Chapter 3

Essentials.

Summary: Group work and software projects. How to set up a team. Carrying out a skills audit. Choosing a way of working. Finding and keeping a client. Day to day activities. Keeping an archive. Some basics of planning. Dealing with problems. When things go wrong. Risk analysis.

1. Software engineering in teams.

Almost all software that is produced commercially is developed with teams of people. The teams might be structured into programmers, testers, requirements engineers etc.. It probably has some hierarchical arrangement with managers, team leaders, sub teams and so on. The team could be a small one, perhaps 2 or 3 people or it could involve hundreds. The team may all be working in the same place or it could be scattered over different locations, countries, even. What is common to all these manifestations is that they share a general objective, the production or development of some software product.

Learning how to work effectively in a team is thus a vital part of one's education as a software engineer. Many universities and colleges provide some place in the curriculum where a team project is set up and you have to participate with colleagues in a design project. In many of these activities the professor or instructor will set some problem and you try to solve it, learning along the way from the many experiences you share, good and not so good, which relate to the way your team worked.

There are many sources of advice about how to make the most of a team but there are no easy rules or procedures. A team comprises of a group of distinctive and independent personalities, we cannot generalise very easily about how these personalities will interact and how the team will progress. However, there are some simple basic rules which seem to work and the purpose of this short chapter is to describe these.

2. Setting up a team.

This may not be an issue since your instructor may allocate you to a team without providing any choice in the matter. This is a reasonable reflection of what happens in industry so one can't really complain. However, if that is not the case and you are asked to form yourselves into teams here are some pointers to doing that.

Suppose that we are trying to form a team of 5. We need to look for a blend of personalities and skills that will knit together and produce an effective force. The nature of the project may determine some of the parameters but let us assume that all the potential candidates for the team are reasonably well prepared in terms of having progressed satisfactorily through the programming, design, and more specialised courses needed for the project.

The key requirement is for the team to be people who get along reasonably well with each other, who can meet in suitable places and who all have a similar interest in doing well in the project - partly because of the desire to get a good grade in the exercise.
A software project involves a number of key activities and it is important that there are members of the team who can make useful contributions to these activities. We obviously need some

good programmers but there are many other things to be done in the project, we need people who have abilities to organise, to plan, to negotiate and communicate - with, for example, the client - and there is always the need to document clearly and systematically various important things.

No single person will come to the project with all these abilities to a high level but most people will be capable of most to some extent. Extreme Programming emphasises the equal involvement of all team members in all the important activities and so the project will be a framework within which all team members will develop significant skills across the board. You will learn both technical things as well as how to co-operate, communicate, organise, resolve problems, deliver a successful product and mature as a software professional in a way that is just not possible in other types of learning.

Doing a *real* project with a *real* business client will change you for ever, your perspective on life, on your colleagues and on the process of working together. Your understanding of the profession of a software engineer will be transformed, as will your job prospects, since future interviewers will be really impressed by your experiences in doing real software development, it will set you apart from the rest of the applicants as someone with extra skills and experiences and value for their business.

A skills audit is an important feature of any team building exercise. It may only happen in an informal way, you gradually learn what your colleagues know and can do. It is best, initially, however, to try to write down what your strengths and weaknesses are and to share this with the rest of the potential team. See if there are people with a good selection of the skills needed. If most of your team are good at programming but not very good at talking to people or organising documentation then that should be a cue to try to recruit someone with these skills. The deal is then that the Extreme Programming approach will help them to develop those skills that they are weak in. Extreme Programming is all about multi-skilling and learning all the key skills needed in software development to a high level.

Itemise your groups relevant skills - or at least your own assessment of them in a table like the following :

Table 1:

skill	excellent	moderate	limited
Programming in java	pete, mary	joe, oscar	jane
Communications skills	oscar, joe	mary, jane	pete
Organisational skills	mary	pete, oscar	joe, jane
Documenting skills	jane, oscar	jane	pete, joe
...			

This is only a rough guide and the definition of the skills and levels is bound to be vagues but it does give you some basis to plan out your project and also a simple benchmark to compare with at the end of the project. One would hope that there is a significant improvement across the board by the end of the course.

3. Keeping you client on board.

Once you have a client and a project it is vital that you make efforts to keep them. Regular feedback to the client is essential, so regular meetings must be held. When you attend these meetings make sure that you approach them in a professional way. Think smart and look smart. Give your client confidence that his/her investment will be worth while and they will get something out of the exercise. Never break appointments, if some other crisis occurs it is vital that the client is warned if it is necessary to change a planned meeting.

Always describe what you have achieved since the last meeting. Always appear interested in the client's business, and express some confidence about how the project is going but do not exaggerate progress. Honesty will ultimately pay.
My experience has been that clients really enjoy the activity, many have never been a client for a software development project before and they are getting some useful insights that may be valuable in later years. They also generally like working with bright and enthusiastic young people. So for them it will be both an enjoyable and a productive experience. It should be the same for you.

4. The organisational framework.

We will now assume that you have been allocated to a team or have organised one yourself. We now describe a few simple, and perhaps obvious, things to do. Do not underestimate these factors, many projects fail because of the simplest and most stupid of mistakes and omissions.

Learn as much about your team members as possible, their names, addresses, phone numbers, e-mail addresses and so on. It is vital that you can contact everyone easily because you will be working on the project in a variety of locations, not just the usual laboratories. This is something that is different to most industrial practice where the team occupies the same premises all day and every day. See if everyone will sign up to a working agreement that identifies the responsibilities and expectations of all the team members.

Agree on the location for the first meeting and make sure everyone turns up on time. This is important if one wants to be treated professionally, as your client will want to do, if you do not behave in a professional way why should anyone treat you like a professional? This is the first test, if a team member does not make it to an important and agreed meeting and they do not have a excellent reason, then this is a major threat to the project and to all of the team's grades. The team agreement should emphasise the obligation on all team members to attend all meetings. If the culprit does not listen to reason then complain to the instructor.

Teams can work well in a variety of ways. Sometimes it is worth agreeing on having a team leader who takes over the responsibilities of organising and chairing meetings, of leading the planning and other key co-ordinating activities. If everyone is happy with this solution then this can work. My recommendation, however, is for the role to be shared, each member of the team taking over the running of the team for, say, two weeks at a time. Thus everyone gets an oppor-

tunity to develop their leadership skills and to take responsibility for the team's progress. This is more in keeping with the democratic nature of Extreme Programming.

It is important to establish an effective method of working. First of all you will need to hold planning and progress meetings. Depending on the time scale and your other activities there might be several of these each week. The current project leader should chair the session. There should be another team member to act as secretary - this could be the person who will take over as project leader after the current one. The meetings should be minuted formally. This requires the following information about the meeting to be recorded:

Date;
Location;
Attendance;
Absences (with reason);

and then the record of the meeting.

See Figure 1 for an example of a template that works.

Each item of discussion should be numbered and a brief description of the item made. Any conclusions and decisions taken must be recorded together with any further actions agreed. These must describe:

what is to be done;
who is to do it and
when it must be done by.

All this is absolutely vital if the project is not to suffer from confusion and recriminations.

Each team should appoint an archivist. This role can also be shared around the team. The key requirement is that someone is given the responsibility to maintain a complete and accurate record of the plans and meetings of the project. This person should set up a suitable filestore on some server where all the team has access so that anyone can consult the archive to see what the status and history of the project is. We will later discuss the archiving of other, more technical documents, requirements documents, test cases, code etc. The regime for these documents is different, however.

Another important activity is the recording of the amount of time each team member spends per week on the project activities. This should be recorded on a weekly time sheet for the team. Examples will be found in a later chapter.

It is vital that we record accurately the time we spend on projects.

Firstly it enables us to track our individual performance and helps us to identify where we are making progress and where we may still have improvements to make as we undertake various types of activity in the software development process. This is vital for apprentice software engineering and, in fact, should be something that we do throughout our professional lives. The Personal Software Process [Humph96] provides an excellent framework for this.

Secondly it will help us to collect data from which we can predict how much effort future activities might take. Estimating the resources - time, people etc. - needed for the development of software is notoriously difficult. Many decisions are made in an *ad hoc* manner and usually lead to disaster or, at best, to a very inefficient and expensive process. We have to learn to do better. As we will see, later, planning is an important part of Extreme Programming.

Minutes of group meetings.

Group no. …….
Date of meeting [dd/mm]…….Time of meeting [hh:mm]……... Place of meeting…….
Present…….
Absent (reason)……...

Agenda item	**Details:**	**Action by:**	**Deadline :**
1			
2			
3			
4			
5			
6			

Figure 1

5. Planning.

The basics of planning include:

decomposing the overall task into a collection of smaller tasks;
identifying the dependencies and relationships between these tasks;
estimating the amount of resource (time and manpower) required to complete the tasks;
setting delivery times for each task;
describing the plan in some suitable notation, eg.a GANTT chart.

Plans will inevitably require review and alteration since the estimates made of the time needed to complete tasks is often wrong, further understanding of the project could lead to a different structure to the previous task decomposition as well as exceptional circumstances, such as ill-

ness, intervening. The regular meetings provide an opportunity to review and re plan the project. Do not shy away from hard decisions in these meetings. It is very easy to pretend that everything is all right when it isn't. Equally one can get depressed about progress. Later in this book we will look at planning again and examine how Extreme Programming can provide some answers to some of the problems met in planning and running software projects.

6. Dealing with problems.

It is inevitable that things will go wrong from time to time. It is the teams that are able to deal with problems effectively that turn out to be successful. It is not about how clever people in the team are but the culture within the team. If this culture is one of co-operation, discussion, building consensus and treating each member as an intelligent individual with a legitimate point of view then resolutions can be found. If people are stubborn, arrogant, dismissive and unco-operative then it is much harder. Try not to loose one's temper, be patient and considerate to others, discuss the issues on the basis of an informed knowledge of the matters under discussion rather than based on prejudice and guesswork. Seek expert advice if the argument is about a technical point, the benefits of different strategies or approaches. Talk to the client as well. All these things can be sorted out if everyone is positive and prepared to give and take. Extreme programming is all about co-operation, communication and treating people with respect and trust. All problems are soluble somehow.

If you really hit a crisis and there seems no way out seek arbitration. Find someone that everyone in the team respects, perhaps a tutor or professor, and explain the issues to them and ask them to make a judgement. This might be a simple compromise that everyone was too uptight to see or it might be a ruling in favour of one side or another.

Try not to be upset if your argument is not the one that is successful in this process. Think about what has happened and see how you might benefit from the experience. Perhaps the way you handled your argument or yourself was counter-productive. Successful people in life reflect on their experiences and learn from them, adapting their future behaviour in order to ensure future success.

++