University of Toronto

Faculty of Arts and Science
Department of Computer Science

CSC340S - Information Systems Analysis and Design

Spring 2000   **John Mylopoulos**

April-May Examination

No Aids Allowed
Duration: 2 hours

Make sure that your examination booklet has 12 pages
(including this one). Write your answers in the space provided.

This examination counts for 35% of your final mark.

Name: _____
(Please underline your last name)

Student Number: _____

Question Marks

1. _____ /10

2. _____ /10
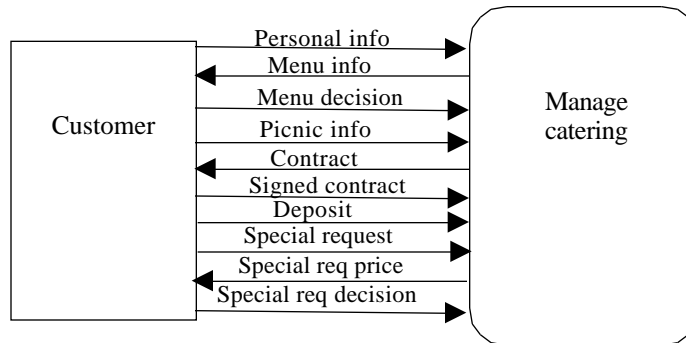
3. _____ /10

4. _____ /10

5. _____ /10
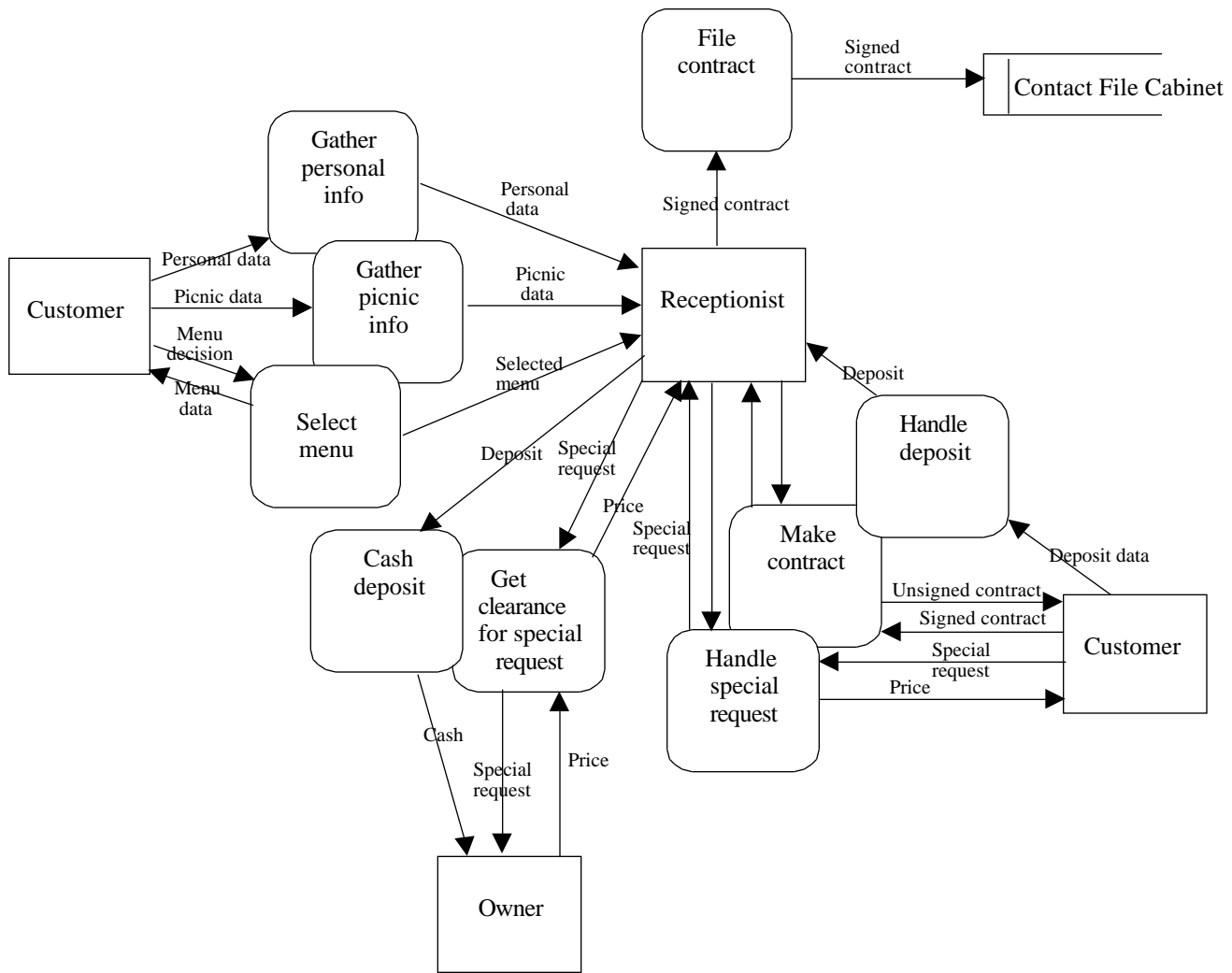
6. _____ /20

Total _____ /70

1. [Data Flow Diagrams; 10 marks]

A systems analyst has collected the following information about a project she is working on for a small catering business. To assist her, draw *physical context and level 0* data flow diagrams .

Picnics R Us (PRU) is a small catering firm with a staff of five employees. During a typical summer weekend, PRU caters 15-20 picnics with 20 to 50 people each. When potential customers call, the receptionist describes the list of 10 standard menus offered by PRU and gives price information. If the customer decides to book a picnic, the receptionist records the customer's name, address, phone number and the information about the picnic (time, place, number of people, selected menu, total price) on a contract. The customer is then faxed a copy which the customer signs and sends back, along with a deposit. The deposit can be in the form of a cheque or a credit card charge. Once the owner cashes the deposit, the picnic is considered officially booked. For each booking, the contract goes in a file cabinet which includes all contracts sorted by week. Sometimes the customer wants something special, such as a birthday cake. In this case, the receptionist takes the information, gives it to the owner who determines a price. The receptionist then calls the customer and gives the price, which is accepted or rejected by the customer.

```
  ┌───────────┐   Personal info  ─────►  ╭──────────╮
  │           │   Menu info      ◄─────  │          │
  │           │   Menu decision  ─────►  │          │
  │           │   Picnic info    ─────►  │  Manage  │
  │ Customer  │   Contract       ◄─────  │ catering │
  │           │   Signed contract ────►  │          │
  │           │   Deposit        ─────►  │          │
  │           │   Special request ────►  │          │
  │           │   Special req price ◄──  │          │
  └───────────┘   Special req decision ► ╰──────────╯
```

1. [Data Flow Diagrams; 10 marks -- continued]

File
contract

Signed
contract

Contact File Cabinet

Gather
personal
info

Personal
data

Signed contract

Customer

Personal data

Gather
picnic
info

Picnic
data

Receptionist

Picnic data

Menu
decision

Menu
data

Select
menu

Selected
menu

Deposit

Deposit

Special
request

Handle
deposit

Price

Special
request

Make
contract

Deposit data

Cash
deposit

Get
clearance
for special
request

Unsigned contract

Signed contract

Customer

Handle
special
request

Special
request

Price

Cash

Special
request

Price

Owner

2. [Entity Relationship Diagrams; 10 marks]

Draw an Entity-Relationship diagram that describes the contents of the PRU file cabinet.

Your diagram should show clearly attributes and keys for all entity and relationship types, also cardinalities for relationships and attributes.

```
                  _____
                                            Contract# (1::1)
  Name (1::1)                               Date (1::1)                          Date (1::1)
  Address (1::1)  ┌──────────┐              ◆              1::1     ┌──────────┐  Time (1::1)
  Phone# (1::1)   │ Customer │──1::*──────  Books  ──────────────  │  Picnic  │  Location (1::1)
                  └──────────┘                                     └──────────┘  Total Price (1::1)
                                                                        │
                                                              0::*  ╱   ╲  1::*
                                            1::1                  ╱       ╲
                  Deposit# (1::1)  ┌──────────┐                 ╱           ╲
                  Amount (1::1)    │          │               ◆              ◆
                  Method (1::1)    │ Deposit  │            Includes        Includes
                  Credit card# (1::1) └────────┘
                                                         1::1                 0::*
                  Name (1::1)      ┌──────────┐                   ┌──────────┐  Name (1::1)
                  Description (1::1)│ Special │                   │   Menu   │  Description (1::1)
                  Price (1::1)     │ request  │                   │          │  Price (1::1)
                                   └──────────┘                   └──────────┘
```
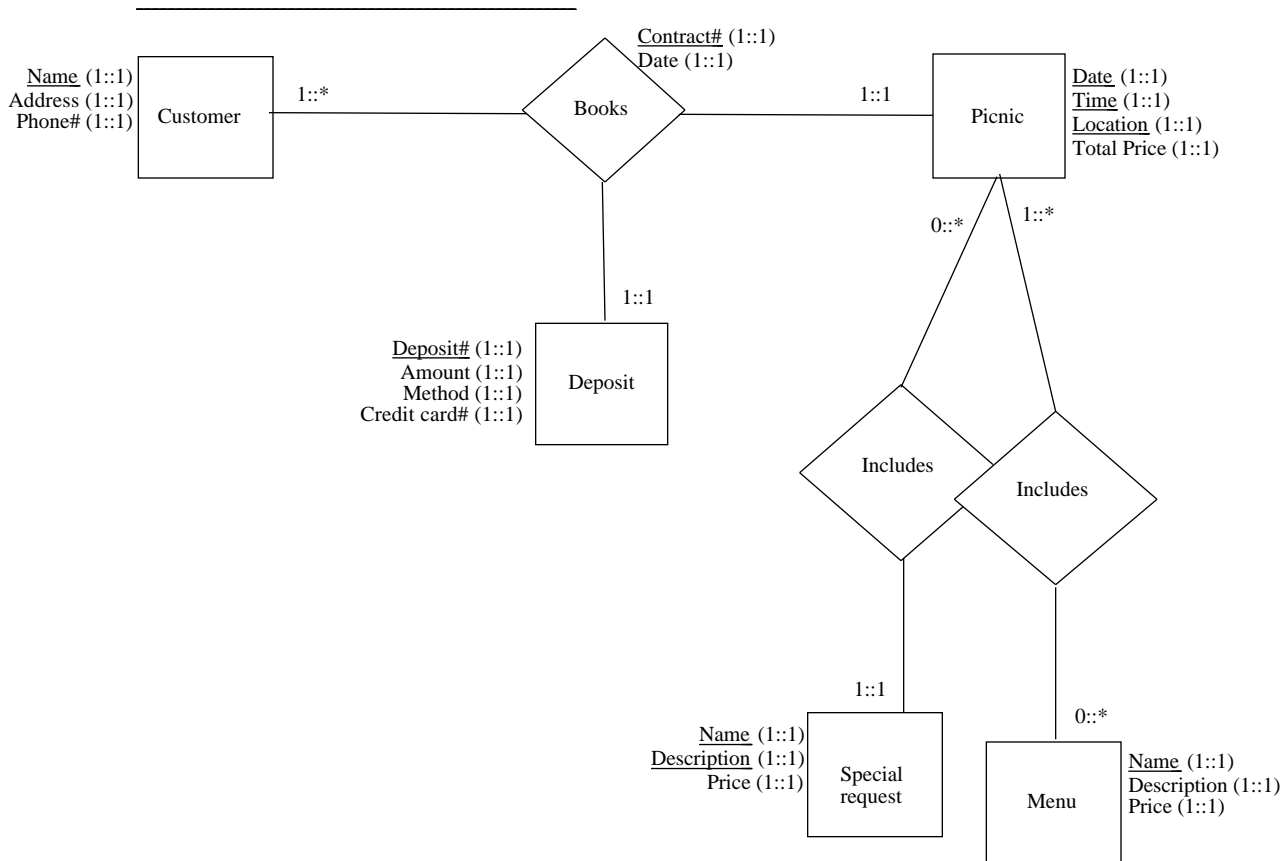
3. [User Interface Design; 10 marks]

eStocks.com has decided to offer a new stock-over-the-phone service to its customers whereby they can buy and sell stocks electronically over a phone. Their idea is to allow a customer to call a number, ESTOCKS or 378-6257, and after the customer has dialed in her account number (e.g., 123456#) and password (e.g., 4567#), she is given the choice (through automatically generated voice messages) of doing one of three things:
- Get the price of a stock for which the customer types in the name (e.g., 426# for IBM);
- Buy stock for a company; here the customer has to give the company's name, and the number of stocks; the system will tell the customer what is the total cost of the transaction and the customer confirms the transaction;
- Sell a number of stocks for a given company; again, the customer has to specify the company name and the number of stocks to be sold, and the service will give the customer full information about the transaction before asking the customer to confirm the transaction.

Of course, the customer is given the option of doing several transactions during a single session. All this is supposed to be done without human intervention on the part of eStocks.com, unless the customer wants at any point of the interaction to talk to a human operator.

Design the dialogue structure for this telephone service in terms of a finite state machine (FSM). Show clearly the inputs and outputs for each transition of your FSM.
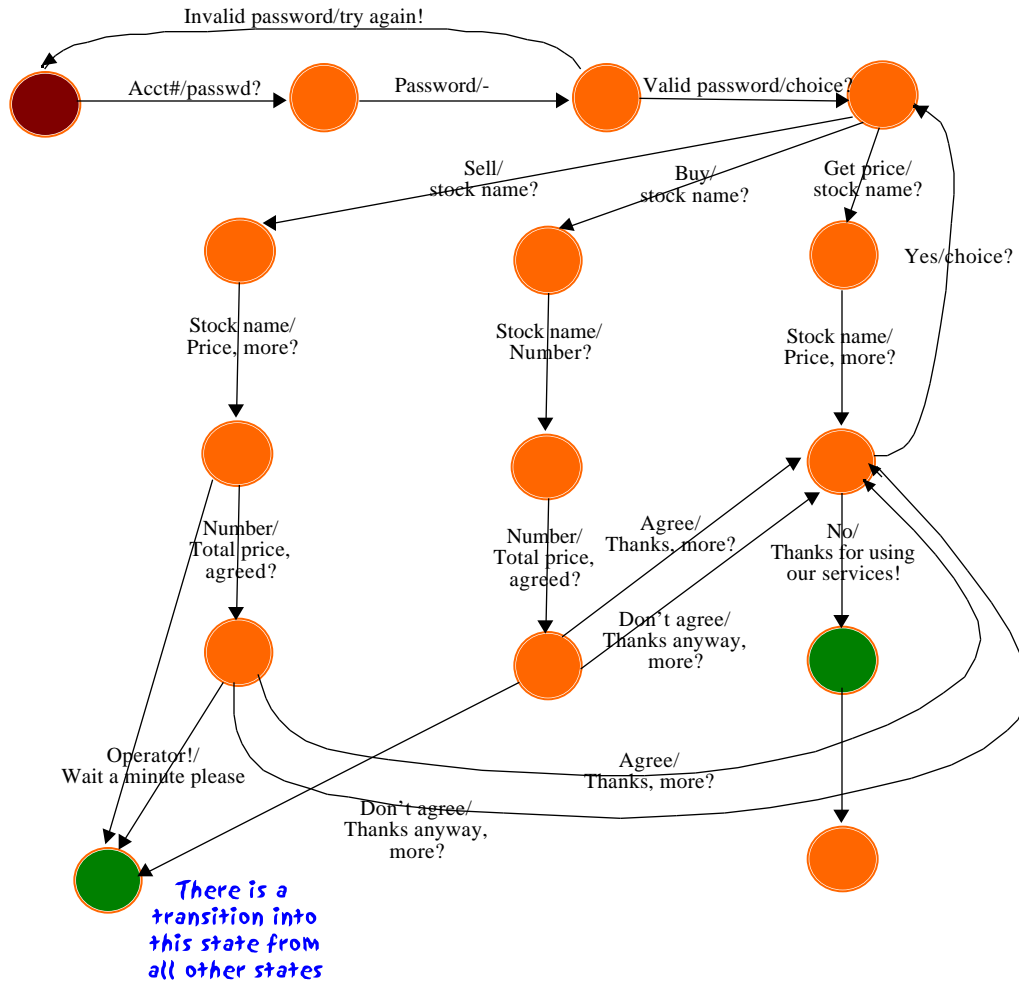
4. [Program Design -- Structure Charts; 10 marks]

Give a structure chart for the stocks-over-the-phone service for eStocks.com. The structure chart should show modules for the different functions supported by the service (get price, buy, sell), with sub-modules for getting the customer's input, fetching data from the stock database, generating voice messages with the information requested by the customer, updating the stock database, etc. Your structure chart should be 3 levels deep and should use special connections to indicate choice, iteration etc.
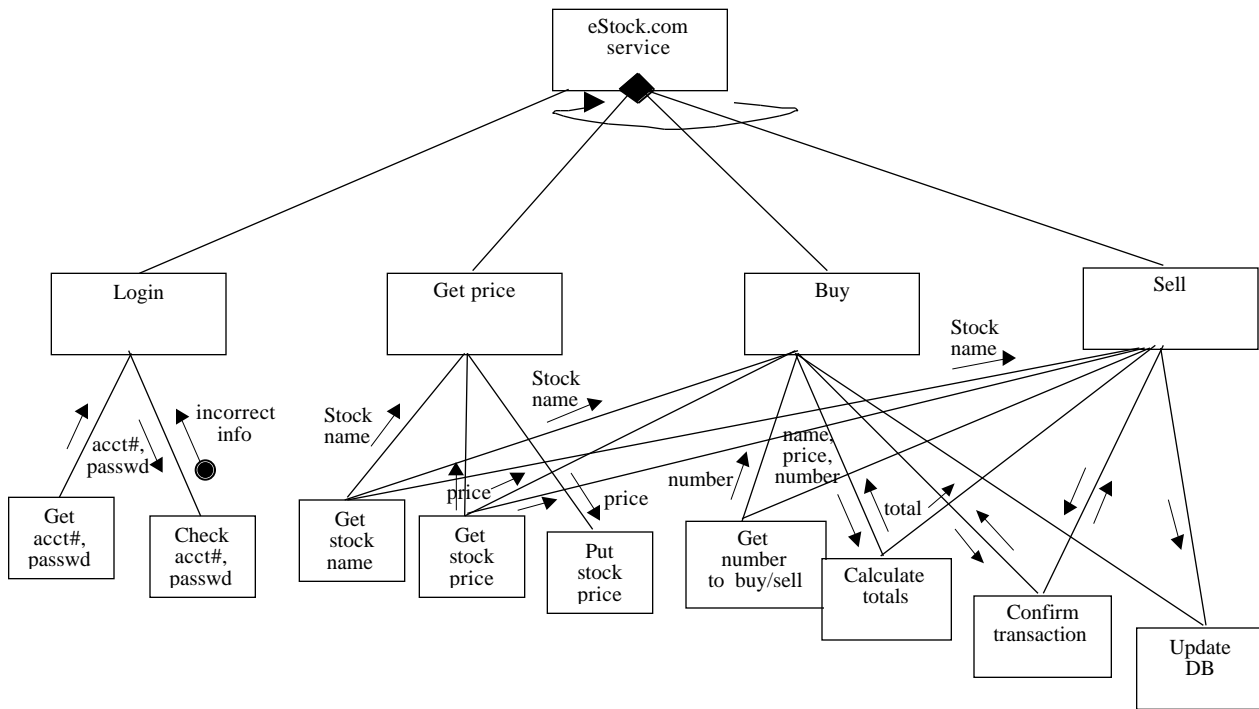
4. [Program Design; 10 marks -- continued]

5. [Database Design; 10 marks]

Here is the relational database schema 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,amount,taxRate)
    239   23/06/99   1135   Black Conrad      Ont   $138.56      0.08
    260   12/09/99   1135   Black Conrad      Ont   $82.03       0.08
    297   30/04/00   1577   Wong  Harry       Que   $75.21       0.07
```

```
ProductOrdered(order#,product#,productDesc,quantity,price)
              239    555   Nut        35      $1.02
              239    444   Bolt       35      $2.17
              297    444   Bolt       25      $2.17
```

Unfortunately, this database is not in 2NF because of the following functional dependencies:
```
        customerID → lastName,firstName,province,taxRate
        province → taxRate
        product# → productDesc,price
```

Place this schema first in 2NF, and then in 3NF.

2NF

```
Order(order#,date,customerID,lastName,firstName,province,amount,taxRate)
ProductOrdered(order#,product#,quantity)
Product(product#,productDesc,price)
```

3NF

```
Order(order#,date,customerID,amount)
Customer(customerID,lastName,firstName,province)
Taxrate(province,taxrate)
ProductOrdered(order#,product#,quantity)
Product(product#,productDesc,price)
```

6. [Short Questions; 5 marks each, 20 marks total]

[Decision tables; 5 marks] Give a decision table for cheque cashing for a "Cheques Cashed Here" store which has the following rules: "A customer can cash personal cheques for up to $75.00 and payroll cheques for any amount, provided they are issued by companies that have been pre-approved by the store"

Make sure your decision table is minimal in the number of rows and columns it uses.

```
 $75            Y      N      *      *
Cheque type     Per    Per    Pay    Pay
Pre-approved    *      *      Y      N
Cash            X             X
No cash                X             X
```

[Implementation Phase; 5 marks] What is the difference between *stub testing*, *unit testing* and *system testing*?

```
Stub testing involves testing of a single module.
Unit testing involves testing a collection of modules.
System testing involves testing the whole software system.
```

[Software Architectures; 5 marks] Consider a software system which tabulates election results as follows:

- The system consists of three modules;
- The first module takes in election results per polling station, as these results become available, and updates the current totals for each candidate, feeding the information to the second module;
- The second module calculates statistics of various kinds and feeds the results to the third module;
- The third module displays the results in a variety of formats.

What kind of architecture is this, among those discussed in class (pipes-and-filters, object-oriented, event-oriented, repository-based, layered, other.) Explain your answer carefully! The explanation counts for 3 out of 5 marks.

```
Pipes-and-filters. Data are processed by each filter as they
become available, and the outputs are added to the queues of
other filters.
```

(d) [Database Design; 5 marks] "Unlike database systems, file systems don't support data sharing." Explain what does this mean.

```
Data sharing is about having several applications operate on
the same database with no fear of interference among them,
e.g., having 1,000 transactions execute concurrently on a
single (bank) customer account database. File systems don't
support this feature.
```

(Scratch paper)

(Scratch paper)

(Scratch paper)