Leading Software Maniacs™

Seven Deadly Habits of Ineffective Software Managers

Date:	Dec 30, 2010	
Title:	Seven Deadly Habits of Ineffective Software Managers	
Code:	W7DH	
Abstract:	These stories should introduce you to the trials and tribulations of software development management. This definitely applies to project management. Happy reading (and don't get too depressed)!	
Version:	1.0 (2010-12-30)	Original (excerpt from Leading Software Maniac's *Seven Deadly Habits of Ineffective Software Managers* keynote)

Habit 1: Releasing a Product Before It is Ready

There's *always* the pressure to deliver a product to market before it is ready—especially if the software engineers on the team feel like it is "nearly done" and the testers aren't sure it will "ever be done." Regardless of executive pressure or team denial, listen to your testers on the team. They tend to havbe the right impression if a project works or not. On one occasion, a VP of Marketing tried to convince me and the executive team that "shipping early shows attention to the customer and first to market. You can always demonstrate customer responsiveness by how fast your fix things. No one really expects software to be bug free anyway."

My experience has been quite the opposite. You usually have one chance to delight a customer and in the case of a product release that becomes a quality disaster, you've probably lost that customer for life. Also, there's the old adage that "good news travels, but bad news travels faster." You don't want that! Your company's reputation needs to be maintained.

Habit 2: Hiring Someone Who is Not Quite Qualified (but Who Everyone Likes)

As if the hiring process isn't difficult enough (advertising, reviewing résumés, phone screening, interviewing, and so on), we've all been in the situation where a candidate becomes "good enough" to hire. Not quite what you were looking for, but close enough—in fact, you talk yourself into believing that they'll somehow learn to do the job. It can get a little dicey if you hear the phrase about the candidate like "he doesn't quite have the technical requirements but he'll fit perfectly with the culture of the team and the department."

Don’t hire that individual! (And we’ve all made that mistake.)

Regardless how well the individual “fits it,” it comes down to one and only one thing: can they perform the work that is required? If after 90 days you have to let the new hire go, you’ve not only set your team back, but you’ve also made it very difficult for the fired person to find another job (especially in this tough job economy).

Habit 3: Making Every Decision a Consensus Decision

Making decisions with a “disruptive yet talented” team can be a tough leadership challenge. It isn’t unusual for a newly appointed manager to attempt to get 100% consensus with every decision. (With software developers, that may actually be close to impossible!) Taking into account all of the varied sources of information required to prioritize (customer input, technical feasibility, market demands, things that need fixed, and so on) getting the team to unanimous euphoria may not be practical.

So, what do you do?

Before collecting all of the information and before making a key decision, you could ensure that the decision criteria is understood. When selecting the features to work on for your next project, you might come to a different conclusion if customer usage is more important than “marketing splash.” In addition, how do you convince a software developer that what features they want to put in really has no customer benefit? What’s worse is if the criteria used in decision making *changes* based on the mood (or pressure) of the day.

The result? Frustrating, long meetings or even team frustration with you as a leader.

Ultimately, a wise thing is to get everyone’s feedback but somebody (probably you along with the product owner) makes the final decision. You don’t need consensus, but you do need a decision based on consistent decision criteria. In other words, what are your organization’s decision making priorities? Try to define them in your next cross-functional team meeting! You might be surprised how in synch everyone is (or how everyone’s decision priorities are completely disjointed).

Regardless, you need decision making unity!

Habit 4: Promising Developers Incentives

The software industry is littered with teams that have been asked to go “above and beyond,” with all sorts of incentives (carrots on a stick) to entice superhuman success. We’ve all seen the impact to the team when only a select few are given bonuses (usually the software engineers).

There’s a famous story some years back of a database company (no longer in business, by the way) that was so frustrated with missed schedules, they pronounced that they were giving the teams unprecedented bonuses *if* the next major release was delivered on time. The idea was that money should be a motivator to release *anything* on schedule. Shoot, it works for sales people! This was even publicized, if you can believe it.

You already know the outcome …

The teams worked non-stop and couldn’t meet the schedule. So what did management do? Since everybody tried so hard, perhaps that schedule was too aggressive. Another schedule (with bonuses) was pronounced. With months consisting of the team again working long hours, that schedule was missed.

As a result (and with considerable "egg on their faces") management withdrew all bonuses and made sure everybody knew that the company needed the release out and jobs were on the line if they didn't deliver on yet a third schedule. Well, they eventually *did* deliver the project (later than expected again) and team morale became so bad that a mass exodus took place. The best software engineers left out of disgust—management was definitely not trusted.

Why *incent* when you can privately *award* (after a project or milestone success)? You don't need big things, but those timely, unexpected bonuses (or benefits) are much more effective. Awards aren't always necessary, but they sure can lift morale. And, just don't focus these benefits on the software engineers! Other members of the team (especially QA) deserve recognition, too.

Habit 5: Delegating Absolute Control to a Project Manager

Hopefully this won't frustrate every project manager that reads this …

There's much dialog on the Web among LinkedIn project management and agile (Scrum) groups that ask the simple question, "just now technical does a project manager *need* to be?"

Project leadership is an amazing skill and you don't have to be technical to build upon the ability to communicate, prioritize, identify risks, report project status, and so forth. However, delegating 100% absolute control to a project manager (or ScrumMaster, for that matter) who doesn't really understand the technical details that can cause a project to derail is a mistake. It is that famous "just one more question" that if a project manager doesn't know to ask could result in project failure.

Bottom line?

If you, as a project manager or ScrumMaster, aren't technical enough to understand the details of the project you're leading, align yourself with somebody on the team who is. Together you can keep the project on schedule and technically sound. (There's no wonder that in at least a couple of software companies in the Seattle area that there best project managers are technical experts.)

And of course, if you are technical and a great project leader, everybody wins!

Habit 6: Taking Too Long to Negotiate Feature Sets and Schedules

Not preparing for a project's completion can lead to disastrous results. Lack of preparation, according to Tuckman's excellent model of team development, can quickly become the "mourning" stage at the completion of the project. This is when the team mills around aimlessly as management decides what to do next.

Waiting to decide a project's next set of features can result in the entire organization going through a horrendous set of never-ending volleys of "document passing" between product management and engineering.

This can be a total waste of time!

Why not get the engineering representatives to work closely with product management to decide on the next set of features and schedules as a unified team?. This not only reduces any hostility that may have developed over time, but this approach definitely gets the buy-in that is so critical when starting a new project.

Another suggestion? Start feature set and schedule definition work as soon as the prior project reaches the “technical feasibility” milestone (that’s when basic feature set is in place). That way, there isn’t the downtime after the current project completes and the next project begins. It can also help motivate the team of what’s coming next after project completion.

Habit 7: Ignoring a Process in Order to Release Quickly

“I’ve got a friend who is the engineering VP of another software company. They don’t need any process and they get their software products out on time all of the time!” A VP of Sales told our management team this years ago. (I still jolt out of bed in a cold sweat hearing that proclamation!)

I guess the grass is always greener …

The idea of following a process can be foreign to most everybody outside of development. It appears bureaucratic, non-intuitive, and slow moving. Well, look at *your* process. Is yours like that, too?

If it is, change it! We’ve all worked with 400+ page SDLC (software development life cycle) specifications that defines every handoff in a waterfall-style culture that is so complicated nobody bothers to follow it. What’s worse is if one team follows a waterfall style of project development and is dependent on a Scrum-led subteam. Whew – talk about “oil and water.” (Don’t laugh, I’ve seen it happen!)

You may be able to get a project release without process *once*, but your luck will run out especially as the technology gets more complicated, feature requests expand, and your team grows. I’ll never forget wise words that I heard from a marketing consultant from Austin, Texas:

> “A company *with* a process will always out-perform better that one *without* a process.”

If your process is too complicated, simplify it. If your process has components that don’t benefit anybody, nuke ‘em all. A perfect example is detailed status reports that, in reality, nobody reads or cares about. All most stakeholders want to know is if the project is on track.

You get the idea …

A Final Word

Through personal experience, these seven habits should be avoided at all costs. In the next article, I’ll present ten habits you’ll want to adopt.

Bibliography

Whitaker, Ken. *Principles of Software Development Leadership: Applying Project Management Principles to Agile Software Development*. Boston: Course Technology PTR, 2009.

Bio

Ken Whitaker of Leading Software Maniacs™ (LSM) has more than twenty-five years of software development executive leadership and training experience in a variety of technology roles and industries. He has led commercial software teams at Software Publishing (remember Harvard Graphics?), Data General, embedded systems software companies, and enterprise software suppliers. Ken is an active PMI® member, Project Management Professional (PMP)® certified, a Lewis Institute instructor, and a Certified ScrumMaster (CSM). Sources for LSM's presentations come from case studies, personal leadership experience, the PMI *Project Management Book of Knowledge* (*PMBOK® Guide*), and Ken's two books: *Managing Software Maniacs* and *Principles of Software Development Leadership*.

Leading Software Maniacs is proudly associated with: