University of Toronto

Faculty of Arts and Science
Dept of Computer Science

CSC340F – Information Systems Analysis and Design

December 2004
Instructor: Steve Easterbrook

No Aids Allowed
Duration: 2 hours
Answer all questions.

Make sure your examination booklet has 10 pages (including this one).
Write your answers in the space provided.

This examination counts for 35% of your final grade

Name: _____
(Please underline last name)

Student Number: _____

Question Marks

1 _____/20

2 _____/20

3 _____/20

4 _____/20

5 _____/20

Total_____/100

1. [Short Questions; 20 marks total]

(a) [Goal Modelling – 5 marks] A telephone company manager has told you that her primary goal in developing a new telephone billing system is to reduce the number of errors made on customers' monthly bills. Give two different types of question that could be used to develop a goal tree from this starting point, and suggest an example goal that each question might elicit.

How questions, to elicit lower level goals,
e.g. "How would you reduce the number of errors?"
might elicit subgoals such as "improve data entry procedures"; "automate rate plan calculations"; "simplify discount structure"; etc

Why questions, to elicit higher level goals,
e.g "Why do you need to reduce the number of errors?"
might elicit higher level goals such as "improve image with customers"; "reduce number of customers moving to competitor phone companies"; etc.

Notes: Need specific example for each type of question and goal elicited to get full marks.

(b) [Economic Feasibility – 5 marks] Explain the difference between return on investment and payback period. Why might you need to know both in order to decide which of two different development alternatives represents a better investment for a particular organisation?

Return on investment (ROI) measures the ratio of benefits to investment made. It allows you to compare development alternatives in terms of overall value for money invested. However, it does not say anything about when (how soon) the return on investment will be realized.

Payback period measures the length of time until the benefits of an investment start to outweigh the costs, but does not measure the size of the return.

You might need to know both because both can affect the desirability of proposed project. In particular, an alternative with a higher return on investment might not be the best to pick if the payback period is very long. E.g. We might prefer an alternative with a lower ROI if the payback period is one year, to one with a higher ROI but a payback period of ten years.

(c) [Non-functional requirements – 5 marks] Give two examples of non-functional requirements, and for each suggest a measure that could be used to test whether a system satisfies the requirement.

Two examples:

Usability – e.g. "The system shall be usable by novice users without any prior training"
Possible measure to test satisfaction: Get sample users to carry out a representative task using the new system, without providing any training, and measure the time it takes them to figure it out, and the number of mistakes they make.

Reliability – e.g. "The system shall be 99.99% reliable"
Possible measure to test satisfaction: Measure mean time to failure for the first week or month of operation, and check that the system downtime is no more than 1 minute for every 10,000 minutes of operation.

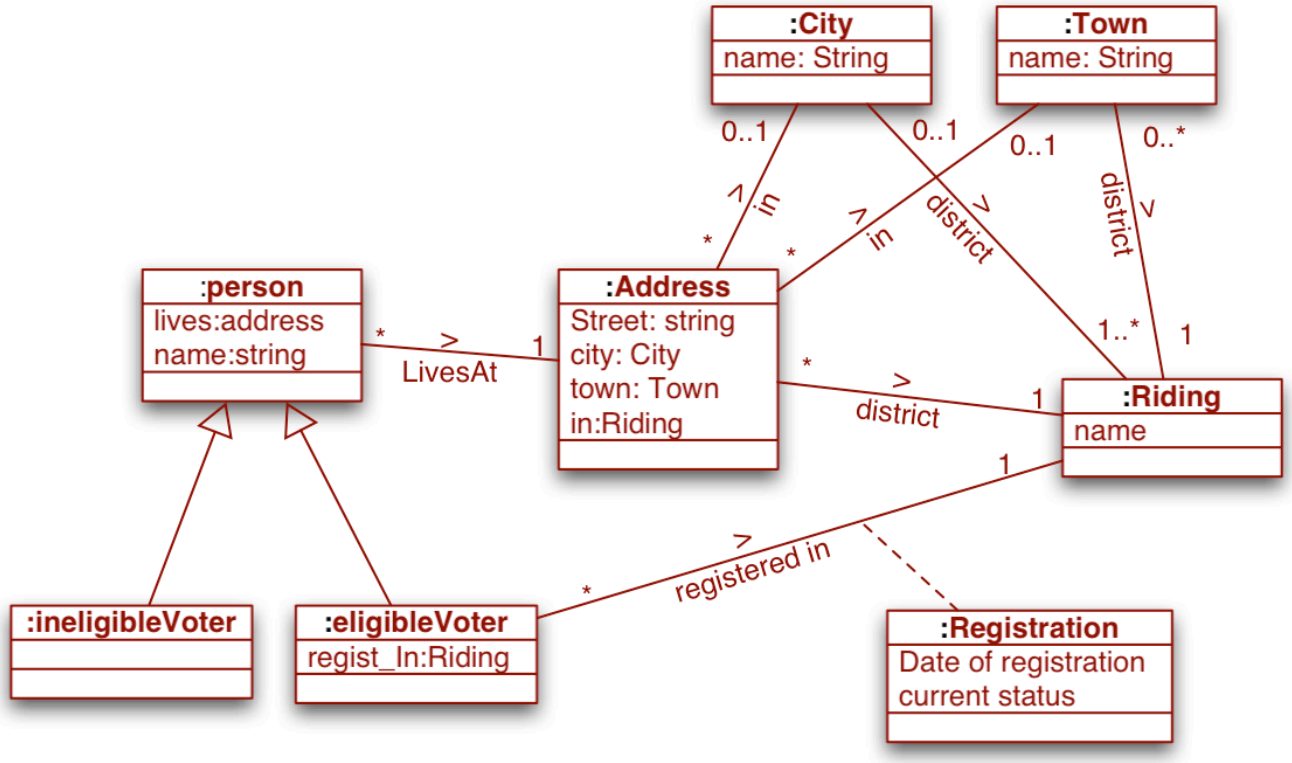
Note: keywords such as "usability" and reliability" are not requirements on their own, they are categories of requirement. Only give partial credit if there specific example requirements are missing.

(d) [Software Architectures – 5 marks] Layered software architectures can be used to reduce coupling between the functions of a system that are closer to the machine and those that are closer to user needs. 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)

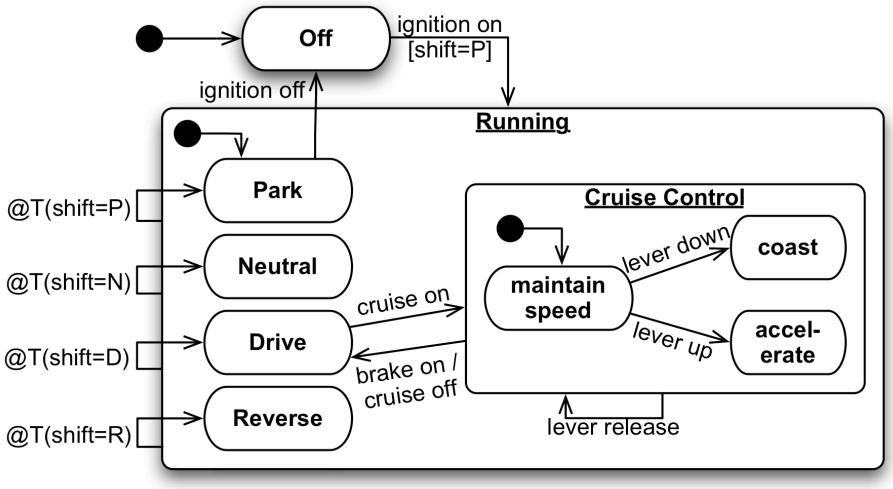
Typical 3-layered architecture has:
Presentation layer, responsible for the user interface. Includes classes to accept input from the user, display results, and manage the appearance of the interface;
Business Logic layer implements the basic functions provided by the system. Includes all the entity classes, and control classes needed to implement the use cases;
Data storage layer, responsible for persistent storage of the information users in the system. Includes a database component, or other mechanism to retain and query the data.
[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] Draw a class diagram to represent the following description of domain objects involved in developing a registration system for voters in provincial elections.
"Each person can be either an eligible voter or an ineligible voter. Eligible voters are registered in only one electoral district (known as a 'riding'), and for each registration instance, details such as the date of registration and current status need to be stored. Eligibility is determined by address. Each person can be assumed to have one current address, and each address is in either a town or city. Some ridings contain several towns, while some large cities may contain more than one riding."
Be sure to include names and multiplicities on all associations, and state any assumptions you make. How would you modify your diagram to handle the following situations: (a) we may need to retain information on people who no longer live in this province; (b) some people own property in more than one riding, and hence apply for registration in more than one riding; (c) people with more than one property are required to designate one as their 'primary residence' to determine which riding they can vote in.



Modifications:
(a) Either make the association from address to riding optional (0..1), or add an extra class for "out of province address", perhaps as subclass of address.
(b) Allow more than one address to be associated with a person, and add address to the Registration class, to indicate which address the registration is for.
(c) Add an additional class for "primary residence", subclassed from address, and require each person to have exactly one primary residence.

3. [Event Modelling – 20 marks] The following statechart diagram shows the main states of an automobile drive system for a car with a cruise control feature. Assume that the notation @T(x) means the event of x becoming true when it was previously false.



a) Write down a *sequence of events* that will put the car into reverse (from the initial state).

Shift=P; ignition on; shift=R

b) Write down a *sequence of events* that will get the car to accelerate using cruise control (from the initial state).

Shift=P; ignition on; shift=D; cruise on; lever up

c) What is the shortest *sequence of events* needed to turn the engine off from the 'coast' state?

Shift=P; ignition off

e) For each of the following properties, state whether the property is true or false, and explain your reasoning:
"The car can only be started when the gear shift is set to N"

False – the only transition from OFF is guarded by [shift=P]

"Cruise control is always disabled when the car is in reverse"

True – the only way to get to cruise from reverse is to pass through the Drive state

"Ignition can never be turned off while the cruise control is active"

True – the ignition off event only has an effect in the Park state; entering Park cancels cruise control, therefore trying to turn off ignition in cruise control has no effect.

"The gear shift is always set to D while cruise control is active"

True – shift=D is true when entering the Drive state, and must remain true throughout use of cruise control, because changing the shift to anything else cancels cruise control

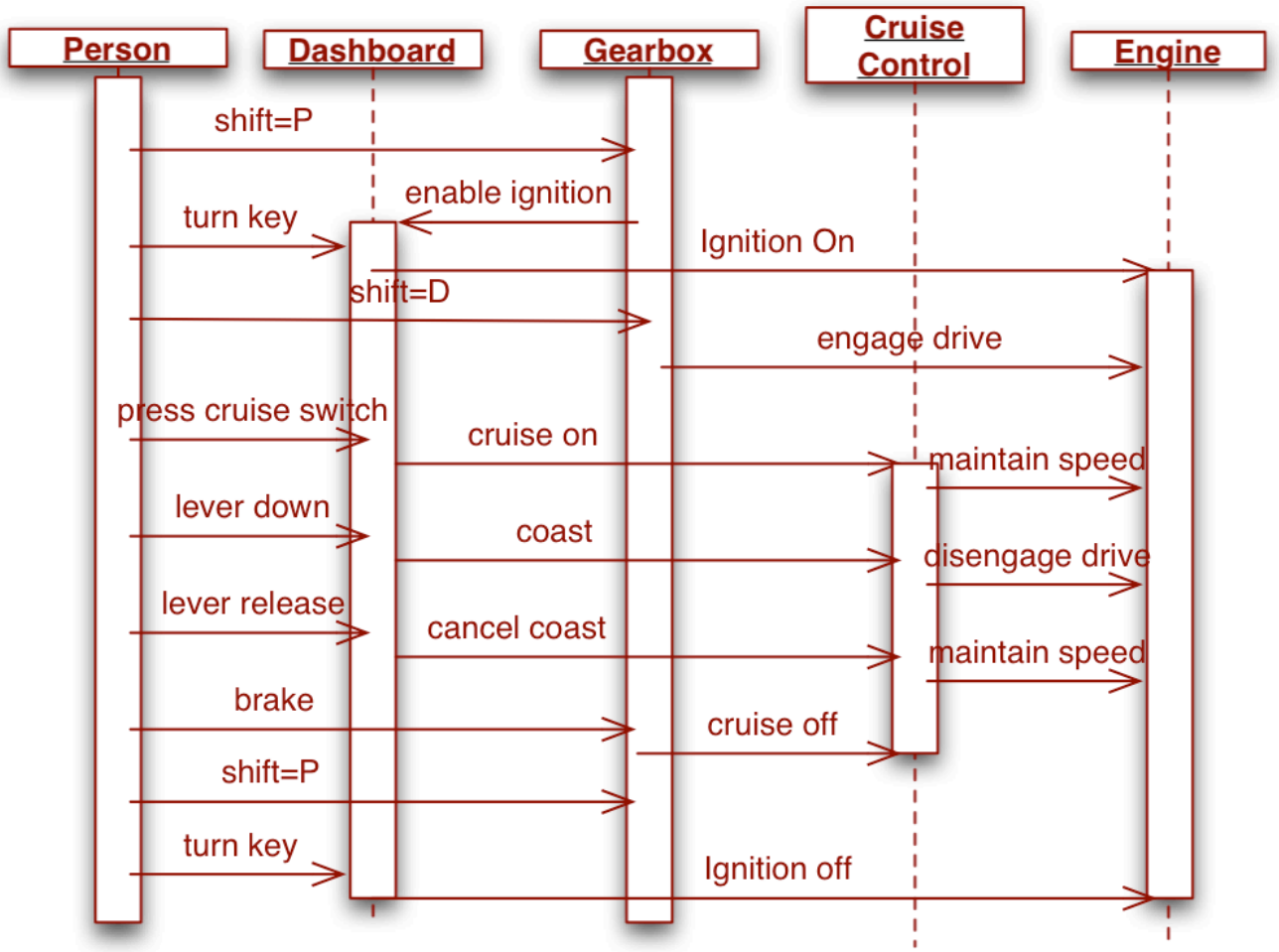
"The only way to cancel the cruise control is to press the brake"

False – can change the shift to something other than D to cancel cruise control

f) Describe any modifications needed for this diagram to add the safety feature that the gear shift cannot be moved from P (park) to any other selection unless the driver is pressing the brake pedal.

Add guards [brake=on] to the transitions into Neutral, Drive and Reverse. However, this is overly general as it also requires the brake to be on to transition between these three states. Really need to make these guards more specific: [not(shift=P) or brake=On]

4. [Sequence Diagrams – 20 marks] Draw a sequence diagram to illustrate the sequence of interactions between the main components of the automobile described in the previous question. Your diagram should cover the sequence that starts from the **off** state, passes through the **coast** state, and back to **off**. Your diagram should clearly show the interactions between *the driver* (who initiates the events), the *dashboard control panel* (where the driver pushes buttons, levers, and turns keys, etc), a *gearbox* (which monitors the position of the gear shift, and enables/disables the ignition switch on the dashboard), the *cruise control subsystem* (which receives messages from the dashboard), and the *engine* itself (which starts and stops when the ignition is turned on/off, and controls the speed of the automobile as directed by the cruise control subsystem).



Note: This answer assumes that the gearshift is not part of the dashboard, ie. that gear shift changes are handled directly by the gearbox.
It also assumes that the dashboard is "turned on" (active) by the enable ignition message from the gearbox, and "turned off" when ignition is turned off.
It also assumes that the engine can maintain its own speed. In reality, the speed would be constantly monitored by the cruise control unit, which would then just send 'accelerate' and 'decelerate' messages to the engine. This is hard to show on a sequence diagram.

5. [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) 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.

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

(3) 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.

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

(Scratch paper)

(Scratch paper)