3/31/2010

Reminder:
Your first task
List 0-5 preferred teammates
on Catalyst WebQ
(open until 5PM today)
Final projects teams set at
section on Thursday

Partial schedule posted
• Requirements
documents due dates
• Weekly status reports

50MLOC = 50 million lines of code
• 50 lines/page-side
  ⇒ 1M page-sides
• 1K page-sides/ream
  ⇒ 1K reams
• 2 inches/ream ⇒ 2K inches
• 2K inches = 167 feet
  ≈ twice the height of
  the Allen Center

50 lines/page @ 50
wpm ⇒ 50MLOC/5M min
• 5M min = 83,333 hr
  = 3,472 days ≈ 10
  years

Addressing software complexity:
Three related dimensions

• Artifacts
  What are/is the…?
  • Requirements
  • Design
  • Implementation
  • Testing plan
  • …

• People
  Who does the…?
  • Requirements
  • Design
  • Implementation
  • Testing
  • …

• Process
  When are … done?
  • Requirements
  • Design
  • Implementation
  • Testing
  • …

• In other words, significant software can’t be built using a single
actant by a single person at a single instant
• Every significant piece of software is built with some view –
  albeit, often implicit – with respect to these three dimensions
• There are huge variations in each of these dimensions and in
  their composition

Just to type!
No breaks and
no thinking allowed!
Software lifecycles

- The process is often characterized by a software lifecycle – “what is done when”
- There are different lifecycle models, with different tradeoffs
- Examples include
  - Ad hoc/code-and-fix
  - Waterfall
  - Spiral
  - Evolutionary prototyping
  - Staged delivery
  - Agile (XP, scrum, …)
  - …

Ad hoc lifecycle

- “Go for it!”
- Advantages
  - Very easy to learn and to apply
  - Might work in some small and short-lived projects

Disadvantages include

- Unclear scope and timing
- Might forget or neglect some important activities like testing, design, documentation, …
- Task assignments are inherently unclear
- Hard to scale across multiple people, really hard to scale geographically
- Finding a problem later in a software lifecycle usually leads to greater cost to fix it
- Hard to change after the fact
- …
With early attention to process

Lifecycle stages

• To organize this morass of ad hoc development, each lifecycle generally considers at least the following activities
  – Requirements
  – Design
  – Implementation
  – Testing
  – Maintenance
• But in what order, and how are they combined?

Adopting/using a lifecycle

• Provides structure
• Encourages progress by the full group – less structure tends to allow individual progress that is locally good by inconsistent with the overall objectives
• It’s a management tool

Software lifecycle: classic waterfall

• The waterfall model was the first software lifecycle description [Royce 1970]
  – Not merely programming
• One develops artifacts for each level in succession
• Structured feedback
Waterfall

- **Advantages**
  - Good for projects that are well-understood but very complex
  - Tackles all planning early on
  - The ideal of no midstream changes can produce an efficient software development process
  - Can provide support for an inexperienced team

---

**Disadvantages?**

Disadvantages include

- Hard to specify all obligations for a stage completely and correctly *a priori*
- Hard for team members – or, indeed, customers – to sense the progress that is being made
- Integration is at the end
  - Harder than expected, almost certainly (“integrate early and often”)
  - Inflexibility
  - Solution may not meet customer needs
- Phase reviews are expensive, changes are discouraged

---

Spiral model – risk-oriented [Boehm]

- A disciplined sequence of activities intended to reduce risk in phases
  - Bigger risks earlier – right? (Phase 1, 2 = does needed technology exist?)
  - Each spiral reduces risks and makes progress towards overall goal
  - Each quadrant is a different stage in planning and actions
  - The length of the spiral represents the cumulative costs
  - One 3/4 turn would a waterfall model

---

**Disadvantages include**

- Lots of planning and management
- Need flexible customer and contract
- Demands risk-assessment expertise
- …
Evolutionary prototyping model

Advantages

• Addresses risks early
• Produces steady signs of progress
• Useful when requirement change rapidly and/or customer is non-committal

Disadvantages

• Requires close customer involvement
• Hard to estimate completion schedule or feature set
• Inexperienced developers may misuse it and increase feature creep, poor use of time, etc.

And many more lifecycle models

• We’ll come back to the agile family soon
So, what's the best lifecycle model?

One size does not fit all

- Hence, so many models
  - Plus we've learned a lot!
- Software is built in many different circumstances under many different constraints
- A mix of models is often used in practice – combine them and tailor them to the environment, the project, etc.
- Criteria for selection include: the task; risk management; available people; quality/cost control; predictability; visibility of progress; involvement of customers; and more…

Upcoming

- Thursday in Section: finalizing groups and the world's briefest introduction to the available tools
  - Each group must quickly do two important things – pick one of the two projects and pick a snazzy name
- Friday: teams, team organizations, etc.

Questions?