System Development Life Cycle

Why do systems fail?

* Poor requirement gathering
* Insufficient analysis
* Poor management

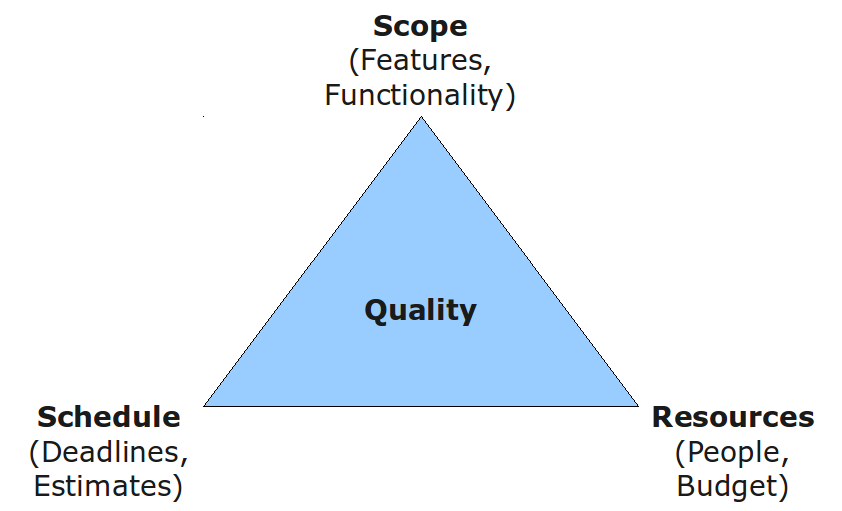
**Phases of SDLC**

* Planning
  + Project initiation
    - Develop a system request
    - Conduct feasibility analysis
  + Project Management
    - Develop a work plan
    - Staff the project
    - Control and direct the project
  + Why should we build that system?
* Analysis
  + Develop analysis strategy
  + Gather requirements
  + Develop a system proposal
  + What should the system do for us? Where and when will it be used?
* Design
  + Develop a design strategy
  + Design architecture and interfaces
  + Develop databases and file specifications
  + Develop the program design
  + How will we build the system?
* Implementation
  + Construct System
  + Install the system
  + Establish a support plan
  + Implement training plan for users
  + Build the system!

Characteristics of SDLC

* Each step leads to specific deliverables
* Phases are interactive
* **What is a deliverable?**
  + Proof that you have completed the current phase and you can proceed to the next.
  + Examples
    - Planning -> Project plan
    - Analysis -> System Proposal
    - Design -> System specifications
    - Implementation -> New system and maintenance plan

**Project Management Triangle**



**Methodology**

* Formalized approach to a series of steps
* Well known methodologies include waterfall development, agile, scrum, etc.

**Methodology Categories**

* Predictive, structured development
  + Projects move methodically from one to the next step
  + Very controlled approach
  + A step is finished before the next begins
  + Recommended for high risk systems
    - Waterfall
      * A: Identifies system requirements long before programming begins and minimizes changes
      * D: Design must be complete before implementation begins. Long time lapses
    - Parallel
      * Design for the project is divided into subprojects (Pilot projects)
      * They are all delivered one at a time
      * Once all projects are delivered they are integrated together to deliver the system.
* Rapid application development (RAD)
  + Phased
  + Prototyping
    - Create a “quick and dirty” look at the system
    - Minimal features
    - Show it to others and get their feedback
    - Re-analyze, re-design, re-implement
    - A: Quick feedback from users
    - D: Users constantly not happy, when are you done? May sacrifice careful analysis
  + Throwaway prototyping
    - When you don’t really understand what others want.
    - Thorough analysis, move to design but do a prototype to make sure you understand.
    - A: feedback on difficult misunderstood parts
    - D: Extra time required.
* Adaptive, agile development
  + 4 core values: Communication, simplicity, feedback, courage.
  + Rapid feedback between developers
  + Simple, rapid iterations of application
  + **Emphasizes:**
    - **Individuals and interactions** over processes and tools
    - **Working software** over comprehensive documentation
    - **Customer collaboration** over contract negotiation
    - **Responding to change** over following a plan
  + Extreme programming
  + Scrum
    - 15 minutes daily meeting where each person says
      * What they’ve done
      * What they’re going to do,
      * Any problems/coordination/help
    - Scrum projects work as a series of sprints.
    - A **sprint** is a duration of time no longer than 30 days
      * A sprint has a specific goal
      * Sprint team has final say in estimating and determining what they can accomplish

Factors in selecting the proper methodology

* Clarity of user requirements
* Familiarity with technology
* System complexity
* System reliability
* Short time schedules
* Schedule visibility

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Usefulness for** | **Waterfall** | **Parallel** | **Phased** | **Prototyping** | **Throwaway Prototyping** | **Agile** |
| Unclear user requirements | Poor | Poor | Good | Excellent | Excellent | Excellent |
| Unfamiliar technology | Poor | Poor | Good | Poor | Excellent | Excellent |
| Complex systems | Good | Good | Good | Poor | Excellent | Poor |
| Reliable systems | Good | Good | Good | Poor | Excellent | Good |
| Short time schedule | Poor | Good | Excellent | Excellent | Good | Excellent |
| Schedule visibility | Poor | Poor | Excellent | Excellent | Good | Good |