Some Thoughts on Management

By Nicholas Dwork, 2012

In my most recent employment, I was tasked with leading a relatively large project. The result was a success. In a live field demonstration, we were able to integrate our software easily, and our product outperformed the others that were present. Several new opportunities arose from our work on this project, and several beneficial relationships were made.

At the time that this project began, I had relatively little management experience. I did some things very well, and some things poorly. As you can imagine, I learned a tremendous amount. What follows are practices that I would like to follow if I were to manage a project again.

Hire good people – You're going to be spending lots of time with these people. You need them to carry their weight and make required contributions. Make sure that the people on your project are capable of doing so.

Break off big chunks of work – Distributing work is a difficult thing to do. You have to have the foresight of dividing up a task into chunks, and then you have to relinquish control of those chunks to your employees. At the beginning of the project, I divided up tasks too finely, and people would often finish those tasks in a matter of hours. This consumed my time, since I had to keep coming up with tasks. And it halted their progress since they had to come to me for new assignments. I realized this rather quickly and divided up the project into large major tasks. This allowed for faster progress with less time spent communicating.

Communicate tasks well – Communicating assignments to someone can be a difficult thing. **Specify the inputs and the outputs.** If you can't specify the inputs and the outputs, then you haven't assigned anything. Communicate the task verbally. Follow up that communication with an email.

Keep track of project progress – Keeping track of who is assigned what is somewhat difficult. I did this using a list on my office's whiteboard. This was a pretty bad way to go. A much better way would have been to have a central location where everyone could see the tasks that they were assigned. A spreadsheet on a central server would satisfy this. I recommend asana.com, a web tool that I have used on my latest project with great success.

Rely on the people you hire – Relinquishing control can be a difficult thing. If you've hired good people and communicated the assignment well, then you can rely on your personnel to do their job. Anything else would be interference.

Let people do what they're good at – Everyone has positives and negatives. As much as possible, let people do what they're good at (or what they want to do). Forcing people to do what they're bad at will make people upset and yield poor results.

Don't dictate. Agree. – Rather than dictating through the project, discuss your plans with your team.

It's important that they agree with the team's mission and vision. If you've hired good people and you can't convince them that your idea is good, then it probably isn't.

Remove people from your project who aren't working out – A former successful CEO once told me "The only regret I have about the people I fired is that I didn't fire them sooner." You may have a tendency to be "a nice guy" and keep people on your project who aren't contributing. (I do not suffer from this affliction.) It is remarkable how much one incompatible employee can drag down a team. Eliminate them from the project as soon as you're reasonably sure they're not contributing. Good things result.

Have a dedicated project manager – At the beginning of the project, I found that progress was significantly stifled since I had to attend to management details often. I hired a person to be the Deputy Project Manager for this project. Suddenly, deliverables were getting delivered, the budget was getting tracked, and customer communications were handled expeditiously without any effort on my part. This luxury allowed me and the rest of the team to focus on the science. And, since she was very good at her job, it was not very expensive to have this wonderful luxury.

Have weekly meetings – There's something magical about weekly meetings; without them, work doesn't get done.

Make your meetings short – Meetings are expensive; make them short. The purpose of the group meeting is to review upcoming deliverables, get a quick status update, budget update, make sure everyone knows who needs to talk with whom, and dole out assignments. Brainstorming, technical exchange, software architecture discussions, and all else should be discussed only by those who are relevant for those discussions. Our weekly meetings lasted less than a half hour.

Setup batch processing early – The first step of a project should be to setup batch processing. Create an infrastructure that reads in the data, passes it to a skeletal main algorithm, and writes out the results. Store the results for each commit of the software; you must know what version of the software exactly generated each set of results. Any algorithm changes should improve results.

Start with the simplest algorithm possible – There's a whole bunch of neat math and ingenious techniques floating around. I had the tendency to try to incorporate these neat techniques rather than starting with the simplest possible algorithm. Start with the simplest possible algorithm, and then iteratively improve it, verifying those improvements with the batch processing you've got setup.

Have a Plan - Your plan should consist of monthly goals for each of the metrics you're using to evaluate the product (e.g. processing time, probability of detection, probability of false alarm, percentage classified correctly). Your plan should describe how you're going to accomplish each goal (e.g. Reduce processing time to 1 sec per frame by porting code from IDL to C). Descriptions and goals are very broad, not specific. Then, use batch processing to measure your metrics and compare your expected outcomes to your actual. This will reveal your weak points. Use this information to modify your plan and guide your work.

Do algorithm development in IDL, Matlab, R, or Python – If you find yourself developing algorithms in C, C++, Fortran, or GPGPU programs then something has gone terribly wrong.

Focus on the current problem – One can imagine all sorts of failure modes. I found that many of my imagined failure modes never materialized. Instead, I was plagued with all sorts of issues that I never could have predicted. Use your batch processing to identify what the biggest problems are. Once identified, you and your team should attack those problems and no others.

Have senior advisors – When you find that you're stuck on a technical problem, make a presentation for a senior technical advisor. Have them review your work. Twice on this project we found ourselves completely stuck; in both cases a senior advisor realized that what we were trying to accomplish was impossible without additional information. Periodically (approximately once a month or once every two months), discuss the status and plan of your project with a senior manager.

Respect your customer – It would have been very easy to blame the non-present customer for problems at several parts of this project. It was useful to remind ourselves that our customer was a group of very capable and bright people focused on solving problems for those who put their lives on the line for our nation. We maintained good communication with our customer throughout the whole project. And when we had requests, questions, concerns, or issues, our customer was willing to listen and accommodate us.

Stay Calm – Management is like Starcraft. For those who haven't played Starcraft, it's a game where you and your opponent each build armies and then attack each other. While building your army, you can't see your opponent's area of the board. It is highly stressful. The entire time I play, I worry that I'm not allocating my resources properly, that I'm not accomplishing tasks in the right order, and that my opponent is getting it all right. I imagine that the shaded area of the board is consumed by my opponents army who is about to destroy me. In reality, my opponent has an army of capability and size about that of my own.

Listen – Make yourself available as often as you can for your team. Allow people to contact you in whatever way they feel most comfortable doing so.

Above all, listen! – Listen to your team members, your supervisor, your vice president, your contracts colleague, your customer, your chief scientist, and anyone else who thinks what they have to tell you is important. If they're taking time out of their day to communicate it to you, it probably is.