Faculty of Arts and Science
University of Toronto

Midterm Test

Department:	Computer Science
Instructor:	Steve Easterbrook
Date and Time:	10:10am, Thursday 1st March, 2012

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 Modeling – 5 marks] It has been suggested that UML models can be used in software engineering as either *sketches*, *blueprints*, or as a *high-level programming language*. Explain each of these different ways of using UML, and describe the limitations of each.

[Note: 1 mark for each defn, 1 mark for a limitation of each use, up to 5 marks total]

Sketches: UML is used primarily to share ideas during meetings, and the diagrams aren't (usually) kept after the meeting is over. Limitations: When sketching, you tend not to worry about precise syntax (thus: possibility of mis-communication), nor do you analyze in detail the more subtle aspects of a model such as the implications of multiplicities on associations (thus: not exploiting the full power of the notations)

Blueprints: UML is used as a specification language to tell the programmers what they need to implement. Limitations: UML does not capture many things the programmers might need to know, such as user's goals, rationale for the design, assumptions about the problem domain, etc. If used in place of face-to-face communication with users and/or domain experts, may lead to code that does the wrong thing

High-level programming language: Tools convert the UML models directly into assembly code to be run on the machine (i.e. no need to use a conventional programming language). Limitations: Hard to capture detailed programming constructs, such as order of operations, conditional logic, etc. Current generation of tools still require some programming code to be added to the translated models, and the translated models are much harder to understand than hand-written code.

(b) [Agile Planning – 5 marks] *Timeboxing* is a technique used in agile planning processes to decide what features to develop for an upcoming software release. But it only works if there are reasonable effort estimates available for how long each feature will take to build. What techniques are available to help *obtain* and *improve* such estimates? How do these techniques differ from traditional (waterfall) style project estimation?

To **obtain** estimates, could use:

- 3-point estimation, where programmers give best case, worst case and typical case estimate for each task (and a suitable formula combines them).
- That card game, where each member of the team selects a card with her estimate on it, and then they all show their cards and discuss differences in estimates (avoids having them bias each other's first estimates).

To **improve** estimates, could use:

- Post-mortem meeting at the end of each development cycle
- "Project velocity" – compare estimates to actual development speed in the last cycle, and use this ratio to adjust estimates for the next cycle.

Key **differences** from waterfall style estimation:

- The programmers themselves discuss and agree on estimates
- Estimate only for small tasks, each of which delivers working code
- Only estimate stuff you're ready to do in the next cycle (a few weeks)
- Fit the tasks to the time available, rather than the other way round

(c) [Software Evolution – 5 marks] Lehmann argued that most useful software is E-type (E for Embedded), meaning the software itself will be embedded in the world, and will change the nature of the problem it solves. Give an example of such software, and an example of software that is ***not*** E-type. How would you assess the success of a project that develops E-type software?

E-type software includes anything that changes the way people work (and hence the way they think about their work). E.g. word processors, spreadsheets, websites, etc are all E-type.
Very little software is not E-type, unless it solves very well defined, simplistic problems. Examples include programming assignments in intro CS courses & simple mathematical problems.
You cannot measure the success of E-type software by looking at whether it passes some tests; rather it requires a subjective judgment of the people who use it, and other people who are affected by its use. Most importantly, E-type software is subject to continuous evolution, so it's never "finished". A project to develop E-type software might best be judged not by any specific release, but by the way the project handles the ongoing evolution of the software. For example:

- Is the project responsive to the ongoing change in users' needs?
- Does the project engage the users in the development process?
- Is the software adaptable/modifiable?
- Does each release make the software more useful?
- Does the project manage the complexity of the design as the software grows?

(d) [Software Architecture – 5 marks] Layered architectures are designed to reduce coupling between components of a software system. Why is this reduced coupling useful? Describe a typical layered architecture, and explain the role of each of the layers.
Reduced coupling is good because it separates the functions that might need to be changed at different times. This is good for:

- Modifiability – changes can be made at one layer without affecting others
- Reusability – layers can be reused in similar systems
- Understandability – easier to understand how the software works

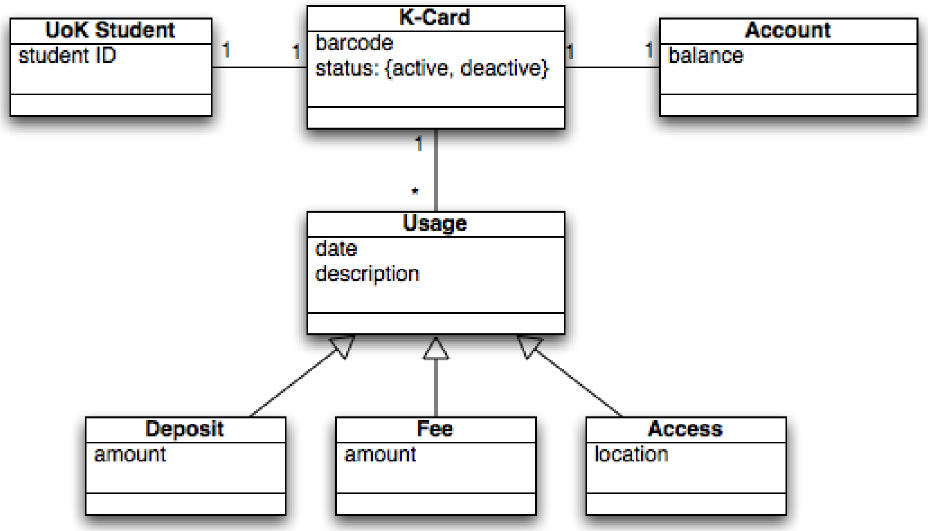
[Must have at least two advantages for full marks]

A typical 3-layered architecture has:

- Presentation layer, responsible for the user interface – e.g. to accept input from the user, display results, and manage the appearance of the interface;
- Business Logic layer for the basic functions provided by the system. Includes all the entity classes, and control classes needed to implement the use cases;
- Data storage layer, responsible for persistent storage of the information users in the system. Usually includes a database component.

[Notes: other possible answers: 2-layers (essentially client-server); 4-layers model splits business logic layer into application layer (responsible for controlling the use cases) and domain entity layer (for basic functions shared by different applications).]

2. **[Class Diagrams – 20 marks]** The University of Kinakuta (UoK) has decided to implement an electronic card system for its students, so that students can use their K-cards to access secure areas (e.g labs and residences), and also as a debit card, linked to an account into which students can deposit money, to be used to pay fees at locations on campus. For the initial release of the system, this will be limited to a few university fees: parking fees at campus parking lots, library fees at campus libraries, and equipment rental at the sports centre. The system will keep a usage record for each K-card. An initial class model for the system has been produced:



(a) [5 marks] In the model provided above, there is a 1-to-1 association between a student and a K-Card. Is this an appropriate way to model students and K-Cards or is there a better way? Defend your answer.

[Note: could argue either way – credit for a well thought out answer]

No, this is not appropriate; it means a student ID can only ever be linked to one card. What if students lose their cards? You would need to deactivate the lost card, so no-one else can use it, and issue a new one. That means you'll need more than one card associated with that student. Even the solution where you don't bother remembering which student a lost card belonged to is not possible in this model, because every card must always be associated with exactly one student.

Also, you won't be able to represent the case when a student does not have a card. So when you create a student record, you'll have to issue a card immediately.

(b) [5 marks] In the model above, there are no explicit associations from the Deposit and Fee classes to the Account class into which these transactions are to be made. Should such associations be added? Defend your answer.

[Again, could argue either way – credit for a well thought out answer]

Associations from the account to the specific transactions are unnecessary. In the implementation, if code in the Account class needs access to info about transactions, it can ask the K-card object with which it is associated to provide the information (and vice versa). As long as there is a path through the model to the relevant information, and no ambiguity about which object a query refers to, then there is no problem. This design reduces coupling between the classes, and therefore makes future modification easier.

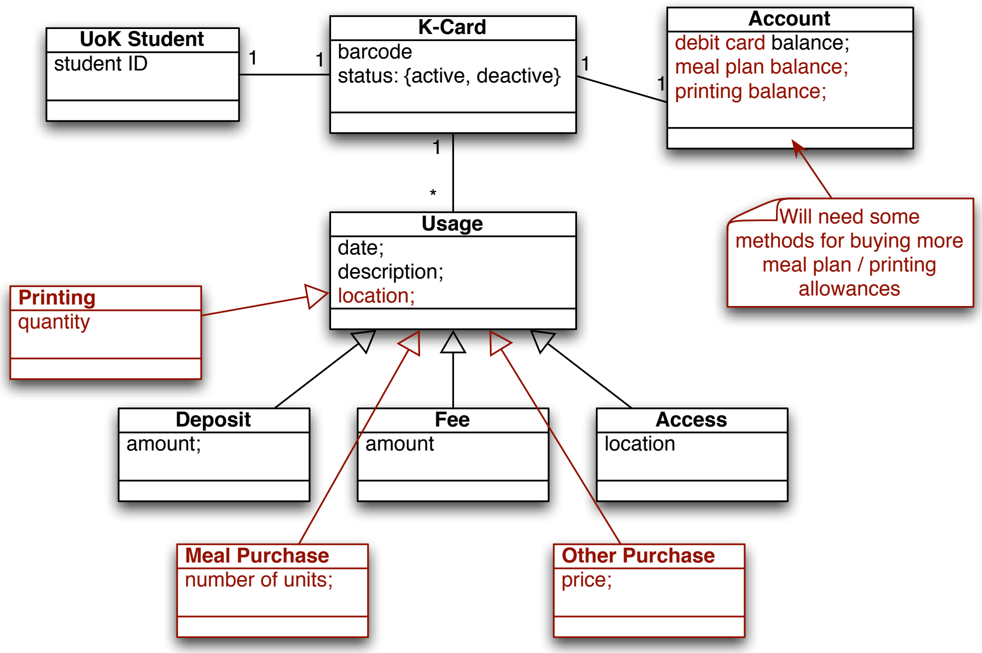
(c) [5 marks] In the original model, the relationship between the Usage class and K-Card class is an association. Alternatively, this relationship could be modeled using an aggregation or composition. Which alternative is better for this problem domain and why?

It doesn't make a lot of difference. The 1-to-many association means that each K-Card will need to keep track of multiple instances of the Usage class that are associate with it. That would probably be via suitable data structure (e.g. an Array) in the K-Card object.

- Making this an aggregation makes no real difference to how it would be implemented, although it does imply that the set of usages "belong to" the K-card, which is a sensible interpretation.
- Making it a composition implies a stronger relationship, i.e. that the individual transactions (deposit, fee, etc) cannot exist independently of the card they belong to. This might be a better design, because it would mean the programmers have to make sure that transactions are always linked to a specific card, and if you delete the record for a given card, you will also delete the record of its transactions.

(d) [5 marks] After further discussions, UoK decides to add more features to the system. Food services wants to keep track of each student's meal-plan allowance when they use their K-cards to buy food and snacks at all campus eateries. Computing services wants to keep track of each student's printing allowance, when they use their K-cards at self-service printers across campus. The meal-plan allowance and printing allowance are separate from the debit-card balance. Student will also be able to use their card to make purchases (as well as paying fees) and to make deposits at many locations across campus, and the system will need to keep track of the location of each transaction. Revise the original domain model to allow for these additional features. Cross out elements that are no longer needed, and add any new elements. Use multiplicities, associations, generalizations, etc as appropriate. You may use the space below to re-draw the entire model if necessary.

Here's one solution; many other variants are possible:



Assumptions: adding location to the Usage class makes sense, although it's not clear whether the "location" in the "Access" class is of a different type (e.g. a room or a lab, rather than a campus shop) – I've left it in "Access" as well, so that it can over-ride the superclass variable.

3. **[Risk Management – 20 marks]** You're being interviewed for a job at NASA to develop flight control software for the next generation of Mars missions (cool, eh?). The NASA project manager conducting the interview shows you a long list that her team has brainstormed, of possible risks – things that could go wrong with the project, either during development or during the actual mission – and she asks you what you would do with such a list. What would you say?
***Hint**: On the wall behind her during the interview is a poster with the following quote on it: "The only way risk management has value is if it affects the way you do business". You sense that she wants to hear more than just a brief answer about measuring risk. And you really want the job…*

[Notes: To get full marks, the answer must cover all of these aspects of risk management well (although exact terminology isn't necessary). Bonus marks for suggesting ways in which all members of the project team can participate, and how the risk management would change their work practices (as per the hint).
Award marks for good insights/suggestions in each area.
Give plenty of credit for good examples, especially ones that pull in ideas from other parts of the course.]

1) Risk prioritization. First we need to decide the relative importance of each of the risks on the list. Several possible approaches:
- Could start with a basic triage process, to select the major risks that will need careful management, and the non-risks (e.g impossible or trivial issues) that can be ignored, leaving the middle group of risk to be monitored. The weakness of this approach is that we haven't systematically measured each risk.
- Could attempt to measure each risk explicitly, by calculating the probability (p) of occurrence and the cost (c) of the loss for each risk, and calculate the risk exposure as RE = p x c. Can then compute this for each risk, and use this to put the list of risks in rank order. However, it may be hard to quantify many of the risks.
- Could use a hybrid approach, using a qualitative matrix, where probability and loss are evaluated on a small number of levels (e.g 3-5 levels for each). Cells in the matrix that have high probability and/or high loss are treated as critical risks to be managed. Each cell in the matrix identifies a type of risk, and we'll handle the different types separately:

		Likelihood of Occurrence		
		Very likely	Possible	Unlikely
Undesirable outcome	(5) Loss of Life	Catastrophic	Catastrophic	Severe
	(4) Loss of Spacecraft	Catastrophic	Severe	Severe
	(3) Loss of Mission	Severe	Severe	High
	(2) Degraded Mission	High	Moderate	Low
	(1) Inconvenience	Moderate	Low	Low

2) Risk Mitigation. Just identifying and ranking risks is not enough. Need to identify a strategy to reduce the likelihood and/or impact of each risk. The position of each risk in the risk matrix will guide this – we'll spend more time on the risks in the upper left, and

less on the lower right. Also, will look for strategies that move risks in the left hand column(s) towards the right, and risks in the upper rows downwards.

For each risk, we can ask (a) how to reduce chances of it occurring, (b) how to reduce the impact if it does occur, and (c) what the warning signs are – i.e. what project variables would we monitor to check whether the likelihood is increasing.

For example, if the risk for a Mars spacecraft is "missing the launch window", then we could reduce its probability by allowing for extra slack in the schedule, using critical path planning, etc. We could reduce its impact by having an alternative launch schedule (if that's even possible!). We could monitor this risk using tools such as burndown charts, which show task completion against schedule. Then we'd define specific trigger points when we have to consider whether other contingency plans need to be put into action to help meet the schedule.

3) Continuous risk management. It's not sufficient to just go through the exercise of identifying and planning for risk once – it has to be a continuous activity throughout the project.

One way of doing this is to maintain a "top ten" risk list, and review this every week with the entire project team. The aim is to focus everyone on what steps they are taking to reduce each of the risks on the list. Periodically, the entire list of risks needs to be re-visited, to check whether any of the risk exposures have changed, and whether new risks have been identified. The whole project team needs to be involved in this exercise, and they need to be encouraged to identify and report new risks (so some kind of reward system might be needed, to overcome the human tendency to assume the worst can't happen, and also the human tendency to downplay risk when talking to managers, to make the project sound like it's going better than it really is).

One idea for project managers is "risk as a resource" – it's one of the four key things a manager can actively adjust about the project (the others being functionality - how much software we're building; resources – people, computing, etc; and schedule). Any adjustments on the other three variables need to be assessed for their impact on risk, and risk can often be reduced by adjusting the other variables.

For example, to continue the previous example, if the burndown chart shows we're getting behind (and hence risk of missing the launch deadline is growing), then we can do things like a requirements scrub (reduce functionality), or add resources (e.g. bring in more engineers to help).

[scratch paper]

[scratch paper]