Inception Objectives

J.W. Benham
CMPT 371 – Fall 2005
Montclair State University
Objectives of Inception

• Understand what to build
• Identify key system functionality
• Determine at least one possible solution
• Understand the costs, schedule and risks associated with the project
• Decide what process to follow and what tools to use

From Kroll & Kruchten,
*The Rational Unified Process Made Easy*
Understand what to build

• Produce a Vision
• Create a “mile-wide, inch-deep” description
• Detail key actors and use cases
Identify Key System Functionality

- Functionality is core of application or it exercises key interfaces of system
- Functionality must be delivered
- Functionality covers an area of architecture not covered by any other critical use case
Determine at least one possible solution

- Make sure there is at least one potential architecture that will allow you to build system at reasonable cost with a sensible amount of risk
- This may not be the architecture you actually use to build system – that’s determined during Elaboration, after you have a deeper understanding of requirements.
Understand costs, schedule, risks

- Business risks
- Project risks
- Product risks

From Ian Sommerville, Software Engineering (7th edition), pp 104-105
Risks of CESS

• Business risks
  – So far, other coop faculty have not expressed interest

• Project risks
  – Developer’s limited experience with Java and database technology

• Product risks
  – Performance risks related to bandwidth and load on server
  – Data integrity risks related to concurrent processing on server
Mitigation of Project Risks

• Developer’s database inexperience
  – Try to use a simple file structure initially, and see if it will handle required amount of data with adequate performance
  – Use Adapter pattern, so that we can replace files with database if required

• Developer’s relative Java inexperience
  • Study the necessary advanced features of Java (concurrency/threads, GUI support, etc.)
Mitigation of Product Risks

• Limited bandwidth
  – Use a “thin client” architecture
  – Proxy pattern may help

• Load on server
  – Risk judged to be low, since application is not large
  – Need plan to purge no-longer-needed persistent data
    (another use case for System Administrator?)

• Concurrency
  – Risk judged to be significant, since it’s probable that at some time more than one user will be on system
  – Learn how to use Java Threads to handle this risk
Determine a possible solution

• Thin client
  – Use of Proxy pattern (remote proxy)
• Simple files (initially) for persistent storage
  – Use of Adapter pattern