CHAPTER 5: WORKING AS A TEAM

You can't have great software without a great team, and most software teams behave like dysfunctional families. --Jim McCarthy

Unlike a typical project team in the industry,

- Your team is a team of peers and a team of novices. While your abilities may vary, none of you is experienced software engineers or managers.
- The size of your team is usually fixed and small.
- You cannot usually hire and fire team members during the project.
- You cannot outsource work.
- You have no access to other expert groups such as QA team, Graphics team, Technical writers, Business Analysts, Legal team, etc.

Here are some tips to help you overcome those limitations.

5A. TEAMING UP

5.1 THE TEAM MATTERS. TAKE IT SERIOUSLY.

If you are allowed to choose your own team members, do not take it lightly. Try to assemble the best team you can - it certainly doesn't hurt to have a great team.

5.2 EXERCISE 'DUE DILIGENCE'

When someone offers to join your team, try to find out whether he has done a good job in previous course projects. Some even hold interviews or ask applicants to answer a questionnaire before selecting team members. You can also check for things such as the workload, extra curricular activities, and external commitments that person is planning to take. This is to ensure that he can commit enough time to the project. Schedule and location compatibility matter, too. For example, teamwork is easier when team members have similar schedules and live close by (e.g. in the same student residence). It is also ideal if all team members have similar levels of expectations on what they hope to achieve; things do not go well if one person is trying to score an A+ while for another a B- is 'more than enough'.

Be mindful that some students are committed, loyal and pleasant to work with, but lacks the 'hard' skills required for the project while some others may be 'top of the class' high fliers lacking in the 'soft' skills necessary for team work.

5.3 BALANCE SKILLS

Most projects require a good combination of various skills. Form your team accordingly. If the software you will build requires a good GUI, try to find someone with good GUI skills. If it is going to be a game, it will be good if at least one person in your team is a 'gamer'. In particular, you need NOT pick only those with good programming skills. Still, programming skills are critical to the project - try to pick at least one or two good programmers.

5.4 LOOK FOR TEAMMATES, NOT BUDDIES

A team out of your best buddies has many advantages, but there is a downside, too. For instance, you will find it hard to maintain a professional atmosphere in your project work. Furthermore, you may be forced to put up with low contributions from some of your team members for the fear of hurting the friendship.

Unfortunately, we have no advice on how to dodge friends who want to join your team for a free ride, except to say that they are probably not worthy of your friendship.

5.5 ASSIGN ROLES

A project needs expertise in different areas, including areas none of you are expert in. Assign at least one team member to become the 'guru' of each such area. The guru's role is to specialise in a given area: to learn the area in-depth, help other team members on matters related to that area, and ensure sufficient quality in your product with respect to that area. For example, the testing guru is responsible for tasks such as looking for good testing tools, defining testing policies, test planning, helping others in writing test cases, troubleshooting test tools, ensuring adequate testing is being done, etc. Put differently, it is the testing guru's responsibility to earn the team as many points for 'good testing' as possible. Other possible guru-type roles include SCM guru, Documentation guru, Build guru, Integration guru, Visual studio guru, C# guru, Performance optimiser, Design maintainer, Deadline watcher, etc.

Note that every team member should have basic knowledge of all aspects of the project, not just the area they are guru of. For example, the testing guru cannot limit himself to testing only.

The idea is to become a 'Jack of all trades, and a master of *at least one*', i.e. to acquire a so-called 'T-shaped' skill set. Here, the horizontal bar of the T represents the breadth of knowledge in many areas i.e. 'jack of all trades', and the vertical bar represents the depth of knowledge in some areas i.e. 'master of at least one'. Such a skill set, said to be in high demand in the job market, should greatly enhance your employability.

Note that all of you are novices to begin with, but everyone can become an expert in a given area if they spend enough time on it. Choosing an area you like will definitely make the job easier.

5B. LEADING A TEAM

5.6 HAVE A LEADER

The 'Everyone will share the leadership role' approach, without explicitly assigning the leadership role to someone, rarely works in a student project. It is highly recommended that someone assumes the role 'team leader'. The team leader role does not put the person holding it *above* others in the team. Treat it simply as another role one has to play in the team, albeit an important one. Having an assistant leader is another good step in this direction.

5.7 ROTATE LEADERSHIP

Leading a team of software engineers is no easy task. It has been likened to "herding cats" (in the book The Pragmatic Programmer).

Playing the team leader role is a valuable learning experience. Ideally, all students in a team should get this experience. If more than one person is willing to try being team leader, consider the following strategy: after completing each major milestone, the current leader steps down if another team member volunteers for the team leader role. This should not be considered 'challenging' the current leader; rather, it is saying: "that looks like something worth learning. I want to try it, too". Rotating leadership is especially useful for projects lasting more than 2-3 months. However, it is OK if you want to keep the same leader for the whole project.

5.8 DELEGATE AUTHORITY

Another way to lessen the burden of the team lead and spread the leadership experience around is to have *group* leaders for each sub-group. The *team* leader can be one of these group leaders, or a separate person altogether. If the latter option is chosen, the team leader will only make high level decisions that affect the whole team (e.g. decisions related to system integration), while he will be working under the respective group leader at other times.

5.9 STRIVE FOR CONSENSUS

A team/group leader is generally expected to consider opinions of the other team members and achieve a consensus before making decisions. However, there will be times when the leader has to make quick and hard decisions on his own, either due to time pressure, or due to irreconcilable differences of opinion between team members. A team/group leader can also consult the supervisor to get one more opinion before making decisions. Once a decision has been made, others should respect this decision and try their best to implement it.

'HOW TO' SIDEBAR 5C: HOW TO FIT INTO YOUR TEAM

Here are some tips on how to deal with "hey, am I part of the team the team or what?" situations.

5.10 MY TEAM OVERRULED ME!

Problem: Your team took a decision against your opinion.

Recommendation: As the saying goes, "there is more than one way to skin a cat"; while the team's decision may be different from what you believe to be the best way to proceed, it might still lead to a reasonable outcome. Do your best to help the current course of action succeed. Do not try to sabotage it or distance yourself from the work. You can also persuade the team to get the supervisor involved in evaluating alternatives. Finally, make sure that the decision point is well documented, as such decisions make an important part of project documentation.

5.11 MY TEAM CUT ME OUT!

Problem: You want to contribute, but the team appears to cut you out (e.g. when the other members have a history together, and you are the 'outsider').

Recommendation: Voice your concern to the team (e.g. "I worry that I'm not pulling my share of the load. Can I have more work?"). Alert the supervisor if the situation does not improve. Above all, do not delay. This problem surfaces in the early stages of the project and it should be solved in the earliest stage. It is your responsibility to notify the supervisor early if the the team does not respond to you positively. If not, you might get penalised for not contributing enough.

5.12 I AM A 'SPARE WHEEL'!

Problem: Your team is doing fine without you. They do not mind your lack of contribution. Their attitude is "stay out of the way, and we'll let you share our success for free". They give you good ratings during peer-evaluations although you did not do enough to deserve it.

Recommendation: If you go along, you are taking a BIG risk. There are ways other than peer-evaluations to gauge your contribution. (e.g. your SVN/CVS activity statistics). It is your responsibility to do a fair share of the work. Most evaluators will penalise you for being a spare wheel if you do not alert them early enough. Follow the recommendations in tip 5.11.

5.13 I LABOURED IN VAIN

Problem: Someone else has done (without telling you!) the work formally assigned to you.

Recommendation: If you did your part on time and to the required quality standards, you have a right to insist that it is used in the product. However, note that this is usually a symptom of other problems (e.g. others have lost confidence in your work). Alert the supervisor so that these problems can be sorted out.

5.14 LOST IN TRANSLATION

Problem: Others are from country X except you. Most team discussions are done in their own language.

Recommendation: This is unacceptable. Ask others to use a common language at all times. Alert the supervisor *immediately* if they do not agree. Not alerting the supervisor early enough and using this problem as an excuse (i.e. "I didn't do much work because I didn't understand team discussions") at the end of the project will not get you much sympathy.

5.15 I LAG BEHIND

Problem: You are slower or not as skilled as others in the team.

Recommendation: No team member is perfect. You may be weak in programming while you still may be good in some other aspect. Do not make your weakness an excuse for a free ride. If your teammates are much stronger than you, do not expect an equal share of the credit they earn for the team, even if you work as hard as them (it is not about how hard you work; rather, it is how much you contribute). Accept tasks that you can deliver and then, deliver to the best of your ability. As you learn more, their confidence in you will grow.

'HOW TO' SIDEBAR 5D: HOW TO HANDLE PROBLEMATIC TEAM MEMBERS

A 'jelled' team is a team functioning smoothly - like a well-oiled machine. Unfortunately, student teams do not jell that often. Problems within teams are common, and working them out is part of the learning experience. This means 'team problems' is not a valid excuse for making a mess of the project.

When a team member is causing problems within the team, we should consider that this may be unintentional. Therefore, it is important that he is made aware of the team's concerns. Keeping silent will not solve the problem. Rather, it will deprive this person of a chance to mend his ways. Do this at the earliest possible moment, but discuss the issue informally and gently. Firing off an 'official' email to everyone and CC'ing the supervisor should not be the first thing you do. However, if the situation does not improve within a short period (say, within a week), it may be time to alert the supervisor. While you cannot blame everything on bad team dynamics, it pays to keep your supervisor informed about problems within your team. This could prevent the whole team getting penalised for one person's actions and could earn extra credit for 'overcoming team management challenges'.

Given next are some undesirable personalities we sometimes see in project teams. Insist that everyone in the team read this list. Read it yourself and see whether you fit any of them. If you do, your team mates probably realise it too, and it is up to you to change for the better. Note that the uncharitable name we have given to each personality is just for ease of reference. Please do not use them to belittle your teammates.

5.16 ABDICATOR

Problem: A team member who delegates *everything*. He acts as the 'management', and treats others as employees!

Recommendation: This is unacceptable. Effective delegation is a part of being a leader; but everyone, including the leader, should do a fair share of the work in all aspects of the project. Alert the supervisor. If the leader continues to evade work, change the leader at the next milestone.

5.17 TALKER

Problem: A team member who talks a lot (especially, if the supervisor is present) and acts like the most active member of the team. But that is about all he does.

Recommendation: Active participation during discussions is good as long as it does not stifle/overshadow other team members. What is not so good is dominating team meetings just to cover up lack of real contributions. Everyone should do a fair share of the work in all aspects of the project. If not, try to sort out the issue with the offender. Alert the supervisor if the situation does not improve soon. Formally keeping track of contributions can deter this type of behaviour.

5.18 LONE RANGER

Problem: A team member who does not abide by collective decisions, or does things on his own way without bothering about getting the team's consensus first.

Recommendation: Let the supervisor know if reasoning with him does not work. However, note that working in a team does not mean every decision has to be made collectively. E.g. if a certain decision's impact is local to a component, let the person in charge of that component decide whether to make the decision individually or collectively. On a related note, you are not helping the team by blowing up trivial decisions (local to your work) into a team discussion.

5.19 DOMINATOR

Problem: A team member who tries to micro-manage everything. He dismisses others' ideas and insists on doing things his way.

Recommendation: If you do not feel comfortable showing your displeasure to the dominator directly, inform the supervisor about the situation and ask his help to balance things out. It is easy for a supervisor to contact such a person privately and 'encourage' him to let others have a more active role in the project.

5.20 SUFFERING GENIUS

Problem: A team member who thinks he is more 'experienced' than the rest and thinks he has no time to waste with 'amateurs'. While he is willing to do a major chunk of the work, he does not want to get involved in anything else such as team meetings. (E.g. "I did the component X, didn't I? That's the most difficult part; you guys sort the rest out and leave me alone!")

Recommendation: This is unacceptable, particularly from a person who thinks of himself as 'experienced'. This behaviour is going to hurt the team no matter how good the code he has produced. Such code usually makes many assumptions the rest do not know anything about (because there is little interaction between the team and this person), and is inflexible in ways that hinder the future progress of the project. Let the supervisor know if the team cannot get this person to co-operate. Teamwork is an essential part of a project course. Those who fail to work as a team should be penalised, no matter how much work they do alone.

5.21 BUSY BEE

Problem: A team member who cannot commit enough energy to the project because he has a busy schedule (extra curricular activities, other courses, competitions, etc.)

Recommendation: As fellow-students and teachers, we should do our best to support this person to succeed in all the things this person is engaged in. However, a team cannot jeopardise the project due to this. If possible, adjust schedules/plans, etc. to accommodate this member's availability. You can also allocate this person less work (as much as he is willing to take, and able to deliver) and keep the supervisor informed about this imbalance of workload. The grade needs to be adjusted accordingly.

5.22 PERSONAL ISSUE GUY

Problem: A team member who is hit with an unexpected personal issue (death of a loved one, financial/health/relationship problems, etc.) that prevents him from contributing equally to the project.

Recommendation: We should do our best to support this person in this moment of personal misfortune. If possible, adjust schedules/plans, etc. to accommodate this member. You can also allocate this person less work (as much as he is willing to take, and able to deliver) and keep the supervisor informed about this imbalance of workload. The final grades will be adjusted accordingly and as per the institute's guidelines.

5.23 UNWILLING LEADER

Problem: A team leader who just sits around and pretends he is just a team member instead of doing 'leader stuff'. This makes the whole project effort unco-ordinated.

Recommendation: Some people are not natural leaders. Change the leader at the next milestone.

5.24 GREENHORN

Problem: A team member who is inexperienced and lacks almost all skills required for the project. Helping this person appears to slow the team down.

Recommendation: No matter how 'green' this person appears to be, it is well worth spending time to help him learn the skills needed for the project. Always try to work in groups during the initial stages. Even coding can be done together if you use pair programming [http://tinyurl.com/PairIntro] (highly recommended technique to improve coding skills). It is a mistake to cut this person out, make him a 'spare wheel' [see tip 5.12] , or only assign trivial tasks to this person; you need all members of the team to contribute fully for the project to succeed.

5.25 WEAK LINK

Problem: A team member who is really weak/slow, and has a track record of ending up at the bottom of the class. This is different from inexperienced team members [see tip 5.24].

Recommendation: We have to learn to make the best use of 'below average' team members. Do not make them 'spare wheels' [see tip 5.12] or assign them trivial tasks. Instead, assign non-critical tasks that match their ability to deliver. Your peer-evaluation should reflect the correct level of contribution from this person. But remember to give him a chance to contribute to the best of his ability. By the way, it is a big mistake to appoint the weak member as the tester of the system.

5.26 CHEATER

Problem: A team member who suggests (or appears to be) using unscrupulous means to do project work, such as plagiarism or outsourcing.

Recommendation: RED ALERT. The whole team will be held responsible in the end. Inform the supervisor immediately.

5.27 SLIPSHOD

Problem: A team member who continues to deliver work of low quality.

Recommendation: Insist on better quality work and refuse to incorporate low quality work into the product. You can use cross-testing (i.e. components being tested by others in the team, in addition to testing by the authors) to reduce the risk of this low quality work affecting the overall quality of the product. You can also assign earlier deadlines to this person to allow enough time for rework if needed. If the quality does not improve over time, keep the supervisor informed so that grades can be adjusted accordingly. If the course uses peer-evaluations, you can use it to reflect the low contribution from this person (i.e. low-quality work should not be counted as contributions, in fact they are negative contributions). Cutting the person out of the team is not the answer [see tip 5.11].

5.28 HERO

Problem: A team member who insists that he will take care of a critical task. He is unwilling to discuss details of how it will be done or the current status of the task.

Recommendation: This may be OK if this team member has a track record of delivering on his promises. However, in general, this is a risky situation. If the project fail because one person promised something but never delivered, the whole team will be penalised (as they should be). Make status reporting compulsory. Insist on incremental delivery. Do not assign critical components to someone before testing out his delivery quality. If in doubt, make critical components joint-work rather than entrusting them to a single person.

5.29 FREELOADER

Problem: A team member who tries to exploit others' (often, those who are good friends of the offender) goodwill to let him share the credit without doing sufficient work. Often, such a person has some 'legitimate' reason for not being able to work, and wants the team to 'help' him out. Freeloading is also called 'social loafing'.

Recommendation: If the team members go along, they will have to lie about the share of the workload during peer-evaluations. The whole team will get into trouble when the imbalance of workload is noticed by the supervisor. Please give correct information about the contribution of each team member. You are being unjust to yourself and other course-mates if you do not.

5.30 OBSERVER

Problem: A team member who does not take an active interest in the project. He keeps silent during meetings and does not contribute any ideas. However, he does all tasks assigned to him.

Recommendation: This is not a big problem as long as you do not have too many such members. If this is due to shyness or for the fear of being overruled, you could entice a more active role from this person by assigning him more responsibility. You can also assign tasks that force this person to participate more actively. For example, assign him to "come up with an alternative design and describe it in the next meeting".
Adopt the practice of recording each person's contributions including ideas they contributed. This could coax the observer to be more active in the project. However, if this person is deliberately shirking responsibilities, then it is a serious matter akin to freeloading [see tip 5.29]. Do not blindly equate the observer's lack of involvement to his support for what the team is doing. The rest of the team should continue to seek any alternative opinions the observer might have.

5.31 LAID BACK JIM

Problem: A team member who is less committed to the project than the rest. While you are working very hard for an A+, this person is 'doing just enough to get a pass'.

Recommendation: While this is frustrating, there is no easy way to motivate this person to work as hard as you are doing. Assign as much work to this person as he is willing to take. However, it should be clear that his level of contribution will be reported in peer-evaluations. In addition, whatever this person does should be of the expected quality. Doing less is not the same as doing a shoddy job [see tip 5.27]; the latter is much more damaging.

'HOW TO' SIDEBAR 5E: HOW TO DEAL WITH TEAM DYNAMICS ISSUES

Here are some tips on how to deal with problems related to how the team works together.

5.32 TUG OF WAR

Problem: The team is highly polarised over a critical decision.

Recommendation: It is healthy to have alternate views, be passionate about your views, and be prepared to defend your views. But protracted arguments can actually hurt the team. Try to find ways to objectively evaluate alternative views (e.g. implement a prototype of both alternatives and measure which one works better). You can add the supervisor's opinion into the equation as well. No matter which alternative you choose, it is important that everyone tries their best to implement the choice made. Furthermore, do not forget to document the decision.

5.33 LIFELINE LEADER

Problem: The team relies on the leader (or another dominant member) to do all decision making ("You are the leader. You tell us what to do!").

Recommendation:

For the team members: You are in this as much as the leader. You should take responsibility for the success of the project as much as the leader. The leader might not see all the alternative paths; it is up to you to point them out.

For the leader: Delegate. Assign a role to each team member (see tip 5.5).

5.34 RANDOM TEAM

Problem: All team members are strangers to each other, preventing the team from achieving the flying start the project needs.

Recommendation: Well, the only possible remedy is the get to know each other, fast! See SIDEBAR 5F for some tips on how to promote team spirit.

5.35 NEGATIVE-SYNERGY TEAM

Problem: The team spends long hours working together, sometimes late into the night, yet accomplish very little. Time is often wasted on casual conversation, teasing, flirting, joking, or arguing. No one is willing to break away from this pattern for the fear of hurting the 'team spirit'.

Recommendation:

- Working together means working in close proximity so that one team member can help another when needed, and issues can be resolved quickly. It does not mean you spend a long time 'talking' together. Programming does not mix well with intermittent distractions. It takes 10-20 minutes of undisturbed work for a programmer to attain an optimum level of productivity (seasoned practitioners call this 'going to the zone'). A single distraction can snap the developer out of this zone. Next time you are about to disturb your team mates for 10 seconds with a joke, think about how much time you are actually wasting.
- Often, it is enough to work together in sub-teams, not the whole team together.
- Identify and deal with distracters [tip 7.14] in your team (gently, of course).
- Commit to an agenda. E.g. Let us work together for one hour and integrate the version V0.2.

5.36 CLIQUE

Problem: Several team members have formed a closed group (i.e. a 'clique'), alienating everyone else. For example, when one of the clique members gets into an argument with a non-clique team member, the whole clique comes to the defense of the clique member, although the discussed issue does not concern them.

Recommendation: If possible, let the clique become a sub-group, and let them work together on a sub-system. This should reduce friction between the clique and the rest. If the clique continues to undermine project harmony, get the supervisor involved to balance things out. For example, if you think the clique is forcing non-optimal decisions on the team, get your supervisor's opinion on the decision as well.

5.37 HERD OF CATS

Problem: Each team member does what he wants. No one follows the directions from the team leader, or there is no clear leadership.

Recommendation: Such behaviour results in problems such as duplication of work, integration problems, and in general, a messy product. In addition, most instructors will explicitly penalise such unprofessional behaviour. Appoint a strong leader and co-operate with him or you will all lose out in the end.

5F. INCREASING TEAM SYNERGY

Working *in* a team is not the same as working *as* a team. A synergistic team's output should be greater than the sum of what each member could have done on his own. Here are some tips to increase your team's synergy.

5.38 BUILD TEAM SPIRIT EARLY

As soon as the team is formed, make it a point to spend some time together. This lets you get to know your team members quickly. Even small adjustments like having lunch together or going for a movie together can help in this area.

Doing early project activities together strengthens your team spirit further and helps you gauge strengths and weaknesses in each other. Consider the task of picking a project topic; instead of emailing back and forth about potential topics, meet together and talk it over face-to-face. At early stages, pick small and easy activities for the whole team to do together quickly. This will boost your confidence in your team.

5.39 BUILD A TEAM IDENTITY

Building an identity for your team can help bring the team together. Here are some things to try:

- Take a 'funky' team photo (Examples: http://tinyurl.com/teamphoto1, http://tinyurl.com/teamphoto2).
- Set up a web page for your team (Examples: http://tinyurl.com/codedroids).
- Brainstorm for a catchy code name for your team/product.

5.40 BUILD POSITIVE GROUP NORMS

Groups norms are mutual understandings as to "the way we do things in our team". Try to build positive group norms. E.g. "we always come to the meeting on time", "we never submit code without unit testing first", "No texting during meetings", etc.

5.41 KEEP IN TOUCH

When a team member is 'unreachable' for an extended period of time, the usual excuse given is "oh, I don't check *that* email account" or "I was using my *other* phone that day". As a team member, it is your responsibility to keep the team informed about how to reach you.

However, it does not mean that you have to be reachable all the time. It is OK to be unavailable for project work for acceptable reasons as long as you keep your team informed of your unavailability. For example, "I don't check mail during 11pm-8am" seems quite reasonable. You can also 'take leave' from the project (e.g. to go on a trip) as long as you inform the team early and your absence is officially incorporated into the project schedule. In addition, try your best to minimise the impact of your absence (e.g. finish your assigned tasks *before* leaving, rather than postpone them).

5.42 WORK TOGETHER, WORK ALONE

Some things are better done as a team while some others are better done individually. Often, you can start something by working together and then split the work and carry on individually. But maintain a healthy interest in

what others are doing. It is also good to review work done by others at milestones, especially work done individually.

Be careful of how you tread on another member's work space. For example, be tactful when you refactor code written by others. It is not recommended to refactor others' code, without their consensus, to suit your coding style.