



Manager



Luke Hohmann

Coaching the Rookie Manager

In last January's Manager column, "Fear of Trying: The Plight of the Rookie Project Manager," Roger Pressman asked a question critical to the future of our profession: "So, how do we grow a good project manager? What do we teach the rookie who has just been appointed to lead her first software project?"

Instead of answering this question directly, Pressman focused his article on four key skills that every project manager needs for success: communication, negotiation, organization, and facilitation. Although I agree with his key points, Pressman failed to explore the central question he first raised: just how do you "grow" the skills of a "rookie" project manager? This article is an attempt to answer that question.

COACHING: MODERN-DAY MENTORSHIP

Mentorship is a time-honored tradition for growing the skills associated with complex tasks. For centuries, apprenticeship was the only way you could learn to perform a complex task, from blacksmithing to carpentry. In "No Silver Bullet" (*The Mythical Man-Month*, Addison Wesley Longman, Reading, Mass., 1995), Fred Brooks points out that to be successful, companies must identify potentially great software designers and managers as early as possible in their careers, and "cultivate" their skills to allow them to realize their full potential: "I think the most important single effort we can mount is to develop ways to grow great designers." Brooks presents several ideas on how to do this, the most important being the assignment of "a career mentor to be responsible for the development of the prospect."

If you are a regular reader of *IEEE Software*, I sus-

pect this is not new information. Mentoring is a powerful mechanism for growing skills in any complex arena. That said, I think a better term exists to describe an optimal relationship between a rookie project manager and the person devoted to improving their skills: Coach. Before elaborating on what I mean, take a brief moment to record your thoughts about the differences between mentoring and coaching (yes, write them down).

When I think of a "coach," I think of someone absolutely committed to the professional growth and development of his or her team, and virtually nothing else (even if it is a "team of one"). A coach's role at extremely high levels of sport becomes even more important, and the best athletes in the world not only accept coaching, they actively seek coaches who can improve their performance—they *want* to be coached.

Mentorship, on the other hand, differs subtly from coaching, primarily in the areas of advising and friendship. To illustrate, think about your willingness to record your thoughts on the differences between mentoring and coaching as I requested earlier. As your mentor, I'd simply recommend that you perform this task, possibly identifying how you might find it useful. Ultimately, however, it would be your choice whether to do this. If you didn't want to do it, you wouldn't (did you?). If I were your coach and you didn't want to do it, I'd have you do it anyway, possibly reminding you of your desire to be coached. I might not even explain why you should do it.

The concept of coaching fits our current cultural norms better than the concept of a mentor, and a coach has more easily referenced role models. By describing novice project managers as rookies, even Pressman falls into this cultural reference and motivates a coaching relationship.



Which brings me to my first point: Establishing the proper relationship between rookie project manager and coach is essential for long-term performance improvement. If you are that rookie project manager, assess your own willingness to be coached. If you aren't willing to follow your coach's advice, it won't be of much value. This could mean, for example, doing tasks you think are unimportant, or cleaning up some mess you've made when you'd rather move on and do something else. Ultimately, it will mean doing something that for some reason or another you don't feel like doing.

If you are a senior project manager and have been given the task of mentoring a rookie, recast your relationship as his or her coach. You'll find a renewed commitment to that person's professional growth and an increased willingness to make demands that allow him or her to make real progress in their work. I find it easier to give negative feedback when I'm a coach than when I'm a manager or a mentor, and such feedback is usually better accepted.

LEARNING THROUGH EXPERIENCE

Learning can be thought of as the process of accumulating experience and wisdom through failure. Coaching (and certain aspects of management) can be thought of as the art of creating potential learning experiences that stretch the individual in key ways. Because these experiences are a stretch, some failure is inevitable. As a result, these learning experiences must be crafted so they do not permanently harm the individual or the company.

To get better at anything, you have to try, fail, reflect, regroup, and try again, until you succeed. Your first successes are likely to be sporadic and hard to repeat. Over time, and with enough practice and experience, you will improve your performance until you can perform skillfully. It doesn't matter if the skill is creating a good object model in UML, writing exception-safe C++ classes, or negotiating a delivery date for your next release.

A coach has many responsibilities in this process. The first is ensuring that "small" failures do not escalate and discourage rookie managers. Although initial preparatory training in management basics,

such as those Pressman described, can help, such training is largely ineffective unless actual work experiences match or exceed the rookie's skill level. When the rookie does fail, or realizes that he or she is failing and seeks help, the coach must be prepared

Coaching can be thought of as the art of creating potential learning experiences that stretch the individual in key ways.

to work with the rookie to attain a higher level of performance.

To illustrate, one of my "rookie" project managers was having trouble managing the inevitable evolution of requirements once the marketing requirements document was frozen. New issues were raised and debated, without the closure that the team needed to proceed.

If you're an experienced project manager, you've probably concluded that what my team needed was a more formal process for managing changes to requirements after they are frozen—something along the lines of a Change Control Board. But instead of telling my rookie to institute a CCB, I assessed the risks to the project of the current process. I felt that the risks were acceptable, and that the small failures the team was experiencing were not going to permanently harm the project manager, the team, or the company. So I waited.

When this manager eventually came to me, described the problems he was experiencing, and asked if I knew a better way, I had a perfect opportunity to coach him on the use of a CCB. He then modified a traditional CCB to fit our environment and introduced it. The improved process was immediately adopted.

Rolling this into my second point, prior introductory training in management can help the rookie project manager be successful. That said, such training is secondary to the actual "on the playing field" learning experiences the coach creates.

CAN I GO BACK?

Pressman identifies some of the fears a rookie project manager has when assuming a new position. One of the biggest was the "Dilbert Factor," which I interpret as the fear of appearing technically



Manager

incompetent in front of your technical peers as you exchange nitty-gritty technical skills for management skills. This fear is strongest in those who aren't certain they want to be managers. To be blunt, if you really want to be a good manager, you'll accept that increasing your management and leadership skills will mean losing some technical skills. And yes, at times you'll end up looking just as clueless as Dilbert's pointy-haired boss. It is inevitable.

I've found that a different kind of fear is often mistaken for the "Dilbert Factor": the fear of the un-

2. Inevitably, the rookie will try at least once during the project to shift responsibility for the project's success to the coach or the release lead. This is when the coach reinforces that, as much as possible, the rookie is the one in control and responsible for the project's success. Given that appointment to a project management role isn't necessarily permanent, you might think a rookie project manager would do a half-hearted job since he knows he can walk away, but I've never had this problem. Once the rookie realizes he really is responsible for

the project's success, he works hard toward that goal. Of course, the coach continually performs risk management and, if the project is at risk, can always provide more direct leadership. That said, we've never taken complete re-

sponsibility from a rookie manager, as that would be too demoralizing and would indicate inept coaching.

3. At Aurigin, a developer can work on one project, be a lead on the next, and go back to working on a third. This provides the necessary time to reflect on "management" experiences. More importantly, we've found that when a rookie manager has once again become a "normal" developer she is a more effective team member on the next project. Simply put, to be an effective leader you must also be an effective follower!

4. A final important factor is strong coaching. In addition to senior managers running my client and server development teams, I am fortunate to have strong leaders in QA and Docs. All these coaches proactively help rookie project leads understand when and how to integrate the "services" they provide into their project. Without some of these more senior people leading the services organizations, I doubt this approach would work as well.

TAKE EVERY OPPORTUNITY TO COACH

When you think of yourself as a coach instead of a mentor, you find yourself constantly focused on how you can improve the skills of every member of your team. Pressman suggests that coaches should help rookies learn how to communicate effectively. Quite often this requires that the coach insert himself into the middle of a communication event in a nonthreatening way.

If you really want to be a good manager, you'll accept that increasing your leadership skills will mean losing some technical skills.

known. Suppose a rookie project manager is genuinely interested in becoming a manager, but harbors a legitimate fear that she may simply not like management. In many companies this fear is compounded by the fact that once you are "promoted" from the technical ranks into a project management position, there is little hope of going back to a technical position without being labeled a failure.

We take a different approach at my company. Instead of a one-time promotion from development into project management, we try to grow the project and general management skills of interested developers through a series of assignments. At Aurigin, a release consists of several projects. Each project is "owned" by a project team. The project team, in turn, has a marketing lead and an engineering lead.

Each engineering project lead has a specifically named senior project manager as coach. Coaches, in turn, are identified by me and my staff. Groups providing services to the project team (such as QA, documentation, or training) are also led by strong leaders who proactively work with the project leads. These leaders ensure that everything that affects a successful product release is covered (from the impact on technical support and product installation through documentation and training). What makes this work is the following:

1. Developers can assume the role of project lead without formally changing job title or role. This allows them to "try on" management/leadership roles without the usual assumption that once you become a manager you're stuck.



To illustrate, consider the following interaction between Frank, a rookie project manager, and Henry, a developer assigned to his team (names have been changed). Frank told me that he had delegated a task to Henry. However, I was concerned by Frank's description; the task did not seem to be delegated with enough information to allow Henry to successfully complete it. So I asked Frank to describe, in detail, what he thought the task was. I asked the same of Henry, asking Henry not to go to Frank for clarification. Then I scheduled a meeting where the three of us reviewed the task together.

What we found was that Frank's detailed description of "what must be done" matched Henry's. This was a great victory—the task was, in fact, successfully communicated. However, Frank's description contained a section called "What Henry does not need to do to succeed." This section was missing from Henry's description of the same task.

Here was a major learning opportunity for both Frank and Henry. Henry learned that good "followership" includes asking what doesn't need to be done to be successful. And Frank learned that effective leadership means communicating these boundaries.

So how do you grow the skills of rookie project managers? Establish yourself as their coach. Craft specific work experiences designed to grow their skills. Work with them to improve their performance when they fail. Organize these experiences in a series of steps so that they can gradually, but steadily, move into management. Finally, remain vigilant, looking for every opportunity to coach, for it is through coaching, not general platitudes, that we grow effective managers and leaders. ❖

Luke Hohmann is vice president of engineering for Aurigin Systems. Author of *Journey of the Software Professional: A Sociology of Software Development*, he spends as much time as possible coaching individuals and organizations to greater levels of performance. He can be reached at lhohmann@aurigin.com.

Computer Scientists & Engineers

your Vision

our Future

Southwest Research Institute is seeking to fill Modeling and Simulation, Software Engineering, and Software Design positions. All require a BS or MS in Computer Science, Computer Engineering, Electrical Engineering, or equivalent, programming in C, C++, and Java, on PC or UNIX based workstations. Candidates must have excellent communications skills and work well independently or as a member of a team. U.S. citizenship required for most positions.

Modeling and Simulation

Analyze, design, develop and implement modeling and simulation algorithms and software for training systems. Perform software lifecycle phases including requirements generation, software design, prototype development, testing, implementation, integration, maintenance and support, and documentation; interact with Institute staff and clients; travel to client sites. **Requires** a minimum 3.25 GPA; experience using Ada; experience with distributed simulation (live, virtual, and constructive); knowledge of computer systems, software architectures, hardware-software interfacing techniques; and the ability to work with MS Word, Excel, and Project.

Software Engineering

Develop software for military and industrial applications on workstations and PCs. Perform software lifecycle phases including requirements generation, software design, prototype development, testing, implementation, integration, maintenance and support, and documentation; and make technical presentations. **Requires** a minimum 3.5 GPA and at least B grades in Computer Architecture, Software Engineering, Data Structures, Operating Systems, Compiler Theory, and Database Management courses or their equivalents.

Software Design

Serve as a member or leader of project teams and design and develop software for military and industrial applications. Manage and execute software development including requirements analyses; system level design and documentation; software design, planning, validation, testing, integration, and maintenance. Applications include real-time data acquisition, graphic user interfaces, system level control processing and embedded algorithm implementation, and network interfaces (Internet, LAN/WAN, etc). **Requires** five years' experience in software system design and development with emphasis on Windows 95®, Windows NT, and UNIX (XWindows). Must be able to take concepts and produce working software solutions, interface with clients, and make technical presentations.

Over 2600 men and women, with a variety of educational and cultural backgrounds, provide creative solutions to diverse technical requests from all over the world. We're problem solvers in all areas of technology. Set the pace for your career and explore the opportunity to run your own project. Join us and continue a 51-year tradition, shaping our future with your vision.

Please submit your résumé to: Southwest Research Institute, Personnel Department #99111, P.O. Drawer 28510, San Antonio, TX 78228-0510 • E-mail: personnel@swri.org or Fax: (210) 522-3990

www.swri.org

An Equal Opportunity/Affirmative Action Employer • A Public and Community Services Employer

- Digital Electronics Design
- Analog Electronics Design
- High-Level Architecture
- MS Access
- Paradox
- C/C++
- Ada
- Java
- PC Platform
- Workstation Platform
- MS Excel
- MS Project
- UNIX
- Windows 95®
- Windows NT
- Internet
- LAN/WAN



Southwest Research Institute