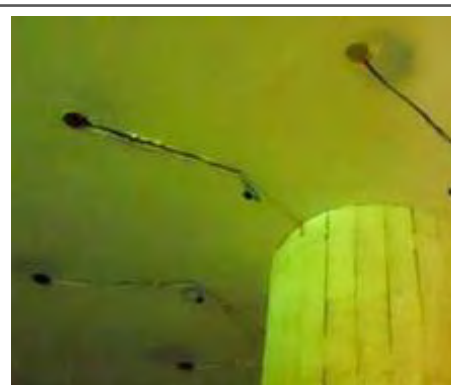
Fortunately, there are ways to save time, *without negatively affecting the Result* of the project, on the contrary! These techniques are collected, and routinely used, in the Evolutionary Project Planning approach as presented here, in order to achieve the best solution in the shortest possible time.

There are many dimensions of saving time:

- **Improving the efficiency of *what* (why, for whom) we do**
Not doing things that later will prove to be not needed. Because people tend to do more than necessary, especially if the goals aren't clear, there is ample opportunity for not doing what is not needed. We use the Business Case, Stakeholder Management, and continuous Requirements Management to control this process (not described in this text). Every week we decide what we are going to do, and what we are not going to do, before we do it. This saves a lot of time we now can use to do the things that *are* needed even better, and still deliver on time.
- **Improving the efficiency in *how* we do it:** doing things differently
This works in several dimensions:
 - **The product**
Choosing the proper and most efficient solution. The solution chosen determines both the performance and cost of the product, as well as the time and cost of the project, and should be an optimum compromise, and not just the first solution that comes to mind. We use short feedback cycles to check the requirements and assumptions with the appropriate Stakeholders.
 - **The project**
We can probably do the same in less time if we don't immediately do it the way we always did, but first think of an alternative and more efficient way. We do not only design the product, we also continuously *design and redesign the way we work*.
 - **Continuous improvement and prevention**
Actively and continuously learning how to do things better, and how to overcome bad tendencies. We use rapid and frequent Plan-Do-Check-Act (PDCA⁶ or Deming-) cycles to actively improve the product, the project *and* the processes. We use Early Reviews to recognize and tackle tendencies before they pollute our work products any further, and we use a Zero-Defect attitude because that is the only way ever to approach Zero Defects⁷.
- **Improving the efficiency of *when* we do it**
Doing things at the right time, in the right order. A lot of time is wasted by synchronization problems like waiting for each other, or redoing things that were done in the wrong order. Actively Synchronizing, and designing the order of what we do.

All of these elements are huge time savers. Of course, we can apply these time savers even if what we think we have to do easily fits the available time, in order to produce results even faster. We may need the time saved later to cope with an unexpected drawback, in order still to be on time, and not needing any excuse. Optimizing only at the end won't bring back the time we lost at the beginning. Optimizing only towards the end also means that there isn't much time to optimize anyway.

Imagine a plasterer plastering a wall in a new building. Then the electrician comes in, carving a furrow through the plaster in the wall to fit some electric wiring. The plasterer returns to repair the wall. Then the plumber comes in, cutting through the plaster and the electric wiring, in order to fit some water tubing. The electrician and the plasterer come back to repair the wiring and the plaster. If only these people made a TimeLine of their plans before they started, they would easily have seen in which order they should have done things in order not to repeat any of their work. They're not stupid. They only didn't think.



Furrows of the electrician, after the plasterer did his job

⁶ See chapter6, or “Plan-Do-Check-Act”, www.malotaux.eu/pdca

⁷ See “Zero Defects”, www.malotaux.eu/zerodefects

7. Killing the project

Of course, the faster we see that we never will get a positive return on investment, the faster we can kill the project, rather than after we've spent 3 times the budget, or worse.

4 The Goal

As the universal goal of any project, we use:

Delivering the Right Result at the Right Time, wasting as little time as possible (= efficiently)

Or, to keep it even shorter: **Delivering Quality on Time**

More formally, we use as the *top-level requirement* of any project:

Providing the customer with

- **what they need**
this is usually different from what they ask for
- **at the time they need it**
this could be earlier or later than they ask for
- **to be satisfied**
then they want to pay
- **to be more successful than they were without it**
if they're not successful, they *cannot* pay; if they're not *more* successful than before, *why would* they pay? Note that the success is ultimately created by the users of the system. Our project and the customer merely *provide the conditions* for the users to create the success

Providing the customer with

what they need
at the time they need it
to be satisfied
to be more successful than they were without it

Constrained by (win - win)

what the customer can afford
what we mutually beneficially and satisfactorily can deliver
in a reasonable period of time

Constrained by

- **what the customer can afford**
what the customer want, they cannot afford: if we start developing that, we may fail anyway
- **what we mutually beneficially and satisfactorily can deliver**
it should be win - win: customer is king, but we don't have to be slaves
- **in a reasonable period of time**
miracles take a bit longer

Better focus on what we are supposed to do saves time. Anything we do in the project should support this top-level requirement, otherwise it's waste. Who wants to waste time, producing waste?

5 Preflection, foresight, and prevention

Of course, *your* projects are different, but a lot of projects fail to deliver the right result at the right time. Cobb's paradox says: "We know why projects fail - we know how to prevent their failure - so why do they still fail?" Apparently, a lot of people do not know how to prevent their failure.

A lot of the things we have to do to deliver the right things at the right time are counter-intuitive. Intuition makes us react automatically on common situations. If intuition would be perfect, what we do would be perfect. Not everything we do is perfect, so apparently intuition sometimes guides us into the wrong direction. Intuition is a very strong mechanism, and it's very difficult to go against it. However, once we recognize (and acknowledge!) that intuition sometimes causes us to make counterproductive decisions, and inhibits proactivity, we can decide to actively do something about it.

Albert Einstein (1879-1955) seems to have said, although Benjamin Franklin (1706-1790) was earlier:

Insanity is doing the same things over and over again, and hoping the outcome to be different

(let alone better - my addition)

If we keep working the way we always did, we will keep making the same mistakes, and our projects will still be late. Only if we change our way of working, the result may be different.

Because "hindsight is easy", we can often use it to reflect on what we did, in order to learn: Could we have avoided doing something that now, in hindsight, proves to be wrong, unnecessary, or superfluous? Or could we have done it more efficiently? Reflection, however, doesn't recover lost time: the time is already spent, and can never be regained.

Foresight is less easy, but with foresight we can *imagine* whether what we're going to do later will prove to be incorrect or superfluous, and then decide *not* to do it. We also can *imagine* different ways to do it to see that it is done more efficiently than if we just would start and do our best.

Reflection is for hindsight is for learning, however, it's useless if we don't actively use what we learned with *Preflection*. Only with *preflection* we can foresee, and thus prevent wasting precious time. Isn't it weird that we have a word 'reflection', but not 'preflection', which is so much more important?

This is used in the Plan-Do-Check-Act or Deming cycle.

6 Plan-Do-Check-Act

One of the basics of Evolutionary Planning is the good old Plan-Do-Check-Act cycle (PDCA or Deming cycle) for continuous learning and improvement.

- **Do** is not a problem: we “do” all the time
- **Plan** we do more or less, usually less
- For **Check** and **Act**, however, we have no time because we are so busy, and we want to go to the next Do

Intuition is how people automatically react on situations, based on previous experience. Our sub-consciousness provides the suggestion how to handle the situation, and we do it immediately as we did it before. We call this the Intuitive Cycle or ‘PI-Do-PI-Do’, as shown in Figure 4. It’s not a conscious *Plan*, therefore we call it just *PI*.

Instead of following the *PI-Do* cycle, let's see how we actively can use the Plan-Do-Check-Act cycle better, as shown in Figure 5.

- First we **Plan**, which is twofold:
 - What we’re supposed to achieve (the product)
 - How to achieve it the best way (the project)

- Then we **Do** according to the Plan

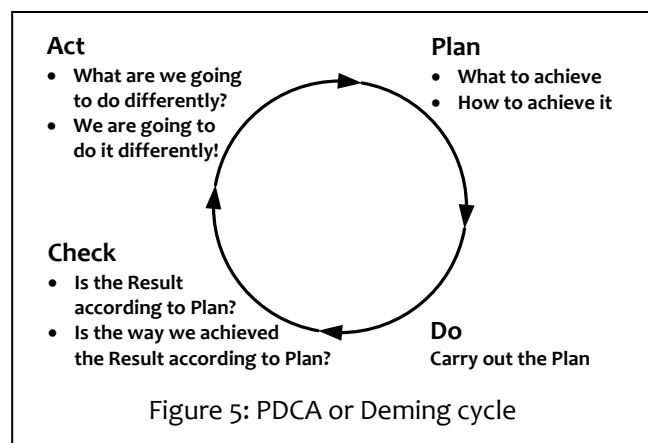
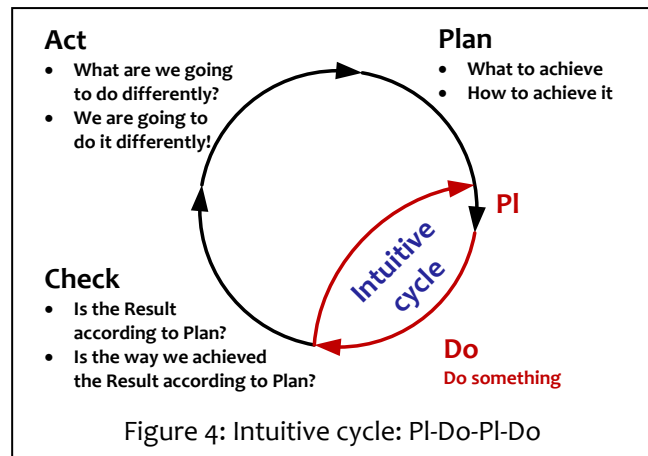
This is the first pitfall, because it means that the Plan must be Do-able, and we must follow the Plan. Let’s assume that this was the case:

- Then we can go to the **Check** or **Study** phase where we analyse:
 - Is *what* we achieved according to the Plan? - thinking of effectiveness
 - Is *the way* we achieved it according to the Plan? - thinking of efficiency
 - If yes: should and can we do it better the next time?
 - If no: should and can we do it better the next time?

- Then, in the **Act** phase we decide what we will do differently the next time

If we don't do anything differently the next time, the result will be the same. We do very small steps at the time, so that we can learn what is clearly better and what is not. Doing these small steps very quickly and frequently, we improve quickly and constantly *what* we are doing (the product), *how* we are doing it (the project), and even how we *organize* all of that (the process). Because in the Act phase we introduce *mutations* of what we do and how we do it, we call this the *Evolutionary approach* and because evolutionary is such a long word, we use the label *Evo*, for everything that works better than the alternatives we know.

People often quite well know what's not going so well, and easily say: “Yes, we should do that, but ...”: “management doesn't allow it”, “it can't be done”, “we’ve tried that before”, etc. If I hear these “Yes, but ...”s, I usually suggest: “You're stuck in the Check phase. If that really is a problem, what *should* and *can* we do about it?” The other person always immediately has a suggestion what we could do about it.



The problems we face are not the real problem. The main problem is that we have to move to do something about it. Not staying in the *Check* phase, but rather moving to the *Act* phase: *What are we going to do about it.* Not complaining what *cannot* be done, but rather thinking what *can* be done. Then you'll see that much more is possible than you thought.

7 Evaluations for fast feedback

Many organizations mandate a Project Evaluation at the end of every project, or at the end of every project stage. Some call it a Retrospective, or Post Mortem Analysis. Even so, few projects do the actual evaluation because they feel that these evaluations do not really contribute to better results. Why is this?

Consider a one-year project (Figure 6). People have to evaluate what went wrong and what went right (many things accidentally go right) and why, as long as a year ago. If something happened three months into the project, we have to remember it nine months later, and think how we could do things better.

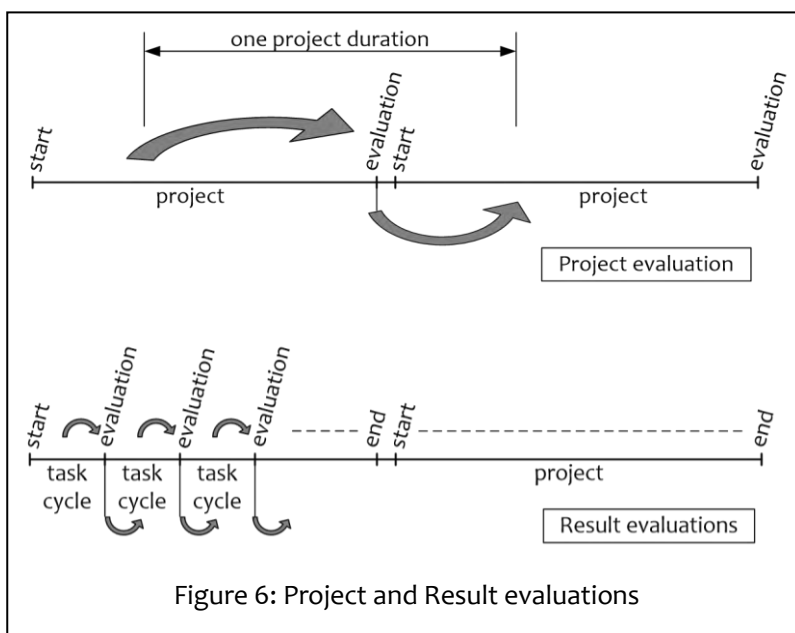


Figure 6: Project and Result evaluations

Three months into the next project, we have to remember that we were going to do something differently, hopefully better. Result: the previous project didn't benefit, because we thought about doing things better only after the project, and the next project doesn't benefit either, because we don't timely remember our evaluation decisions, as they aren't ingrained into our intuition yet.

The principle is right: as with PDCA, first we Plan the project, then we Do things, then we Check what we could have done better, and in the Act-phase we decide what to do differently the next time. But in practice it doesn't really help us. The time-frame isn't tuned to our human capabilities.

How can we tune this process, which is correct in principle, to actually work for us?

How about doing an evaluation *every week*? It is already difficult to remember what happened in the past week. Do you remember what you had for dinner 4 days ago? Most people don't (well, you don't really have to know, unless you're the cook).

If we evaluate every week, now PDCA starts to work for us:

- If we tune our mind to remembering what went wrong and what went right the past week, it's not too difficult to remember it for evaluation by the end of the week. Some people use a whiteboard or a post-it to catch an issue when it happens, so that they don't forget to evaluate it later. Some people evaluate immediately, not to keep doing something that can be improved immediately. After some time, it becomes a way of life.
- Not too much happens in one week, so the evaluation doesn't have to take long (unless it has to).
- Because most kinds of work take more than one week, we can try out our idea to improve immediately the next week. One week later we can check: Was the result better? If yes, should and can we do it better? If no, should and can we do it better?
- Because we actually do it differently the next week, the new way of working becomes internalized immediately, so that our intuition will hand it to us when we need it again later.

Now the evaluation is tuned to our human capabilities and needs, and it starts to work for us.

8 Evolutionary Planning

Evolutionary (Evo) Planning is designed to continuously improve our efficiency and effectiveness, not wasting time on things that nobody will need, and making sure that we will finish our endeavours ever more successfully in the shortest possible time.

In order to keep things manageable, we organize our work at several levels:

- Weekly **TaskCycles**, to optimize the way we work, improving our efficiency
- Bi-weekly (fortnightly) **DeliveryCycles**, to optimize what we deliver, improving our effectiveness
- **TimeLine**, to see:
 - what will happen if we work the way we are currently working
 - what to do if that won't produce the expected results on time, making sure we *will* be on time, no excuses needed

These elements together make our individual work more effective, efficient, and predictable, which subsequently can make our projects more effective, efficient, and predictable.

8.1 TaskCycles

In the TaskCycle we organize the work. We are checking whether we are *doing* the right things, in the right order, to the right level of detail. We are optimizing our estimation, planning and tracking abilities to better predict the future. We select only the highest priority Tasks, never do lower priority Tasks, and never do undefined Tasks. This improves our efficiency (= effectiveness in the least time).

As a practical rule, we plan 2/3 of the available time, in the remaining 1/3 of the time handling small interrupts like helping each other, project meetings, and many other things we also have to do. If we plan 100% of our available time, we will still do all those other things, hence we will never succeed in what we planned (sounds familiar?).

TaskCycles normally take one week, in some special cases less. Every TaskCycle we decide how much time we will have available, what is most important to do, how much time it takes to do it completely (we define what completely done means), and then what we can do in the available time. We also decide what we will *not* do in this Cycle, because there is no time to do it. Now we can focus all our energy on what we *can* do, making us more relaxed and more productive. Some managers fear that planning only 2/3 of the available time makes people do too little. In practice we see people do more. This is a typical example of counter-intuitivity.

8.2 DeliveryCycles

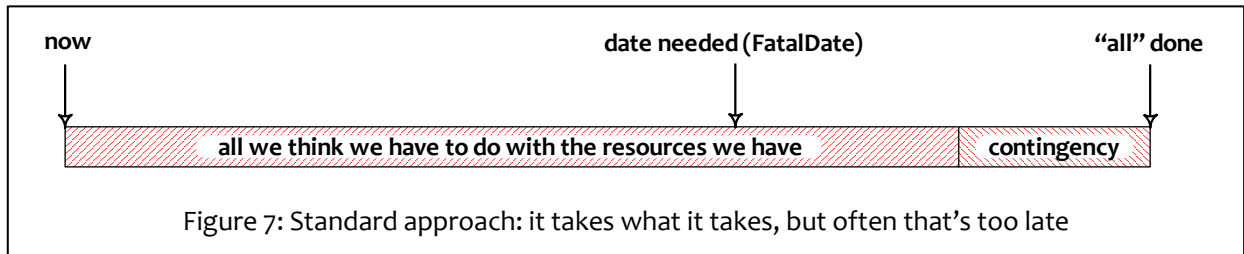
In the DeliveryCycle we organize Results to be delivered to selected Stakeholders. We are checking whether we are *delivering* the right things, in the right order, to the right level of detail. We are optimizing the requirements, and checking our assumptions. This improves our effectiveness.

A DeliveryCycle normally takes not more than two weeks. Novice Evo practitioners, almost without exception, have trouble with the short DeliveryCycle. They think it cannot be done. In practice we see that, without exception, it *always* can be done. It just takes some practice. One of the important reasons for the short length of the cycle is that we want to check our (and the stakeholders') assumptions before we have done a lot of work that later may prove to have been unnecessary, wasting valuable time. Short DeliveryCycles help us minimize risk and cost.

A common misconception of Deliveries is we always have to deliver to users or customers. On the contrary, we can deliver to any Stakeholder: the user or customer, ourselves, or any Stakeholder in between. This makes it easier to define Deliveries. However, we must always optimize Deliveries for optimum feedback: we must check what we are doing right, and what we are still doing wrong. Hence, for every Delivery we ask the question: "*What* should we deliver, to *whom*, and *why*?"

8.3 TimeLine

In many projects all the work we think we have to do is cut into pieces (some call this work packages, we call it 'chunks of work'), the pieces are estimated, and the estimates are added up, to arrive at an estimate of the effort to do the work. Then this is divided over the available people to arrive at an estimate of the duration of the work, which, after evaluation of interdependencies, and adding some contingency, is presented as the duration of the project (Figure 7).

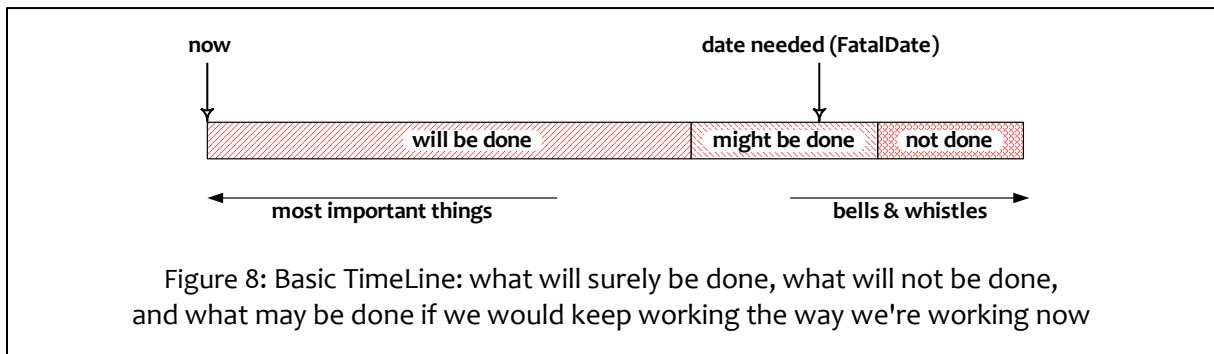


A problem is that in many cases the required delivery date is earlier. The tension between expected, and estimated delivery time causes extra discussions which cost even more time, however, the required delivery date doesn't change, leaving even less time for the project.

Because the delivery date is a requirement just as all the other requirements, it has to be treated as seriously as all other requirements. With TimeLine, we treat delivery dates seriously, and we meet these dates, or we very early explain, based on facts, why the delivery date really is utterly impossible to meet, because it's in the nine-mothers' area. We don't wait until the FatalDate to tell that we didn't make it, because if it's really impossible, we know it much earlier. If it is possible, we deliver, using all the time-saving techniques we have at our disposal, to optimize what we can deliver when.

TimeLine can be used on any scale: on a program, a project, a sub-project, on deliveries, and even on TaskCycles. The technique is always the same. We estimate what we think⁸ we have to do, see that we need more time than we have, then discuss the TimeLine with our customer or other appropriate Stakeholders, and explain (Figure 8):

- What, at the FatalDate, surely will be done
- What surely will not be done
- What may be done (after all, estimation is not an exact science)



If what surely will be done is not sufficient for success, we better stop now, to avoid wasting time and money. Note that we put what we plan in strict order of priority, so that at the FatalDate at least we'll have done the most important things. If Time to Market is important, customers don't mind about the bells and whistles. Because priorities may change quite dynamically, we have to constantly reconsider the order of what we do when.

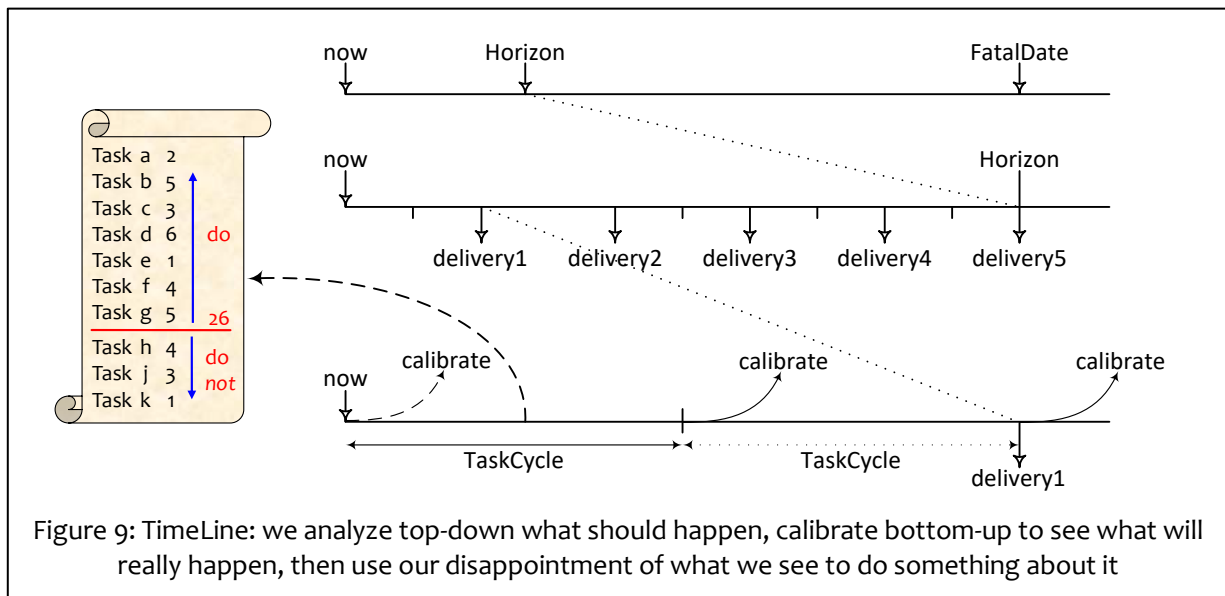
8.4 Setting a Horizon

If the total project takes more than 10 weeks, we define a Horizon at about 10 weeks on the TimeLine, because we cannot really oversee longer periods of time. A period of 10 weeks proves to be a good compromise between what we can oversee, while still being long enough to allow for optimizing the order in which we deliver results for optimum feedback. We don't forget what's beyond the horizon, but for now, we concentrate on the coming 10 weeks.

⁸ We keep saying "what we think we have to do", because however good the requirements are, they will change, because we learn, they learn, and the circumstances change. The longer the project, the more the requirements have a chance to change. However, what we don't know yet, we cannot plan for. So we keep improving our knowledge about the real requirements, but we only can plan based on what we currently think we have to do.

8.5 DeliveryCycles

Within these 10 weeks, we plan DeliveryCycles (Figure 9) of maximum 2 weeks, asking: "What are we going to deliver, to whom, and why?"



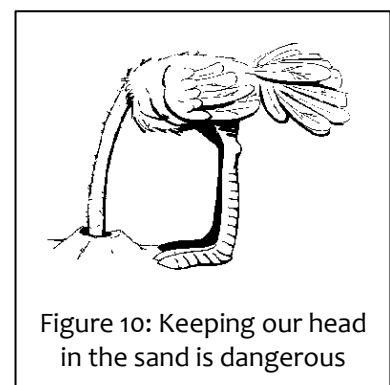
Deliveries are for getting feedback from Stakeholders. We are humble enough to admit that our (and their) perception of the requirements is not perfect, and that a lot of our assumptions may be incorrect. Therefore, we need communication and feedback. We deliver to *eagerly waiting* Stakeholders, otherwise we don't get feedback soon enough. If the appropriate Stakeholders aren't eagerly waiting, either they're not interested, and we may better work for other Stakeholders, or, if we really need their feedback now, we have to *make them* eagerly waiting by delivering what we call *Juicy Bits*. How can Juicy Bits have a high priority? If we don't get appropriate feedback, we will probably be working based on incorrect assumptions, causing us to do things wrong, which will cause delay later. Therefore, if we need to deliver Juicy Bits to Stakeholders to make them eagerly waiting, to get the feedback that we awfully need, this may have a high priority.

8.6 TaskCycles

Once we have divided the work over Deliveries, which are Horizons as well, we now concentrate on the first few Deliveries, and define the actual work that has to be done to produce these Deliveries. We organize this work in TaskCycles of one week. In a TaskCycle we define Tasks, estimated in net effort-hours. We plan the work in plannable effort time, which defaults to 2/3 of the available time (~26 hrs in case of a 40-hr week), confining all unplannable project activities like email, phone-calls, planning, small interrupts, etc, to the remaining 1/3 of the available time. We put this work in optimum order, divide it over the people in the project, have them estimate the time they would need to do the work, see that they don't get overloaded, and that they synchronize their work to minimize the total duration.

8.7 Top down – bottom up

Having estimated the work that has to be done in the first week, we have captured the first metrics for calibrating our estimates on the TimeLine. If the Tasks for the first week would deliver about half of what we need to do in that week, we now can extrapolate that our project is going to take twice as long, if we keep working the way we are, that is: if we don't do something about it. Initially the data of the first week's estimate may seem weak evidence, but it's already an indication that our estimates may be too optimistic. Keeping our head in the sand for this evidence is dangerous: I've heard all the excuses about "one-time causes". Later there were always other "one-time causes".



One week later, when we have the actual results of the first week, we have slightly better numbers to extrapolate and scale how long our project really will take. Week after week we will gather more information, continuing top-down and bottom-up, with which we calibrate and adjust our notion of what will be done at the FatalDate, or what will be done at any earlier date. This way, the TimeLine process provides us with very early warnings about the risks of being late. The earlier we get these warnings, the more time we have to do something about it. In practice the actual TimeLine process may entail a bit more, but the basics are as described.

Failure is not an option. The earlier we get warning signals of possible failure, the earlier we can start making sure that failure is not going to happen.

9 Evo practice example: TaskCycle planning

Not to make this text longer than it already is, I'll provide here as an example some more detail about the first thing we do when turning a project towards Evolutionary Planning.

Actually, the result of the TaskCycle planning is a to-do list for everyone in the project. Not the to-do list most of us already make from time to time. This to-do list is checked *before we do the work*, on feasibility (will it fit the available time?), priority (is this more important than anything else?), synchronization (does it fit with the order we should do things, and with the work of others), and any useful other aspects. Before we do the work, we can still optimize what we are going to do, and what not to do. Afterwards we can only complain that we wasted time because we didn't check beforehand. Because what we plan is doable, people say that from the first week all stress is gone. They work much more focused, know why they are doing just this, and why they are not doing other things. Productivity surges.

At the start of the weekly TaskCycle, this is what we do:

1. First determine the gross number of hours you have available for this project this TaskCycle

We define how much time we have available *before* we think what we have to do. If what we think we have to do takes more time than we have available, we tend to 'give' ourselves more time than we have, fooling ourselves, leading to failing to deliver as planned. The rule is that what we plan will be done, completely done, no excuses needed. Deduct all non-project time, or otherwise already planned time, like visiting a dentist, from your total available time.

2. Divide this gross number of available hours into:

- Available Plannable Hours (default $\sim 2/3$ of gross available hours)
- Available Unplannable Hours (default $\sim 1/3$ of gross available hours)

In a 40-hour work week, we usually use 26 hours plannable time, and 14 hours unplannable time. In many projects this proves to be realistic. If it's not, use any other value that works for you.

We only plan those Tasks that don't get done unless planned. If it's in your plan, you *have* time, and after that time, the Task will be done, completely done. During planning you define what 'completely done' means, and base the required time on that.

We do *not* plan Tasks that will get done anyway, even without planning. If they happen anyway, why waste time putting them in your planning? These are done in the unplannable time, or the time has already been deducted from the gross available time.

3. Define Tasks for this cycle

Focus on finding Tasks that are most important *now*, and don't waste time on less urgent tasks for the moment. Based on what we learn from current tasks, the definition of later Tasks could change, so don't plan too far ahead. Use the Delivery definition to focus on what to work on in the Tasks.

4. Estimate the number of effort hours needed to completely accomplish each Task

We always estimate net *effort* hours. If people estimate in days, they usually come up with lead time (the time between starting and finishing the Task). If people estimate in hours, they can quickly learn to come up with effort (the *net* time needed to complete the Task). The reason for keeping effort and lead time separate is that the causes of variation are different: If effort is incorrectly estimated, it's a *complexity* assessment issue. If there is less time than planned, it's a *time-management* issue. Keeping these separate enables us to learn.

Only the person who is going to do the Task is allowed to define the duration of the Task. Others may not even hint, because this influences the estimator psychologically. If others do not agree with the estimate, they should challenge the (perceived) contents of the Task, *never* the estimated time itself.

Ultimately, when we agree on the requirements of the Task, the implementer decides how much time (s)he is going to need, otherwise there will be no commitment to succeed. If there is no commitment, there will be no pain if we still happen to fail. If there's no pain, we don't learn.

5. Split Tasks of more than about 6 hours into smaller Tasks

We split the work into manageable portions. Estimation is not an exact science, so there will be some variation in the estimates. We are not bound by the exact estimated effort hours. We are only bound by the Result: by the end of the week all committed work is done. If one task takes a bit more, and the other a bit less, who cares? If you have several tasks to do, and the *average* estimated time is sufficiently accurate, the variations will cancel out. If you have a massive task of 26 hours, it is more difficult to estimate, and the averaging trick cannot save you anymore. Besides, 6 hours is about the maximum you can do in one day (2/3 of available time!).

6. Fill the available plannable hours with the most important Tasks

Never select less important Tasks. Always fill the available plannable hours completely, otherwise the unfilled time will evaporate.

7. Ascertain that indeed these are the most important Tasks to do, and that you are confident that the work can, and will be done in the estimated time

- Any doubt undermines your commitment, so make sure you can deliver
- Acknowledge that by accepting the list of tasks for this cycle means accepting the responsibility towards yourself and your team. By the end of the cycle these tasks will be done, completely done: no excuses needed.

At this point, you will have a list of Tasks that will get done (Figure 11). If you cannot accept the consequence that some other Tasks will not be done, do something! You could:

- Reconsider the priorities.
- Get additional help to do some Tasks for you. Beware, however, that it probably costs some time to transfer the Task to somebody else. If you don't plan this time, you won't have time.
- If no alternative is possible, accept reality. Hoping that the impossible will happen will only postpone the inevitable. The later you choose to do something about it, the less time you have left to do it. Don't be an ostrich: in Evo we take our head out of the sand, and actively confront the challenges.

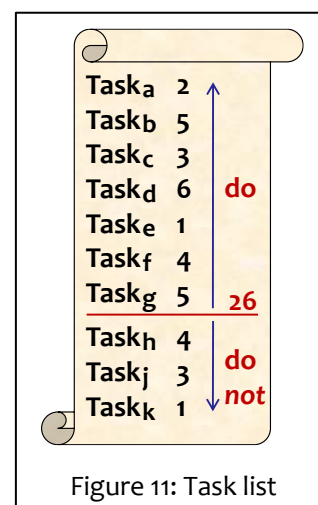


Figure 11: Task list

9.1 At the end of the TaskCycle we Check, Act, and Plan:

8. Was all planned work really done? If a Task was not completed, we have to learn:

- **Was the time spent on the Task, but the work not done?**
This is an *effort estimation* problem. Discuss the cause, and learn from it to estimate better the next time. What did you *think* first, and what do you *know* now, after having done it.
- **Was the time not spent on the Task, and therefore the work wasn't done?**
This is a *time management* problem:
 - Too much distraction or interrupts
 - Too much time spent on other (poorly-estimated?) Tasks
 - Too much time spent on unplanned Tasks
 Discuss the causes, and decide how to change your time management habits.
Not having all Tasks completed is expected the first few cycles, but the learning should help us within a few weeks to complete all TaskCycles as planned.

9. Conclude unfinished Tasks after having dealt with the consequences:

- Feed the disappointment of "failure" into your intuition mechanism for next time. This is why commitment is so important: only with commitment we can really feel disappointment. We must use the right psychology to feed our intuition properly to come up with better estimates. Not for estimation's sake, but because it improves our predictability.

- Define new Tasks to finish what you didn't do. Put these Tasks on the Candidate Task List. They will surface in due time. If they do not surface immediately for the next cycle because there are more important things to do, we apparently stopped at the right time
- Declare the Task finished after having taken the consequences: remember that you cannot work on this Task anymore, as it is impossible to do anything in the past

10. Now continue with planning the Tasks for the next cycle

Don't forget to apply what you learnt while planning the next TaskCycle!

9.2 Weekly 3-step process

Many projects have weekly team meetings. We found several shortcomings in these meetings, and based on the experience gained we eventually arrived at a weekly 3-step process, which proves instrumental for the success of Evo planning. In this process we minimize and optimize the time spent on organizing the Evo planning. Still, communication between all project people is greatly enhanced.

The steps are:

1. Individual preparation

In this step the individual team members do what they can do alone:

- Conclude current tasks
- Determine the most important Tasks for the next week
- Estimate the time needed for these Tasks
- Determine how much time is available for the project the coming week

The Project Manager and/or Systems Engineer also prepare for their team what they think are the most important Tasks, what time they think these Tasks may take (based on their own perception of the contents of each Task, and the capabilities of the Individual), and how much cooperation they expect to need with any person in the Team

2. 1-on-1's: Modulation with, and coaching by Project Management/Coach/Systems Engineer/Peer

In this step individual team members meet individually (1-on-1) with Project Management (Project Manager/Systems Engineer/Architect/Coach/Peer)⁹. In this meeting we *modulate* on the results of the individual preparations:

- We observe the status, and coach where people did not yet succeed in their intentions
- We compare what the Individual and Project Management thought to be the most important Tasks. In case of differences, we discuss until we agree
- We check the feasibility of getting all these Tasks done, based on the estimates
- We iterate until we are satisfied with the set of Tasks for the next cycle, checking for real commitment. Now we have the work package for the coming cycle

We use e.g. a notepad, spreadsheet, or a Web-based tool (like Figure 12), to capture the task descriptions and estimates. In physical meetings, even in the 1-on-1s, we use a projector, to ensure that we all are looking at, and talking about the same things. If people scribble their own notes, they all scribble something different. The others don't see what you scribble, and cannot correct you, if they wonder about what you wrote. Furthermore, when people are scribbling, they miss what is being said.

The screenshot shows a web-based task management interface. At the top, there's a header with 'Tasks' and a timestamp 'Last update: 02 Mar 2024 16:09:27 (CET)'. Below the header, there are several input fields for 'DeliveryCycle: Del 1', 'TaskCycle: Cyc 2', and 'memo for id 9'. There are also buttons for 'add', 'delete', 'reload', 'filter:Off', 'priv', 'next', 'hours of: niels', 'total: 26', 'done: 0', 'to do: 26', and 'Edit other: names, deliveries, taskcycles, close, help'. At the bottom, there is a table with the following data:

# id	# delCyc	delDue	# taskCyc	cycDue	# name	# hrs	ok	description
9	Del 1	2024-03-16	Cyc 2	2024-03-16	niels	3		task description 9
8	Del 1	2024-03-16	Cyc 2	2024-03-16	niels	4		task description 8
7	Del 1	2024-03-16	Cyc 2	2024-03-16	niels	3		task description 7
6	Del 1	2024-03-16	Cyc 2	2024-03-16	niels	5		task description 6
5	Del 1	2024-03-16	Cyc 2	2024-03-16	niels	8		task description 5
4	Del 1	2024-03-16	Cyc 2	2024-03-16	niels	3		task description 4
3	Del 1	2024-03-16	Cyc 1	2024-03-09	niels	6	ok	task description 3
2	Del 1	2024-03-16	Cyc 1	2024-03-09	niels	3	ok	task description 2
1	Del 1	2024-03-16	Cyc 1	2024-03-09	niels	1	ok	task description 1

Figure 12: Web-based Evo Task Admin tool

What's shown in the red box is the essence. The other parts are 'ornamental', and only to be used if it helps you to do a better job

⁹ Agilists usually don't like the terms 'Project' and 'Project Manager'. They can replace the word 'project' with 'work', and 'project manager' with 'team colleague' 😊.

Important: There is no scribe. People write down *their own* Task descriptions, and *their own* estimates. Then we'll see how they tune what they write, really thinking of what it means. This enhances commitment a lot. The other(s) watch, and discuss, if what is typed is not the same as what's in their mind.

Note: after having learnt from the estimates vs real time, these are consumed for learning, and lose their value.

3. Team meeting: Synchronization with the team

In this step, after all the 1-on-1s are concluded, we meet with the whole team. In this meeting we do those things we really need all the people for:

- While the Tasks are listed on the projected or shared screen, people read aloud their planned Tasks for the week. This leads to what we call the *synergy effect*: People say: "If you are going to do that, we must discuss ...", or "You can't do that, because ..." when we apparently overlooked something. Now we can discuss what to do about it, and change plans accordingly. The gain is that we don't together *generate* the plans, we only have to *modulate*. This saves time. We see a huge increase in very efficient communication *before* we do things, helping each other not to do the wrong things, not to do things incorrectly, or at the wrong time.
- If something came up at a 1-on-1, which is important for the group to know, it can be discussed now. In conventional team meetings we regularly see that we discuss a lot over the first subject that pops up, leaving no time for the more important subject that may come up later. In the Evo team meetings, we *select* which subject is most important to discuss together.
- To learn, and to socialize.

Developers, Project Managers and Systems Engineers/Architects invariably say that these 1-on-1s are one of the most powerful elements of the Evo Planning approach. And in the team meeting getting a much clearer insight into the real status of the work of all individuals than they ever had before.

After a few weeks, the whole planning process takes about 1 hour per person: 20 min preparation, 20 min 1-on-1, 20 min team meeting. Do we discuss less than before? No, we just discuss the right things effectively and efficiently. Of course, the Project Manager and the Systems Engineer need more time to see their whole team, but what they are doing here is what they should have been doing all along.

Recently I asked a Systems Engineer: "With all this planning and scheduling, do you still have enough time for your other SE work, that is overseeing the systems development itself?" Simple answer: "Yes. No problem."

10 Conclusion and invitation

I tried to paint some reasons and background, the principles, and just one element of the Evolutionary Planning approach. This approach enhances communication, prevents issues before they happen, and improves the effectiveness and efficiency of our work, leading to improved productivity as well as predictability.

This may leave you with a dilemma. While now you know that your project quite easily can become much more productive and predictive, you don't anymore have an excuse not to do it. Perhaps I should have warned you at the start of this text rather than at the end ☺.

The proof of the pudding is in the eating.

I invite people to do a five-week exercise with me, to get a real taste, and to see how this works for you to become much more effective, efficient, and predictive. Because I've done this with many people before, I can predict that the first few weeks you will have a hard time: seeing your current way of working in the mirror, in ways you still cannot imagine now. People actually call it quite 'painful' initially, but in several cases, they got promoted soon after. After five weeks you will have a much better understanding of a lot of details that really make this work. It also will give you a good feeling how this can work for your project and your organization.

This invitation may cause an avalanche of exercise requests, so I reserve the right to reject and select requests for the exercise, or to put you on a waiting-list. In order to improve your chances, you'd better explain why it would be interesting for you, and for me to help you. www.malotiaux.eu/contact


Start with the steps of chapter 9, and see what happens.

- **Plan-Do-Check-Act**
 - The powerful ingredient for success
- **Business Case** Why
 - Why we are going to improve what
- **Requirements Engineering**
 - What we are going to improve and what not
 - How much we will improve: quantification
- **Architecture and Design**
 - Selecting the optimum compromise for the conflicting requirements How
- **Early Review & Inspection**
 - Measuring quality while doing: learning to prevent doing the wrong things

Evolutionary Project Management elements (Evo)-

Tom Gilb

www.malotaux.eu/?id=processes



- **Weekly TaskCycle**
 - Short term planning
 - Optimizing estimation
 - Promising what we can achieve
 - Living up to our promises
- **Bi-weekly DeliveryCycle**
 - Optimizing the requirements and checking the assumptions
 - Soliciting feedback by delivering real Results to eagerly waiting Stakeholders
- **TimeLine**
 - Getting and keeping control of Time: Predicting the future
 - Feeding program/portfolio/resource management

Evo Project Planning - Niels

Zero Defects Attitude

What How much Are we done

Check and learn as early as possible

Efficiency of what we do

Effectiveness of what we do

What will happen, and what will we do about it?

Malotaux - Quality on Time

148

Niels Malotaux

Predictable Projects

How to deliver the Right Results at the Right Time

This text describes a very pragmatic approach that can quickly make your projects more successful in significantly shorter time, without bad stress, at the same time being able to predict what will be done when, with quite remarkable accuracy.

Does this sound impossible? That's what many people thought before they did it. Does it sound incredible? After all, you have been working in projects for many years. You have all the diplomas and experience. What can it be that you missed, and that will suddenly make your projects delivering better results much faster, while everybody knows that projects easily take longer than hoped for? I almost always hear the reserves like: "It can't be done", "We've tried it before", "I've heard that before", "It doesn't work here", "Our projects are different", "We build laaaaaarge systems", or "I'm doing these things already". These arguments are quite understandable, because people simply cannot imagine that there are techniques to deliver better results in substantially less time. Otherwise, they would be applying these techniques already, wouldn't they?

Niels Malotaux is an independent Project Coach specializing in optimizing project performance. Since 1974 he designed electronic hardware and software systems, at Delft University, in the Dutch Army, at Philips Electronics, and 20 years leading his own systems design company. Since 1998 he devotes his expertise to helping projects to deliver Quality On Time: delivering what the customer needs, when he needs it, to enable customer success. To this effect, Niels developed an approach for effectively teaching Evolutionary Project Management (Evo) Methods, Requirements Engineering, and Review and Inspection techniques. Since 2001 he taught and coached over 400 projects in 40+ organizations in the Netherlands, Belgium, China, Germany, India, Ireland, Israel, Japan, Romania, South Africa, Serbia, the UK, and the US, which led to a wealth of experience in which approaches work better, and which work less in the practice of real projects. He is a frequent speaker at conferences, see www.malotaux.eu/conferences

Find more booklets at: www.malotaux.eu/booklets

1. Evolutionary Project Management Methods
2. How Quality is Assured by Evolutionary Methods
3. Optimizing the Contribution of Testing to Project Success
- 3a. Optimizing Quality Assurance for Better Results (same as 3, but now for non-software projects)
4. Controlling Project Risk by Design
5. TimeLine: Getting and Keeping Control over your Project
6. Recognizing and Understanding Human Behaviour
7. Evolutionary Planning (similar to booklet#5 TimeLine, but other order, and added predictability)
8. Help! We have a QA problem!
9. Predictable Projects - How to deliver the Right Results at the Right Time (this booklet)

N R Malotaux Consultancy

Niels R. Malotaux
phone +31-655 753 604
mail niels@malotaux.eu
web www.malotaux.eu