University of Toronto

Faculty of Arts and Science
Dept of Computer Science

CSC340F – Information Systems Analysis and Design

December 2005
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) [Non-Functional Requirements – 5 marks] What distinguishes a non-functional requirement from a functional requirement? Give an example of each.

Non-functional requirements refer to general properties or qualities of the system as a whole. It is usually hard to identified a particular place in the program code where they will be implemented. Functional requirements refer to individual functions (e.g. expected responses to each input) that the system must carry out.

E.g. Functional requirement: "The system shall print a login prompt when the program starts up"

Non-functional requirement: "The system shall ensure that all data is stored securely and that unauthorized access is prevented".

(note: need something more specific than "the system shall be secure" etc for full marks.

(b) [Requirements elicitation – 5 marks] What are the advantages and disadvantages of interviews as a primary technique for eliciting stakeholder requirements? How might you overcome the disadvantages?

Advantages include:

- Relatively easy to set up
- Good for collecting rich detailed information
- Adaptive – can adjust questions to follow up on interesting points

Disadvantages include:

- Say/do problem – person being interviewed away from where they normally work may give answers that don't reflect what they actually do
- Time consuming to interview many stakeholders
- Hard to compare responses from many different stakeholders
- Interviewer bias – body language and question phrasing may lead interviewee to give more or less detail in response to perceptions about interviewer's interest in each question

Overcoming the problems:

- Supplement interviews with other techniques such as observational study
- Train the interviewers carefully
- Use a standard set of questions as a guide

(Note: 1 mark for each good point; must have at least one good point in each part for full marks)

(c) [Software Evolution – 5 marks] Lehman's studies showed that any software that supports human activity must undergo continuous change, or become steadily less useful over time. Why is this? What regularities would you expect to see in this process of change over the lifetime of a software system?

Software must undergo continuous change because the human activities that it supports change over time – new business models, new processes, changes in the technology etc. Plus, the system itself will change the users' perceptions of their work, and lead them to demand further improvements and new features over time.

Regularities include:

Amount of change in each release of the software will be roughly constant over the life of the system

Work output of the maintenance team is roughly constant over time, irrespective of the resources available.

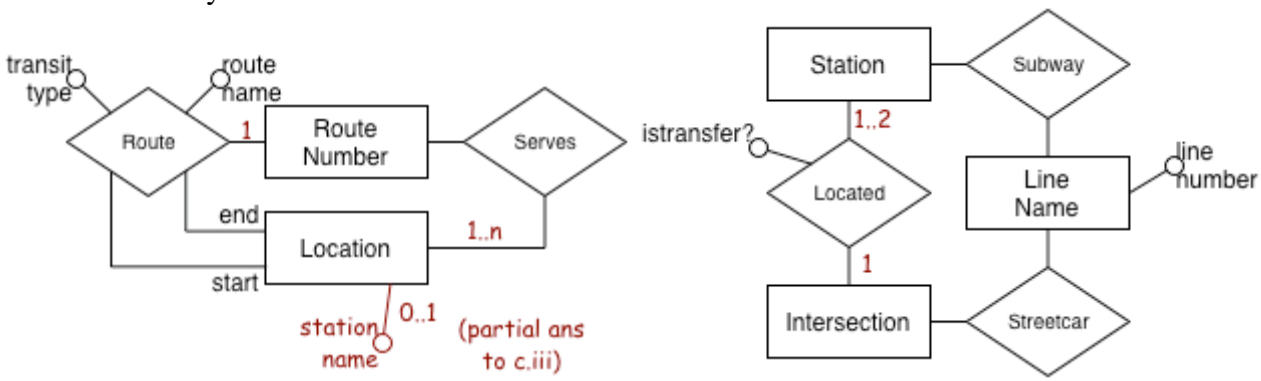
(d) [Software Architectures – 5 marks] What are coupling and cohesion, and why are they important in software design? Suggest measurable properties of a software design that can be used as indicators of the amount of coupling and cohesion.

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.

2. [Entity-Relationship Diagrams – 20 marks]] The following two alternative Entity-Relationship models have been proposed as the basis for a database to hold information about the Toronto transit system:



a. [8 marks] An instance of a relationship is a tuple of entity instances (and attribute values where applicable). Write down one example instance for each of the four relationships shown above. Make up some names if you're not familiar with the Toronto transit system!

Route: 501, Long Branch (start), Neville Park (end), Streetcar(type) Queen(routename),
Serves: 501, Queen&Yonge (location)
Subway: Bloor/Danforth (line name), Yonge (station)
Located: Yonge (station), Bloor&Yonge (intersection), yes (istransfer?)
Streetcar: 501, Queen&Yonge (intersection)

b. [4 marks] Give an example of a situation that can be represented in the first model, but not the second, and an example of a situation that can be represented in the second model but not the first. In each case, could you accommodate the missing situation just by adding more attributes?

The second model doesn't have any way to represent where the routes start and end. These cannot be attributes of station or intersection as each may be midpoints on other lines; could add a "is_endpoint" attribute to subway/streetcar relationships.
First model doesn't distinguish between intersections and subway stations, so cannot represent places where you can transfer between streetcars and subway. This could be handled by adding an optional "station name" attribute to location. (NOTE: other answers are possible)

c. [8 marks] Add multiplicities to both diagrams to show the following constraints. If a constraint cannot be expressed using multiplicities on one or both models, say so below.

i. Each transit line may have 2 or more endpoints.

Cannot be represented on either model

ii. Each location (i.e. a station or intersection) must be served by at least one transit line.

1..n between 'location' and 'serves' on first model; cannot be represented on second

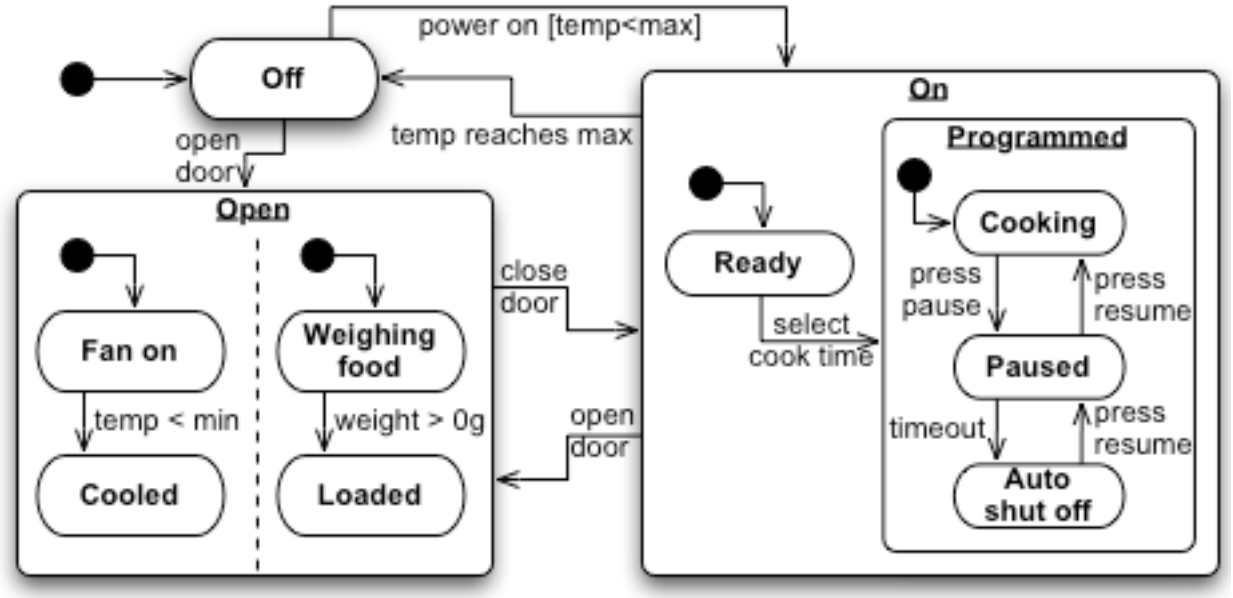
iii. A station can have entrances at either 1 or 2 intersections, but each intersection has at most one station.

Cannot be represented on first model; 1..2 above, and 1 below 'located' on second model.
Note: also accept as answer that the second part of this property can be captured on the first model with at 0..1 attribute "Station name" on location entity.

iv. No two routes can have the same route number.

1 between Route and Route Number on first model; cannot be represented on second.

3. [Event Modelling – 20 marks] The following statechart diagram is a sketch of some of the behaviours of a microwave oven. Assume that max > min.



a) Write down a *sequence of events* that will get to cooking (from the initial state).

Power on (assuming temp<max); select cook time.

(note: also accept open door; close door; select cook time)

b) What is the shortest *sequence of events* needed to get back to cooking again if the oven overheats?

Open door; close door; select cook time.

(note: "power on; select cook time" is acceptable if we assume 'overheats' does not mean 'temp>max')

c) What is the shortest *sequence of events* needed to change the cooking time while cooking?

Open door; close door; select cook time.

d) For each of these properties, state whether the property is true or false, and explain your reasoning:

"the oven remembers the selected cooking time when the door is opened"

false – the oven forces you to select a cook time again after you close the door

(note: "unknown" is also acceptable on the basis that the model doesn't tell us how cook times are stored)

"the oven remains in the 'auto shut off' state if it overheats"

false – the "temp reaches max" event causes it to transition to 'off'.

"sometimes you need to press resume more than once to get it to cook"

true – in the "auto shut off" state, two "press resume" events are needed to get to "cooking".

"there is no way to turn the oven to off other than getting it to overheat"

true – this is the only transition to the off state.

(note: 'false' is also acceptable on the basis that "auto shut off" can also be considered an off state)

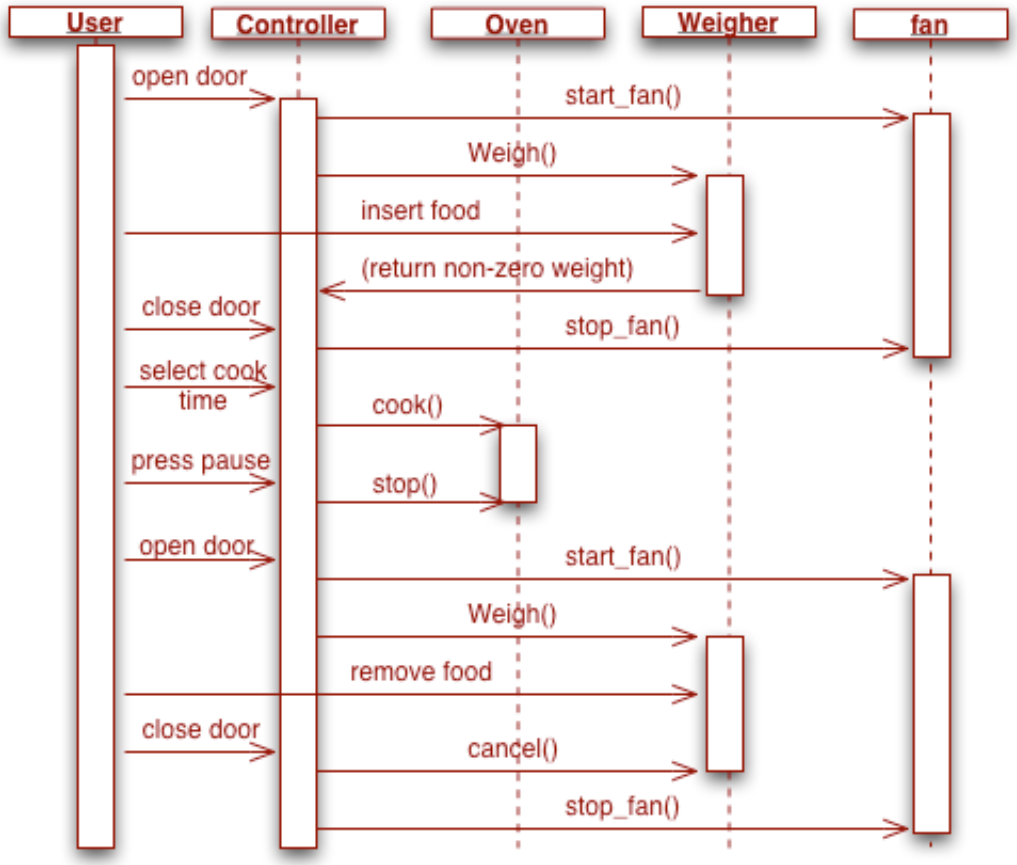
e) What does the model say about what would happen if the oven overheats while the door is open?

The model doesn't account for this. If the oven overheats in the open state, it is ignored. When the door is closed, it transitions to the ready state (presumably displaying cook time options), and then transitions immediately to off. (note: some statechart variants actually ignore this last transition!)

f) The weight sensor is intended to prevent the oven from cooking when no food is loaded. How would you modify the model to include this constraint?

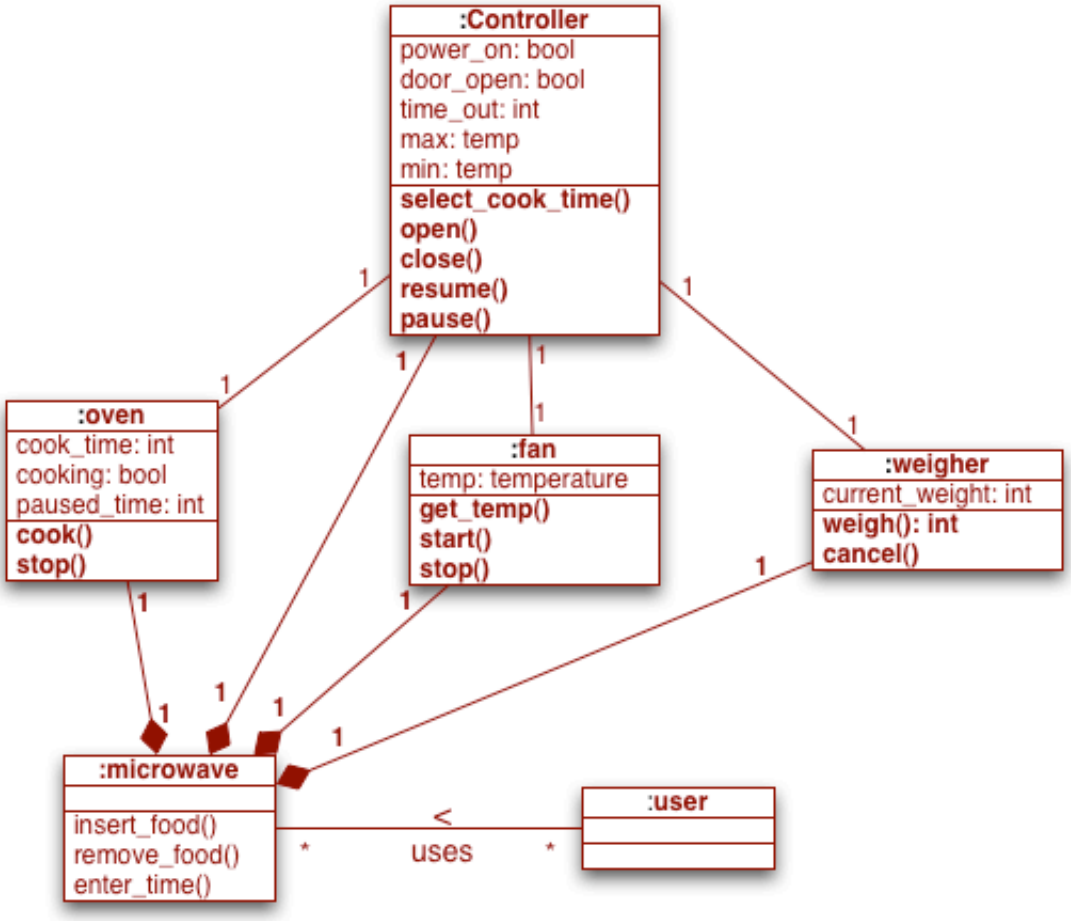
Two changes needed – (1) a guard [weight > 0g] on the "select cook time" transition, and (optionally) a transition from ready to an error state if this guard is not true. (2) need a transition from loaded back to weighing, in case food is removed again while door is open.

4. [Sequence Diagrams – 20 marks] Draw a sequence diagram to illustrate a sequence of interactions between the user and the main components of the microwave oven described in the previous question. Your sequence should start in the **off** state, and describe the scenario in which the user opens the door to put some food in, cooks the food for some time, pauses the cooking, opens the door to remove the food, and ends by closing the door again. Your diagram should clearly show the interactions between *the user* (who initiates the events), the *controller* (which provides the only user interface), the *oven* (which actually cooks the food), the *weigher* (which weighs food while the door is open), and the *fan* (which cools the oven whenever the door is open). State any assumptions you make.



Assumes weigher and fan are only active when door is open, and that the controller is responsible for telling them when the door opens and closes.
Assumes that only the weigher can detect food insertion and removal, i.e. the controller does not detect these events directly.
Assumes that weigher is active until it detects a change in weight that results in a non-zero weight. – it then reports this weight back to the controller

5. [Class Diagrams – 20 marks]. Draw a class diagram to capture all the classes involved in the statechart and sequence diagram from the previous two questions. Be sure to include all associations, attributes and methods necessary for the classes to interact in order to provide all the behaviours described on the statechart and sequence diagram. Also include multiplicities on all associations. State any assumptions you make.



Assumes that a microwave oven only has one fan, one weigher, one controller, etc.
Assumes that the microwave itself routes user events to the appropriate component.

Note: solutions that omit the composite "microwave" class are okay, as long as the user actions can be routed to the appropriate component via a unique association path.
Note: Names of classes and methods must be consistent with answers in previous questions.

(Scratch paper)

(Scratch paper)

(Scratch paper)