

Goals for Use Cases Again...

- Reference: Section 6.11
- Last time we discussed using *goals* to design use cases
 - The alternative was using *procedures*
- Let's review the relevant slide 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

Goals for Use Cases – Comments

- Today is the 3rd use cases class
- We've discussed goals all 3 times
- This shows the importance of the connection
- Review question:
 - Give one advantage of basing use cases on *goals* rather than current *procedures*

Goals, Use Cases, & UIs

- Recall: goals are good, procedures less so
- Thinking about goals focuses attention on
 - ...the problem *essence* (*essentials*)
- Thinking about procedures focuses on
 - ...*concrete* issues (individual details)
- Do UIs draw attention to essence or detail?

Goals Vs. UIs – Example

- Suppose the user's goal was to log in
 - The *user* may be thinking in terms of ctrl-alt-del
 - *You* (the requirements engineer) need not
- Focus on the goal:
 - What is/are the goals of logging in?
 - What is/are the goals of that/those?
 - ...and of that/those? ...
- Can these be met in a new, non-ctrl-alt-del way?

Use Cases, Goals, and UIs II

- Which is to be more recommended?
 - Generate a sample UI, then
 - use it as a guide to use case development
 - Generate something other than a UI, then
 - use that instead for use case development

Lesson: a Goals Focus Helps...

- ...helps determine new solutions
- ...helps determine creative solutions
- ...helps think outside the box
- ...helps address the essence of the problem
- ...helps make the "same old way"...
- This is consistent with computing as a changing, growing discipline
 - ...an *option*, not a *necessity*
- ...and computer engineers as changing, growing pros

Goals Vs. UIs/Procedures

- UIs are great for expressing look and feel
- They model procedures-with-branches
- Start with goals
 - Move into the more concrete UIs as followup
 - UIs are great for communicating ideas to users
 - But first you've got to get users to communicate to you
- Use case style: *essential* vs. *concrete*
 - (Essential is better)
- See example, pp. 69-70

Let's Think About the UML System

- Someone suggested MS Paint as a good model for the UML assistant system
 - This could work, but is concrete (it's a UI)
- Let's take 10 min. to try a goal-oriented approach...
 - Write a major goal of the system
 - Then decide on preconditions and postconditions
 - Then concretize with a main success scenario
 - Finally, explain an appropriate UI for this

Use Case Diagrams

(Reference: section 6.13)

- Use Case diagrams are handy for:
 - Communicating use cases in summary form
 - Summarizing system boundary, actors, and their major activities
- Are Use Case Diagrams...
 - ...a primary use case development tool?
 - Why/why not?

Use Case Diagrams – an Imperfect Language

(But it's unclear why)

- "Use case diagrams...are secondary"
- "Use cases are text documents"
- "Doing use case work means to write text"
- "World-class use case experts ... downplay use case diagrams"
- But use them judiciously anyway
 - (Everyone else is)

What Makes Languages Good or Bad?

- Write one thing down
- I'll call on "volunteers" and we'll make a list...

(Note that Use Case Diagrams are a language)

Use Case Diagrams – an Imperfect Language II

- An ideal language in the computing field...
 - Is expressive (you can say what you want)
 - Is constraining (you can only say good things)
 - Supports need for high productivity
- C++ is not constraining enough (neither is C)
- C is also not expressive enough (e.g. UIs) (C++?)
 - Hence, Java and other OO languages
- Larman's criticism of use case diagrams...
 - ...suggests text is more expressive and more productive
 - ...but the reasons are mostly missing
- Possibility:
 - a future language will be better than either text or use case diagrams

Use Cases are not Specifically OO

- Use cases are not object-oriented
- Objects aren't part of the discussion
- ...they can be used in
 - Object oriented requirements analysis
 - Non-object oriented requirements analysis
- This makes use cases more powerful
- Use cases are an important input for OOA/D
- Use cases are an important part of UP...(see next)

Use Cases and the Unified Process

- Use cases are an intrinsic part of the UP
 - They are the primary requirements method
 - Use cases help define content of iterations
 - The functionalities dealt with in an iteration are determined by ...
 - use cases, or
 - use case scenarios
 - This makes sense!

Exercises – Examining Our Use Cases

100 pts
Due: Tuesday 10/12/04 by class time

- What is the goal of your fully dressed use case?
- Can it be abstracted usefully? E.g., from "log in" to "authenticate user"
 - Give an abstracted version of the goal and say why this is better or worse than the previously written goal
- Write a *brief format* use case, if you haven't already got one
- Use cases should provide value
 - Write the value produced by each use case scenario
- To which letter of FURPS+ are use cases best matched?
- For each use case scenario, write the number of any white box (design-providing) steps
 - How could they be rewritten in black box (design-neutral) style?
- Similarly, for each use case scenario, are there any steps that assume a user interface characteristic?
 - For any such steps, how could they be rewritten to be UI-neutral?
- If you used 1-column format, say why 2-column might have been better (if you used 2-column, why might 1-column have been better?)
 - If you were to redo it would you use the other format? If not, why was your original choice to be preferred?
- Explain why your use case represents an EBP (or does not). Do the same for each of its scenarios
- For each use case scenario, what are the supporting actors?
- For each use case scenario, is time a primary actor? If so, why, if not, why not? See textbook for relevant discussion
- For each use case scenario, is there a system (not a person) that is a primary actor? If so, why? If not, why not?
- For the use case, each team member should draw it using a *different* graphics editor or CASE tool