

10 Lessons in Guerilla Tactics of Project Management

A satirical approach to software project management by Matthias Gelbmann

While many brilliant books and articles have been written on good project management, there is little attention paid to all the possibilities a project manager has to create the appearance of managing a project well without the burden of actually doing it. The following article is a summary of such tactics, which I could not help noticing in many years of being involved in software projects. It is intended for project managers as well as people who monitor or supervise project managers, because we can prevent dirty tricks better if we have studied them.

1. Natural enemies of a project manager

As a project manager, you have three classes of natural enemies:

- the customer
- the project team
- the higher-level management

Luckily, only the third category really matters. The higher-level management decides on your salary, on your future assignments, on your career. Consequently, you need to direct all your attention to the higher-level management, or more precisely to the impression you make on the higher-level management. Doing good project management is one way of creating a good impression, but fortunately there are easier ways of achieving that goal.

2. The customer is a mean to an end

The customer is often referred to as *the* counterpart of the project manager. This is wrong. The customer is only a minor figure in the game. In principle, it doesn't matter whether he is satisfied or not, whether he is happy or not, or what opinions or feelings he has about the project manager. The only significance the customer has at all, is in his impact on the higher-level management, and, contrary to popular belief, the fact is that the impact is negative if he is treated too well. The higher-level management will think it's child's play to run a project with such an easy-going type of client, whereas it must be your goal to make them believe that only your heroic efforts could bring any amount of success dealing with such a trouble-seeking customer.

Don't make it look easy.

Having a satisfied customer is a dangerous thing

Satisfied customers belong to the mission statement, where they cannot do any harm. Avoid them in your projects at all cost. One has to be realistic. Who will be assigned to manage the next project: Project Manager A, who can manage trivial projects, but has never been challenged in difficult circumstances, or Project Manager B, who shows some degree of success even when confronted with extremely hostile customers?

Additionally, an overly satisfied customer will always generate the impression, that your loyalty to the customer is higher than your loyalty to your company. Allow this, and you are looking for a job.

3. Plan the planning

Serious mistakes can be made in the planning phase, as this is the stage to foresee future difficulties, and prepare for them accordingly.

Never get involved at the very early phases

Nothing can be gained by joining a project too early. Medals are being distributed at the end, not at the beginning. If you join later, you can always claim that things were messed up at the initial stage. No-one will remember exactly.

Plan most tasks to be finished just before the last milestone

Too many people can read Gantt charts. Task end dates are like mini-checkpoints and the very last thing you need are umpteen checkpoints on which anyone can put his finger. Combining most of these checkpoints with the unavoidable last milestone gives you the much needed degree of freedom to interpret the progress during the project.

Use person-years as estimation unit

It is in your interest to use the most ambiguous unit when estimating the effort, so that you can do some adjusted form of reverse calculations when needed. Avoid any detailed clarifications on what a person-year stands for, such as whether holidays or non-project overhead are included or not. If someone asks, say you used the standard definition, and therefore it's unnecessary to write it down.

Add contingency at all levels

Contingency is your little friend and helper. However, you must be aware that removing contingency is the only contribution of higher-level management to project planning. Therefore, it is essential to add plenty of contingency in many different forms, so that you will have enough after the removal. You have to ask the engineers to add contingency in their estimates, which later you refer to as "raw figures". Participate in all estimation meetings in order to talk the figures up. Then add contingency both in percentages and as extra activity at task, module and project level. Finally you add risk premiums at the various levels. The key skill here is creativity in naming, as you cannot expect any mercy from higher-level management when they recognize contingency as such.

4. Requirements are not required

Incomplete and ambiguous requirements are often said to be the most common obstacle to project success. That may or may not be true, in any case that's not your concern. Your concern is more the project manager's success. To this end, incomplete and ambiguous requirements are, generally speaking, quite helpful.

Never ask for completion of obviously missing requirements

Obviously missing requirements are the basis of your bonus. You can include them in your planning, and when the gaps in the documents surface later on, use them to justify all that extra effort that accumulates naturally during a project.

You have to be a bit more careful when it comes to ambiguous requirements. Skilful professional judgment is required to find the optimal point in time when to address them. Some can be used as a reason for early delays, as it takes time to clarify them, others can be helpful in drawing attention away from something else at a convenient moment, still others can be helpful in justifying poor implementation at a late stage.

5. Risks are too risky

The problem with risks is, that the project manager is expected to manage them. That would be hard. Therefore, if you see any risks in the project, make sure they don't appear as such in the project plan. Fortunately, there are more suitable places in typical project plan templates.

Hide risks as assumptions

It would be a mistake not to document those risks at all. You could be asked, "why haven't you taken care of that?" Assumptions are the perfect place for risks. No-one was ever asked to manage assumptions. Of course, a little rewording is required, by saying "it is assumed that such and such will not happen". If the thing goes wrong, you can always say that the project started under different preconditions, and that this was agreed with everyone beforehand. You don't need to be afraid that it will not be agreed. Nobody ever reads these boring parts of the project plan in a review.

Make sure to keep some easy-to-manage risks for the risk table, otherwise people could get suspicious. Furthermore, as we have seen above, you need risks to justify the risk premiums, and you need some easy risk-related activities to fill your progress report.

After the inevitable removal of the contingency discovered, as mentioned above, don't forget to add "best-case planning required by management" as top risk. You are safe in not adding any risk aversion action to this one. As a matter of fact, quoting that risk is only a small favor you can do for higher-level management, who like to be pictured as brave managers, determined to take risks if necessary.

6. Keep the reports clean

Status reports are a project managers appraisal sheets, and it comes in handy, that these are written by yourself. It would be a pity not to use that to its full advantage. Besides the obvious acclamation of any real or plausible sounding achievements, one has to follow just a few simple rules.

Never report delays as long as it makes sense to stop the project

This is one of the basic principles for successful project managers. Nobody stops a project that is 80% completed because of a 30% budget overrun. That's quite different in the first half of the project. Fortunately, this principle is dead easy to follow, as it just requires not doing something. Fill the time in the steering meetings with success stories and complaints about the customer. That should be sufficient to convince higher-level management that these meetings need to be short.

Never report delays as long as there is a chance that someone else has to

If you have a delay, wait until some external deliverable doesn't come in time. When that happens, your own delay to be enforced is calculated as the external delay plus your replanning time. The replanning time is never less than two weeks. If you need an external delay urgently, ask the customer to provide some clarification till the end of the week. Then say it's insufficient.

Multiply enforced delays by team members

There is a strikingly simple formula to calculate the budget overrun from an enforced delay. Just multiply it by the headcount in your project, no matter how many people are actually affected by the delayed deliverable. Even avoid that concept of "people affected" in your reports, and rely on the sheer beauty and simplicity of the argument.

In progress reports, only report actuals and planned totals

Project tracking is comparing actuals with plans. What you have to avoid, however, is comparing actuals with up-to-today totals, as this would mean comparing apples with apples, which in turn could lead to dangerous conclusions. You have to start this practice at the beginning, when it doesn't really matter yet, and later when people start to complain, say it's the way it was always done and it would be unwise to change reporting procedures in critical phases of a project.

Call it extreme project management

This is a Jolly Joker explanation. Whenever you fail to come up with a good enough reason for anything, tell them you are applying extreme project management. Never get involved in discussions about details here. If someone insists and asks what exactly that means, or wants to know why that would be appropriate, then just refer them to "the literature" and advise them to learn a little more about modern management methodologies before one can continue such a discussion.

7. Milestones are fixed

If milestones were supposed to be shifted around they would be called pebbles. Just like a week is a week, no matter what happens, a project phase has to end when the time is there, no matter what has been accomplished. The purpose of a milestone meeting is to celebrate the achievements of the passed project-phase with a glass of champagne or coffee and donuts, whatever your company's style allows.

Occasionally, people have the notion that the passing of a milestone needs to be "decided", as if the passing of a year needed to be decided. That rarely comes from higher-level management, as they love their champagne, but if it does, you have to be careful. In that case, you discuss whatever issues they bring up at length and then, while they pour the champagne, say you will document the decisions in the milestone report.

Report vague results of milestone meetings

No questions, the milestone is still passed, but you have to bring in the higher-level management objections in a constructive way. You do this by using some attributes, such as calling the milestone "conditionally passed". Don't list any conditions in that case, that could later be used against you. Another variant is calling it "provisionally passed", or something along those lines, just make sure you can refer to the milestone as "passed" in the next meeting.

8. Embrace re-use

Re-used modules are blame-tickets. Make sure to re-use plenty of code from previous projects, the more the better. Then track all the difficult bugs to some re-used code, even if the code was never intended to be used the way you use it. As the code comes from a team that doesn't exist anymore, it's as easy as winning a tennis match against a ghost. The good thing now is, that you can still prove that despite all these bugs, your decision to re-use was right, because writing all that tons of code from scratch would have been prohibitively expensive, so you saved an enormous amount of money.

No-one was ever accused of having too much re-use

Another brilliant aspect of code re-use is, that a fair amount of extra costs can be justified by claiming that you make your code re-usable. Ideally, of course, your code will never actually be re-used, but even if it were, you wouldn't be there anymore. It also wouldn't make a difference to your bonus, because no higher-level manager has ever been known to distinguish between re-usable code and re-used code.

9. Your project team is not your electorate

Never mention the project team in communication with higher-level management. Talk about you, or at most talk about "the project". The team is just a tool to accomplish some work, and you don't talk a lot about your hammer, do you?

Don't be popular within the project team

On the other hand, make sure the team talks about you whenever they happen to communicate with higher-level management. Furthermore, make sure they don't spread the impression that they like working for you. A project manager is not being paid for making the project team happy. If they are, it is always seen as an indication that you are unable to squeeze the last bit of productivity out of them, which is one of the worst things that could happen to your reputation.

10. All good things must come to an end

Project end dates could be treated like other milestones, if it weren't for the peculiarity of product shipment. The principle of immovability still holds, but creativity may be needed for the delivery part. If the product isn't finished, you can still ship it, as long as you make sure that it's also untestable. There are a number of techniques to achieve this, which all have in common that they exploit properties of the test environment. You can make the product uninstallable, or you can make use of a limitation of the operating system, the platform or the database that is used, so that it won't run. These considerations cannot, of course, be left to the last minute. It must be a top priority task of your best engineers to identify these shipment variants before they build the product, and it must be a guiding design principle from the very beginning.

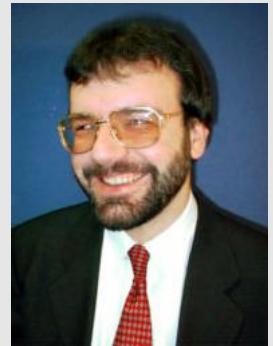
Declare your uncompleted project as finished

Whatever the outcome of the attempts to test the product, after the final shipment, there must be no such thing as an open task. That's easy to achieve, because, after all, it's the project manager who decides when a task can be considered finished. The only things that are left after you close all tasks, are the so-called "follow-up activities". These are, however, much better taken care of outside the project, for instance as "maintenance" under the supervision of line management, or within the next project, as long as it isn't yours.

It is rarely necessary to point out to higher-level management how wise it is to close the project as long as it is still possible to call it a huge success. They wouldn't be higher level if they wouldn't know the advantages of having successful projects within their area of responsibility. Keeping those simple tactics in mind really creates a win-win situation for everyone.

Matthias Gelbmann has over 20 years of experience in software development. He worked in different technical and managerial positions in diverse areas ranging from artificial intelligence research to commercial database applications and embedded software. He is now Chief Executive and consultant for software quality management at Q-Success.

[Q-Success](#) is an Austrian consulting company, specialized in CMMI-based software process improvement, software quality assurance and software project management.



Republishing this article

We grant the permission to electronically republish this article in pdf format on the internet without any modifications and with intact hyperlinks to the Q-Success website.