Today: begin study of the **Unified Process** (An information system life cycle model)

Often abbreviated **UP**

A version is called **RUP**

RUP = “Rational Unified Process”

‘Rational’ from the name of the company, now part of IBM
“I have dealt a lot with software/solution specifications through my work. One of the common problems I have encountered dealing with specifications is when a client decides to make significant changes to the original specifications well into the implementation phase. This might be acceptable if I were following an iterative approach where the specifications were intended to evolve over time, however, it has almost always been the case that the original set of specifications were intended to be final and static. This has resulted in me having to do significant rework including recoding certain aspects of the software that were not robust enough to accommodate the unexpected changes.” (N. Pippenger, 4/09)
The Unified Process

- Reading for today’s material: chapter 2
- Background on life cycle models
  - Fountain model – for OOA&D (see Fig.)
  - The fountain may be incremental or not
- Unified Process –
  - Based on the incremental progress idea
  - Let’s look at the Unified Process more...
Fountain Life Cycle Model

A waterfall model. Notice similarity to fountain!
(Source: http://www.softprayog.in/papers/waterfall_model.png)
Iterative and Incremental Development

- Using the Unified Process means building an information system...
  - ...iteratively
  - ...incrementally
- See next slide (also see Fig. 2.7 in text)
  - (also Fig. 2.3/2.6 in Larman 2\textsuperscript{nd}/3\textsuperscript{rd} ed.)
  - Let’s relate it to the incremental fountain
- The OO analysis, design, implementation, etc., phases map \textbf{very} fuzzily to the figure
Unified Process (UP) Life Cycle Model

Iterations

Inception  Elaboration  Construction  Transition

Phases

time
Finding Middle Ground

- The relatively rigid fountain often isn’t ideal
  - Often, you can’t *lock in* analyses and designs
  - Non-iterative fountain can work for *standard* systems
  - Developing *novel* systems needs more flexibility
- Incrementalism is between two extremes
  - the too-inflexible fountain model, and
  - the (*really bad!*) build-and-fix model
- So what makes the incremental model good?
Models, Ideologies, and Extremes

- If the fountain idea has limits...
  - Is incrementalism the perfect answer?
- Like any model, philosophy, etc...
  - ...incrementalism has limits too!
- Example: temperate fruit system
  - (From Schach, 5th ed., p. 497; see URL only accessible to students in this class)

- Note:
  - UP fans would say UP could handle this
Models, Ideologies, and Extremes II

A philosophy:

- Models, ideologies, philosophies, etc., *have limits*
  - (Some people extend political and religious ideologies beyond what most would consider reasonable)

- Can this philosophy itself be over-extended?
Unified Process: More Motivation

- Recall: using the Unified Process means building software
  - ...iteratively
  - ...incrementally
- Specs often change as projects proceed
  - ...often, clients can’t reliably specify until they see the context – a working program
  - ...work with that fact, instead of fighting it
Unified Process: Some Details

- Iterate a analyze-design-implement loop
- For each iteration:
  - Specify some “small subset of” specs
  - “Quickly” design, implement, and test them
  - Get feedback from users, performance tests, etc.
- Next iteration:
  - Do more requirements
  - Or, rework the last iteration to get it right
Does the System Stagger about Uncontrollably??

- Recall the build-and-fix life cycle model
  - That’s the problem that motivated the creation of the software engineering field:
    - *Too much* staggering around

- Fountain, waterfall, etc. models:
  - *Insufficient* staggering around!

- Incremental life cycle models
  - (e.g. the Unified Process)
  - “Controlled staggering” is the way to go
    - See fig., next slide (and 2.2 of Larman, both eds.)
Theoretical ideal (short)
Build-and-fix model
Unified process
What would a strict waterfall model look like?
Some Advantages of Iterative Development

- Reduce risk of major backtracking
- Backtracking is expensive!
  - That’s why the build-and-fix model is bad
- (Recall illustrative figure)
- Lesson: do the most risky things first
  - ...i.e., in an early iteration
Relative cost to fix a fault that could have been fixed in requirements
Figure 26.1: The Cost of Change

http://www.osl.iu.edu/~lums/swc/www/xp.html
Advantages of Iterative Development II

- Everyone (you & client) can see progress
- Feedback from clients is early & often
- Breaks overall task into manageable bites
  - ...avoid “analysis paralysis”
- Each iteration holds lessons for the next
  - ...this facilitates process improvement
  - CQI = ‘Continuous Quality Improvement’
  - Can use Capability Maturity Model
    - (see Fig. 2.1, Schach 5th ed., and next slide)
  - ISO 9001 is another approach to process quality
Characteristics of the Maturity levels

- **Level 1: Initial**
  - Processes unpredictable, poorly controlled and reactive

- **Level 2: Managed**
  - Process characterized for projects and is often reactive.

- **Level 3: Defined**
  - Process characterized for the organization and is proactive.
    (Projects tailor their process from organization’s standard)

- **Level 4: Quantitatively Managed**
  - Process measured and controlled

- **Level 5: Optimizing**
  - Focus on process improvement

More on the Iterations

- Too short iterations –
  - Not enough is done to get meaningful feedback

- Too long iterations –
  - Backtracking risks increase
  - Start losing the benefit of iterative development

- Just right iterations
  - 2-6 weeks is usually reasonable
  - Huge teams – up to 6 months
  - Example: Canadian air traffic control system
    - 150 programmers
    - 6 month iterations
    - ...but subteams can have shorter sub-iterations
Unified Process – Best Practices

- Use timeboxed iterations
  - ...i.e. **scheduled** iteration completion time
  - ...this can vary from one iteration to another
- OO (is this intrinsic to incremental model?)
- Do the riskier things in the earlier iterations
  - Why?
- Keep clients involved
- Settle basic architecture early
- Employ use cases and UML diagramming
Grouping Iterations into Phases

- See fig. again (next slide, and fig. 2.7 in text)
  (2.3/2.6 in Larman 2nd/3rd ed.)
- Phases impose some extra structure
- Some iterations are different from others!
- Phase 1: Inception
  - often only one iteration
    - Develop vision and project scope
    - Rapid prototyping (not reusable!) is good
    - Really, the inception phase is an admission:
      - Iterative development can be taken too far
      - (Recall earlier note on over-extending philosophies/extremism)
Unified Process (UP) Life Cycle Model

Phases

Inception | Elaboration | Construction | Transition

time
Phase 2: Elaboration

- Iteratively develop core of system
- Resolve riskier parts of project
  - If project not doable...
    - find out sooner rather than later
  - Resolve project scope and resource estimates
- Take-away lesson – iterations in the elaboration phase are *non-routine*
Phase 3: Construction

- The iterations are more routine
- Each iteration results in a better program
- Hopefully, no major surprises occur
  - See Fig. yet again...
Unified Process (UP) Life Cycle Model

Iterations

Phases

Inception  Elaboration  Construction  Transition

time
Phase 4: Transition

- System is moved into being deployed
- Includes
  - Beta (final) testing
  - Bug fixes
  - Tuning
  - Training
  - Etc.
    - Larman 2nd ed. devotes about ½ page to this phase
    - Appears to show limits of iterative concept
Unified Process “Disciplines”

Rational Unified Process: Overview

Disciplines

- Business Modeling
- Requirements
- Analysis & Design
- Implementation
- Test
- Deployment
- Configuration & Change Mgmt
- Project Management
- Environment

Phases

- Inception
- Elaboration
- Construction
- Transition

Iterations

Initial, Elab #1, Elab #2, Const #1, Const #2, Const #N, Tran #1, Tran #2

Unified Process: More Comments

- Unified Process contains many, many activities and artifacts
  - Activity – something you *do*
  - Artifact – a work *product*
- Many of these are “optional”
  - ...use the ones appropriate to the project
- Toolbox analogy –
  - UP is like a toolbox – use what’s needed
  - (Like the many options in UML)