McGill

April 2008
Final Examination

Introduction to Software Engineering

ECSE 321

9:00am April 17, 2008

Examiner:	Michael Rabbat	Assoc Examiner:	M Maheswaran
	398-1847		398-1465

Student Name:	Solutions	McGill ID:									

INSTRUCTIONS:

- This is a **CLOSED BOOK** examination.
- You are permitted **TRANSLATION** dictionaries ONLY.
- This examination consists of 10 questions worth a total of 100 marks. Questions 1-9 are short answer questions worth a total of 55 points. Question 10 is a longer case study consisting of five parts and is worth 45 points.
- Some of the questions consist of multiple sub-problems. The marks associated with each question or sub-problem are shown in square brackets.
- This examination has a total of 16 pages, including the cover page.
- **SPACE IS PROVIDED** on the examination to answer each question. **ANSWER DIRECTLY ON THIS EXAMINATION.**
- This examination is **PRINTED ON BOTH SIDES** of the paper.
- **Show all your work and explain your answers.** Please use only the space allocated in this examination booklet. Several questions require a written explanation. Be concise and precise, and do not use more space than allocated to you.
- This examination paper **MUST BE RETURNED.**

Problem 1 – Requirements [6 marks]

You have been contracted to build a cash register system, called CHECKOUT, for a local grocery store. After meeting with the client you establish the following list of requirements. In the space following each statement, label the requirement as functional or non-functional, and for non-functional requirements identify which category they belong to (usability, reliability, performance, supportability, implementation, interface, operations, packaging, or legal). [Each part is worth 1 mark]

a. CHECKOUT has two displays, one display facing the customer which shows the running total and the cost of the most recently scanned item, and one display facing the checkout worker which also has additional controls. Functional

b. When charging for fruits or vegetables, the checkout worker places the item on a scale and selects the type of product from a list on their display. CHECKOUT then calculates the price to charge and adds the cost to the bill. Functional

c. The scale should be accurate to within 0.01 Kg. Non-functional, performance

d. The display facing the worker shows both the name and a picture of all acceptable types of fruits and vegetables. Functional (although usability was accepted also)

e. CHECKOUT should displays item names in both French and English. Non-functional, usability

f. CHECKOUT should be up and running 99.99% of the time, and all transactions must be backed up with triple redundancy in case of an audit. Non-functional, reliability

Problem 2 – Writing Good Requirements [5 marks]

During requirements elicitation and analysis, our goal is to come up with a system description which is complete, consistent, unambiguous, correct, and traceable. Explain what is meant by each property in the space provided. [1 mark each]

Complete Includes *all* user requirements

Consistent No contradicting requirements

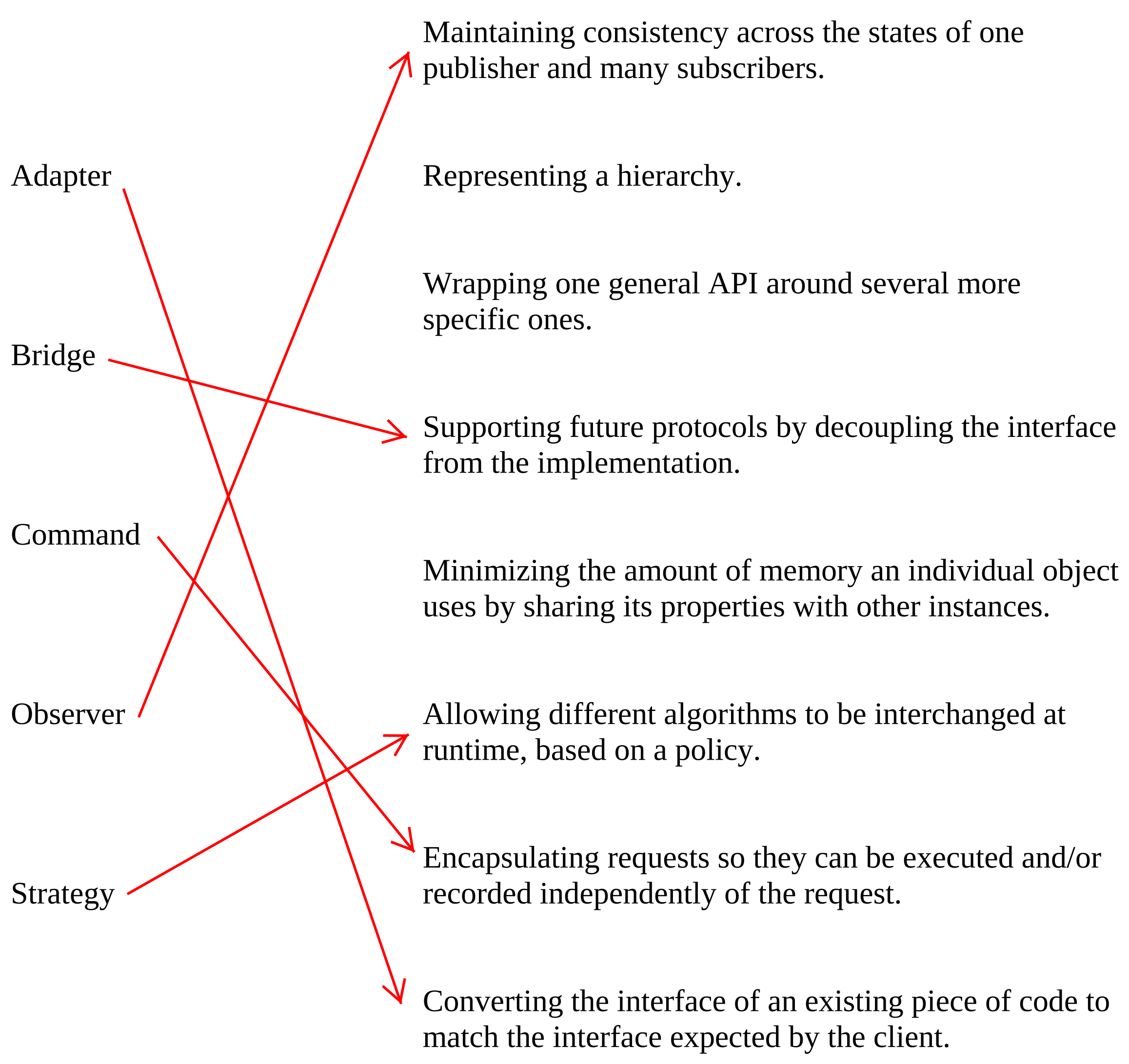
Unambiguous There is only one way to interpret each requirement

Correct Requirements math user/client/customer's vision for the system capabilities and operation

Traceable Can trace each requirement back to its origin, and can also trace use cases and tests back to their corresponding requirements

Problem 3 – Design Patterns [5 marks]

Draw a line connecting each design pattern name on the left to the description that best matches it on the right. (Note: There are more descriptions than design pattern names. Your response should only involve drawing five lines.)
[1 mark each]



Design Pattern Name

Adapter

Bridge

Command

Observer

Strategy

Description

Maintaining consistency across the states of one publisher and many subscribers.

Representing a hierarchy.

Wrapping one general API around several more specific ones.

Supporting future protocols by decoupling the interface from the implementation.

Minimizing the amount of memory an individual object uses by sharing its properties with other instances.

Allowing different algorithms to be interchanged at runtime, based on a policy.

Encapsulating requests so they can be executed and/or recorded independently of the request.

Converting the interface of an existing piece of code to match the interface expected by the client.

Problem 4 – Definitions [5 marks]

Complete the following sentences or definitions. [1 mark each]

a. Greenfield engineering is the process of starting a new project from scratch .

b. The Liskov substitution principle states that S is a subtype of T if S can be substituted everywhere T is expected .

c. In blackbox testing we check that the actual output matches the expected output for a given input. .

d. Regression testing is redoing all previously passed tests after making a change or fixing a bug in order to make sure that previously working functionality has not been broken

e. Extreme programming is the right methodology to use when user requirements are expected to change frequently during the project lifetime, the project team is small, and the project team has easy access to the customer for frequent feedback

Problem 5 – Prototypes [5 marks]

What is the difference between a vertical and a horizontal prototype? Describe a concrete example scenario where one would use a vertical prototype.

A horizontal prototype usually illustrates what the user interface will look like, without any of the features being functional. (E.g., buttons do nothing when clicked, but all menu items are shown.)

A vertical prototype focuses on one feature or use case and implements the functionality for that feature completely. (E.g., to assess the complexity involved in implementing a feature, or to gauge whether it will be feasible to satisfy a performance requirement.

Problem 6 – Class Diagrams [5 marks]

For this problem, consider the UML class diagram shown in Figure 1. For each pair of objects listed below the figure, describe the relationship between objects as either “generalization”, “delegation”, “composition”, “aggregation”, or “association”.
[1 mark for each part]

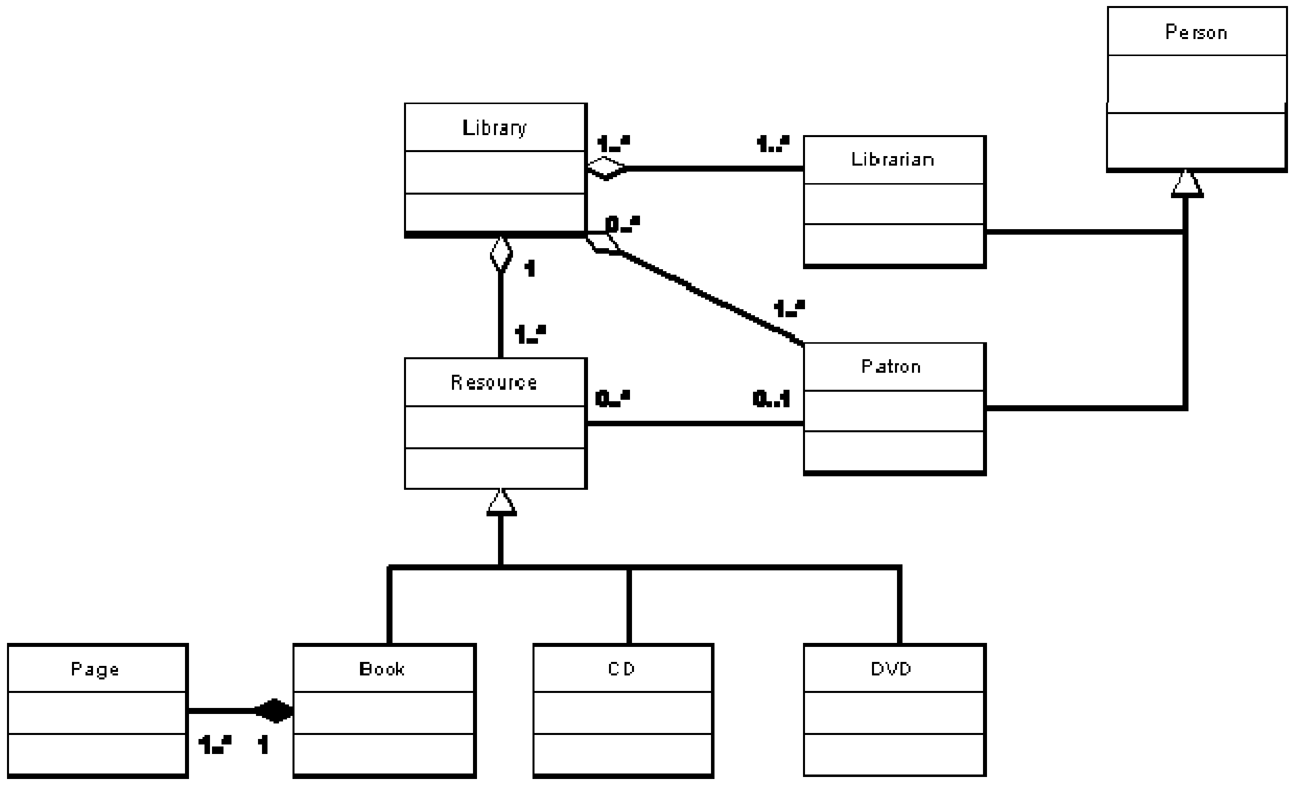


Figure 1: UML Class diagram for a Library.

a. Book and Resource: generalization

b. Library and Patrons: aggregation

c. Book and Pages: composition

d. Patron and Resources: association

e. Librarian and Person: generalization

Problem 7 – Contracts and OCL [8 marks]

Consider the Library system depicted in Figure 1. Suppose that every Resource has a private idNumber attribute, which is a positive integer, and a private Boolean variable available, indicating whether or not the Resource is checked out. Suppose, further, that Resources have public methods: checkOut(), checkIn(), and isAvailable().

a. Using the Object Constraint Language (OCL), specify a contract for the Resource object interface with *at least 5 different constraints (*i.e., 5 OCL statements).
[5 marks]

context Resource **inv:** self.idNumber > 0
context Resource::checkOut() **pre:** r.available = true
context Resource::checkOut() **post:** r.available = false
context Resource::checkIn() **pre:** r.available = false
context Resource::checkIn() **post:** r.available = true

b. Suppose that Book idNumbers lie in the range 1..1000, CD idNumbers lie in the range 1001..2000, and DVD idNumbers lie in the range 2001..3000. Does this violate the principles discussed in class for inheriting contracts? Explain why or why not.
[3 marks]

No, this does not violate the principles discussed in class.

Invariant conditions can be the same or stricter in the child class. Since these ranges are more restrictive (stricter) than just being positive, this is ok.

Problem 8 – True or False [10 marks]

Classify each of the following statements as True or False, and justify your response. (If the statement is false, explain what needs to be changed to make it true. If it is true, explain why.) [2 marks each]

a. During requirements analysis we typically draw one UML state-chart diagram for each use-case in the analysis model.

True or [✓ False] Justification: One state chart for each object
One sequence diagram for each use case

b. In general, it is good practice to use the Façade design pattern to encapsulate the service interface for a subsystem whenever possible.

[✓ True] or False Justification: Good practice

c. When a subclass S inherits a contract from its parent, T, the preconditions on a method inherited by S must be equivalent to or stricter than those on the same method in T.

True or [✓ False] Justification: Should be weaker

d. The Three-tier architecture is used to separate the internal representation of data from the displayed version of the data, and control actions executed on the data.

True or [✓ False] Justification: Should be model/view/controller

e. Coupling describes the amount of association between objects within a single subsystem.

True or [✓ False] Justification: Coupling = associations between subsystems
Coherence = associations within a subsystem

Problem 9 – Coding Conventions [6 marks]

In the following function, which lines do not follow proper coding conventions?

```
00  public static long abs (long x)
01  {
02      long y;
03
04      if(x == 0)
05          y = 0;
06      else {
07      if (x < 0) {
08              y = -1 * x;
09          } else { y = x; }
10      }
11
12      return y;
13  }
```

List the lines and errors below. [1 mark for each error *you* identify.]

Line Number	**Error Explanation**
00	The space between abs and the leading parenthesis should be removed
01	Bracket on same line as method declaration
02	y is a bad variable name (ambiguous)
02	Attribute variable y should be declared private
04	Should be a space between if and opening parenthesis
05	Body of if-else statement should be enclosed in curly braces
07	Should be indented one level
09	Body of the else statement should be on a separate line

Problem 10 – Requirements Analysis and System Design [45 marks]

This question has five parts which are all based on the following system description:

ParkingMontreal.com is a new system to track the availability of parking spots around the city of Montreal. This system will provide services both to the public, by helping them find nearby available parking spots, and to the City of Montreal parking attendants, by helping them locate parked cars whose meters have expired. Users who are registered with ParkingMontreal.com can also pay for their parking spot using their cellular phone. And, of course, the general public still has the option to pay for parking using the existing kiosk-based system.

Public users interact with the system either via an internet-enabled handheld device such as a Blackberry or iPhone, or via SMS messages from their cell phone. For example, suppose a user has a Blackberry device and wants to find a nearby parking spot. They direct their browser to www.ParkingMontreal.com, and query the system by entering the intersection of their current location. If the user's phone is GPS enabled, then the GPS coordinates are automatically entered instead of the intersection information. If a user does not have a web-enabled phone, then they can send an SMS containing their location to 514-555-PARK. When the ParkingMontreal.com system receives a query, it returns the 3 nearest available parking spots in its database and provides driving directions to the first choice (the closest spot).

After parking their car, if the user is already registered with ParkingMontreal.com, then they can pay for their parking spot by sending an SMS message with the parking spot identifier (e.g., M631), and the amount of time they would like to purchase (maximum 2 hours). When a user registers with the ParkingMontreal.com website, in addition to their local address and email, they provide their mobile phone number, the license plate number of their car, and credit card information. At the end of each month, their credit card is automatically charged for any parking fees incurred, and a billing statement is sent via email. Also, at any time, a user can log into their ParkingMontreal.com account over the internet to view the history of their parking charges.

ParkingMontreal.com users also have the option to receive warning messages. Ten minutes before their time is up, the user is sent an SMS message notifying them that their parking meter is about expire. At that time, they have the option of paying again to renew their spot.

In order to detect which spots are being used automatically, a massive network of sensor devices will be deployed around the city of Montreal. Each node is responsible for detecting the presence or absence of vehicles in two neighboring parking spots on the street. These nodes will operate on a sleep-wake cycle to conserve battery power. Once a minute, the node will activate and sense for the presence of vehicles in the two spots it is responsible for. If the state of a parking spot has changed since the last activation period (i.e., in the last minute, either a new vehicle arrived or the previous vehicle departed), then the node transmits a message via WiFi back to ParkingMontreal.com, to update the database.

ParkingMontreal.com also provides information to the City of Montreal, to help them identify and fine people who park without paying. Parking attendants driving around the city carry GPS-enabled tablet PCs with a specialized interface that automatically shows them which occupied spots have been paid for, and which are in violation. A spot which has been occupied for at least 5 minutes without being paid for is in violation. The parking attendant display uses coordinates from the GPS system to center a map showing the current location, and the state of every parking spot in the area (unoccupied, paid for, or violation). In addition, the attendant can click on a particular spot and see how much time is remaining on the meter for that spot. The current ParkingMontreal.com system only provides these limited features for city parking attendants. In the future, the system will also use the information about how much time is left on each meter and which spots are in violation for a particular neighborhood in order to plan out a schedule for where the city parking attendants should go in order to maximize the number of violators that get ticketed.

In general, the answers to all parts of Question 10 have many different answers which are all correct. The answers shown below are just one possibility.

Part A – Requirements Analysis [6 marks]

Respond to each of the following statements by quoting directly from the text on the previous page.

Identify two functional requirements. ____________________

1) Public users interact with the system either via an internet enabled handheld device such as a Blackberry or iPhone, or via SMS messages from their cell phone.

2) When a user registers with the ParkingMontreal.com website, in addition to their local address and email, they provide their mobile phone number, the license plate number of their car, and credit card information.

Identify two non-functional requirements. ____________________

1) At any time, a user can log into their account and view an account history.

2) These nodes will operate on a sleep wake cycle to conserve battery power.

Identify a requirement that is currently ambiguous and needs further clarification. ____________________

If a user doesn't have a web-enabled phone then they can send an SMS containing their location to 514-555-PARK.

For the ambiguous requirement identified above, what question would you ask the client in order to resolve the ambiguity? ____________________

How is the user supposed to specify his location? E.g., street number, intersection?

Part B – Architecture [8 marks]

What architecture (or combination of architectures) would you use in the design of this system? Justify your response. Recall the architectures discussed in class: repository, model-view-controller, client/server, peer-to-peer, three-tier, four-tier, and pipe-and-filter.

I would go with a repository architecture for the following reasons:
- It provide bi-directional communications for public users (e.g., user wants to query for available parking spots, but system also needs to notify user 10 minutes before his/her spot expires).
- Publishing updated information to subscribers (parking attendants) is possible, so they can know which spots are empty or about to expire soon.

Part C – Subsystem Decomposition [12 marks]

List the subsystems for ParkingMontreal.com. Briefly explain the role of each subsystem.

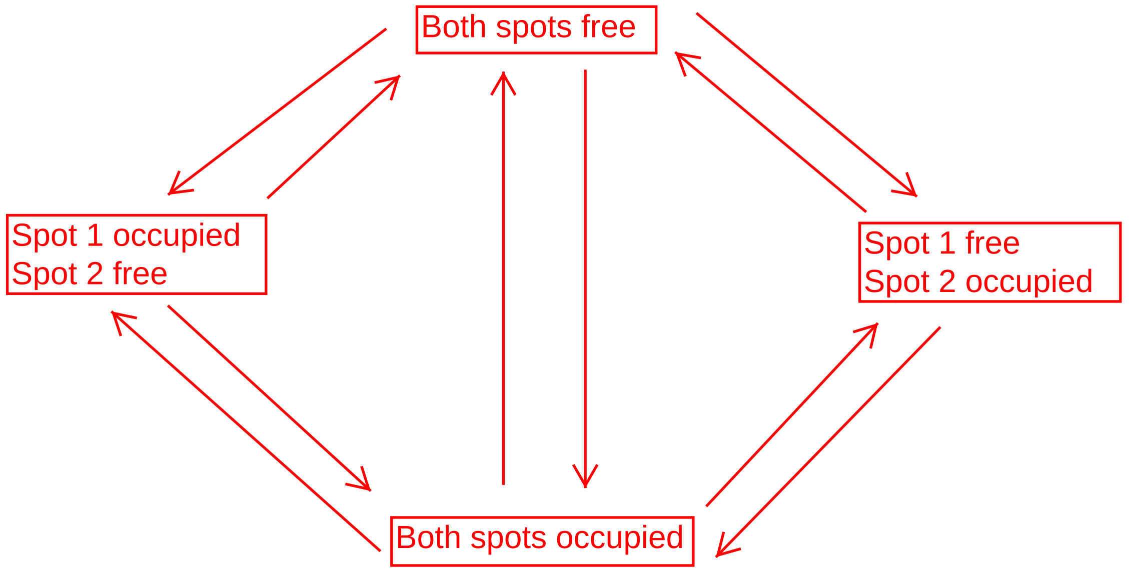
This part is fairly open-ended and there are many different was it could have been approached. It is up to you to justify and argue for why you went with a particular subsystem decomposition.

Examples of some subsystems I would expect to see are:
- Sensor management subsystem : receive measurement updates from sensors
- Billing
- Timing and Notification : Message customers before their spots expire
- Database (repository) : to store entity data such as user accounts and current state of parking spots around town
- Spot location : Respond to user queries with three closest available spots

Draw the subsystem/deployment diagram.

Part C – State Diagram [8 marks]

The ParkingMontreal.com description mentions sensor nodes that will detect and report on the arrival and departure of vehicles at a parking spot. Based on this description (paragraph 5 on page 10 of the exam), draw a state diagram illustrating the operation of a single sensor node.



To properly complete the diagram, also annotate each arrow with the message transmitted back to the base. For example on the arrow from "Both spots free" to "Spot 1 free, Spot 2 occupied" indicate message "New vehicle arrived in spot 2"

Again, there are multiple options for each of these parts. Make sure you justify your response adequately.

Part D – Design Patterns and Reuse [6 marks]

Name two design patterns you would use in developing the interface between sensor nodes and the rest of the ParkingMontreal.com system. Explain how these patterns would be used, and why you chose these particular patterns.

1) Observer design pattern, where the attendant tablets subscribe to information about parking spots, so that when a sensor reports new state information to the repository, the attendant tablets are notified of the change.

2) The facade design pattern, since it is good practice to encapsulate the service provided by each subsystem into a facade class, in order to reduce coupling between subsystems.

Part E – Test Plan [5 marks]

In this final part of the problem, you will specify an integration test plan. Choose which integration testing strategy you would adopt for this project, and explain why. Recall the strategies discussed in class: big bang, bottom-up, top-down, and sandwich.

I would probably go with a bottom up test plan. Because this system involves interactions between so many different parts (user handhelds, sensors, attendant machines, database, web server...), it will be critical to get the repository and lower layers right from the start. Then the additional components can be incorporated in parallel.