CSC444F Software Engineering I

Notes on the Software Trading Game

1. Overview of the trading game

The practical work on the course involves a "software trading game". You will be working in small teams (typically 4 students to a team). The aim is for your team to develop a complete working software system by the end of the term, by developing software and trading it with other teams. The entire class is divided into four lab sections, with approximately 48 students in each section. Each lab section constitutes a "trading block": you will trade software only with other teams within your lab section. Within a trading block, all programs shall be written in the same programming language. The choice of language for each lab section will be decided by (majority) vote in the first lab session of term.

You will be given a Software Requirements Specification (SRS) in your first lab session. The specification describes an overall architecture for the system to be developed. Essentially, the system will consist of four modules, or Computer Software Configuration Items (CSCIs). The SRS will describe the requirements for each CSCI. Different teams will be given different CSCIs to work on. With four different CSCIs, and twelve teams in a lab section, there will be three teams developing competing versions of each CSCI.

The course consists of three main phases. In the first phase, each team will implement the CSCI that was allocated to them. At the end of this phase, each team will need to buy three further CSCIs from other teams, to make up the complete system. In the second phase, each team will integrate the CSCIs they bought with the one they developed, and test the whole system. At the end of the second phase, there is a second trading session. Each team will need to buy an integrated system from another team. In the third phase, a change request will be handed out, and each team will modify the system they have bought according to the change request.

Phase 0: Requirements Analysis - 1 week

This phase gives you time to get your team organized, to analyze the specification you have been given, and to ask any questions. The Prof and the Tutors will act as the customers for the software, and will answer questions about the requirements. There will almost certainly be missing requirements, ambiguities and inconsistencies in the initial specification. You will want to identify as many of these as possible before embarking on Phase 1 of the project. You will also want to spend some time thinking about how to plan your time and organize your team. Some thought about project management will be useful here. You will at least want to identify your risks, and think about how to measure your progress.

Phase 1: Module Development - 4 weeks

In this phase, each team will implement the CSCI they have been assigned, write all the associated documentation, and test and debug the code. Towards the end of this phase, teams will begin the trading process. You will be given time during the lab sessions to make 'sales' pitches to the other teams. Some thought will be needed both about how you can 'sell' your software to other teams, and about how you will determine which teams' software to buy. At the end of the phase each team will need to buy the other three CSCIs from teams in the same lab section, so that each team has all four modules.

Phase 2: System Integration and Test - 3 weeks

In this phase, each team will integrate the three CSCIs that they bought with the one that they developed in phase 1, in order to arrive at a fully functional system. Each team will also test their system. Towards the end of phase 2, a second round of trading will begin. Each team must buy a fully working system from another team in the same lab section, to be used for phase 3. Teams will not be allowed to use their own system for phase 3, nor should the system they buy for phase three contain their original CSCI from phase 1.

Phase 3: Software Maintenance - 4 weeks

At the beginning of this phase, a change request will be released. This will describe new functionality requested by the customer. Each team will attempt to modify the system they have bought to include the new functionality.

2. Team Organization

We will organize you into teams (≈4 people) at your first lab session. Hence it is vitally important to turn up to the first lab session. You must remain in the same team throughout the course. All members of a team must be taking the course for credit.

If your team fails to work together satisfactorily, it is up to you to find a way to improve things. Under exceptional circumstances, the prof or Tutors may make changes to the team composition. Typically, this will only happen if students have dropped out of the course.

It takes some effort to ensure that a team functions properly. At the very least, all teams are advised to do the following:

a) Arrange for at least one full team meeting per week, outside the timetabled sessions, and ensure that all members are available at the designated time. Produce a written agenda for each meeting, and designate a chairperson and a secretary. The secretary should take minutes, and type them up for circulation by the next meeting. Rotate these roles during the term.

b) Do not take important decisions at meetings if the whole team is not present. By all means discuss the issues, but save the decision until you are all there.

c) Try not to resort to voting too often. If you find you have to vote on every decision, then its time to rethink how you arrive at consensus. In particular, if the same person or persons lose every vote, then your team is in danger of fragmenting. Bear in mind that to be a productive team, you need to learn to listen to one another.

d) Try not to get personally attached to ideas. Use techniques such as brainstorming, in which the team generates as many ideas a possible without any reflection on whether the ideas coming out are sensible or not. At the end of the exercise, all ideas are owned by the team as a whole: there should be no attempt to remember who said what.

Note that all the assignments during the term are team assignments. Each team submits one report each week. The reports will be graded, and in general, all members of the team will receive the same grade. In exceptional circumstances, the prof may decide to give some members of a team a different grade than others. In each submitted assignment, you will need to give a short account of how well the team worked together, and who did what.

3. Trading Blocks

Each lab section acts as a trading block. You will only be allowed to trade software with other teams in your lab section. Hence, once you have signed up for a lab section at the start of term, you will not be permitted to change to a different lab section.

Each lab section will have approximately 48 students in it, hence 12 teams. As the software system to be built consists of four CSCIs, there will be three versions of each CSCI developed in phase 1 available for you to buy. You will need to select from these competing versions when you trade. A number of factors will affect your selection, the most important of which is software quality. Remember that software quality is a property not just of the program code, but of the documentation, packaging, technical support, and delivery schedule.

As well as developing a strategy for evaluating the software you are buying, you will need to develop a strategy to sell your CSCI to other teams. Remember that in phase 3, you will need to buy a system that does not contain your original CSCI. This rule acts as an anti-monopoly mechanism. At the end of the first phase, each team could potentially sell their software to nine of the other eleven teams in their lab section (the remaining two will have written their own version of the CSCI). If you manage to sell your CSCI to all nine of these teams, there will only be two other teams left for you to buy a complete system from at the start of phase 3. Think carefully about your sales strategy.

The open market rule: there is a stand-by rule to prevent shut-outs at the end of the first phase. This rule will be invoked by the prof if one or more teams become shut out of trading—i.e. if they cannot buy a full set of CSCIs because other teams refuse to sell. If the open market rule is invoked, any team that has not already sold its CSCI to more than half of the potential customer base (ie. five or more other teams) will be required to sell its CSCI to any other team that requests it at a price fixed by the prof. The price will normally be S$1 higher than the highest previous sale of that CSCI.

4. Bank Accounts

The currency used for buying and selling software will be the Software Dollar, denoted S$. The Software Dollar is not a hard currency – all funds will be held by the bank. The bank is operated by the Tutors. At the beginning of the course, a bank account will be set up for each team, with an initial balance of S$300. The aim of the game is to maximise your team's profit by selling as many copies of your module to other teams as possible, and by having a good quality product that will attract a high price. When buying and selling software, the buyer and seller must agree a price. Although the bank publishes a price list, this is entirely advisory: you are free to negotiate any price you like. The negotiated price could include maintenance time, technical support, warrantees on the software, penalties for late delivery, and so on.

Overdrafts are available from the bank on demand. These will accrue interest at the rate of 1% for each working day (Monday to Friday), calculated from midday to midday.

There will be a prize awarded to the team that makes the largest profit by the end of the term. In addition, those teams that end the course with a profit (i.e. at least one dollar more than you started with) will receive a 5% bonus to their grade for the coursework assignments (i.e. your raw percentage grade will be multiplied by 1.05). However, those teams that end the course with a loss will **not** have their grades penalised.

5. Price List

The bank publishes an advisory price list (in Software Dollars) for software as follows:

Grade	**Price**
A	S$200
B	S$125
C	S$50
D+ or below	S$1

The bank also suggests that a reduction of S$10 per weekday is reasonable for late software. This price list is published only as a guide; teams are free to set their own prices when buying and selling. Note that we do not directly grade all aspects of the software you produce — the coursework assignments only cover some aspects of the project. Hence, buyers and sellers must judge for themselves what grade they think the software deserves.

6. Contracts

You *must* sign a contract whenever you buy or sell software. The contents of the contract are up to you, but needless to say, both the buyer and seller must agree on the wording. Signed contracts should be lodged with the bank, and your banker will not transfer money from one account to another until given a copy of the contract signed by both parties.

If any party to a contract believes that the other party has broken the agreement, they may call in the Software Mediation Executive (SME), at a consultancy rate of S$100 per meeting (under exceptional circumstances, teams with trading deficits may apply for legal aid). The SME has the power to set a compensation rate if it deems that a contract has not been met. Note that it may just decide that the original contract was unreasonable. The consultancy fee may be returned to a team if the SME determines that the team is an injured party. Only contracts lodged with the bank and signed by both parties are enforceable. To contact the SME for a consultation, contact <sme@cs.toronto.edu>.

What kind of thing might you want to put in a contract? Here are some suggestions:

- the price;
- a list of the deliverables (e.g. what documentation will be supplied);
- dates for each of the deliverables;
- transfer mechanisms (how will the software be delivered?)
- an agreed quantity of support work;
- costs for support work over and above that included in the original price;
- penalty costs for failure to deliver on time;
- a warranty specifying minimum quality levels.

Equally important are things not included in the deal. E.g. if there is no support effort included in the price, you might be advised to state this. If the software is not guaranteed under certain conditions, the contract should probably describe these conditions.

7. Deliverables

There are six graded assignments due during the term, one every other week from week 2 onwards. These are due to be submitted at the beginning of the lab session two weeks after they are handed out. Graded assignments are team efforts. Each team will submit one report, and under normal circumstances, all members of the team will receive the same grade. See the course orientation handout for deadlines, penalties for late submission, grading scheme, etc. Precise requirements for the contents of each assignment will be given to you during the term.

Along with each of the graded assignments, you should attach a weekly progress report. These will not be graded, but is a chance for your tutor to see what progress you have made, and for you to make your tutor aware of any problems that have arisen. Weekly reports provide you with a framework for the team's decision making.

The weekly report should say **what** you have been doing, not just that nothing is worth reporting. Use this as a chance to critically assess what you have done each week, listing both positive and negative things. If there really is *nothing* to report, then you really do have a problem, and you should discuss what to do about it with your tutor or the prof.

Some things to include in the weekly report:

- **Risks identified.** List any new risks you have identified this week.
- **Technical Problems.** List any technical problems (e.g. design problems) that arose, and if solved, say how.
- **Policy decisions.** List any decisions you've taken this week about buying/selling strategy, testing strategy, management strategy, etc.
- **Work accomplished**. What milestones did you reach this week?

Note that your tutor does not want to read an extensive essay. The weekly report should be brief (e.g. 1 page) but informative. Your team's secretary should be able to produce the report in about 10 minutes or less.

8. Working to Deadlines

An important aspect of this course is to do with working with limited resources. Engineering always involves tradeoffs, because you will never have sufficient resources to develop a perfect product. In particular, time is an important limited resource that you will need to manage carefully. For this course, there is no such thing as a perfect assignment. Assignments will be graded based on what was achieved given the resources available, so it's up to you to think carefully about managing your schedule, and scaling back your goals if your schedule starts to slip.

In particular the following rules will be enforced rigidly:

1) Under no circumstances will a deadline be extended because some other group delivered their module to you late. You are specifically advised to protect yourselves from this by signing a contract with a delivery date in it. If another team had broken a contract, sue them. Furthermore, you should predict such risks and develop risk mitigation plans. Watch for signs that a group you are trading with may be behind schedule, so that you have advanced warning and can implement your backup plan early. The earlier you can detect such problems, the more prepared you will be when they happen. If you end up worse off because a team has delivered something to you late, then your risk management strategy has failed.
2) Under no circumstances will a deadline be extended because "there's only one programmer in the team who is capable of doing the work". The assignments you have been set have very little to do with programming. Try to avoid getting into a situation where only the best programmer knows what's going on in the team.
3) Under no circumstances will a deadline be extended because "we need more time". See above.
4) In general, extensions will not be granted because "one member of our team never does any work", except in rare cases where there are other factors involved. In a team project you all share responsibility, and you need to put some effort in figuring out how to get the team to work together. Note: while the deadlines will not be extended, some adjustment of the grades of the team members may be possible in such situations.

5) Under no circumstances will a deadline be extended because “one of our team is on vacation”. You do not get vacations in term time.

6) You can nearly always get an extension in the case of illness, although you should always let your tutor know of any absences through illness, and requests for extensions should always be accompanied by documentary evidence.

9. Calendar

The dates of the three phases are as follows. Note that teams may agree to deliver all or part of the software on dates other than the last day of the phase. Software delivered early might attract a high price, as it will allow the purchaser to begin integration sooner; software delivered late might attract a bargain basement price. This may matter to some teams more than others: for example if the purchaser is not yet in a position to begin integration, late delivery might not be a disadvantage.

Monday, 10 Sept	Week 2	Phase 0 (preparation) begins
Friday, 14 Sept	Week 2	Phase 0 ends - you should have your team organised by now
Monday 17 Sept	Week 3	Phase 1 begins
Friday, 12 Oct	Week 6	Phase 1 ends - you should have purchased 3 CSCIs by now
Monday, 15 Oct	Week 6	Phase 2 begins.
Friday, 2 Nov	Week 9	Phase 2 ends - you should have purchased a complete system by now
Monday, 5 Nov	Week 10	Phase 3 begins.
Tuesday, 6 Nov	Week 10	The Change Requests will be available from 9:00am
Friday, 30 Nov	Week 13	Phase 3 ends. All outstanding debts must be paid by this date. Bank accounts will be frozen at noon.

10. Advice

Keep it simple, stupid (KISS)

On of the best strategies for successful software development is to keep the project goals reasonable relative to the resources available. This is known as the KISS principle. While it is always tempting to produce a product that exceeds the original specifications, it is rarely beneficial to do so. Experience has shown that extra functionality that was not asked for buy the customer causes the most problems, both for quality of the product and for impact on cost and schedule. If you choose to deliver more than the specification asks for, you are adding unnecessary complexity to your software, which will make it less reliable, and harder to understand and modify. You should avoid adding extra functionality, and no credit will be given for ‘extras’ in grading the assignments.

Use as is, where possible

One of the things you will be asked to do is to give a detailed list of the modifications made to the modules that you bought, and an evaluation of those modules. If, from this assignment, it becomes clear that you have crossed over the line from minor adaptations for integration purposes to completely re-writing the modules, you will be marked down very heavily. This is because the exercise is about adapting other people's code, not writing everything yourself. Try to avoid buying cheap, poor quality code. Also, don’t fall into the trap of thinking that you could do a better job by re-writing everything yourself. The KISS principle applies here too.

Caveat Emptor

Caveat Emptor is Latin for “let the buyer beware”. You have a number of mechanisms to protect yourself from unscrupulous vendors and shoddy workmanship. Use them.