1.264 Lecture 2

System process fundamentals

Today: Find a homework partner.
Next class: Read chapters 4-6. Hand in exercise solution after class

Case study: Demand forecasting, version 1

- Do you have questions on what happened?
- What are your overall reactions to this?
  - Does it seem familiar? Has this happened to you?
  - What related experiences have you had?
- Discussion items
  - List as many errors that were made by this team as you can.
  - What did the team do right?
  - What project management method was used? Was it appropriate?
  - What should they have done to succeed?
- Summary

Case study: Demand forecasting, version 3

- Do you have questions on what happened?
- What are your overall reactions to this?
  - Does it seem familiar? Has this happened to you?
  - What related experiences have you had?
- Discussion items
  - At the 4 month point, what do you, Pat, do? You can have some additional resources; specify those you would like to have.
  - With your suggested actions, will you be able to deliver the system on time, in 11 months? Why or why not?
  - With your suggested actions, how certain will you be at month 8 whether you can deliver on time?
- Summary

Technical fundamentals

Spiral model as basis for development

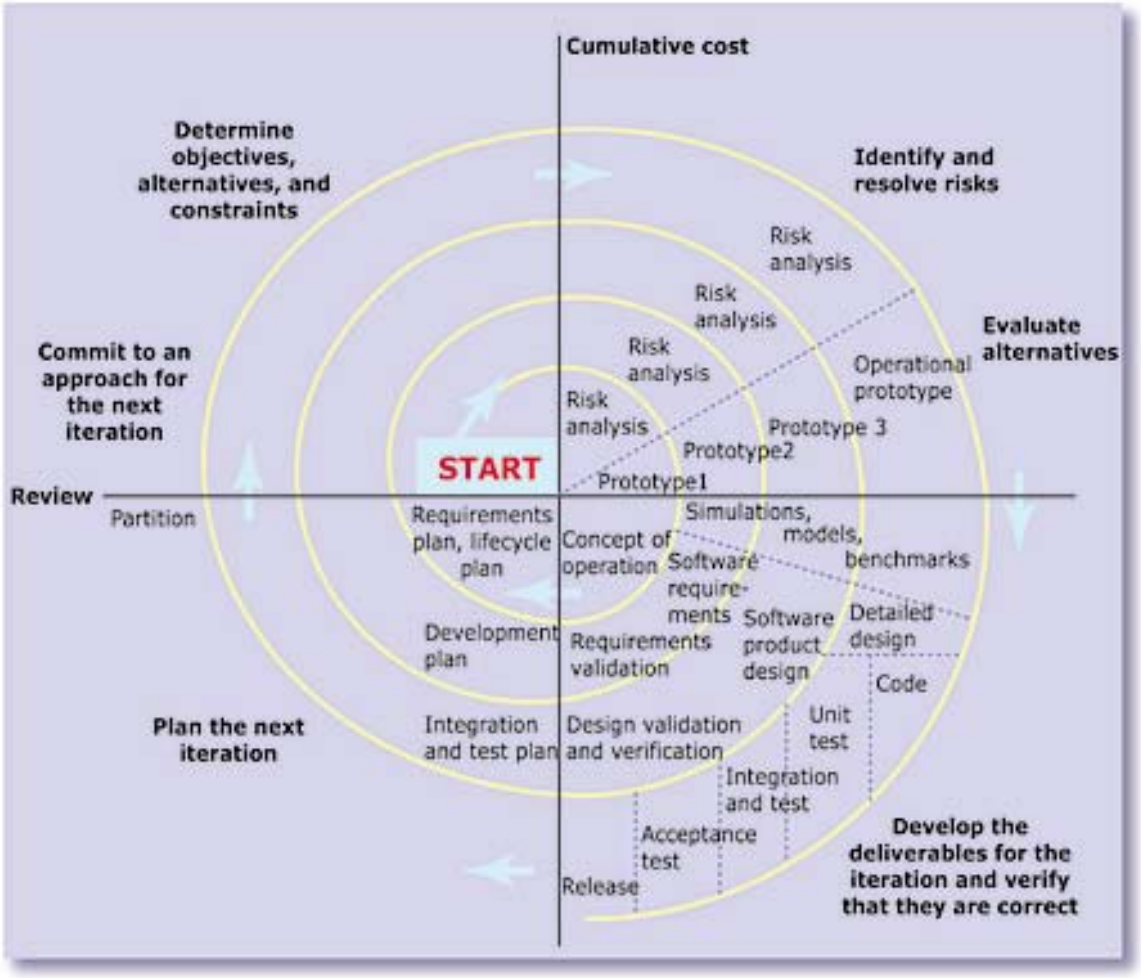

Image by MIT OpenCourseWare.

Process choices

Lifecycle model	Strengths	Weaknesses
Code and fix	None known	Unpredictable, chaotic
Waterfall	Efficient if requirements known. Good for repeated applications.	No feedback or change in process. Cumbersome, likely to fail (2% success DoD)
Rapid prototype	Aligns with client needs	Insufficient structure to deliver production system
Open source	Uses skills of large number of people	Unstructured, almost all efforts fail.
Agile process	Flexible, can be fast	Works best/only on small projects
Spiral model	Manages risks, feedback, well-defined. Works on large projects	Requires skill and discipline

Different process models for 12 month project

- Traditional, chaotic approach:
  - 1 month requirements, left incomplete
  - 1 month design, left incomplete
  - 9 months development, with substantial rework
  - 1 month test/QA (quality assurance), which is insufficient: poor quality, late
- Waterfall, based on past metrics:
  - 3 months requirements
  - 3 months design
  - 3 months development
  - 3 months test/QA, produces system but with limited scope
- Spiral (often called ‘agile’ with ‘sprints’ rather than spirals)
  - 3 spirals, each 4 months:
    - 1 month requirements
    - 1 month design
    - 1 month implementation
    - 1 month test/QA/review

Exercise

- What process would you use?
  - Off-the-shelf accounting system implementation in a middle size company, your 20th one
  - Reducing number of distribution centers significantly in a large company
  - Privatizing bus operations funded by a public transportation agency
  - Revamping your company's marketing strategy
- Take 10 minutes:
  - Recommend a process
  - List top 3 factors or key unknowns to be researched early in the decision

Solution (one of many)

- What process would you use?
  - Off-the-shelf accounting system implementation in a middle size company, your 20th one
    - Waterfall
  - Reducing number of distribution centers significantly in a large company
    - Spiral.
      - 1: identify key issues, risks
      - 2: develop plans based on overall corporate goals
      - 3: refine plans with the field, vendors
  - Privatizing bus operations funded by a public transportation agency
    - Spiral:
      - 1: define procurement process, contract options, suppliers
      - 2: develop plans based on agency goals
      - 3: review and revise plans after discussion with vendors
  - Revamping your company's marketing strategy
    - Rapid prototype: Mock up and assess many ideas

Requirements fundamentals

- Requirements: what should the system do?
  - What we're doing in homework this semester is essentially an extended requirements analysis
    - The first spiral can often be viewed as requirements step
  - Requirements steps
    - Text description of the system: a necessary overview
    - Use cases (UML) to list scenarios
    - Text descriptions of scenarios to give more detail
    - Initial version of user interface/new process and manual
    - Data model (entity-relationship diagram)
      - As complete picture of all data (or objects) in the system as possible. Determines the business rules.
    - Other UML diagrams as needed: state, activity, component
  - These needs are the same whether you are implementing, configuring, modifying or developing a system, business process, ….

Design fundamentals

- Design: how does the system or process work?
  - Data model, complete
  - User interface or process mockup
  - UML diagrams
    - System architecture (components, interfaces, hardware…)
    - Use cases (lists of scenarios), complete
    - Scenarios, as text
    - Class diagrams (for software systems)
      - Extend data models to cover all behaviors in the system
    - Sequence and collaboration models (dataflow diagrams)
      - Dynamic view of multiple flows of data and control in the system
    - State models (state transition diagrams)
      - Dynamic and complete view of the data values and logic
  - These needs are the same whether you are implementing, configuring, modifying or developing a system

Implementation fundamentals

- Requirements and design dictate development success
  - 60% of system defects exist at requirements/design time
  - Cost of correcting errors (relative) at different stages:
    - Requirements: $100
    - Design: $500
    - Implementation/QA: $2,500
    - Operation: $12,500
- Implementation practices: CMMI (capabilities maturity model) –development, acquisition, services
  - Have requirements, design documents, UML, data models
  - Measure team size, system size, defects, effort, schedule
  - Use a defined implementation process: spiral, agile, etc.
  - Integrate and bring system to usable state frequently
  - Perform quality assurance continuously
  - Get mechanics right: version control, documents, reviews

Quality assurance fundamentals

- QA starts at project initiation
  - Requirements scrubbing and reviews
  - Design reviews
  - Implementation inspections and walk-throughs
- Testing
  - System tests find 10-60% of defects
  - Reviews and inspections find 60-90%: more critical than testing
  - This holds for software, hardware, process changes, ….
- Error prone components: identify and re-do
  - 57% of errors in 7% of software modules (IBM surveys)
  - Similar numbers for non-software projects
  - Often one “god” component that implements all the logic is very complex and has many errors
    - Indicates system was not decomposed properly into modules or cooperative roles

Risk management

- **Risk management**
  - **Spiral model is all about managing risk**
    - **First spiral focuses on riskiest areas: requirements, design, implementation in most difficult areas**
    - **First spiral assessment then allows substantial revision of requirements and design, based on having tried to do it once already**
    - **Second spiral has much cleaner requirements, design, and can usually produce a system close to what's needed**
    - **Third spiral cleans up issues, makes system manageable and stable**
  - **Keep a top 10 risks list**
    - **Assess probability of risk, magnitude of loss if it occurs**
    - **Rank and manage the list frequently (often weekly)**

Summary

- Project definition and development process is time consuming and labor intensive
  - There are massive pressures to do this quickly
- The seemingly straightforward, but deceptively difficult, part of this process is to clearly understand and specify the requirements the project must satisfy
  - Because of the cumulative nature of the project process, mistakes made in early stages but only identified at a later stage result in major delays and cost increases
  - The spiral model, based on requirements, UML, data and class diagrams is used to manage these risks
    - Other agile models can also be used
  - "Lord Krishna said, you and I have been reborn many times. I remember them but you do not." -Bhagavad Gita

1.264J / ESD.264J Database, Internet, and Systems Integration Technologies
Fall 2013