The Secrets of Project Management

Robert Mattson
9545 Delphi Road. S.W.
Olympia, Wa 98502

Introduction

So much of the work we do in "systems" is in the form of "projects." The problem seems to be that there are a lot of ways a systems project can get messed up. I've been challenged by the nature of such projects for a number of years. From this I've concluded that there are some key concepts project managers need to understand and focus on to succeed on system projects. Therefore, the first goal, in this paper, is to pass on these "secret" concepts as "food for thought." The second goal is hopefully to provide something that will help further the success of your projects.

The Seven Secrets

With those goals in mind, I'm going to discuss seven "secrets" of system project management. I don't claim that these are the only important concepts/secrets. In fact, there are so many different challenges in system projects that no single set of answers will ever exist. However, I'm convinced that understanding and focusing on **these** seven "secrets" can result in many significant benefits. Further, my experience and research has led me to the conclusion that a lot of experts are missing key points or focusing on the wrong ones. To the extent that I haven't seen the points that follow discussed or emphasized...I call them "secrets." Finally, the "secrets" are applicable to any type of project although my emphasis throughout is that of a system project. So, let's get started at looking at these "secrets."

Understanding "Goals" and "Success"

Understanding "goals" and "success" is where system projects should start...unfortunately I believe, too few do!

Let's take the issue of "goals" first. A common error is to focus on the goals of a "systems" nature over those of

the "business" and "user." System nature goals have to do with things like "trying a new language", or "ease of coding". The user's goals are usually oriented toward doing a "business'" or "organization's" function better, faster, easier and cheaper. The key is to understand what this means for the user! Too many times I see the key systems person on a project unable to express these "user's goals" in users terms. I believe this situation accounts for much bad press about how systems people cannot "understand" or "talk" in terms of the "user". Therefore, we need to focus on and clearly understand the user's goals. We, also, need to make these a prime driving force in system projects.

Related to the area of "goals" is the understanding of the meaning of the word "success." I've made the strong point in previous paper, Why System Projects Don't Quite Succeed, that each key player in the system project "game" measures the "success" score differently. The following is a chart showing how key players usually measure "success".

	Functionality ("User Goals")	Time/Cost	Internal Quality	Style
Users	80%	10%	0	10%
"Programmer"	30	40	30	0
User Mgmt	15	80	0	5
Top Mgmt	5	80	0	15

Note the differences in how the various groups and people measure success. Of special significance is how much weight management places on time and cost in measuring success. Equally, note how the user places an equally disproportionate weight on functionality. The significance of these differences is that if you please the user to the detriment of time and cost then many times the project will be judged less than a success by top management. Conversely, there are many completed systems projects that top management thinks were very "successful" which resulted in achieving only a marginal functionality. Further many of these systems may have very high life cycle costs. Top management based its belief on the fact these system projects were on time and on budget. Many "PC" projects are like this.

One may argue with the above percentages...but that doesn't really negate the key point. That is, different people view "success" differently. This knowledge doesn't necessarily solve every trade-off problem a project manager

faces. It does ,however, help us understand how different players in the project will judge the project and by association those working on the project team. This knowledge is very important in understanding the "politics" of projects. With it, we can better balance our actions, goals and emphasis to achieve the result we desire.

So the first secret is:

One needs to understand a system project's many goals and measures of success in order to maximize desired results.

People versus Techniques

We now have a clearer understanding of the meaning of "goals" and "success" in system projects. So, let's focus on how we can achieve them. Our first step is to explore the people versus technique issue. To do so, we start with what appears to be an "assumption" hidden in a lot of literature on project management. This assumption is that with the application of the "right" project management "techniques", one can make any project a success. This may sell books, but from my experience and observations it's not that easy! In fact, other things being equal, give me a person with "right" knowledge, skills, experience and attitude over any magic project management techniques.

What I am saying is that the first assignment and emphasis for the software project manager is to find the right people rather than focusing on "technique." Given the right people the chances for success go up tremendously. Conversely, without the right people chances for success are less than good no matter how great the project management.

Unfortunately, most managers appear to have less than full leeway in finding, selecting and getting rid of people on projects. In addition, our knowledge of how to find and select the person we need is less than perfect. It is when we are faced with this common reality, that we'd better understand and know how to apply all the rest of the secrets of project management and any other useful techniques. But let us not lose sight of the fact that the "right" person is worth a pound of techniques.

The place you can really notice the lack of attention to the value of people is in the area of "people resources".

Books on system project management almost universally gloss over the fact that there are ten to one or more differences in capabilities of people. In other words, the time it will take and the very quality of the product in systems is intimately tied to who does it. The nature of most system tasks and system personnel differences is such that no real assumption can be made about how long it will take an "average" systems person to do it. Yet, over and over again I see project management books or lectures treat system people as units such as "programmer" or "system analyst".

A related issue has to do with the commonly held belief of "staff or personnel development." Many managers' performance is rated on their success in developing personnel. Although, it may be heresy, I'm going to suggest that our ability and knowledge about how to "develop" people may be far less than we commonly believe. In addition, on many projects, the time and resources are just not available accomplish the needed development. For validation of this idea, try looking squarely at how much time and energy you have spent trying to "develop/change" a person. How successful was it? Was it really a good "business decision?" I would suggest that project managers may be way ahead to spend their time figuring out how to get "wrong" or "unmatched" people off their projects rather than trying to change them.

So the second secret is:

The "right" people are more important than "right management" techniques in system projects.

Communication

A great deal of the problems that arise in projects stem directly from "communication" related behavior that is in conflict with human limitations. As a consequence, people up and down the project organization "think" they've "communicated" "things" successfully when in fact they haven't. The "things" that are communicated in a project are major items like coordinating directions, specifications and requirements.

If we could, I'd like to magically measure the number of times in projects that an action isn't taken because the

person never "heard" the request. Then I'd like magically to measure all the time spent by people doing the task as they thought someone wanted it done, but which later was found to be not on target. The number of times and more importantly the consequential cost of these two situations would be very significant. In fact, many projects fail or are much less than successful because of the poor quality of the communication going on.

Thus from this standpoint, project management is largely about how we communicated what, who, when, why and how much. It's about how successful we are in communicating up and down and across the project organization.

There are many challenges when one is trying to improve communications. For a discussion of many of these see the paper Communications - Why do Systems People Have Such a Hard Time With It?. Although there are many problems related to communications, the heart of many of the solutions involve an emphasis on written over verbal communications. This especially applies to system project management... usually it's the lack of clear "hard copy" of various kinds that leads to many of the projects problems.

So the third secret is:

Good project management must include good communication.

Task Management

There is no more key concept in the "management" of projects than that of understanding "task management." Projects are made up of a whole bunch of sub-projects or "tasks." From this perspective "project management" is made up of a whole bunch of "task management." It amazes me to see people worried about how they are going to manage a project when...they can't tell you how to manage a single task that makes up the project. To belabor the point, if one can't manage the tasks that make up the project...one can't manage the project!

The best example in software systems is that of the lowly program. If you can figure out how to manage the development of a single program and you're on your way to understanding how to plan and manage a "system." Conversely, if you are having problems with managing the development of a

single program... then you probably will have even more problems with the whole system.

So to improve, we need to focus our attention on the "tasks" and their management first. How do we manage a task? First we need to understand that each task is a project. Sounds like circular reasoning, but it isn't really. What it means is that the same techniques and concepts that apply to "projects" also apply to the "tasks" in the project. It also follows, if we are having a problem managing this "task" then we need to figure out what its sub-tasks are and how to manage those. Our goal is to reach a point where we are "successfully" managing a task. When we achieve this goal we are ready to move up a "level" in our management.

So the fourth secret is:

Focus on managing "tasks" before "projects."

Managing for Quality

A little discussed issue is that of how to manage for "quality." Quality, in the sense I mean, deals with the measurement of the excellence of the "deliverable" or "product" of a task or project. Quality is the third leg of the project "triangle." A project triangle describes the balance between cost/resources, time and quality. Changing any one of the parameters most likely affects the others. The understanding and "managing" of this relationship is a significant part of what "project management" is all about.

In spite of the fact that we need to manage all three areas, the emphasis in almost all the project management thought is on the time and cost parameters. In fact, out of all the pages in the texts on project management I've read, less than 2% deal with this issue of the management of quality. In part this emphasis is due to a lack of understanding of the importance of quality. Also, the "management" of cost and time is "easier" than that of quality. Finally, remember that "top management" is emphasizing cost and time in its judgement of project success.

Why do I feel this lack of attention is unwarranted and significant? In software projects it is the quality of the product that greatly affects both long term success and

costs. If we place too much emphasis on initial project's time and cost then I will almost guarantee that the quality of the product will suffer.

Why will quality suffer? It is because people doing a project will tend to place their emphasis and effort in the areas where they are being measured. So when "managers" emphasize time and cost their "task workers" will do so also. Add to this the fact that the majority of original system estimates for time and cost are understated. And, as commonly happens, these estimates become "gospel." Finally, we usually don't plan for or "measure" product quality. The net result of this scenario is that people know they can let product quality slide a little in order to meet the time and cost deadlines!

So how do we manage for quality? The keys are 1) defining the quality parameters 2) defining levels of achieving on each parameter 3) assigning "value" to each parameter/level agreed to 4) measuring against this definition 5) analyzing and adjusting our actions to improve overall quality for the next time. For a more complete discussion of this topic see the paper Quality - Let's Discuss this "Can of Worms".

We will have achieved a significant milestone in systems when our project management includes managing for quality.

So the fifth secret is:

Manage a project's quality as well as its time and cost.

Planning and Re-planning

Few would disagree that management of a system project can be helped greatly by the use of a "good" plan. And, in general, the best plan is one that is as accurate as possible. But most plans lose their "accuracy" rapidly. So one could conclude that to keep a plan of value it must be constantly updated. The truth is that most plans are not updated anywhere near as often as they should be. As a consequence, most system project plans are nearly useless!

Why are inaccurate plans of little value? It is because one of the prime uses of plans is as a communication tool. A good plan communicates at least the who, what, when and how much of the tasks of the project. When this data is not

accurate on a plan, people will rapidly ignore the plan. They will ignore it for planning their own work. More significantly, they will not provide needed updates to a plan they perceive as worthless.

Additionally and unfortunately, when we don't re-plan we don't re-think. Much of the value derived from planning comes from the thinking we do about the tasks. If this thinking is not being done, then critical issues affecting the project may easily be overlooked. Revising a plan should provide the impetus to step back and see the forest as well as the trees.

Based on my experience in the systems arena, most tasks should be re-planned based on the following rules.

Re-plan when:

1) we are at 10%, 50% and 70% of elapsed time

AND

2) once a day

AND

3) when any fact comes to our attention that we feel may affect, by greater than 2%, the schedule completion date, resources required or quality.

In producing a plan or a "re-plan" we need to be sure that we understand what is in a plan. A "plan" tells who, when, how long, why, how much, what products and of what quality. This may sound pretty obvious. Unfortunately, I believe that the majority of the "plans" done each day leave out at least one of these items. This is unfortunate, because leaving out any this data diminishes the usefulness of the plan. It follows that when we "re-plan", we must also re-specify all these items in a plan.

Another often missed feature of most re-plans is that of changes and comparisons. We should in our re-plan compare it to the original plan. The re-plan should point out our tasks added, tasks removed and tasks completed. It should also highlight variances between the "original" plan and the re-plan as to who, when, how long, how much, and the quality.

So the sixth secret is:

The latest "good" plan is worth many times the former.

Self-Management versus Managing Others

The chances for project success go up directly in relationship to how well each person on the team accepts individual responsibility for its success. This in a sense could be called "self-management." Conversely, if we try to "manage" people into doing what is right, in the right amount of time, by the right date, etc.... we will almost surely not achieve it. My experience and observation has convinced me that trying to control a project by externally monitoring and controlling others is much less successful than commonly believed.

If self-management in a project is desired, then how do we achieve it? The way you get everyone to be responsible and their own self manager has to do with the personnel issue. In line with earlier comments, I believe it is not easy to develop or change someone into a self manager. But it can be done sometimes. Mostly, this is another issue where having the people with the right attitudes on the team is worth a great deal. Give me someone who wants to do something over someone who has to do something!

It may be helpful to focus on what the self manager on a project or task takes as their responsibility. They need to take responsibility for implementing the concepts("secrets") discussed so far. It also means accepting responsibility for achieving the plans as to time, resources and product quality. So responsibility in a project or task includes the following:

Understanding Goals and Success
Correct Staffing/Knowledge/Skills
Right Type, Timing and Quality of Communication
Understanding and Managing for Quality
Planning and Re-planning
Reporting of Problems with Re-plans
Periodic Status Reports
Review of Future Plans
Task/Project Reviews for Time, Resources and Quality
Achieving the Plans and Goals for Time, Resources and product Quality.

If you do get **everyone** on the project team to accept responsibility for the above items then the likelihood of success goes up tremendously. The job of project manager then becomes much more one of coordinator. On the other hand, if we try to "manage" and/or "control" people in these areas the job is much harder and success less certain.

So the seventh secret is:

The best management on a project is self-management.

Tools and Techniques

This section really isn't about any secrets. Rather it's about the value of many of the tools and techniques for the management of projects. Some techniques and tools are useful but some are very overrated. But, if you have successfully dealt with the "secrets" I've discussed then these "other" tools and techniques probably become more important.

Project management software is one of the magic answers that has been pushed in recent times. Unfortunately, most of the software available is much better suited for a project such as "building a house" than for the development of software. Two key characteristics of software projects are its high level of change and differences in project personnel. Most project management software works best on projects with much less change and personnel uncertainty than is typical of software projects. There is a real potential for this type of tool to help us in our software projects. We unfortunately are just not there yet in the state of the art.

Here are some additional techniques/concepts which I feel are very overrated for most system projects:

Critical Path Method
Pert Charting
Networks
Contingency Planning
Risk Analysis
Float Calculations
Milestones
Skills Inventory/matrices
Earned Value Concepts

Time and space does not allow me to make the case for why the above are overrated. Nevertheless, one of the real challenges for those people involved in managing software projects is to not waste time trying to apply techniques and methods of dubious value. Focusing on the right techniques and issues will pay the largest dividends.

So the final "non-secret" is really a philosophical challenge:

Knowing what technique not to apply is often times as important as knowing what to apply.

Conclusions

I've spent considerable time studying, dealing with and thinking about system project management. The breadth of the issues and the challenges are amazing. There are no easy "silver bullets" to slay the beast. Many of the tools and techniques exposed are sorely inappropriate to many environments and projects. What **may** have worked on a multi-billion dollar missile project may or may not work on your mult-thousand dollar software project.

I believe that the "secrets" I've outlined are very significant to the successful management of software projects. Maybe I believe it's as simple as that... clear understanding, clear goals, clear people, clear plans, clear tasks, clear responsibility, clear quality means clear success! If that sounds too simple and easy it probably is. If there is one thing I've learned over the years there is usually one or more new challenges on each project. As the saying goes... **I may not have all the answers but I sure admire the problem!**

REFERENCES

Mattson, Robert R., "Why System Projects Don't Quite Succeed," Proceedings 1985 International Meeting - HP3000 Users Group, Washington, D.C., September 8-13, 1985.

Mattson, Robert R., "Communications - Why do Systems People Have Such a Hard Time With It?", Proceedings 1987 North American Conference of Hewlett-Packard Business Computer Users, Las Vegas, Nevada, September 20-25, 1987.

Mattson, Robert R., "Software Quality - Let's Discuss This "Can of Worms.", Proceedings 1988 North American HP Business Computer Users Conference, Orlando, Florida, August 7-12, 1988.