Training Software Project Managers

Lawrence Peters, Software Consultants International Limited

Abstract. The presumed goal of training software project managers is to equip them with the knowledge and competencies that will help them to be successful. These will not guarantee success but make success more likely. Over the years, the notion of success has expanded greatly from simply meeting requirements, delivering on time and not exceeding the budget to include a plethora of other success criteria. In fact, what success is often changes many times during the project. This and many other facets of software project management frustrate and perplex untrained software project managers since most enter into this role untrained [1]. This article presents what anecdotal and experimental evidence has shown software project managers need to know that can be conveyed via training programs. Today's software project manager can also benefit from this information to overcome many of the misperceptions about nearly everything regarding software project management.

Targeted Issue

Software Engineering Education Issue - The importance of adequate management of software projects is slowly becoming more apparent. Most current software project managers and those seeking to become software project managers have either been inadequately trained to address today's software engineering issues or have not been trained at all. This article examines what we now know is needed to successfully manage software engineering projects and how education can help transfer this information.

Introduction

Software project management represents a paradox within the software engineering community. It has been described as being more vital to software project success than all other factors combined [2], yet there are still no conferences or journals devoted to this topic. In fact, international conferences on software engineering rarely list software project management as a topic in the call for papers topic list. Finally, we are slowly realizing what other knowledge work related professions have known for a long time – project managers are not born, they are made – through education. The problem is, there is no general agreement on what knowledge and skills a software project manager needs in order to be successful. In fact, we have yet to agree on just what success in software engineering is. This article examines what software engineering project managers need to know, what skills they must possess and how these may be acquired through education. The resources cited are not restricted to software engineering alone as there is a lack of research in this subject.

The Nature of Software Engineers

Some years ago, psychologists sought to answer the following question, "What is it about software development that has attracted so many people from such a broad range of disciplines? They found that most software engineers shared two unique (taken in combination) psychological characteristics [3]:

1. High Growth Needs Strength - A desire to solve challenging problems.

2. Low Social Needs Strength - A strong tendency to work alone.

To summarize, these people are attracted by software development because they want to deal with significant, technically challenging problems and not have to deal with other people. Unfortunately, the scale and complexity of today's software development efforts requires that teams of software engineers build and maintain them. Since software project managers are drawn from the ranks of the software engineering teams, they also display the psychological profile of most software engineers and tend not to have great "people skills." This works against them in a management role [4].

What Software Project Managers Do

Without doing any research, it is difficult to figure out just what software project managers do. The model we will work from proposes that software project managers are responsible for performing 5 basic functions [4] often in parallel, executed in concert with their team:

Scheduling – Laying out a list of milestones and dates consistent with the contract. Contrary to what you may have read [5], scheduling and planning are not the same activity.

Planning – Detailing the tasks and subtasks that must be successfully executed in order to proceed from one milestone to the next. This is an ongoing process throughout the project to adapt to unforeseen problems.

Controlling – Monitoring the project's progress, taking action to recover deviations from the project plan, as necessary.

Staffing – Acquiring the human resources needed to successfully execute the project plan.

Motivating – Creating and maintaining a physical and psychological environment that ensures the development team works at its full potential.

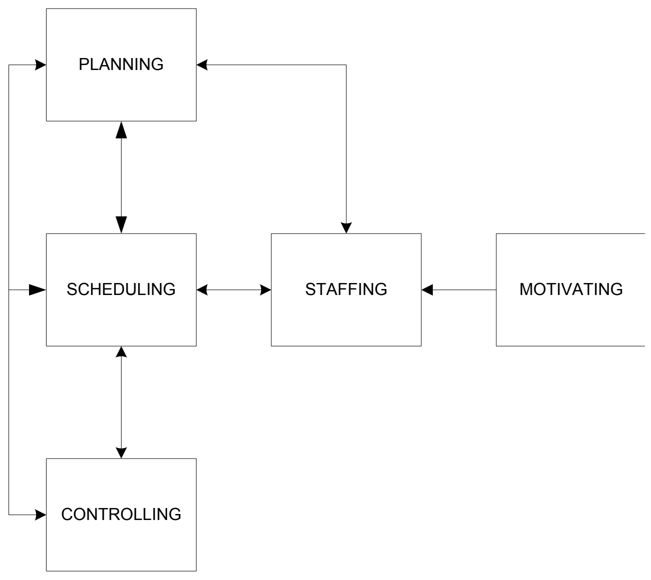


Figure 1: Software Project Manager Activities

Figure 1 shows the relationship among these activities. The relationship among these functions is shown in Figure 1. If those topics appear distant from programming, it is because the software project manager is more like the manager in baseball, not a player. Before heading into software project management, a software engineer needs to consider whether or not they are ready to move into this very different domain.

Where Do We Get Software Project Managers From?

Several decades ago, a very successful CEO (Robert Townsend) wrote a book detailing what needed to be done to improve how companies functioned [6]. One of the points he made seems to have been ignored by the software engineering industry [1]. What Townsend was trying to communicate was the fallacious belief that the most capable person in a team should become its manager. For example, we have a team of 5 people. Someone has to be responsible for meeting with the client, preparing status reports and assigning responsibility for specific development tasks to team members with their concurrence. If software project management duties are assigned to the most skilled software engineer, the productivity of the group is reduced because this highly skilled person has little or no time to do programming. Besides, management tasks and development tasks require very different mindsets. Worse, it is likely that this highly skilled person will not be empathetic to less skilled team members and help them improve their skills, help them solve difficult programming issues, and so forth.

A Few Misconceptions Seen as Self-Evident Truths by Many Software Project Managers

People work for money – Money ranks as 5th in importance [7] (see Table 1). But more importantly those who have studied why we work [8, 9, 10] have concluded we work for self-esteem, fulfillment and other reasons, not just for money. We have a paradox here in that the most expensive reward (salary) is the least appreciated while the most appreciated reward (a simple "Thank You") is the least expensive.

If we get behind, we can catch up – using Earned Value Management, if the project is 15% complete and behind schedule, based on a study of 700 DoD contracts, the chances of getting the project back on plan are nil [11].

Putting Pressure on the Team will Improve Productivity – Presumably the knowledge and experience the team brings to the project are what are needed to do the job. It has been shown that pressure to perform causes the team to break up into individual problem solvers effectively destroying the collective intellectual power of the group [12].

To Avoid Getting Behind, we will start with a bigger team – This is done to avoid adding people to a late project but it is surprisingly ineffective [13].

Offer some big reward, that will get them working – It has been shown that if the reward is big enough, people will cheat to get it [14].

Break up successful teams to other groups to spread the knowledge – The most successful software company(s) in the world do not do this but keep teams intact as much as possible and work to help other teams to learn to be successful [15].

Factor	Manager's Importance Rank	Non-Manager's Importance Rank
Salary	1	5
Job Security	2	4
Promotion/Growth Opportunities	3	7
Working Conditions	4	9
Interesting/Challenging Work	5	6
Personal Loyalty to Workers	6	8
Tactful Discipline	7	10
Appreciation for Work Done	8	1
Help with Personal Problems	9	3
Being in on Things	10	2

Table 1: Relative Importance of Project Factors

Software engineers Do not really need to be reviewed, they know how they are doing – Evaluating the performance of software engineers using the usual "one size fits all" human resource system is inappropriate for most high technology workers [1, 4]. A better approach is to tie evaluations to an individual's contribution to achieving corporate or group strategy [4, 16].

We can accurately predict the future – Nobody has been successful at this but now we know why we fail to accurately estimate software projects and how to correct for our over optimism and failure to fully recognize risk [16, 17].

Treat everyone the same – This century may mark the highest occurrence of multi-generational teams since we were an agrarian society. In addition, both in-house and outsourced projects will involve multiple cultures. Being aware of and responding to the issues of the value systems of these various groups will require knowledge and diplomacy. Due to the worldwide financial crisis, these are issues all software project managers must address [18].

The preceding represent both misinformation and some antipatterns. An antipattern is a solution to a problem that actually makes matters worse. More than 95 of these have been identified, categorized and published [19] and the list continues to grow.

What Do Software Project Managers Need to be Taught?

One way to answer this question is to look at what successful software projects have in common, then glean from that what we need to provide to current and prospective software project managers. A study of nearly 600 software projects worldwide that were reported to have been successful found a number of common "success oriented" factors [20]. These factors did not guarantee success but made it more likely (Table 2).

In examining Table 2, you might conclude that you could have guessed at some of the items without the benefit of a disciplined literature search worldwide. But there are a couple of items you may find need further explanation. One is the term, "Competent project manager." The knowledge and competencies that constitute a competent software project manager is what this article is about. At the present time, we do not know whether or not a software project manager is competent until it is too late – the project has either been successful or failed. The other item that may seem curious occurs on the output side of Table 2, "Job satisfaction." It turns out that if software engineers do not find the work they are doing to be professionally satisfying, challenging and so forth, their productivity is low. Presumably, successful software projects result in large part from having a productive development team [16].

Input Factor	Brief Description
Clearly stated requirements	Clear and well understood requirements
Involved users	Active and continuous participation of users during the development process
Engaged, competent project manager	A project manager with the required management and leadership skills, able to share the project's vision
Project planned and scheduled	A project plan and schedule developed with stakeholder participation to achieve user goals
Engaged, skilled team members	Competent team members with domain and technical knowledge, as well as positive attitude about the project
Teamwork and communication encouraged	Development team with compatible personalities who enjoy working in a team environment and have a cooperative and mutually responsive relationship
Output Factor	**Brief Description**
Schedule and budget estimate maintained	Finishing the project within estimated budget and timeliness of delivery
Customer and user needs satisfied	Making easy-to-use, user friendly systems that meet requirements
Job satisfaction experienced on development team	The development team has a sense of accomplishment that sufficient quality and functionality were delivered and that they were given enough freedom and independence to be successful
Product quality, functionality and performance meet high standards	The working product reflects the desired scope and overall quality

Table 2: Input and Output Factors of Successful Projects [20]

Who Will Teach Them?

This issue presents us with a dilemma. Most successful software project managers are very busy with little spare time to contribute to the educational system(s) in their area. Besides, the fact that the word "management" would appear in the title of the software project management course can and has set off a turf war between the software engineering department and business administration departments in many universities. Professional development courses or extension courses help but content, quality and instructor qualifications come into play. For example, has the prospective instructor ever been a software project manager? Some very famous people lecturing on this subject have never managed a software project.

What Kind of a Job Are We Doing Now?

Not a very good one. Training in software project management if offered at all at the bachelor, master or doctorate level as part of a degree in software engineering is often optional – should not it be required? Why? Even if the software engineer never goes into management, he/she is going to be asked how long a task will take, what confidence level they have in the estimate, what they will need and so forth. Without training in planning and scheduling the answer to such queries can be problematic. During the execution of the plan, the issue of the status of the work will also come up. Without training in the use of earned value management, the most likely answer will be, "OK." If you have ever sat in on a project status meeting and wondered what metric was used to determine "OK" when you had heard rumors that things were not OK, you know some objective method needed to be applied.

Analysis of the content those university programs that offer software project management, whether it is required or optional, reveals that they are focused on programming issues, Agile, software tools and a general condemnation of the Waterfall Lifecycle.

Is Software Project Management Important?

The software engineering industry has spent more than a half century developing dozens of new methods, and techniques all directed at solving, "the software problem." What effect has all this had – an abysmal one [21]. Programming productivity has improved in a linear way by less than one source line per person month per year from 1960 to 2000. But facets of our profession continue to annoy customers. These facets are uncertainty of delivery date, cost, quality, ease of use, maintainability, communication and so forth. Remember, the software project manager is the biggest single factor in determining project success – bigger than all other factors combined [2]. One would think that with such importance, software project management would be the last subject we would choose to be optional.

What Software Project Managers Need to Know

For those who feel they would like to manage a software project here is a short list of skills you will need and some you won't.

Needed

Interpersonal skills – the ability to connect with each member of your team as a friend and colleague, a servant – leader who team members view as having their best interests at heart.

Communication skills – both in written expression (e.g. written reports) and presentations (e.g. PowerPoint or other format in front of an audience).

Basic Cost Accounting – if you do not know the 3 to 5 most common factors that make a $75,000 per year software engineer cost your project $120,000 or more per year, you do not have these.

Negotiating, motivating, evaluating personnel - the list goes on and on but notice the absence of programming skills.

Mentoring – work to improve the performance of team members and the team as well as creating your own replacement.

Encouraging Prosocial behavior – simply thanking the team and individual members helps to overcome independent tendencies to form an effective team [22].

Sensitivity to Cultural and Generational differences More than at any time in the history of the United States and some other developed countries, multiple generations are having to work together. The differences between the value systems of different generations can cause frictions within the team. Similar comments apply to cultural differences due to the world wide influence of software engineering in nearly every aspect of people's lives.

Self-confidence – consistently hire people smarter and more knowledgeable than you. The productivity of the team will reflect your skill in spotting talent and your job will become easier.

Not Needed

Programming, being quick to anger at failure, tending to fix the blame rather than the problem, embarrassing a team member in front of their colleagues because they made a mistake. This may be a sudden reaction to bad news but it can have negative consequences that go beyond the team member being chastised. Other team members may see this as an indication that trying something new or difficult should be avoided. Regardless, this reduces team productivity.

What Would a Curriculum for Software Project Managers Look Like?

More than 30 years ago I wrote the first M.S. in Software Engineering curriculum published by the Association for Computing Machinery [23]. It was adopted and implemented by Seattle University. Since that time, with the exception of software engineering, the engineering profession in general has incorporated personnel issues into management [24] training as well as the nature of management itself [25]. What is becoming apparent is that the nature of management in general and software project management in particular has changed dramatically since that original curriculum development.

So what would a curriculum look like to train software project managers? A lot different from the ones I found on the internet which emphasized specific lifecycles blaming the waterfall lifecycle for everything from athlete's foot to zits, programming methods (e.g. Agile) and programming languages. But that is to be expected. We tend to teach what we know, what we have experience with and so forth. In the hope that this will help the professors of today and tomorrow, here are some suggestions and subjects that need to be incorporated into software project management curricula:

Suggestions –

Require that all students, regardless of whether they are at the B.S., M.S. or PhD level take and pass at least an introduction to software project management in order to receive their degree. This is already the case in Europe as part of the European Master on Software Engineering (EMSE) program.

Eliminate any discussion of programming languages, methods, software tools and so forth from the software project management class(s).

Seek out software professionals, software project managers and key stakeholders in your area to determine what training software project managers appear to be lacking.

Form a review committee from the people in item 2 to review and critique the content of the software project management course(s).

Prepare a presentation containing an overview of the course and deliver it in person at major employers as well as publicly to ensure sufficient enrollment to fund the class.

Here are Some Things You Can Do

Look at what your local university(s) offer in the way of software project management classes – not just the course titles but their content as well. Ask yourself if someone had the knowledge imparted by these classes, would they be more likely to be successful? If the answer is no, identify what topics should be added, which should be dropped from the course(s) and speak with the college or university focal point for these classes. Be sure to emphasize the increased attendance and revenue that could result from these changes. Also, enlist the help of colleagues from other firms to bring about the required changes. If the university senses there exists an area wide market, they may be more amenable to change.

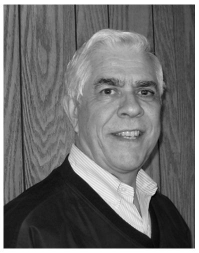
Closing Comments

Decades ago, we began training software engineers in a broad range of methods and techniques directed at both improving the quality and quantity of software systems. Though the progress may seem slow, it is progress. If we do the same for software project management, the benefits will likely accelerate the pace of improvement far beyond what we have seen so far.

Disclaimer :

(Portions of this article contain excerpts from the Kindle eBook, "Managing Software Projects: On the Edge of Chaos, From Antipatterns to Success")

ABOUT THE AUTHOR



Dr. Lawrence (Larry) Peters has 4 decades experience in software engineering as a software engineer, project manager, consultant, and educator (he currently teaches Software Project Management in Spain via Skype). He has worked on many defense systems. His clients have included many Fortune 100 companies, the US DoD, and the Canadian Defense Establishment. He has a B.S. in Physics, an M.S. in Engineering and a PhD in Engineering Management. He created the first Software Engineering laboratory for the Canadian Air Force, written 4 books and published several papers.

E-mail: ljpeters42@gmail.com

REFERENCES

1. Katz, R., "Motivating Technical Professionals Today," IEEE Engineering Management Review, Volume 41, Number 1, March, 2013, pp. 28-37.
2. Weinberg, G., Quality Software Management, Volume 3: Congruent Action, Dorset House Publishing, New York, NY, pp. 15-16
3. Couger, D. J. and Zawacki, R. A., Motivating and Managing Computer Personnel, Wiley-Interscience, New York, N. Y., 1980.
4. Peters, L. J., Getting Results from Software Development Teams, Microsoft Press Best Practices Series, Redmond, Washington, 2008
5. McConnell, S., The Software Manager's Toolkit, IEEE Software, From the Editor column, July/August, 2000
6. Townsend, R., Up the Organization: How to Stop the Corporation from Stifling People and Strangling Profits, Josey-Bass, 2007
7. National Study of the Changing Workforce, Published by the Families and Work Institute, New York, NY, 1993.
8. Herzberg, F., Work and the Nature of Man, The World Publishing Co. , Cleveland, Ohio, 1966
9. Maslow, A. H., The Farther Reaches of Human Nature, Viking Press, New York, N. Y., 1971
10. McClelland, D. C., The Achieving Society, Van Nostrand Reinhold, Princeton, N. J., 1961
11. Fleming, Q. W. and Koppelman, J. M., Earned Value Project Management - 4th Edition, Project Management Institute, Newtown Square, Pennsylvania, 2010.
12. Gardner, H.K., "Performance Pressure as a Double Edged Sword: Enhancing Team Motivation While Undermining the Use of Team Knowledge," Harvard Business School, Working Paper 09-126, 2012.
13. Staats, B. R., Milkman, K. L., and Fox, C. R., The Team Sizing Fallacy: Underestimating The Declining Efficiency of Larger Teams, Forthcoming article in Organizational Behavior and Human Decision Processes.
14. Gino, F. and Ariely, D., The Dark Side of Creativity: Original Thinkers Can be More Dishonest, Harvard Business School Working Paper 11-064, 2011.
15. Huckman, R. and Staats, B., The Hidden Benefits of Keeping Teams Intact, Harvard Business Review, December, 2013.
16. Peters, L. J., Managing Software Projects: On the Edge of Chaos, From Antipatterns to Success, Kindle eBook, Software Consultants International Limited, 2014.
17. Flyvberg, B., From Nobel Prize to Project Management: Getting Risks Right, Project Management Journal, August, 2006, pp. 5 – 15
18. Aiman-Smith,L., Bergey, P., Cantwell, A.R., Doran, M., "The Coming Knowledge and Capability Shortage," Research-Technology Management, Vol. 49, No. 4, July-August, 2006.
19. LaPlante, P. A. and Neill, C. J., Antipatterns: Identification, Refactoring and Management, Boca Raton, Florida, Taylor and Francis, 2005.
20. Ghazi, P., Moreno, A. M. and Peters, L. J., Looking for the Holy Grail of Software Development, IEEE Software, Jan/Feb, 2014, pp. 92-94.
21. Jensen, R., Don't Forget About Good Management, CrossTalk, August, 2000, pp. 30.
22. Grant, A. and Gino, F., A Little Thanks Goes a Long Way: Explaining Why Gratitude Expressions Motivate Prosocial Behavior, Journal of Personality and Social Psychology, Volume 98, Number 6, pp. 946-955, 2010.
23. Peters, L. J. and Stucki, L., A Software Engineering Graduate Curriculum, ACM 1978 Annual Conference Proceedings, ACM New York, NY, pp. 63-67.
24. Thamhain, H. J., Changing Dynamics of Team Leadership in Global Project Environments, American Journal of Industrial and Business Management, 2013, Number 3, pp. 146-156, 2013.
25. Thomas, J. and Mengel, T., Preparing project managers to deal with complexity – Advanced project management education, International Journal of Project Management, Volume 26, pp. 304 – 315, 2008.

WANTED

Electrical Engineers and Computer Scientists
Be on the Cutting Edge of Software Development

The Software Maintenance Group at Hill Air Force Base is recruiting **civilians** *(U.S. Citizenship Required)*. Benefits include paid vacation, health care plans, matching retirement fund, tuition assistance, and time paid for fitness activities. **Become part of the best and brightest!**

Hill Air Force Base is located close to the Wasatch and Uinta mountains with many recreational opportunities available.

Like facebook
www.facebook.com/309SoftwareMaintenanceGroup

Send resumes to:
309SMXG.SODO@hill.af.mil
or call (801) 777-9828