Faculty of Arts and Science
University of Toronto

Midterm Test

Department: Computer Science
Instructor: Steve Easterbrook
Date and Time: 10:10am, Thursday 26th Feb, 2009

Conditions: Closed Book
Duration: 50 minutes

This test counts for 20% of your final grade

Name: _____
(Please underline last name)

Student Number: _____

Question Marks

1 _____/20

2 _____/20

3 _____/20

Total _____/60 = _____%

1. [Short Questions; 20 marks total]

(a) [Software Architectures – 5 marks] What are coupling and cohesion, and why are they important in software design? Suggest measurable properties of a software design that can be used as indicators of the amount of coupling and cohesion.

Coupling – a measure of the degree of interaction between software modules, e.g. the amount of knowledge each module needs to have about the other modules. Can be measured by looking at the number of method calls between modules, the number of parameters passed, etc.

Cohesion – a measure of how well the things grouped together in a module belong together logically. Could be measured by looking at the number of internal method calls within a module, etc.

These are important because they affect maintainability – if coupling is low and cohesion is high, it should be easier to change one module without affecting others.

(b) [Scaling Agile methods – 5 marks] Describe two aspects of agile software development that don't work well on very large software projects, and identify alternative strategies that can be used in their place. At what size of project do you expect such problems to kick in?

[Many possible answers. Must have two distinct aspects, and cover alternative strategies and project size considerations for each.
Suggestions include:]

(1) Emphasis on face-to-face communication over documentation. This starts to breakdown when you have more many team members than can comfortably meet regularly as a team. E.g. meetings of more than around 10 people are unproductive, because it is hard for them all to contribute to the discussions. For larger teams, some of the communication must be in written form, e.g. through use of wikis, email, bulletin board systems, etc, to ensure the larger team communicates effectively.
(2) Self organizing teams – developers decide for themselves which tasks to work on. This starts to break down when there are so many team members that you can no longer remember who's working on what. Again, that's probably around 10-12 people, although the size depends on how well the team members know one another. Alternative strategies include breaking the team up into subteams, and getting each subteam to work on a different module, or using a workbreakdown structure to identify tasks and allocate people to work on them.

(c) [Uses of UML – 5 marks] What would you use a UML Sequence Diagram for? What are its advantages when used for this purpose?

A UML Sequence diagram shows one possible way in which a set of objects can collaborate by message passing to carry out some task. It can be used as a detailed elaboration of a use case during the design process, to decide how the use case will be implemented, and to check that the implementation is feasible.

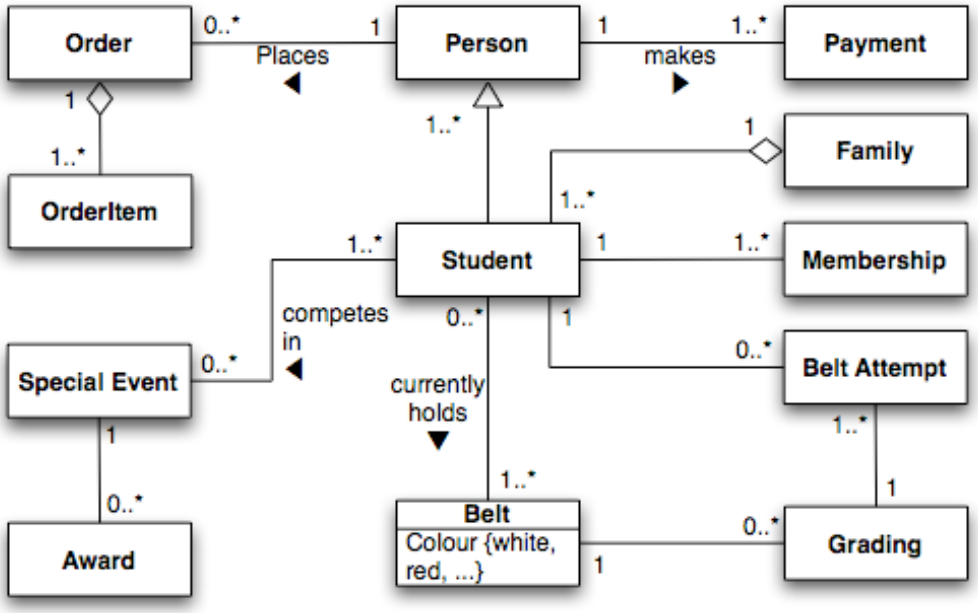
The advantages of this use is that it bridges between the high level description of a use case that would be appropriate for the user to see, and the design of the program in terms of classes and methods. The sequence diagram demonstrates that a specific sequence of method calls between objects can implement the behaviour described in the use case.

[note: other possible answers: use for debugging a complex program behaviour; use to compare two different designs to see how much coupling there is; use for documenting the expected behaviour of a design pattern, etc.

(d) [Software Estimation – 5 marks] What is 3-point estimation? Why would you use it?

Three point estimation involves asking a programmer to provide three different estimates of the amount of time it will take to perform a specific task: the expected time (e), the worst case (w) and the best case (b). These three estimates are then combined by calculating (w+4e+b)/6.
3-point estimation has been shown to provide more accurate estimates than asking for a single number, because it forces the programmer to think more carefully about the upper and lower bounds, and hence to be more realistic.
As this type of estimation works best on small tasks, it could be used when planning an upcoming release on an agile development project, for calculating the estimated time needed to build each function requested by the user.

2. **[Domain Models – 20 marks]** The following domain model captures some basic information about a kids' karate club. In answering the following questions, state any assumptions that you make.



a) How many times can a student attempt to earn a Black belt? [2 marks]

As many times as they like (including none). This is because the student can make 0..* belt attempts, each of which is associated with exactly one grading, and each grading is associated with exactly one belt colour.

b) The model distinguishes between 'student' and 'person'. Why do some associations go to 'Student', and some to 'Person'? Are these modeling decisions sensible? [2 marks]

The distinction allows for people other than the students themselves (e.g. parents) to make payments for their membership. The distinction is useful because it allows us to say there are somethings only students can do (e.g. belt attempts) and some than anyone can do (e.g. pay).

c) What are the implications of the multiplicities on the association between Person and Payment? [2 marks]

Each person can make many payments, and each payment belongs to exactly one person. The 1..* implies that the only people we are interested in are those that have made at least one payment. The '1' implies that it is not possible for several people to split a bill.

d) Does it matter that 'Person' is not associated with 'Family'? [2 marks]

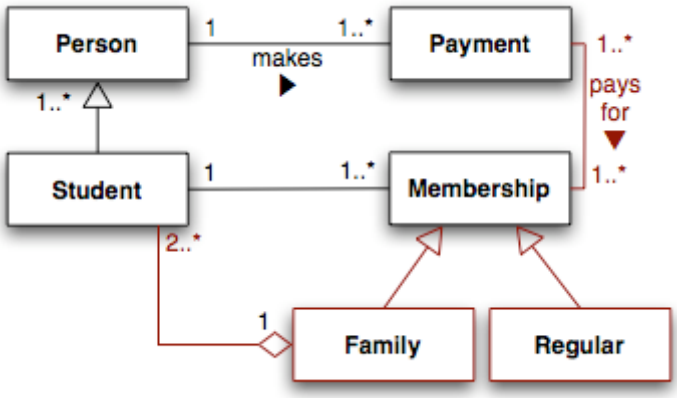
The model only allows us to record family relationships between students (e.g. siblings who are all members). This is good enough if we want to adjust membership fees when kids from the same family are members. But it doesn't record any relationship between the person paying and the student they are paying for, so we might not know which students have been paid for.

e) In the model, each student is shown as belonging to exactly one family. If this rule is enforced in a database, what problems could this lead to? [2 marks]

It's probably okay, except in some rare circumstances. E.g. it is possible for a kid to changes families when the kid is living with a foster family. It might also cause problems if parents divorce and each starts a new family. We would need to consider how to decide whether half-brothers and half-sisters should be considered part of the same family or not.

f) The owner of the club wants to offer a family discount, so that if more than one student from the same family is a member of the club, they each get a 10% discount. How would you alter the model to capture this? [5 marks]

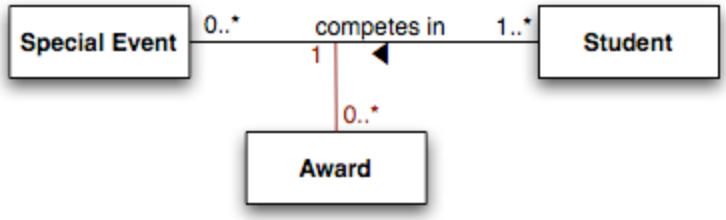
Probably the simplest way is to introduce two subclasses of membership, a family membership and a regular membership, and enforce the constraint that a family must contain at least 2 students. Also, we should probably ensure that payments are connected to the memberships that they pay for:



This solution assumes that the only reason to record family relationships is for discounted memberships, because it makes 'family' just a type of membership, rather than a more general concept.

g) A student can only hold an award for a special event that he or she competes in. How would you modify the model to capture this constraint? [5 marks]

Several solutions are possible, but most of them create dependency loops between classes. The best solution is to make the 'award' an association object, belonging to the association between student and special event. This ensures that award objects can only be instantiated as part of a relationship between a specific student and a specific event:



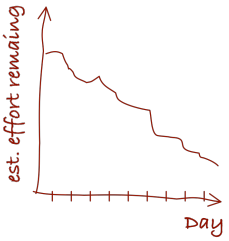
3. **[Project Management – 20 marks]** A good software project manager should develop a realistic plan for the project, and then manage according to the plan, making adjustments if necessary. What measurements of a project does a manager needs in order to do this, and what kinds of adjustments can the manager make if the project is not going according to plan? Identify three tools a manager can use to track the progress of the project, and describe their advantages and disadvantages.

To manage a project, the manager needs to keep track of:
How much effort has been spent on the project so far (and how much is still needed to complete the project)
How well is the project keeping to schedule?
How much software has been built so far (e.g. number of line of code, number of modules complete, etc)
How many defects does it have (e.g. number of bugs reports, number of bugs fixed, etc).

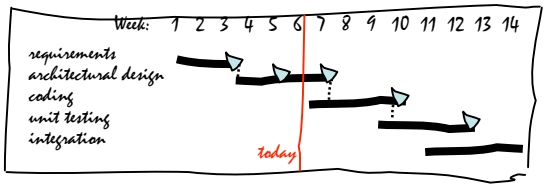
If the project plan needs to be adjusted, there are essentially four types of control:

1. Resources (can adjust the size of the team, get better equipment and tools, etc)
2. Time (can adjust the schedule if it's not going to plan)
3. Product (can adjust how much functionality will be built)
4. Risk (can accept higher risk of not delivering quality software, or risk of not being ready on time)

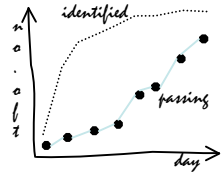
Three tools available for tracking progress:
Burn down chart - plots effort remaining (sum of effort estimated for all unfinished tasks) against time. This shows day-by-day how the project is doing, compared to the plans.



Gantt chart – shows the tasks planned against key dates, with milestones and task dependencies marked. Can mark current date, and check that all milestones have been completed to date.



Test case trend chart. Shows number of test cases identified versus number of test cases passing. If the lines are diverging, will need lots more testing time. When the lines start converging, testing is nearly done.



[scratch paper]

[scratch paper]