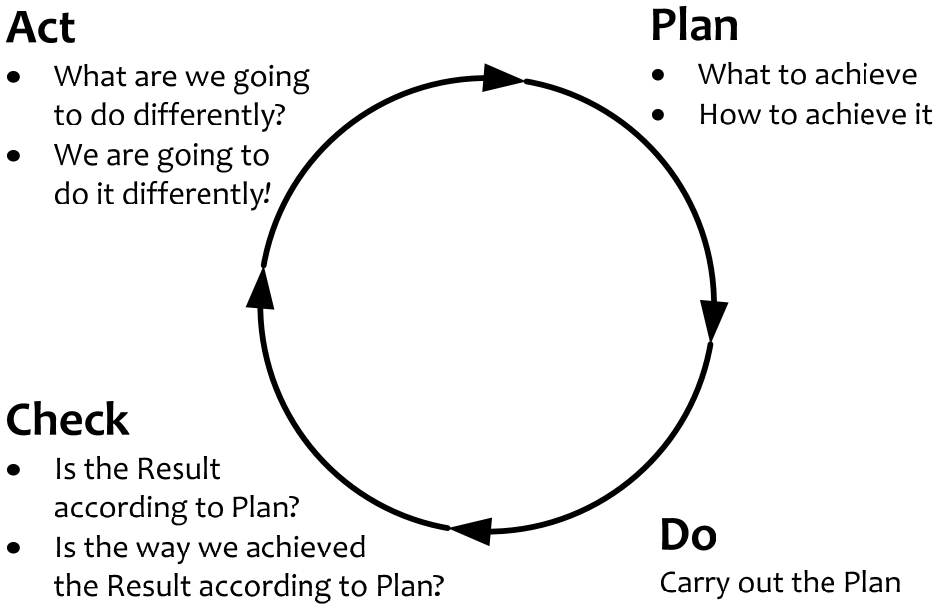
Niels Malotaux

How Quality is Assured by Evolutionary Methods



How Quality is Assured by Evolutionary Methods

1 Introduction

After several years of experience as a Project Coach introducing Evolutionary Project Management Methods (Evo) in development projects, I think I can claim that Quality can be Assured if projects apply these methods. Does this mean that the Quality Assurance function is not needed any more? No. QA is still needed, because one of the main factors jeopardizing the Assured Quality is lack of discipline - discipline to keep applying the methods in order to meet our commitments.
In many cases, people know the best way to do their work. However, if nobody is watching, people tend to take shortcuts. If somebody is watching over their shoulder, people tend to take fewer shortcuts. The Project Manager can watch over the shoulders of the team. The team can watch over each other's shoulders. But who's watching over the Project Managers' shoulder? This task is the responsibility of management, but the Quality Assurance function can help.

Even with an assurance function in place, team members still have to know what is the best way to do their work in the first place. Since there is no absolute "best way", while the "best way" is even dynamically changing, we must also provide the people with an ability to actively find out the best way while working in the project.
Evo is actually rapidly and frequently applying the Plan-Do-Check-Act (or Deming) cycle, not just for the development of the *product*, but at the same time for the organization of the *project* and even for assessing and improving the *methods* used on the project. We need to continuously ask ourselves: "What should we do *now*, in which *order*, to which *level of detail for now*".

Working the Evo way means organizing the work in weekly (or even shorter) Task-cycles. In these Task-cycles we optimize estimation, planning, and tracking. Task-cycles feed bi-weekly (or shorter) Delivery-cycles by which we optimize the requirements and our assumptions. We use a practice known as TimeLine to create and maintain the total project scope and to connect the Project Result, through the Deliveries, with the actual work organized in Tasks. Evo combines Estimation, Planning, Tracking, Requirements Engineering, Requirements Management, and Risk Management into *Result Management*. *Result* is defined as the combined value we provide to all the Stakeholders of our product, ultimately leading to customer success. Evo has a fanatical view on ROI: Whatever we do should contribute to the Result and we try to avoid whatever does not contribute.

In this paper I will explain the basics of this Evolutionary approach and practical details people can start applying immediately.

2 The Goal

Let's assume that the purpose of development projects is to deliver *what* the customer needs, at the *time* he needs it, to create substantially *greater value* than the *cost of development* and to enable *customer success*. In short, we call this Quality On Time: the right things at the right time.

It is important to note that the *functionality* we are working on in most development projects *already exists*. Usually, all we are supposed to do is enhance the performance of specified functionality to create more value for the customer. The set of functions we are enhancing defines the scope of the project. The scope should be chosen such that it provides more value for cost than another scope.

> *Banks have banked for thousands of years. First using clay tablets, then using card-trays and now using computers. Banks are, however, still doing what they did before. The function is still the same, while the performance (ease, speed, complexity of transactions) is enhanced. If a new system does not deliver sufficiently more value than the old system, there will be no funds to pay for the new system and the developers.*

It would be nice if we could in one project develop the ultimate solution, creating the ultimate value. Apart from the risk that, when done, we could be out of work, this is not possible because of limited resources such as:

- The available time (time to market may strongly influence time to profit)
- The available money
- The available people and the capabilities of these people (it would be nice if we could hire the best people. Normally, however, the challenge is to succeed with *average* people)
- The available experience on the subject
- The available technology
- The capability of the users to adopt the new system

In development projects we can only strive to optimize the compromise between value creation and the available limited resources. If the results we can achieve, given these limited resources, are insufficient to provide significant value for customer success, we shouldn't even start the project. Given these limited resources we are not even satisfied with *good* results, we actively want to *maximize* the Result created. Looking back at the end of a project, not only should our customer have a big smile of satisfaction, we should ourselves also be confident that we couldn't have done better.

This implies that we should feel a Sense of Urgency to constantly optimize the results we are working on, to constantly optimize our success. Without this Sense of Urgency, Evo doesn't work.

3 Plan-Do-Check-Act

Since childhood we learn intuitively through experience. Besides learning from our own experience, we also learn from accepting the experience of others: at school, in workshops and at conferences. This learning process is rather slow. We can, however, stimulate the learning process by actively using the Plan-Do-Check-Act cycle, as presented by Deming:

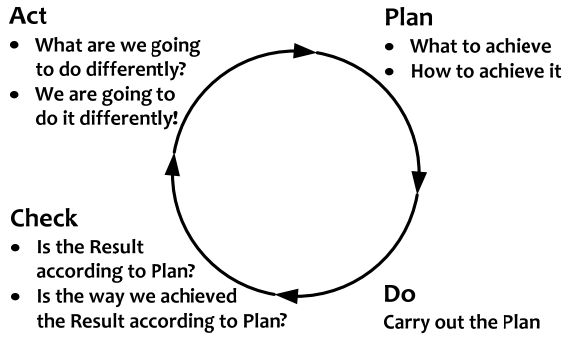


Figure 1: PDCA or Deming cycle

- **Plan**
  What are we supposed to accomplish and how are we going to accomplish it?
- **Do**
  Carry out the Plan
- **Check**
  Is the result and the way we achieved the result according to the Plan?
- **Act**
  - If the result was not according to the Plan, what are we going to differently the next time to achieve a better result?
  - If the result was according to the Plan, was it accidental? How do we make sure next time the result is equally according to Plan?

Do is never 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 think we want to go to the next Do. Well, that's what I believed until recently. Taking a closer look at what really is happening we can see that Check *is* often done: people seem to be quite aware what is going wrong and often even know what should be done about it. The real problem is that we don't Act: taking what we know and doing something about it.

Sometimes I hear people in a project week after week complaining about the same problem, usually that somebody else is doing something wrong. My advice: either deal with it or stop complaining. Don't keep wasting energy complaining about the same problems over and over. Do something: Act! Find a solution, plan the time needed and solve the problem.

4 Evo

4.1 Evo

Evo is short for Evolutionary Development, Evolutionary Delivery, Evolutionary Project-Management, deliberately going through the Plan-Do-Check-Act learning cycle rapidly and frequently, for product, project and process, continuously thinking "*what* to do, in which *order*, to which level of *detail* for now". It's a label for a set of methods that allow us to effectively and efficiently run projects, delivering Quality On Time. Evo integrates Planning, Requirements and Risk Management into *Result Management*. It's *actively induced* evolution because we don't wait for evolution to happen, we make it happen.

Many organizations mandate a Project Evaluation at the end of every project. Even so, few projects do the actual evaluation because they feel that these evaluations do not contribute to better results. Why is this? Consider one-year projects (see figure 2). People have to evaluate what went wrong and what went accidentally right (and why) as long as a year ago. In addition, they may not be able to use the learning from an event until as long as a year after the fact. The idea of evaluation is valuable. The time constants of this process as described above are, however, beyond the capabilities of the human

mind. In Evo, we do evaluations (PDCA) every week. This tunes the time dimension to the human mind's abilities and enables us to rapidly implement what we learn.

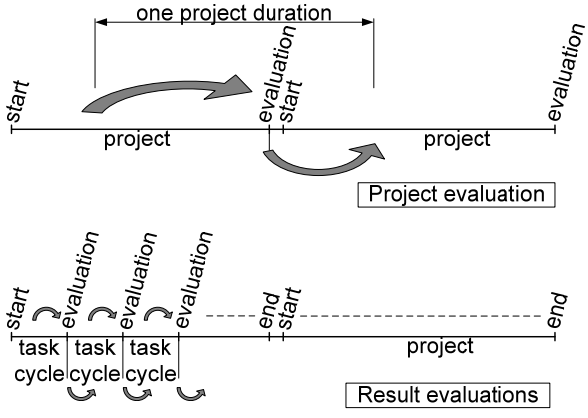


Figure 2: Project and Result evaluations

4.2 Evo and the Product

We don't know the real requirements. *They* don't know the real requirements either. So, stop pretending we know, and accept that we have to find out what the real requirements are, together. This includes finding out who *they* are. We can make the nicest systems, given unlimited time and money. However, our customer doesn't have unlimited time and money. If the customer cannot afford all what is possible, we must find out the best Result we can achieve within the limited resources. If that's less than the customer needs for success, we shouldn't even start.

Result is the value gained by the use of what we developed. Result ultimately is customer success. If no value is gained, there is nothing to pay our salaries from. Because not all customers are aware of this, we have to work with the customer to find out what the optimum Result is to make sure that we are generating significant value. In Evo we work with a no-cure no-pay attitude. Whatever does *not* contribute to customer success, we don't do.

4.3 Evo and the Project

The optimum Result is the best product for the least cost. At the start of the project we don't know what the optimum Result is, so we must organize the project in such a way that we discover and implement the optimum Result at the lowest cost. This implies optimizing the effectiveness and efficiency of discovery and implementation. It also means that we have to change our estimation practice from optimistic to realistic, so that we can predict the future more accurately. We have to accept the realistic estimates and plan accordingly. We have to dynamically keep our plans up to date in order to keep control over the Result. We must learn better time management and better priority management. These are among many issues we can improve. In Evo we are constantly, dynamically improving on these issues because our success is at stake. We do not only design the product, we also *design* the *project.*

We take time and money budgets very seriously. This means that we don't ask for more when we were supposed to deliver. If the budgets really were insufficient we could have predicted this way before the budgets ran out and, together with the customer, we could have acted accordingly.

4.4 Evo and the Process

Because it is continually being improved as a process, Evo is made up of the best set of methods we know at a given time. If we find a better way, we change to the better way. Not only do we employ PDCA on the product and project activities, we also constantly and dynamically apply the PDCA cycle to the methods we use. If another method seems better, we try it. We may experiment. But we deliberately Check and Act: if the new method is better, we change to the better method. If the new method is not better, we revert to the last known best method. Methods, processes and procedures are there to help us. If they don't, we discard them. A side effect is that Evo processes may be different between projects and between organizations because of differences in culture or differences in experience. The common property is always the urge for success in defined goals.

4.5 Does Evo cost more time?

Some people fear that all these evaluations, intensive planning and constant improvements will cost a lot of extra time. It does not: experience based on many projects proves that it *saves* time. Why else would we do it? The "extra" things we do in Evo projects are the things that should be done anyway on any project to make it successful. So, we don't really do "extra" things. We only do those things that contribute to Quality On Time.

4.6 When do you *not* need Evo?

There are circumstances where you may consider not using Evo, such as:

- The requirements are completely clear and nothing will change. *This is production, not development.*

- The requirements can be easily met with the available resources in the available time. *Still, Evo can make you achieve better results in shorter time.*
- The customer can wait until you are ready. *Still, Evo can make you achieve better results in shorter time. Why waste your time while you can do more interesting things?*
- The customer doesn't care about the result. *Should we contemplate this project? Is he going to pay?*
- You don't care about the cost or time. *Could be a hobby or a vacation.*
- Your boss doesn't care about the cost or time. *He probably doesn't know what to do with his money.*
- Management doesn't know what to do with the time saved. *Be careful, they may frustrate your project.*
- There is no Sense of Urgency.

Sense of Urgency is an important issue to watch for. Most people, including management, will immediately affirm the urgency of the best Result at the lowest cost. That's trivial. However, there are cases where their actions tell a different story. The remedy is either to educate them by coaching, or not to bother them with Evo. There are plenty of places where you can be successful with Evo, so why bother if they don't want to be more successful. Besides, Evo is never a goal in itself. Result is all that counts.
If they get optimum results *their* way, you shouldn't complain, but rather learn from how they do it.

5 Evo basics

We organize Evo projects on several levels. We use the TaskCycle to organize the work, the DeliveryCycle to organize the Results and TimeLine for making sure we'll be on time.

5.1 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 the highest priority Tasks, never do lower priority Tasks and never do undefined Tasks. As a practical rule, we plan 2/3 of the available time and in the remaining 1/3 of the time we do all those things we also have to do in the project, like small interrupts, helping each other, project meetings and many other things. If we plan 100% of our available time, we will still do all those other things, and we will never succeed in what we planned.

TaskCycles take at most one week, in some cases even less. Every Cycle we decide what is most important to do, how much time it takes to do it completely (we define what completely 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.

5.2 Task Selection Criteria

The following set of Task Selection Criteria proved useful for deciding the priority of Tasks:

- Most important requirements
- Highest risks
- Most educational or supporting things
- Active Synchronization with others outside your project

Remember: Every Cycle delivers a useful, completed Result.

5.3 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.

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 practice. One of the important reasons for the short length of the cycle is that we want to check our (and their) assumptions before we have done a lot of work that later may prove unnecessary, losing valuable time. Short DeliveryCycles help us do this with minimum risk and cost.

A common misconception of Deliveries is that people think they 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.

5.4 Delivery Selection Criteria

The following set of Delivery Selection Criteria proved useful for deciding the contents of Deliveries:

1. What will generate optimum feedback
2. Delivering to *eagerly waiting* Stakeholders (otherwise, we won't get optimum feedback)
3. Delivering the *juiciest*, most important Stakeholder values that can be made at the least cost, to raise the Stakeholder's interest to provide optimum feedback
4. What will make Stakeholders more productive *now*

Also remember that:

- Every Delivery must have a useful set of values, otherwise the Stakeholders get stuck (for example, if there is a *Copy* function, there should also be a *Paste* function)

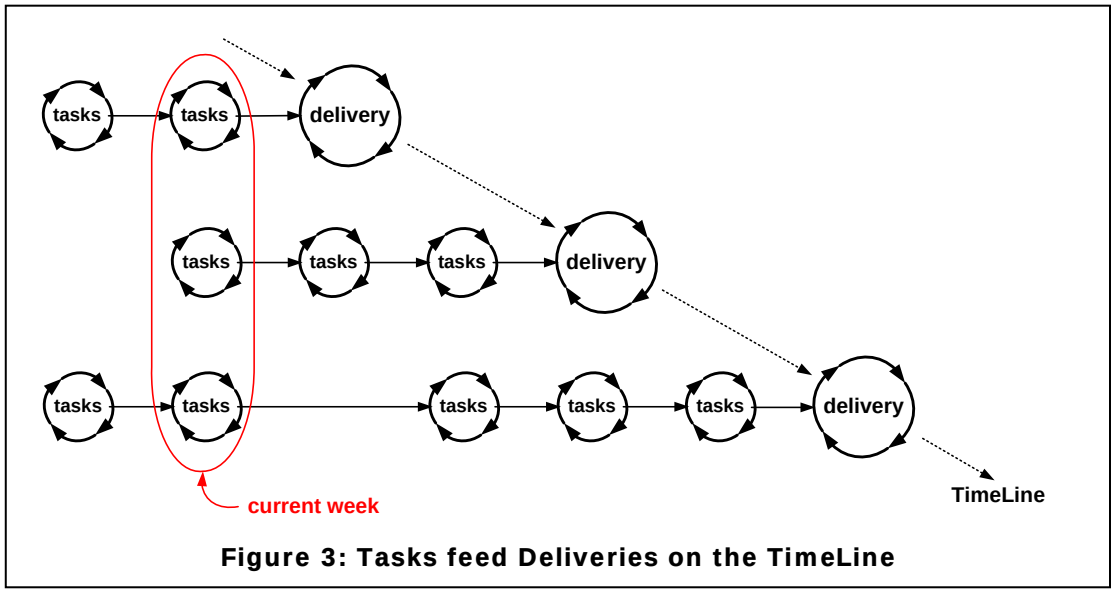


Figure 3: Tasks feed Deliveries on the TimeLine

- Every Delivery must offer clear incremental value, otherwise the Stakeholders stop producing feedback
- Every Delivery delivers the smallest clear increment, to get the most rapid and frequent feedback
- If the contents of a Delivery takes more than two weeks, it can be shortened: *try harder*

5.5 TimeLine

We use the TimeLine technique to make sure that we will be on time (or even early). A TimeLine is a line between now and then. *Then* is any deadline (we also call it FatalDate): End of Task, End of Delivery, End of sub-project or milestone, or End of Project. A FatalDate is a commitment to deliver successfully, no excuses. We took the responsibility, so the Result will simply be there. If it is not, we failed to deliver on time. At the FatalDate, any excuse is pointless, because you *could have known before*. The moment you can foresee that, for whatever reason, you are not going to meet the FatalDate, you could have told the appropriate Stakeholders and we could have adapted our plans accordingly. Any day later you realize that you cannot meet the FatalDate, you have a day less to cope with it. If the time is up, there is no time left. You cannot change history.
During a project we constantly monitor where we are now, what the FatalDate is, and constantly optimize what we should and what we can do in between.

5.6 Tasks, Deliveries and TimeLine

Tasks feed Deliveries (see figure 3). Deliveries create focus for what to do in Tasks. In any TaskCycle we are working on the current Delivery. Because some Deliveries need more than two weeks to prepare we may also work on Tasks for future Deliveries. That said, we shouldn't start working on future Deliveries too soon, because the longer we work on a Delivery, the more the world may have changed, so that what we already did has become irrelevant. It really is a challenge to define Deliveries and to start working on the right Delivery, Just-in-Time.
On the TimeLine we are scheduling Deliveries in the best order to achieve the best Result in the least time. This is a dynamic process, because we may have to redefine Deliveries based on experience of

the developers, feedback from Stakeholders, and market changes. We are constantly challenging the order of Deliveries to get the best route to the Result, with the fewest iterations. We may also have to change the order of Deliveries if somebody crucial for a Delivery is ill, or is needed temporarily on another project.

6 Evo practice

By collecting the experience of more than twenty-five projects between 2001 and 2004, we have arrived at several best practices that you can use to start new Evo projects.

These practices do not describe theoretical processes or how someone *thinks* we should work. They rather describe what works in day-to-day reality, where we have to cope with human psychological behavior that is not always as logical as we intuitively assume or might wish were true. In fact, Evo thrives on reality. Because of this, you can start using these practices tomorrow and immediately benefit. You don't have to call it Evo. Result is all that counts. That is never just "following process". Result is always measured as customer success at the least cost.

6.1 TaskCycle planning

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

1. **Determine the number of hours you have available for this project this TaskCycle**
   People may work less than the full week. For example, they may take a vacation, follow a course, visit a dentist or work for more than one project. So we determine the number of available hours for this project first, because then we know when we can stop adding Tasks.
2. **Divide this gross number of available hours into:**
   - **Available Plannable Hours (default 2/3 of gross available hours)**
   - **Available Unplannable Hours (default 1/3 of gross available hours)**

   We only plan those Tasks that don't get done unless planned. If you plan, you *have* time, and after that time, the Task will be done.
   We do *not* plan Tasks that will get done anyway, even without planning. As a default ratio we start with 2/3 plannable and 1/3 unplannable time. In many projects this proves to be realistic. In a 40 hour work week, this means 27 hours plannable time, 13 hours unplannable time.
3. **Define Tasks for this cycle, using the Task Selection Criteria**
   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 in the Tasks.
4. **Estimate the number of effort hours needed to completely accomplish each Task**
   We always estimate *effort* hours. Ask people to estimate in days, and they come up with lead time (the time between starting and finishing the Task). Ask people to estimate in hours, and you'll find that they usually come up with effort (the *net* time needed for completing 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 estimation, they may only 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 he is going to need; otherwise there will be no commitment to succeed.
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: at 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, the variations can cancel out. If you have a massive task of 27 hours, it is more difficult to estimate and the averaging trick cannot save you any more.
6. **Fill the available plannable hours with the most important Tasks**
   Never select less important Tasks. *Always* fill the available plannable hours completely.
7. **Ascertain that indeed these are the most important Tasks to do and that you are confident that the work can 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, and that these tasks will be done, completely done, at the end of the cycle.

At this point, you will have a list of Tasks that will get done. 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 of the Tasks for you. Beware, however, that it may cost 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.

6.2 Evo Task Administrator tool

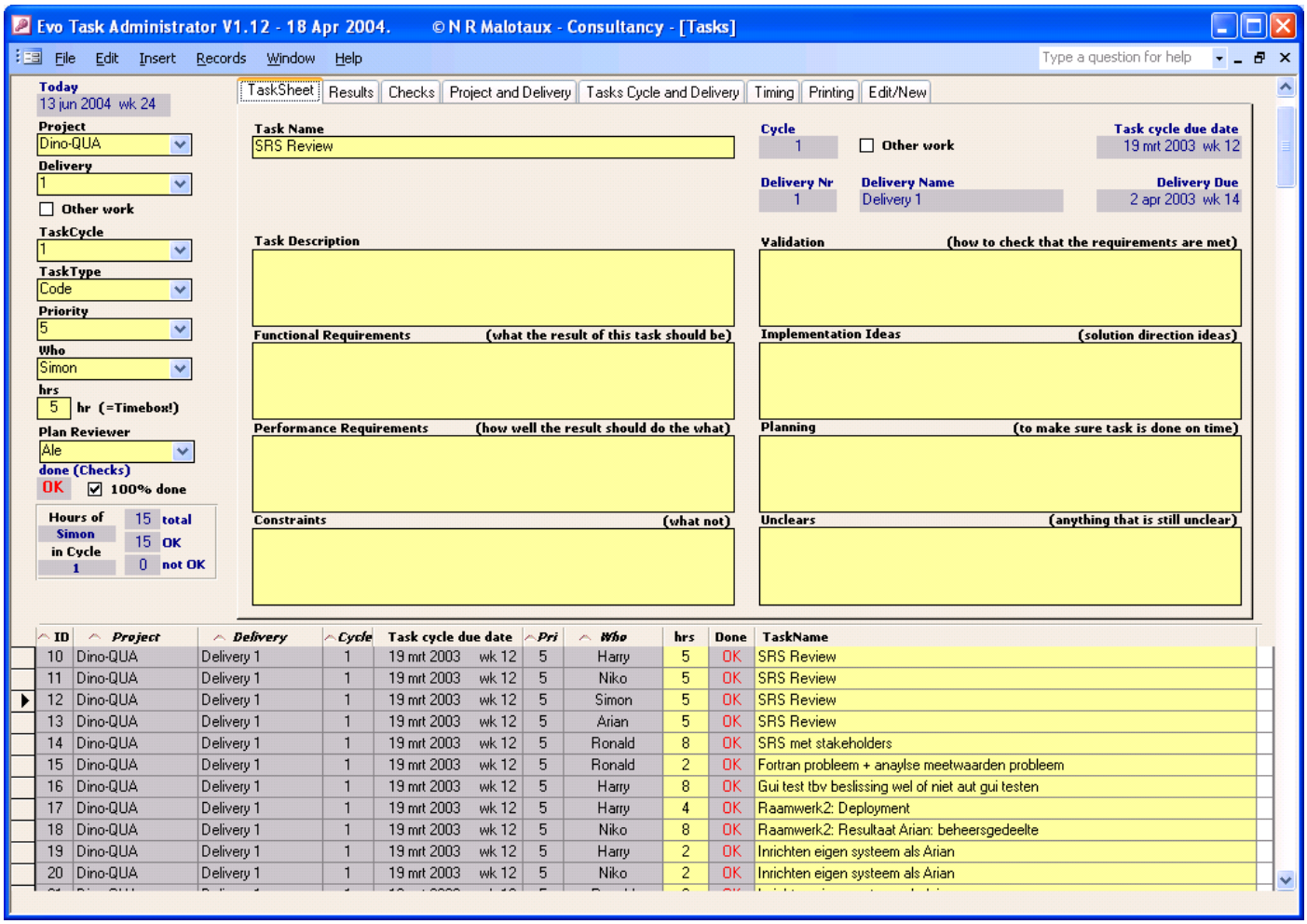


Figure 4: Evo Task Administrator tool screen shot

In all the projects coached since 2002, we introduced the Evo Task Administrator, or ETA tool, which is used to administer the Tasks. This MS-Access application can be downloaded free from www.malotaux.nl/nrm/Evo/ETAF.htm, together with an explanatory text. A screen shot is shown in figure 4.

6.3 TaskSheet

We use the TaskSheet to define what "completely accomplished" means. It helps us to check whether we are going to do exactly what is needed at this moment, not less and not more.
On the TaskSheet we can document:

- The requirements of the Task (Functional: *what*, Quality: *how well*, Constraints: *what not*)
- Task validation: how we are going to establish that the Task's requirements are met
- The strategy to succeed this Task (planning within the Task, design approach)

- Whatever is still unclear

Before starting with the "real" Task, we ask the Project Manager, the Architect, or a colleague to review the TaskSheet. This may take only a few minutes, but it can also take more time. The longer it takes, the more important the review. Most reviews lead to changes in the TaskSheet. That's nice, because we will be working more on the right things than we would have otherwise. After the definition of the Task has been changed, or better defined, the Task time estimate should be reconsidered by the person who is going to execute the Task.

You might be concerned that the TaskSheet takes extra time. Using the TaskSheet doesn't cost time. It *saves* time. Try before you decide. If it ever proves to cost you time, find out why and act accordingly.

Note: If "completely accomplished" is defined as "first half of larger task finished", the TaskSheet should indicate how "first half finished" can be established. Don't settle for weak, un-measurable outcomes.

In the Evo Task Administrator (ETA) tool we have incorporated the TaskSheet for each Task, as shown in figure 4.

6.4 TimeBox

The number of effort hours planned for a Task is a TimeBox: this is the time available for finishing the Task *completely*, *no need to think about it any more*. If a Task proves to need more time than anticipated, don't just use more time:

- People tend to do more than necessary, so we may be able to do less without doing too little. The better the requirements of the Task are defined, the more focused you can go straight for the goal. That's why we use the TaskSheet.
- If you really cannot finish your task within the TimeBox, first complete the other Tasks. These were also chosen to have the highest priority: others may be waiting for their results.
- If you have time left after all other Tasks are done, you may still try to complete the Task.
- If the Task really cannot be finished, check:
  - What did you do
  - What did you not yet do
  - What do you still have to do

Then define new Tasks with estimations. These new Tasks may be considered in subsequent cycles.

If the immediate continuation of the Task really seems to be more important than anything else: use the InterruptProcedure (see below).

Never decide *alone* that you can use more time than the TimeBox. As soon as you find out that the Task is going to need more time than you have available, discuss with the Project Manager: We decided to do this Task, based on the expected outcome (Result) against the expected estimation (cost). If the Task turns out to cost much more than expected, will the investment still be worth it? We might not even have started the Task, so the moment you find out, reconsider the priority: *don't just go on*.

6.5 At the end of the Cycle we Check, Act and Plan:

1. **Was all planned work really done? If a Task was not completed, we have to learn:**
   - **Was the time spent but the work not done?**
     This is an effort estimation problem. Discuss what the causes may be and decide how to change your estimation habits.
   - **Was the time not spent?**
     This is a time management problem:
     - Too much distraction
     - Too much time spent on other (poorly-estimated) Tasks
     - Too much time spent on unplanned Tasks.

     Discuss what the causes may be and decide how to change your time management habits.
2. **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 feel disappointment. We must use the right psychology to feed our intuition properly.
   - Define new Tasks, with estimates, and put them on the Candidate Task List. They will surface in due time. If they do not surface immediately, we apparently stopped at the right time. This ensures that we first work on the most important things.

- Declare the Task finished after having taken the consequences: remember that you cannot work on this Task any more, as it is impossible to do anything in the past.

3. **Now continue with planning the Tasks for the next cycle**

6.6 Analysis Tasks

If it will take significant time to define or estimate Tasks, we define an Analysis Task. In such a Task we don't do anything, we just analyze what we may have to do. At the end of the Analysis Task we check:

- What we know now
- What we still do not know
- What we still have to know

Then we define new Tasks or Analysis Tasks with estimations

Analysis Tasks get a deliberately small TimeBox. After, say, 2 hours we probably know a lot more than before starting. So after the short timebox we can much better define new Tasks or even new Analysis Tasks. By using a deliberately short timebox, we avoid spending more time than necessary. Analysis Tasks allow us to explore Requirements or to explore new techniques: we don't just start, we rather first analyze.

6.7 Interrupt

We know that requirements may change at any time, but we try to keep them stable during the TaskCycle. Sometimes, however, there are interruptions during the TaskCycle. For example: what do you do when the boss comes in and asks you to paint his fence? Or what do you do when a customer of your previous project reports a bug? In Evo, we don't immediately do such things because it's the boss or a customer. We also don't immediately reject the request, because it could be more important than anything else we are doing. However, because interrupts usually seem more important than they may be, we must never decide to change the plan and execute the interrupt on our own. Always consult the Project Manager.

If a new task suddenly appears in the middle of a TaskCycle (we call this an Interrupt) we follow this procedure, based on the principle "We shall work only on planned Tasks":

1. Define the expected Result of the new Task properly
2. Estimate the time needed to perform the new Task, to the level of detail needed
3. Consult the Project Manager, or if unavailable, a colleague. You *must* seek a second opinion.
4. Check the Task planning
5. Decide which of the planned Tasks are going to be sacrificed (up to the number of hours needed for the new Task)
6. Weigh the priorities of the new Task against the Tasks to be sacrificed
7. Decide which is more important
8. If the new Task is more important: replan accordingly
9. If the new Task is *not* more important, then do *not* replan and do *not* work on the new Task. Of course the new Task may be added to the Candidate Task List
10. Now we are still working on planned Tasks

Small interrupts don't need the InterruptProcedure, as long as they don't jeopardize the completion of all the planned Tasks. Because our life is full of small Interrupts (drinking coffee, going to the bathroom, telephone calls, helping each other, and much more), we reserve the unplannable time for these unplannable Interrupts. The InterruptProcedure itself may be handled as a small Interrupt. If it needs more time, define an Interrupt Analysis Task first.

I know this may seem rather formal and bureaucratic. The only reason why we accept the bureaucratic rule in this case is because Interrupts are a big risk for the project and must be handled as such.

6.8 TimeLine

TimeLine is simply a line from Now to Then. We all can apply TimeLine quite well if we have to catch a plane: We know when the plane leaves and count back the time for checking in, the time to go to the airport, the time to get dressed and eat. This leads us to how we have to set the alarm clock the night before to make sure we will catch the plane. We also know that as soon as we can predict that we are going to miss the plane, we can abort the process even before going to the airport: we know we will be late, so it's no use trying any more.

In projects it is not very different, other than that what happens between now and then is a bit more complicated and a bit less predictable.

We call this technique of making sure we will be on time “TimeLine”. It can be used on any scale: on a project, on deliveries, on tasks, the technique is always same:

1. Define a deadline or FatalDate. It is better to start with the end: planning beyond the available time/money budget is useless, so we can stop quicker if we find out that what we have to do takes way more time than we have.
2. Write down whatever you have to accomplish
3. List in order of priority
4. Write the same down in logical groups of Results
5. List these groups in order of priority
6. Translate the groups into Tasks: what you have to do
7. Estimate the Tasks in hours of effort (estimate less urgent tasks in less detail: they will be done later and hence will probably different from what you think now. Don’t waste time on irrelevant detail)
8. Cut the most urgent Tasks into work-Tasks of ~6 hrs effort or less
9. Review the order of the list
10. Ask the team to add forgotten tasks and add effort estimates
11. Get consensus on large variations of estimates (use a Delphi process)
12. Add up the number of effort hours
13. Divide by the number of available effort hours: This is the first estimate of the duration

What the customer wants, he cannot afford

The estimate of the duration is usually way beyond the required duration. At least we know now:

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

We also made sure that we plan to work on the most important issues first and the bells and whistles last.

Now you can discuss this with your customer. If what is surely done is not sufficient for success, you better stop now, to avoid wasting time and money and to spend it on more profitable activities.

In the beginning, customers can follow your reasoning, but still want it all. Remember that they don’t even exactly know what they really want, so “wanting it all” usually is a fallacy, although you’d better not say that.
What you can say is: “OK, we have two options: In a conventional project, at the fatal day, I would come to you and tell that we didn’t make it. In *this* project, however, we have another option. We already *know*, so I can tell you *now* that we will not be able to make it and then we can discuss what we are going to do about it. Which option shall we choose?”
If you explain it carefully, the customer will, eventually, choose the latter option. He will grumble a bit the first few weeks. Soon, however, he will forget the whole issue, because what you deliver is what you promise. This enforces trust. Remember that many customers ask *more*, because they *expect* to get less. He also will get confident: He is getting deliveries way before he ever expected it. And he will recognize soon that what he asked was not what he needed, so why bother to getting it “all”.
The very first encounter with a new customer you cannot use this method, telling the customer that he will not get it all. You competitor will promise to deliver it all (which he won’t, assuming that you are not less capable than your competitor), so you lose if you don’t tell the same, just as you did yourself before using Evo. If, after you won the contract, you start working the Evo way, you will soon get the confidence of your customer and on the next project he will understand and only want to work with you.

6.9 Weekly 3-step procedure

Based on the experience gained, starting with the weekly team meetings we found in most projects, we arrived at a weekly 3-step process, which proves instrumental for the success of Evo implementation. In this process we minimize and optimize the time used for organizing the Evo planning.
The steps are:

1. **Individual preparation**
   In this step the individual team members do what they can do alone:
   - Conclude current tasks
   - Determine what they think the most important Tasks are for the next week
   - Estimate the time needed for these Tasks

- Determine how much time they will have available for the project the coming week

The Project Manager also prepares for his team what *he* thinks are the most important Tasks, what he thinks these Tasks may take (based on his own perception of the contents of each Task and the capabilities of the Individual) and how much time he needs from every person in the Team.

2. **1-on-1's: Modulation with and coaching by Project Management**
   In this step the individual team members meet individually (1-on-1) with Project Management (Project Manager and/or Architect). In this meeting we modulate on the results of the Individual preparations:
   - We check the status and coach where people did not yet succeed in their intentions
   - We compare what the Individual and the 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 estimations
   - 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 an LCD projector at every meeting, even at the 1-on-1's. Preferably we use a computer connected directly to the Intranet, so that we are using the actual files. This is to ensure that we all are looking at and talking about the same things. If people scribble on their own paper, they all scribble something different. The others don't see what you scribble and cannot correct you if you misunderstand something.

   If there is no projector readily available for your project: buy one! The cost of these projectors nowadays should never be an obstacle: you will recover the cost in a very short time.

   There is not just one scribe. People change place behind the computer depending on the subject or the document. If the Project Manager writes down the Task descriptions in the Task database (like the ETA tool), people watch more or less and easily accept what the Project Manager writes. As soon as people write down *their own* Task descriptions, you can see how they tune the words, really thinking of what the words mean. This enhances the commitment a lot. And the Project Manager can watch and discuss if what is typed is not the same as what's in his mind. And when we are connected to the Intranet, the Task database is immediately up to date and people can even immediately print their individual Task lists.

3. **Team meeting: Synchronization with the group**
   In this step, usually at the end of the day, after all the 1-on-1's 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 projection screen (as in figure 4), people read aloud their planned Tasks for the week. This leads to the synergy effect: People say: "If you are going to do that, we must discuss …", or "You can't do that, because …" Apparently we overlooked something. Now we can discuss what to do about it and change the plans accordingly. The gain is that we don't together *generate* the plans, we only have to *modulate*. This saves 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 real important subject that happened to be mentioned later. In the Evo team meetings we select which subject is most important to discuss together.
   - To learn and to socialize.

At every step of the process we try to minimize the number of people involved. First we added the 1-on-1's to the process. The aim was to relieve the team meeting from individual status reporting and from too detailed 1-on-1 discussions. We found, however, that these 1-on-1's easily took about one hour each. One Project Manager said: "Niels, with 6 people in my team, I can just manage in one day. But what would you do if there are 15 people in the team? I want these meetings to take not more than 30 minutes". Watching closely what was happening in the 1-on-1's, we saw that there was a lot of thinking and waiting: "What are you going to do the next cycle?" Pause for thinking. "What effort do you estimate for this Task?" Pause for thinking. "How much time do you have for the project this week?" "I don't know. I have to discuss with the Project Manager of the other project". Sigh. Why didn't you check *before* the meeting? Now we cannot decide!
This led to the Individual Preparation step, where people prepare these issues before the meeting. The result was that the 1-on-1's went from one hour to 20 minutes. That was much better than we expected. The reason is probably that now people come to the meeting much more prepared, needing even less time to get to the point.

Now having optimized the 1-on-1's, Project Managers invariably say that these 1-on-1's are one of the most powerful elements of the Evo approach.
Team meetings usually take not more than 20 minutes. Do we discuss less than before? No, we just discuss the right things effectively and efficiently.

7 Conclusion: How Quality is Assured by Evo

Deming said that quality cannot be tested into a product; it has to be designed in from the beginning. Aren't we doing just that? In Evo projects we define what Quality is and then we pursue the defined Quality, constantly optimizing based on what we learn along the way. All the Quality Assurance people need to do is guide and coach us, watching over our shoulder to ensure we stay disciplined. Not because we like discipline, but because we like success.

-o0o-

Niels Malotaux

How Quality is Assured by Evolutionary Methods

After several years of experience as a Project Coach, introducing Evolutionary Project Management Methods (Evo) in development projects, I think I can claim that Quality is Assured if projects apply these methods. This booklet describes an evolutionary approach to project management that emphasizes that the purpose of development projects is to deliver what the customer needs, at the time he needs it, to create substantially greater value than the cost of development and to enable customer success. We describe the basic Evo approach, followed by several practical details, which the reader can implement tomorrow in development projects, to increase the quality of the products being developed, as well as decrease the time needed to implement the products.

Working the Evo way means organizing the work in weekly (or even shorter) Task-cycles. In these Task-cycles we optimize estimation, planning, and tracking. Task-cycles feed bi-weekly (or shorter) Delivery-cycles by which we optimize the requirements and our assumptions. We use a practice known as TimeLine to create and maintain the total project scope and to connect the Project Result, through the Deliveries, with the actual work organized in Tasks. Evo combines Estimation, Planning, Tracking, Requirements Engineering, Requirements Management, and Risk Management into Result Management. Result is defined as the combined value we provide to all the Stakeholders of our product, ultimately leading to customer success. Evo relies on a fanatical dedication to ROI: Whatever we do should contribute to the Result and we avoid whatever does not contribute.

Deming said that quality cannot be tested into a product; it has to be designed in from the beginning. That's exactly what we are doing in Evo projects. We define what Quality is and then we pursue the defined Quality, constantly optimizing based on what we learn along the way. All the Quality Assurance people need to do is guide and coach us, watching over our shoulder to ensure we stay disciplined. Not because we like discipline, but because we like success.

Niels Malotaux (niels@malotaux, www.malotaux.nl) is an independent Project Coach specializing in optimizing project performance. He has over 30 years experience in designing 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 some 80 projects in 20+ organizations in the Netherlands, Belgium, Ireland, India, Japan, Romania and the US, which led to a wealth of experience in which approaches work better and which work less well in practice. He is a frequent speaker at conferences, see www.malotaux.nl/nrm/Conf .

Find more at: www.malotaux.nl - Evo pages are at: www.malotaux.nl/nrm/Evo

Download booklets:

- Evolutionary Project Management Methods — www.malotaux.nl/nrm/pdf/MxEvo.pdf
- How Quality is Assured by Evolutionary Methods — www.malotaux.nl/nrm/pdf/Booklet2.pdf
- Optimizing the Contribution of Testing to Project Success — www.malotaux.nl/nrm/pdf/EvoTesting.pdf
- Optimizing Quality Assurance for Better Results (same as EvoTesting, but for non-software projects) — www.malotaux.nl/nrm/pdf/EvoQA.pdf
- Controlling Project Risk by Design — www.malotaux.nl/nrm/pdf/EvoRisk.pdf
- TimeLine: Getting and Keeping Control over your Project — www.malotaux.nl/nrm/pdf/TimeLine.pdf

ETA: Evo Task Administration tool — www.malotaux.nl/nrm/Evo/ETAF.htm

N R Malotaux
Consultancy

Niels R. Malotaux
Bongerdlaan 53
3723 VB Bilthoven
The Netherlands
Phone +31-30-228 88 68
Fax +31-30-228 88 69
Mail niels@malotaux.nl
Web www.malotaux.nl
Evoweb www.malotaux.nl/nrm/Evo

Originally prepared for the Annual Pacific Northwest Software Quality Conference, Portland, USA, 2004
Version 1.0g (updated PDCA picture) - 1 June 2008