

Chapter 6 - Teams, Managers, and Leaders

Programmers tend to think of themselves as rugged individualists, but in fact the project succeeds only if their work is carefully coordinated; and leading programmers can be as frustrating as trying to herd cats. I want to draw a distinction between leading programmers and managing programming projects, because the two activities can be very different, and managers are not the only leaders in a project. In an active project the question, “Who’s in charge here?” can often be a very debatable issue.

In most companies, the lowest level of organization is the “programming team.” Teams range in size from three to ten programmers, and are directed by a Team Leader. Teams can be formed for a number of reasons:

- ❖ to implement a particular product feature,
- ❖ to track down a particularly difficult bug,
- ❖ to improve the program’s performance,
- ❖ and on and on...

A team might stay together for a few days, if they’re just tracking down a bug, or for years if they are responsible for a product feature, but the average team lifetime is around six months, and programmers are accustomed to being moved among teams, much like baseball players being traded.

Entering a new team might be very natural, if the programmer has worked with most of the team members and is respected; or it can be difficult if she is an outsider and has to prove her competence to be accepted. And there is an inexplicable team chemistry: some teams seem to function together as though they were telepathic, and others seem to spend most of their

time squabbling over details. Putting together the right team for the task is one of the more difficult jobs of a programming manager.

Team Leaders

The rank and file programmer generally has to contend with three layers of management. Her most immediate manager is actually a member of her team, called something like “Team Leader” or “Technical Lead.” This is a rather strange position, lying halfway between technical development and management.

A Team Leader is an active programmer who is responsible for developing software just like everyone else in the project. But she also serves as the team’s representative to the real project management. The Team Leader is the one who attends the Monday morning status meetings, reports on her team’s progress (or lack of progress), pleads for resources, and generally tries to take care of her team.

Typically, she doesn’t have any of the “real management” tasks like negotiating schedules or allocating resources or making hiring decisions. She can’t directly discipline team members or bestow extraordinary rewards, but her requests are usually granted and her advice followed by her manager.

But she is responsible for running her team smoothly and efficiently and for this she has only the force of her personality and the borrowed authority of her boss, the Project Manager. To be perfectly clear, the position of Team Leader is one of the most difficult ones in a software development organization, since the Team Leader is at once responsible for technical development and team management, and has all the re-

sponsibility but little of the authority that should go with that role.

Programmers who are thinking of going into management often try out their skills by becoming Team Leaders, and many of them are scared back into the technical ranks by the experience.

Project Managers

The next level of management is called something like “Project Manager” or “Department Manager,” and is the lowest level of the real management tree. The Project Manager of a large project may manage up to thirty or forty programmers through five to ten Team Leaders.

Nearly all Project Managers came from the programming ranks, but have left active programming behind, and their skills may be very rusty. Still, they are expected to have enough technical background to understand in detail everything that is going on in their project and be able to summarize it and report it to upper management. This job calls for someone who can keep thousands of details in her mind, recognize problems as they are emerging, and react to them quickly and decisively.

A Project Manager can typically command the resources in her project, but doesn't have the authority to hire or fire or to transfer programmers into or out of the project. Project Managers have the additional burden of the non-technical bits of management: disciplining unruly employees, doing performance evaluations, encouraging her Team Leads, and generally keeping up morale and discipline in her project.

Project Managers are responsible for formulating the project's schedule and shuffling resources to make it feasible, though they usually recruit the Team Leaders to assist in this effort. And when the project gets into trouble and the schedule starts to slip, it is the Project Manager who has to report the problem to upper management, take the consequent heat, and plan how to get the project back on schedule. The role of Project Manager carries quite a lot of stress, and Project Managers tend to burn out quickly and dramatically.

Upper Management

In most companies, programmers are aware of the Project Manager's bosses only as shadowy figures who generally don't understand what is going on in the project and have a penchant for causing trouble.

The One-Way Management Door

There's a strange phenomenon in the world of the programmer: the move into management is a one-way journey. A programmer may decide to get into management either because it suits her temperament or for professional advancement. But from that point she is forever marked as a manager, and the programmers seem to assume that she left her technical skills behind when she acquired the office with the name on the door.

If a manager decides she made a mistake and would rather return to the world of development, she may not be able to. Even if she returns to the cubicles and becomes a programmer or a Team Leader of her own volition, the upper management labels her as a failed manager, rather than a dedicated developer. And

meanwhile, her programming peers distrust her because she has allegedly let her skills slip. It's a very distressing position to be in, and many managers who want to return to the ranks of programmers leave their company to avoid it.

About Leadership

If you tell a programmer, "Take me to your leader." she will probably be a little surprised at the question and respond, "Which one do you mean?" Leadership in the programmer's world is not a formal attribution, and it certainly doesn't follow the organization chart. Let's look at several sorts of leadership that can be active concurrently:

- ❖ Formal leadership derives from the organization tree and flows from the Project Manager through the Team Leads.
- ❖ Personal leadership comes from project members with strong personalities who can enforce their opinions on the rest of the project through their presence.
- ❖ Technical leadership is found in project members who are specialists in a technology important to the project.
- ❖ Legacy leadership comes from project members with long history in the project and the company, and provide the organizational memory of the project.

And because there are multiple kinds of leadership, the "leaders" of a team or project tend to emerge and recede as it goes through different phases. When the team is pursuing a particularly difficult bug, the Team Lead may defer to the technical expert and follow his

direction until the bug is fixed. When the project has to make a difficult design decision, the project leader may call on the most experienced members to make the call, based on their memories of past decisions.

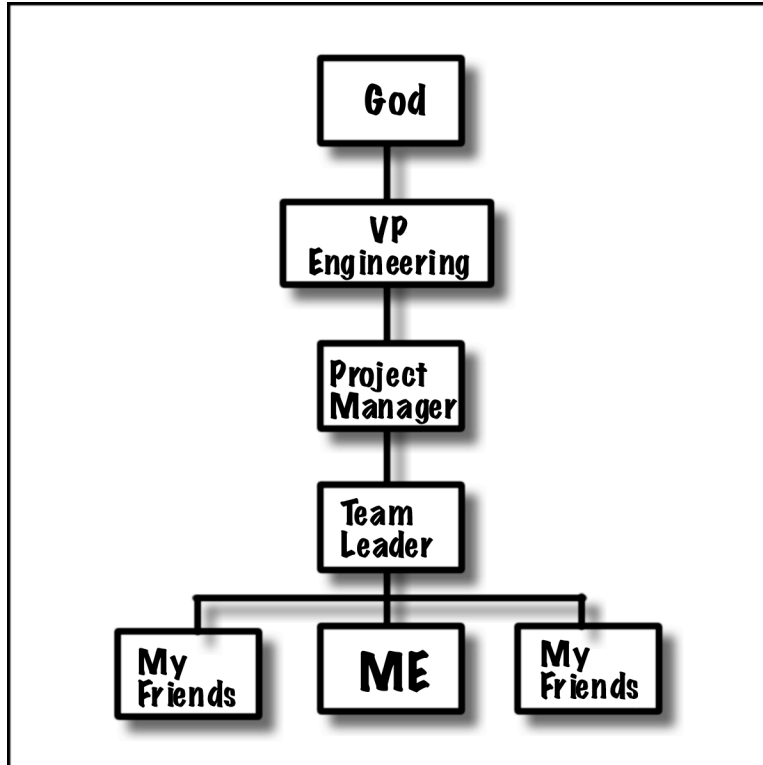
In a healthy project, leadership is a little like geese flying in their characteristic V formation: at any moment there is only one leader, but the leader may change from moment to moment. When all the potential leaders understand the project's goal and priorities this doesn't present a problem because programmers are used to the notion of changing leadership, and leaders are willing to step back into the ranks when it's appropriate.

But things can go badly wrong when multiple leaders emerge, or when one leader tries to play all the roles. If, for example, the Team Leader uses her formal leadership to make all the technical decisions for the team, she is likely to alienate the rest of the team members and make an enemy of the member who is rightly the Technical Leader on those decisions.

Or if a project member with strong personal leadership uses it to hijack the responsibilities of the Team Leader, the team is headed for paralysis. If you find yourself working with a team that can't seem to get along with itself, or make decisions, or to share responsibility, you should immediately expect that they are suffering from a crisis of multiple leadership.

The Fictional Organization Chart

In most companies, you can get a map of the management structure by asking people, "Who do you take orders from?" and the map you derive will usually follow a classic organization hierarchy like this:

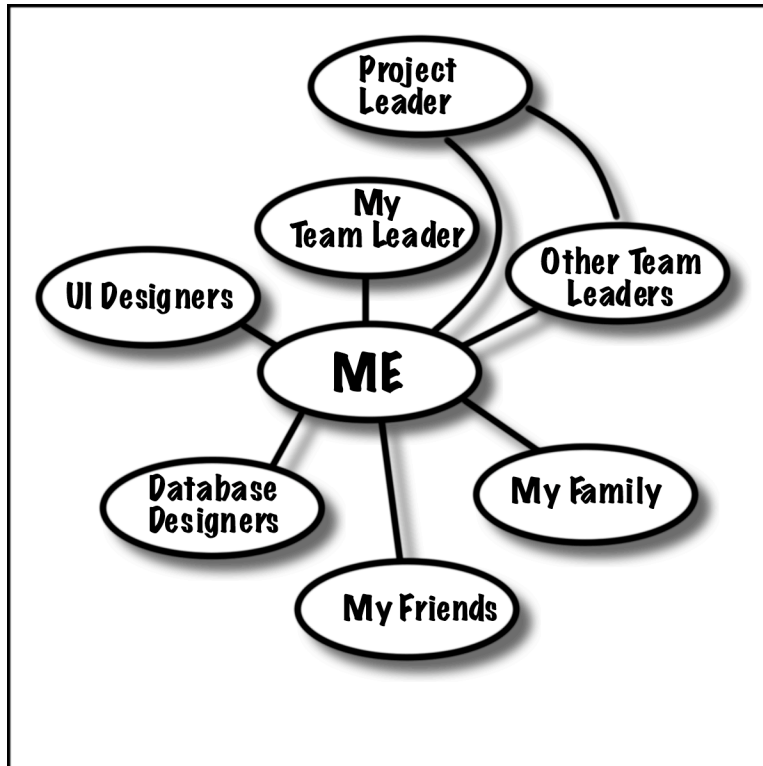


But if you ask the same question of a programmer, you get a very different answer. The programmer might say,

“Well, my Team Leader tells me what the schedule is and when I have to get my piece done, but I have to work with the database designer and he tells me what SQL statements to use for database access. And I m working with the product manager for this feature and she tells me what the customer wants, and so on...”

So from the programmer’s point of view, she takes direction from many people at once, and she is responsible for integrating all of their orders into one plan, and for dealing with inconsistent directions. It can get

very confusing, but successful programmers learn to deal with multiple managers as best they can.



If you reflect for a moment that everyone in the company has a map of responsibility like this, you'll see that the formal organization chart is just a myth. The company (or at least the engineering department) looks more like a network of people with responsibilities to each other. A lot of a programmer's effort goes into managing the network and making sure that the balance of responsibility is stable.

For Those In a Hurry

1. Team Leaders are active programmers who also participate in management. They have a lot of responsibility but not much authority.
2. Project Leaders are the lowest level of “real management” and are responsible for several programming teams. It’s a tough job and they burn out quickly.
3. Programmers think about “upper management” as little as they can.
4. Management isn’t the same thing as leadership.
5. Leaders are made by formal authority, by technical expertise, or by long tenure.
6. Leaders have to know when to step forward and when to relinquish the lead.
7. Managers tend to believe in a hierarchical “organization chart.” Programmers don’t.
8. Programmers see their responsibilities as a complex network of people that they take direction from and people they direct. The network pervades the company.