University of Toronto

Faculty of Arts and Science
Dept of Computer Science

CSC340F – Information Systems Analysis and Design

December 2002
Instructor: Steve Easterbrook

No Aids Allowed
Duration: 2 hours

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) [Software Lifecycles – 5 marks] Name two alternatives to the waterfall model. What are the advantages and disadvantages of each model when used to manage a software development project?

Spiral model.
Advantages: allows for iterative development, with prototyping and risk management built in to the process. Disadvantages: Might be expensive (and slower) to do lots of iterations. Not clear what happens if there are unexpected changes in business priorities,

Incremental development.
Advantages: don't need to understand all the requirements before developing the first version. Lessons from early versions feed into later versions. Disadvantages: hard to plan for versions beyond the first. Lessons from early versions might be learnt too late. Might mislead customer if first version doesn't match many of their requirements.

[Notes: Other possible lifecycles models include: Incremental development, prototyping, V-model. Must have both advantages and disadvantages for two different models to get full marks.]

(b) [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 core functions (e.g. business logic) from functions that are platform dependent (e.g. data storage) and also from how the system interacts with users (e.g. the user interface). 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 layered architecture has three layers:

Presentation layer is 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 is 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).]

(c) [Legal Issues – 5 marks] Under copyright law, at what point is information considered to be copyrighted? Suggest two ways in which copyright law can affect an information systems analysis project, and in each case, explain the professional responsibility of the analyst.

Under copyright law, information is considered to be copyrighted as soon as it is fixed in a tangible medium (e.g. written down, entered into a computer system, etc), whether or not a copyright notice is included.

1) Copyright law can affect an information systems project because some of the information (e.g. pictures, text, etc) to be stored by the system may be copyrighted. If the system provides wide access to this information, it may infringe the copyright of the owner of the information. It is the responsibility of the analyst to check whether copyright protection might apply to any of the information to be stored in the system, to explain to the customer all such issues, and to secure agreement of the copyright owner when necessary, or propose a design that avoids the copyright infringement.

2) Another way in which copyright law can affect a project is that information to be used for the analysis (e.g. source documents, photos, etc) might be copyrighted. In this case including them in reports written for the customer may infringe the copyright. It is the responsibility of the analyst to check the copyright of any source material, and secure permission to use it before including it.

[Notes: other suggestions are possible, give credit for sensible suggestions, as long as analyst's responsibilities are identified.]

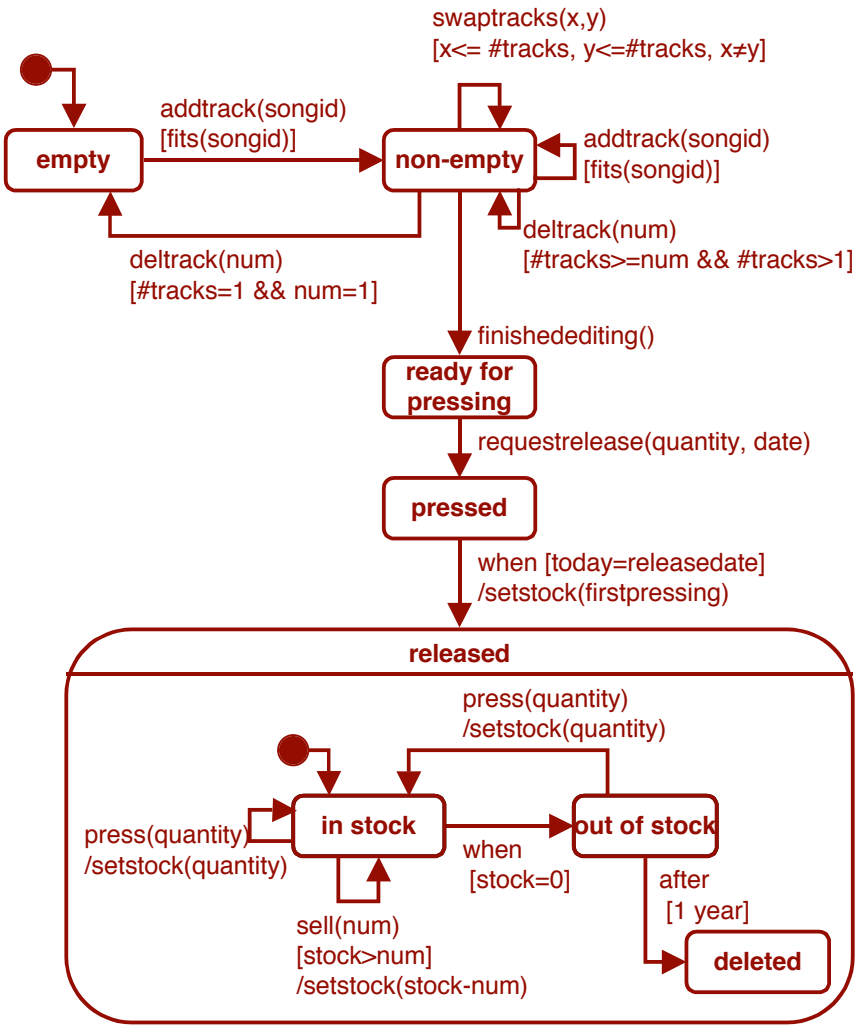
(d) [Specifications – 5 marks] Project managers sometimes regard work put into writing high quality specifications as "gold plating", and claim that it is unnecessary as it doesn't contribute to producing program code. Under what circumstances is this view sensible, and under what circumstances is it foolish? In the latter case, how would you persuade such a manager that the specification actually does need to be high quality?

This view is sensible for small projects where there is a well understood problem to be solved, and where the analysts exploring the requirements will also implement the system (i.e. the specification does not have to bridge this gap.
This view is foolish for big projects, where there is a large team and lots of stakeholders. In this setting, proper communication among the team, and with the stakeholders is important, and the specification is used to make sure everyone understands the problem fully. Higher quality specs should result in easier integration and higher quality software.

If the list of benefits above doesn't convince the manager, then one could look for articles and research reports that describe the role of a high quality specification in large projects. One could also use anecdotal evidence from past projects (e.g. identify problems on a previous project that could have been avoided with a good specification).

[Notes: give credit for any reasonable suggestions for how to persuade the manager]

2. [StateChart Diagrams – 20 marks] The Verging Record Company (VRC) is planning a new information system to assist with the creation and distribution of CDs. Draw a statechart diagram to represent the following behaviour of an object representing a CD in VRC's system. When a CD object is first created, it is "empty". The artist can add a track to it, and if it fits, the CD is then "non-empty". The artist can continue to add tracks to a non-empty CD, as long as they fit. She can also delete tracks, or swap the order of the tracks. If she deletes a track and there are none left, the CD becomes empty again. When she has finished editing, the artist will declare the CD to be "ready for pressing", after which no more changes can be made. Once a CD is "ready for pressing" the record company may request a certain number (the "initial run") of copies to be pressed, at which time they also set a release date. The CD is then "ready for release". It stays in this state until the release date. Once the CD is released, it stays released for the rest of its life. However, during this state, the CD can either be in stock (i.e. copies are available for sale), out of stock (i.e. no copies are available, but the record company may create another pressing soon, to put it back in stock), or deleted. Deletion occurs automatically if and only if the CD has been out of stock for more than a year, after which no further pressings can be made. Note: a deleted CD is still considered to be 'released'. State any assumptions you make.

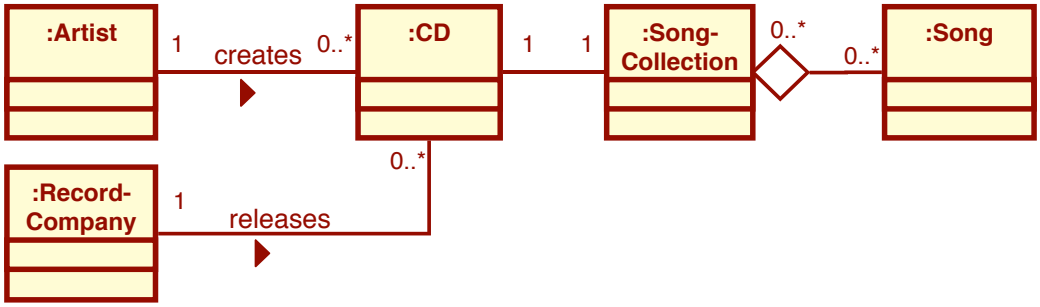


Assumes that no operations other than those shown are available in each state, and that suitable error messages or warnings are generated if such operations are called.

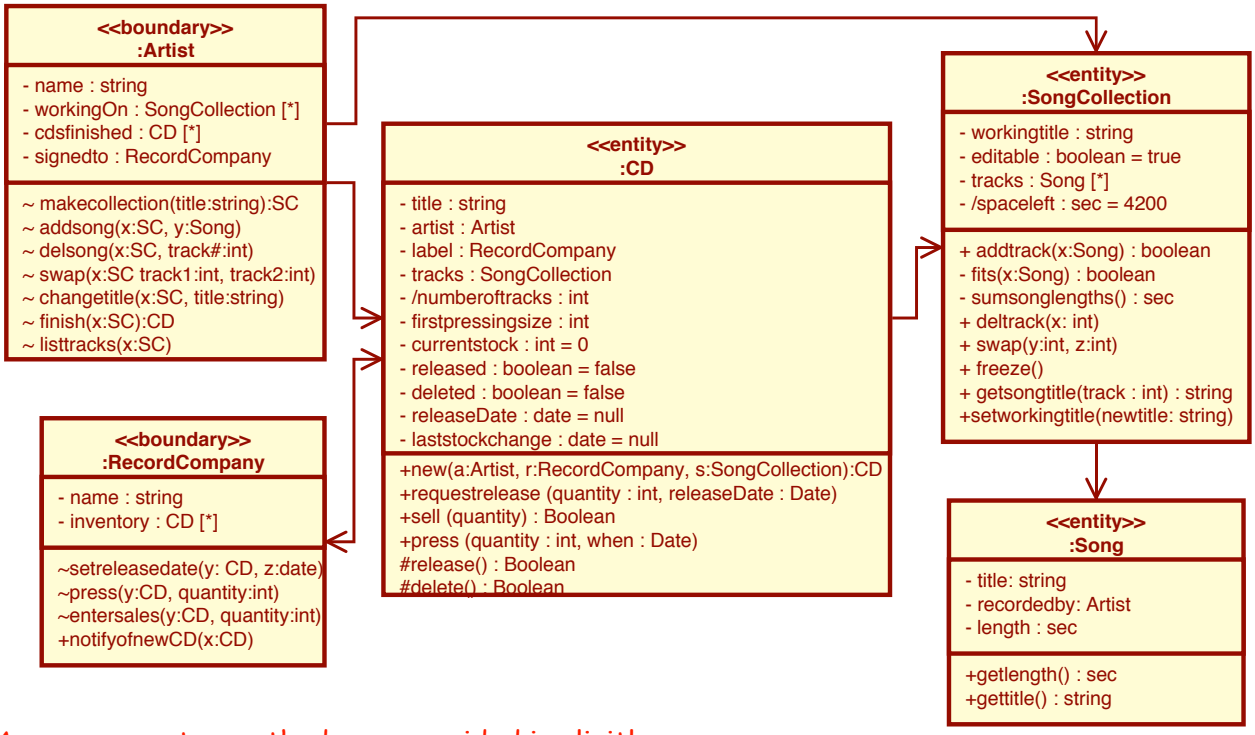
Assumes that objects in the system representing CDs are never destroyed (hence no end state). If this assumption isn't made, then there should be an end state accessible from anywhere.

[Notes: allow other sensible choices for names of operations, and choices of which operations are events and which are generated actions. Must have suitable guards on the addtrack and deltrack operations.]

3. [Object Design – 20 marks] Here is an analysis class model for VRC's proposed system.

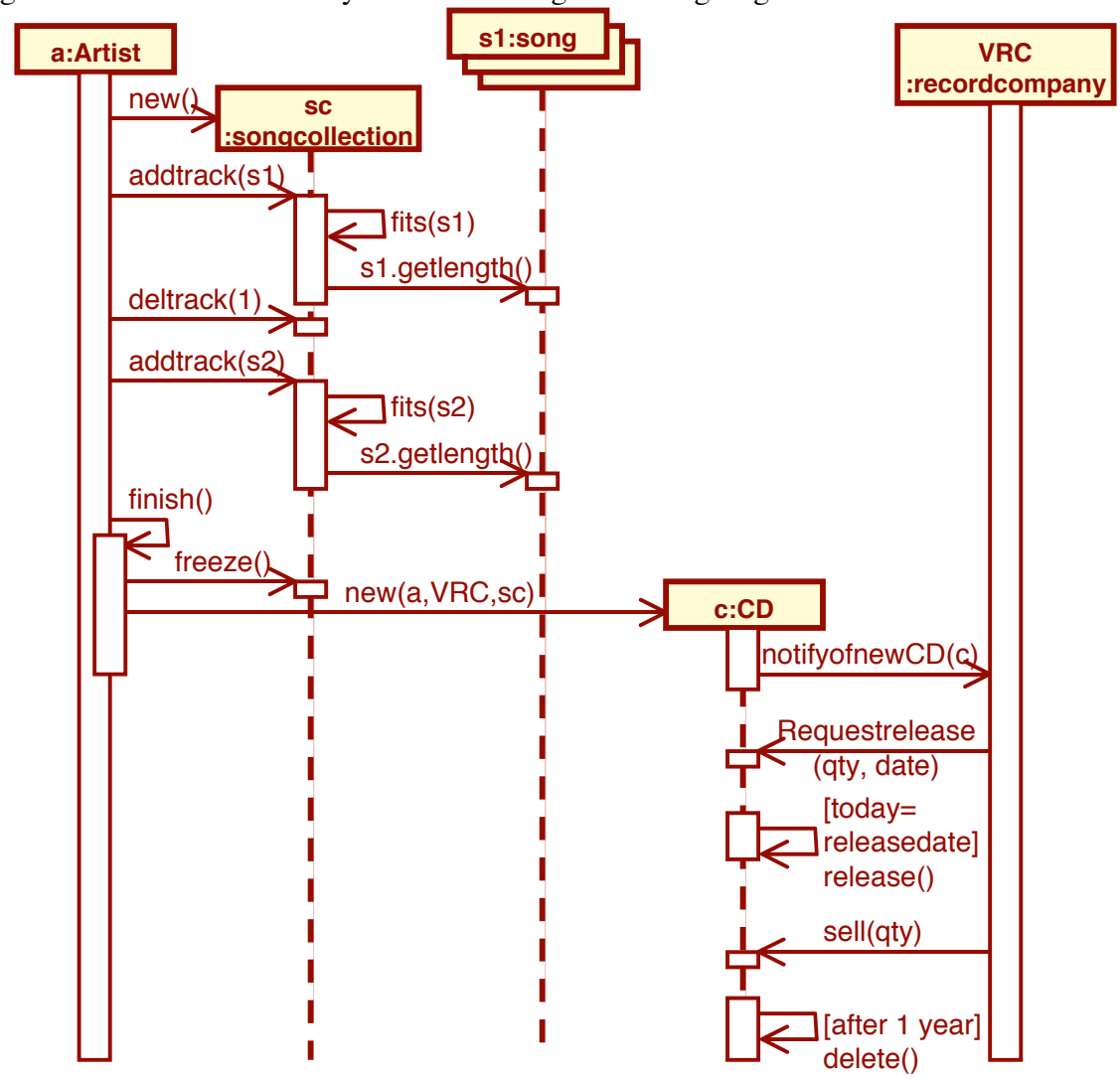


Create a detailed design for these five classes. Be sure to define all the attributes and operations needed to implement the behaviour described in the previous question. Your answer should include: type definitions for all attributes, operations and parameters; visibility for all attributes and operations; and navigation arrows to show which (classes of) objects will call other (classes of) objects' operations. Assume that an empty CD can hold 70 minutes of music. Explain any other assumptions you make, and any design decisions you take.



Assumes creator methods are provided implicitly.
Assumes that record company and artist are boundary (user interface) classes, with (~) operations only available from within the user interface package.
Design decision: contain all the editing operations in SongCollection – an artist creates a song collection first, edits it as needed, and then the corresponding CD object is created when the artist selects the finish() method. The CD creator method adds the newly created CD object to the record company's inventory, by calling notifyofnewCD().
[Note: many alternatives are possible, depending on where responsibility for editing the contents of a song collection sits. E.g. An alternative would be for Song to be responsible for adding itself to a CD, so the add song message would go straight from Artist to Song, and then Song would call the appropriate method of songcollection to get itself added. All the operations shown on the statechart from the previous question must be included somewhere].

4. [Sequence Diagrams – 20 marks] Draw a sequence diagram showing a typical sequence of interactions between objects of the five classes given in the previous question. Your sequence diagram should cover the typical life of a CD, from creation to deletion, with songs being both added and deleted by the artist during the editing stage.



Assumes that returns from method calls are implicit
Assumes the design shown for question 3, that the editing is done within songcollection, and the CD object isn't created until the artist calls the finished() method. Note that this design contradicts the statechart for question 2, which would have to be revised!
Assumes that it is okay to release a CD with only one track on it (maybe a CD single?)
Assumes that the songs pre-exist.
Assumes the CD objects are responsible for monitoring for release dates and delete dates. Alternative design could include a separate clock or timer object.
Note: This answer doesn't show use of the swap method, nor some of the other methods shown in question 3, but then this question doesn't explicitly ask for any of them.
Note: Many other designs are possible. Typical designs will assume that CD objects are responsible for the adding and deleting of tracks, in which case "c:CD" will be created first, and will receive addtrack() and deltrack() messages, and pass these on to sc:SongCollection.
Note: message passing should be between objects, not classes, hence the boxes at the top of each timeline name an instance, rather than just the class (we did not penalize for this)

5. [Relational Database Design – 20 marks] Here is the relational database schema (with some sample data) proposed by a database designer who is creating a database that keeps information about product orders for a small auto parts company:

Order(order#, date, customerID, lastName, firstName, province, ProvincialTaxRate, OrderTotal)

order#	date	customerID	lastName	firstName	province	Provincial-TaxRate	OrderTotal
239	23/06/2001	1135	Black	Conrad	Ont	8%	$138.56
260	12/09/2001	1135	Black	Conrad	Ont	8%	$82.03
297	30/04/2002	1577	Wong	Harry	Que	7%	$75.21

OrderContents(order#, product#, productDesc, quantity, price)

order#	product#	productDesc	quantity	price
239	555	Nut	35	$1.02
239	444	Bolt	35	$2.17
297	444	Bolt	25	$2.17

Identify all likely functional dependencies (and describe any assumptions you make about the data).
Which of the following normal forms is this schema in?
b) First Normal Form (1NF) – none of the relations have composite attributes
c) Second Normal Form (2NF) – 1NF and every non-key attribute is fully dependent on the primary key
d) Third Normal Form (3NF) – 2NF and no dependencies between non-key attributes.
Describe any transformations needed to get the schema into 3NF.

Functional dependencies in the Order relation:
order# → date,customerID,lastName,firstName,province,ProvincialTaxRate,OrderTotal
customerID → lastname, firstName, province, taxrate
(assumes customers never move between provinces; alternatively, province might only depend on order# if we assume customers can made orders from different provinces)
province → taxrate
Functional dependencies in the OrderContents Relation:
order#, product# → quantity
product# → productDesc, price
(assumes price is "price per unit of the product", and products have a fixed price)

The schema is in 1NF because of the part key dependency: product# → productDesc, price
(it's okay to say that Order is in 2NF and OrderContents is in 1NF)
(also assumes that date is atomic with respect to this application, otherwise the schema wouldn't even be in 1NF)

Transformations needed to get to 3NF:
- get OrderContents to 3NF by pulling out the things that depend on product#
  product(product#, productDesc, price)
- get Order to 3NF by pulling out the dependencies on customerID, and the dependencies on province:
  customer(customerID, lastname, firstname, province)
  taxrates(province, taxrate)
- This leaves the two original schema as:
  Order(order#, date, customerID, ordertotal)
  OrderContents(order#, product#, quantity)

[Notes: must give a complete set of normalized schema, with suitable keys identified, to get full marks. Allow alternative assumptions about the data where reasonable.]

(Scratch paper)

(Scratch paper)

(Scratch paper)