Midterm Review

CSCE 740 - Lecture 12 - 10/12/2015

General Questions

- Today: Go over practice midterm questions.
- First - any general questions on course content or homework?

Question 1

Briefly explain why a software system must change or become progressively less useful?

Question 1 - Solution

- The world is constantly changing, and if software does not change too, it will be out of date and useless.
- Changes might be
  - organizational (user's needs have changed).
  - infrastructure (hardware and OS are changing).
  - changed computational model (standalone systems are now networked)
  - … etc...

Question 2

The properties of the environment of a system are generally of critical importance for the system to be able to satisfy its stated system requirements. It is essential to capture environmental assumptions in a requirements document.

Briefly discuss how the environment may influence a system's ability to satisfy its requirements.

Question 2 - Solution

- A system cannot meet its requirements without assuming properties of the environment.
  - Patient-Monitoring System: Is the nurse close enough?
- Environment must cooperate with the system.
- Requirements must capture what we assert to be true, so that informed decisions can be made.
  - And we can argue that we wrote a correct specification and can meet the real-world requirements.

Question 3

Explain the following tenets of XP (and other agile processes):

- Collective Ownership
- Sustainable Pace
- Open Workspaces
- Customer as a Team Member
- Test-First Development
- Short Iterative Cycles
- Stories as Requirements

Question 3 - Solution

- Collective Ownership
  - All developers own the code, and can make changes. They do not need approval.
- Strict Use of Coding Standards
  - Allows code to be readable by other developers.
- Open Workspaces
  - Enables quick communication and informal meetings.
- Customer as a Team Member
  - Rapid feedback is essential.
- Test-First Development
  - Refines requirements, clarifies implementation.

Question 3 - Solution

- Short Iterative Cycles
  - We are able to quickly get feedback on the current status of a project.
- Sustainable Pace
  - Developers produce worse results if forced to work too much overtime. Keep iteration scope in check.
- Stories as Requirements
  - We can use the expected usage of the system as the basis for development. Lightweight requirements document.

Question 4

You are involved in the development of a new software product. The product is an insurance application intended to determine what insurance products a potential customer is eligible for. The eligibility requirements are captured in various laws and regulations.

An external contractor is doing most of the work. Your organization has hired this contractor to assist with all aspects of planning, management, and development since your organization is lacking the expertise to complete the project. The plan was to do a waterfall process. The product will be long lived and good documentation is a must. The laws were thought to constitute a good start for the requirements.

During the requirements capture process, the team discovers the laws are incomplete, ambiguous, and obtuse. The elicitation is taking longer than planned, and required a lot of interaction with case workers. Towards the end of the requirement process, the requirements document is incomplete and there is more work to be done.

At this point, the contractor decides that since there is so much requirements risk and the schedule is behind, the project will switch to an agile method (XP). In addition, to save money, the contractor is offshoring the coding to a development center in a low-cost country.

1: Is this approach likely to succeed?

2: What would your recommendation have been?

Question 4 - Solution

Likely to succeed:

- NO!!!
- One reason: Agile requires access to customers. Off-shore development will make this hard. Lack of familiarity with US laws and regulations is a problem.
- Project now involves three organizations, unlikely to produce a good design even with traditional plan-driven methods. Existing requirements are poor, not written in a style conducive to incremental design.

Recommendation:

- Cancel the project?
- Can we build incrementally or iteratively from the existing requirements? With good design, new or changed rules can be integrated into system.

Question 5

- Explain the difference between *verification* and *validation*.
- Which of these is considered harder? Why?

Question 5 - Solution

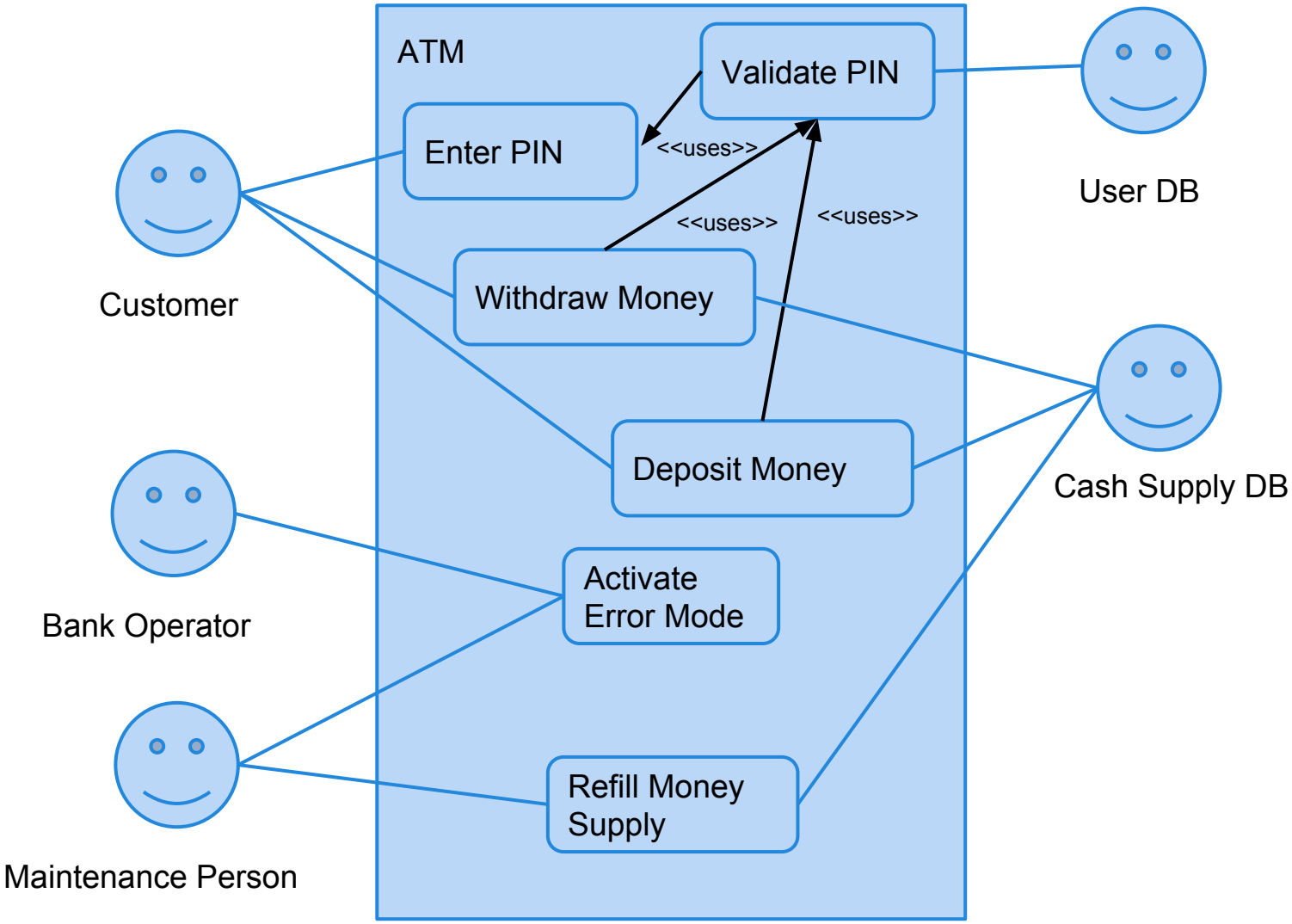
- Explain the difference between *verification* and *validation*.
  - Validation: Does the system meet the customer's needs? "Are we building the right product?"
  - Verification: Does the system meet the specifications we laid out? "Are we building the product right?"
- Which of these is considered harder? Why?
  - Validation is harder.
  - It requires that we understand the customer's actual desires. They might not have told us those, or changed their minds.

Questions 6 & 7

ATM

- What are the actors and use-cases involved in an ATM system?
- Draw a use-case diagram
- Pick one use case and write a scenario.

Questions 6 & 7 - Solution



Question 8

The airport connection check is part of a travel reservation system. It is intended to check the validity of a single connection between two flights in an itinerary.

validConnection(Flight arrivingFlight, Flight departingFlight) returns ValidityCode.

A Flight is a data structure consisting of:

- A unique identifying flight code (string, three characters followed by four numbers).
- The originating airport code (three character string).
- The scheduled departure time (in universal time).
- The destination airport code (three character string).
- The scheduled arrival time (in universal time).

There is also a flight database, where each record contains:

- Three-letter airport code (three character string).
- Airport country (two character string).
- Minimum connection time (integer, minimum number of minutes that must be allowed for flight connections).

Question 8

A Flight is a data structure consisting of:

- A unique identifying flight code (string, three characters followed by four numbers).
- The originating airport code (three character string).
- The scheduled departure time (in universal time).
- The destination airport code (three character string).
- The scheduled arrival time (in universal time).

There is also a flight database, where each record contains:

- Three-letter airport code (three character string).
- Airport country (two character string).
- Minimum connection time (integer, minimum number of minutes that must be allowed for flight connections).

Derive representative values of the parameters from this specification for the validConnection function.

Question 8 - Solution

Parameter: Arriving flight

Flight code:
- *malformed*
- *not in database*
- *valid*

Originating airport code:
- *malformed*
- *not in database*
- *valid city*

Scheduled departure time:
- *syntactically malformed*
- *out of legal range*
- *legal*

Destination airport (transfer airport):
- *malformed*
- *not in database*
- *valid city*

Scheduled arrival time (tA):
- *syntactically malformed*
- *out of legal range*
- *legal*

Parameter: Departing flight

Flight code:
- *malformed*
- *not in database*
- *valid*

Originating airport code:
- *malformed*
- *not in database*
- *differs from transfer airport*
- *same as transfer airport*

Scheduled departure time:
- *syntactically malformed*
- *out of legal range*
- *before arriving flight time (tA)*
- *between tA and tA + minimum connection time (CT)*
- *equal to tA + CT*
- *greater than tA + CT*

Destination airport code:
- *malformed*
- *not in database*
- *valid city*

Scheduled arrival time:
- *syntactically malformed*
- *out of legal range*
- *legal*

Parameter: Database record

This parameter refers to the database time record corresponding to the transfer airport.

Airport code:
- *malformed*
- *not found in database*
- *valid*

Airport country:
- *malformed*
- *invalid*
- *valid*

Minimum connection time:
- *not found in database*
- *invalid*
- *valid*

Question 9

The following requirements are unclear and ambiguous. Explain why, and then rewrite the statements so that they can be objectively evaluated.

a. The response time should be minimized.

b. The alarm should be raised quickly after a high fuel level has been detected.

Question 9 - Solution

a. The response time should be minimized.

“should” != shall

What does minimized mean?

Response time to what?

“The system shall respond to a user request within ten seconds.”

Question 9 - Solution

b. The alarm should be raised quickly after a high fuel level has been detected.

Quickly?

Is “high fuel level” a boolean condition or a specific quantity?

“The alarm shall be raised within 5 seconds of the fuel level reaching 10 cm.”

Question 10

In class, we discussed the importance of defining a test case for each requirement. What are the two primary benefits of defining this test case?

Question 10 - Solution

1. A test case will greatly help us in the integration testing phase. Groups can start defining tests early and be ready when the system comes online.
2. Test cases force us to write testable (thus, good) requirements. If a requirement is not testable, we cannot write a test case.

Question 11

You are setting out to develop a new GUI for an old application. The system has a diverse set of users and the system has to be acceptable to all of the user types.

What development process would you use? Justify your answer.

Question 11 - Solution

Any process that makes use of evolutionary prototyping. Build something rapidly and get use feedback, then build something new that incorporates that feedback.

Question 12

Briefly discuss the concept of incrementality (from now on, this is a real word) as it applies to software development.

Question 12 - Solution

Incrementality is the principle of breaking the software project (or anything else) into smaller manageable pieces that can be used by the customer while other pieces are still in development.

As new pieces are completed, they are integrated until we have a complete system.

Question 13

The main function of a vending machine is to allow the customer to buy products from the machine (soda, candy, etc). When the customer wants to buy some of the products, they insert money, select one or more products, and the machine dispenses the product to the customer. Should the product cost less than the amount of money inserted, the machine will dispense change. The machine must be restocked when it runs out of products. A collector comes and collects money from the vending machine.

1: Identify the actors and use cases and draw a use case diagram.

2: Define the basic course of events for one of the use cases.

3: What are the exception or alternate paths for that scenario?

Question 14

There is a difference between the internal and external completeness of a requirements document.

1: Describe internal and external completeness.

2: Which is harder? Why?

Question 14 - Solution

Internal Completeness is concerned with making sure we have not left and holes in the requirements document.

- If we have requirements for when a button is pushed down, do we have requirements for when it is released?

External Completeness is related to making sure we have covered the user's requirements.

- If we have internal completeness regarding button A, but the customer expects button B, we lack external completeness.

Question 14 - Solution

External completeness is harder.

- It, again, requires capturing the needs of the customer.
- Internal completeness is testable.
- External completeness is subjective.

Question 15

Why is it better in a requirements document to use TBD with no other information than to simply write nothing at all?

Question 15 - Solution

Why is it better in a requirements document to use TBD with no other information than to simply write nothing at all?

It is better to point out when a piece of information is missing than to just leave it out. Errors of omission are easier to catch (search for TBD).

Question 16

In the class, we discussed non-functional requirements.

1: Explain the concept of non-functional requirements and give two examples.

2: How is the notion of "non-behavioral" requirements different from non-functional requirements?

Question 16 - Solution

1: Explain the concept of non-functional requirements and give two examples.

Requirements that do not impact the correctness of the functional behavior (the services the system performs).

Usually related to security, performance, reliability, maintainability, etc.

Question 16 - Solution

2: How is the notion of "non-behavioral" requirements different from non-functional requirements?

Non-behavioral requirements do not impact the execution of the system in any way. Cover organizational and development constraints.

"The system shall be programmed in C++."

"The developers shall use the Eclipse IDE."

Any other questions?

Next Class: The Midterm

Any questions?