

TimeLine: Getting and Keeping Projects under Control

Niels Malotaux¹

(Project Coach, N R Malotaux - Consultancy, Bilthoven, The Netherlands)

Abstract

Many projects deliver late and over budget. The only way to do something about this is changing our way of working. If we don't change, the result will not magically improve. The Evolutionary Project Management (Evo) approach continuously introduces small changes (hence *evolutionary*), constantly improving the performance of what we are doing. One of the Evo techniques is TimeLine, for getting and keeping the timing of projects under control while improving the results, using just-enough estimation and calibration. If we then discover that the project will be late, we actively deal with it. Many solutions for getting a project on time, however, don't work. A usually overlooked solution is actively and constantly *saving time*. By foreseeing what later will prove to be unnecessary, we can prevent spending time on it. This paper describes the TimeLine technique, which has helped many projects to deliver the right results at the right time.

Keywords

Project Management, Planning, Control, "Saving time", Calibration

1. How do we get projects on time?

Insanity is doing the same things over and over again and hoping the outcome to be different (let alone better)

(Albert Einstein 1879-1955, Benjamin Franklin 1706-1790, it seems Franklin was first)

Many projects deliver late. If we don't change our ways, projects will continue to be late. The Evolutionary Project Management (Evo) approach (Gilb (1988, 2005), Malotaux (2001, 2004, 2006)) is about continuously introducing small changes (hence *evolutionary*) in the way we do things, constantly improving the performance and the results of what we are doing. Because people can imagine the effect of the change, this evolutionary change can be biased towards improvement, rather than being random.

One of the techniques having emerged out of the Evo way of working is the TimeLine technique, which allows us to get and keep the timing of projects under control while still improving the project results, using just-enough estimation and then calibration to reality. TimeLine doesn't stop at establishing that a project will be late. We actively deal with that knowledge: instead of *accepting* the apparent outcome of a TimeLine exercise, we have ample opportunities of *doing* something about it. One of the most rewarding ways of doing something about it is *saving time*. An essential prerequisite of getting projects on time is, however, that we *care* about time.

2. Many projects deliver late

If we ask people of a project why they were late, they have a lot of excuses, usually others being the cause of delays. If we ask them what we could have done about it, they easily can make suggestions. We usually know why we are late and we know some ways to do something about it. The problem is that we don't do something about it. One of the problems is that customers fatalistically think that this is the way it is and *keep paying*. If the customers would insist on the delivery date or else wouldn't pay, the problem would have been solved a long time ago.

Some project managers say: "This all has to be done, and it simply takes as much time as it takes; we cannot stop before all is done". What "all" is, should be defined by the requirements, which have to be in tune with the business case. These project managers for some strange reason forget that delivery time usually is as much a requirement as "all" other requirements. In many cases delivery time is even one of the most important requirements.

3. What can we do about it?

What can we do if what we think we have to do doesn't fit the available time, or if we want to do things faster?

3.1 Deceptive options

Let's first do away with the deceptive options. Deceptive options are options we often see applied, but which don't work. It's surprising that people don't learn and keep using these options.

¹ Bongerdlaan 53, 3723 VB Bilthoven, The Netherlands – niels@malotaux.nl

Hoping for the best (fatalistic)

Most projects take more time than expected. 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 your project will be on time this time won't help. We call this ostriching: putting your head into the sand waiting until Murphy² strikes again.

Going for it (macho)

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

Working Overtime (fooling yourself)

Working overtime is fooling yourself: 40 hours of work per week is already quite hard. If you put in more hours, you'll get more tired, make 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. It won't work. This is also ostriching. 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.

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. We may call this Parkinson's Law³ or the Student Syndrome⁴. At the new deadline we probably hardly have done more, getting the project result even later. Not a good idea, unless we really are in the nine mother's area (see next), where nobody, even with all the optimization techniques available, could do it. Even then, just because of the Student Syndrome, it's better to optimize what we can do in the available time before the deadline. The earlier the deadline, the longer our future afterwards, in which we can decide what the next best thing there is to do. We better optimize the time spent right from the beginning, because we'll probably need that time anyway at the end. Optimizing only at the end won't bring back the time we lost at the beginning.

3.2 A riskful option: adding people ...

A typical move is to add people to a 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 or in 3 months with 4 people, as shown in Figure 1. In his essay *The Mythical Man-Month*, Brooks (1975) shows that this is a fallacy, defining Brooks' Law: *Adding people to a late project makes it later*.

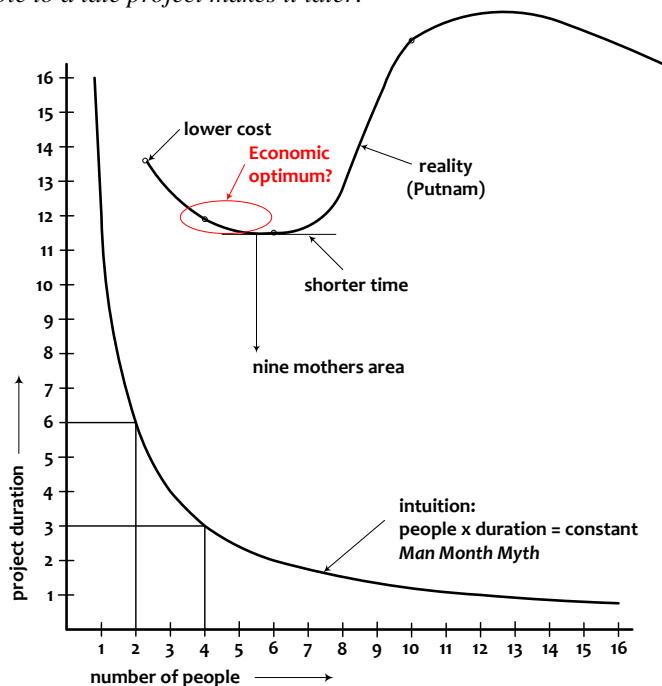


Figure 1: The Myth of the Man-Month: reality is completely different

² *Whatever can go wrong, will go wrong* is the popular version of Murphy's Law. The real version is: *What can go wrong, will go wrong, so we have to predict all possible ways it can go wrong, and make sure that these cannot happen.* Spark (2006).

³ Parkinson's Law: "Work expands so as to fill the time available for its completion". Parkinson (1955) observed: "Granted that work (and especially paperwork) 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 Starting as late as possible, only when the pressure of the FatalDate is really felt. Term attributed to E. Goldratt (1997).

Putnam (1996) confirms this with measurements on some 500 (software) projects. 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. 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. 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. After all, many projects seem to find out that they are late only by the end of the project. 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 architect usually can oversee only a limited number of people before becoming a bottleneck. Therefore, adding people is not automatically a solution that works. It can be very risky.

How can those mega-projects, where 100's of people work together, be successful? Well, in many cases they are not. They deliver less and later than the customer expects and many projects simply fail. 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 only from time to time.

3.3 The best option that always works: saving time

Instead of letting things randomly be undone at the deadline (we call the deadline the *FatalDate*), it's better to *choose* what we won't have done, preferably those things that weren't needed anyway. We know that we won't have enough time, so let's save time wherever possible from the very start of the project!

There are several ways to save time, *without negatively affecting the Result of the project (on the contrary!)*:

Efficiency in *what (why, for whom) we do*: *doing only what is needed, not doing things that later prove to be not needed, preventing mistakes and preventing working on superfluous things.* Because people tend to do more than necessary, especially if the goals are not clear, there is ample opportunity for *not* doing what is *not* needed. We use the *Business Case* and continuous *Requirements Management* to control this process.

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. Because performance and project time are usually in competition, the solution should be an optimum compromise and not the first solution that comes to mind. We use *Architecture* and *Design* to control this process. We use *DeliveryCycles* to check the requirements and assumptions.

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 the project*. We use *Evolutionary Planning* to control this process.

Continuous improvement and prevention processes

Actively and constantly learning how to do things better and how to overcome bad tendencies. We use rapid and frequent *Plan-Do-Check-Act* (PDCA- or Deming⁵) 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 it's the only way ever to approach Zero Defects.

Efficiency in *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 saves time. We use *Evolutionary Planning* to control this process, with *TaskCycles* and *DeliveryCycles* to make sure we do the right things in the right order, and *TimeLine* to get and keep the whole project under control. Elements of these are *Just Enough Estimation*, *Dynamic Prioritizing* and *Calibration* techniques. For some more details of the techniques mentioned: see Malotau (2004, 2006). All of these elements are huge time savers. Of course we don't have to wait for the project getting into trouble. We can also apply these time savers if what we think we have to do easily fits in the available time, to produce results even faster.

TimeBoxing provides incentives to constantly apply these ways to save time, in order to stay within the TimeBox. For TimeBoxing to work properly, it is important to change from optimistic or pessimistic, to realistic estimation. If

⁵ Deming (1986), Malotau (2006, chapter 6)

the TimeBox is too short, we cause stress with adverse effects. If the TimeBox is too long it doesn't work. In the experience of the author, people in projects can easily change into realistic estimators in a few weeks time, *if* we are serious about time. TimeBoxing is much more efficient than FeatureBoxing (= waiting until we're ready), because with FeatureBoxing we lack a deadline, causing Parkinson's Law and the Student Syndrome to kick in badly. Note that this concept of saving time is similar to "eliminating waste" in Lean thinking, and already indicated by Henry Ford in his book "My Life and Work", back in 1922.

The **Evolutionary Project Management** (Evo) approach uses and constantly evolutionarily optimizes the elements of saving time as shown above: Plan-Do-Check-Act cycles, Zero-Defects attitude, Business Case techniques, Requirements Management techniques, Design techniques, Early Reviews, TaskCycles, DeliveryCycles and TimeLine. Projects starting to use the Evo approach almost immediately become 30% more productive. Background of the Evo approach can be found in Gilb (1988, 2005) and Malotau (2001, 2004, 2006).

4. TimeLine

We will now explain the TimeLine technique in some more detail.

4.1 Determining what can be done

In many projects all the work we think⁶ we have to do is cut into pieces, the pieces are estimated, and the estimations 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 adding some contingency, is presented as the duration of the project (Figure 2). A problem is that in many cases the required delivery date is earlier. Discussions about this tension between estimated and expected delivery consume time, while the required delivery date doesn't change, leaving even less time for the project.

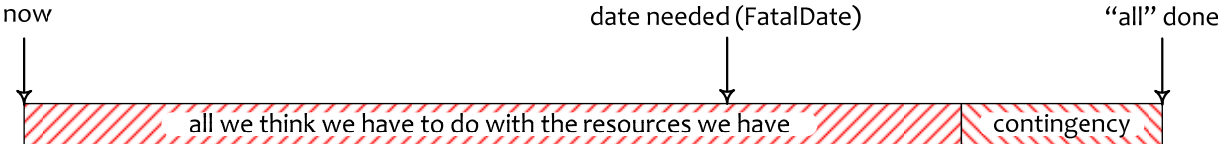


Figure 2: Standard approach: it takes what it takes, but that may pose a problem

Because the delivery date is usually a requirement just as all the other requirements, it has to be treated as seriously as all the other requirements. With TimeLine, we treat agreed delivery dates seriously and we meet these dates, or, we very quickly explain why the delivery date cannot be met. We don't wait till the FatalDate to tell that we didn't make it, because if it's really impossible, we know it much earlier. If it's possible, we deliver.

TimeLine can be used on any scale: on a program, a project, a sub-project, on deliveries, and even on tasks. The technique is always the same. We still estimate what we think we have to do. Then we discuss the TimeLine with our customer and explain (Figure 3):

- 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)

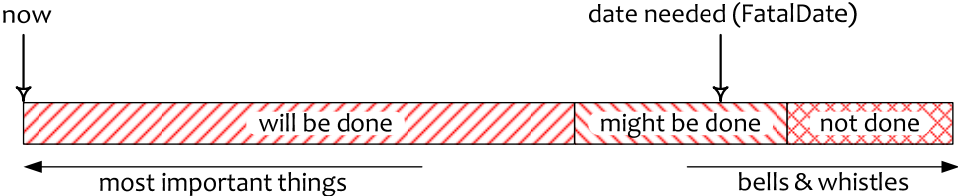


Figure 3: Basic TimeLine: what will surely be done, what will not, and what may be done

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 we'll only have done the most important things. Customers don't mind about the bells and whistles if Time to Market is important. Because priorities may change very dynamically, we have to constantly reconsider the order of what we do when.

⁶ 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. And they *will* change!

4.2 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.

4.3 DeliveryCycles

Within these 10 weeks, we plan DeliveryCycles (Figure 4) 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 many of our assumptions may be incorrect. Therefore we need communication and feedback. We deliver to *eagerly waiting* Stakeholders, otherwise we don’t get feedback. If the appropriate Stakeholders aren’t eagerly waiting, either they’re not interested and we may better work for other Stakeholders, or they have to be made eagerly waiting by delivering what we call *Juicy Bits*.

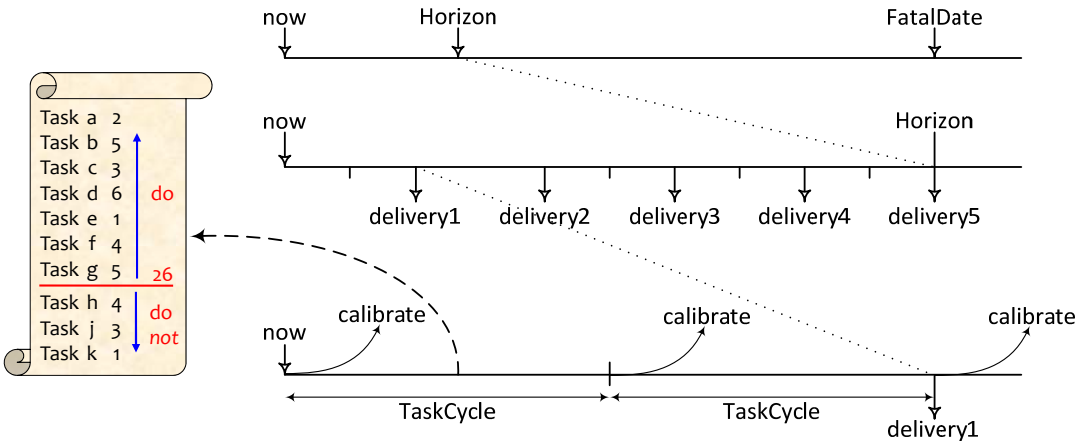


Figure 4: TimeLine summary: setting a FatalDate, a Horizon, Deliveries, TaskCycles, and then calibrating back

4.4 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 effort-hours (see Malotau (2004), section 6.1, for a more detailed explanation). 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), reserving and confining all unplannable project activities like email, phone-calls, planning, small meetings, etc. to the remainder of the time. We put this work in optimum order, divide it over the people in the project, have these people estimate the time they would need to do the work, see that they don’t get overloaded and that they synchronize their work to optimize the duration.

4.5 Calibration

Having estimated the work that has to be done in the first week, we have captured the first metrics for *calibrating* 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 don’t do something about it. At the start this seems weak evidence, but it’s already an indication that our estimations may be too optimistic. Putting our head in the sand for this evidence is dangerous (I’ve heard all the excuses that there were “one-time causes”; next time there were more “one-time causes”). One week later, when we have the *actual* results of the first week, we have even better numbers to extrapolate and scale how long our project really may take. Week after week we will gather more information 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.

Let’s take an example of taking the consequence of the TimeLine (Figure 5): At the start, we estimate that the work we think we have to do in the coming 10 weeks is about 50 person Weeks of Work (‘WoW’, line a). We start with 5 people. After 4 weeks, we find that “only” 10 WoW have been completed (line b), instead of the expected 20 WoW. If we don’t change our ways of working, the project will take twice as long (line d) or produce only half (line c). If the deadline is really hard, a typical reaction of management is to throw more people at the project (line e). However, based on our progressing understanding of the work, we found that we forgot to plan some work that also “has” to be done: we now think we still have to do 50 WoW in the remaining 6 weeks (line f). Management decides to add

even more people to the project, because they don't want the project to take longer. This solution probably won't produce the desired outcome, and even may work out counterproductive, as explained in section 3.2. We can counter this dilemma by *actively saving time*, doing only what is really necessary (line g), or a combination of not doing what is not necessary (line g2) and doing things more productively (line h), as explained in section 3.3. Actively *designing* what exactly to do and in which order, saves a lot of time.

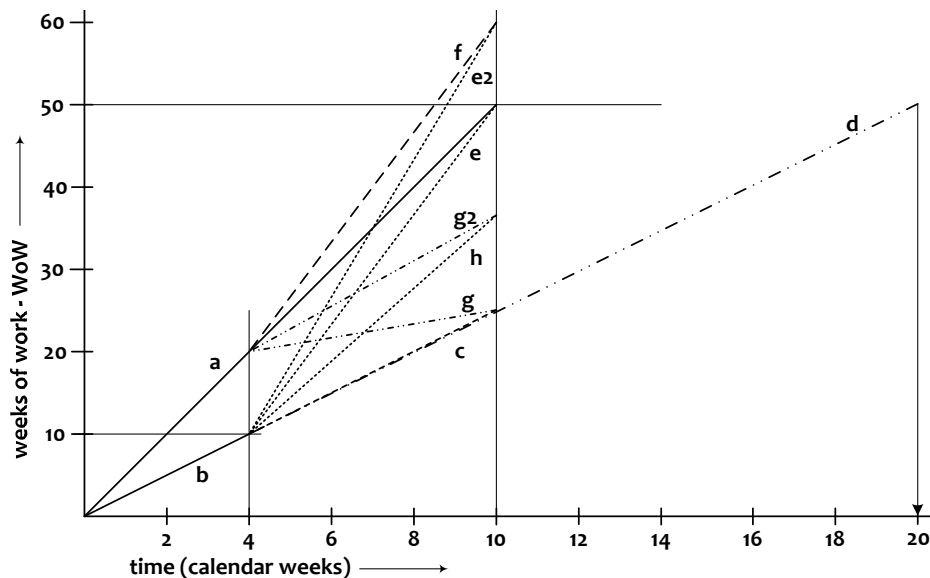


Figure 5: Earned Value (up to week 4) and Value Still to Earn (from week 5)

5. Conclusion

TimeLine doesn't solve our problems. It is a set of techniques to early expose the real status of our project. Instead of *accepting* an undesired outcome of a TimeLine exercise, we have ample opportunity of *doing* something about it. People do a lot of unnecessary things in projects, so it's important to identify those things before spending time on them. If we later find out that we did unnecessary things, the time is already spent, and never can be regained. By revisiting the TimeLine every week, we stay on top of how the project is developing and we can easily report to management the real status of the project and show the consequences of management decisions affecting the project.

Doesn't all this TimeLining take a lot of time? The first few times it does, because we have to learn how to use the technique. After a few times, however, we dash it off and we can start optimizing the results of the project, producing more than ever before. TimeLine allows us to take our head out of the sand, stay in control of the project and deliver Results successfully, on time. Still, many Project Managers hesitate to start using the TimeLine technique. However, after having done it once, the usual reaction is: "Wow! I got much better oversight over the project than I ever expected", and the hesitation is over. An other reaction: "We never did this before ..."

The TimeLine technique is not mere theory. It's highly pragmatic, and successfully used in many projects coached by the author. The most commonly encountered bottleneck is that no one in the project has an oversight of what exactly the project is supposed to accomplish. May be this is why Project Managers hesitate to start using the technique. If you don't know well what to do, planning isn't easy. Redefining what the project is to accomplish and henceforth focusing on this goal is the first immediate timesaver of the technique, with many savings to follow.

References

- Brooks, 1975, *The Mythical Man-Month*, Addison Wesley, ISBN 0201006502, Reprint 1995, ISBN 0201835959.
- Deming, 1986, *Out of the Crisis*, MIT, ISBN 0911379010.
- Gilb, 1988, *Principles of Software Engineering Management*, Addison Wesley, ISBN 0201192462.
- Gilb, 2005, *Competitive Engineering*, Elsevier, ISBN 0750665076.
- Goldratt, 1997, *Critical Chain*, Gower, ISBN 0566080389
- Malotaux, 2001, *Evolutionary Project Management Methods*, www.malotaux.nl/nrm/pdf/MxEvo.pdf
- Malotaux, 2004, *How Quality is Assured by Evolutionary Methods*, www.malotaux.nl/nrm/pdf/Booklet2.pdf
- Malotaux, 2006, *Controlling Project Risk by Design*, www.malotaux.nl/nrm/pdf/EvoRisk.pdf
- Parkinson, 1955, *Parkinson's Law*: www.adstockweb.com/business-lore/Parkinson's_Law.htm
- Putnam (1996?), *Team Size Can Be the Key to a Successful Project*, www.qsm.com/process_01.html
- Spark, 2006, *A History of Murphy's Law*, Periscope Film, ISBN 0978638891.