Life Cycle Objectives (LCO)
Readings

• *Anchoring the Software Process*, Barry Boehm, USC
Outline

• Life Cycle Objectives
• Assignment 1 – LCO review for your product idea
Life Cycle Objectives Elements

1. Operational Concepts
   What is it? high level scope and objectives

2. System Requirements
   What is it? lower level actual planned deliverables

3. System and software architecture
   How – technically?

4. Lifecycle plan
   How – resources?

5. Feasibility Rationale
   Is this realistic?
1: Operational Concepts

- Top-level system *objectives* and *scope*
  - What problem are you trying to solve? Why? For whom?
  - User community, environment, major benefits?
  - Goals and non-goals – set realistic expectations
1: Example

- Problem:
  - What: create a system for displaying and playing musical chords
  - Why: Classroom demonstrations
  - For whom: A music teacher

- User community: music teachers, music students

- Goals: Display musical chords, modify them interactively, save chords for use in future classes

- Non-goals: This is not general music editing software.
2: System Requirements

• Essential features of the system
  ○ defined at a level appropriate to the spin cycle
  ○ capabilities, interfaces, reliability levels, appearance
  ○ Easy to change early on, grows increasingly more difficult

• Customer’s involvement very important
  ○ they know the domain of interest far better than you do
  ○ what fits with their daily work and life patterns
  ○ what might the future bring

• Neither you nor the customer know everything
  ○ try to build joint ownership of the process
  ○ open communication can make change more acceptable
2: System Requirements (cont.)

• The act of writing the spec -- describing how the program works [from user perspective] in detail -- will force you to actually design the program
  • you get a chance to see the potholes before you fall in
    □ The customer can review it and sign off quickly or complain early.
    □ If you put it in writing, it’s less ambiguous than if you just say it.
    □ Forces you to think of major functional areas, major architectural defects
  □ A picture/diagram is (often) worth 1000 words.
  □ Scenarios/stories help, but don't be verbose.

• “Failing to write a spec is the single biggest unnecessary risk you take in a software project” Joel Spolsky

• HW tip: You do not have to do a full spec for this assignment. You will have a chance for that in the LCA.
2: Example

• The chord visualizer will be used to demonstrate chords in a music class. Users can input chords by mouse, external keyboard*, or file+.

• The chords will be displayed on a musical staff. A highlighted box will indicate the currently selected chord.

• Users can play back chords*

• Users can save chords to a file to be reloaded later+.

* User input
+ Developer input
3: System and Software Architecture

- High-level description but with enough detail to allow feasibility analysis
  - Can this really be built with the available resources?
  - Architectural flaws will only deepen as you go forward, so consider alternatives while it’s still early.
  - This is necessarily a technical discussion (unlike the prior two elements).
- **HW Tip:** Identify clients and servers and major software components and interactions between them
  - If attaching to 3rd party software, verify the interface exists
  - Pictures say 1000 words
3: Example

- System will be implemented in Java
- File format will be in XML
  - Challenge: Are the necessary XML parsers available? Is the overhead of XML necessary?
  - Alternatives: Flat text file, binary file
- Input and output will be MIDI
  - Challenge: Does Java support MIDI input and output
- System will have a concept of notes, chords, and chord sets
  - Alternatives: What other formats could support this? Can this be extended if you want to have multiple sets of chords?
3: Example

- Input file reads
- MIDI input writes
- Chord manager generates
- MIDI output generates
- Visual display manipulates
- MIDI input manipulates
4: Life Cycle Plan

• Identify stakeholders and their roles:
  ◦ Users, customers, architects, developers, etc.

• WWWWWWHHH: Why/What/When/Who/Where/How/How
  ◦ Objectives - Why is the system being developed?
  ◦ Schedules - What will be done, When?
  ◦ Responsibilities - Who will do it? Where are they?
  ◦ Approach - How will the job be done?
  ◦ Resources - How much of each resource?

• HW Tip: You only need do a little now.
4: Example

- Stakeholders: client, teachers, students, development team
- **Why** is the system being developed?
  - This system is being developed as an aid for teachers to teach students about chords
- **What** will be done, **When**?
  - Ex. Chord display by and input by week 4, file input and output by week 6, etc.
- **Who** will do it, **Where** are they?
  - This can wait until there is a team
- **How** will the job be done?
  - This can wait until there is a team
- **How much** of each resource?
  - 6-8 team members, 9 weeks, 1 MIDI input device…
5: Feasibility Rationale

- Conceptual integrity and compatibility
  - What assumptions are you making? Any unwarranted ones among them?
  - Identify risks
5: Example

• Assumptions:
  ◦ Simple chords sufficient for teaching
  ◦ Java supports MIDI input/output
  ◦ Users want to be able to save/load chords

• Risks:
  ◦ Does the team have enough musical knowledge?
  ◦ Does the team have enough GUI programming knowledge?
  ◦ Unwarranted assumptions?