

An Example System

- Today's material: chapters 3-5 (short chapters!)
 - Check Web for HW exercises due next Thursday
- The book's major example is a *POS system*
- The course project will involve transferring understand of the POS system to a UML tool
- Our learning strategy is
 - Learn ideas and concepts using UML on the POS system
 - Apply them to the UML tool
- We first discuss the POS *domain* (chap. 3)
- (...which prepares us to learn OOA/D using POS as an example)
- ...which prepares us to use OOA/D on the UML or registration system

The POS System Domain: a Case Study

- POS = **P**oint **O**f **S**ale
- 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 sales
 - Knowing what is sold helps control inventory

More Aspects of POS Systems

- Fault-tolerance is important
- Fail-softness is important
- Generic POS systems must have generality
 - They must adapt to different clients
 - ...i.e. different sales environments
 - Thus, at standard points in the sale
 - Custom code must be invoked
 - Note the mix of generic and custom

Layered Structure and the POS System

- Complex software has a layered structure
- Example: Figure 3.1 (next slide)

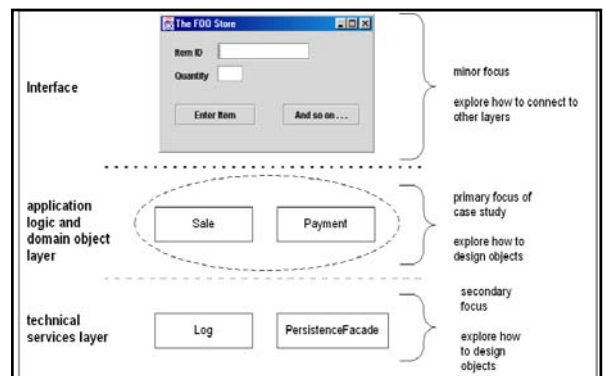


Figure 3.1. What layer(s) are emphasized in VB? C? C++? Java? Device interfacing? OOA/D? What might an analogous figure look like for a course registration system or UML editor?

Creating the POS System

- Recall the phases:
 - Inception, Elaboration, Construction, Transition
- See Figure 2.3
- We'll start with which one?

Inception

- See Chapter 4
- "Inception in one sentence: Envision the product scope, vision, and business case."
- Scope:
 - What it will do, without details
 - Pick something in the scope of the UML or registration system, and something outside the scope
- Vision:
 - The problem and solution
 - Pick something in the vision of the UML or registration system, and something outside the vision
- Business case:
 - See next slide...

The Business Case

Quoted from: <http://www.arsanjani.org/business-modeling/Lesson%203%20-%20RUP.pdf>

- Purpose of the **Business Case** is to develop an economic 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 move ahead. [Else, not!]
 - • The description should not delve deeply into the specifics of the problem, but rather it should create a compelling argument why the product is needed.
 - • It must be brief...so that 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, and whether the project should be continued [recall spiral model – see fig]
 - [How do these points apply to the UML or registration system?]

Inception in Two Sentences

- (See p. 36)
- "Inception in one sentence: Envision the product scope, vision, and business case."
- Inception answers the question,
 - "Do the stakeholders [agree] on the vision ...and is" the project worth serious consideration?"

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 elaboration phase effort & duration)
- 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 (chap. 5)

- "capabilities and conditions to which the system...must conform" – p. 41
 - 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 are the largest single cause of "problems" causing "challenged projects" – p. 42
- See Figure 5.1
- Why? Early problems are hard to fix later (see Schach Figure)
- The UP uses the FURPS+ model:



FURPS+ Model 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...



FURPS+ Model (cont.)

- + is for extra stuff
 - Legal, packaging, etc.
- Let's apply FURPS+ to the UML or registration system...