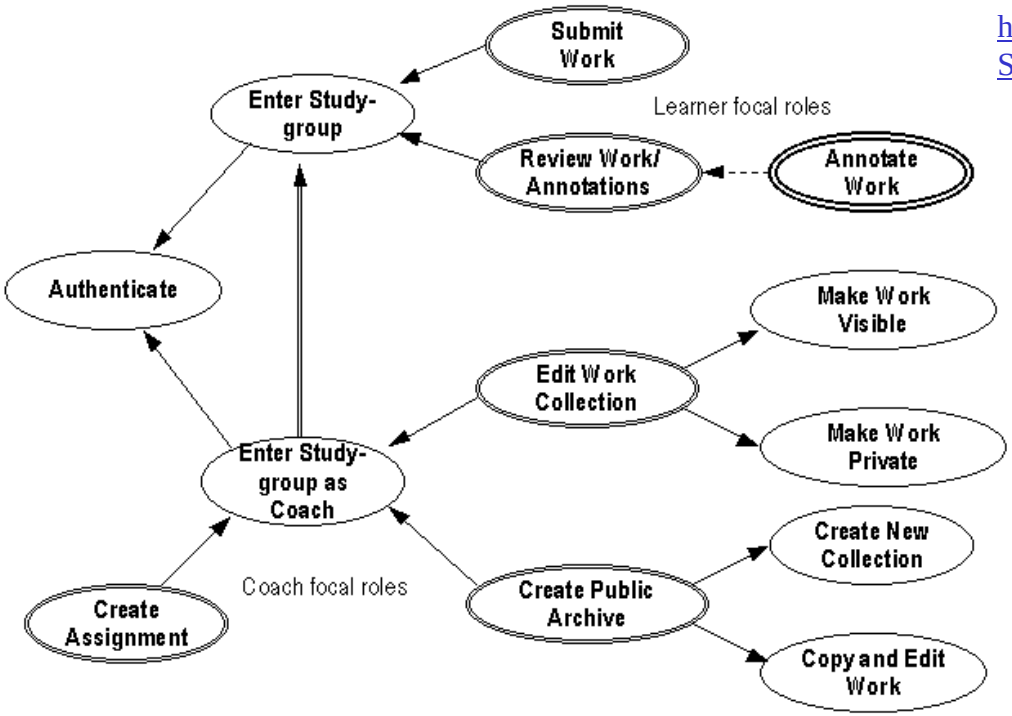


Use case map for the NetLearn project, from http://lilt.ics.hawaii.edu/classes/ICS691/Spring2000/Projects/Dan-Suthers/use-case-map.html.

Elaboration of Use Cases

CSSE 371, Software Requirements and Specification
Steve Chenoweth, Rose-Hulman Institute
October 21, 2004

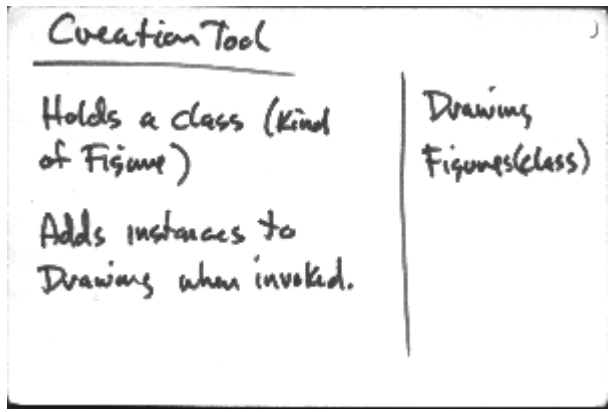
In the book – This is Ch 25 - 26

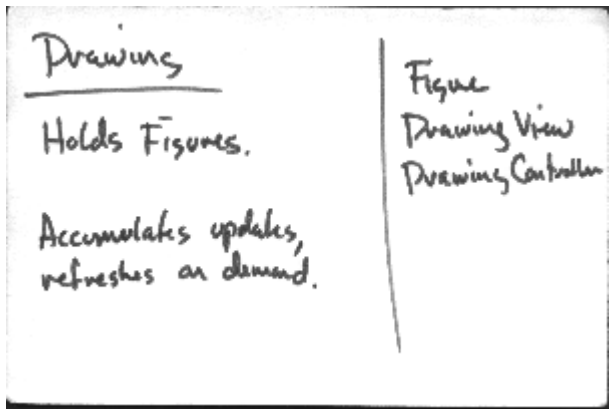
What do use cases become?

- Eventually, like we see at the web site http://wiki.cs.uiuc.edu/cs497rej/OIMS+-+DETAILED+USE+CASES.

- Goal – People can design from these! How do they do that?

- One example – From CRC Cards…
  → See the Cal Poly tutorial at http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/crc_b/.
  Or Alistair Cockburn's tutorial at http://alistair.cockburn.us/crystal/articles/ucrcc/usingcrccards.html.

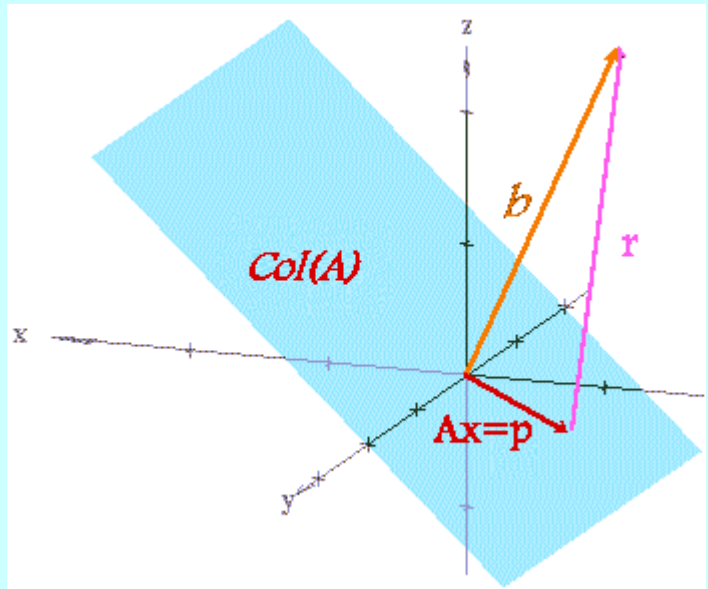




CRC = "Class Responsibility Collaborator"

Getting There: What are the issues?

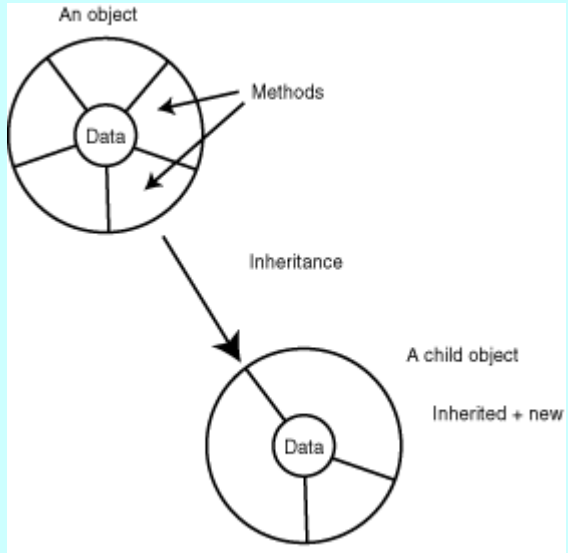
- Sometimes it is easier to design and implement a system feature than others
- For instance, if a feature mostly involves interaction between the software and one or more actors, the use case events correspond well with the steps of the implementation algorithm
- However, in other cases, such as when quality requirements or complex data structures are involved, the design does not follow as directly – this is problem of *orthogonality*.



→ You *have* to stir in use cases and those quality attribute scenarios!

Getting There: And the fix for "orthogonality"?

- Some potential solutions (don't we hope!):
  - Object-oriented design
  - Well-written use cases
  - Modeling the *software architecture*
- The software architecture helps us understand what the system does and how it works, how the elements interact, and what type of patterns and pieces or the system are involved.



→ Example – A typical OO design process (like CRC cards) assumes you are running on a single processor, without ever saying so…

→ OO hides architectural problems, along with other "detail"

So, we try to resolve some design issues with "architecture"

The software architecture helps us understand what the system does and how it works, how the elements interact, and what type of patterns and pieces or the system are involved:

- Architectural views :
  - Logical (for users)
  - Implementation (programmers)
  - Process (integrators)
  - Deployment (system engineers)
  - Use-case view (designers/testers; ties things together)

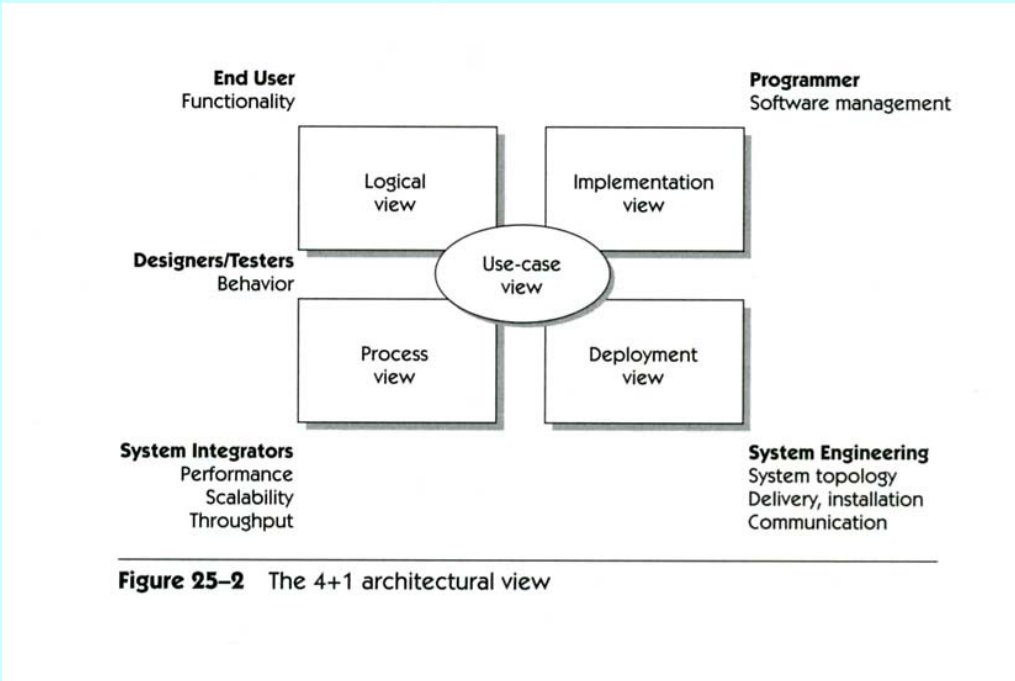


Figure 25–2 The 4+1 architectural view

→ Likely test question – "What is System(s) Engineering?" (2 answers)

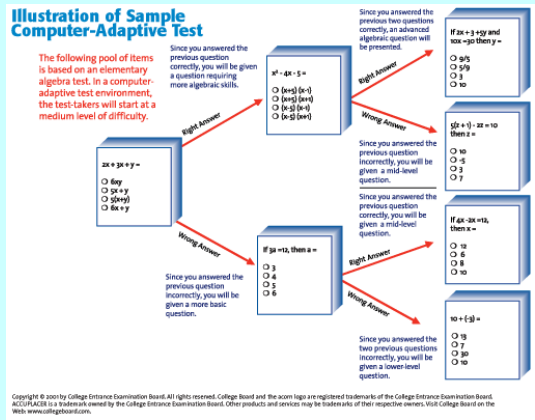
One answer – See http://www.rose-hulman.edu/Class/EngMgmt/HTML/Courseoffering.htm#MG%20587.

When Tori[1] was here on Tuesday…

She said, “We do the architecture first, then get requirements from the customers.”

→ Why would you develop software that way?

→



?

Left – Architecture example from www.bigfoto.com/ themes/architecture/. *Right* – Requirements example from the dreaded www.collegeboard.com/ highered/apr/accu/accu.html.

[1]Tori Bowman, of Rockwell Collins. RHIT CS grad, June 2004.

Later in the development cycle … What about testing with use cases?

- *Validation* test cases are ones that are performed from the user's point-of-view.
  - They are examples of *black-box* test cases
- Use cases are not exactly validation test cases, although use cases do drive the process of validation test case generator
- Steps in creating a test case:
  - Identify use-case scenarios (basic and alternate flows)
  - Identify the test cases
  - Identify the test conditions
  - Add data values to the test case

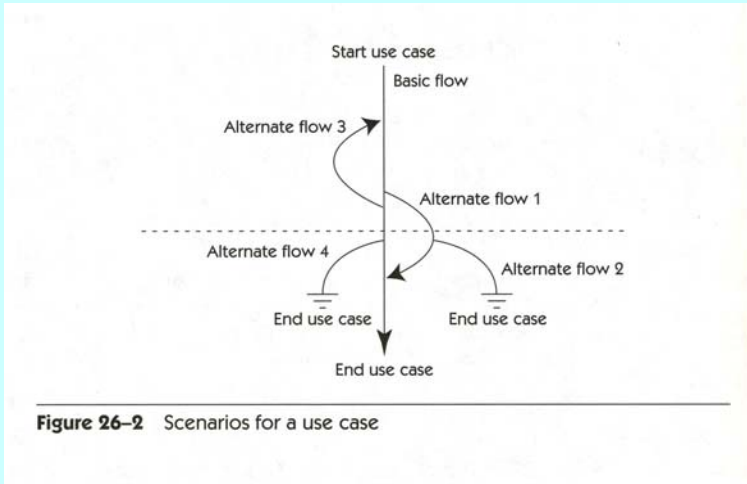


Figure 26–2 Scenarios for a use case

→ You'll have a chance to try creating a few of these in HW 10.

But, not so fast…

Here are the basics…

1. There's a "test plan" for how to do these generally.
2. The use cases each become multiple test cases.

Team Exercise (5 min.):
Where in your term project would you want lots of test cases?

And how do you know?

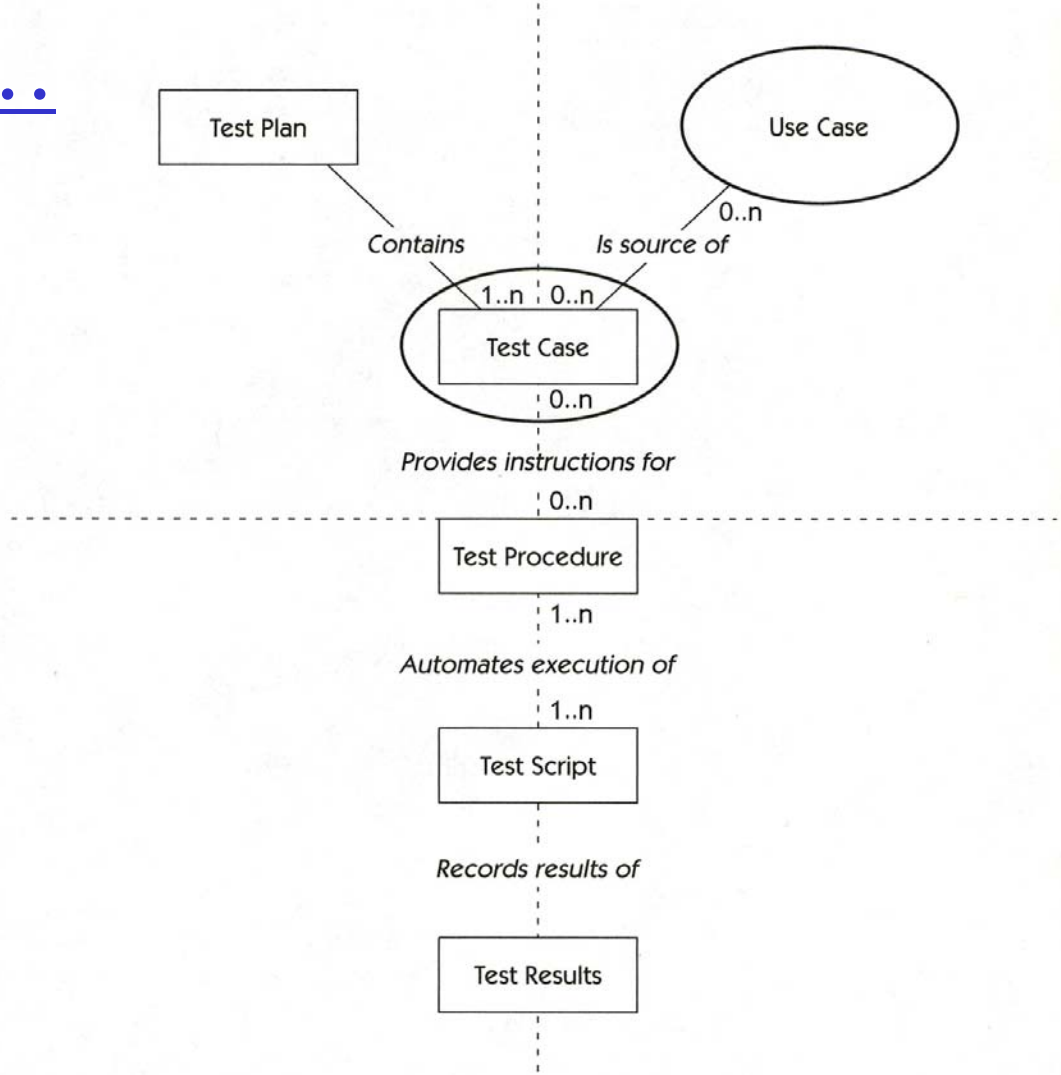


Figure 26–1 Relationships among test artifacts

→ Read the rest of Ch 26 before you do HW 10 !