Avoid...Limiting peoples' perspective

This is a generalization of many of the following tips. Some organizational systems stimulate or force people to specialize in only one skill or function. The deep knowledge of a skillful specialist is unquestionably an important organizational asset.

However, by doing so they limit the cross-functional learning pivotal for fast cycle time development and for agility. You do not want people to be exceptionally skillful in just one specialty. Products rarely need one specialty; they need an integration of talent from different areas.

Having excessively narrow specialists leads to communication and integration problems because those specialists do not understand or appreciate one another's domain. Thus, organizational systems need to support the creation of deep specialized knowledge and stimulate broad learning. It does not have to be a false dichotomy.

Avoid...Job titles

Job titles promote hierarchy and specialization.

Hierarchy—Job titles reflect status and hierarchy that often leads to less openness within a group. You do not want John to agree with Charles simply because Charles has a higher position. You want John to agree because he thinks that is the best way forward. Removing hierarchy is critical when establishing self-organizing teams.

Specialization—Job titles also reflect specialization, which leads to a narrow perspective of work. For example, developer John will not help with (or learn about) testing tasks because "I am a developer, not a tester." In Scrum, there are no predefined roles in the team because members of a self-organizing team think in terms of skill, not roles or job titles. For example, Sanjay does most of the testing because he is a true expert. However, when there are many testing tasks, John helps out—and learns more.

Try...Create only one job title

An alternative to removing job titles is to define only one. Sometimes this is easier because it does not require a change of HR policy. W. L. Gore is an unusual company that practices ‘unmanagement’ and ‘unstructure’ [MS95]. The size of every plant is limited to 200 people. When the unit grows above 200, it is split in two. The company has no organizational chart or formal hierarchy. Every employee has exactly the same job title: ‘associate.’

Try...Let people make their own titles; encourage funny titles

Another technique for removing status and hierarchy from job titles is to let all employees make up their own. Semco, an unconventional Brazilian company does this and they had three ‘gods’ working in the company. Having everyone select their own job title can lead to less resistance when a job title matters to people in their social environment.

Try...(if all else fails) Generic title with levels

Suppose none of the preceding options is feasible in the short term because HR insists on job titles with gradations or levels. An alternative in some companies is to use a relatively generic title with a level, such as “R&D member—level 3.”

Try...Simple internal titles map to special external titles

In some societies, job title status and moving up is understandably critical to people. It is not very impressive to tell your mother-in-law, or next prospective employer, that your official job title is...team member. A solution that strikes a balance is to define both an internal and external title. The internal title may be R&D member but when a person leaves the company, something more special may be provided to help the person in the job market, such as senior architect or senior manager.

Avoid...Job descriptions

Specific job descriptions and job titles lead to the same dysfunctions. Job descriptions might lead to people following them—the lean waste of under-realized talent or potential. This can also increase a blaming culture in which people look at the job descriptions to find the guilty party..."Well, he's the tester. It says on page two that he should have done that."

Try...Simple general job descriptions

The time and energy that people use to define specific job descriptions may in fact significantly contribute to the heat death of the universe. For months, pages of detailed points are passed around and reviewed for the "Fault Manager" job description. We wonder if the person who finally does this job ever reads the list. If removing job descriptions is not currently an option then at least make them simple, broad, and high level. For example, the description of "team member": A team member has the shared responsibility for the outcome of an iteration. High-level general job descriptions promote thinking, innovating, learning, and doing whatever is necessary to deliver the product, rather than strictly following limited responsibilities.

Avoid...Career paths

Large organizations promise their employees a future inside the company by providing road maps—career paths. We have seen and coached many organizations with career paths, and in all cases they fomented de-motivation and disappointment and stifled the multi-talented potential of people. Employees became frustrated because they wanted to follow the path faster and this resulted in more single-specialization.

Large-scale Scrum can lead to major changes in career paths. One product group we worked with used to have three paths:

- Project manager career path—This ceased to exist after the move to Scrum.
- Management career path—This became less attractive in a flatter organization where management responsibilities moved to Scrum teams.
- Technical/engineering career path—Originally, this meant to move away from hands-on programming and become a “Power Point architect.”

Scrum renewed the focus on the real-value work of development.

Predefined career paths made these changes more difficult because they upset people who had bet their career on one particular path.

Try...Job rotation

Job rotation—changing your job to a different functional domain—is an excellent way to create cross-functional learning. These ‘rotations’ can be for a few months or a few years.

I (Bas here) used to work in Nokia and always considered job rotation to be one of the strengths of the company. It was common to meet people who started their Nokia career in a completely different functional area. Why was this so common?

One reason is the history. Nokia was founded as a paper company and transformed itself into a rubber company, then to a steel cable company, then to a consumer electronics company, and lately to a telecommunication company. I met people who joined Nokia when it was a steel cable company. My first manager moved from engineering to sales and back to engineering, and this gave him a broad perspective.

Nokia is not unique in this. Honda lets engineers work in a different department for one week every year. W. L. Gore encourages its ‘associates’ to find their own work [MS95]. And, Toyota uses job rotation as one way of broadening people’s perspective.

Try...Start people with job rotation

When to start exposing people to a different functional areas? As soon as possible. During their first year at Toyota, new people go through “general training,” in which they build cars in the factory for a couple of months. They then work for a dealership to sell cars, even door to door (in Japan). This general training creates a broad feel for what their business is about. “The message is clear: Each employee works for Toyota, not for a specific function”.

Try...Hire the best

Great people are arguably the most important success factor in new product development. So companies that are serious about products invest time to make sure they hire the right people. For example, Microsoft and Google are famous for their strict hiring policies—and for talented developers. Great developers attract more great developers. They love working in a challenging environment with peers who are doing the challenging. Therefore, one way of hiring the best is to...hire the best.

In 2000, Joel Spolsky, an ex-Microsoft Excel developer, was looking for a great place to work. But he could not find a company where a focus on great developers was paramount. So he decided to start his own company—Fogs Creek. What was his business plan?

Eight years later, Joel is still proud of the way his company treats developers. They also build successful products and have been profitable ever since the company was founded. Their foundation: Hire great developers.

Avoid...Hiring when you cannot find the best

Though this tip seems obvious, it is not a common practice. In some large companies, the planned “head count” for a department determines how many people they should hire. And if the department does not hire them now, then they will not be allowed to hire them later.

This policy triggers these departments to locally optimize and hire “the best available” instead of “the best.” In product development it is better to hire a few great people than many average people. If you cannot find great people, then it’s better not to hire at all.

Try...Team does the hiring

How will you know whether a new hire will fit in a team? Involve the team in the hiring; better yet, let the team do the interviewing and let them make the final decision.
Michael Lopp, author of Managing Humans , feels that involving “some team members” is not good enough. He writes “Everyone on the team needs to interview every candidate ”.
Semco, the extraordinary Brazilian company, practices **collective interviewing**. They internally broadcast upcoming interviews and invite anyone to join. Having thirty plus people interviewing one candidate is not an exception. The more important positions attract more interviewers and—they say—this leads to a better selection. In one instance, no interviewer showed up for an interview and therefore that position was cancelled. It must not be needed if no one was interested in acting as an interviewer.

Try...Long and in-depth hands-on evaluation

“So, are you a good programmer?”... “Well, pretty good. I’ve been programming for seven years. On my resume you can see the products I’ve worked on.”... “Great! Next question. ...”

Not good enough. Talking about programming is easy. At ObjectSpace where I (Craig here) used to work, a candidate would have to spend six or more hours in a room with expert developers. During this period the candidate would have to sketch out (on the whiteboard) detailed design solutions for problems, and do relatively long and in-depth programming. The reviewers would carefully look at the code. If you want to hire great talent, inspect the real work of a candidate through some kind of long and in-depth, hands-on evaluation.

Although this tip used developer candidates as an example, it applies more broadly to other categories—digital electronics engineers, graphic designers, and more.

Try...Pair programming with developer candidates

Arguably the most in-depth approach to really seeing what a developer candidate can create and how they work with others is to evaluate them through long pair programming sessions. The candidate may write great code but be difficult to work with. Pair programming with a developer starts to expose his or her true technical and social work skills. Pair a candidate for a half-day or one day with one or more potential future team members.

Nothing beats evaluating a candidate on real work. Therefore...

Try...Trial iteration

How do you know if someone will really fit in with the team and has the right mix of skills? If the candidate passes the first round of pair programming evaluations, consider a second round of evaluation in which the candidate joins the team for one iteration. Of course, he would be compensated for working that iteration.

Try...Lots of formal education and coaching

Agile development and lean thinking involves new thinking tools, technical skills, behaviors, mindset, Scrum, test-driven development, continuous integration, self-managing teams...it's a non-trivial list. impediment #5 "failure to learn from outside"

The most common education mistake is: too little for too few. Within the Xerox lean development initiative, on the other hand, each team goes through at least four weeks of in-depth education and coaching in lean principles, Scrum, and agile development skills, both in the classroom and at their team location.

In another product group we coached in adopting large-scale Scrum, 50 percent of the people went through a ScrumMaster course even they were not all going to serve as a ScrumMaster. It provided common broad understanding.

It is important that virtually everyone in the product group is well educated in the new non-technical topics—Scrum and lean thinking, for example. All people need a common understanding and vocabulary of the big picture. Plus, development staff need in-depth education and coaching in agile development skills. Unequivocally, the large product groups that have been most successful in adopting agile and lean development focused on education and coaching.

Try...Lots of coaching

Some ideas can be well taught in a classroom or workshop. Some cannot. A good example is test-driven development (TDD). We have given many TDD courses, but when we ask the participants after some months whether they use it, a frequent reply is, “The course was great! We really want to use this, but we are currently working with this legacy code and it doesn’t apply there.”

Well, it does apply there as well. This is an example of the kind of skill that requires coaching for weeks or months at the person’s place of work, during their regular tasks.

The same is true for other practices. Educating a novice ScrumMaster while an expert coach facilitates the Sprint Retrospective is much more effective than a course in retrospective techniques. Have an experienced ScrumMaster come in and coach the team for the first few iterations and let other ScrumMasters observe.

Our advice to the organizations that we work with is to have some structured classroom education, but to primarily focus on using expert coaches who spend time with Scrum teams and the Product Product Owner Team during their Scrum events and during their day-to-day work.