SE463 / CS445 / CS645 / ECE451

Fall 2009 — Final exam

14 December 2009, 12:30pm–3:00pm

Instructor: M. W. Godfrey

No aids allowed (*i.e.,* closed book).

There are 5 questions for a total of 100 marks.
Plan your time wisely.

Answer all of the questions on this exam paper.

In the immortal words of the late Douglas Adams,

Don't Panic!

Q1		/ 14
Q2		/ 20
Q3		/ 20
Q4		/ 30
Q5		/ 16
TOTAL		/ 100

1. **[14 total marks]** Short answer

   (a) **[2 marks]** What is the unofficial three word motto of this course, as mentioned in class many times. Hint: The middle word is "not".

   (b) **[2 marks]** List two kinds of stakeholders, apart from users, clients, and customers.

   (c) **[4 marks]** Give an example situation of two non-functional requirements that might cause a conflict if you wanted to optimize them both. Explain why.

(d) **[2 marks]** Briefly explain what a "motherhood" NFR is. Give an example.

(e) **[2 marks]** List two kinds of stakeholder-based elicitation techniques.

(f) **[2 marks]** The Unified Process consists of four main phases. Name one that is *largely* concerned with requirements and their analysis.

(g) **[No marks, but you get bragging rights]** With respect to question 5, why is the motto of the company appropriate to its name?

2. **[20 total marks]** Answer **True** or **False**. Each correct answer is worth 2 marks; each incorrect answer is worth -1, but you cannot get less than zero in total.

   (a) In a state diagram, a do-activity that is associated with a state is considered to be interrupted if an event occurs that the state has an outgoing transition for.

   Answer: _____

   (b) A (natural language) sentence about the domain is in the *indicative* mood, asserting truths about the domain, describing the world as it is, independent of any computation placed in it.

   Answer: _____

   (c) It is important during a brainstorming session to be highly critical of suggestions as early as possible in the process.

   Answer: _____

   (d) In general, a scenario within a use case should end with the first step that fails.

   Answer: _____

   (e) A GUI-independent specification of a user interface is generally preferred because although it's more work to create initially, it's usually easier to maintain as low-level changes to the UI may not impact it.

   Answer: _____

   (f) "Quality attribute" is, roughly speaking, a synonym for "non-functional requirement".

   Answer: _____

   (g) If a team is practising agile development, in theory they should not be performing any requirements-related tasks in the later iterations.

   Answer: _____

   (h) The main goal behind *model checking* is to verify that a given user interface (typically specified as a state or activity diagram) meets a particular set of usability constraints.

   Answer: _____

   (i) If an event occurs while two concurrent substates are active at the same time, the first substate to react to that event will "consume" it, making it impossible for the other substates to react to it also.

   Answer: _____

   (j) A use case actor need not be human; it could, for example, be another computer system that is external to the SUD, or even a trained chimpanzee named Bonzo.

   Answer: _____

3. **[20 total marks]** State machine modelling

   Create a UML state machine diagram for the control unit software for a piece of executive furniture called a RumbleSeat, which is a reclining chair with a coin-operated massage function built in.

   The normal use of the chair is this: A client arrives, sits in the chair, inserts a one dollar coin, and then the chair gives him/her a rumbling massage for 10 minutes.

   If the client inserts another one dollar coin while a massage is in progress, the coin should be returned immediately. (If someone inserts a coin at any time that is not a one dollar coin, it will be returned immediately by the coin mechanism without the rest of the system being informed, i.e., assume the hardware handles automatically rejects "slugs", so you don't have to worry about them in your model.)

   The RumbleSeat has an emergency stop button; when pressed, it sends an *emergStop* signal to the control unit. If the button is pressed while the chair is massaging a client, the chair should stop the massage; the client loses the remainder of his/her time.

   There are three hardware mechanisms that the control unit interacts with:

   - The coin mechanism responds to two signals from the control unit — *acceptCoin* and *rejectCoin* — and sends one — *coinReceived*.
   - The massage mechanism responds to two signals from the control unit: *startRumbling* and *stopRumbling* (the rumbling massage).
   - The emergency stop button sends one signal to the control unit: *emergStop*.

   You should break your state machine model down into two concurrent pieces: one modelling the coin handling, and one controlling the massage mechanism. Do not try to specify the behaviour of the hardware mechanisms; just specify how the control unit should interact with them.

   Hint: You probably don't need any UML do-activities, but you might want to use some entry and exit actions. You also might need an internal signal or two for co-ordination. Use the UML built-in facilities for modelling time.

4. **[16 total marks]** Temporal and time-based logic

   (a) For each of these statements, write a formula where time is an explicit parameter of each predicate e.g., $P(i)$ means that P is true at time i, where i is a non-negative integer.

      i. P and Q are never true at the same time.

      ii. P is true at most once.

      iii. If P ever becomes true, it will stay true forever from that point on.

      iv. Assume that R means *"I have regrets"*, that each time unit is one day, and that I am immortal. Translate this: *"I won't have regrets today or tomorrow, but I will soon and (once I have regrets for the first time) for the rest of my life."*

   (b) For each of the (same as above) statements, write a formula using temporal logic operators.

      i. P and Q are never true at the same time.

      ii. P is true at most once.

      iii. If P ever becomes true, it will stay true forever from that point on.

      iv. Assume that R means *"I have regrets"*, that each time unit is one day, and that I am immortal. Translate this: *"I won't have regrets today or tomorrow, but I will soon and (once I have regrets for the first time) for the rest of my life."*

5. **[30 total marks]** Use case and domain modelling

   Read through the following text carefully. At the end, you will be asked to create a use case model and a domain model. We strongly advise you to devise your solution on scrap paper, and then copy it over only when you are happy with it. Also, clearly state any assumptions that you make.

   The new electronic auction house ibid.ca (motto: "We're just like all the others!") wants you to create a partial specification for their proposed web-based application, which we'll refer to as the SUD.

   You can assume that the SUD has access to a program called *UIDgen* that generates unique identifiers (UIDs) on demand (i.e., each time it's called, it generates a new identifier that it has never generated before; a simple but insecure version of this would be to just return the value of a counter that is incremented with each use).

   A new user can register with `ibid.ca` by providing an email address to the SUD; in turn, the SUD will asks the *UIDgen* program for a new UID and forwards it to that email address as the User's UID (or UUID). An existing user should also be able to ask the SUD for a reminder of their UUID by providing their email address.

   A registered user can login to the SUD by providing their email address and UUID. If the login attempt is successful, the user is considered to be validated until they explicitly log out. Validated users can perform these actions:

   - `addItem (description, reservePrice, deadline)`
   - `removeItem (OAUID)`
   - `bid (OAUID, bidPrice)`

   In response to `addItem`, the SUD will generate an UID for the item using *UIDgen*; we call this the OAUID (*objet d'art UID*). The OAUID will be returned to the user. The SUD should also check that the deadline is in the future. In the case of `removeItem`, the SUD should check that the user is the one who put the item up for bid in the first place. In the case of `bid`, the SUD should check if the bid is greater than the maximum of the reserve price and the maximum bid so far. All three operations should send a confirmation email to the user, and `bid` should also send email to the item's seller.

   In addition to the actions by validated users, the `ibid.ca` website can also be queried by anyone to see what is currently up for auction:

   - `query (keyword)`

   In this case, the SUD will then perform textual matching of the keyword against the descriptions of all current auction items (likely using some library function similar to `grep`), and then returns full information for the set of all matching items.

   Finally, each day at 12:01 AM Eastern, the SUD checks for items whose auction deadline has just expired. For each such item, the item is removed from auction and email is sent to both the seller (with the winning bidder's email address) and the winning bidder (with the seller's email address).

   (Now see the next page.)

(a) **[15 marks]** Create a "brief" use case description of the SUD, as you did for deliverable #1. Start with a use case diagram on this page, then on the next page provide a short description of each use case.

(5a cont'd)

(b) **[15 marks]** Create a domain model for this problem domain, as you did for deliverable #2. Be sure to annotate actors with UML stereotypes.