Being a Good Teammate

If you're like most people, much of your professional life will be spent working on teams. Why? Because most projects that are economically interesting are too large and complex for individuals to do alone. Teams of talented people can accomplish great things that seem almost impossible, such as sending a man to the Moon. Such objectives would obviously be impossible if we had to work alone. Therefore, it is important that you learn to work effectively as a team member.

While teams allow us to accomplish far more than we could alone, there are also significant downsides to teamwork. As the number of people on a team increases, so does the amount of communication and coordination required to keep the team organized and moving in the same direction. Meetings, emails, phone calls, etc. increase, and these things take time. Also, the more people you add to a team, the more likely it is that at least one of them will be unreliable, antisocial, cranky, lazy, distracted, uncommitted, or otherwise difficult to work with. Such people take more energy from a team than they contribute, and often the team would be better off without them. You should try very hard to not be one of these people. There are specific things you can do to make yourself easy and enjoyable to work with. In the context of this class, here are some important things you can do.

Be reliable. Be someone your teammates can count on. Do what you say you will do. Attend team meetings. Complete the deliverables you've committed to produce (design documents, source code, etc.). Don't be the one your teammates talk about behind your back as the one who just isn't getting the job done.

Be early. Be on time to meetings. Get your part of the design document done early so your teammates will have time to review it and give you feedback. Get your code written and checked into Subversion early so your teammates will have time to see what you've done and integrate their code with it well before the deadline. If you get things done at the last minute, it will be very hard on your teammates, and your reputation will suffer. Part of getting your piece done is getting it done early.

Be prepared. Prepare well for team meetings. Read the specs before the meeting. Think about how you would solve the problem before the meeting. Have your part of the document done before the meeting. Whatever the purpose of the meeting is, make sure you are prepared for it. Such preparation will allow you to be a productive contributor rather than a drag on the meeting. If you show up unprepared, meetings will be longer because your teammates will have to bring you up to speed before the productive part of the meeting can begin.

Be responsive. Communicate with your teammates. Send frequent emails letting them know the status of your work. Send them your documents so they can review them. Check your code into Subversion early and often (every day or two) so they can see your

progress and integrate with your work. It will make them feel a lot better if they can see evidence that you are making progress and getting the job done.

Don't disappear for extended periods of time without explaining to your teammates what happened to you. If your wife is going to have a baby, tell them. If you get sick, tell them. If you have to focus on another class for a few days, tell them. Don't just disappear for a week without explaining to your teammates what is happening. If you tell them what is going on, they will understand completely. If you just disappear, they will see you as unresponsive and unreliable.

If one of your teammates disappears without explanation, talk to your instructor quickly before the situation gets out of control.

Be careful. As stated above, check your code into Subversion early and often. However, don't check in code that doesn't compile. That will just make your teammates mad. Before checking in new code, do an "update" first to make sure you have all code that has been previously checked in by others. After doing an "update", do a clean build to ensure that everything still compiles successfully. Once you know everything compiles, run all of the unit test cases to ensure that your new code hasn't broken the unit tests. Once you know that everything still compiles and that all unit tests still work, then you are ready to "commit" your changes to the repository.

Be willing. Be eager in accepting assignments. Do your fair share of the work. Don't always be the last to volunteer. Don't let one or two members of the team do all the work.