The Manager and Scrum

goodagile>

Important Note

A thorough understanding of the principles and practices of Scrum is recommended prior reading this guide. We recommend *The Scrum Guide,* available for free at www.scrumguides.org.

About the Author

Pete Deemer is a well-known figure in the Agile software community, and has spent the last 25+ years leading teams building products and services at global companies. Pete served on the Board of Directors of Scrum Alliance from 2012-2017, and as Chairman in 2016.

Pete is the lead author of *The Scrum Primer*, one of the most widely read introductions to Agile development, as well as *The Distributed Scrum Primer*, a guide to multi-location Scrum.

Pete was VP of Product Development at Yahoo! in the early-mid 2000's, and prior to that served as SVP of c|net Networks. Pete was the co-founder of GameSpot, which is now part of CBS Interactive.

Pete is an honors graduate of Harvard University, and studied computer science at Stanford University. He spent a number of years as adjunct faculty at University of California Berkeley, most recently at the Haas School of Business, where he received the prestigious Club 6 teaching award. Pete has been a visiting lecturer at the Institute of Systems Science at the National University of Singapore since 2010.

Pete divides his time between India, Singapore, and Sri Lanka, and works with teams globally.

When an organization starts to explore Scrum, there's often an uncomfortable moment early on when someone points out that the role of "manager" seems to be missing entirely. "Well I guess we'll have to just get rid of 'em all!" wisecracks one of the developers, and all the managers in the room shift uncomfortably in their seats.

The Scrum Team consists of a Product Owner, a Development Team, and a ScrumMaster – and the basic direction given to others in the organization is to "support the Scrum Team, or get out of the way". This is not very detailed advice, especially if you're a manager expected by senior management to ensure that everything goes well.

The traditional role of the manager in the corporate world is based on a model known as "command and control". Here, the job of the manager is to identify what needs to be done, to issue detailed instructions to the employees, and then to ensure the employees complete the work according to the instructions. The role of employees in this model is simply to follow the directions as given, trusting the judgment and wisdom of the manager to ensure that the right work is being done in the right way.

In complex, dynamic environments such as software development, this approach can often be a major impediment. First, it is difficult and time-consuming for a manager to understand every requirement in full detail and issue precise instructions to guide the work of every employee. Within a software development team, the work is highly complex and interconnected, with constant discovery and course-correction. To expect a manager to "pre-think" this for his or her team is unrealistic, and often limits the team's productivity to the manager's ability to give instructions. In addition, this approach tends to be demotivating for employees; their role is simply that of "order follower", and they feel a limited sense of ownership. Accountability consists of simply answering the question, "did I complete the orders I was given?" If the answer is "yes", the job has been done – regardless of whether the right thing was built, built well, or built to satisfy the goals of the customer.

Scrum is based on a different approach: The Self-Organizing Development Team, a group of skilled professionals that work together

to produce potentially releasable increments of the Product in each Sprint.

Self-organization begins with the Development Team deciding how much Product Backlog to set as their goal for each Sprint. When Development Teams are empowered to decide how much is realistic and achievable, their sense of ownership, commitment, and motivation tends to be higher, and as a result they produce better results. Often, when managers first learn of this practice in Scrum, they voice the concern, "What if the Development Team under-commits?" This is not typically a problem, since the process of deciding the goal is very transparent and open. Indeed, it's much more common in the early Sprints for Development Teams to significantly *over*commit; most Development Teams have very little experience doing their own estimation, and it takes a number of Sprints before their optimism is tempered by experience. Moreover, in the event the Development Team *does* under-target, they can always pull additional items from the Product Backlog into the Sprint; no harm, no foul.

Development Team Tango had just completed Sprint Planning for a two-week Sprint. They brought in their manager, Jason, and walked him through the portion of the Product Backlog they'd set as their goal for the Sprint.

Finally, they asked Jason, "Does this seem like a realistic amount to target?"

Jason turned the question around to the Development Team:

"Do you truly believe you can finish this work, at high quality, by the end of the Sprint? Do you really feel committed?"

The Development Team members all nodded, looking quite convinced.

"Then it's the right amount for you to target." Jason replied. "And if it turns out to be too much or too little, you'll know two weeks from now, and you'll have learned something you can apply in the following Sprint".

The next aspect of self-organization happens during the Sprint, when the Development Team works together to decide what each person needs to do to achieve the goal. When the Development Team is responsible for this decision-making, they remain focused on the fact that they own the goal – and if the goal is to be achieved, they are the ones who must do it. When someone outside the Development Team is steps in and overrides this decision-making – for example, a manager – the Development Team receives a subtle but real signal that they are *not* actually responsible: it's the manager's job to worry about how to achieve the goal, not the Development Team's. This does not mean that managers are not providing support during the Sprint – on the contrary – but managers are careful not to interfere in a way that would reduce the Development Team's sense of ownership of the goal, or responsibility for organizing themselves during the Sprint.

On the first day of their first Sprint, the Development Team called their manager Sanjay over to join them for their Daily Scrum. Sanjay, wanting to be helpful, agreed to the request. He stood just outside the circle as the Development Team gave their updates to each other. Sanjay noticed that people seemed to be emphasizing how much they got done the day before, and weren't spending very much time reporting the impediments they were hitting. And after each person gave their short update, they looked over to Sanjay expectantly, hoping to catch a glance of approval. By the end of the Daily Scrum, Sanjay noticed that the entire circle of people had shifted, so they were now facing him.

After the last update was given, a Development Team member raised his hand, and asked "Sanjay, do you have any feedback or guidance for us?"

Sanjay knew that he had to say something.

"You know, I'm actually a little concerned. I feel like this Daily Scrum was for my benefit. I feel like you're still looking to me to make sure everyone's doing the right thing. Here's the deal: I'll give you any help you need, at any point in the Sprint. If you hit an impediment and you're not able to resolve it, I'm here to provide any assistance I can. But at the end of the day, you are responsible for doing what's necessary to meet the goal you've

set. So from now on, I'm not going to join this meeting. This is your meeting to organize yourselves, to achieve the goal you've set. If I'm here, I'm afraid I'm just going to undermine that."

The Development Team was silent. Then Victor, a member of the team, spoke up.

"So let me get this straight. We are the ones responsible here? We really do own this…?"

A subtle jolt of realization passed through the Development Team, and at that moment, they took their first step towards truly becoming a self-organizing team.

One of the biggest challenges in making the shift from a "following orders" mindset to self-organization is: The Development Team will not begin to self-organize until *everyone* outside the Development Team *stops* directing their day-to-day work. Development Teams are so conditioned to follow orders that they will often not begin to self-organize until there are no orders available to follow. This requires a leap of faith for the manager, and it can be scary. This is not to say that the manager abandons the Development Team – rather, the manager needs change their style of interaction, and constantly signal to the Development Team that they are now the ones with ownership and responsibility.

Eileen was an Engineering Manager at RedAlpha Systems, working with a Team of 7 relatively junior developers. During their first Sprint Planning Meeting, she sat at the back of the room working on email, as the Development Team completed the task breakdown for a big feature at the top of the Product Backlog.

When they finished, they turned to Eileen and said "How does this look to you?"

Eileen could see immediately that the Development Team had overlooked several important database tasks necessary to implement the feature. It would be very simple for her to simply

point out the tasks they'd overlooked, and the problem would be solved. Or would it?

Eileen decided to try a different approach. She stood up and announced, "Folks, this is a good start, but you're not quite

done. There are a couple important tasks that you've overlooked. But I'm not going to tell you what they are. I will give you a hint: Think more carefully about the user session data. Now I'm going to go and refill my coffee, and I'll be back in about 5 minutes. See if you can figure it out before I get back."

And at that, Eileen strode out of the room.

The Development Team looked at each other, slightly bewildered. Eileen had always been quick to point out what they'd missed; they depended on her for that. But this time, she was making them figure it out.

They stood in silence for a moment at the whiteboard, then slowly discussion began. They went through task by task, looking at each from different angles. Then, after a few minutes of discussion, Tony spoke up.

"Wait a minute… where are we going to store the user session data? We're going to have to make some changes to the database for that, right?"

There was a round of forehead slaps from the other team-members.

"Of course! How did we miss that!" several people murmured. There was a chuckle of embarrassment, and Sam started writing yellow Post-It Notes for each of these new tasks and putting them on the white-board. A few minutes later, Eileen returned with her cup of coffee. She looked at the white board, and nodded in agreement.

"Good job. Now why don't you all continue with planning your Sprint… I've got a bunch more emails I need to get through!"

In this example, it would have been faster and easier for Eileen simply to tell the Development Team what to do. But had she done that, she would have encouraged them to wait for solutions from her, and not think for themselves. Instead, Eileen did something harder, but ultimately much more valuable: She placed the responsibility on the shoulders of the Development Team to figure out what they had forgotten, and provided the least amount of help necessary to enable them to get it done. Had Eileen returned to find the Development Team still struggling, she could have provided another hint or asked another probing question, and continued to do so until the Development Team finally figured out the missing tasks. Eileen could even have let the Development Team proceed, and discover their oversight during the Sprint; mistakes often produce the most powerful learning experiences.

In simplest terms, the manager in Scrum is less of a "nanny" for the Team and more of a mentor or "guru", helping them learn, grow and perform.

In order for managers to be effective in this new mode, the organization must redefine the role and expectations of the manager. For example, in Scrum, the Development Team is responsible for organizing themselves to achieve the goal in the Sprint, and for this to work, it must be clear to all that the manager is not responsible for this. Similarly, in Scrum, it is the Product Owner's responsibility to decide the tradeoffs between scope, schedule, cost and quality to achieve the business goals of the release, not the responsibility of engineering management, and the organization needs to make this clear to everyone. Problems occur when the organization "talks the talk" on the new role of the manager, but does not "walk the walk" when things get difficult.

The Galaxy Team had been doing Scrum for several months, and the Development Team was well on its way to being truly self-organizing. Their motivation was high, they were focused, and after a few Sprints of not quite achieving their Sprint goal, they now had a good pattern of setting realistic Sprint goals and achieving them. Morale was high, and there was a real sense of

"flow" in the work they were doing. The engineering manager

Francis had come a long way – once a habitual micromanager, he was now acting like much more of a mentor and coach for the Development Team. Unfortunately, though, in the eighth Sprint, the Development Team encountered some unexpected difficulties, and about halfway through the Sprint, they were significantly behind in their progress. The VP of the group, Simon, had ventured into the Development Team's work area to see their Sprint Burndown Chart, and called Francis to his office.

"Francis, it looks like this Sprint is a disaster. What's going on?" he asked.

Francis responded, "Well, the Development Team hit some bumps along the way, and they're trying hard to get everything done that they committed to, but it's a bit touch-and-go right now."

Simon grimaced.

"Francis, this project is critical, and we can't let it fall behind. I'm counting on you to make sure the Development Team hits their Sprint goal, this Sprint and <u>every</u> Sprint. As a manager, your job is to make sure the Development Team gets it done; if things are going well, then you can back off a bit, but the minute the going gets tough, I want you in there making sure that no time is being wasted, and everyone is doing exactly what needs to be done."

Francis was exasperated. Simon had been too busy to attend the in-house Scrum trainings, but Francis had emailed him a Powerpoint presentation about self-organizing teams and the new role of the manager, and Simon hadn't voiced any disagreement. Francis spoke up:

"But what about the self-organization, Simon? What about our shift away from micromanagement?"

There was a glimmer of recognition, as Simon recalled a Powerpoint he'd seen a few months before.

"Yes, the Team <u>is</u> responsible, but when they start to fail, I hold <u>you</u> responsible. We want maximum accountability, so I'm holding them accountable and I'm holding you accountable. In

our department, everyone is accountable! Now make it happen."

At that, Simon spun his chair around and started typing. Francis took the hint and left the office.

The next day, Francis showed up at the Daily Scrum.

"Folks, we're going to do a different format for the meeting today. Due to the criticality of this project, Simon has instructed me to more actively… uhhh… 'facilitate' your self-organization during the Sprint. So what I'd like to do this morning is get a status update on each of the features you've committed to – whose done what so far, and what's left to be done – and I'm going to be giving some more detailed feedback so hopefully we can get everything 100% finished by the end of next week."

The Development Team looked at each other. Philip, the ScrumMaster, spoke up.

"Francis… uhhh… does this mean that the Development Team is no longer responsible for organizing itself?"

Several team members nodded in agreement.

Francis replied, "Look, we're all responsible. You're responsible for organizing yourselves, and I'm responsible for making sure you get everything done. We're all responsible together!" Francis didn't see the sideways glances.

As the Sprint proceeded, Francis was more and more involved. The Daily Scrum became an update meeting for the Development Team to tell Francis what they'd been able to complete, and for him to assign them the next day's tasks. The mood of the Development Team shifted; motivation seemed to go down, and Development Team members seemed to be reverting to their previous mode, what they used to sarcastically call "servants-of-Francis-the-Great". By the end of the Sprint, the Development Team was fully back into "order-following" mode, and Francis was directing their efforts task-by-task.

At the Sprint Review, the Development Team was surprised when Simon joined the meeting just as it was starting.

"So…" Simon announced, "Did you deliver everything?"

The Development Team looked at each other. Francis answered.

"Simon, unfortunately there are a couple Product Backlog items that weren't finished."

There was a flash of anger in Simon's eyes.

"How did this happen? Who is responsible for this?"

The Development Team was silent, but their heads all turned slowly to Francis.

Simon continued. "Francis, I told you to get it done. Next Sprint, I don't want to see this happen again. If it does, there will be hell to pay…"

Upon hearing this, everyone on the Development Team made a mental note to think very carefully about just how much to commit to in the next Sprint. The last thing they wanted was to get shouted at again two weeks from now.

As the Sprints passed, Francis became more and more involved in directing the Development Team at every stage of their work. Gone was any semblance of self-organization, and with it disappeared the improved motivation, drive, and focus that the Development Team had started to display. Morale had plummeted, and so too had productivity. Lunch breaks were getting longer, coffee-breaks more frequent, and Francis felt like he was spending more and more of his time just making sure people were at their desks working. Those amazing few Sprints, when the Development Team was truly self-organizing, and performing at the level they were really capable of, were becoming more and more of a distant memory. The return to micromanagement was made all the worse because they'd had a taste of the self-organization "good life".

There were errors of judgment at every step of this situation. The ScrumMaster didn't protect the Development Team from Francis' micromanagement, or call Francis out on the "double-talk". Francis didn't make any effort to reason with Simon, or help him see the consequences of his actions. But perhaps the biggest mistake was an early mistake: Simon was never properly educated about the shift in the management model that Scrum requires to be successful, and how this applies not only in good times but also when the going gets tough; and this shift was never made "official" in the form of a change to Francis's job description. And as a result, a successful, high-performance Development Team rapidly deteriorated back to its previous under-performing state.

The above scenario is extremely common and is a frequent point of failure for Scrum transitions. Furthermore, in an organization where this scenario plays out, word spreads very quickly, often causing other managers to proactively return to micro-management as a self-protective measure. So how does one prevent this kind of failure from occurring?

First, one has to make a clear-eyed assessment of management's willingness and ability to change, at every level. If there exists a fundamental belief in the effectiveness of the "command and control" approach within the management and executive ranks, and a heavy dependence on intimidation, threats, or shaming as a management tool, it is going to be particularly difficult to make the transition to a new way of working. As a result, an adoption of Scrum risks being incomplete and dysfunctional, producing little if any improvement for the organization.

However, if there is an openness to change, and a recognition that the existing command and control habits may not be the most effective approach, then there needs to be education and coaching at every level of management; in practice, this means Scrum training for all managers, up to senior leaders in the organization.

The final necessary step for completing this redefinition of the role of the manager is to "make it official" within the organization. One option is to use the pre-written job description included below as guide. The other option is to complete the interactive exercise that follows with

managers in the organization, to break down their existing job descriptions and rebuild them to be compatible with Scrum values and practices. With either of these approaches, it is critical to get formal approval of the manager's new job description from *his or her manager* (for example, the Engineering Director, or department head). Without a clear, "official" approval from senior management, the manager's new role will be more difficult to protect when difficulties arise.

THE MANAGER AS SCRUMMASTER

One approach to redefining the role of the manager is to convert them into ScrumMasters, but this has a poor track record of success. When the manager plays the role of ScrumMaster, it will be much harder for the Development Team to begin to self-organize. The pre-existing patterns of "order-giver" and "order-follower" are very difficult to break, and what's likely to happen instead is that command-and-control will be transplanted into the heart of the Scrum practices. As a result, the benefits that flow from a self-organizing Development Team – ownership, focus, drive, pride in quality, improved morale, and better productivity – will not be achieved. It would often be better to have a member of Development Team play the role of ScrumMaster, even if they do this in parallel with development responsibilities.

HANDS-ON: REDEFINING THE MANAGER'S ROLE

Step 1. Ask the manager to write down all of their current job responsibilities on Post-It Notes. The manager should try to include both official and unofficial responsibilities, and things they *should* have been doing but haven't had time to do. Here's a sample list:

Decide what work needs to be done	Assign the work to team members	Keep track of what everyone on the team is doing	Make sure the team gets their work done	Give input on what functionality the team should build
Give input on how to make features better	Help remove impediments that the team is not able to resolve by themselves	Make commitments to mgt about how much team can do by a certain date	Be responsible for the team meeting the commitments made to management	Provide advice and input to the team on technical difficulties that come up
Do weekly team staff meeting	Do weekly status update report for management	Have regular 1:1 meetings with team to provide coaching and mentoring	Plan training and other skills development for team	Do career-development and career planning with team
Recruit, interview and hire new team-members	Stay up to date on advances in tools and technologies	Stay up to date on industry news and develop-ments	Remove team-members who are not able to perform well	Plan and manage budgets and financials
Anticipate tools, skills and other future needs	Do performance evaluations and provide feedback			

Step 2. Draw two columns on the whiteboard: “Good in Scrum” and “Conflicts with Scrum / Not Needed in Scrum”. Ask the manager to go through the Post-It notes one by one, and place them in one column or the other.

Good in Scrum

1 Help remove impediments that the team is not able to resolve by themselves	2 Provide advice and input to the team on technical difficulties that come up
3 Do regular 1:1 meetings with team, to provide coaching and mentoring	4 Give input on how to make features better
5 Stay abreast of developments in tools and technologies team is using	6 Plan training and other skills development for team
7 Stay up to date on industry news and developments	8 Anticipate tools, skills and other future needs

Conflicts with Scrum or Not Needed in Scrum

15 Decide what work needs to be done	16 Assign the work to team members
17 Keep track of what everyone on the team is doing	18 Make sure the team gets their work done
19 Make commitments to mgt about how much team can do by a certain date	20 Be responsible for the team meeting the commitments I’ve made to management
21 Do weekly status update report for management	22 Do weekly team staff meeting

Good in Scrum (cont'd)

Plan and manage budgets and financials	9	Give input on what features / functionality the team should build (to the P.O.)	10
Do performance evals and provide feedback to team	11	Do career-development and career planning with team members	12
Recruit, interview and hire new team members	13	Remove team-members who are not able to perform well	14

Step 3. Take all the items in the "Good in Scrum" column, and turn them into a new job description for the manager.

Step 4. Ask the manager, "Will you be more useful or less useful to the organization in this new role?" and "Will this role be more interesting or less interesting for you to do?" In most cases, the immediate response will be "more useful" and "more interesting".

Step 5. Get formal approval of the manager's new job description from *his or her manager*. This is critically important. Without formal agreement, the manager's new role will not be "official", and there will be an even greater risk of falling back into prior patterns.

EXPLANATION

Good in Scrum

1. Help remove impediments that the Development Team is not able to resolve by themselves
While the ScrumMaster does this hour-to-hour / day-to-day, managers will need to focus on helping remove more systemic or organization-wide impediments. These are often the most vexing problems in the organization, and will require the management's influence, authority, or spending power to overcome. In *The Enterprise and Scrum,* Ken Schwaber recommends creating an enterprise transition team of managers and executives, who are responsible for evolving the organization based on a backlog of impediments.

2. Provide advice and input to the Development Team on technical difficulties that come up
Managers should be available to give advice or assistance whenever the Development Team asks for it.

3. Do regular 1:1 meetings with Development Team members, to provide coaching and mentoring
Managers should spend 1:1 time with team members, at a frequency that feels right. This is not a task update meeting – this is time for coaching and mentorship. Some managers like to do this sitting side-by-side, writing code!

4. Give input on how to make features better
This input goes directly to the Product Owner, typically during the Sprint Review.

5. Stay abreast of developments in tools, technologies, and techniques the Development Team is using
A very important and often neglected activity. Managers are can sometimes be "frozen in time" at the technology and development practices that were current when they were last doing actual development themselves.

6. Plan training and other skills development for Development Team members
Managers should think carefully about areas where the Development Team's skills could use development, or capabilities the Development Team will need to have to handle upcoming Product Backlog items.

7. Stay up to date on industry news and developments
Again, an important and often neglected activity.

8. Anticipate tools, skills and other future needs
Another important and often neglected activity. Be sure to get input from the Development Team.

9. Plan and manage budgets and financials
Another important and often neglected activity. Be sure to get input from the Development Team.

10. Give input on what features and functionality the Development Team should build
This input goes directly to the Product Owner.

11. Do performance evaluations and provide feedback to Development Team members.
A necessity within most organizations (despite well-documented flaws in the methodologies typically used). Managers should base their evaluations on their own observations and well as on feedback from the employees' fellow Development Team members.

12. Do career-development and career planning with Development Team members
Career opportunities are one of the most significant valuable forms of compensation people receive from their employer.

13. Recruit, interview and hire new Development Team members
Some of the best – and in other cases, worst – management actions are hiring decisions. Great hires pay dividends every single day they are employed – and poor hires are an invisible daily "tax" on the Development Team's ability to deliver business value. And for the record – the Development Team itself should play a central role in selecting new members of the team.

14. Remove team members who are not able to perform well within the Development Team
If even after extensive coaching a team member is not able to contribute, work harmoniously with other team members, or perform at the level required, they may need to be moved off the Development Team, or out of the organization. Typically managers will need to guide this process, in coordination with HR.

Conflicts with Scrum or Not Needed in Scrum

15. Decide what work needs to be done.
The Product Owner decides the features and functionality that needs to be built, and the Development Team determines what tasks are necessary to deliver this.

16. Assign the work to Development Team members
The Development Team does this itself, during the Sprint.

17. Keep track of what everyone on the Development Team is doing
The Development Team does this, in the Daily Scrum and on the Sprint Backlog.

18. Make sure the Development Team gets their work done
The Development Team is responsible for this.

19. Make commitments to management about how much Development Team can do by a certain date
The Product Owner measures the Development Team's velocity, and makes forecasts of how much of the Product Backlog the Development Team can complete by a specified date. If the Product Owner makes a hard-date release commitment, the Product Owner is responsible for including the necessary scope and schedule buffer in the plan, or making the tough decisions about which Product Backlog Items to de-scope or delay if the date proves unrealistic.

20. Be responsible for the Development Team meeting the commitments I've made to management.

The Product Owner is responsible for making decisions about what to do if velocity is lower than anticipated – either moving the release date, removing Product Backlog items, or simplifying Product Backlog items.

21. Do weekly status update report for management
Not needed in Scrum. If management wants to know how the project is going, they ask the Product Owner for the Release Burndown chart.

22. Do weekly Development Team staff meeting
Not needed in Scrum. The Development Team updates each other daily, and managers can get an update on the Sprint in the Sprint Review Meeting.

PERFORMANCE APPRAISALS IN SCRUM

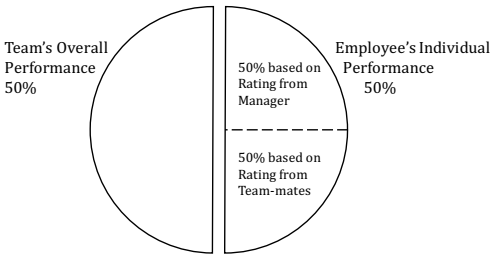
After adopting Scrum, many organizations continue doing performance appraisals using their traditional methods – for example, stack-ranking employees, or "curve"-based ratings.

These competition-based approaches might be effective in certain job contexts – for example, within sales team– but within software teams they undermine the teamwork and collaboration that are the foundation of team effectiveness. It is not uncommon to see a Development Team shift from a "We" mindset to a "Me versus You" mindset in the month or two leading up to performance appraisals, with a corresponding decrease in overall team performance. This is particularly unfortunately since the goal of doing these appraisals is to improve results.

The most successful organizations evolve their approach to evaluating and rewarding employee performance, as part of their adoption of Scrum. The goal is to shift the evaluation and rewards away from a competitive model and towards a more collaborative one.

One example is the approach shown below. Fifty percent of each employee's individual performance appraisal is based on the team's overall performance. The other fifty percent is based on the employee's individual performance; half of that rating comes from the employee's manager, and the other half from ratings given by team-mates.

This approach puts a strong focus on the overall team performance, and on recognizing team contribution, as opposed to encouraging competition between team-mates.



SAMPLE JOB DESCRIPTION FOR A MANAGER IN A SCRUM-BASED ORGANIZATION

- Lead the recruitment and hiring of new Development Team-members (with the active involvement and input of the existing Development Team-members)
- Provide input to the Product Owner on the product strategy and vision, and give feedback to the Product Owner on the content and prioritization of the Product Backlog.
- Provide support and assistance to Development Teams and their ScrumMasters. Be prompt and proactive in helping remove impediments that are harming Development Teams' ability to be effective.
- Actively support ScrumMasters' efforts to protect Development Teams from disturbance, disruption, or outside interference.
- Be available to provide advice and assistance to Development Teams on technical difficulties that arise in the course of doing their work.
- Identify issues to Development Teams that they might overlook, such as scalability, performance, security, etc.
- Provide mentorship and career development advice and guidance to Development Team-members. This mentorship should include both technical mentorship, as well as soft-skills and other aspects of being effective and successful in a development organization.
- Plan and manage skills development and training for Development Team-members. Think carefully about areas where their skills need greatest development, or where the most opportunity for improvement exists; work with the person to identify appropriate training; and obtain budget and time allowance to complete it.
- Stay abreast of developments in the tools and technologies that Development Teams are using. Solicit input from Development Teams and other stakeholders on tools and technologies that could be useful. Spend time getting hands-on familiarity with these tools and technologies.
- Stay up to date on industry news. Be knowledgable about developments from our company, our competitors, and our

largest customers, including financial performance, market share, product roadmap, and overall business strategy.

- Remove Development Team-members are not able to perform well within a given Development Team, work effectively with their fellow Development Team-members, or perform work at the level of expertise or quality required. This should come only after coaching and training has failed to correct the under-performance.
- Do financial planning and budgeting for Development Teams, including anticipating future people requirements, skills development and training needs, tools and technologies required, hardware, travel, and any other resources that people will require.
- Provide performance feedback and complete performance evaluations for Development Team-members. Informal performance feedback should be provided on a frequent basis, and should include feedback from fellow Development Team-members. Feedback should be focused on recognition for achievement, and opportunities for growth.

RECOMMENDED BOOKS

The Enterprise and Scrum
Ken Schwaber

Essential Scrum: A Practical Guide to the Most Popular Agile Process
Kennth Rubin

Conscious Business: How to Build Value Through Values
Fred Kofman

The Fifth Discipline: The Art & Practice of The Learning Organization
Peter M. Senge

Management 3.0: Leading Agile Developers, Developing Agile Leaders
Jurgen Appelo

Agile Retrospectives: Making Good Teams Great
Esther Derby

Coaching Agile Teams: A Companion for ScrumMasters, Agile Coaches, and Project Managers in Transition
Lyssa Adkins