SOEN 342 Software Requirements Specifications

Fall 2011

Midterm Exam #2 – Example Questions

Name: ______________________ Total Points:

ID: ______________________ ________ / ____

Instructions. *This example SOEN 342 Midterm #2 contains questions to help you testing your preparedness. Note that the midterm is a* ***closed book*** *exam. The real exam will contain about 4–5 questions of similar difficulty that you will need to solve in about 10–15mins each. Also, note that the actual midterm will not cover the same questions as the ones here (or even the same type of questions)!*

(7 pts) **1.** Consider the following domain description for email clients:

7 pts

> The client has one mailbox which consists of a number of different folders. Each folder contains a number of messages. A message cannot exist in more than one folder and cannot exist outside a folder. A user can invoke a view on a message and in fact a user may have multiple views, each corresponding to a single message.

(a) (1 pt) Name an appropriate method for identifying *conceptual classes* in this domain description: ____________________

(b) (1 pt) Use the method to create a list of domain concepts based on the provided description:

- ____________________
- ____________________
- ____________________
- ____________________
- ____________________
- ____________________

(c) (4 pts) Create a domain model for the email client as a UML class diagram. Make sure you show all appropriate details, including associations, multiplicities, and aggregations.

(d) (1 pt) Illustrate the difference between *aggregation* and *composition* using your domain model. Give a brief explanation for each:

Aggregation: ____________________

Composition: ____________________

(7pts) **2.** You elicited the following requirements for a library loan system:

7 pts

1. A book can be on stack if and only if it is not on reserve or on loan
2. A book can be on reserve if and only if it is not on stack or on loan
3. A book can be on loan if and only if it is not on reserve or on stack
4. A book can be requested if and only if it is on stack or on reserve

(a) (2 pts) Translate these requirements into propositional logic:

1. ____

2. ____

3. ____

4. ____

(b) (1 pt) Consider the two requirements 1. and 3. together. Are they *consistent*? Prove or disprove (*Hint: you do not need to create a complete truth table*):

⇒ ***Continued on next page!***

(c) (4 pts) Using a *proof by resolution*, show that the statement

If a book is on loan then it can not be requested

logically follows from the requirements:

(6pts) **3.** Consider the following specification for a course planning system:

6 pts

> The course section life cycle starts from its planning. Once the decision for opening the registration for the course is received, the course is opened. While the course is opened, the requests for registering can be accepted. The course will not be actually taught until the class size reaches a certain minimum. The requests to register are accepted until the course reaches the predefined maximum number of students, or the registration deadline has passed; in both cases, the course section becomes closed. If the class size is below the minimum, the class is cancelled. Closing the section when there are not enough students will have the same effect as cancelling it.

(a) (4 pts) Draw an UML state machine diagram to specify Course Section behaviour. Use hierarchical states where appropriate:

(b) (2 pts) *True* or *False*? ***Note:*** Each correct answer gives you $\frac{1}{2}$ points, each wrong answer $-\frac{1}{2}$, "don't know" 0 (however, you will never get a negative score for this question).

State machines can be used to describe legal system events within a use case.

☐ True ☐ False ☐ Don't know

Activity diagrams are useful to model use cases with many alternative flows or extensions.

☐ True ☐ False ☐ Don't know

State machines are an alternative modeling technique to domain models in RE.

☐ True ☐ False ☐ Don't know

State diagrams can show how a *single* object behaves across *different* use cases.

☐ True ☐ False ☐ Don't know

(6 pts) **4.** Consider the following Z schema specification for a birthday book application:

6 pts

$[NAME,\ DATE]$

BirthdayBook

$known : \mathbb{P}\, NAME$
$birthday : NAME \nrightarrow DATE$

$known = \mathrm{dom}\ birthday$

(a) (3 pts) Write a non-robust Z schema for the *UpdateBirthday* operation, which changes the date of an *existing* entry (i.e., if a name is not in the system, it will not be added by the UpdateBirthday operation).

UpdateBirthday

⇒ ***Continued on next page!***

(b) (1 pt) Now make the operation robust by using the following two schemas for error handling:

$$REPORT ::= ok \mid already\_known \mid not\_known$$

Success

$result! : REPORT$

$result! = ok$

NotKnown

$\Xi BirthdayBook$
$name? : NAME$
$result! : REPORT$

$name? \notin known$
$result! = not\_known$

Define a robust version of *UpdateBirthday* that returns *ok* in case of success and *not_known* in case of an error:

$RUpdateBirthday =$ ______________________________________

(c) (2 pts) Now show the combined schemas for the *RUpdateBirthday* operation:

UpdateBirthday

Note: This sheet will also be provided in the actual exam

Truth tables for ¬, ∧, and ∨

p	$\neg p$
T	F
F	T

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

Truth tables for ↔ and →

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Equivalence Rules

Equivalence Rule	**Name**
$p \Leftrightarrow \neg\neg p$	double negation
$p \rightarrow q \Leftrightarrow \neg p \vee q$	implication
$\neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$ $\neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q$	De Morgan's laws
$p \vee q \Leftrightarrow q \vee p$ $p \wedge q \Leftrightarrow q \wedge p$	commutativity
$p \wedge (q \wedge r) \Leftrightarrow (p \wedge q) \wedge r$ $p \vee (q \vee r) \Leftrightarrow (p \vee q) \vee r$	associativity

Inference Rules

Inference Rule	**Name**
$\left.\begin{array}{r} p \\ q \end{array}\right\} \Rightarrow p \wedge q$	conjunction
$\left.\begin{array}{r} p \\ p \rightarrow q \end{array}\right\} \Rightarrow q$	*modus ponens*
$\left.\begin{array}{r} \neg q \\ p \rightarrow q \end{array}\right\} \Rightarrow \neg p$	*modus tollens*
$\left.\begin{array}{r} p \rightarrow q \\ q \rightarrow r \end{array}\right\} \Rightarrow p \rightarrow r$	chaining
$\left.\begin{array}{r} p \vee q \\ \neg p \vee r \end{array}\right\} \Rightarrow q \vee r$	resolution
$p \wedge q \Rightarrow p$	simplification
$p \Rightarrow p \vee q$	addition