Recognizing and Understanding Human Behavior to Improve Systems Engineering Results

Niels Malotaux
N R Malotaux - Consultancy
Bongerdlaan 53, 3723 VB Bilthoven, The Netherlands
+31-30-2288868 - niels@malotaux.nl

Abstract. One of the sub-systems that's always part of any system is the human. Therefore, we have to understand both the interface and the behavior of humans, in order to let them work properly within the system. In practice we see however that many systems fail because engineers ignored, forgot to include, or incorrectly assumed how people interface and behave.

This paper describes human behavior elements which we have to recognize and understand for all the various phases of the Systems Engineering, because humans are involved not only in the operation and use, but also in the conception, ordering, design, realization and maintenance, and (in)tolerance of systems. Systems Engineers even have to recognize that they are humans themselves, with a behavior to be understood well in order to successfully realize and improve the results of their projects. Human psychology is an integral part of Systems Engineering, whether we like it or not.

Note - This is not a scientific paper but rather based on empirical evidence collected by coaching over 100 projects in the past 8 years.

Not all systems are perfect

Engineers learn to combine components into systems, which then may become the components of systems of systems. In order to let these components work together properly as a system for the proper purpose, engineers have to understand both the interface and the behavior of these components. Conventional engineering tends to sub-optimize the results within a limited field of expertise. Systems Engineers are supposed to optimize the whole system over several dimensions: the total life-cycle of the system, all disciplines involved, and making sure that the system as a whole performs properly for all Stakeholders.

Although there are systems being successfully developed, a lot of systems developed deliver less than expected, are not even put into use, take way more time than expected to realize, or need a lot of tuning before the performance becomes acceptable. If Systems Engineering were perfect, then all systems developed would be perfect. Not all systems developed are perfect, so apparently Systems Engineering is still doing some things wrong.

Murphy's Law

Whatever can go wrong, will go wrong is the popular version of Murphy's Law. However, the real version for engineers is: *If it can go wrong, it will go wrong, therefore we have to predict all possible ways it can go wrong, and make sure that these cannot happen* (Stark 2006). Systems Engineers, who are supposed to optimize the whole system over all necessary dimensions, have to predict all possible ways how things can go wrong, so

that they can make sure that the system is designed in such a way that success is guaranteed.

Human behavior

Many risks that threaten the success of the system are caused by human behavior, or rather ill-understood or even ignored human behavior. Things that can go wrong by humans acting in unpredictable ways are caused for example by:

- customers not knowing well to describe what they really need
- users not understanding how to use or operate the system
- users using the system in unexpected ways
- developers and systems engineers, who also happen to be humans, doing wrong things during the development of the system

Actually, in these examples, the humans aren't acting unpredictably at all, because it happens again and again in many systems and in many projects. If we don't learn to understand why people act like this, projects will continue to be affected by these issues.

Real human behavior

Based on our cultural, social and technical background we consciously or even subconsciously assume certain human behavior. When humans do not behave like we assume they should, the behavior seems unpredictable. When behavior is unpredictable, it is difficult to create proper control functions with humans in the loop. Even if the engineers don't forget to include human behavior, they may find out that the humans in the system don't behave as expected, with unexpected results.

Example: the engineers who designed and built the baggage handling system of London Heathrow Airport Terminal 5 claimed that their system was a huge technical success and that the failure to get tens of thousands of bags on board of the proper aircraft was caused by "human error". After all, the terminal was delivered on time and on budget, which admittedly was quite an achievement. However, a passenger is not interested in the technical detail of baggage handling in one airport. The passenger checking in his baggage expects to receive it back in correct condition as quickly as possible after arriving at his destination. Everything in between is irrelevant to that passenger.

If we can overcome our intuitive tendency to assume how people *should* behave and start studying how people *really* behave, human behavior turns out to be much more predictable than we think. Therefore, understanding of real human behavior and the incorporation of this behavior in the loop should be an integral part of Systems Engineering in order to create really successful systems.

The behavior of people responsible for success

Project Management is responsible for the success of the project producing the system. Therefore, Project Management must understand the behavior of all people involved in the project and adapt to this behavior to make sure that things that can go wrong don't go wrong. During the project, Project Management can reach all these people involved, can observe what tends to go wrong and make sure it doesn't.

Systems Engineers have an even wider responsibility, being responsible for the success of the system not only during the project, but also after delivery, in operation, maintenance, and disposal. After the project, however, the people involved are beyond

the influence of the Systems Engineers, so that the system must be designed in such a way that success happens by design, automatically. This calls for thoroughly understanding how humans behave, to make sure that the system, together with the humans using it, and the humans being affected by it, successfully performs its mission.

Before understanding the particularities of other people's behavior, it's good to start with understanding our own behavior and from there extrapolate and extend our understanding of all types of behavior.

In the following, we'll discuss some elements of human behavior which may pose risks for the successful and timely delivery of the systems our projects are supposed to produce.

Communication

Merriam-Webster dictionary (www.m-w.com) defines communication as '*a process by which information is exchanged between individuals through a common system of symbols, signs, or behavior*'. A problem we should be very aware of is that the information we think is exchanged, often is diluted, distorted and misunderstood, if it is received at all.

Witnesses tell a story they made up in their mind. We hear a bang behind us, we turn around, and we see the scene of a traffic accident. That very moment, our grey cells start constructing a story of this accident. It is based on the basic concept of traffic accident, which was constructed earlier based on what we saw on television, or on earlier experienced accidents. We make up how the accident came about. We didn't even see it happen, but we assume how it may have happened, and what will happen afterwards. We mold the standard concept of traffic accident in our mind with what we see, hear and smell, into an instance of this particular accident. Because different people start with different concepts of accident and see different elements of the scene, every witness tells a different story (men may notice that there is a garage at the corner, women may notice that there is a fashion shop right behind the demolished car). Witnesses don't lie about it, it's just that every person has a different history to start with and sees different details, and hence composes a different story in his mind.

Throwing sounds to one another. The same happens with words we use when we try to communicate: we throw sounds to each other and hope that in the mind of the other person the same concepts emerge as we have in our own mind. Because we all have different histories and different interest in details, the concepts we try to activate in the minds of others are probably different from the concepts we think we are conveying.

"*But I told you!*"… Well, did the other person 'receive' the sounds we threw? Was he really paying attention or was he dreaming? If he received the sounds, how were these interpreted? "*You nodded in agreement!*"… Well, may-be he just moved his head to look into another direction and we only assumed that this movement was an acknowledgement. May be the other person lacks some concepts in his mind, so that he even *cannot* imagine what we want to say.

The more distance in descent, the more difference we may expect from the interpretation of the sounds we exchange. In one family we may already have differences in interpreting the same word. In our work environment the differences are probably greater and if we try to communicate between people from different cultures (think about off-shoring) the differences in concepts in our minds caused by the sounds we exchange are even greater, especially if we try to communicate in a non-native language.

Our mind is quite happy with fuzzy thoughts. An additional problem with communication is that the concepts in our minds are rather fuzzy: we think our thoughts are clear, but in reality they are not. If you ask somebody "*Do you have a plan?*" the answer may be: "Yes I made a plan". "*Where is it?*" "It's in my head!" Then ask: "*Can you write it down?*" "Why should I, it's completely clear what I'll do". "*If it's so clear, it shouldn't be too difficult to write it down, so that I better understand what you'll be doing. Can you still write it down?*" Now, if the person tries to write down the plan, it suddenly becomes clear that the plan wasn't all that clear in his mind. He starts writing it down, starts changing, adding and moving the order of things. "*And you said it was all clear in your mind?*"...

Our mind is quite happy with fuzzy thoughts. That's probably a survival strategy: Fuzzy thoughts may create errors in our thoughts and if we suddenly recognize something new in an erroneously generated thought (this usually happens in our sub-consciousness), we call it *creativity*. But in communication it is a risk when the receiver interprets the message differently from what the transmitter of the message wants to say.

Explain it to a colleague. Did you ever encounter that you were thinking about a problem and got stuck? You go to a colleague and start explaining, and suddenly you say "*Ah, now I see!*" Because you tried to tell what was in your mind, you had to *unfuzzy* it for the other person, at the same moment unfuzzying it for yourself. The other person didn't even say a word. He may even still be puzzled about what you were going to say …

Explaining it to paper. In order not to bother our colleagues too much, why not explain to paper what's in our mind? Explaining to paper is called *documenting*. Thinking along these lines, it becomes clear that documenting is not in the first place for *others*, but rather *for ourselves*, to unfuzzy and hence understand out own thoughts better. Now we can also understand why most people don't *like* to document: When we have to write down what seemed so clear in our mind, it proves to be so difficult. It's easier to skip the documenting and stick to the false feeling of clearness in our fuzzy mind. It's a risk that the unresolved fuzziness will cause a problem later. If it does cause a problem later, we conclude that '*to err is human*', as if it's just fate and that we couldn't have done anything about it. Once we recognize that the fuzziness can be reduced by documenting what's in our mind, we understand that it's not just inevitable fate. Many mistakes can be avoided by proper understanding why we are making them and then doing something about it!

If we write it down, it can be discussed and changed. What we don't write down, we cannot discuss because other people cannot see what we are thinking. If it's written down, we can discuss and change it. "*But I cannot yet write it down because it's not yet clear enough!*" Then the advice is: Write anything down, even if you know it's incorrect. When you write it down, others can see it and start helping you to make it better quickly. And again: by writing it down, you may see a solution yourself easier as well. This is why we need large whiteboards in every meeting room. Do *you* have large whiteboards in your own room and in your meeting rooms?

In any meeting with more than one person, we use a projector. In meetings people scribble notes. The problem with these notes is that many people don't write clearly, so that later they cannot decipher half of their scribbling any more. If we compare the scribbling of the various people in the meeting, we will see that they all write things differently and that there are inconsistencies in what they have written down based on their perceptions (see next) of what has been said. Furthermore, while people write

down their scribbling, their attention is distracted from what is being said, so they *miss part of the discussion*. This poses a risk that people start working inconsistently or on inconsistent things. When this is found out later, people may have to repair the inconsistencies. If this can be avoided it saves time.

Therefore we use the rule: "*In any meeting with more than one person, we use a projector, with a computer connected to the intranet.*" Now, in stead of everybody scribbling his notes on a piece of paper, we have a centralized place where things are written down legibly and saved for all involved, easily to turn to later. We use as second rule: "*The owner of the text writes it down.*" If the person responsible for what is written down doesn't write it down himself, we see that that person may nod in agreement "*Yes, that's about it*", but there is no real commitment. If the person writes it down himself, we see a greater responsibility for the correctness of what he writes down. This causes a better commitment for what is being written. In stead of being distracted while scribbling, the other people now can keep their attention to what is being written and react on it. If they don't understand or disagree, they can ask for an explanation. This provides better communication. The cost of the projector is regained quickly because people work more efficiently.

The commitment for what's written down is also very much enhanced because we all saw it in big size projected on the wall. That has psychologically a lot more impact that if we have seen it on paper or on a monitor screen.

Perception

What people *say* when asked and what they *do* when they actually have to decide may be different. 'Customers who said they wanted lots of different ice cream flavors from which to choose still tended to buy those that were fundamentally vanilla' (Åfors and Zuckerman, 2001).

What *we* think people say and do may even be not the same as what *they* think they say and do. It's both about *their* perception and *ours*.

Different people perceive the 'same' things differently (remember the traffic accident). Because of this, we may even wonder whether we then still can speak of the 'same' things …. Perception is what we intuitively, sub-consciously observe and notice. These silent observations influence our interpretations, decisions and actions. We don't even realize that we logically (with the conscious mind) think one thing and still with our emotions decide another thing: the head knows, but the heart decides.

Asking the customer to produce the requirements of the system will usually not produce the *real* requirements. Customers usually aren't even users of the system, which makes specifying the correct requirements of the system even more difficult. For customers, writing requirements isn't a core business. For Systems Engineers, it is. The problem with requirements engineering is, however, that it's not well taught in education and that it's even partly a craft that has to be mastered by attitude and experience.

Furthermore, what the customer wants he cannot afford, so if we start making what the customer 'requires' we'll probably fail from the beginning. After all, the requirements are what the Stakeholders require, but for a project, the requirements are what the project is *planning to satisfy*. This difference often causes a problem, if the perceptions aren't managed well. Because of the perception issue, trying to find out the real value to the customer, or to the many Stakeholders, can show many paradoxes.

Better not simply *believe* what they say: we have to *check*! An additional problem is that many people cannot very well imagine a system from a description or from written requirements. They need to see more physical representations of the system to be able to understand what the requirements really mean. Asking such people to sign-off the written requirements is pointless.

Developers, engineers, Systems Engineers are deformed. Developers, engineers and Systems Engineers are deformed in their perception what 'normal' people find 'normal'. Software engineers find it quite normal if the system asks "*OK or Cancel?*" Normal people don't even expect to see that question. Software engineers think a "Save" button is quite normal. Normal people don't see any value in a "save" button. After all, if they talk to someone, they also don't need to press a "Save" button in order for the other person to start receiving the message, do they?

When I built a website with forms to fill in, without the need of a save button, the software developers asked "*Where is the Save-button?*" Other people never asked.

If we as engineers try to imagine the right interface of the technical parts of the system with people, we are probably wrong. We simply cannot imagine any more how normal people want to or will use our system. Therefore it's better to mistrust our assumptions and check them all the time. Better assume that a lot of the requirements and our assumptions are wrong and that we have to find out what the real and right ones are.

Intuition

Intuition makes us react automatically on common situations. Our sub-consciousness provides the solutions for these situations the moment we need them.

We live intuitively. When we are hungry, do we first collect all the possible solutions of alleviating our hunger, evaluating which solution provides the best Return on Investment and then choose which option to use, based on documented criteria? By the time we are done with our analysis we may be starved to death. In practice we simply find something to eat and eat it.

Intuition is fed by experience. Since we are born, we learn from experience to intuitively avoid bad situations and seek nice situations. If we enter a room, we intuitively know how to move around, without bumping into the tables. The more experience we gather, to more complex situations we can intuitively handle.

Intuition is not perfect. We also do our work mostly intuitively. This works well in many cases, but, especially in engineering, where we often encounter new situations or problems, intuition may not produce the right solution. Considering that not everything in our projects runs perfectly, apparently intuition doesn't always point us into the right direction. Probably we don't have the right experience for all of the situations we encounter in projects.

Intuition is free, we always carry it with us. We cannot even switch it off. It's so strong that it's almost impossible to go against it. An example of a typical counter-intuitive situation is a software project with a team of 20 people going too slowly. If the deadline is really hard, the usual intuitive move to go faster is *adding people*, which probably will make the project running even more slowly. The counter-intuitive move: *decreasing* the number of people in the project, is so incredible, that hardly any project manager even dares to contemplate this solution. Still it would make the project probably running

faster (Brooks' Law, 1975 - ref Malotaux 2007, chapter 4.1). In order to improve the performance of people in projects, new experience has to be created to improve the intuitive response. A Project Coach[1] can help to provide new experience.

Intuition versus a Quality Manual with written procedures. Procedures are formalized best practice: to our current knowledge, this is the best way to do it. Documenting procedures in a Quality Manual is not bad at all, but in practice the document is hardly read by the people who should execute the proper procedures and even if the person knows the procedure, it's a discipline risk (see next) that the procedure is not always properly followed. Once we put the procedure into the intuition of the people, it will be executed "automatically", without the need for "following" the written procedure. Still we should make sure that at the same time we also learn to apply continuous improvement, so that we learn to challenge our intuitive reactions all the time, in order to continuously tune these reactions to the actual situation, which may change over time.

In many cases the head knows, the heart not. We think that we make decisions logically with our mind. However, we mostly decide with our hart (or call it gut-feeling or emotions, fed by our sub-consciousness). If you think that this is wrong, it actually proves the point. Logically we think that people should decide with their mind, but in practice we see people react emotionally, with their intuition, fed by their perceived experiences. All we can do is feeding the intuition with our thinking and trying to bias or intuitive decision process into the right direction.

Sleeping on it. Recent research (Dijksterhuis, 2007) indicates that with complex problems, logical thinking produces worse decisions than 'decisions' made by intuition, because our mind isn't capable of balancing more than a few elements at the time. With logical thinking we often focus on less relevant factors, 'forgetting' some more important factors. However, we can make an even better decision if we first think logically, set a deadline to decide, in the meantime do something entirely different and then decide. Apparently our sub-consciousness went on processing when we did the other thing and presented us with a better solution. Sub-conscious processing proves to be much more powerful and more capable of complex correlations than our conscious thinking. Hence the saying that we should "*sleep a night on it*".

The users of our system. The designers of the system should be aware of the power and the risks of working on intuition during the project. We can try to adapt our behavior to producing ever better results. The users of our systems also use intuition when being confronted with our system, but we cannot influence their behavior. We have to analyze how they actually behave, check whether our assumptions about their behavior are right and if not, adapt. If the users incorrectly use our system, in principle the system is wrong, not the users.

[1] This may look like selling the concept of a Project Coach. It is. Whether this is a coach from inside or outside your organization is not the point. If you have internal coaches, great. If you don't, hire a coach to show how to do it and to coach the coaches to become self-sufficient quickly.

Discipline

Discipline can be defined as: *control of wrong inclinations*. With discipline I do not mean what others impose on us. I mean the discipline of doing things how we know they should be done. We know how we should do it, but if nobody is watching, we do it an easier way. Don't we? This creates a risk that we will have problems with quality later (if it doesn't create a risk, we should catch it as a potential better way of working). Even while we know this is a risk, *easy now* easily prevails over *problems later*. Discipline is difficult.

When in a lecture I suggested that the Bible as well as other religious books probably talk about discipline, someone in the audience immediately replied: "Yes, it's written in chapter Romans 7-19: "*The good that I want to do, I do not. But the evil which I don't want to do, I do.*" This event taught me that the discipline problem in humans is known for thousands of years and who are we to think that we can suddenly change that? We won't. We cannot fight the genes! In stead of wasting time fighting the genes, is there still something we can do to decrease the risk of lack of discipline?

I found three things we can do to somewhat improve on the discipline problem:

Helping each other. It is easier to keep the discipline to do the right things right if somebody is watching over our shoulder. This is why in organizations with many one-person projects, we let *two* people work on *two* one-person projects. This way they can watch over each other's shoulder, helping each other to do the right things. In a larger project, people can watch over each other's shoulder, helping each other to keep the discipline of doing what is best for the best result in the shortest time.

Rapid success. If we ask people to keep the discipline of the process from now on, we actually ask them to change their way of working (otherwise we didn't have to ask). Some people claim that people resist *change*. I think people don't resist change, they rather (subconsciously) don't like and hence shun *uncertainty*. Change creates uncertainty. This makes people seem to dislike change. If we ask people to change their way of working, telling that if they work hard, in about two years things will go much better, people have to endure the uncertainty of improvement for a long period of time (example: moving from CMM level 1 to CMM level 2 takes about two years[2]). That doesn't work well. People cannot cope with uncertainty for such a long time.

With the help of a coach, it's acceptable to bear uncertainty for a few weeks. If the coach says: "*Do this three weeks for me*", and within two weeks people feel that the suggested better way of working really is successful for them, then there is a chance that they keep doing it. We should create rapid success and adapt our rate of change to small consecutive steps.

Making mistakes. People learn more easily from mistakes if they feel the pain of the mistake. If we suggested another way of doing and the person insisted in his own way and failed, *and felt the pain*, then there is a chance that the person now will accept our suggestion. If that suggestion quickly works better for him, the person now may accept it.

[2] www.sei.cmu.edu/appraisal-program/profile/pdf/SW-CMM/2006marSwCMM.pdf , slide 26.
CMM: Capability Maturity Model, which sets goals for process improvement, (purposely) not in detail defining how to achieve these goals.

Insanity

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)

Only if we change our way of working, we may expect the result to be different. Hindsight is easy, but reactive. Using hindsight we can learn from what we did right and what we did wrong. But that's in vain unless we use what we learnt as input to foresight. Foresight is less easy, but proactive: we can prevent doing something that later will prove to be wrong or unnecessary, or we can do something better than we did before. We may not be able to foresee it *all*, but all that can be prevented will save time.

How do projects get late? One day at the time. How do projects get earlier? By saving time every moment from the very beginning. If we start saving time at the end of the project, there is not much time to be saved any more.

This is why we use the Plan-Do-Check-Act or Deming cycle (see Deming 1986, Walton 1990, Malotaux 2006 - chapter 6). If we just can keep the discipline to frequently applying the Plan-Do-Check-Act cycle, then we can constantly challenge and optimize our discipline, intuition, communication and perception of how we best handle the product (how we engineer an optimal solution), the project (how we optimally organize the engineering of the optimal solution) and even the processes (how we optimize all of this), overcoming the insanity.

Evolutionary Project Management

Evolutionary Project Management (Evo) is a project management approach that actively applies study and knowledge of human behavior to constantly improving project results. See (Gilb 1988, 2005) and (Malotaux 2001, 2004, 2006, 2007).
Elements of Evo are:

- **Plan-Do-Check-Act**
  The powerful ingredient for constant improvement, proactive action, and checking and acting upon the effects of human behavior (as well as upon any other things we have to take care of).
- **Zero-Defects attitude**: We are not perfect, but the customer should never find out.
- **Business Case**: *Why* we are developing the new system.
  Most projects don't even have a clue of their Business Case, and whether their endeavor will have a positive return on investment. If the return on investment is undefined, people don't realize that every day delay potentially costs much more than only the cost of running the project. It also includes the missed revenues of the new system. If missing these revenues is not an issue, the project probably is not worth the investment anyway and we should better work on something else. Note that return on investment is not solely about money. It's about creating value. Still, even in the public environment many projects do influence the economy, so a day earlier or later can have an important economic value.
- **Requirements Management**: What we are going to improve and *what not*?
  - What the customer wants he usually cannot afford. So it's as important to define which Stakeholder requirements we are going to satisfy and *which not*.
  - How much we will improve: quantification. The functions we deliver in a project are already there. How well the functions are performed is the real reason for the project. If the project delivers a technically perfect system which doesn't perform

well because it's not or incorrectly used by the users, the value isn't realized, so the project still fails.
  - We are well aware that however good or bad the requirements at any time may be, they will change over time, because *we* learn, *they* learn and the circumstances change.
- **Architecture and Design**
  - Selecting the optimum compromise for the conflicting requirements. Requirements are always conflicting. For example if we improve a certain performance more, the available budget goes down.
  - We use methods like Impact Estimation (Gilb 2005) to quantitatively measure and plan the impact of what we do towards optimally achieving our goals. Impact Estimation is a technique for communication and better understanding, and for feeding our sub-consciousness to make better decisions.
- **Early Review & Inspection**
  - We are not perfect, so if we've done something, we know that we did some things wrong.
  - So, we ask others to check what we're doing, preferable *before* we're done, so that we can start preventing what we are doing wrong as quickly as possible.
- The weekly **TaskCycle** is designed to:
  - Plan before doing, to unfuzzy our ideas about what we should do.
  - Plan before doing, so that we still can avoid doing things that later will prove to be unnecessary.
  - Quickly change from optimistic to realistic estimation.
  - Be able to promise what we can achieve, and living up to our promises.
  - Actively communicate with people internally and externally to the project.
  - Handling interrupts to minimize time lost. Interrupts usually seem more important than they are.
- The bi-weekly **DeliveryCycle** is designed to:
  - Check and optimize the requirements and (our and their) assumptions.
  - Solicit feedback by delivering Real Results to appropriate and *eagerly waiting* Stakeholders. We need feedback because probably we (as well as they) don't understand the requirements well and several of our (and their) assumptions are probably wrong.
  - Manage perceptions.
- **TimeLine** is designed to:
  - Getting and keeping control of Time.
  - Estimating the time needed for all we think we have to do.
  - Calibrating our estimations based on *Earned Value* in order to predict how much *Value Still to Earn* we can achieve in the time remaining, or how quickly we can deliver sufficient value so that the users already can start earning our salaries.
  - *Taking the consequence* if the value we can deliver in a certain time doesn't generate sufficient Return on Investment. Time is usually one of the most important requirements. Still, note that as most projects deliver late, they apparently don't treat the time requirement as seriously as all other so called requirements.

5 times "*Why?*"

We ask from our customers or Stakeholders what they want or what they need. In many cases the Stakeholders come up with their wishes rather than their needs. In many cases the Stakeholders do not even recognize their own needs, or the problems they need to be solved in order to improve whatever they are doing.

From Freud and Jung we can learn that the problems of people mostly stay in the sub-consciousness and that (perceived) solutions come out. If we start developing these solutions, we are probably developing a perfect solution for the wrong problem.

We'd better ask "*What's your problem?*" which usually causes as an answer: "Problem? I don't have a problem." Then you say: "*If you don't have a problem, I don't have to do anything for you*". After which the Stakeholder says: "Ah... Ok, I want you to do this because …". Then you ask: "*Now why is this a problem?*" Now the Stakeholder starts to understand what's going on: "Well, this is a problem, because …" Great, we're getting closer. "*Now why is that a problem*". This technique is called the "*Ask 5 times Why*" technique, which usually in 3 to 5 steps yields quite a good idea about the real problem that has to be solved.

What would have happened if we had started implementing the solution the customer gave us in the first place? That's why we say: *First develop the problem, then develop the solution, and only then develop the implementation.*

Conclusion

We discussed several elements of human behavior and how these can affect the success of our Systems Engineering projects. We also very shortly introduced the Evolutionary Project Management approach, which is fully aware of the risks of human behavior and constantly tries to improve the way we run projects and design systems, taking into account the behavior of all people involved. That's not only customers and users of our systems, but first of all, our own behavior and how that can be improved to create success faster and more predictably.

References

Åfors, Cristina and Zuckerman, Marilyn, 2001. A Quick, Accurate Way To Determine Customer Needs. *Quality Progress*, Vol. 34, No. 7, July 2001, pp. 82-87. ASQ. www.culturalimprint.com/emerging needs.pdf

Deming, W. Edwards, 1986, *Out of the Crisis*, MIT, ISBN 0911379010.

Dijksterhuis, 2007. *Het slimme onbewuste* (*The smart sub-consciousness*, in Dutch). Bert Bakker. ISBN 9789035129689.

Gilb, Tom, 1988. *Principles of Software Engineering Management*. Addison Wesley. ISBN 0201192462.

Gilb, Tom, 2005. *Competitive Engineering*. Elsevier. ISBN 0750665076.

Malotaux, Niels, 2001. *Evolutionary Project Management Methods*. www.malotaux.nl/nrm/pdf/MxEvo.pdf

Malotaux, Niels, 2004. *How Quality is Assured by Evolutionary Methods*. www.malotaux.nl/nrm/pdf/Booklet2.pdf

Malotaux, Niels, 2006. *Controlling Project Risk by Design*. www.malotaux.nl/nrm/pdf/EvoRisk.pdf

Malotaux, Niels, 2007. *TimeLine: Getting and Keeping Control over your Project*. www.malotaux.nl/nrm/pdf/TimeLine.pdf

Spark, Nick T., 2006. *A History of Murphy's Law*, Periscope Film, ISBN 0978638891.

Walton, Mary, 1990. *Deming Management At Work*, The Berkley Publishing Group, ISBN 0399516859.

Author

Niels Malotaux is an independent Project Coach specializing in optimizing project performance. He has 35 years experience in designing embedded, 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 100 projects in 20+ organizations in the Netherlands, Belgium, Germany, Ireland, India, Japan, Romania and the US, which led to a wealth of experience in which approaches work better and which work less in practice.