Faculty of Arts and Science
University of Toronto

Midterm Test

Department: Computer Science
Instructor: Steve Easterbrook
Date and Time: 10:10am, Thursday Nov 9, 2006

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

4 _____/20

Total_____/80 = _____%

1. [Short Questions; 20 marks total]

(a) [Problem Statements – 5 marks] For complex software systems, it is useful to write a description of the *problem to be solved* separately from any description of the *proposed solution*. Give three reasons why a separate problem description is useful. Why is it hard to write a problem description without also thinking about the proposed solution?

A separate problem statement is useful because:
- it allows you to check with the stakeholders that you have understood their problem correctly
- it allows you to evaluate different design choices with respect to how well they solve the problem
- it provides a good source of test cases.

It is hard to separate the problem statement from the solution because the problem will evolve as you attempt to solve it, and because people might not know what problem they have until they consider some possible solutions

(b) [Risk Management – 5 marks] In order to assess which risks are the most important, it is common to calculate *risk exposure*, which gives a simple numeric value for each risk, allowing them to be compared. How would you calculate risk exposure for common software development project risks?

Risk exposure is calculated as the size of the potential loss times the probability of occurrence. To compare risks, the estimates of size need to be comparable, so it is common to use either a dollar value, or a simple 5-point scale.

For a software development project risk, e.g. risk of key personnel leaving the project:
- estimate the size of the loss. E.g. add up the cost of delay plus the cost of finding a replacement. Express these as dollar values.
- Estimate the probability of occurrence as a number between 0 and 1. Staff turnover data from previous similar projects could be used to get a good estimate.
- Multiply size of loss and probability to get a number for risk exposure.

(c) [Economic Feasibility – 5 marks] *Net Present Value* measures the total value of an investment over its lifetime. Why is it useful to know the Net Present Value of a proposed software development project? How would you calculate the Net Present Value of a proposed project?

It is useful to know the net present value of a software project, as it helps you to determine whether the project offers a good investment (compared to other alternatives, or other ways of using the resources). Net present value includes all the expected benefits from future years, all expressed in today's dollars. This makes it possible to compare options that have a different payback period.

To calculate Net Present Value for a proposed project, you need to estimate all the benefits that it will deliver and all the costs that will occur in each year of its expected operational life. Express these all in dollar values, and adjust each year's figures into today's dollars using the (industry-specific) discount rate.
The Net Present Value is then calculated as the total cumulative present value of all the benefits, minus the cumulative present value of all the costs.

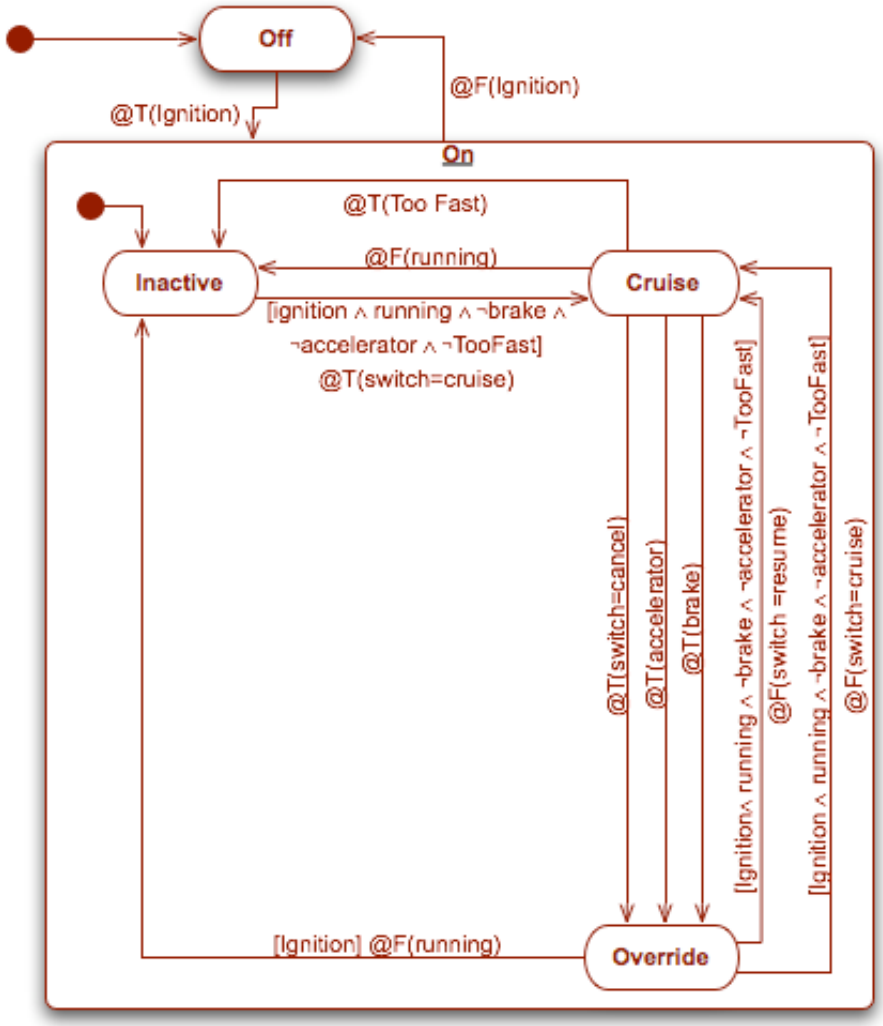
(d) [UML Activity Diagrams – 5 marks] Give two examples of information about a problem domain that can be captured in UML Activity Diagrams, and two ways in which these diagrams can be useful for Requirements Analysis.

Examples of info represented:
(1) Activity diagrams show how tasks depend on one another in a business process. For example, they can show that some tasks can be carried out in parallel, while other tasks have to be synchronized so that one doesn't start until the tasks on which it depends are complete.
(2) Activity diagrams show the flow of activity between different business units, using swimlanes. In other words, they show how different business units must work together to carry out some process.

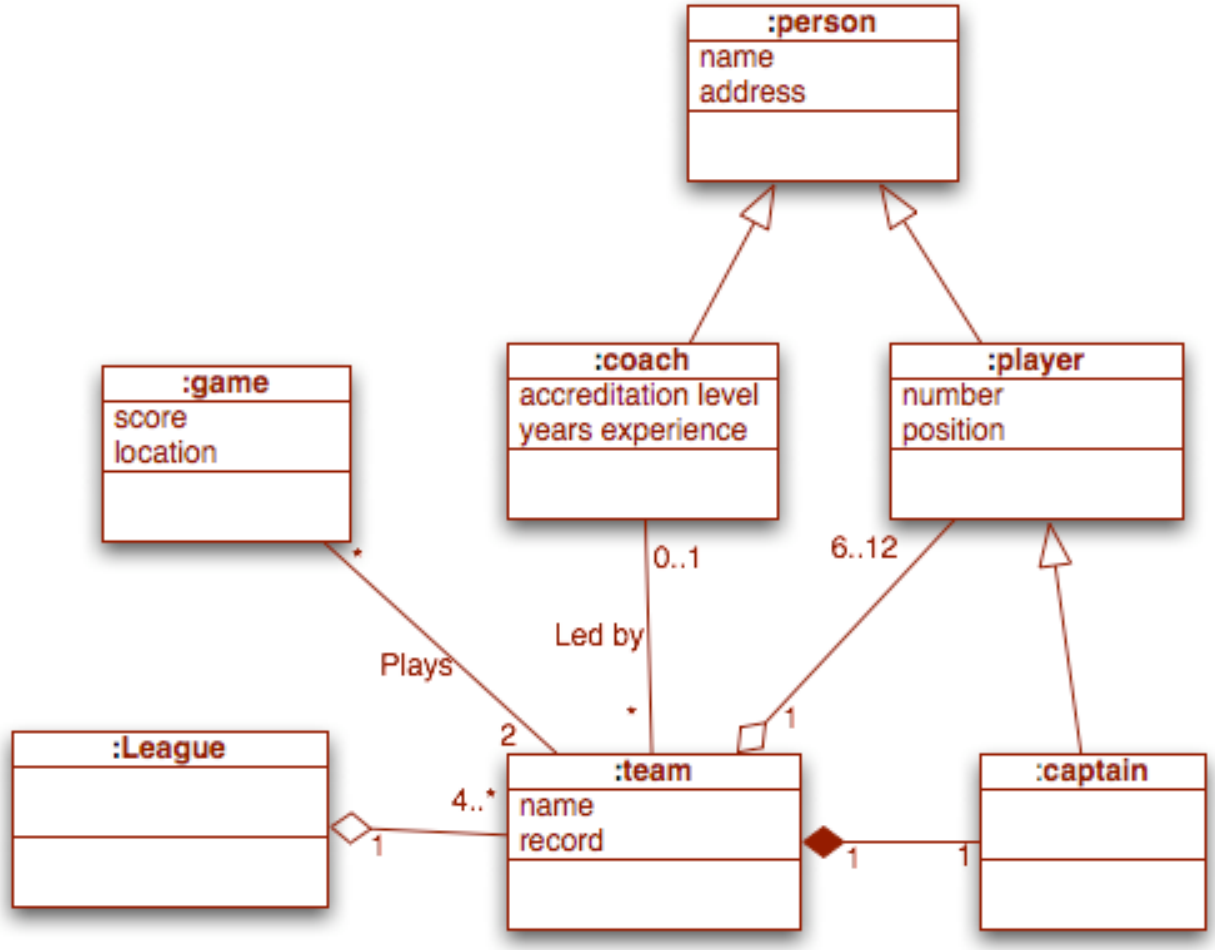
Activity diagrams are useful in requirements analysis (1) to understand the existing business processes, showing how an organization currently solves the problem, and (2) for sketching proposed solutions to the stakeholders to show how a proposed solution would change the way in which the organization works.

2. **[State Modelling – 20 marks]** The following SCR Mode Table describes the required behaviour of a car's cruise control system. Draw a UML StateChart Diagram to show the same information. Use superstates (where appropriate) to simplify your diagram, and be sure to label all transitions with the relevant events and conditions.

Old Mode	Ignition	Cruise Switch	Running	Brake	Accelerator	Too Fast?	New Mode
Off	@T	-	-	-	-	-	Inactive
Inactive	@F	-	-	-	-	-	Off
	T	@T(cruise)	T	F	F	F	Cruise
Cruise	@F	-	-	-	-	-	Off
	-	-	-	-	-	@T	Inactive
	-	-	@F	-	-	-	Inactive
	-	-	-	@T	-	-	Override
	-	-	-	-	@T	-	Override
	-	@T(cancel)	-	-	-	-	Override
Override	@F	-	-	-	-	-	Off
	T	-	@F	-	-	-	Inactive
	T	@T(resume)	T	F	F	F	Cruise
	T	@T(cruise)	T	F	F	F	Cruise



3. **[Class Diagrams – 20 marks]** Draw a UML Class Diagram representing the following elements from the problem domain for a hockey league. A hockey league is made up of at least four hockey teams. Each hockey team is composed of six to twelve players, and one player captains the team. A team has a name and a record. Players have a number and a position. Hockey teams play games against each other. Each game has a score and a location. Teams are sometimes lead by a coach. A coach has a level of accreditation and a number of years of experience, and can coach multiple teams. Coaches and players are people, and people have names and addresses. Draw a class diagram for this information, and be sure to label all associations with appropriate multiplicities.



Notes: captain could alternatively be represented as a second, named association between player and team.

Assumptions: each player only plays on one team, each captain only captains one team, each team only plays in one league.

4. **[Application Domains – 20 marks]** Michael Jackson has proposed a conception of requirements engineering that distinguishes machine domain phenomena from application domain phenomena, as illustrated in the following diagram:

(a) *[5 marks]* Explain the distinction Jackson makes between Requirements, R, and Specifications, S. What additional properties should a Specification have in order to distinguish it from Requirements?

Requirements are any properties of phenomena in the application domain that a stakeholder would like to be made true by some new system. Requirements may refer to any phenomena, whether accessible to the machine or not.
Specifications are a subset of requirements, covering only phenomena that are shared between the application domain and the machine.

To distinguish it from requirements, a specification must be expressed only in terms of inputs and outputs to the machine, i.e. phenomena that are shared between the machine and the application domain

(b) *[5 marks]* Give examples of R, D, S, C and P for a particular problem domain.

Many possible answers – marks given for demonstration of the key distinctions. The example given in class was:
For an aircraft
R – "Reverse thrust should be disabled unless the aircraft is moving on the runway"
D – "pulses from the wheel sensors will be received only when the aircraft is moving on the runway"
S – "reverse thrust should be disabled unless wheels sensor pulses are on"
P – the flight control software that takes commands from the pilot's controls and makes the aircraft do things, written according to specification S.
C – the flight computer (hardware) on the aircraft

(c) *[5 marks]* Using Jackson's conceptual model, explain why checking that a program meets its specification is not a sufficient test of fitness-for-purpose.

Even if a program meets its specification, the specification could still be wrong:

- The specification could fail to express the requirements;
- The requirements themselves may not adequately capture what the stakeholders really want/need;
- Assumptions made about the domain might be wrong, so that the reasoning that associates input and output events to domain phenomena is wrong.

In any of these cases, the program will not suit its purpose well, assuming that the purpose is to solve the stakeholders' problem(s).

(d) *[5 marks]* Suggest two techniques that you would use for checking whether statements of the Requirements, R, and Domain Properties, D, are valid.

Any of the following are acceptable:

(1) Modelling – build a model (e.g. statechart, class diagram, etc) that captures your understanding of R and D, and then analyse it to check that it has the right properties, e.g. check it against scenarios or sequences of events that you know should or should not occur.

(2) Stakeholder interviews or questionnaires – ask stakeholders directly whether they agree with statements of R and/or D.

(3) Prototyping – build a quick and dirty version of the system based on your understanding of the requirements, and show it to the customer, to find out whether it really does solve the right problem, and whether the assumptions you made about the domain are correct.

(4) Inspection – document all your statements about R and D, and then hold a formal inspection meeting, with key domain experts and stakeholders participating.

[scratch paper]