Case Study: Inception Phase

L. ch. 3-5
An Example System

- Let’s consider a familiar example: a *POS system*

- A learning strategy:
  - Learn ideas and concepts on the POS system
    - UML itself is among those ideas/concepts
  - Apply ideas/concepts to a UML editor tool

- We first discuss the POS *domain* (L chap. 3)
  - ...this prepares us to learn OOA/D using the POS example
  - ...which in turn prepares us to use OOA/D on the UML tool system
The POS System Domain: a Case Study

- POS = **Point Of Sale**
- POS systems are typically used for retailing
  - Supermarkets, bookstores, etc. use POS systems
- They are ubiquitous nowadays
- They are important to build right
  - ...or the grocery store has to close!
  - Walmart uses two, just in case
- *Objects* are good for building them with
Facts About POS Systems

- They handle customer payments
  - ...so they work in real time
  - They incorporate cash registers
  - What else do they incorporate?
- They record what is sold
  - This enables tracking inventory
More Aspects of POS Systems

- Fault-tolerance is important
  - Especially fail-softness

- *Generic* POS systems must have generality
  - They must adapt to different clients
    - ...i.e. different sales environments
    - ...different hardware platforms
  - Therefore at standard points in the sale
    - Custom code must be invoked
  - Note the mix of *generic* and *custom*
    - This provides an information systems analysis challenge
Layered Structure and the POS System

- Complex systems have a layered structure

- Example: fig., next slide
What layer(s) are emphasized in VB? C? C++? Java? Device interfacing? OOA/D? What might an analogous figure look like for a course registration system or UML editor?
Creating the POS System

- Recall the phases (*in alphabetical order*)
  - Construction, Elaboration, Inception, Transition

- What do these mean?
- What is their order?

- Let’s recall a Figure
  - (2.3/2.6, 2nd/3rd eds., and next slide)
Inception and the Fountain Model

- Recall the UP diagram:

- What corresponds to inception in a typical non-iterative life cycle model?
Inception is about:

... the project’s **scope**, **vision**, and **business case**

**scope:**
- What it will do, in general terms
- *Try this:* pick something in the scope of the UML or registration system, and something outside the scope

**vision:**
- The problem and its solution
- *Try this:* pick something in the vision of the UML or registration system, and something outside the vision

**business case:**
- See next slide...
The Business Case

Source: mostly quoted from: http://www.arsanjani.org/business-modeling/Lesson%203%20-%20RUP.pdf

- Purpose of the Business Case is to develop an economical plan for realizing the project vision.
  - Assessment of the return on investment (ROI) provided by the project.
  - This justifies the project and establishes its economic constraints.
  - If justified: the project should proceed.
    - If not, cancel it!

- The business case should not delve deeply into problem specifics
  - It should argue compellingly why the project is needed.

- It must be brief, so it is easy for project team members to understand and remember.

- At critical milestones, the Business Case is re-examined to see if estimates of expected return and cost are still accurate
  - Worst case – discontinue project (recall spiral model – see fig)
  - This is better than continuing the project!

- How do these points apply to the UML or registration system?
Spiral Model (an elaborated waterfall) – note risk analyses

Inception in Two Bullets

- Project scope, project vision, and the business case
- Reach stakeholder agreement on the project vision and business case
Artifacts Often Started During Inception

Artifact = a thing made by people
  - (same root as “artificial”)
- Vision and business case
- Use-Case model (to be covered later)
- Supplementary Specification
- Glossary (of domain terms)
- Risk List and Risk Management Plan
  - (Risks and what-if responses)
- (Rapid) prototypes (i.e. throw-away code)
- Iteration Plan (what to do in 1st elaboration iteration)
- Phase Plan (guesstimate of phase efforts & durations)
- Software Development Plan (resources of all kinds needed)
- Development Case (what UP items apply to this project)

Let’s apply a few of these to the UML or registration system now...
Requirements (E.g. Larman ch. 5)

- What the system should do and under what conditions in general terms
  - Does this differ from specifications?
  - Does this differ from scope?
- UP assumes requirements can change over time
  - Must therefore be able to iteratively change them
  - Think of a requirement that might be added to the UML or registration system later...
- Problems with requirements have major effects on project time, budget, and overall success
- Why? Early problems are hard to fix later (recall Figure, next)
- The UP uses the FURPS+ framework (after figs.)
Relative cost to fix a fault that could have been fixed in requirements
Figure 26.1: The Cost of Change

http://www.osl.iu.edu/~lums/swc/www/xp.html
FURPS+ Framework for Requirements

- **F is for Functionality**
  - “features, capabilities, security”

- **U is for Usability**
  - “human factors, help, documentation”

- **R is for Reliability**
  - MTBF, “recoverability, predictability”

- **P is for Performance**
  - “response times, throughput, accuracy, availability, resource usage”

- **S is for Supportability**
  - “adaptability, maintainability, internationalization, configurability”

- **+ is for extra stuff (legal, packaging,...)**
Let’s apply FURPS+ to the UML or registration system...