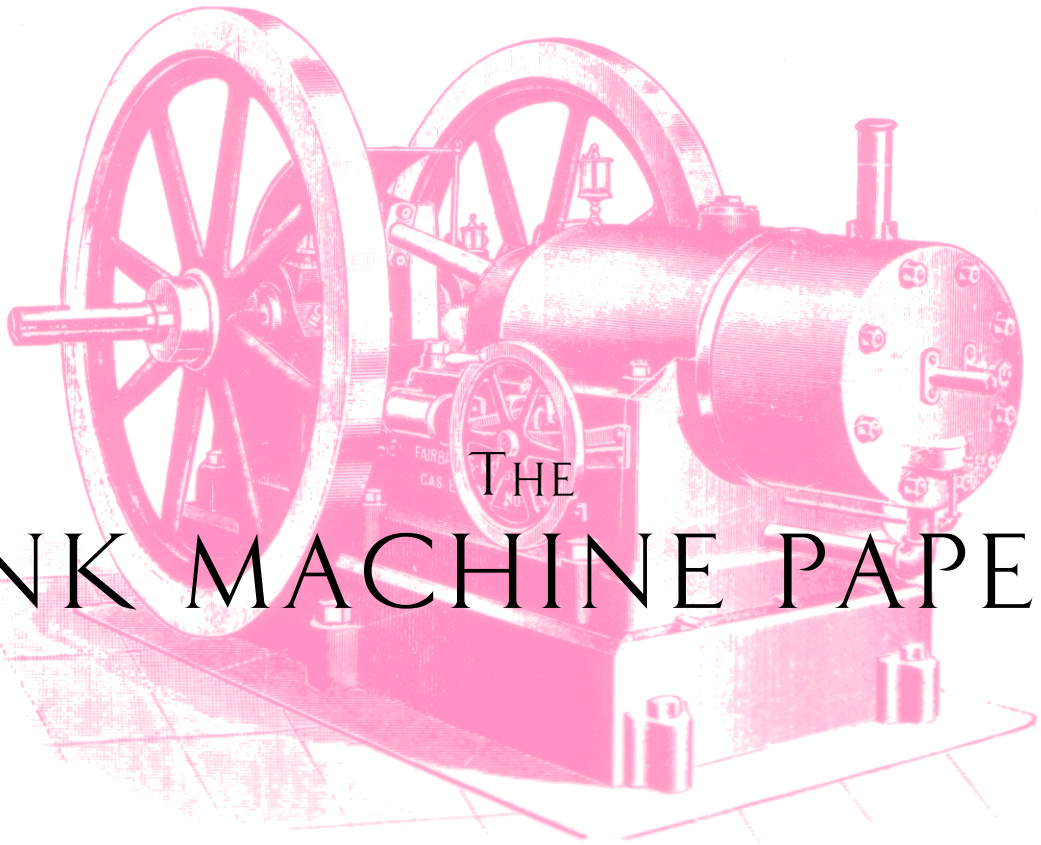


The Pink Machine Papers

#30 – Nº 3 / 2006

Functions of an estimate

Erik Piñeiro

#30
Nº 3 / 2006

Functions of an estimate

Erik Piñeiro
Royal Institute of Technology, KTH
Dept. of Industrial Management, INDEK
Stockholm, Sweden
erik.pineiro@indek.kth.se

The Pink Machine Papers
ISSN 1650 - 4062
Stockholm, 2006

This working paper reports some of the initial findings of a research project that is to be conducted from Jan 06 to Dec 07. Please note that it is work in progress.

An estimate should, ideally, be an objectively obtained number that tells us the resources needed to complete a project. Such a number is important because it, in theory, allows the developing company to present a plan and a price to the customer. Numbers presented by different bidders can be compared, and the customer can then choose the most interesting alternative. Hence, the function of this ideal estimate seems to be, on one side, to allow for realistic pricing and planning of a project and, on the other, to allow for comparisons.

However, everybody seems to know, and accept, that estimates are neither objective nor always accurate. Occasionally, they may even be totally artificial. And nevertheless, estimations are still carried out, offered and considered essential. Probably the reason behind this, as will be argued in the continuation, is that the most elemental function of an estimate is not to offer the correct answer to "How much will this project cost?" but rather to reduce different kinds of uncertainties that make decision-making impossible. In the process of obtaining the estimates, and thus of reducing those uncertainties, other goals are accomplished, such as that of encouraging engagement and motivation and that of particularizing the specifications. The ideal goal, that of giving a correct answer to "How much will this project cost (or how long will it take)?" is never quite achieved. It might in fact be unachievable.

Eliminating uncertainty

The book that perhaps deals best with the notion of uncertainty in organizations is James D. Thompson's Organizations in Action, originally published in 1967. The main tenant in this book is based on a double assumption:

On one side, in order to achieve maximum effectivity, organizations need to operate under conditions of total certainty. Such conditions would make it possible to plan and structure the organization in an optimal manner.

On the other side, conditions of total certainty are impossible to achieve. Thompson's book discusses the effects of "irregularities stemming from external sources" (:12); leaving aside irregularities that originate from internal sources.

Now, Not all of what Thompson says in that work is directly transposable to our study but the essential idea is:

> *... the central problem for complex organizations is one of coping with uncertainty. As a point of departure, we suggest that organizations cope with uncertainty by creating certain parts specifically to deal with it, specializing other parts in operating under conditions of certainty or near certainty. (:p13)*

We shall not look at parts created in order to deal with uncertainty but at numbers (estimates) created to deal with uncertainty so as to allow the organization to act as if under conditions of (near) certainty. Let us now see what uncertainties an estimate reduces, both on the developer's and on the customer's side.

Speaking in general terms, deciding to get on with an IT project means involving oneself with uncertainties. The most pressing one is that of whether the system can be carried out at all, a question highlighted in Software Runaways (Glass 97), where the author presents a number of IT projects that collapsed under way. In the cases we have studied, however, the projects are of such a nature that this question is seldom actual.

Two other important uncertainties that do apply to all our cases are: 1) when will the project be finished; and 2) how much it should cost. The most obvious way to eliminate this uncertainties is to find the correct answer to the questions. Unfortunately, no-one seems to know how to calculate those numbers in a correct way. So instead, the answer to these questions is first offered as an estimate, then negotiated and finally agreed upon and included in a contract. This process reduces the uncertainties for both the developer and the customer. They both need an agreed-upon estimate: the customer in order to plan the budget and the introduction of the system, training, new routines, etc.; the developer in order to plan its development (and other parallel developments that might exist).

It might feel counter-intuitive to think of a way to reduce the above-mentioned uncertainties that does not include objective and accurate estimates. But this is a misconception: the uncertainty is present at the beginning of the project, and at this date what is needed is a believable number to agree upon. Once this number has been agreed upon, the uncertainty has been reduced. It has not been eliminated because everyone knows that estimates can be wrong, but both customer and developer can now move on. They can both plan the coming activities and start working on them.

Estimates are, as I mentioned above, not always accurate, and gravely optimistic ones (we shall ignore the case of pessimistic estimates since they seem practically not to exist) will give raise to problems. Everyone is aware of this, but those problems lie ahead and the current processes can be carried out without disturbance. In fact, those future problems may very well be insurmountable, be indeed so grave so as to throw companies out of business (cf. Glass 97); so I am not suggesting that any number will do as an estimate simply because both customer and developer have agreed upon it.

Early estimates should naturally be as accurate as possible, but they do not need to be in order to reduce the uncertainty present at the beginning of the project. This is essential because it explains why companies can do well without perfectly unfailing estimation practices: they have established mechanisms that allow them to adapt the project to the estimate:

- Well dimensioned buffers.
- pre-study phases.
- formal modifications to the specifications which entail re-estimations.
- trimming of the original specifications.
- developers ready to work under extreme conditions.

This mechanism can be implemented differently: developers that love programming, radical management routines, etc. We have not come across this kind of mechanisms in our cases but it can be found and commented in much of the literature about software development projects. Kidder's *The soul of a new machine* tells a good story and is a great place to study development under extreme conditions.

Another important uncertainty that is seldom considered but that has great effect in the project is that of unclear specifications. As one of our interviewees said regarding how much information they had to build the first estimate on: "... while other customers phone you up on monday and want a price on friday." Or as another said, more moderately: "the received specifications hold varied quality and they set the standard for what is possible to estimate." The estimate itself, the number, does not reduce this specifications uncertainty, but some process of estimation do, as we shall see in the following section.

Clarifying the specifications

As mentioned above, one of the greatest risks in a software development project seems to be unclear specifications. Theoretically, unclear specifications should not be allowed into a project, but in reality they very often are. Estimates, per se, are not directed towards clarifying specifications, but we have seen that the process of estimating a project will often result in a clarification of the specifications.

One can identify two main steps in the clarification of specifications: the first one has to do with obtaining specifications of at least a certain level of detail, to avoid the monday-to-friday specifications situation cited above. Being forced to offer an estimate will both urge developers not to accept poor specifications and offer them the justification to require more detailed documents: they can argue it is also in the customer's interest to provide with as clear requirements as possible.

The second step in the clarification of specifications has to do with their vagueness. Specifications are seldom, if ever, written so that they can simply be programmed directly, they are often riddled with ambiguities. This seems to occur in practice regardless of the detail of the specifications. Ideally, this should not be so, requirements phase properly carried out should result in detailed, unambiguous specifications. I wouldn't go as far as saying that unambiguous specifications are a metaphysical impossibility but they are most certainly a practical impossibility. Helen Ullman (1997) puts it very convincingly:

> *The "system" comes to [the programmers] done on paper, in English. "All" they have to do is write the code. But somewhere in that translation between the paper and the code, the clarity breaks down. [...] As the months of coding go on, the irregularities of human thinking start to emerge. You write some code, and suddenly there are dark, unspecified ideas. All the pages of careful documents, and still, between the sentences, something is missing. Human thinking can skip over a great deal, leap over small misunderstandings, can contain ifs and buts in untroubled corners of the mind. But the machine has*

> *no corners. [...]Now starts a process of frustration. The programmer goes back to the analysts with questions, the analysts to the users, the users to their managers, the managers back to the analysts, the analysts to the programmers. It turns out that some things are just not understood. No one knows the answers to some questions. Or worse, there are too many answers. A long list of exceptional situations is revealed, things that occur very rarely but that occur all the same. Should these be programmed? Yes, of course. How else will the system do the work human beings need to accomplish? Details and exceptions accumulate [...] (:21)*

One of the effects of being forced to produce an accurate estimate is that the specifications must be studied with extra intensity: vague descriptions must be concretized and ambiguities identified and solved. The specifications will seldom be as profoundly studied as when they must be implemented in code, but an estimate based on them will require they be analyzed attentively. From our interviews we have gathered that this study is the process that leads to a Work Breakdown Structure, which is then used to estimate the effort of the project and further to write a plan for it. This plan is based on the sequence of the tasks (which ones must be ordered sequentially and which ones can be carried out in parallel) and the estimated length of each task. All these three activities (outlining of a WBS, late estimation and planning) are closely related and it may be misleading to think of them in different terms.

In sum, being forced to carry out a late estimate will have as a result an interested analysis of the specifications and a more detailed action plan. These two activities should of course be carried out regardless of the obligation of producing a low-level estimate, but such a requirement will make them even more likely to happen.

Delegation and motivation: internal negotiations and involvement

In a such a knowledge intensive industry as software development, there is nothing as important as assuring loyalty and motivation among the people who develop the system (architects, designers, programmers, testers, etc.). The management literature on how to enhance motivation and create loyalty would probably fill whole libraries and there is little point in going through it here. Nevertheless, it seems plausible to suggest the following two ideas:

1) An estimation process based on a calculation carried out centrally (by higher management or external estimation experts, for instance) and whose results are later distributed and forced upon the developing team will not enhance motivation and allegiance to the established deadlines. It doesn't need to, and ideally it won't, disturb motivation, but there is a risk. Particularly if the estimate is an aggressive one. It goes without saying that the question of the development team's relationship to aggressive estimates made by external (to the project) actors is very complex and that factors such as the corporate culture or the charisma – or reputation – of the "estimator" will play an essential role in how those deadlines are experienced.

2) Distributed estimation, organized by a manager but carried out by the development team, stands, everything else equal, a better chance at enhancing motivation and loyalty. In this case, the estimation is in fact a process of negotiation among members of the development team and also between them and the project manager (in some cases, the project manager will team up with the developers and negotiate with some "higher" managers). Practices included in that process – such as meetings, arguments, discussions, formal and informal questions directly posed to the developers, etc. – offer the stage upon which negotiations are carried out and agreements are made. These agreements are in fact the development team's promises about when (and at what effort) tasks will be delivered. The essential aspect here is that developers themselves make the promises, creating an environment of self-discipline. The goals of the project have now been internalized by the developers and the following mechanisms are generally activated:

Shame and honor. Breaking one's promises is reason to feel ashamed, simply because we are brought up with the maxim that one should not break one's promises. However banal this observation is, the force of shame in organizational processes must not be underestimated. It would be very interesting to make a detailed study of the different reasons offered when a deadline is missed. It would probably tell us a lot about the way in which the broad maxim presented above has been adopted at the organization and, hence, about its corporate culture.

Professional pride. Related to the previous one but not the same. Missing a deadline that you have proposed yourself does not only mean you've broken your promise, it also means that you a) aren't a developer good enough to be able to make realistic estimates; and / or b) aren't a developer good enough to meet realistic deadlines. Developers, as many other professions, quickly establish meritocracies that order them in different classes. In the case of programmers this may be reinforced by the fact that the literature insists that star programmers are ten to twenty times more productive than normal programmers.

In one of our cases, we found that estimates made by developers were formalized into contracts. The answer to the question "How long do you think this will take?" was in fact "I hereby sign a contract between my project manager and myself where I guarantee that this task will be delivered by such and such a date." The effects that such a formalization may have on the developer's internalization of the project's goals is difficult to speculate about. It is clear, however, that a promise, which is something exchanged between humans, has now been turned into a contract, which is something exchanged between juridical entities. A simple theoretical analysis might say that the following alternatives are open:

1) the contract is just a waste of paper. Both developers and project managers know that it carries no practical force – i.e. it is not referred to in further discussions and it won't be used in case of delays – and that what is important is the agreement reached between them as colleagues and perhaps as friends. In such a situation the contract has no effect on the internalization.

2) the contract is referred to in case of a delay, but only as some sort of vague threat. There are no concrete economic or organizational consequences for the individual (such as withheld bonus or relegation). Will that vague threat enhance the developer's motivation? I guess it depends on the developer's personality, and I believe it must be used with care.

3) the contract includes clearly stated economic and / or organizational consequences for the individual. Both positive and negative (bonus if the deadline is met and withheld bonus is it is missed). Once the promise has been transformed into a purely formal contract, the personal mechanisms of shame and pride are likely to weaken. Whether the economic ambitions may counterbalance this loss will depend on the developer's personality. At any rate, economic promises are not excluded from a non-contractual system.

Since we have no empirical cases, these are all speculations, but I thought the discussion would be incomplete without them. As it would be incomplete if we do not mention the question of the role played by estimation "experts", those developers whose opinions weight much more than others. We have seen that in some companies there was a number of programmers whose estimates set the standard. No project estimate is concluded until they have had the chance to study the project and proposed an estimate. In fact, and this is the interesting aspect, it appears that estimates made by other developers may be changed if they do not coincide with theirs. Without wanting to speculate in this issue, it appears that motivation and loyalty to the new estimates should not be as high as to the estimates made by the developers themselves. Unless, of course, these star estimators are also star developers, occupy the highest ladder of the developers' hierarchy and are therefore greatly admired. In which case, the effect might actually be to enhance the motivation. But the exact effects will most likely vary from case to case and with time.

Establishment of sponsorship

One of the most important success factors listed by software project management literature in is that of assuring support from higher management, something also called "sponsorship." This factor does sound commonsensical but we are not in a position to neither deny it nor confirm it. In the course of our interviews we seldom came across this sort of ideas, but we have no good reasons to believe that "sponsorship" plays no role in the success of a project. It would of course be very interesting to study it in the future.

This said, however, it is interesting to notice that early estimates (high level) are often done by "higher management", in cases with the aid of project managers or even developers. One would expect that the "higher" manager that proposed this early estimate will have reason to feel involved with it, particularly considering the weight early estimates have. However, it is not clear whether this involvement will have positive or negative effects. Hopefully, the estimate is realistic and the project will benefit from the sponsorship, which may solve some – administrative, for instance – concerns faster. But the early estimate may also be too optimistic, and the sponsorship may turn into an intensified pressure to keep that impossible budget and deadline, which may have disastrous consequences – at this point I recommend the book Deathmarch (Yourdon 03) to those readers curious about severe project conditions.

References

Glass, R. L. (1997). Software Runaways: Monumental Software Disasters, Prentice Hall.

Ullman, E. (1997). Close to the Machine: Technophilia and Its Discontents, City Lights Publishers

Yourdon, E. (2003). Death March, Prentice Hall.

Pink Machine is the name of a research project currently carried out at the Department of Industrial Economics and Management at the Royal Institute of Technology, Stockholm. It aims to study the often forgotten non-serious driving forces of technical and economical development. We live indeed in the reality of the artificial, one in which technology has created, constructed and reshaped almost everything that surrounds us. If we look around us in the modern world, we see that it consists of things, of artefacts. Even the immaterial is formed and created by technology - driven by the imperative of the economic rationale.

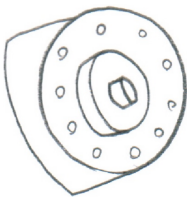
As Lev Vygotsky and Susanne Langer have pointed out, all things around us, all these technological wonders, have their first origin in someone's fantasies, dreams, hallucinations and visions. These things, which through their demand govern local and global economical processes, have little to do with what we usually regard as "basic human needs". It is rather so, it could be argued, that the economy at large is governed by human's unbounded thirst for jewellery, toys and entertainment. For some reason - the inherent urge of science for being taken seriously, maybe - these aspects have been recognised only in a very limited way within technological and economical research.

The seriousness of science is grey, Goethe said, whereas the colour of life glows green. We want to bring forward yet another colour, that of frivolity, and it is pink.

The Pink Machine Papers is our attempt to widen the perspective a bit, to give science a streak of pink. We would like to create a forum for half-finished scientific reports, of philosophical guesses and drafts. We want thus to conduct a dialogue which is based on current research and which gives us the opportunity to present our scientific ideas before we develop them into concluding and rigid - grey - reports and theses.

Finally: the name "Pink Machine" comes from an interview carried out in connection with heavy industrial constructions, where the buyer of a diesel power plant worth several hundred million dollars confessed that he would have preferred his machines to be pink.

Claes Gustafsson



also available at
www.pinkmachine.com
indek kth / 10044 sthlm / sweden