

Coaches, Managers, Collaboration and Agile, Part 1

Monday, November 07, 2016 3:52 PM

Part 1

by [johanna](#) | Oct 11, 2016 | [agile](#), [MPD](#) | [0 comments](#)

There was a fascinating Twitter conversation last week when I was busy writing other things. (I also find Twitter to be a difficult-for-me arena to have a conversation. I need more than 140 characters.)

The conversation started when [Neil Killick](#) tweeted this:

orgs need coaches not because "agile is unintuitive", but because effective sw delivery is a team game requiring folks at their best.

I liked that and retweeted it.

[Mohinder Khosla](#) asked (quite reasonably, in my opinion):

Do we have stats that this is real? Just saying

Mohinder, I am sure we do not have stats. We rarely do what I would categorize as real research in how software product development works. That's because we continually learn and even if we start a new project with exactly the same people, we have learned to work together in previous projects. If we don't keep teams together, we have to relearn how to work together. (See Hackman's [Leading Teams](#) book for this and much more insight on teams.)

Agile challenges all of our previous assumptions about how software product development works.

- We think in product, not just project. See terms as releasing, product owner, and MVP.
- We think in [flow efficiency](#), not resource efficiency.
- We integrate all of the work we might have done as phases into small stories that we release often.
- We invite change, we don't try to control it
- and much more...

Agile is a [mindset change](#) and a cultural change. (See [When is Agile Wrong for You?](#))

One of the big changes in agile is that we work as a team. We work as a team to move work across the board. We don't hand off work from developers to testers. We don't hand off work from architects to developers. We don't hand off work from testers to operations.

Can each team do that now? Probably not the first time they try to use agile. This notion of the cross-functional team being the team in question challenges everything we, our managers, and our organizations "know" about product development.

Coaches can help teams and managers learn what it's like to create the agile mindset and an agile culture. Do you *need* a coach? Maybe not. Here are some questions I have asked my clients when they asked whether a coach would be useful for them:

- Do you have a cross-functional team? Or, is it a team of developers, followed by a team of testers, etc.
- Can you create features as one cross-functional team, by yourselves? (You might not be doing this now, but can you? Or, do you need other people across the organization?)
- Does the team demonstrate on a regular basis? (My guideline: at least once every two weeks.)
- Does the team retrospect on a regular basis? (My guideline: at least once every two weeks.)

Notice that these questions are about team collaboration and delivery.

Agile creates transparency. We can see when a team is not "performing." I put that in quotes because often the *environment* is what causes teams to not "perform."

Some teams have trouble working as a team. Coaches can help. Other teams cannot answer yes to my four questions. If so, coaches can help the team and the managers. Can the organization help itself? Of course. Would it be easier with a coach? Probably.

You don't need agile coaches to be successful. You can do it yourself. And, if you look at people and teams who fit your definition of success, they will often tell you they have coaches. (I do.)

Agile teams deliver working product on a regular, frequent cadence. They can use flow-based agile or iteration-based agile to do so. It doesn't matter.

When teams deliver on a regular basis, they work as a collaborative team. That idea and practice might be quite new to everyone in the organization. Coaches might help.

The next post will be the role of managers in coaching in an agile environment.

Inserted from <http://www.irothman.com/mpd/agile/2016/10/coaches-managers-collaboration-and-agile-part-1/>

Part 2

by [johanna](#) | Oct 13, 2016 | [agile](#), [MPD](#) | [0 comments](#)

In [Coaches, Managers, Collaboration and Agile, Part 1](#), I wrote about circumstances under which a team might want a coach. It wasn't an exhaustive list. It had several questions defining when coaches might help the team to become agile, not be cargo cult agile.

One of the reasons we might need coaches for a team is because of the changed manager role in agile.

In functional organizations, managers coached the team members on how to do their jobs better. (Okay, some managers did.) Development managers coached developers. Test managers coached testers.

Project manager-managers coached project managers, etc. Each functional manager coached the people in their functions on how to do their functions better.

Some managers learned how to coach the functional people to perform better in teams. To be honest, that was often quite difficult. The organization rewarded the function, not the teamwork.

Agile is changing this mindset of "coach the function, let the teamwork take care of itself." (Yes, that's a

generalization, and not fair. However, I see a lot of it.)

In agile, we want a product as a result of team effort. That leads us to think about the role of managers and teams and coaching in different ways. (Well, it does for me.)

If you are not aware of Human Systems Dynamics, let me introduce you to the ideas of [Containers, Differences, and Exchanges](#). I read about this in [Adaptive Action: Leveraging Uncertainty in Your Organization](#).

When people and projects choose agile, the managers have new and different roles. People no longer affiliate with managers (the container part). They affiliate with projects. This means that manager-as-coach changes in any number of ways.

Instead of being able to, or even having the time, to coach people on their functional roles, the manager might create communities of practice, or other ways of encouraging people to learn about their role better. If the manager knows about the function, the manager can coach the people.

And, many managers only know about or have experience in their own functions, such as strictly development or testing or writing. I call these managers “ignorant.” That’s because unless a manager can see the entire picture (including the dynamics of a project, the manager might not realize the problem with the impediments the team faces.

Ignorant managers cannot create the environment in which a team can deliver working product. Yes, the team has responsibility for a significant part of that environment, especially working as a team to move features across the board. And, the manager creates the container in which the team works.

I’ve seen ignorant managers create these problems for the team:

- They don’t realize the effect of [multitasking](#) on the people or the team
- They don’t realize that a team being able to go fast is a result of [technical excellence and getting to done on small features](#)
- They want [guarantees](#) (often in the form of estimates) instead of seeing the [learning](#) that teams require

You might see other problems in your organization. An ignorant manager cannot help the team or the team members at all, because the ignorant manager does not realize the problems these impediments cause.

The agile manager’s role changes, especially managers who “manage” technical teams. I wrote about this years ago in [Agile Managers: The Essence of Leadership](#).

Agile managers help facilitate the environment for the team (possibly along with an agile project manager/Scrum Master or a program manager). They are servant leaders. They provide coaching and feedback and meta coaching and meta feedback. The manager’s role is about facilitating what the team members can deliver, to the team, to the product, and to themselves. It’s about inspect and adapt for team members.

That’s where we encounter the affiliation problem. Who do we want the team members to identify with? With the functional manager—who with good intentions—ask a team member to do work not on the board? Or, do we want team members to identify with the team? I vote for the team. (I vote for the team regardless of the approach, agile or otherwise.)

Managers might need coaches so they learn about what the team members do. Then, managers can provide feedback and coaching and meta feedback and meta coaching as someone who can see from the outside of the team. Managers can learn what the impediments are for agile (too-long duration commitments, [resource vs. flow efficiency](#), HR and finance issues). Managers can help lead the organization’s agile approach.

And, many managers need coaching to do this: agile coaching and how the functions work in an agile team. That’s because we want managers to be the best managers they can be to help the organization move to agile.

When managers change what they do, they might need coaching to learn how, and to get the feedback to see the possibilities of their decisions. Management coaching is not about weakness. It’s about the realization that if we want to change culture, we change behavior. Not just team behavior. Management behavior is a key part of the agile transformation.

In part 3, I’ll address coaching and collaboration for middle and senior managers.

Inserted from <http://www.jrothman.com/mpd/agile/2016/10/coaches-managers-collaboration-and-agile-part-2/>

Part 3

by [johanna](#) | Oct 31, 2016 | [agile](#), [MPD](#) | [0 comments](#)

I started this series writing about the need for coaches in [Coaches, Managers, Collaboration and Agile, Part 1](#). I continued in [Coaches, Managers, Collaboration and Agile, Part 2](#), talking about the changed role of managers in agile. In this part, let me address the role of senior managers in agile and how coaches might help.

For years, we have organized our people into silos. That meant we had middle managers who (with any luck) understood the function (testing or development) and/or the problem domain (think about the major chunks of your product such as Search, Admin, Diagnostics, the feature sets). I often saw technical organizations organized into product areas with directors at the top, and some functional directors such as those test/quality and/or performance.

In addition to the idea of functional and domain silos, some people think of testing or technical writing as services. I don’t think that way. To me, it’s not a product unless you can release it. You can’t release a product without having an idea of what the testers have discovered and, if you need it, user documentation for the users.

I don’t think about systems development. I think about product development. That means there are no “service” functions, such as test. We need cross-functional teams to deliver a releasable product. But, that’s not how we have historically organized the people.

When an organization wants to use agile, coaches, trainers, and consultants all say, “Please create cross-functional teams.” What are the middle managers supposed to do? Their identity is about their function or their domain. In addition, they probably have MBOs (Management By Objective) for their function or domain. Aside from not working and further reducing flow efficiency, now we have affected their compensation. Now we have the container problem I mentioned in [Part 2](#).

Middle and senior managers need to see that functional silos don't work. Even silos by part of product don't work. Their compensation has to change. And, they don't get to tell people what to do anymore.

Coaches can help middle managers see what the possibilities are, for the work they need to do and how to muddle through a cultural transition.

Instead of having managers tell people directly what to do, we need senior management to update the strategy and manage the project portfolio so we optimize the throughput of a team, not a person. (See [Resource Management is the Wrong Idea; Manage Your Project Portfolio Instead](#) and [Resource Efficiency vs. Flow Efficiency](#).)

The middle managers need coaching and a way to see what their jobs are in an agile organization. The middle managers and the senior managers need to understand how to organize themselves and how their compensation will change as a result of an agile transformation.

In an agile organization, the middle managers will need to collaborate more. Their collaboration includes: helping the teams hire, creating communities of practice, providing feedback and meta-feedback, coaching and meta-coaching, helping the teams manage the team health, and most importantly, removing team impediments.

Teams can remove their local impediments. However, managers often control or manage the environment in which the teams work. Here's an example. Back when I was a manager, I had to provide a written review to each person once a year. Since I met with every person each week or two, it was easy for me to do this. And, when I met with people less often, I discovered they took initiative to solve problems I didn't know existed. (I was thrilled.)

I had to have HR "approve" these reviews before I could discuss them with the team member. One not-so-experienced HR person read one of my reviews and returned it to me. "This person did not accomplish their goals. You can't give them that high a ranking."

I explained that the person had finished more valuable work. And, HR didn't have a way to update goals in the middle of a year. "Do you really want me to rank this person lower because they did more valuable work than we had planned for?"

That's the kind of obstacle managers need to remove. Ranking people is an obstacle, as well as having yearly goals. If we want to be able to change, the goals can't be about projects.

We don't need to remove HR, although their jobs must change. No, I mean the HR systems are an impediment. This is not a one-conversation-and-done impediment. HR has systems for a reason. How can the managers help HR to become more agile? That's a big job and requires a management team who can collaborate to help HR understand. That's just one example. Coaches can help the managers have the conversations.

As for senior management, they need to spend time developing and updating the strategy. Yes, I'm fond of continuous strategy update, as well as continuous planning and continuous project portfolio management.

I coach senior managers on this all the time.

Let me circle back around to the question in [Part 1](#): Do we have *evidence* we need coaches? No.

On the other hand, here are some questions you might ask yourself to see if you need coaches for management:

- Do the managers see the need for flow efficiency instead of resource efficiency?
- Do the managers understand and know how to manage the project portfolio? Can they collaborate to create a project portfolio that delivers value?
- Do the managers have an understanding of how to do strategic direction and how often they might need to update direction?
- Do the managers understand how to move to more agile HR?
- Do the managers understand how to move to incremental funding?

If the answers are all yes, you probably don't need management coaching for your agile transformation. If the answers are no, consider coaching.

When I want to change the way I work and the kind of work I do, I take classes and often use some form of coaching. I'm not talking about full-time in person coaching. Often, that's not necessary. But, guided learning? Helping to see more options? Yes, that kind of helping works. That might be part of coaching.

Inserted from <http://www.irothman.com/mpd/agile/2016/10/coaches-managers-collaboration-and-agile-part-3/>

Resource Efficiency vs. Flow Efficiency

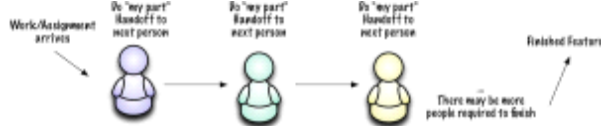
Monday, November 07, 2016 3:56 PM

Part 1: Seeing Your System

by [johanna](#) | Sep 13, 2015 | [agile](#), [MPD](#) | [2 comments](#)

I've been working with a number of people who want to work in a more agile way. These nice folks have one stumbling block: resource efficiency vs. flow efficiency. This is partly because of how they see the system of work.

If you ever used phases or a waterfall approach, you might have tried to optimize resource efficiency:



In this image, you see that the work flows from one person to another. What this picture does not show is the fact that there are delays in the workflow.

Each person is a specialist. That means they—and only they—can do their work. The more senior and the more specialized they are, the more they need to do the work and the less capable other people are (for that work). Think of a UI designer or a database admin. I often see teams who don't have those necessary-for-them roles.

With resource efficiency, you optimize for each person along the way. You get the feature when you get it. Each person is "fully utilized." This leads to a [cost of delay](#). (See [Diving for Hidden Treasures](#) to see more costs of delay.) It also leads to problems such as:

- "It takes forever to bring people up to speed around here."
- "Only Fred can work on that. He's the only one who knows that code (or whatever)."
- "You can't take a vacation. That's just before we want to ship and you're the only one who knows that part of the product."
- Many features are partly done and too few are complete. (The work in progress is quite high.)

Contrast that with flow efficiency:



In flow efficiency, the team takes the feature. The *team* might specialize in that feature area (I see this a lot on programs). If anyone needs to be away from work for a day or a week or two, the team can continue to do the work without that one person. Yes, the team might be a little slower, but they can still release features.

In flow efficiency, it doesn't matter what each person "knows." The team optimizes its work to get features done. You can see this when teams limit the backlog coming into an iteration, when they [pair, swarm, or mob](#) to finish features. If the team uses kanban and they keep to their work in progress limits, they can see flow efficiency also.

Resource efficiency is about optimizing at the level of the individual. Flow efficiency is about optimizing for the feature. If you are transitioning to agile, ask this question, "How do we optimize for features? It doesn't matter if we keep everyone busy. We need to release features." This is a mindset change and can challenge many people.

Here's why you should ask this question: Your customers buy features. They don't buy your busy-ness.

When I tell managers about resource vs. flow efficiency, they often react, "Yes. But how do we know the features won't take more time?" and "How will we know how to do performance management?" I'll address that in [parts 2](#) and [3](#).

Inserted from <<http://www.jrothman.com/mpd/agile/2015/09/resource-efficiency-vs-flow-efficiency-part-1-seeing-your-system/>>

Part 2: Effect on People

by [johanna](#) | Sep 13, 2015 | [agile](#), [MPD](#) | [2 comments](#)

If you haven't read [Resource Efficiency vs. Flow Efficiency, Part 1: Seeing the System](#), I explain there about optimizing for a given person's work vs. optimizing for features. Some people (including managers) new to agile have questions about working in flow vs. optimizing for a person.

The managers ask:

- How do I know the work won't take longer if we move to flow efficiency?
- How do I do performance management if a single person isn't responsible for his/her work? (What's the accountability, etc.?)

This post is about the length of the work and how people feel when they can't finish work.

When you have experts, as in resource efficiency, the work queues up behind the expert. Let's say you have three senior people with these separate expertise areas:

- Cindy has deep knowledge of the internals of the database and how to make things fast. (I think of this as the platform.)
- Brian has deep knowledge of the transaction layer and how to move data from one place to another in the product. (I think of this as the middleware.)
- Abe has deep knowledge of how to present data to the customers and how to create a great customer experience. (I think of this as the UI layer.)

You want Features 1 and 2, which have significant UI impact. Abe is a master at iterating with all the necessary people to get the UI just right. In the meantime, Cindy and Brian go on to Features 3, 4, and 5 because they aren't needed (yet) on Features 1 and 2.

If you measured cumulative flow, you would see that all five features are open for a while, because these three people have started on them and not finished anything.

Abe encounters a problem with the UI. The customer doesn't respond or the product management people are not available. Something impedes his progress. So, he starts Feature 9, which is the next feature with significant UI design.

Notice that he doesn't start the next ranked feature. He starts the next one *he* can work on.

Cindy and Brian also encounter delays because the test automation isn't there, or the build takes too long (or something). Choose whatever happened to you last week as an impediment.

They need to stay busy, so they see that Feature 6 needs them both. They start on it. They realize there is a need for UI work. They ask Abe if he is available. No, Abe is working on Feature 9, not Feature 6.

Now, Cindy and Brian have another feature in progress.

If this sounds like your project, you are not alone. This is a result of resource efficiency.

The human effect of resource efficiency is multitasking, a feeling of impending pressure because you can see the deadline but you're not getting closer. You wonder if you will ever finish the work.

Instead, imagine if Cindy, Brian, and Abe along with a tester took Feature 1. They might prepare their platform and middleware parts of the work. They might help Brian with the prototype generation and iteration. They might be able to bang on doors if Abe needs to concentrate on something specific. "I'll be done with this in a couple of hours. Can you reserve a conference room and ask the product manager to be there? She always gives me a hard time on the UI. I want to know what she thinks of it now." Or, "Can you call Customer A and get ready for a walkthrough in a couple of hours?"

You might think of this reserve-a-room or call-people work as something a project manager *should* do. In reality, these actions are servant leadership actions. Anyone can do them.

We often have lead-time for some parts of development. Even if we want to work in flow, we might need other people to finish.

Even if Cindy and Brian can't directly help with the UI, they can make it easier for Abe to succeed. And, if the tester is involved at the beginning, the tester can create automated tests that don't depend on the GUI. Maybe the developers not working on product code can help with an automated test framework. (I find that testers new to data-driven or automated testing don't always know how to start. Developers might be able to help.)

Imagine if Cindy, Brian, and Abe are not the only people on their team. They are the most senior, and there are two or three other developers on the team. What happens when those more junior developers have a question? Cindy, Abe, and Brian have to stop working on their stories to work with the other people. Or, maybe they don't and the other people are stuck. I see this in teams all the time. I bet you do, too.

When we optimize for resource efficiency, we have people with unfinished, open work. The more work they have not done, the more they have new work queuing up behind them. They work hard all day and don't feel as if they accomplish anything, because nothing gets to done.

When we optimize for flow efficiency, people finish things, together. They have less work in progress. They feel a sense of accomplishment because they can point to what they have completed.

I can't guarantee a team can finish faster in flow because I am not an academic. However, you've heard, "Many hands make light work." That's the idea. When we help each other move a chunk of work to done, we all succeed faster.

[Part 3](#) will talk about what managers perceive as performance management.

Inserted from <<http://www.irothman.com/mpd/agile/2015/09/resource-efficiency-vs-flow-efficiency-part-2-effect-on-people/>>

Part 3: Managing Performance

by [johanna](#) | Sep 13, 2015 | [agile](#), [MPD](#) | [2 comments](#)

[Resource Efficiency vs. Flow Efficiency, Part 1: Seeing Your System](#) explains resource efficiency and flow efficiency. [Resource Efficiency vs. Flow Efficiency, Part 2: Effect on People](#) explains why flow efficiency helps you get features done faster. Here, in part 3, I'll address the performance management question. New-to-agile (and some experienced) managers ask, "How can I manage performance? How will I know people are accountable for their work?"

These are good questions. Performance management and accountability are two different things in flow efficiency.

Here are some ways to manage performance:

- Ask for the results you want.
- Ask the team to work together to produce features.
- Create communities of practice to help people learn their craftsmanship.
- Provide the team members with the knowledge of how to provide feedback and coaching to each other.
- As a manager, you provide meta-coaching and meta-feedback to team members. (The team members [provide each other feedback](#) and coaching, managing their daily performance.) (See also [Four Tips for Managing Performance in Agile Teams](#).)

If you do these things, you will discover that people are accountable to *each other for their work*. The point of a standup is to help people vocalize their accountabilities. If the team works as a swarm or as multiple pairs/triads/whatever, they might not need a standup. They might need a kanban board with WIP (work in progress) limits. If your organization likes iterations because it provides boundaries for decision-making or providing focus, that works. It can work with or without a kanban board.

Here's a question I like to ask managers, "Have you hired responsible adults?" The managers almost always say, "Yes." They look at me as if I am nuts. I then ask, "Is there a reason for you to not trust them?"

Now we get to the real issues. If the managers have encouraged/enforced resource efficiency, the people often multitask. Or, they have to wait for other people to finish their work. People have a difficult time finishing their work "on time." Managing "performance" is a function of the system. The system of resource efficiency requires someone to check up on people, because the expertise bottlenecks can become severe.

Instead, if you manage the system by focusing on what you want—features—instead of tasks, you don't have to do much performance management. Will you make a mistake and hire someone who doesn't fit? Maybe. The team can tell you.

What if you hire a superstar? Maybe you're worried that person won't have enough to do. My experience is that the team will ask the so-called superstar to help them with other things, making her even more of a superstar. In addition, this superstar can help with everyone learning more.

If you don't rub people's noses in the fact that someone might be "better" than they are, they will use that person well. Yes, sometimes, I was the person who learned from the superstar. Sometimes I was the superstar. I never noticed. I noticed I got better when I worked with certain people and asked to

work with them more often.

Think about what makes people happy at work. Once you take money off the table by paying people enough, it's all about [mastery, autonomy, and purpose](#).

As managers, you create the system to provide mastery, autonomy, and purpose. You don't have to manage what people do all day. If you think you do, why would you want to use agile?

BTW, managing for results isn't new. Peter Drucker first published [Managing for Results](#) in 1964.

In [part 4](#), I'll address accountability and what it could mean in flow efficiency as opposed to resource efficiency.

Inserted from <<http://www.jrothman.com/mpd/agile/2015/09/resource-efficiency-vs-flow-efficiency-part-3-managing-performance/>>

Part 4: Defining Accountability

by [johanna](#) | Sep 20, 2015 | [agile](#), [MPD](#) | [0 comments](#)

This is the next in a series of posts about resource efficiency vs. flow efficiency:

- [Resource Efficiency vs. Flow Efficiency, Part 1: Seeing Your System](#)
- [Resource Efficiency vs. Flow Efficiency, Part 2: Effect on People](#)
- [Resource Efficiency vs. Flow Efficiency, Part 3: Managing Performance](#)

Managers new to agile often ask, "How do I know people will be accountable?" Let's tease apart the pieces of accountability:

- Accountable to the project for finishing their own work
- Accountable to their team for participating fully by doing their work
- Accountable to help other people learn what they did by documenting their work
- Accountable for meeting their estimates
- Accountable for how the project spends money
- ... There might be more accountabilities

Let's take the first two together:

Accountable for finishing their own work and by doing their work

I suspect that managers mean, "How do I know each person does their work? How do I know they aren't asking other people to do their work? How do I know these people are learning to do their own work?"

Those are good questions. I have yet to see a single-person feature. At the least, a developer needs someone to test their work. Yes, it's possible to test your own work. Some of you are quite good at that, I bet. Many people are not. If you want to prevent rework, build in checking in some form or another: pairing, design review, code review, unit tests, system tests, something.

So the part about "own work" seems a little micro-managing to me.

The part about doing their work is a little trickier. When people get stuck, what do they do? Often, they ask someone else for help. The help might be: talk to the duck, an explanation about what is going on in that area of the product, pairing on solving the problem, or even talking to more people.

There is no such thing as "cheating" at work. Managers are right to be concerned that people work to their capabilities. And, if those people are stuck, I don't want them mired in the problem. We want people to learn, not be stuck.

Here's the answer: You can't know as a manager. You never did know as a manager.

The team knows who is hardworking and who slacks. The team knows how people are learning and if they are stuck. Even in waterfall days, the team knew. Managers may have had the illusion they knew, but they didn't. Managers only knew what people told them.

Accountable for documentation

For me, the question is who will use the documentation? I am always amazed at how many managers want documentation other than end-user documentation and how few teams find this useful. In agile, you could make it part of the definition of done.

If people build documentation time into their estimates and the team finds the internal documentation useful, the team will pressure each person to deliver their documentation. The team knows whether the

person documents their work.

Accountable for living up to estimates

When I ask managers if they want good estimates or completed features, they almost always say, “completed features.” Too often, the people multitask on fixing defects or production support work while they are supposed to work on a feature. I do understand the need for [estimates](#), but holding people to their estimates? Sometimes, that’s about money.

Accountable for how the project spends money

Of all the accountabilities, this one makes the most sense to me. However, it doesn’t make sense in agile, where the customer/product owner is the one responsible. As long as the team completes features, the PO determines when either there is no more value in the backlog, or there is no more money for the project. With any luck, the team has met the release criteria by this time.

For me, the idea of accountability is a team-based idea, not a person-based idea. In flow efficiency, you can ask the team to be accountable for:

- Finishing features
- Knowing where they are with respect to the features in progress
- Helping the PO understand the value of features and how long features will take
- Providing an estimate, if necessary
- If the team works in iterations, committing to work and not overcommitting to too much work

When you look at accountability like this, it looks pretty different than a single person’s accountability.

[Part 5](#) is the summary post for this series.

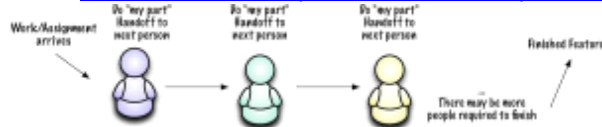
Inserted from <<http://www.jrothman.com/mpd/agile/2015/09/resource-efficiency-vs-flow-efficiency-part-4-defining-accountability/>>

Part 5: How Flow Changes Everything

by [johanna](#) | Sep 20, 2015 | [agile](#), [MPD](#) | [0 comments](#)

The discussion to now:

- [Resource Efficiency vs. Flow Efficiency, Part 1: Seeing Your System](#)
- [Resource Efficiency vs. Flow Efficiency, Part 2: Effect on People](#)
- [Resource Efficiency vs. Flow Efficiency, Part 3: Managing Performance](#)
- [Resource Efficiency vs. Flow Efficiency, Part 4: Defining Accountability](#)



When you move from resource efficiency (experts and handoffs from expert to expert) to flow efficiency (team works together to finish work), everything changes.



The system of work changes from the need for experts to shared expertise.

The time that people spend multitasking should decrease or go to zero because the team works together to finish features. The team will recognize when they are done—really done—with a feature. You don’t have the “It’s all done except for...” problem.

Managers don’t need to manage performance. They manage the system, the environment that makes it possible for people to perform their best. Managers help equip teams to manage their own performance.

The team is accountable, not a person. That increases the likelihood that the team will estimate well and that the team delivers what they promised.

If you are transitioning to agile, and you have not seen these things occur, perform a retrospective. Set aside at least two hours and understand your particular challenges. I like [Agile Retrospectives: Making Good Teams Great](#), [Getting Value out of Agile Retrospectives – A Toolbox of Retrospective Exercises](#), and [The Retrospective Handbook: A Guide for Agile Teams](#) for creating a retrospective that will work for you. You have unique challenges. Learn about them so you can address them. I hope you enjoyed this series. Let me know if you have questions or comments.

Inserted from <<http://www.jrothman.com/mpd/agile/2015/09/resource-efficiency-vs-flow-efficiency-part-5-how-flow-changes-everything/>>

Cost of Delay

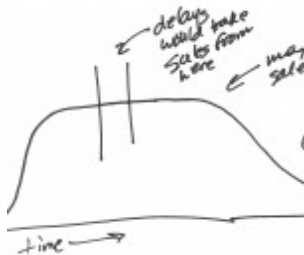
Monday, November 07, 2016 3:56 PM

Not Shipping on Time, Part 1

by [johanna](#) | Feb 5, 2014 | [MPD, portfolio management](#) | [14 comments](#)

Do you know about the Cost of Delay? It's the way to think about the revenue you can lose plus the cost of continued development.

One of my managers many years ago had a back of the napkin approach to cost of delay.



Cost of Delay,
Back of the Napkin

"Johanna, we want to ship this product in the second quarter this year. We estimate it will take us a quarter to ramp up sales. We think there is a lifetime sales of about five years for this product. Any delays in our shipment will not push our sales figures to the right. They will remove our max sales from the middle. Got it? We have to make our ship date."

I got it.

There weren't too many of us developers in that organization. We all knew what our job was: to release on time, and make sure the product was usable. The product didn't have to be perfect. The product had to be usable, because we could release fixes *if we needed to*, but we had to climb that sales ramp to get to that max sales point.

We worked on one project at a time, until it was done. Then we went to the next project. We worked on that project. None of our projects was very long. Why? Because we needed to realize revenue. You can't realize revenue with a product-in-a-box if you don't ship it.

We didn't ship too many fixes. Oh, not because we were perfect the first time. We asked each other for review, and we found problems, so we fixed them inside the building. Before we shipped. Because the cost of not shipping on time was too great.

When you delay your release and don't ship on time, you miss the revenue from the *maximum* sales times. Take your delay in weeks, and remove the revenue weeks. That's your cost of delay, back of the napkin approach.

You can go through more serious calculations. Troy Magennis of [Focused Objective](#) talks about a "compounding impact" on other projects. I am sure he's correct.

But even if you said, "Every week we slip, we lose at least a week of revenue from our *maximum* sales week of revenue," do you think people would notice?

How do you release on time? You fix scope (ha!). You have [release criteria](#). You have [shorter projects](#), because they are easier to estimate and deliver. You use an [incremental or an agile lifecycle](#), so you have more predictability about your release.

This post is the simple idea of not shipping on time. But what about when you have competing projects and not enough people? Look for Part 2, next.

P.S. After I wrote this post, I realized I was not living this post. (Why do you think I write? It's not just for you. It's for me too.) I published the incomplete [Essays on Estimation](#). I have another essay or two to release. But if I don't release it, you can't read it and get any value from it, can you? The point: if you don't release your products, your customers can't get any value. Hat Tip to @jlottsen who said in a tweet, "you have to release it, or no one can use it, and you can't realize any revenue at all". Very true.

Inserted from <http://www.jrothman.com/mpd/portfolio-management/2014/02/cost-of-delay-not-shipping-on-time-part-1/>>

Due to Multitasking, Part 2

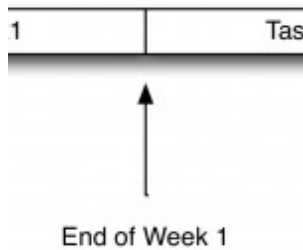
by [johanna](#) | Feb 6, 2014 | [MPD, portfolio management](#) | [9 comments](#)

In [Cost of Delay: Not Shipping on Time, Part 1](#), I introduced you to the notion of cost of delay. I said you could reduce the cost of delay by managing your projects: have shorter projects, using release criteria, or selecting a lifecycle that manages the risk.

Sometimes, you don't have short projects, so projects get backed up, and your managers ask you to work on several things at once. Or, the managers can't decide which project is #1. Somehow, you end up doing several things "at once." This is the multitasking problem, and this is the cost of delay in this part.

You know me, I hate [multitasking](#). The [costs](#) are significant. But those are just the costs of multitasking on the work you are doing now. That has nothing to do with the cost of delay to your *projects*.

In [Manage It!](#), I have this nice picture of what happens when you try to multitask between two projects.

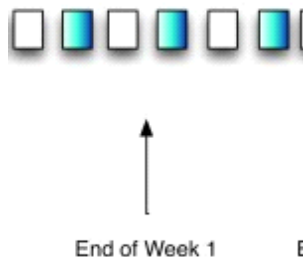


Two Tasks or Projects

Imagine you have two projects. Each of them should take a week. Hey, they're short projects. I'm trying to illustrate a point.

You can deliver one of them at the end of the first week. You can deliver the second one at the end of the second week.

But, imagine you have to multitask between them. Instead of being able to deliver anything at the end of the first week, you can't deliver anything at all until the second week. And, depending on how much context switching you do, you might not be able to deliver anything until sometime during the third week. The blue boxes show the time lost due to context switching.



Effect of Multitasking

Delay to Delivery

This is a huge cost of delay due to multitasking.

How do you calculate this?

"Big" is not quantitative enough for some people. This is where using some of the tools from the people who do this for a living might be good.

I've always drawn a picture like this and explained, "I don't know how to estimate the time in the blue boxes. The blue boxes are the delay times. I can't estimate them, because everyone else is delayed by multitasking on their projects, too. Sometimes, it takes me an entire day to get an answer to a question that should only take me an hour to get an answer. All I know is that "ideal" time is irrelevant to actual time. And even calculating using relative estimation is hopeless. That's because we are multitasking. All of our estimations are out the window because we are multitasking.

"The amount of time we spend working on a project might be accurate. It's the wait time that's killing us. I don't know how to estimate that."

What I've done before is take a two- or four-week window, and ask people to write down their

predictions of what they thought some task would take. Then, I ask them to write down, as they spend their time on each task, how much time they actually spend on each task. Now you can compare the prediction to reality.

Remember that all of the context switching and wait time is the time you need to remove from the maximum sales revenue, had you released the product.

This is very difficult to do. It saps the morale of the people doing the work. They quickly realize how often they go back to the same thing, over and over and over again, making zero progress, trying to realize where they are. Do not do this for more than a four-week window. If you must do this, label this an experiment to obtain data. Explain you are trying to obtain actual work time, context switching time, and wait time. Let people label the time as they will.

The managers will be astonished by how little time the technical staff spend on real work. They will be amazed by how much time the technical staff spend on context switching and wait time. This is why managers think technical staff are bad at estimation. Who can be good at estimation when they are multitasking?

The problem is this: the cost of delay due to multitasking is real. It's invisible to most people, especially management. It's not just the cost of the blue boxes in the picture above. It's the fact that nothing gets out of your organization until the end of Week 2 or later. Remember the back of the napkin in [Part 1](#)?

Even if you use that calculation, you can see you are losing revenue left and right due to multitasking.

Remind the managers: Remember that all of the context switching and wait time is the time you need to remove from the maximum sales revenue, had you released *any* of the products.

Can you use this as an argument for your managers to end the multitasking and [manage the project portfolio](#)?

[Cost of delay](#) due to multitasking is real. What would you need to know to calculate it in your organization?

Inserted from <<http://www.jrothman.com/mpd/portfolio-management/2014/02/cost-of-delay-multitasking-part-2/>>

Due to Indecision, Part 3

by [johanna](#) | Feb 6, 2014 | [MPD, portfolio management](#) | [0 comments](#)

In Part 1, we discussed the [cost of delay of not shipping on time](#). In Part 2, we discussed the [cost of delay of multitasking](#). In this part, we'll discuss a cost of delay due to management indecision.

Here's a problem I encounter often. A middle manager calls me, and asks for an estimation workshop. I ask about the environment. The manager tells me these things:

- The developers meet their dates and the testers never do (this is not an estimation problem)
- The project starts on time, and the project staff comes in close, but not quite on time (this might be an estimation problem)
- The project starts off late (this is not an estimation problem)

When the developers meet their dates but the testers never do, it's almost always a couple of things: [Schedule Chicken](#), or technical debt masquerading as done in a waterfall project.

If the project starts on time and it's close, it might be an estimation problem, assuming no one is multitasking.

But if the project starts late, this is not an estimation problem. This is a cost of delay due to management indecision.

Wouldn't it be nice if you could say,

A lack of planning on your part does not constitute an emergency on my part

to your managers? Well, just call me the Queen of the Career Limiting Conversation. This is why managers are supposed to [manage the project portfolio](#).

When I hear the plea for the estimation workshop, it's for these reasons:

- The managers have received estimations for the work, and they don't like the estimates they have received.
- Someone other than the project team did the estimation two years ago. They know the estimate is out of date, and they need a new estimate.

- The managers still can't make a decision, because they are trying to decide based on ROI, date, or cost or some sort of hard to predict quantitative measure, rather than on [value](#).

The managers are, ahem, all tied up in their shorts. They can't decide, which is a decision. They don't fund the potentially transformative projects or programs. They go with the safe bets. They do the same projects over and over again.

If they get lucky, they choose a project which is small and has a chance of success.

How do you discuss this cost of delay? You ask this question:

When did you first start discussing this project as a potential project? or When did this project first go on our backlog?

That provides you the first date you need. This is your next question:

When did we actually start working on this project?

You want to see how long it takes you to consider projects. It's okay to consider some projects for a while. The difference between the time you first start discussing a project and the time you start working on it is your *management cost of delay*. I just made that up. I bet there's a real name for it.

What is the difference in those two dates?

Project lead time is the time you started discussing a potential project and the time you *finished* it.

Project cycle time is the time you started a project and the time you finished it. Yes, we normally discuss lead time and cycle time for features. But sometimes, it makes sense for projects, too.

If you release projects, not features, and you have managers who have decision problems, they need to know about this cost of delay. Because the project lead time will take time out of your maximum revenue. The longer that lead time is, the more it will take.

The worst part is this: how much value does the project have, if the project lead time is "too long?"

What can you do?

1. Track your project lead times. Decide how much time a project can spend on the backlog in the project portfolio before you decide to transform or kill it. Yes, I am serious.
2. Shorten all your projects, so you release something at least every six months, or more often. That provides you more capability in assessing your project portfolio and frees teams for more work.
3. Move to an incremental lifecycle or an agile approach.

I didn't say it was easy. It's healthier for your organization.

Who do you think can measure this cost of delay in your organization? What do you think might have to change to make this cost of delay visible?

Inserted from <<http://www.jrothman.com/mpd/portfolio-management/2014/02/cost-of-delay-due-to-indecision-part-3/>>

Due to Technical Debt, Part 4

by [johanna](#) | Feb 13, 2014 | [MPD, portfolio management](#) | [4 comments](#)

[Cost of delay part 1 was about not shipping on time](#). [Cost of delay part 2 was due to multitasking](#). [Cost of delay part 3 was due to indecision](#). This part is the cost of delay due to technical debt.

One of the big problems in backlog management is ranking technical debt stories. It's even more of a problem when it's time to rank technical debt projects. You think product owners have [feature-itis](#) problems? Try having them rank technical debt projects. Almost impossible.

But if you really want the value from your project portfolio, you will look at your impediments. And, if you are like many of my clients, you have technical debt: a build system that isn't sufficiently automated, insufficient automated system tests, too many system-level defects, who knows what else.

If you addressed the build system, and maybe some of the system tests, if you created a timeboxed technical debt project, you could save time on all of the other projects in this code base. All of them. Imagine this scenario: you have a 2000-person Engineering organization. It takes you 3 weeks (yes, 21 calendar days) to create a real build that you know works. You currently can release every 12-18 months. You want to release every 3-6 months, because you have to respond to market competitors. In order to do that, you have to fix the build system. But you have a list of possible features, an arm and a leg long. What do you do?

This client first tried to do more features. They tried to do features in iterations. Oh, they tried.

By the time they called me, they were desperate. I did an assessment. I asked them if they knew how much the build system cost them. They had a group of 12 people who “supported” the build system. It took at least 10 days, but closer and closer to 20-25 days to get a working build. They tried to estimate the cost of the build in just this group of people: 12 people time 21 days. They did not account for the cost of delay in their projects.

I showed them the back of the napkin calculation in [part 1](#), and asked, “How many releases have you postponed for at least a month, due to the build system?” They had an answer, which was in the double digits. They had sales in the millions for the maximum revenue. But they still had a sticking point.

If they funded this project, they would have no builds for four weeks. None. Nada. Zilch. And, their *best* people (whatever that means) would be on the build project for four weeks.

So, no architecture development, no design, no working on anything by the best people on anything other than the build system. This company was convinced that stopping Engineering for a month was a radical step.

Does it matter how long your iterations are, if you can’t build during the iterations and get feedback? They finally did fund this project, after about six months of hobbling along.

After four weeks of intense work by 16 of their smartest people, they had an automated build system that anyone in Engineering could use. It still took 2 days to build. But that was heaven for everyone. They continued the build system work for another month, in parallel with regular Engineering work to reduce build system time.

After all the build system work, Engineering was able to change. They were able to transition to agile. Now, Engineering could make progress on their feature list, and release when it made sense for their business.

What was the payback for the build system work? Almost immediate, Engineering staff said. When I asked one of the VPs, he estimated, off the record, that they had lost more than the “millions” of dollars of revenue because they did not have the features needed at the time the market demanded. All because of the build system.

People didn’t plan for things to get this way. They got that way a little at a time, and because no one wanted to fund work on the build system.

This is a dramatic story due to technical debt. I bet you have a story just like this one.

The cost of delay due to technical debt is real. If you never look at your technical debt and see where it impedes you, you are not looking at the value of your entire project portfolio.

If you eliminated a technical debt impediment, would that change one of your costs of delay?

Inserted from <<http://www.jrothman.com/mpd/portfolio-management/2014/02/cost-of-delay-due-to-technical-debt-part-4/>>

Due to Other Teams' Delay, Part 5

by [johanna](#) | Feb 15, 2014 | [MPD, portfolio management](#) | [2 comments](#)

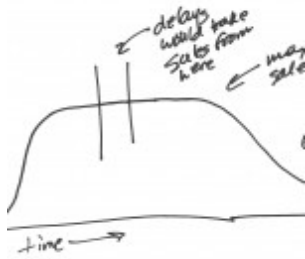
Imagine you have a large program, where you have several teams contributing to the success of one business deliverable. You are all trying to achieve a specific date for release.

One team is having trouble. Maybe this is their first missed deliverable. Maybe it’s their second. Maybe they have had trouble meeting their deliverables all along—they have “delivered,” but the deliverables don’t work.

Now, say you’re halfway through the desired program time. You, as a program manager, can see that this program is in trouble. That one team is not going to make it. What do you do?

This is where you start talking to people and understanding what the *value* of everything is.

1. Does the program need everything on that team’s backlog?
2. Does the program need everything on other team’s backlogs?
3. Does the program product owner understand the cost of delay?
4. How about the sponsors for the program? Do they know what’s going on?



Cost of Delay, Back of the Napkin

Take out your back of the napkin calculation and show it to anyone who will listen.

Does the team understand the problem of lateness, as we discussed in [part 1](#)? They might. They might be overwhelmed by the technical difficulty of what they are doing. Maybe their stories are huge. Maybe they aren't so hot at agile. Maybe they aren't working in an agile way. There are 1001 reasons for this problem. If you are a manager of any stripe, program manager, development manager, whatever, you are responsible for understanding first, not yelling. (I often see senior development managers, VP Engineering encountering this, not realizing that they are the program managers in this situation.)

Does the program product owner really need everything in their backlog? It's time to consider how [little can that team do](#), and still deliver something of value. Or, it's time to consider how little other teams do and still deliver something of value. Or, it's time to rejigger who is doing what. Otherwise, you are going to lose the money in the middle of the revenue stream.

Are the teams working by layer, front end, middleware, back end, instead of through the architecture? Something has to change in this program. You are not going to make the desired date. This problem is a larger case of the not shipping on time problem, but it's not just one team. It involves more teams. And, with a program, it involves more money. It's almost always a bigger stake.

This is when you want to start considering cost of delay for features. Yes, not just for releases, but for features.

Each feature has value when you deliver it at a certain time. Before a certain time, it might have more or less value. After a certain time, it has less value.

Who knows when that time is? Who can make these decisions in your organization. Sometimes that person is called a product owner, a product manager, or, gasp, a customer. Sometimes, that person is called a Marketing Manager or Director, or even a CEO. Rarely is that person called a developer or tester, or even a development manager or test manager or an Engineering manager or CIO or VP of Engineering. Sometimes the product development side knows. Almost never does the product development team know by themselves.

If you make decisions as a cross-functional team, product development and marketing, you can get the best of both worlds. You can learn about the technical risks—especially good if the team is having technical problems. You can learn about the cost of delay risks—especially good from the customer perspective. Now, you can make a decision about what to do for the good of the program.

You have to optimize for the program's throughput, not any one team's throughput.

In my world, you work with the team: do they want to stay together? Are they working well together, and having a tough time technically? If so, the team has many options:

- The team can ask for technical help from another team (get an architect/designer to work *with* the team)
- The team can split their backlog among several other teams, to give them a fighting chance.
- If the team is not working well together (and they will tell you if they are still storming), offer them a chance to split. Or, you can facilitate a better working relationship so they can work well together.

If you don't ask the team, you don't know what's wrong.

The problem with this cost of delay is that it's tricky to discuss, it's tricky to estimate, and it's tricky to fix. It's real. I bet you've seen it.

I would take out the napkin and remind the team that their delay costs the entire program. I want to help them remove that delay.

If you are using a waterfall approach to your programs, this cost of delay is invisible until the end of the

program, when testing occurs. Why? Because that's when you start to see features emerge. If you are using an agile approach or at least an incremental lifecycle, you will start to see this much sooner, and you can do something about it.

Inserted from <<http://www.jrothman.com/mpd/portfolio-management/2014/02/cost-of-delay-due-to-other-teams-delay-part-5/>>

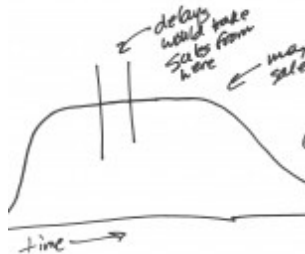
Why You Should Care, Part 6

by [johanna](#) | Feb 18, 2014 | [MPD, portfolio management](#) | [2 comments](#)

I've outlined five potential costs of delay in the previous five posts:

- The delay from not [releasing on time](#), part 1
- The delay from [multitasking, part 2](#)
- The delay from [indecision, part 3](#)
- The delay from [technical debt, part 4](#)
- The delay from [other teams as part of a program](#), part 5

The real problem is this: Why should you care? Maybe you have a "sweet spot" in the way you start your projects or release them. "It just takes that long here." (That's the sign of a system impediment.)



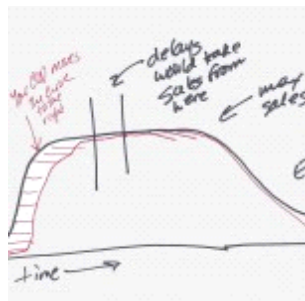
Cost of Delay, Back of the Napkin

Let me show you two pictures. The first you've already seen, the back of the napkin, where you see the effect on your potential maximum revenue.

Now, let's try to calculate that cost of delay.

The longer your delay, the more your cost of delay moves your curve to the right. The more sales you lose out of the *Maximum* potential sales.

The longer it takes for you to release, the more sales you lose from the maximum potential sales. You wait a month to release? You lose a month of max sales. You wait a year to release? You lose a year of max sales.



Cost of Delay,

Showing Effect of Delay on Sales

That's right. Do those numbers start to look big and scary now?

You not only have aggravation and impediments in your current project from the delays from multitasking, technical debt, indecision, slowdowns from other teams, you lose the potential revenue from maximum potential sales, every week you don't release.

Now, I am not saying you should release horrible product. Goodness knows, we have enough of horrible product that barely works or doesn't work at all out in the field. I want to see great products! The idea behind this picture is so people understand that it is worth their while to consider change.

You can change to shorter projects and release something useful faster. (See [Manage It! Your Guide to Modern, Pragmatic Project Management](#))

You can change to agile or incremental lifecycles so they can see progress and release something useful faster. (See [What Lifecycle? Selecting the Right Model for Your Project](#) and [Manage It! Your Guide to Modern, Pragmatic Project Management](#) for an in-depth discussion of lifecycles. You have more options than just waterfall and agile.)

You can adopt a more reasonable approach to the project portfolio, and make decisions more frequently. (Does anyone really think project portfolio decisions last a year? Come on.) (See [Manage Your Project Portfolio: Increase Your Capacity and Finish More Projects](#))

You can consider paying off some technical debt. I used the build system in my example. I could have used automated tests or the thousands of defects some of you have in your defect tracking systems. It's whatever prevents you from knowing you could release on a "moment's" notice. I would like that moment to be under an hour. I bet for many of you, less than a day would be a huge improvement. You can optimize for the entire project or program, not their feature or team. Too often, I hear, "My part is done." That does not matter. It matters what the entire team, project, or program finishes. In my talks about project portfolio management, I am fond of saying, "It doesn't matter how many projects you start, it matters how many you finish."

With cost of delay, it matters *when* you finish them. Because until you finish them, no one pays you. I have taken a high level approach to cost of delay in these posts. I wanted to simplify it so you and I could understand it, a zeroth approach, if you will. If you want more information, start here:

- [Blackwanfarming.com](#)
- [Focused Objectives](#)

Cost of delay is why you cannot take an ROI or use date or budget estimates to manage the project portfolio. This is why you must use [value](#). I look forward to your comments.

Inserted from <<http://www.jrothman.com/mpd/portfolio-management/2014/02/cost-of-delay-why-you-should-care-part-6/>>

Four Tips for Managing Performance in Agile Teams

Monday, November 07, 2016 3:56 PM

Four Tips for Managing Performance in Agile Teams

by [johanna](#) | Mar 3, 2015 | [agile](#), [MPD](#) | [4 comments](#)

I've been talking with clients recently about their managers' and HR's transition to agile. I hear this common question: "How do we manage performance of the people on our agile teams?"

1. Reframe "manage performance" to "career development." People on agile teams don't need a manager to manage their performance. If they are retrospecting at reasonable intervals, they will inspect-and-adapt to work better together. Well, they will if managers don't interfere with their work by creating experts or moving people off project teams.
2. The manager creates a trusting relationship with each person on the team. That means having a one-on-one weekly or bi-weekly with each person. At the one-on-one, the manager provides tips for feedback and offers coaching. (If the person needs it or wants it from the manager.) The person might want to know where else he or she can receive coaching. The manager removes obstacles if the person has them. They discuss career development.
3. When managers discuss career development, each person needs to see an accurate view of the value they bring to the organization. That means each person has to know how to give and receive feedback. They each have to know how to ask for and accept coaching. The manager provides meta-feedback and meta-coaching.
4. If you, as a manager, meet with each person at least once every two weeks, no problem is a problem for too long. The people in the team have another person to discuss issues with. The manager sees the system and can change it to help the people on the team.

Now, what does this mean for raises?

I like to separate the raise from the feedback. People need feedback all the time, not just once a year. That's why I like weekly or biweekly one-on-ones. Feedback isn't just from the manager to the employee; it's two-way feedback. If people have trouble working in the current environment, the managers might have a better chance to change it than an employee who is not a manager.

What about merit raises? This is tricky. So many managers and HR people continue to think one person is a star. No, on well-functioning agile teams, the *team* is the star—not individuals. You have options:

- Make sure you pay each person at parity. This might not be trivial. You need expertise criteria for each job level.
- When it comes to merit raises, provide a pot of money for the team and ask them to distribute it.
- Distribute the merit money to each person equally. Explain that you are doing this, so people provide feedback to each other.
- Here's something radical: When people think they are ready for a raise or another level, have a discussion with the team. Let the team vote on it.

Managers have to not get in the way when it comes to "performance management." The people on the team are adult humans. They somehow muddle through the rest of their lives, successfully providing and receiving feedback. They know the worth of things outside work. It's just inside work that we keep salary secret.

It might not fit for you to have open-book salaries. On the other hand, how much do your managers and HR do that interferes with a team? You have to be careful about this.

If you reward individuals and ask people to work together as a team, how long do you think they will work together as a team? I don't know the answer to that question.

Long ago, my managers asked me to be a "team player." One guy got a huge raise—and I didn't, although I had saved his tush several times—I stopped working as a "team" member. I got my big raise the following year. (Year!)

This incongruent approach is why people leave organizations—when the stated way "we work here" is not congruent with the stated goals: agile and self-organizing teams.

What do you want? Managers and HR to manage people? Or, to lead people using servant leadership, and let the teams solve their problems and manage their own performance?

If teams don't know how to improve, that's one thing. But, I bet your teams do know how to improve. You don't have to manage their performance. You need to create an environment in which people can do their best work—that's the manager's job and the secret to "managing performance."

Building a Team Through Feedback

by [johanna](#) | Oct 23, 2012 | [Articles](#) | [0 comments](#)

By [Lisamarie Babik](#), and Johanna Rothman – October 23, 2012

If you walk through a high-performing agile team space, you'll hear a buzz about the product:

"Do you see this?"

"Bump that."

"Ah ha!"

"Tell me more about what you want."

"But that is part of the acceptance criteria."

"We should do it this way."

What you won't hear is just as telling, especially on a not-so-high-performing team. Sometimes, you have to see the sideways glances and grimaces to know that things are not working well interpersonally. And, if things are not working well between people, then they also will not work well in the product. Conway's Law (paraphrased: "The product architecture reflects the team's architecture") proves that. In traditional team cultures, feedback has been the sole responsibility of the manager. On agile teams, it's more important for feedback to be peer to peer, because the manager doesn't know the minute-by-minute details of what's going on within the team.

What if it were not only the sole responsibility of managers but also of the team itself to provide guidance for an individual's growth? Building feedback into the culture creates a team with members who have a stronger sense of responsibility for one another. They will self-correct interpersonal and work-quality issues better and faster than a traditional team.

A Peer-to-peer Model of Feedback

In order to provide each other feedback, the first thing you need is a peer-to-peer model of feedback. This is the one we like from [Behind Closed Doors: Secrets of Great Management](#) by Johanna Rothman and Esther Derby:

- Create an opening to deliver feedback.
- Describe the behavior or result in a way that the person can hear.
- State the impact using "I" language.
- Make a request for changed behavior.

This model is useful for providing change-focused feedback as well as feedback that reinforces positive behaviors.

What About Performance Reviews?

What about performance reviews for helping people to receive feedback about their behavior? Might that not be enough? We have ample evidence that says once-yearly performance reviews don't work. (See the further reading below.) We don't need manager-led performance management or formal performance management. We need in-the-moment feedback, so that people know if they are breaking the build, picking their noses, or doing something else that drives their colleagues crazy.

We don't need formality in an agile environment. We need just a little bit of tooling, such as a way to provide each other direct, reinforcing, and change-focused feedback about things that work and don't work so we know if should continue or change what we are doing.

Feedback That Builds a Team

First, let's discuss what feedback looks like. You've seen interactions that tear a team apart. People label each other. They say things such as "You always do that," "You never do that," or "What were you thinking?" None of those is helpful.

Instead, imagine the following scenario.

Jimmy normally leaves at 4 pm so that he has a slightly longer weekend during the summer—an agreement he has made with the team. Lauren has noticed that, on the past two Fridays, Jimmy has checked in code at 3:45 pm that has broken the build, and then Jimmy has disappeared. Lauren wants to make sure this does not happen again and catches Jimmy on Friday at 2 pm.

"Do you have a minute?" Lauren says.

"I'm trying to finish this story before I leave. Can it wait until Monday?" Jimmy says.

"Well, that's what I want to talk to you about. The past two Fridays, you checked in code that broke the build. I know you want to finish your stories, but I don't want the build to break. When the build breaks, it affects me as a technical lead because the team can't build. Everyone is running around saying, 'What the heck happened?' When we discover it's in your code, you're not here to help. When I hear the team complaining about that, I tell them, 'Wait a minute, Jimmy has an arrangement to leave early on Friday.' One person told me, 'Well, we might as well all leave early on Fridays.'"

"Oh. I didn't realize."

"I figured you didn't realize. How can we resolve this?"

"Well, I guess I could pair on my stories."

"That's one great solution. Do you have another idea?"

"You want another idea?"

"Well, what if no one is available? I happen to know that no one is available right now. So, today, you can't pair with anyone. You would have had to ask yesterday to pair today. So, you need another option—maybe even two."

"Well, I could do test-driven development. That way, I would know that my code would pass the tests."

"I like that one!"

"I thought you would. Hmm, Maybe I could stop taking stories alone. I thought I had to, since I was leaving early, but maybe I don't."

"I don't think you have to take stories alone either. Even if you leave early, your other teammates can continue to work together when you're gone. So, now you have three great options: pairing, test-first development, and swarming. Or—a fourth option—you could wait until Monday to finish, when your brain is less focused on getting out of the office! What are you going to do today?"

"Uh," Jimmy says. "Not check anything in until I have worked with someone else?"

"Great idea!" Lauren says. "Thanks!"

This feedback scenario took fewer than five minutes. It didn't go on anyone's HR record. It's not a management issue. And, we bet that in two or three months, no one will remember it. That's because the team managed it.

Unpack the Feedback

A lot occurred in the above scenario. Let's break it down.

1. Ensure that there is no embarrassment.

First, it was private. Lauren took Jimmy aside. She could have done this in a group of people, but she didn't want to embarrass him. She assumed he was doing the best job he could. That assumption led her to believe that he was rushing through his work rather than leaving things to the last minute. Since her assumption was a generous interpretation, she decided Jimmy needed a private conversation.

Because feedback often requires privacy, it can be hard to pull off in the team room. However, two people can often find a quiet corner, move into a hallway, take a walk, or find an empty conference room. One of our favorites is a coffee shop, because no one around you has any interest in what you're talking about. The key is to find a neutral location so the two people have a place to discuss the issue. Do not try to talk about the conversation in the team room, where the buzz is too loud—or, worse, where the buzz might stop just as you start giving feedback and the feedback will be out in the open. Find a private place for feedback.

2. Focus on the data.

Lauren provided Jimmy data. Lauren did not label Jimmy. Lauren did not use the word "always" or "never" or "everyone." She provided data and explained how his actions affected her directly. She did not speak for other people.

3. Explain how you are affected, using "I" language.

Feedback is most effective when you explain how you are personally affected. It's least effective when you try to intervene on someone else's behalf. Lauren said, "When the build breaks, it affects me as a developer manager because I can't build." If she had said, "The developers can't build," that would have been weaker. She was speaking from first-hand experience.

4. Ask for joint problem solving.

Lauren also didn't assume she had all the answers. She could have told Jimmy what to do. She certainly had some ideas. Here, Lauren moved into coaching Jimmy by helping him to consider more

than one option.

Sometimes you don't have more than one option, but often you do. If you can consider at least three options, your solution will be stronger for it.

Feedback Works Regardless of the Issue

You might think that this type of feedback works for only work-related issues. However, we also have seen this kind of feedback work for people who don't bathe regularly, who have bad breath, and who sneeze into their hands before a handshake—in other words, all the interpersonal issues that make you say, "Ick!"

When you want to provide this kind of feedback, say, "You may not realize this, but you just sneezed into that hand. I don't want to get sick, so I'll wait until you wash your hands to shake." The other person almost always says, "Oh, I didn't realize! Let me wash with soap and be right back!" Chances are quite good the other person did not realize.

Getting Started

Somewhere in your life—either at work or at home—there is someone who needs feedback. Does your pair partner eat Cheetos while he's got the keyboard? Did your manager come to the office with a nasty cold rather than taking a sick day? If you don't talk to them about the effects of their behavior on you, then they might never realize that there's an issue. Remember that feedback is not always negative! If your son emptied the dishwasher instead of putting his dirty glass in the sink, let him know how much you appreciate it!

Keep these four steps in mind:

1. Ensure that there is no embarrassment. That's why you want to create an opening and ask for a private place to deliver the feedback.
2. Focus on the data.
3. Explain how you are affected using "I" language.
4. Ask for joint problem solving.

We hope this article helps you build your teams through feedback.

Further Reading

On feedback:

- Esther Derby has written about this extensively on her blog at www.estherderby.com/tag/feedback.
- Rothman, Johanna and Esther Derby, *Behind Closed Doors: Secrets of Great Management* (Dallas and Raleigh: Pragmatic Bookshelf, 2005).
- Seashore, Charles, Edith Seashore, and Gerald M. Weinberg, *What Did You Say? The Art of Giving and Receiving Feedback* (Bingham House Books, 1997).

On why performance reviews are so stupid:

- Buckingham, Marcus and Curt Coffman, *First, Break All the Rules: What the World's Greatest Managers Do Differently* (New York: Simon & Schuster, 1999).
- Hope, Jeremy and Robin Fraser, *Beyond Budgeting: How Managers Can Break Free from the Annual Performance Trap* (Harvard Business Press, 2003).
- Kohn, Alfie, *Punished by Rewards* (New York: Houghton-Mifflin, 1993).
- Pfeffer, Jeffrey, *The Human Equation: Building Profits by Putting People First* (Boston: Harvard Business School Press, 1998).
- Pfeffer, Jeffrey, *What Were They Thinking? Unconventional Wisdom About Management* (Boston: Harvard Business School Press, 2007).
- Pfeffer, Jeffrey and Robert I. Sutton, *Hard Facts, Dangerous Half-Truths And Total Nonsense: Profiting From Evidence-based Management* (Boston: Harvard Business School Press, 2006).
- Rothman, Johanna, "Agile Managers: The Essence of Leadership," www.jrothman.com/2010/03/agile-managers-the-essence-of-leadership.

Inserted from <<http://www.jrothman.com/articles/2012/10/building-a-team-through-feedback-2/>>

When is Agile Wrong for You?

by [johanna](#) | May 24, 2016 | [agile](#), [MPD](#) | [5 comments](#)

People often ask me, “When is agile right or not right for a project?” I’ve said before that if the team wants to go agile, that’s great. If the team doesn’t, don’t use agile.

That answer is insufficient. In addition to the team, we need management to not create a bad environment for agile. You might not have a great environment to start. But a bad environment? That’s a horror show.

I had a coaching conversation recently. My client has a typical problem. He sees multiple ways to accomplish the work. He’s taking ideas from agile and lean, and smashing them together to create a project approach that works for them, at their company. It’s not quite agile. And, that’s the sticking point.

His management wants to “go agile.” They have no idea what that means. They think agile is a way to get more good stuff faster with less cost. It’s possible that with agile approaches, they can achieve that as a by-product. To be honest, any approach that stops people from waiting for phases to finish will help. That’s not necessarily agile.

The management team does know about one of the well-known approaches. They want everyone to go through that training. My client doesn’t think this will work. He has a number of concerns:

- Management wants to control how people work at the project level. Management wants to define the iteration duration, what the standup questions will be, who will be on which team, and what the teams will do. (That’s enough right there, but there’s more. They are geographically dispersed across the globe. Going with an out-of-the-box solution does not make sense.)
- Management wants to use team measurements for personal compensation. Specifically, they want to use personal velocity as a way to compensate people. (This is stupid, dangerous and wrong.)
- Every manager my client has spoken with thinks that he or she does not need to change. Only the tech people need to change. (They could not be more mistaken.)

If you work in an agile organization, you know the problems with these assumptions.

Teams manage their own work: their intake is via the Product Owner. They decide how to work, flowing work through the team. Hopefully, the team focuses on their throughput, not who does what.

Remember, [Velocity is Not Acceleration](#). When managers abuse velocity and use it to measure the team members (not even the entire team!), they create schedule games and a toxic team environment. At best, a manager’s abuse of velocity leads to people taking shortcuts and incurring technical debt. At worse, it destroys teamwork.

Managers can create the environment in which people can succeed. Especially in agile and lean, managers do not have to “incent” people, or push people to do well. People will do a good job because they get feedback often and they want to. When managers attempt to manipulate an environment to deliver more with less work (what they think agile is), I’m not sure if anyone can succeed.

I asked my client if the managers understood what agile might mean for them, as managers. He was sure the managers had no idea.

I suggested that trying agile in this environment would give agile a bad name in the organization. I suggested these alternatives:

- Ask about the three questions that might help the managers articulate their goals. See [Define Your Agile Success](#).
- Do a simulation with management to have them feel what agile is like.
- Explain the system of agile and how the ideas that management have is not helpful.
- Request a reasonable environment for a short-ish timebox (I was thinking about a month, maybe two months) to show management that their ideas are not the only ideas that could work. I suggested a number of measures my client could suggest to his management.

Don’t start agile in a toxic environment. It’s not worth it. Agile is wrong for you. Remember that [Agile is Not a Silver Bullet](#) and [Agile is Not for Everyone](#).

Remember, agile is a cultural change, not merely a project management framework.

Instead of agile, consider using all the ideas of agile (for example, teamwork to deliver small chunks of value) to show steady progress and decide how to influence your managers. Don’t ask teams to be

collaborative when management wants to stay command-and-control.

Inserted from <<http://www.jrothman.com/mpd/agile/2016/05/when-is-agile-wrong-for-you/>>

A Working Definition of Agile

by [johanna](#) | May 4, 2016 | [agile](#), [MPD](#) | [8 comments](#)

In a recent workshop, a participant asked me, “What does agile mean? How do you know if you are agile?” He wants to use kanban to see the flow of work through his group. Someone told him he needed to use iterations to be agile. (I had a little rant about this in [What Does Agile Mean to You?](#))

I suggested this could be his working definition:

- You can deliver what you want (some form of value).
- You can deliver that value when you want.
- You can then change to the next most important chunk of valuable work.
- You learn from the previous work you did, both about the work and the process of doing the work.

That’s not all agile is, but it might be a good working definition. If you work towards being able to deliver what and when you want, move to the next thing, and learn, you have the feedback cycles. (You might also look at the [agile principles behind the Manifesto](#).)

These are practices that increase your agile capabilities:

- Iterations, because they limit the work a team can commit to in a given time period.
- Kanban with work in progress limits, because they limit the work a team can do, and show the flow of work.
- Retrospectives because you learn from previous work. (Someone important said if they only did one practice it would be retrospectives. I can’t remember who said that. Sorry.)
- Standups because they reinforce micro-commitments to finishing work.
- Technical excellence practices from XP, because they make changing the code and tests easier.

You don’t need any of these to be agile. They help. You might find other practices to be more helpful in your context.

I have some previous posts that might be interesting if you also are wondering what agile means for you:

- [Self Assessment Tool for Agile Maturity](#)
- [Agile is Not a Silver Bullet](#)
- [Agile is Not for Everyone](#)
- Agile is about cultural change. See [Stuck in the Middle of Your Agile Transformation, Part 3](#) for a series I wrote earlier this year.

For me, practices are interesting, especially if I choose to experiment with them. What could I do to increase my throughput and learning, and therefore, my ability to adapt? Agile is not about specific practices. It is about a mindset of finishing small valuable chunks, feedback, and change.

Inserted from <<http://www.jrothman.com/mpd/agile/2016/05/a-working-definition-of-agile/>>

Resource Management is the Wrong Idea; Manage Your Project Portfolio Instead

by [johanna](#) | Sep 6, 2016 | [MPD](#), [portfolio management](#) | [4 comments](#)

I had the chance to teach agile ideas to some project portfolio managers, or people who were considering setting up a PMO to manage the project portfolio.

Many of these folks had a real concern: how to manage “resources.” (People to you and me.) Part of what I said to them was that resource management was the wrong idea. Instead of flowing work through people, optimize at a higher level and flow projects through teams.

They were surprised.

Part of their surprise is that for too long, we have treated projects (in any industry) as piece work. We

parcel out the pieces to the expert. We wait for the expert to complete the work and then parcel out the work to the next expert. See my posts about [resource vs. flow efficiency](#) earlier this year.

If you want to implement strategy through the project portfolio, you have to select the right projects—at least for now—to start and finish. That means you stop thinking about “resource” management and start thinking about project portfolio management and how you see the flow of value through the project for the organization. You see more finished work and less incomplete work. You see the value that finished projects provide, instead of much work in progress.



Optimizing for a given person’s contributions make little sense for many organizations. People are too interdependent on one another to finish projects.

Instead, optimize for the project’s contribution to the organization. Manage your project portfolio.

Inserted from <<http://www.jrothman.com/mpd/portfolio-management/2016/09/resource-management-is-the-wrong-idea-manage-your-project-portfolio-instead/>>

Agile – Classroom session

Friday, March 31, 2017 2:30 PM

Subject	Agile – Classroom session
From	Corporate Training
To	Corporate Training
Cc	Pankaj N Parmar; Ghiridharan Surendran; Naveen Nemani; Kranthi Popuri; Pavani Maddali; Jyothy Heda
Sent	Friday, March 31, 2017 12:57 PM
Attachments	 Guidelines for Regist...  List of Participants



	<p>Subject: Agile– Classroom session</p> <p>Audience: QA Analyst, Software Engineers, AMs & Managers</p>	
	<p>You are Invited to the session Agile. This is a 2 day workshop that will cover theoretical and practical aspects of Agile.</p> <p>Course Content</p> <ul style="list-style-type: none">+ Introduction to Agile+ Agile Foundation+ Introduction to Scrum+ Scrum Framework- Roles+ Agile planning - Project level planning	

- + Agile planning - Release level planning
- + Agile planning - Release level planning
- + Scrum Framework – Sprint planning
- + Scrum Framework – Daily scrum
- + Other sprint activities
- + Tracking and monitoring a sprint
- + Scrum Framework –Sprint review
- + Scrum Framework –Sprint retrospective
- + Scalability of Scrum
- + Conclusion

Session details

Training Date	Time	Venue
6 th & 7 th April, 2017	10:00 AM to 6:00 PM	1107-Divyasree

Other Important details

- + Please check guidelines document attached
- + We do not encourage cancellation, however, in case of emergency or pressing business requirement please follow the attached guidelines document for

Instructions for Class room session Registration:

- + For all training programs in the general shift which **end by 4 PM** - there is no need for employees (FDS -1) **to change their shift timing.**

- + Corporate Training will not trigger transport RPD for **shift change**. If any training program extends beyond 4 PM – Transport will be

- + For employees attending training in a different shift: **Transport will be arranged**; In case Transport is not provided , Corporate Training will de-register the participants, They need not attend the training and can continue with

For queries regarding the course, please contact [Sunil](#).

Raise an RPD to [Surya Pratibha](#) in case of any LMS Support.

Attachments: Participants' list and participation guidelines.

BCC: Participants and their reporting managers.



[Contact Us](#)



[Reach us
on
Undergroun](#)



[Join us on
Socialcast](#)

If Humble People Make the Best Leaders, Why Do We Fall for Charismatic Narcissists?

Saturday, April 15, 2017 3:10 AM



The [research](#) is clear: when we choose humble, unassuming people as our leaders, the world around us becomes a better place.

Humble leaders improve the performance of a company in the long run because they create more collaborative environments. They have a balanced view of themselves— both their virtues and shortcomings – and a strong appreciation of others' strengths and contributions, while being open to new ideas and feedback. These "unsung heroes" help their believers to build their self-esteem, go beyond their expectations, and create a community that channels individual efforts into an organized group that works for the good of the collective.

For example, [one study](#) examined 105 small-to-medium-sized companies in the computer software and hardware industry in the United States. The findings revealed that when a humble CEO is at the helm of a firm, its top management team is more likely to collaborate and share information, making the most of the firm's talent.

Another study showed that a leader's humility can be [contagious](#): when leaders behave humbly, followers emulate their modest attitude and behavior. A study of 161 teams found that employees following humble leaders were themselves more likely to admit their mistakes and limitations, share the spotlight by deflecting praise to others, and be open to new ideas, advice, and feedback.

Yet instead of following the lead of these unsung heroes, we appear hardwired to search for superheroes: over-glorifying leaders who exude charisma.

The Greek word *Kharisma* means "divine gift," and charisma is the quality of extraordinary charm, magnetism, and presence that makes a person capable of inspiring others with enthusiasm and devotion. [German sociologist](#) Max Weber defined charisma as "of divine origin or as exemplary, and on the basis of it, the individual concerned is treated as a leader." [Research evidence](#) on charismatic leadership reveals that charismatic people are more likely to become endorsed as leaders because of their high energy, unconventional behavior, and heroic deeds.

While charisma is conducive to orchestrating positive large-scale transformations, there can be a "dark side" to charismatic leadership. Jay Conger and Rabintra Kanungo describe it this way in their seminal [book](#): "Charismatic leaders can be prone to extreme narcissism that leads them to promote highly self-serving and grandiose aims." A [clinical study](#) illustrates that when charisma overlaps with narcissism, leaders tend to abuse their power and take advantage of their followers. Another [study](#) indicates that narcissistic leaders tend to present a bold vision of the future, and this makes them more charismatic in the eyes of others.

Why are such leaders more likely to rise to the top? [One study](#) suggests that despite being perceived as arrogant, narcissistic individuals radiate "an image of a prototypically effective leader." Narcissistic leaders know how to draw attention toward themselves. They enjoy the visibility. It takes time for people to see that these early signals of competence are not later

realized, and that a leader's narcissism reduces the exchange of information among team members and often negatively affects group performance.

It's not that charismatic and narcissistic people can't ever make good leaders. In some circumstances, they can. For example, one study [found that](#) narcissistic CEOs "favor bold actions that attract attention, resulting in big wins or big losses." A narcissistic leader thus can represent a high-risk, high-reward proposition.

And it's not that humble leaders can't ever be charismatic. Researchers agree that we could classify charismatic leaders as "negative" or "positive" by their orientation toward pursuing their self-interested goals versus those of their groups. These two sides of charismatic leadership have also been called [personalized and socialized charisma](#). Although the *socialized* charismatic leader has the aura of a hero, it is counteracted with low authoritarianism and a genuine interest in the collective welfare. In contrast, the *personalized* charismatic leader's perceived heroism is coupled with high authoritarianism and high narcissism. It is when followers are confused and disoriented that they are more likely to form personalized relationships with a charismatic leader. Socialized relationships, on the other hand, are established by followers with a clear set of values who view the charismatic leader as a means to achieve collective action.

The problem is that we select negative charismatic leaders much more frequently than in the limited situations where the risk they represent might pay off. Despite their grandiose view of themselves, low empathy, dominant orientation toward others, and strong sense of entitlement, their charisma proves irresistible. Followers of superheroes are enthralled by their showmanship: through their sheer magnetism, narcissistic leaders transform their environments into a competitive game in which their followers also become more self-centered, giving rise to organizational narcissism, as [one study](#) shows.

If humble leaders are more effective than narcissistic leaders, why do we so often choose narcissistic individuals to lead us?

The "[romance of leadership](#)" hypothesis suggests that we generally have a biased tendency to understand social events in terms of leadership and people tend to romanticize the figure of the leader.

[My own research](#) shows that our psychological states can also bias our perceptions of charismatic leaders. High levels of anxiety make us hungry for charisma. As a result, crises increase not only the search for charismatic leaders, but also our tendency to *perceive* charisma in the leaders we already follow.


Economic and social crises thus become a unique testing ground for charismatic leaders. They create conditions of distress and uncertainty that appear to be ideal for the ascent of charismatic figures. Yet at the same time, they also make us more vulnerable to choosing the wrong leader. Crises and other emotionally laden events increase our propensity to romanticize the grandiose view of narcissistic leaders. The paradox is that we may then choose to support the very leaders who are less likely to bring us success. In a time of crisis, it's easy to be seduced by superheroes who could come and "rescue" us, but who possibly then plunge us into greater peril.

While this may sound hopeless, there is another way of looking at it. Essentially, we have the leaders we deserve. As we collectively select and construct our leaders to satisfy our own needs and desires, we can choose humility or socialized charisma over narcissism.

From https://hbr.org/2017/04/if-humble-people-make-the-best-leaders-why-do-we-fall-for-charismatic-narcissists?utm_campaign=hbr&utm_source=linkedin&utm_medium=social

SOAP UI Training

Wednesday, August 30, 2017 4:08 PM

Subject	RE: My team wants to join SOAP UI training sessions
From	Shiva Kumar Macharla
To	Eyme Ann Mathew; Bhupender Agarwal; QAAutomation TLs
Cc	Mustaq M. Mohammed
Sent	Wednesday, August 30, 2017 3:22 PM
Attachments	 RE SOAP UI recorded ...

Hello All,

Ramya has shared the recorded sessions and references of SOAP UI with me. Please find the same in the attachment.

Thanks,
Shiva

From: Eyme Ann Mathew
Sent: Tuesday, August 29, 2017 6:22 PM
To: Shiva Kumar Macharla <smacharla@factset.com>; Bhupender Agarwal <bhagarwal@factset.com>; QAAutomation TLs <qaautomation_tls@factset.com>
Cc: Mustaq M. Mohammed <MMMohammed@factset.com>
Subject: RE: My team wants to join SOAP UI training sessions

Shiva, please reach out to Ramya to see if she has the recordings.

-Eyme

From: Shiva Kumar Macharla
Sent: Tuesday, August 29, 2017 6:14 PM
To: Bhupender Agarwal <bhagarwal@factset.com>; QAAutomation TLs <qaautomation_tls@factset.com>
Cc: Eyme Ann Mathew <emathew@factset.com>; Mustaq M. Mohammed <MMMohammed@factset.com>
Subject: RE: My team wants to join SOAP UI training sessions

Thanks Bhupender for reminding me that we have sessions recorded already.
@Eyme/@Bhupender – Can you please share if you have the location of recorded sessions, so my team can listen to them before we attend any live session?

-Shiva

From: Bhupender Agarwal
Sent: Tuesday, August 29, 2017 6:04 PM
To: Shiva Kumar Macharla <smacharla@factset.com>; QAAutomation TLs

<qaautomation_tls@factset.com>

Cc: Eyme Ann Mathew <emathew@factset.com>; Mustaq M. Mohammed

<MMMohammed@factset.com>

Subject: RE: My team wants to join SOAP UI training sessions

Hi Shiva,

Currently, there is no training scheduled within my team for Soap UI. But if you happen to find any, please have the session recorded so that we can listen to it later (I know that we have few sessions already recorded).

Thanks,

Bhupender Agarwal

Senior Team Leader, QA Automation

FactSet Systems India Pvt. Ltd.

Work: +91 91 00 931356

Cell: +91 90 30 337694

bhagarwal@factset.com

www.factset.com

FACTSET › SEE THE ADVANTAGE

FINANCIAL DATA | ANALYTICS | TECHNOLOGY | SERVICES

FactSet is an honoree of *Fortune's* [100 Best Companies to Work For](#) and a Best Workplace Award recipient in the [United Kingdom](#) and [France](#).

From: Shiva Kumar Macharla

Sent: Tuesday, August 29, 2017 4:39 PM

To: QAAutomation TLs <qaautomation_tls@factset.com>

Cc: Eyme Ann Mathew <emathew@factset.com>; Mustaq M. Mohammed

<MMMohammed@factset.com>

Subject: My team wants to join SOAP UI training sessions

Hello TLs,

My team has a project which makes use of SOAP UI and we want to join the training sessions. So, please let me know if any training session is planned/going on in your team for SOPA UI.

Thanks,

Shiva Kumar Macharla

QA Automation Analyst II

FactSet

smacharla@factset.com

www.factset.com

FACTSET › SEE THE ADVANTAGE

FINANCIAL DATA | ANALYTICS | TECHNOLOGY | SERVICES

Unlock honest feedback with this one word

Wednesday, September 27, 2017 3:13 PM

Learn more here: <https://knowyourcompany.com/learn/>
<https://m.signalnoise.com/>

Unlock honest feedback with this one word

This one little word makes all the difference...



``



A few years ago, a CEO told me how she was struggling to get honest feedback from her board. No one seemed willing to be critical or give her pointers on things she could improve. After every board meeting, she would turn to them and ask directly:

“What feedback does anyone have for me?”

She’d hear crickets. Every single time.

No one would speak up. Even though they were board members — people who are supposed to hold *her* accountable as the CEO of the company — they shied away from offering their honest input.

This was so perplexing to the CEO. She felt like she was being very clear with what she wanted... Why weren’t they just giving her the feedback she was asking for?

One day, she decided to try something different.

Instead of asking, “What feedback does anyone have for me?” ... she asked this:

“What advice does anyone have for me?”

All of sudden, everyone started weighing in. “*Well I might try this...*” and “*The way you brought up this point could’ve been better...*” and “*You could try structuring the meeting like this...*”

The word “advice” unlocked all the honest feedback that CEO needed.

Why? The word “feedback” carries a lot of baggage. To some, they automatically associate it with a “critique” or something negative. It can seem scary and formal.

But “advice” is a much more welcoming word. Advice is about lending someone a hand. When someone gives you advice, they’re just looking out for you.

And when you ask for advice, it’s an invitation. You’re signaling that another person has expertise or knowledge that you find interesting and valuable. That person is often flattered you even asked for advice in the first place.

Who doesn’t love to give advice? :-)

The next time you’d like to get honest feedback, try asking for advice instead. Notice how much more people open up to you. See how swapping that one word makes a difference.

Inserted from <https://m.signalnoise.com/unlock-honest-feedback-with-this-one-word-dcaf3839e7ee>

The Set-Up-To-Fail Syndrome

Thursday, November 16, 2017 4:56 PM

[Organizational culture](#)

The Set-Up-To-Fail Syndrome

- [Jean-Francois Manzoni](#)
- [Jean-Louis Barsoux](#)

From the March–April 1998 Issue

- [SAVE](#)
- [SHARE](#)
- [COMMENT](#)
- [TEXT SIZE](#)
- [PRINT](#)
- [PDF](#)
- [8.95 BUY COPIES](#)

View more from the

[March–April 1998 Issue](#)

THE POLITICS OF E-MAIL		PAGE 12
Harvard Business Review		
MARCH–APRIL 1998		
		
Kathleen M. Eisenhardt and Shona I. Brown	TIME PACING: COMPETING IN MARKETS THAT WON'T STAND STILL	59
Jean Magnette	THE POWER OF VIRTUAL INTEGRATION: AN INTERVIEW WITH DELL COMPUTER'S MICHAEL DELL	72
Richard K. Lester, Michael J. Piore, and Kamal M. Malik	INTERPRETIVE MANAGEMENT: WHAT GENERAL MANAGERS CAN LEARN FROM DESIGN	86
Jean-Francois Manzoni and Jean-Louis Barsoux	THE SET-UP-TO-FAIL SYNDROME	101
Joseph I. Badaracco, Jr.	THE DISCIPLINE OF BUILDING CHARACTER	114
SHIKHAR CHOSH	MAKING BUSINESS SENSE OF THE INTERNET	126
ROBERT GILFORD	HBR CASE STUDY WHY DOESN'T THIS HR DEPARTMENT GET ANY RESPECT?	24
PAUL SHARPE AND TOM KEELIN	IDEAS AT WORK: HOW SMITHKLINE BEECHAM MAKES BETTER RESOURCE-ALLOCATION DECISIONS	45
JOHN S. HAMMOND, RALPH L. KEENEY, AND HOWARD RAIFEA	MANAGER'S TOOL KIT EVEN SWAPS: A RATIONAL METHOD FOR MAKING TRADE-OFFS	137
JEFFREY ELTON AND JUSTIN ROE	BOOKS IN REVIEW BRINGING DISCIPLINE TO PROJECT MANAGEMENT	153
BRIEFINGS 12	LETTERS 160	EXECUTIVE SUMMARIES 177


[Explore the Archive](#)

Recommended

→ SMARTER THAN THE AVERAGE GUIDE

HBR Guide to Building Your Business Case

Pinpoint the need
Crunch the numbers
Pitch your idea



DOC

XLS

PPT

EBOOK + TOOLS

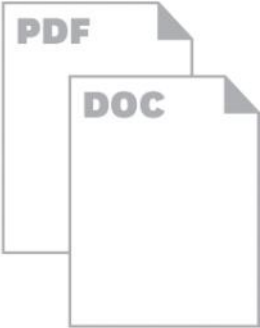
[HBR Guide to Building Your Business Case...](#)

Communication Book

59.95 [Add to Cart](#)

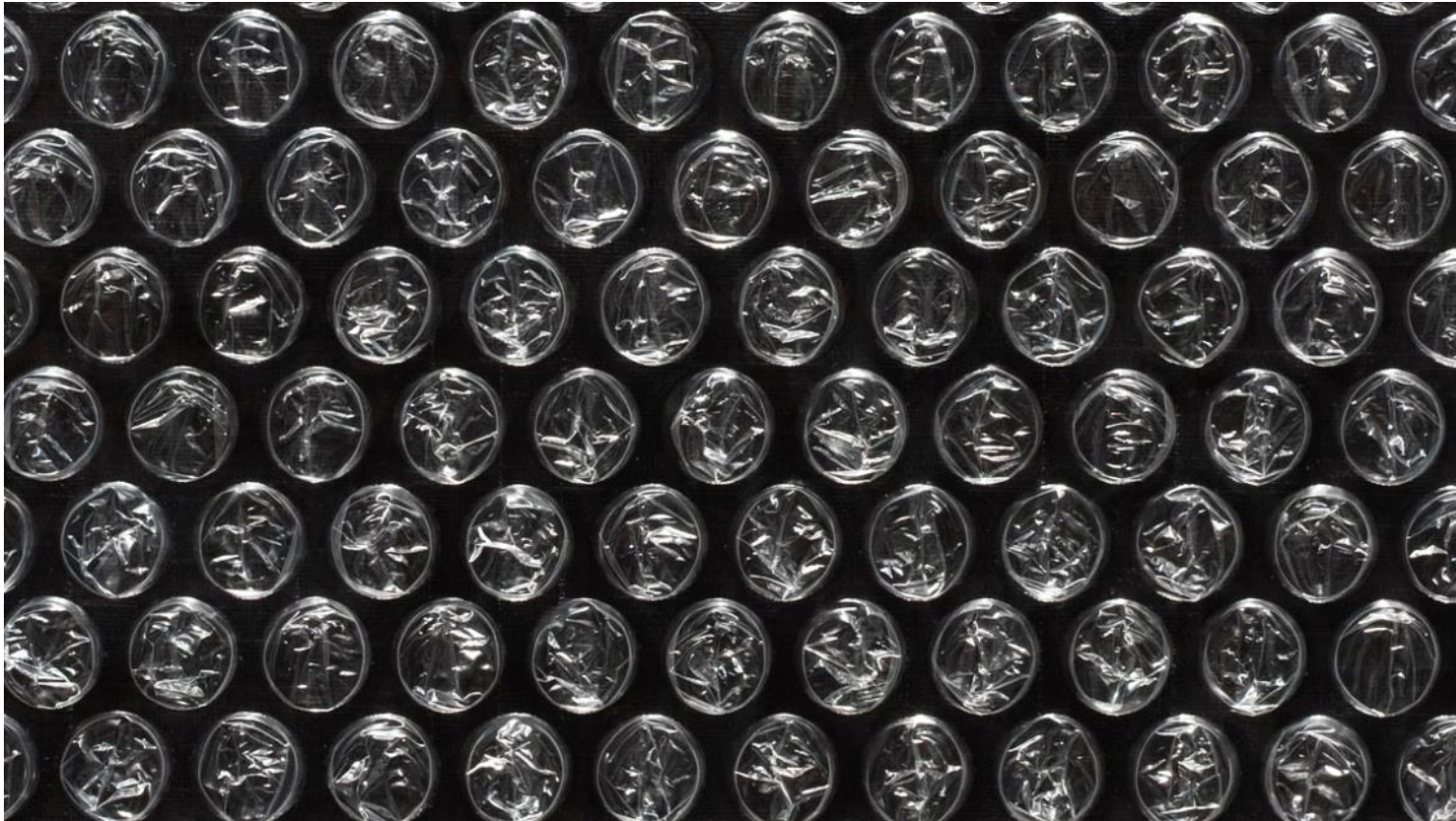


[What Makes a Leader? \(HBR Bestseller\)](#)
Leadership & Managing People HBR Article
8.95 [Add to Cart](#)



EBOOK + TOOLS

[HBR Guide to Making Every Meeting Matter...](#)
Organizational Development Book
69.95 [Add to Cart](#)



When an employee fails—or even just performs poorly—managers typically do not blame themselves. The employee doesn't understand the work, a manager might contend. Or the employee isn't driven to succeed, can't set priorities, or won't take direction. Whatever the reason, the problem is assumed to be the employee's fault—and the employee's responsibility.

But is it? Sometimes, of course, the answer is yes. Some employees are not up to their assigned tasks and never will be, for lack of knowledge, skill, or simple desire. But sometimes—and we would venture to say often—an employee's poor performance can be blamed largely on his boss.

Perhaps "blamed" is too strong a word, but it is directionally correct. In fact, our research strongly suggests that bosses—albeit accidentally and usually with the best intentions—are often complicit in an employee's lack of success. (See the insert "About the Research.") How? By creating and reinforcing a dynamic that essentially sets up perceived underperformers to fail. If the Pygmalion effect describes the dynamic in which an individual lives up to great expectations, the set-up-to-fail syndrome explains the opposite. It describes a dynamic in which employees perceived to be mediocre or weak performers live down to the low expectations their managers have for them. The result is that they often end up leaving the organization—either of their own volition or not.

About the Research

This article is based on two studies designed to understand better the causal relationship between leadership style and subordinate performance—in other words, to explore how bosses and subordinates mutually influence each other's behavior. The first study, which comprised surveys, interviews, and observations, involved 50 boss-subordinate pairs in four manufacturing operations in *Fortune* 100 companies. The second study, involving an informal survey of about 850 senior managers attending INSEAD executive-development programs over the last three years, was done to test and refine the findings generated by the first study. The executives in the second study represented a wide diversity of nationalities, industries, and personal backgrounds.

Read more

The syndrome usually begins surreptitiously. The initial impetus can be performance related, such as when an employee loses a client, undershoots a target, or misses a deadline. Often, however, the trigger is less specific. An employee is transferred into a division with a lukewarm recommendation from a previous boss. Or perhaps the boss and the employee don't really get along on a personal basis—several studies have indeed shown that compatibility between boss and subordinate, based on similarity of attitudes, values, or social characteristics, can have a significant impact on a boss's impressions. In any case, the syndrome is set in motion when the boss begins to worry that the employee's performance is not up to par.

The boss then takes what seems like the obvious action in light of the subordinate's perceived shortcomings: he increases the time and attention he focuses on the employee. He requires the employee to get approval before making decisions, asks to see more paperwork documenting those decisions, or watches the employee at meetings more closely and critiques his comments more intensely.

These actions are intended to boost performance and prevent the subordinate from making errors. Unfortunately, however, subordinates often interpret the heightened supervision as a lack of trust and confidence. In time, because of low expectations, they come to doubt their own thinking and ability, and they lose the motivation to make autonomous decisions or to take any action at all. The boss, they figure, will just question everything they do—or do it himself anyway.

Ironically, the boss sees the subordinate's withdrawal as proof that the subordinate is indeed a poor performer. The subordinate, after all, isn't contributing his ideas or energy to the organization. So what does the boss do? He increases his pressure and supervision again—watching, questioning, and double-

checking everything the subordinate does. Eventually, the subordinate gives up on his dreams of making a meaningful contribution. Boss and subordinate typically settle into a routine that is not really satisfactory but, aside from periodic clashes, is otherwise bearable for them. In the worst-case scenario, the boss's intense intervention and scrutiny end up paralyzing the employee into inaction and consume so much of the boss's time that the employee quits or is fired. (For an illustration of the set-up-to-fail syndrome, see the exhibit "The Set-Up-to-Fail Syndrome: No Harm Intended—A Relationship Spirals from Bad to Worse.")

THE SET-UP-TO-FAIL SYNDROME

NO HARM INTENDED – A RELATIONSHIP SPIRALS FROM BAD TO WORSE



1 Before the set-up-to-fail syndrome begins, the boss and the subordinate are typically engaged in a positive, or at least neutral, relationship.



2 The triggering event in the set-up-to-fail syndrome is often minor or surreptitious. The subordinate may miss a deadline, lose a client, or submit a subpar report. In other cases, the syndrome's genesis is the boss, who distances himself from the subordinate for personal or social reasons unrelated to performance.



3 Reacting to the triggering event, the boss increases his supervision of the subordinate, gives more specific instructions, and wrangles longer over courses of action.



4 The subordinate responds by beginning to suspect a lack of confidence and senses he's not part of the boss's in-group anymore. He starts to withdraw emotionally from the boss and from work. He may also fight to change the boss's image of him, reaching too high or running too fast to be effective.



5 The boss interprets this problem-hoarding, overreaching, or tentativeness as signs that the subordinate has poor judgment and weak capabilities. If the subordinate does perform well, the boss does not acknowledge it or considers it a lucky "one off." He limits the subordinate's discretion, withholds social contact, and shows, with increasing openness, his lack of confidence in and frustration with the subordinate.



6 The subordinate feels boxed in and under-appreciated. He increasingly withdraws from his boss and from work. He may even resort to ignoring instructions, openly disputing the boss, and occasionally lashing out because of feelings of rejection. In general, he performs his job mechanically and devotes more energy to self-protection. Moreover, he refers all nonroutine decisions to the boss or avoids contact with him.



7 The boss feels increasingly frustrated and is now convinced that the subordinate cannot perform without intense oversight. He makes this known by his words and deeds, further undermining the subordinate's confidence and prompting inaction.



8 When the set-up-to-fail syndrome is in full swing, the boss pressures and controls the subordinate during interactions. Otherwise, he avoids contact and gives the subordinate routine assignments only. For his part, the subordinate shuts down or leaves, either in dismay, frustration, or anger.

Find this and other HBR graphics in our [Visual Library](#)

Perhaps the most daunting aspect of the set-up-to-fail syndrome is that it is self-fulfilling and self-reinforcing—it is the quintessential vicious circle. The process is self-fulfilling because the boss's actions contribute to the very behavior that is expected from weak performers. It is self-reinforcing because the boss's low expectations, in being fulfilled by his subordinates, trigger more of the same behavior on his part, which in turn triggers more of the same behavior on the part of subordinates. And on and on, unintentionally, the relationship spirals downward.

A case in point is the story of Steve, a manufacturing supervisor for a *Fortune* 100 company. When we first met Steve, he came across as highly motivated, energetic, and enterprising. He was on top of his

operation, monitoring problems and addressing them quickly. His boss expressed great confidence in him and gave him an excellent performance rating. Because of his high performance, Steve was chosen to lead a new production line considered essential to the plant's future.

In his new job, Steve reported to Jeff, who had just been promoted to a senior management position at the plant. In the first few weeks of the relationship, Jeff periodically asked Steve to write up short analyses of significant quality-control rejections. Although Jeff didn't really explain this to Steve at the time, his request had two major objectives: to generate information that would help both of them learn the new production process, and to help Steve develop the habit of systematically performing root cause analysis of quality-related problems. Also, being new on the job himself, Jeff wanted to show his own boss that he was on top of the operation.

Unaware of Jeff's motives, Steve balked. Why, he wondered, should he submit reports on information he understood and monitored himself? Partly due to lack of time, partly in response to what he considered interference from his boss, Steve invested little energy in the reports. Their tardiness and below-average quality annoyed Jeff, who began to suspect that Steve was not a particularly proactive manager. When he asked for the reports again, he was more forceful. For Steve, this merely confirmed that Jeff did not trust him. He withdrew more and more from interaction with him, meeting his demands with increased passive resistance. Before long, Jeff became convinced that Steve was not effective enough and couldn't handle his job without help. He started to supervise Steve's every move—to Steve's predictable dismay. One year after excitedly taking on the new production line, Steve was so dispirited he was thinking of quitting.

How can managers break the set-up-to-fail syndrome? Before answering that question, let's take a closer look at the dynamics that set the syndrome in motion and keep it going.

Deconstructing the Syndrome

We said earlier that the set-up-to-fail syndrome usually starts surreptitiously—that is, it is a dynamic that usually creeps up on the boss and the subordinate until suddenly both of them realize that the relationship has gone sour. But underlying the syndrome are several assumptions about weaker performers that bosses appear to accept uniformly. Our research shows, in fact, that executives typically compare weaker performers with stronger performers using the following descriptors:

- less motivated, less energetic, and less likely to go beyond the call of duty;
- more passive when it comes to taking charge of problems or projects;
- less aggressive about anticipating problems;
- less innovative and less likely to suggest ideas;
- more parochial in their vision and strategic perspective;
- more prone to hoard information and assert their authority, making them poor bosses to their own subordinates.

Up to 90% of all bosses treat some subordinates as though they were part of an in-group, while they consign others to an out-group.

It is not surprising that on the basis of these assumptions, bosses tend to treat weaker and stronger performers very differently. Indeed, numerous studies have shown that up to 90% of all managers treat some subordinates as though they were members of an in-group, while they consign others to membership in an out-group. Members of the in-group are considered the trusted collaborators and therefore receive more autonomy, feedback, and expressions of confidence from their bosses. The boss-subordinate relationship for this group is one of mutual trust and reciprocal influence. Members of the out-group, on the other hand, are regarded more as hired hands and are managed in a more formal, less personal way, with more emphasis on rules, policies, and authority. (For more on how bosses treat weaker and stronger performers differently, see the chart “In with the In Crowd, Out with the Out.”)

IN WITH THE IN CROWD, OUT WITH THE OUT	
Boss's behavior toward perceived stronger performers	Boss's behavior toward perceived weaker performers
Discusses project objectives, with a limited focus on project implementation. Gives subordinate the freedom to choose his own approach to solving problems or reaching goals.	Is directive when discussing tasks and goals. Focuses on what needs get done as well as how it should get done.
Treats unfavorable variances, mistakes, or incorrect judgments as learning opportunities.	Pays close attention to unfavorable variances, mistakes, or incorrect judgments.
Makes himself available, as in "Let me know if I can help." Initiates casual and personal conversations.	Makes himself available to subordinate on a need-to-see basis. Bases conversations primarily on work-related topics.
Is open to subordinate's suggestions and discusses them with interest.	Pays little interest to subordinate's comments or suggestions about how and why work is done.
Gives subordinate interesting and challenging stretch assignments. Often allows subordinate to choose his own assignments.	Reluctantly gives subordinate anything but routine assignments. When handing out assignments, gives subordinate little choice. Monitors subordinate heavily.
Solicits opinions from subordinate on organizational strategy, execution, policy, and procedures.	Rarely asks subordinate for input about organizational or work-related matters.
Often defers to subordinate's opinion in disagreements.	Usually imposes own views in disagreements.
Praises subordinate for work well done.	Emphasizes what the subordinate is doing poorly.

Why do managers categorize subordinates into either in-groups or out-groups? For the same reason that we tend to typecast our family, friends, and acquaintances: it makes life easier. Labeling is something we all do, because it allows us to function more efficiently. It saves time by providing rough-and-ready guides for interpreting events and interacting with others. Managers, for instance, use categorical thinking to figure out quickly who should get what tasks. That's the good news.

The downside of categorical thinking is that in organizations it leads to premature closure. Having made up his mind about a subordinate's limited ability and poor motivation, a manager is likely to notice supporting evidence while selectively dismissing contrary evidence. (For example, a manager might interpret a terrific new product idea from an out-group subordinate as a lucky onetime event.)

Unfortunately for some subordinates, several studies show that bosses tend to make decisions about in-groups and out-groups even as early as five days into their relationships with employees.

Are bosses aware of this sorting process and of their different approaches to "in" and "out" employees? Definitely. In fact, the bosses we have studied, regardless of nationality, company, or personal

background, were usually quite conscious of behaving in a more controlling way with perceived weaker performers. Some of them preferred to label this approach as “supportive and helpful.” Many of them also acknowledged that—although they tried not to—they tended to become impatient with weaker performers more easily than with stronger performers. By and large, however, managers are aware of the controlling nature of their behavior toward perceived weaker performers. For them, this behavior is not an error in implementation; it is intentional.

What bosses typically do *not* realize is that their tight controls end up hurting subordinates’ performance by undermining their motivation in two ways: first, by depriving subordinates of autonomy on the job and, second, by making them feel undervalued. Tight controls are an indication that the boss assumes the subordinate can’t perform well without strict guidelines. When the subordinate senses these low expectations, it can undermine his self-confidence. This is particularly problematic because numerous studies confirm that people perform up or down to the levels their bosses expect from them or, indeed, to the levels they expect from themselves.¹

What bosses do not realize is that their tight controls end up hurting subordinates’ performance by undermining their motivation.

Of course, executives often tell us, “Oh, but I’m very careful about this issue of expectations. I exert more control over my underperformers, but I make sure that it does not come across as a lack of trust or confidence in their ability.” We believe what these executives tell us. That is, we believe that they do try hard to disguise their intentions. When we talk to their subordinates, however, we find that these efforts are for the most part futile. In fact, our research shows that most employees can—and do—“read their boss’s mind.” In particular, they know full well whether they fit into their boss’s in-group or out-group. All they have to do is compare how they are treated with how their more highly regarded colleagues are treated.

Just as the boss’s assumptions about weaker performers and the right way to manage them explains his complicity in the set-up-to-fail syndrome, the subordinate’s assumptions about what the boss is thinking explain his own complicity. The reason? When people perceive disapproval, criticism, or simply a lack of confidence and appreciation, they tend to shut down—a behavioral phenomenon that manifests itself in several ways.

Primarily, shutting down means disconnecting intellectually and emotionally. Subordinates simply stop giving their best. They grow tired of being overruled, and they lose the will to fight for their ideas. As one subordinate put it, “My boss tells me how to execute every detail. Rather than arguing with him, I’ve ended up wanting to say, ‘Come on, just tell me what you want me to do, and I’ll go do it.’ You become a robot.” Another perceived weak performer explained, “When my boss tells me to do something, I just do it mechanically.”

Shutting down also involves disengaging personally—essentially reducing contact with the boss. Partly, this disengagement is motivated by the nature of previous exchanges that have tended to be negative in tone. As one subordinate admitted, “I used to initiate much more contact with my boss until the only thing I received was negative feedback; then I started shying away.”

Besides the risk of a negative reaction, perceived weaker performers are concerned with not tainting their images further. Following the often-heard aphorism “Better to keep quiet and look like a fool than to open your mouth and prove it,” they avoid asking for help for fear of further exposing their limitations. They also tend to volunteer less information—a simple “heads up” from a perceived underperformer can cause the boss to overreact and jump into action when none is required. As one perceived weak performer recalled, “I just wanted to let my boss know about a small matter, only slightly out of the routine, but as soon as I mentioned it, he was all over my case. I should have kept my mouth closed. I do now.”

Finally, shutting down can mean becoming defensive. Many perceived underperformers start devoting more energy to self-justification. Anticipating that they will be personally blamed for failures, they seek to find excuses early. They end up spending a lot of time looking in the rearview mirror and less time looking at the road ahead. In some cases—as in the case of Steve, the manufacturing supervisor described earlier—this defensiveness can lead to noncompliance or even systematic opposition to the boss’s views. While this idea of a weak subordinate going head to head with his boss may seem irrational, it may reflect what Albert Camus once observed: “When deprived of choice, the only freedom left is the freedom to say no.”

The Syndrome Is Costly

There are two obvious costs of the set-up-to-fail syndrome: the emotional cost paid by the subordinate and the organizational cost associated with the company’s failure to get the best out of an employee. Yet there are other costs to consider, some of them indirect and long term.

The boss pays for the syndrome in several ways. First, uneasy relationships with perceived low performers often sap the boss’s emotional and physical energy. It can be quite a strain to keep up a facade of courtesy and pretend everything is fine when both parties know it is not. In addition, the energy devoted to trying to fix these relationships or improve the subordinate’s performance through increased supervision prevents the boss from attending to other activities—which often frustrates or even angers the boss.

Furthermore, the syndrome can take its toll on the boss’s reputation, as other employees in the organization observe his behavior toward weaker performers. If the boss’s treatment of a subordinate is deemed unfair or unsupportive, observers will be quick to draw their lessons. One outstanding performer commented on his boss’s controlling and hypercritical behavior toward another subordinate: “It made us all feel like we’re expendable.” As organizations increasingly espouse the virtues of learning and empowerment, managers must cultivate their reputations as coaches, as well as get results. One strong performer said of his boss’s hypercritical behavior toward another employee: “It made us all feel like we’re expendable.”

The set-up-to-fail syndrome also has serious consequences for any team. A lack of faith in perceived weaker performers can tempt bosses to overload those whom they consider superior performers; bosses want to entrust critical assignments to those who can be counted on to deliver reliably and quickly and to those who will go beyond the call of duty because of their strong sense of shared fate. As one boss half-jokingly said, “Rule number one: if you want something done, give it to someone who’s busy—there’s a reason why that person is busy.”

An increased workload may help perceived superior performers learn to manage their time better, especially as they start to delegate to their own subordinates more effectively. In many cases, however, these performers simply absorb the greater load and higher stress which, over time, takes a personal toll and decreases the attention they can devote to other dimensions of their jobs, particularly those yielding longer-term benefits. In the worst-case scenario, overburdening strong performers can lead to burnout.

Team spirit can also suffer from the progressive alienation of one or more perceived low performers. Great teams share a sense of enthusiasm and commitment to a common mission. Even when members of the boss's out-group try to keep their pain to themselves, other team members feel the strain. One manager recalled the discomfort experienced by the whole team as they watched their boss grill one of their peers every week. As he explained, "A team is like a functioning organism. If one member is suffering, the whole team feels that pain."

In addition, alienated subordinates often do not keep their suffering to themselves. In the corridors or over lunch, they seek out sympathetic ears to vent their recriminations and complaints, not only wasting their own time but also pulling their colleagues away from productive work. Instead of focusing on the team's mission, valuable time and energy is diverted to the discussion of internal politics and dynamics. Finally, the set-up-to-fail syndrome has consequences for the subordinates of the perceived weak performers. Consider the weakest kid in the school yard who gets pummeled by a bully. The abused child often goes home and pummels his smaller, weaker siblings. So it is with the people who are in the boss's out-group. When they have to manage their own employees, they frequently replicate the behavior that their bosses show to them. They fail to recognize good results or, more often, supervise their employees excessively.

Breaking Out Is Hard to Do

The set-up-to-fail syndrome is not irreversible. Subordinates can break out of it, but we have found that to be rare. The subordinate must consistently deliver such superior results that the boss is forced to change the employee from out-group to in-group status—a phenomenon made difficult by the context in which these subordinates operate. It is hard for subordinates to impress their bosses when they must work on unchallenging tasks, with no autonomy and limited resources; it is also hard for them to persist and maintain high standards when they receive little encouragement from their bosses.

Furthermore, even if the subordinate achieves better results, it may take some time for them to register with the boss because of his selective observation and recall. Indeed, research shows that bosses tend to attribute the good things that happen to weaker performers to external factors rather than to their efforts and ability (while the opposite is true for perceived high performers: successes tend to be seen as theirs, and failures tend to be attributed to external uncontrollable factors). The subordinate will therefore need to achieve a string of successes in order to have the boss even contemplate revising the initial categorization. Clearly, it takes a special kind of courage, self-confidence, competence, and persistence on the part of the subordinate to break out of the syndrome.

Instead, what often happens is that members of the out-group set excessively ambitious goals for themselves to impress the boss quickly and powerfully—promising to hit a deadline three weeks early, for instance, or attacking six projects at the same time, or simply attempting to handle a large problem without help. Sadly, such superhuman efforts are usually just that. And in setting goals so high that they are bound to fail, the subordinates also come across as having had very poor judgment in the first place. The set-up-to-fail syndrome is not restricted to incompetent bosses. We have seen it happen to people perceived within their organizations to be excellent bosses. Their mismanagement of some subordinates need not prevent them from achieving success, particularly when they and the perceived superior performers achieve high levels of individual performance. However, those bosses could be even more successful to the team, the organization, and themselves if they could break the syndrome.

Getting It Right

As a general rule, the first step in solving a problem is recognizing that one exists. This observation is especially relevant to the set-up-to-fail syndrome because of its self-fulfilling and self-reinforcing nature. Interrupting the syndrome requires that a manager understand the dynamic and, particularly, that he accept the possibility that his own behavior may be contributing to a subordinate's underperformance. The next step toward cracking the syndrome, however, is more difficult: it requires a carefully planned and structured intervention that takes the form of one (or several) candid conversations meant to bring to the surface and untangle the unhealthy dynamics that define the boss and the subordinate's relationship. The goal of such an intervention is to bring about a sustainable increase in the subordinate's performance while progressively reducing the boss's involvement.

It would be difficult—and indeed, detrimental—to provide a detailed script of what this kind of conversation should sound like. A boss who rigidly plans for this conversation with a subordinate will not be able to engage in real dialogue with him, because real dialogue requires flexibility. As a guiding framework, however, we offer five components that characterize effective interventions. Although they are not strictly sequential steps, all five components should be part of these interventions.

First, the boss must create the right context for the discussion.

He must, for instance, select a time and place to conduct the meeting so that it presents as little threat as possible to the subordinate. A neutral location may be more conducive to open dialogue than an office where previous and perhaps unpleasant conversations have taken place. The boss must also use affirming language when asking the subordinate to meet with him. The session should not be billed as "feedback," because such terms may suggest baggage from the past. "Feedback" could also be taken to mean that the conversation will be one-directional, a monologue delivered by the boss to the subordinate. Instead, the intervention should be described as a meeting to discuss the performance of the subordinate, the role of the boss, and the relationship between the subordinate and the boss. The boss might even acknowledge that he feels tension in the relationship and wants to use the conversation as a way to decrease it.

Finally, in setting the context, the boss should tell the perceived weaker performer that he would genuinely like the interaction to be an open dialogue. In particular, he should acknowledge that he may be partially responsible for the situation and that his own behavior toward the subordinate is fair game for discussion.

Second, the boss and the subordinate must use the intervention process to come to an agreement on the symptoms of the problem.

Few employees are ineffective in all aspects of their performance. And few—if any—employees desire to do poorly on the job. Therefore, it is critical that the intervention result in a mutual understanding of the specific job responsibilities in which the subordinate is weak. In the case of Steve and Jeff, for instance, an exhaustive sorting of the evidence might have led to an agreement that Steve's underperformance was not universal but instead largely confined to the quality of the reports he submitted (or failed to submit). In another situation, it might be agreed that a purchasing manager was weak when it came to finding off-shore suppliers and to voicing his ideas in meetings. Or a new investment professional and his boss might come to agree that his performance was subpar when it came to timing the sales and purchase of stocks, but they might also agree that his financial analysis of

stocks was quite strong. The idea here is that before working to improve performance or reduce tension in a relationship, an agreement must be reached about what areas of performance contribute to the contentiousness.

We used the word “evidence” above in discussing the case of Steve and Jeff. That is because a boss needs to back up his performance assessments with facts and data—that is, if the intervention is to be useful. They cannot be based on feelings—as in Jeff telling Steve, “I just have the feeling you’re not putting enough energy into the reports.” Instead, Jeff needs to describe what a good report should look like and the ways in which Steve’s reports fall short. Likewise, the subordinate must be allowed—indeed, encouraged—to defend his performance, compare it with colleagues’ work, and point out areas in which he is strong. After all, just because it is the boss’s opinion does not make it a fact.

Third, the boss and the subordinate should arrive at a common understanding of what might be causing the weak performance in certain areas.

Once the areas of weak performance have been identified, it is time to unearth the reasons for those weaknesses. Does the subordinate have limited skills in organizing work, managing his time, or working with others? Is he lacking knowledge or capabilities? Do the boss and the subordinate agree on their priorities? Maybe the subordinate has been paying less attention to a particular dimension of his work because he does not realize its importance to the boss. Does the subordinate become less effective under pressure? Does he have lower standards for performance than the boss does?

It is also critical in the intervention that the boss bring up the subject of his own behavior toward the subordinate and how this affects the subordinate’s performance. The boss might even try to describe the dynamics of the set-up-to-fail syndrome. “Does my behavior toward you make things worse for you?” he might ask, or, “What am I doing that is leading you to feel that I am putting too much pressure on you?”

As part of the intervention, the boss should bring up the subject of how his own behavior may affect the subordinate’s performance.

This component of the discussion also needs to make explicit the assumptions that the boss and the subordinate have thus far been making about each other’s intentions. Many misunderstandings start with untested assumptions. For example, Jeff might have said, “When you did not supply me with the reports I asked for, I came to the conclusion that you were not very proactive.” That would have allowed Steve to bring his buried assumptions into the open. “No,” he might have answered, “I just reacted negatively because you asked for the reports in writing, which I took as a sign of excessive control.”

Fourth, the boss and the subordinate should arrive at an agreement about their performance objectives and on their desire to have the relationship move forward.

In medicine, a course of treatment follows the diagnosis of an illness. Things are a bit more complex when repairing organizational dysfunction, since modifying behavior and developing complex skills can be more difficult than taking a few pills. Still, the principle that applies to medicine also applies to business: boss and subordinate must use the intervention to plot a course of treatment regarding the root problems they have jointly identified.

The contract between boss and subordinate should identify the ways they can improve on their skills, knowledge, experience, or personal relationship. It should also include an explicit discussion of how much and what type of future supervision the boss will have. No boss, of course, should suddenly abdicate his involvement; it is legitimate for bosses to monitor subordinates’ work, particularly when a subordinate has shown limited abilities in one or more facets of his job. From the subordinate’s point of view, however, such involvement by the boss is more likely to be accepted, and possibly even welcomed, if the goal is to help the subordinate develop and improve over time. Most subordinates can accept temporary involvement that is meant to decrease as their performance improves. The problem is intense monitoring that never seems to go away.

Fifth, the boss and the subordinate should agree to communicate more openly in the future.

The boss could say, “Next time I do something that communicates low expectations, can you let me know immediately?” And the subordinate might say, or be encouraged to say, “Next time I do something that aggravates you or that you do not understand, can you also let me know right away?” Those simple requests can open the door to a more honest relationship almost instantly.

No Easy Answer

Our research suggests that interventions of this type do not take place very often. Face-to-face discussions about a subordinate’s performance tend to come high on the list of workplace situations people would rather avoid, because such conversations have the potential to make both parties feel threatened or embarrassed. Subordinates are reluctant to trigger the discussion because they are worried about coming across as thin-skinned or whiny. Bosses tend to avoid initiating these talks because they are concerned about the way the subordinate might react; the discussion could force the boss to make explicit his lack of confidence in the subordinate, in turn putting the subordinate on the defensive and making the situation worse.²

As a result, bosses who observe the dynamics of the set-up-to-fail syndrome being played out may be tempted to avoid an explicit discussion. Instead, they will proceed tacitly by trying to encourage their perceived weak performers. That approach has the short-term benefit of bypassing the discomfort of an open discussion, but it has three major disadvantages.

First, a one-sided approach on the part of the boss is less likely to lead to lasting improvement because it focuses on only one symptom of the problem—the boss’s behavior. It does not address the subordinate’s role in the underperformance.

Second, even if the boss’s encouragement were successful in improving the employee’s performance, a unilateral approach would limit what both he and the subordinate could otherwise learn from a more up-front handling of the problem. The subordinate, in particular, would not have the benefit of observing and learning from how his boss handled the difficulties in their relationship—problems the subordinate may come across someday with the people he manages.

Finally, bosses trying to modify their behavior in a unilateral way often end up going overboard; they suddenly give the subordinate more autonomy and responsibility than he can handle productively. Predictably, the subordinate fails to deliver to the boss’s satisfaction, which leaves the boss even more frustrated and convinced that the subordinate cannot function without intense supervision.

We are not saying that intervention is always the best course of action. Sometimes, intervention is not

possible or desirable. There may be, for instance, overwhelming evidence that the subordinate is not capable of doing his job. He was a hiring or promotion mistake, which is best handled by removing him from the position. In other cases, the relationship between the boss and the subordinate is too far gone—too much damage has occurred to repair it. And finally, sometimes bosses are too busy and under too much pressure to invest the kind of resources that intervention involves.

Yet often the biggest obstacle to effective intervention is the boss's mind-set. When a boss believes that a subordinate is a weak performer and, on top of everything else, that person also aggravates him, he is not going to be able to cover up his feelings with words; his underlying convictions will come out in the meeting. That is why preparation for the intervention is crucial. Before even deciding to have a meeting, the boss must separate emotion from reality. Was the situation always as bad as it is now? Is the subordinate really as bad as I think he is? What is the hard evidence I have for that belief? Could there be other factors, aside from performance, that have led me to label this subordinate a weak performer? Aren't there a few things that he does well? He must have displayed above-average qualifications when we decided to hire him. Did these qualifications evaporate all of a sudden?

The boss must separate emotion from reality: Is the subordinate really as bad as I think he is?

The boss might even want to mentally play out part of the conversation beforehand. If I say this to the subordinate, what might he answer? Yes, sure, he would say that it was not his fault and that the customer was unreasonable. Those excuses—are they really without merit? Could he have a point?

Could it be that, under other circumstances, I might have looked more favorably upon them? And if I still believe I'm right, how can I help the subordinate see things more clearly?

The boss must also mentally prepare himself to be open to the subordinate's views, even if the subordinate challenges him about any evidence regarding his poor performance. It will be easier for the boss to be open if, when preparing for the meeting, he has already challenged his own preconceptions. Even when well prepared, bosses typically experience some degree of discomfort during intervention meetings. That is not all bad. The subordinate will probably be somewhat uncomfortable as well, and it is reassuring for him to see that his boss is a human being, too.

Calculating Costs and Benefits

As we've said, an intervention is not always advisable. But when it is, it results in a range of outcomes that are uniformly better than the alternative—that is, continued underperformance and tension. After all, bosses who systematically choose either to ignore their subordinates' underperformance or to opt for the more expedient solution of simply removing perceived weak performers are condemned to keep repeating the same mistakes. Finding and training replacements for perceived weak performers is a costly and recurrent expense. So is monitoring and controlling the deteriorating performance of a disenchanted subordinate. Getting results *in spite of* one's staff is not a sustainable solution. In other words, it makes sense to think of the intervention as an investment, not an expense—with the payback likely to be high.

How high that payback will be and what form it will take obviously depend on the outcome of the intervention, which will itself depend not only on the quality of the intervention but also on several key contextual factors: How long has that relationship been spiraling downward? Does the subordinate have the intellectual and emotional resources to make the effort that will be required? Does the boss have enough time and energy to do his part?

We have observed outcomes that can be clustered into three categories. In the best-case scenario, the intervention leads to a mixture of coaching, training, job redesign, and a clearing of the air; as a result, the relationship and the subordinate's performance improve, and the costs associated with the syndrome go away or, at least, decrease measurably.

In the second-best scenario, the subordinate's performance improves only marginally, but because the subordinate received an honest and open hearing from the boss, the relationship between the two becomes more productive. Boss and subordinate develop a better understanding of those job dimensions the subordinate can do well and those he struggles with. This improved understanding leads the boss and the subordinate to explore *together* how they can develop a better fit between the job and the subordinate's strengths and weaknesses. That improved fit can be achieved by significantly modifying the subordinate's existing job or by transferring the subordinate to another job within the company. It may even result in the subordinate's choosing to leave the company.

While that outcome is not as successful as the first one, it is still productive; a more honest relationship eases the strain on both the boss and the subordinate, and in turn on the subordinate's subordinates. If the subordinate moves to a new job within the organization that better suits him, he will likely become a stronger performer. His relocation may also open up a spot in his old job for a better performer. The key point is that, having been treated fairly, the subordinate is much more likely to accept the outcome of the process. Indeed, recent studies show that the perceived fairness of a process has a major impact on employees' reactions to its outcomes. (See "Fair Process: Managing in the Knowledge Economy," by W. Chan Kim and Renée Mauborgne, HBR July–August 1997.)

Such fairness is a benefit even in the cases where, despite the boss's best efforts, neither the subordinate's performance nor his relationship with his boss improves significantly. Sometimes this happens: the subordinate truly lacks the ability to meet the job requirements, he has no interest in making the effort to improve, and the boss and the subordinate have both professional and personal differences that are irreconcilable. In those cases, however, the intervention still yields indirect benefits because, even if termination follows, other employees within the company are less likely to feel expendable or betrayed when they see that the subordinate received fair treatment.

Prevention Is the Best Medicine

The set-up-to-fail syndrome is not an organizational fait accompli. It can be unwound. The first step is for the boss to become aware of its existence and acknowledge the possibility that he might be part of the problem. The second step requires that the boss initiate a clear, focused intervention. Such an intervention demands an open exchange between the boss and the subordinate based on the evidence of poor performance, its underlying causes, and their joint responsibilities—culminating in a joint decision on how to work toward eliminating the syndrome itself.

Reversing the syndrome requires managers to challenge their own assumptions. It also demands that they have the courage to look within themselves for causes and solutions before placing the burden of responsibility where it does not fully belong. Prevention of the syndrome, however, is clearly the best option.

The set-up-to-fail syndrome can be unwound. Reversing it requires managers to challenge their own assumptions.

In our current research, we examine prevention directly. Our results are still preliminary, but it appears that bosses who manage to consistently avoid the set-up-to-fail syndrome have several traits in common. They do not, interestingly, behave the same way with all subordinates. They are more

involved with some subordinates than others—they even monitor some subordinates more than others. However, they do so without disempowering and discouraging subordinates.

How? One answer is that those managers begin by being actively involved with all their employees, gradually reducing their involvement based on improved performance. Early guidance is not threatening to subordinates, because it is not triggered by performance shortcomings; it is systematic and meant to help set the conditions for future success. Frequent contact in the beginning of the relationship gives the boss ample opportunity to communicate with subordinates about priorities, performance measures, time allocation, and even expectations of the type and frequency of communication. That kind of clarity goes a long way toward preventing the dynamic of the set-up-to-fail syndrome, which is so often fueled by unstated expectations and a lack of clarity about priorities.

For example, in the case of Steve and Jeff, Jeff could have made explicit very early on that he wanted Steve to set up a system that would analyze the root causes of quality control rejections systematically. He could have explained the benefits of establishing such a system during the initial stages of setting up the new production line, and he might have expressed his intention to be actively involved in the system's design and early operation. His future involvement might then have decreased in such a way that could have been jointly agreed on at that stage.

This article also appears in:



[HBR's 10 Must Reads on Managing People](#)

Leadership and Managing People Book

24.95 [Add to Cart](#)

- [Save](#)
- [Share](#)

Another way managers appear to avoid the set-up-to-fail syndrome is by challenging their own assumptions and attitudes about employees on an ongoing basis. They work hard at resisting the temptation to categorize employees in simplistic ways. They also monitor their own reasoning. For example, when feeling frustrated about a subordinate's performance, they ask themselves, "What are the facts?" They examine whether they are expecting things from the employee that have not been articulated, and they try to be objective about how often and to what extent the employee has really failed. In other words, these bosses delve into their own assumptions and behavior before they initiate a full-blown intervention.

Finally, managers avoid the set-up-to-fail syndrome by creating an environment in which employees feel comfortable discussing their performance and their relationships with the boss. Such an environment is a function of several factors: the boss's openness, his comfort level with having his own opinions challenged, even his sense of humor. The net result is that the boss and the subordinate feel free to communicate frequently and to ask one another questions about their respective behaviors before problems mushroom or ossify.

The methods used to head off the set-up-to-fail syndrome do, admittedly, involve a great deal of emotional investment from bosses—just as interventions do. We believe, however, that this higher emotional involvement is the key to getting subordinates to work to their full potential. As with most things in life, you can only expect to get a lot back if you put a lot in. As a senior executive once said to us, "The respect you give is the respect you get." We concur. If you want—indeed, need—the people in your organization to devote their whole hearts and minds to their work, then you must, too.

1. The influence of expectations on performance has been observed in numerous experiments by Dov Eden and his colleagues. See Dov Eden, "Leadership and Expectations: Pygmalion Effects and Other Self-fulfilling Prophecies in Organizations," *Leadership Quarterly*, Winter 1992, vol. 3, no. 4, pp. 271–305.
2. Chris Argyris has written extensively on how and why people tend to behave unproductively in situations they see as threatening or embarrassing. See, for example, *Knowledge for Action: A Guide to Overcoming Barriers to Organizational Change* (San Francisco: Jossey-Bass, 1993).

A version of this article appeared in the [March–April 1998](#) issue of *Harvard Business Review*.

Inserted from <<https://hbr.org/1998/03/the-set-up-to-fail-syndrome>>