Why Agile Won't Fix All Your Problems

John Levy, Ph.D.

Welcome to the webinar Why Agile Won't Fix All Your Problems. I'm John Levy and I'll be guiding you through the next 45 minutes or so.

I hope you're ready to respond to the questions I'll put out to you as we go along.

Just to review how you can ask questions during the webinar … look for …….

OK? Let's get started.

In this webinar, we will cover

- What Agile is good for
- How Agile will make certain problems *worse*
- Problems that *won't* be solved by Agile
- Key success factors for Agile

Today we're going to talk about Agile software development. You'll find out that I actually like Agile, and I tend to promote it. But Agile has become popular, and a lot of people are disappointed with their lack of results. By the end of this webinar, my intention is to make it clear why those disappointments are happening, and what you can do to prevent them.

There are problems that are not solved by Agile, and in fact there are situations that could be made worse by adopting an Agile process.

We'll end with some of the key success factors for Agile, so you can focus on avoiding the disappointment.

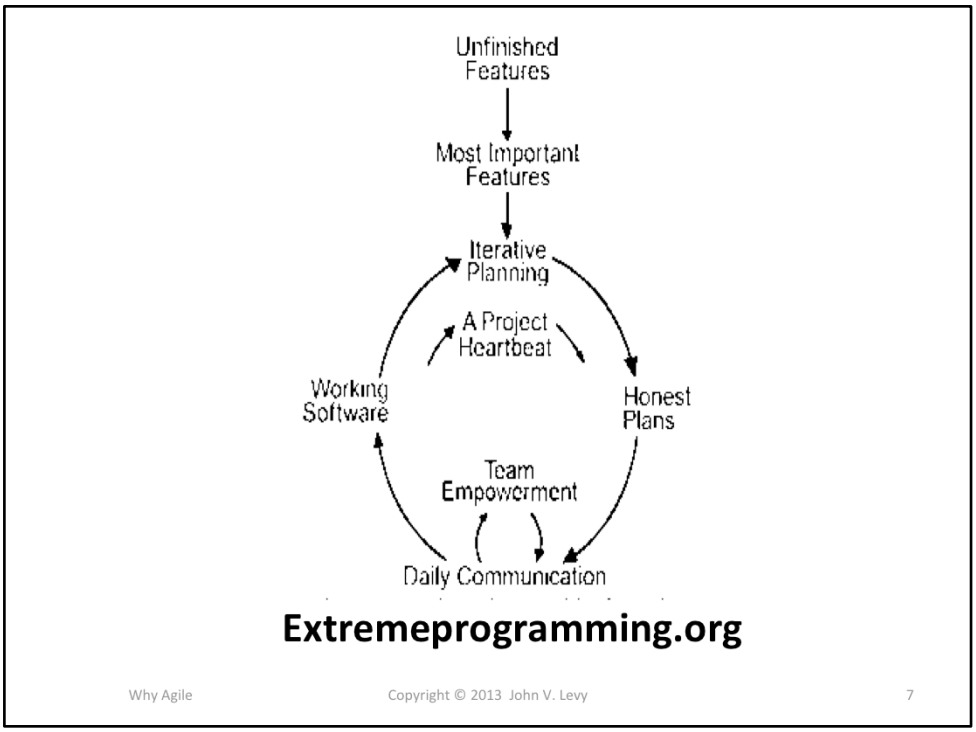
What I like about Agile

- It's not a theory – it's practice
- It's from grassroots

I do like Agile methods, and I am an advocate of adopting Agile – in the right circumstances.

One of the things I like about Agile is that it came from people who actually do a lot of software development – not from academics, not from theorists. Not only do the Agile principles embody practical experience, the whole flavor of Agile is to adapt the process to fit the particular situation of the development team and its environment. That's why there are retrospectives, and that's why experimentation is encouraged.

The other thing I like about Agile is that it came up from those practitioners – the grassroots of software development – and was negotiated by a group of practitioners working together. Furthermore, there is a lively community of Agile practitioners who continue to refine the practice of Agile.



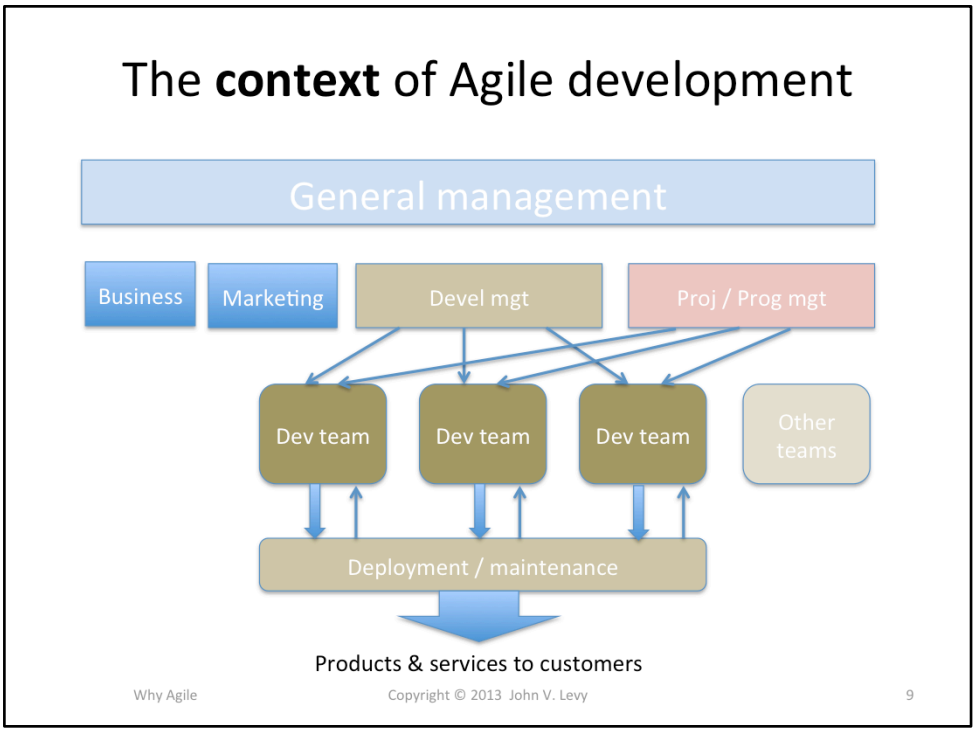
For example, you may want to have a look at extremeprogramming.org, which is where this diagram came from. If this kind of diagram of an Agile process is not already familiar to you, I suggest you look up some of the references on Agile, Scrum or XP (extreme programming).

The Essence of Agile

- Iterative, frequent demonstrations
- Self-managed team, adjustable process

What is the essence of an Agile process?
I think that these two features are the core of any process that wants to be called Agile.
The first is that development is iterative – using a cycle of development that has a fixed, pre-defined duration. Furthermore, that duration is short relative to the overall development timeline. For example, one week, two weeks or three weeks. Not a choice between these three, but just one of them, repeated over and over. At the end of each iteration, working software is demonstrated. Every feature or function that is demonstrated – working as expected – is considered "done" at the end of the iteration. Every function that is not completed is considered "not done". Furthermore, selecting which features or functions are to be added to the system in the next iteration is done after the completion of the previous iteration. This allows re-prioritization of the features to happen often – once per iteration.

The second essential feature is that the development team is self-managed. This means that the team should be allowed to choose it members, including voting someone off the island (as they say in reality TV) when they aren't working well as a member of the team. It also means that the team adjusts its own process by trying out different ways of working. For example, the team may do what is called pair-programming, where two people work side-by-side on a single problem.
This kind of flexibility requires commitment by the management to let the team alone, to allow it to find and use a process that works for them.



Here's a diagram that summarizes the working environment of a development team. Agile generally addresses how a single development team works. But each team works in the context of other people and processes, so we have to pay attention to the context.

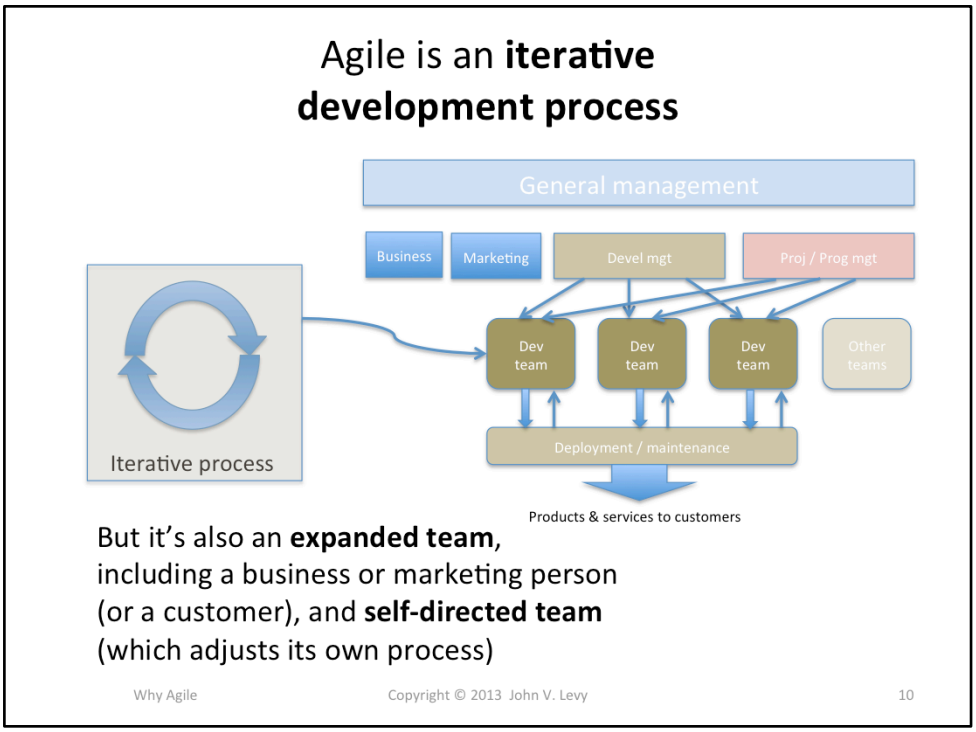
For example, even though the box marked "Dev Team" seems to be entirely under the "Development Management" shown here, in an Agile team, there is at least one representative from either Marketing or Business management (or Product Marketing) present in the team. There may also be an Agile coach.

Next, notice that there are other teams out there that a given Dev team may have to interact with – and even be dependent on.

Notice that the organization may have a separate Project or Program Management function that is embedded in the company and monitors – and even controls – the development teams.

And don't forget the General Management shown at the top. Often, upper management people get involved in development projects, too. This can be an advantage, such as when you want your project to have more visibility and therefore more clout in getting funding; and it can be a disadvantage, such as when a top-level manager tries to direct the details of the project.

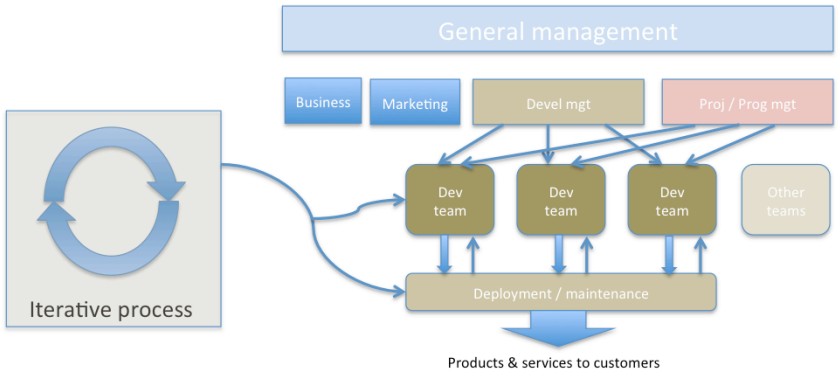
Deployment and maintenance, shown at the bottom, is not only responsible for getting the "finished" product into a form that can be delivered (or served on a server), it should also be providing feedback to the Dev teams about the product's performance.



As we saw in the XP diagram and discussed before, Agile is above all an iterative process – a process that repeats on a short cycle of a couple of weeks or so.

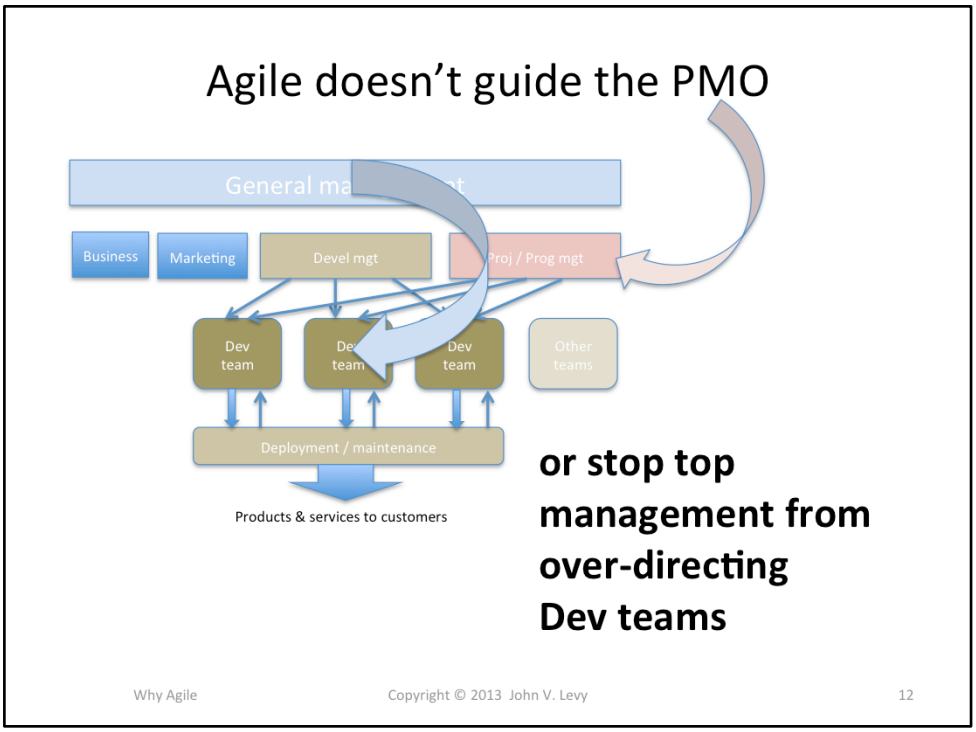
This cycle applies to the dev team, but we should be aware that the team is not just the technical developers – it includes at least one "customer" or a substitute for the customer who comes from a business group or product marketing. The availability of such a "customer" on a daily basis makes a big difference for the development team, because it means that face-to-face discussions can be had every day to clarify the meaning of features and functions. In spite of the informal nature of these discussions, they are very important in assuring that each iteration produces something of value to the customer.

Agile can also be a *Devops* process



with iteration including both development and operations
(deployment / maintenance)

The iterative process can be carried outside of the core dev team and into QA and Deployment or Maintenance as well. Since the Deployment function is usually considered to be part of Operations, creating a cross functional team between development and operations is called DevOps. This can speed up the process of getting developed software into production – and reduce the number of problems seen after release.



One of the problems with adopting an Agile process is that your project managers – particularly if they are centralized in a project management office (PMO) – won't be able to apply their usual measures and controls to the dev team.

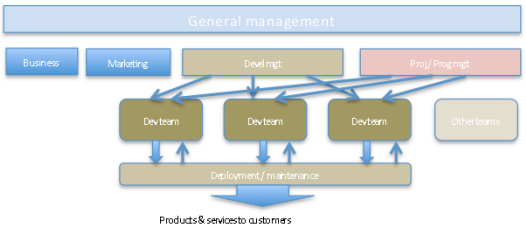
The PMO usually has control of budgets & expenditures and may also control the "gates" of a project that determine whether development will be continued.

The PMO typically reports status upward to management and outward to other groups, but mostly these reports are against budgetary goals, not completed feature of the software.

Also, if you have upper management that tends to get involved in the day-to-day decisions made in development, using an Agile process won't stop them.

What Agile is good for (1)

- Overcoming invisibility of progress
- Dealing with incomplete requirements



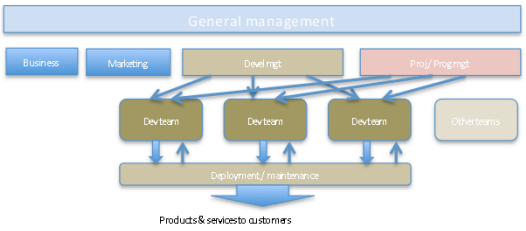
So let's look at things that Agile processes are good for.

First of all, Agile tends to overcome the fault of many projects in which they show "no progress", or if they do have progress, no one can see it – which is what I mean by invisibility.

Second, Agile tends to help deal with requirements that aren't finished or aren't clear, because the developers are regularly asking questions about what the requirements (or "stories", in Agile speak) mean. When these questions point out incompleteness of the story, there is someone representing the customer who can help clarify it with very short notice.

What Agile is good for (2)

- Driving automation of test & integration
- Creating team internal alignment

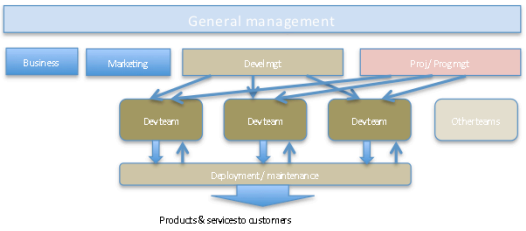


Third, Agile processes tend to drive automation of testing and integration, because the short iteration duration makes the dev team need those tools to keep up the pace; so the team tends to select useful tools and become proficient at using them.

Fourth, Agile drives (internal) team alignment & collaboration, because the team is focused on a few key activities with well-defined outputs – which is the working software. Also, Agile teams usually start out with an Agile process coach, which helps the team adapt their process and learn what works for them.

What Agile is good for (3)

- Bringing Marketing/Business into the process
- Increasing trust of managers in a dev team

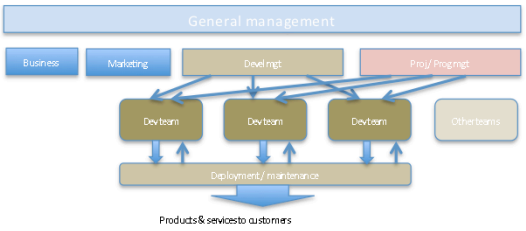


Fifth, Agile makes the development department recruit Business or Marketing into the development process, by placing a designated business person on the dev team.

And sixth, Agile can help overcome lack of trust between middle management and a development team, because Agile processes tend to show visible results much earlier than "waterfall" processes. Of course, the middle managers have to accept launching Agile methods at the outset or it won't work.

What Agile will make worse (1)

- Team / management misalignment
- Lack of sponsorship



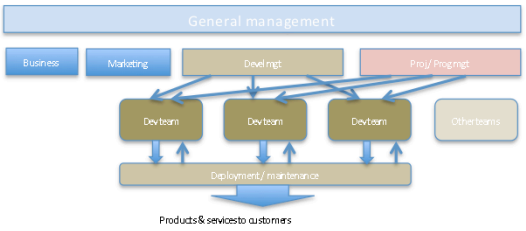
OK, let's move on to things that Agile can make worse.

First of all, if the dev team and its management are not aligned on the goals for the project – or any other significant part of the team's charter – then Agile will not make a difference. Usually such a misalignment comes from having a fair amount of power in the team to select technologies or features, and the management may have different ideas about what to build and when to build it. Moving to an Agile process won't help.

Second, transitioning to Agile requires support from upper management, because of the disruption it can cause at several levels. Lack of sponsorship is the quickest way to insure that the Agile transition won't succeed.

What Agile will make worse (2)

- Out-of-control marketing demands
- Micro-management by senior executives
- A culture of "do or die" and "ship it now"

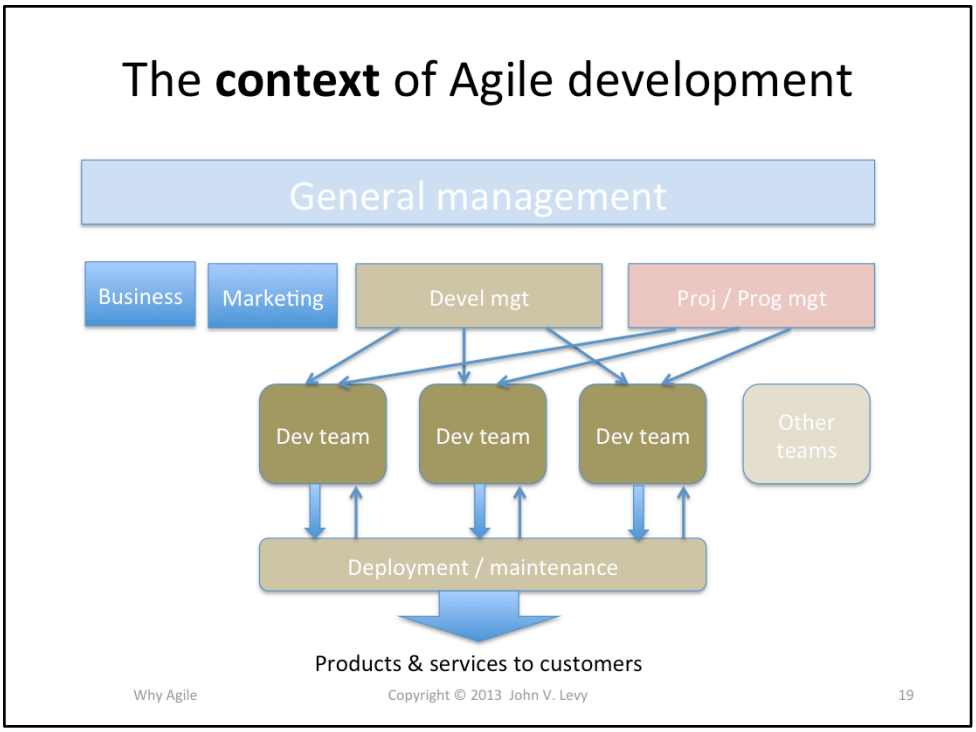


Why Agile Copyright © 2013 John V. Levy 18

Third, if Marketing demands everything & wants it all at once, Agile won't help you. It will only make the problems worse, because if marketing has the power to demand anything it wants, then the dev teams will not be able to isolate themselves enough to carry on the Agile processes.

Fourth is meddling by senior executives in technical decisions. We all know that Steve Jobs was very involved in the development of Apple products. But he had an outstanding record of sensing or knowing what would succeed. If you have a top executive who makes technical and product detail decisions, your odds of having a successful product are not as high as Apple's, because most executives don't have Jobs' level of intuition and understanding. Going Agile won't protect you.

Fifth, you may have executives who insist that dev teams should work 80 to 90 hours per week all of the time. They believe in on-time completion "at any cost" and they demand that you "ship it now" even when you know the product isn't ready. Agile won't help you here, either.



OK, those are some of the situations made worse by Agile. There are also problems that may not get worse with Agile, but also won't be solved by Agile.

Remember that we're dealing with an entire environment for development. When you start using an Agile process, you change the way the dev team interacts with Business, Marketing, Operations and other teams. It takes a lot of management support and understanding to make an Agile transition succeed.

Problems that Agile won't solve (1)

- Over-commitment by Marketing
- Star system
- Insufficient technical capability in the team
- Excessive project complexity

OK, so I'm going to list some of the the problems that won't be fixed by Agile. These are selected from a very long list, so they are just a representative sample.

If Marketing consistently tells customers that development can deliver things that are just barely possible, Agile won't stop them.

If your management system rewards prima donnas (what I call the "star" system), Agile will tend to fail because the stars don't want to be just part of a team.

If your organization fails to attract and retain competent & capable technical people, Agile won't fix that, either. You need good technologists, no matter what processes you use.

If the project has taken on too many constraints or has too many interfaces, it will fail due to the complexity, even with Agile.

Problems that Agile won't solve (2)

- Conflict over goals & requirements / 2 agendas
- Over-control by PMO
- Excessively dispersed team
- Delegation of quality to a remote QA team
- Multiple agendas in management

If you management is waffling on the goals & objectives & requirements of the project, or if there is a conflict between Marketing and Engineering (for example) on what's to be built, Agile won't help.

The PMO can undermine Agile projects by insisting on control by Project Managers who are not part of the development team

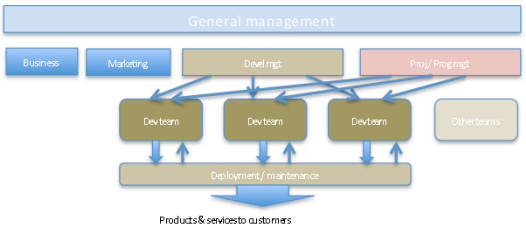
Using a dev team that is spread around the globe may be economically efficient, but it makes Agile processes harder to use.

Furthermore, if the QA or testing is remote from the people writing the code, this tends to cause overload on the QA people, and finger-pointing with regard to bugs. Agile won't help.

Finally, if your Management has multiple agendas, only one of which is to get the product out, Agile will not overcome it. This kind of management doesn't want to hear the truth about what it will take to develop a product anyway.

Key success factors for Agile (1)

- Executive sponsorship
- Delegation of real authority to the team
- Real customer or a very good surrogate



Why Agile Copyright © 2013 John V. Levy 24

In my experience, here are six things that will help you succeed with Agile.

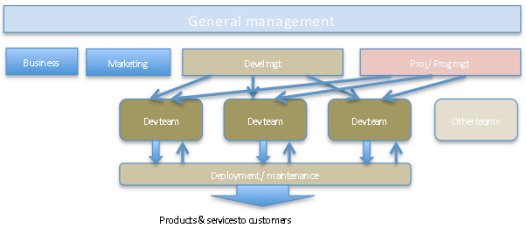
First, sponsorship enables the team to behave differently from previous non-Agile teams and also tends to drive a focus on results rather than process. This is good, because Agile is all about delivering visible & usable results.

Second, real authority means that the team can adjust its own process, select **or de-select** team members, set their working hours, and buy the tools they need.

Third, a real customer is someone who has experience in the domain of the intended product or service, and knows when the problem is actually solved well.

Key success factors for Agile (2)

- Commitment to quality
- Budget for automation tools
- Agile process coach during transition



Fourth, without a commitment to quality, the resulting products will satisfy neither the customer nor the developers. In addition, cutting corners on quality leaves what is called "technical debt" which must be paid later. This will lower the effectiveness of the maintenance team (if it is different from the development team) and create a hole that will be hard to climb out of.

Fifth, automation tools are essential to effective Agile development (and other development), so the team should not have to jump through a lot of hoops to get the money for proper tools.

Sixth and finally, if Agile is new to the organization, an Agile process coach will help the team show results sooner and get comfortable with the new process faster.

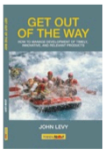
If you need access to an Agile coach, feel free to contact me. I can refer you to some excellent coaches and other specialists. And if you need help designing a transition to Agile, or need correction to the transition you're already in, I would be glad to talk with you about what it will take.

References

White paper: **9 Mistakes that Lead to IT Project Failure**

http://johnlevyconsulting.com

Book: **Get Out of the Way**



http://johnlevyconsulting.com/john-levy-book/

Agile Management **blog**

http://johnlevyconsulting.com/blog/

Websites & groups

http://Extremeprogramming.org
http://Xprogramming.com
http://tech.groups.yahoo.com/group/extremeprogramming/
http://www.scrumalliance.org
http://www.agilealliance.org
http://www.agilemanifesto.org/principles.html

Here are some references you may want to take advantage of.

I'm offering a free white paper titled "9 Mistakes that lead to IT project failure". If you go to my website, you'll find the signup for this paper right on the home page. This paper is aimed at executives, but you may learn from it at any level of project management.

When you request the white paper, you'll also be subscribed to my every-2-weeks articles, which I call "Agile Management". You can of course unsubscribe at any time.

And if you're interested in my book on managing high-tech teams and people, have a look at "Get Out of the Way". It's primarily aimed at technologists who have become managers, but there are tips in there that you will enjoy reading related to project management, including a lot of stories about projects I've been involved in over the years.

I've also listed a number of websites and one forum group you may want to have a look at.