COMP 3111
SOFTWARE ENGINEERING

LECTURE 10
SYSTEM REQUIREMENTS CAPTURE
REQUIREMENTS EXERCISE

QUESTION 1

When developing software, it might be tempting to include features because they would be ingenious and/or possibly useful to someone now or in the future.

Do you think this is good idea? Why or why not?

Ans. This is not a good idea for the following reasons.

1. All features should be discussed with, approved by and prioritized by the client. You may well be implementing a feature the client does not want or care about.
2. Including such features will most likely lead to project scope creep causing the project to be either over time, over budget, or both.

QUESTION 2

What, if anything, is wrong with the following use case?

Use-case:	Withdraw Cash from ATM
Actor:	Bank Customer
Preconditions:	Bank customer has a checking or savings account with the bank
Postconditions:	Bank customer has cash or the reason the withdraw request failed
Basic Flow:	1. The bank customer (BC) inserts his/her bank card into the ATM machine and enters his/her PIN number. 2. The BC requests a withdraw transaction from checking and enters the amount. 3. The BC indicates he/she is finished and takes the cash and bank card from the machine.

Ans. The basic course of events makes clear what the bank customer does, but not what the ATM does. This is a sign the requirements analyst has not thought through or is not sure what the system should do in detail. A use case documents the request/response exchange between the actor and proposed system. It should be just as specific about what system does as what the the actor does.

QUESTION 2

After thinking through what the system should do, the resulting use case might look something like:

Use-case:	Withdraw Cash from ATM
Actor:	Bank Customer
Preconditions:	Bank customer has a checking or savings account with the bank
Postconditions:	Bank customer has cash or the reason the withdraw request failed
Basic Flow:	1. The bank customer (BC) inserts his/her bank card into the ATM machine. 2. The ATM prompts for the BC's preferred language. Options are English, Spanish or French. 3. The BC selects a language. 4. The machine responds with a request for the BC's PIN. 5. The bank customer enters his/her PIN and presses #. 6. The ATM prompts for the type of transaction. Options are deposit, withdraw and balance inquiry. 7. The BC requests a withdraw transaction. 8. The ATM asks whether the withdraw will be from checking or savings. 9. The BC selects checking.

QUESTION 2

	10. The ATM prompts for amounts in units divisible by 10. 11. The BC enters the amount and presses enter. 12. The ATM dispenses the cash and asks the customer if he/she would like another transaction. 13. The BC selects "no". 14. The ATM ejects the BC's bank card and reminds him/her to take his/her cash and bank card. 15. The ATM beeps until the card is removed. 16. The BC takes the cash and bank card from the machine. 17. The ATM stops beeping as soon as the card is removed.

Notice how many more details emerge when you have to be explicit about the system behaviour on the system-side of the interaction.

QUESTION 3

Is the following a functional or non-functional requirement?

The system shall be responsive. No query should last longer than 3 seconds without some type of feedback or progress indication sent back to the user.

Ans. It is both.

This illustrates the challenge of completely separating the two.

The non-functional requirement is performance. It expresses the desire for the system to be fast or at least not slow.

This non-functional requirement leads to the functional requirement that progress or feedback be issued for all transactions that take longer than 3 seconds.

QUESTION 4

There can be defects in requirements the same way there can be defects in code. What is the defect in the following requirements?

The system will inform the student whether he or she passed an assessment. The system shall take two inputs: score and possible points. The system will return true if score / (possible points) is >= 50% else it returns false. The system shall report an error if either input is negative or if score is > (possible points).

QUESTION 4

Ans. The requirements do not say what to do when both the score and the possible points equal 0. This is an example of incomplete requirements.

There are three likely scenarios.

1. The programmer could just make an assumption "Hmm? Seems reasonable to report an error when both inputs are zero."
2. The programmer might not even recognize the requirements are incomplete. Depending on the implementation, this might result in an unintentional divide by zero error at runtime (hopefully something detected before release).
3. Ideally, during requirements analysis someone recognizes the requirements are incomplete and requests clarification from the client.