

## Announcements

Please take out a sheet of paper and:

Write your name

Write a grade-posting code

(last 4 digits or anything else)

Fold paper

Pass or bring to front

## Use Cases for Requirements

- Today's material: Start at section 6.8
- Consider 3 use cases, all valid, 1 good:
  - Log in
  - Handle a customer returning item(s)
  - Negotiate a supplier contract
- These have different *granularity*
  - For UML assistant system...
    - granularity can also be good or bad
- Granularity must suit the purpose
  - For POS system requirements analysis
  - The right granularity is the "EBP"

## "EBP"s – What are they?

- EBP=Elementary Business Process
  - Of course, registration is not a business
  - ...nor is software for mobile phones
  - ...nor is a UML assistant system
  - ...etc., etc.,...
- The text discusses EBPs
  - But the concept is more general than I T
- Let's define EBPs, then generalize to other apps...

## Elementary Business Processes

- "A task performed by one person in one place at one time, in response to a business event, which adds measurable business value and leaves the data in a consistent state."
- This is a good definition
  - It's precise, with little fuzziness
  - It is highly business-domain oriented, alas
  - Luckily, it doesn't have to be
- How might it be generalized to apply beyond just the business domain?

## Another Granularity Guideline

- A good use case takes several steps
  - Not one small step (print a document)
  - Not several people and sessions (negotiate a contract)
  - What about logging in? Logging out?
    - Do they meet the previous slide as well?
    - Larman is not sure (pp. 62-63)
- Sometimes, there are exceptions
  - (We'll discuss them another time to avoid muddying the waters now)

## Goals

- Each use case should accomplish a user goal
  - (as noted last time)
- If a use case meets the EBP constraint
  - ...so will the goal it accomplishes
  - (possible exercise: explain this fact)
- If you ask the user "What do you do?"
  - ...the resulting use cases will reflect **current procedures**
- If you ask "what are your goals?"
  - ...the resulting use cases may creatively **improve procedures**
- Which is better?
- So that's the way Larman recommends (p. 6)
- Example: processing a sale
  - The goal-derived use case could contain new ideas
- Example: course registration
- Example: UML assistant

## Use Case Development Procedure

- (See section 6.9)
- **Step 1:** determine the system **boundary**
  - What's in the system and what's not?
  - This has got to be basic to *any* system development task
  - You can't begin to build a system without knowing what it is!
  - ...that's why it is step 1
  - Question: are there things that interact with system components that are outside the system?
- What are some things not in an operating system
- What are some things not in a POS system?
- What are some things not in a newscaster support system?
- What are some things not in a men's gymnastics real-time score calculation and display system?
- What are some things not in the course registration system?
  - (courses? students?)
- What are some things not in a UML assistant system?
  - (printer? paper? what else?)

## Use Case Development Procedure (II)

- Systems interact with things outside them
- **Step 2:** we must identify the **primary** such actors
- Primary actors: system supports *their* goals
- Supporting actors: they support the *system's* goals
- Offstage actors: interested in what the use case does, but not primary or supporting
  - E.g. the sales tax division of the revenue agency (sec. 6.12)
- Larman proposes "reminder questions" (p. 64) for finding actors
  - Which identify primary actors? Supporting actors?
  - ... (see next slide)

## The "Reminder Questions"

- Primary actors: system supports *their* goals
- Supporting actors: they support the *system's* goals
- **Which identify primary actors for POS system? Supporting actors?**
- (following quoted/paraphrased from p. 64)
  - People who start and stop the system
  - People who do system administration
  - People who do user and security management
  - Is 'time' an actor because the system does something in response to time events?
  - People who evaluate system activity or performance
  - People who evaluate logs
- p. 65: "Be suspicious if no primary actors are external computer systems"
  - The POS system is a primary actor for what?
  - Is another system a primary actor for the POS system?
  - What other primary actors does the POS system have?

## About System Boundaries and Primary Actors...

For the POS system, why is the cashier a primary actor but the customer is not?

What is a primary actor for the customer?

## Use Case Development Procedure (III)

- Recall: primary actors - system supports *their* goals
  - **Step 3:** Identify the goals of each primary actor
    - State each goal consistently with the EBK guideline:
- "A task performed by one person in one place at one time, in response to a business event, which adds measurable business value and leaves the data in a consistent state."
- For the POS system:
    - What goals do the primary actors have?

## Use Case Development Procedure (IV)

- **Step 4:** define the use cases
  - Make one *use case* for each *goal*
  - *Name* the use case similarly to the goal
    - (makes sense...)
  - Start name with a *verb*
- Defining use cases can take
  - a few minutes ... to
  - a few weeks
- Let's go with the "few minutes" option for goals derived in the previous slide...

## How to Perfect the Use Cases

- They will not be perfect
  - ...so they have to be improved
- Traditional waterfall approach:
  - Try harder. Spend more time on the requirements phase.
  - Fail anyway (often)
- Unified Process (UP) approach:
  - Continuous communication between developers and client
  - Continuous improvement from one iteration to the next
  - Recall the disciplines diagram (next slide, again)
- Extreme Programming (XP) approach:
  - "User full-time on the project, in the project room"

## Disciplines in the RUP (not quite fig. 2.4)

