Print Name:__________________________

Student ID#: __________________________

SOLUTION

CPSC 410 — Advanced Software Engineering
Mid-term Examination (Fall 1999)
Instructor: Gail Murphy

Do NOT start until you are informed you can start!

This examination has 7 questions. The question and answer space is from page 2 to page 9. Check that you have a complete paper when told that you can start the examination.

You have **80 minutes** to complete this exam. Before you start, you must write your name and student id number on the top of **every** sheet of this exam. We will not mark your exam without this information. You must also sign your name in the space provided below.

Write legibly. We also have to be able to read it to mark it.

This exam is closed book and closed neighbour. Notes, book, or other materials are not allowed. Candidates guilty of any of the following, or similar, dishonest practices shall be immediately dismissed from the examination and shall be liable to disciplinary action:

- Making use of any books, papers or memoranda, calculators or computer, audio or visual cassette players, or other memory aid devices, other than those authorized by the examiners.
- Speaking or communicating with other candidates.
- Purposely exposing written papers to the view of other candidates. The plea of accident or forgetfulness shall not be received.

If you have a question during the exam, raise your hand and I will come and answer your question.

The exam is out of 100 marks. The value of each question is indicated. Use your time wisely. If you do not know the answer to a question, move on to the next question and come back to the question at the end.

The exam is double-sided. There are questions on **both** sides of the page.

Good luck!

Signature:___________________________________

Print Name:___________________________

Student ID#: ___________________________

On each, 1 mark for T/F;

1 mark for justification.

1. ***(Total: 20 points. 2 points each.) True or false with justification. For each statement below, circle T if the statement is true or F if the statement is false. In the space provided for each statement, write at most two sentences explaining your choice.***

(a) There is always one dominant view (diagram) used to describe the architecture of a software system.

T or [✓ F.] Justification:

Many views are typically needed to describe a system's architecture. For example, a static view may help a developer reason about work assignments while a run-time view may help a developer reason about potential concurrency issues.

(b) In COM, interfaces can be reused.

[✓ T] or F. Justification:

An interface may be reused by being declared the interface as a sub-interface (subclass) of another interface. All interfaces must inherit from the IUnknown interface.

(c) Events are commonly used as connectors in a virtual machine architectural style.

T or [✓ F.] Justification:

Events *could* be used as such a connector, but it is unlikely because there is not likely much benefit to be derived from separating the components of a virtual machine and because performance is likely to degrade. A (procedure or method) call is more likely to be used.

(d) Forward references in a requirements specification are desirable because they make it easier for a user of the specification to understand how parts of the requirements are related.

T or [✓ F.] Justification:

As described by Meyer, *implicit* forward references are *not* desirable in a requirements specification because they make it harder to read and understand the specification. [Since explicit forward references are sometimes ok, I'll accept an answer of true if the justification explicitly refers to explicit references ☺]

(e) A formal approach to specification is not sufficient to ensure that a produced specification is consistent.

[✓ T] or F. Justification: A formal specification makes it possible to check that a produced specification is consistent, but unless those checks are made, the specification may be as inconsistent as a natural language specification.

(f) The batch sequential architectural style can help achieve the goal of modifiability when building a system.

[✓ T] or F. Justification:

Individual programs that are part of the batch sequential architecture may potentially be modified without affecting other programs. New programs may also be added into the system.

(g) In JavaBeans, interfaces are the primary means by which a client determines the operations available on a component.

T or [✓ F.] Justification:

Introspection is the primary means by which a client of a JavaBeans determines what operations are available on a bean.

(h) In the Z specification language, schemas are used solely to describe the dynamic aspects of a system.

T or [✓ F.] Justification:

Static schemas are used to describe the state space of the system.

(i) The design principle of abstraction is about dealing with different aspects of a problem individually.

T or [✓ F.] Justification:

The design principle of abstraction is about identifying the important aspects of a design problem at hand and ignoring the details. The principle of separation of concerns is about dealing with different aspects of a problem individually.

(j) Constraints placed on the architecture of a system at design time may be violated in an implementation.

[✓ T] or F. Justification:

The constraints may be violated to improve some aspect of the system, such as performance. For example, the layered architecture in the A-7E was not pure in order to accommodate future anticipated modifications.

Print Name:__________________________

Student ID#: __________________________

2. (Total: 15 points. 3 points each.) Short Answer. Define the following terms in five sentences or less each.

a. Uses relation between modules.

One module (A) uses another module (B) if a correctly functioning module B must be present in order for module A to meet its requirements.

b. Multi-tiered architecture.

A system that adheres to a multi-tiered architecture is broken into several parts: a presentation (user interface) part, an application (and business object) part, and the data services (database) part. These parts can then be distributed across different machines to help achieve scalability. The presentation part typically resides on a user's desktop PC. The application and business object part may be distributed across multiple application servers. The data services part typically resides on a data server.

c. Coupling.

Coupling is an indication of the strength of interconnections between components in a design or system. Coupling is described (or measured) on a relative scale. Two modules are tightly coupled if they share a variable; they are more loosely coupled if one module calls the other module.

d. Formal specification.

A formal specification is a specification that is expressed in a language whose vocabulary, syntax, and semantics are formally defined (i.e., the language has a mathematical foundation).

e. Interface.

An interface describes the set of services a module provides to its clients.

Print Name:__________________________

Student ID#: __________________________

3. ***(Total: 10 points) Characterize the implicit invocation architectural style according to the following features: kinds of components, kinds of connectors, control topology, the binding time of control, the flow of control and data, and the purported benefits of the style.***

2 marks

Components: Arbitrary, could be procedures, objects, processes

2 marks

Connectors: Events (could be implemented by procedure calls, messages over sockets, etc.)

2 marks

Control Topology: Arbitrary

2 marks

Binding Time of Control: Event announcement and registrations may be bound when the programs is written or at run-time.

1 mark

Flow of Control and Data: Logically, data typically flows in the same direction as the event (because it is encoded as part of the event). Sometimes, as in the Observer pattern, there is a direct call back after an event is announced to retrieve the data. I'll take either answer if you explain why it might be opposite in the opposite case.

1 mark

Benefits: Make the system more modifiable by decoupling parts of the system

Other answers were possible and were evaluated on a case-by-case basis.

4. ***(Total: 5 points) Describe the role of the IDL in CORBA.***

1 mark

The IDL (Interface Definition Language) is used to describe an interface to an object.

Stub for client: 2marks

Skeleton for object access: 2 marks

In CORBA, a developer specifies one or more interfaces in the IDL and from that IDL definition generates the stubs and skeletons needed for a client to access an object and for the object to respond to a request.

Print Name:__________________________

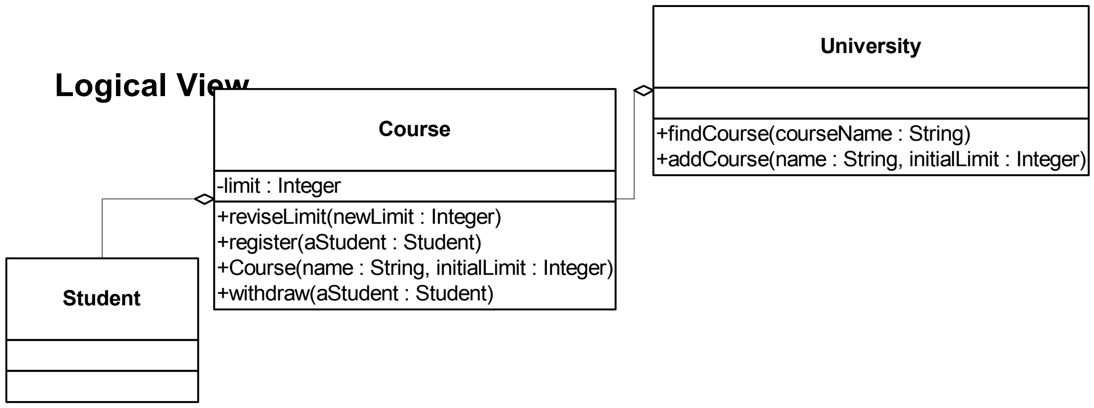
Student ID#: __________________________

5. (Total: 15 points) Consider the following specification:

The Student Registration System shall track the students who have registered for a particular course offered by the University. Each course shall have its own limit on the number of students allowed in the course. A member of the registrar's office shall be able to define the courses offered and their limits. Course limits may be revised upwards after students begin to register. A student shall be allowed to register in a particular course only if the limit for that course has not yet been reached. A student shall be able to withdraw from a course, canceling that student's registration in the course. Each course shall have only one section.

a. (6 points.) Design an architecture for the system that conforms to the object-oriented architectural style. You do not need to fully elaborate the architectural design, but you must describe the components, the connectors, and how the system will basically work. State any assumptions you make.

(You could state your answer either graphically or textually.) Here is a UML class diagram outlining the design for an object-oriented solution to this problem.


Logical View
University
+findCourse(courseName : String)
+addCourse(name : String, initialLimit : Integer)
Course
-limit : Integer
+reviseLimit(newLimit : Integer)
+register(aStudent : Student)
+Course(name : String, initialLimit : Integer)
+withdraw(aStudent : Student)
Student

1 mark	The components in this design are classes.
2 marks	There are three components. The University class provides a means for the member of the registrar to add a new course and to find a course. The Course class tracks the students registered in the course, and provides a means to revise the limit on registrations as well as to allow a student to withdraw from the course. The Student class encapsulates information about a student.
1 mark	The connectors in this design are method calls.
2 marks: use respects the style	A member of the registrar's office will have access to the University object. Through that object, method calls can be made to define the courses offered (using addCourse). To revise the limit on registrations for a course, a member of the registrar's office can use the findCourse method to locate the desired Course object, and can then call the reviseLimit method on the Course object. To register a student in or withdraw a student from a course, the findCourse method on University can again be found to locate the appropriate Course object; the register or withdraw method can then be called.

Print Name:__________________________

Student ID#: __________________________

b. (6 points.) Design an architecture for the system that conforms to the data-repository architectural style. You do not need to fully elaborate the architectural design, but you must describe the components, the connectors, and how the system will basically work. State any assumptions you make.

A graphical view of the architecture is shown below.

1 mark

In this solution, there are two kinds of components: a database component and client components.

2 marks

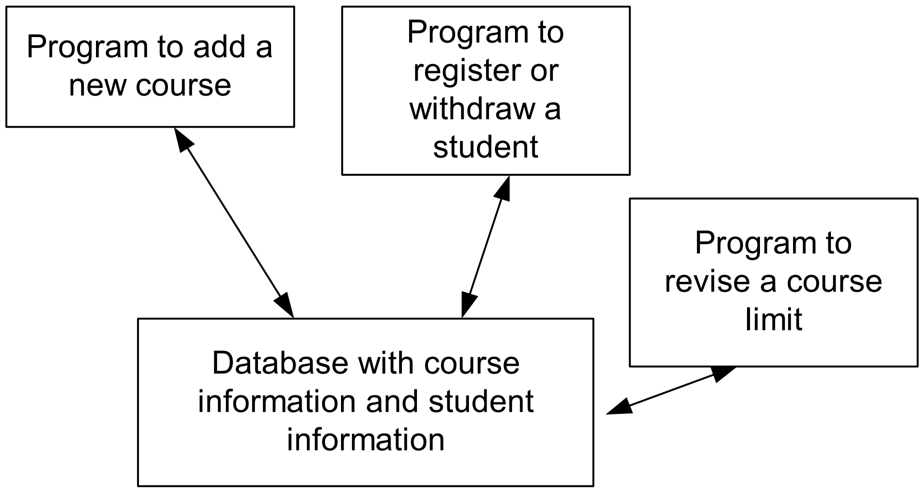
The database component which holds the course and student information (perhaps as tables, perhaps in a hierarchical database, perhaps in an object-oriented database). Second, there are three client program components which perform operations on the database.

1 mark

The connectors are queries on the database.

2 marks

Various clients may operate at once if the database supports synchronous access (and has the write kinds of concurrency controls). To add a course, a member of the registrar's office, uses the client "program to add a new course". To revise the limit of registrations upward, a member of the registrar's office uses the "program to revise a course limit". To register and withdraw from a course, the remaining client program is used.



Program to add a new course
Program to register or withdraw a student
Program to revise a course limit
Database with course information and student information

c. (3 points.) Describe (in one or two sentences) a benefit of your data-repository architectural solution. (In other words, why might you choose to proceed in a development with the data-repository approach?) Is this advantage also present in your object-oriented solution? Why or why not?

1 mark for a benefit; 2 for the argument of why the OO solution does or does not have the same benefit.

One benefit of the data-repository solution is that it is easy to add in new client programs (assuming the database does not need modification to support the client program) that, if the database supports the feature, can operate concurrently with other client programs. This benefit is not present in the straightforward object-oriented solution above which does not mention a shared object space, concurrent access to objects, etc.

(Total: 15 points) Compare and contrast Microsoft's COM and SUN Microsystems JavaBeans integration approaches.

[Here is an abstracted version of an answer to make the marking clearer. Other answers were possible and were evaluated on a case-by-case basis.]

Both define standards to achieve component inter-operability. COM is a binary standard; COM components can thus be used from several languages. [1 mark] A JavaBean is accessible only from Java (unless a bridge is used). [1 mark]

Offering Services:
A client of a JavaBean finds out about the services on the JavaBean primarily through introspection, although a BeanInfo interface may also be used [2 mark]. A client in COM finds out about services on a COM component by asking the component if it supports a desired service (through QueryInterface). [2 mark] In the COM case, the client asks the component explicitly for services it requires. In the JavaBeans case, a client can determine all services provided by the bean. [1 mark]

Find Component:
In COM, the COM layer interacts with the registry to find a component [1 mark]. JavaBeans currently has no mechanism to find a component [1 mark].

Compatibility:
In COM, a client interacts with interfaces (through QueryInterface) to determine if the component is compatible [1 mark]. In JavaBeans, a client may also query a bean dynamically (through either introspection or a BeanInfo interface) to determine if a service is available before invoking the service [1 mark]. In COM, interfaces and components have unique identifiers; these identifiers can be used to help distinguish components and interfaces. [1 marks] In JavaBeans, components can be distinguished if a special stream unique identifier is used; Java interfaces cannot be distinguished (without building in some support). [1 mark]

Interact:
A client in COM interacts with a component through method (procedure) calls. [1 mark] In JavaBeans, events are the primary means of interaction but method calls are also possible. [1 mark]

Print Name:___________________________

Student ID#: ___________________________

6. ***(Total: 20 points) Write a Z specification for the system described by the following natural language description. You need only specify the correct operation of the system (i.e., you can ignore error handling). The last page of this examination booklet provides some Z syntax you may find helpful (but not necessarily all you may wish to use). State any assumptions you make***

   A system is needed to help manage the lending of books from a library. The system shall allow a librarian to enter books into the system. Once entered, the status of a book shall be tracked. The system shall track when a borrower (user of the library) checks out a book from the library. The system shall also track when a borrower returns a book to the library. Borrowers may borrow books indefinitely. The system shall also allow a borrower to reserve a book that another borrower currently has on loan. Each book that is currently on loan may be reserved by at most one borrower. A book that has been reserved may only be borrowed by the library user who reserved the book.

A "real" Z specification would include natural language text describing the Z. Given the time constraints on the exam, I did not expect any of this descriptive text. Given that the question only asked for the correct operation of the system to be specified, we won't do any validity checking on courses and students.

We use a static Z schema to describe the state space of the system. First, we introduce some basic types, BOOK and USER, which we don't need to describe in detail. The static schema defines an invariant. The invariant states that a book that is placed on reserve has to be a book known to the library. Similarly, a book that is on loan must be a book known to the library.

[BOOK, USER]

STATUS ::= in | out

3 marks

```
 Library ──────────────────────────────────
books: BOOK -> STATUS  (this is a partial function)
booksOnLoan: BOOK -> USER (this is a partial function)
reservedBooks: BOOK -> USER (this is a partial function)
───────────────────────────────────────────
dom reservedBooks ⊆ dom books
dom booksOnLoan ⊆ dom books
───────────────────────────────────────────
```

2 marks

We then need to state the initial state of the system

```
 InitLibrary ──────────────────────────────
Library
───────────────────────────────────────────
dom books = ∅
───────────────────────────────────────────
```

Print Name:__________________________

Student ID#: __________________________

There needs to be a way to add books to the library.

3 marks

```
AddBook
Δ Library
b?: BOOK
----------------------------------------
b? ∉ dom books
books’ = books ∪{ b? -> in } (the arrow should be a maplet arrow)
reservedBooks’ = reservedBooks
booksOnLoan’ = booksOnLoan
```

There needs to be a way of reserving a book.

5 marks

```
ReserveBook
Δ Library
b?: BOOK
u?: USER
----------------------------------------
b? ∈ dom books
b? ∉ dom reservedBooks
books( b?) = out
booksOnLoan(b?) ≠ u?
reservedBooks’ = reservedBooks ∪ { b? -> u?}  (the arrow should be a maplet arrow)
books’ = books
booksOnLoan’ = booksOnLoan
```

There needs to be a way to check out a book. (Note that the arrows below in the set parentheses should be maplet arrows.)

5 marks

```
LoanBook
Δ Library
b?: BOOK
u?: USER
----------------------------------------
b? ∈ books
books ( b? ) = in
( ( ( b? ∈ dom reservedBooks ) ∧ ( reservedBooks( b? ) = u? ) ∧
    ( books’ = books ⊕{ b? -> out }  ) ∧
    ( booksOnLoan’ = booksOnLoan ∪ { b? -> u? } ∧
    ( reservedBooks’ = reservedBooks \ { b? -> u? } )  )
 ∨ ( ( b? ∉ dom reservedBooks ) ∧ ( books’ = books ⊕{ b? -> out }  )  ∧
     ( booksOnLoan’ = booksOnLoan  ∪ { b? -> u? } ) ∧
     ( reservedBooks’ = reservedBooks ) ) )
```

Print Name:___________________________

Student ID#: ___________________________

2 marks

```
ReturnBook
Δ Library
b?: BOOK
----------------------------------------
b? ∈ books
books ( b? ) = out
books’ = books ⊕{ b? -> in }
booksOnLoan’ = booksOnLoan \ { b? -> booksOnLoan(b?) }
reservedBooks’ = reservedBooks
```

Print Name:__________________________

Student ID#: __________________________

Some Z Notation

$\varnothing$	Empty Set
$\neq$	Inequality of Terms
/	Set Difference
$\notin$	Non-membership
P	Power set
#	Cardinality (of a set or sequence)
Z	Integers
N	Natural Numbers
$\neg$	Negation
$\wedge$	Conjunction
$\vee$	Disjunction
$\Rightarrow$	Implication
$\forall$	Universal Quantifier
$\exists$	Existential Quantifier
$\rightarrow$	Maplet in an ordered pair
$\leftrightarrow$	Relation
dom	Domain
ran	Range
$\oplus$	Addition (Overriding)
	Relational Image

Print Name:__________________________

Student ID#: __________________________

$\rightarrow$	Function
$\rightarrow$	Total Function
$\rightarrow$	Injection
$\rightarrow$	Total Injection
$=$	Schema Definition
$\vee$	Schema Disjunction
$\wedge$	Schema Conjunction