SOFTWARE PROJECT MANAGEMENT

LECTURE # 3

SOFTWARE DEVELOPMENT FUNDAMENTALS

29th September, 2011

Engr. Ali Javed

Contact Information

- Instructor: Engr. Ali Javed
  Lecturer
  Department of Software Engineering
  U.E.T Taxila

  - Email: ali.javed@uettaxila.edu.pk
  - Contact No: +92-51-9047592
  - Office hours:
    - **Monday, 11:00 - 01:00, Office # 7**

Course Information

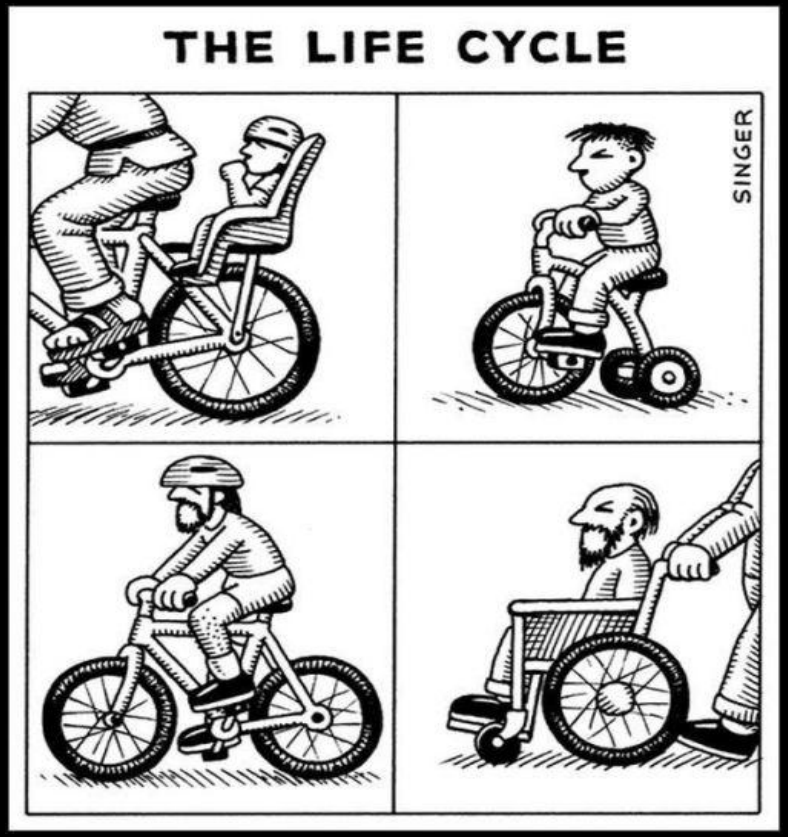
- **Course Name: Software Project Management**
- **Course Code: SE-401**
- **CMS Link:** http://web.uettaxila.edu.pk/CMS/AUT2011/seSPMbs/index.asp

Software Development Fundamentals

- Project Lifecycle
- SDLC

Project Lifecycle

- Project Phases
- Characteristics of Project Phases
- Characteristics of Project Lifecycle
- Fast Tracking
- SDLC
- Planning



Project Phases

- All projects are divided into different phases
- Each project phase is marked by completion of one or more deliverables.
- All Project phases together are known as the ***project life cycle.***
- This view of software development is referred to as the **software development life cycle.**

Characteristics of Project Phases

- A *deliverable* *is a tangible, verifiable work product such as a feasibility study,* a detail design, or a working prototype.
- The conclusion of a project phase is generally marked by a review of both key deliverables and project performance to date, to determine
  - ✓ If the project should continue into its next phase
  - ✓ Detect and correct errors cost effectively.
- These phase-end reviews are often called *phase exits, stage gates, or kill points.*

Fast Tracking

- A subsequent phase is sometimes begun prior to approval of the previous phase deliverables when the risks involved are deemed acceptable.

- This practice of overlapping phases is often called ***fast tracking.***

What Project life cycles define?

- What technical work should be done in each phase
- Who should be involved in each phase (e.g., implementers who need to be involved with requirements and design).
- Project life-cycle descriptions may be very general or very detailed.
- Highly detailed descriptions may have numerous forms, charts, checklists to provide structure and consistency. Such detailed approaches are often called *project management methodologies.*

Software Development Lifecycle

- Software development, just like most other activities, has a beginning, middle and an end.
- The end of one development activity is sometimes perceived as being linked to the beginning of a new development activity thus producing a cycle of beginning-middle-end, link, and so forth.
- Deliverables from the preceding phase are usually approved before work starts on the next phase.
- This view of software development is referred to as the software development life cycle.

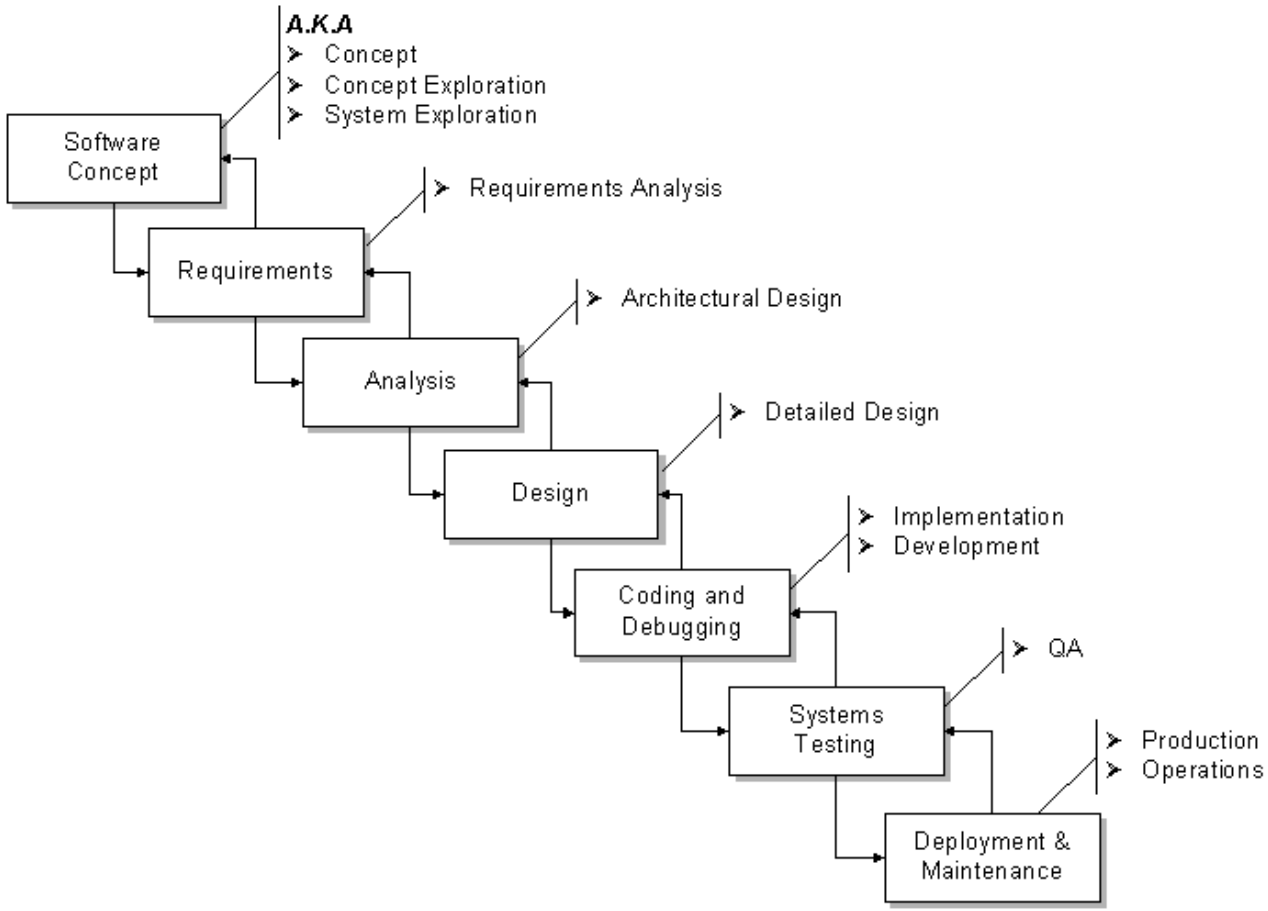
Lifecycle Planning

- a.k.a. Lifecycle Management or SDLC
- Greatly influences your chance of success
- Not choosing a lifecycle is a bad option
- Different projects require different approaches
- You do not need to know all models by name
- You should know how that if given a certain scenario what sort of SDLC would be appropriate

Planning

- Characteristics leads to Poor Planning
  - ✓ Inexperienced management
  - ✓ Ineffective change control
  - ✓ Political Pressures
  - ✓ Unrealistic Deadlines

Project Phases



Concept Elaboration

- The “Why” phase
- Not a “mandatory formal” phase
  - ✓ Sometimes called the “pre-project” phase
- Collecting project ideas
  - ✓ Then the “funneling” process
- Project Justification
- Initially rough Estimations

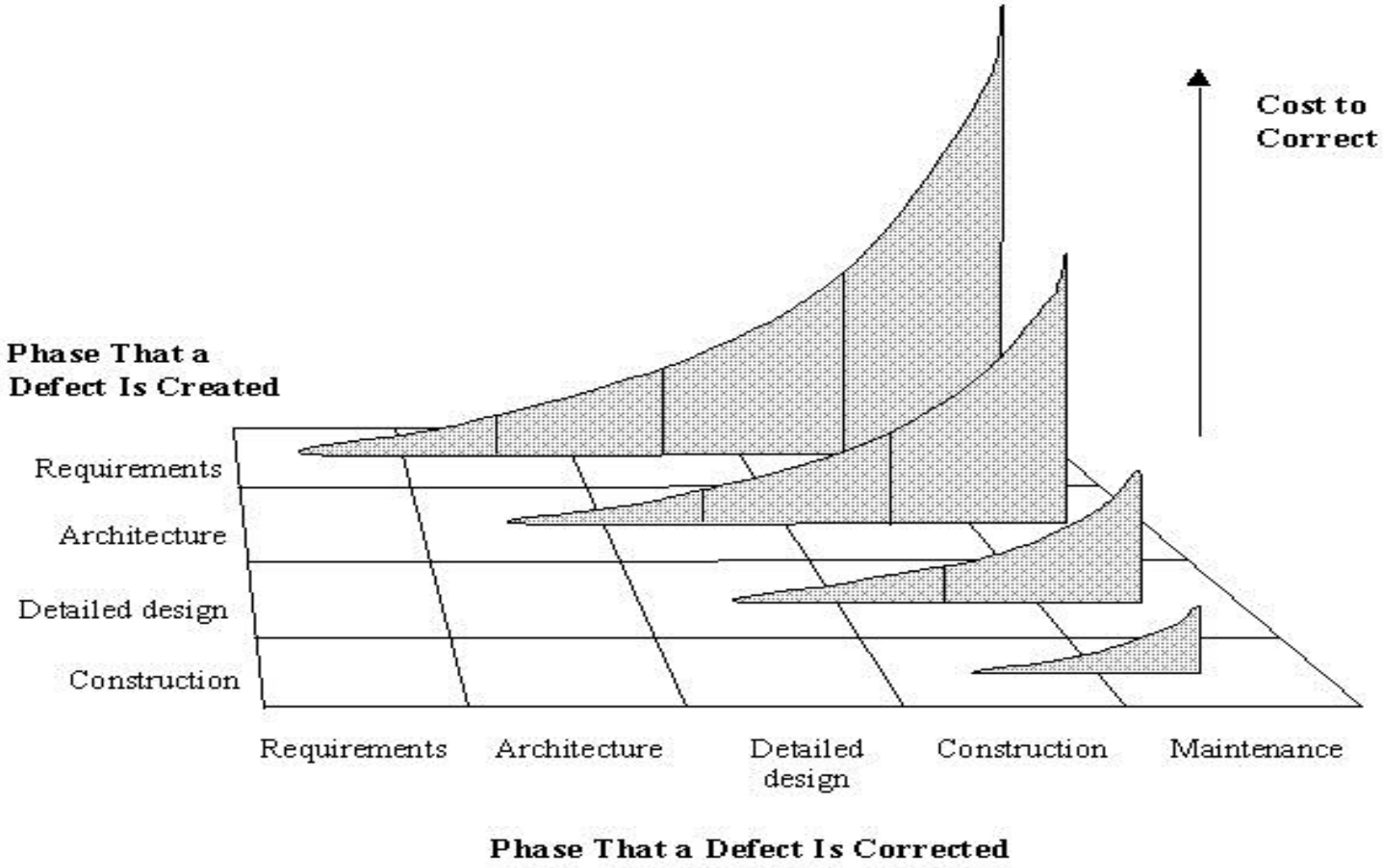
Concept Elaboration

- Possibly includes Procurement Management:
  - ✓ RFP Process [4]
  - ✓ Vendor selection
  - ✓ Contract management[6]
- Gathering the initial team
  - ✓ Including PM if not already on-board
- Identify the project sponsor
  - ✓ Primary contact for approval and decision making
- Potential Phase Outputs:
  - ✓ Concept Document, Product Description, Project Charter ,SOW (Statement of work) [5]

Requirements

- The "What" phase
- Inputs: SOW, Proposal
- Outputs:
  - ✓ Requirements Document (RD)
    - a.k.a. Requirements Specification Document (RSD), Software Requirements Specification (SRS)
  - ✓ 1st Project Baseline
  - ✓ Software Project Management Plan (SPMP)
  - ✓ Requirements Approval & Sign-Off
    - Your most difficult task in this phase

Why are Requirements so Important?



Requirements

- Characteristics & Issues
  - ✓ Conflict of interest: developer vs. customer
  - ✓ Potential tug-of-war:
    - Disagreement on Features & Estimates especially in fixed-price contracts
  - ✓ Frequent requirements changes
  - ✓ Achieving sign-off

Requirement Types

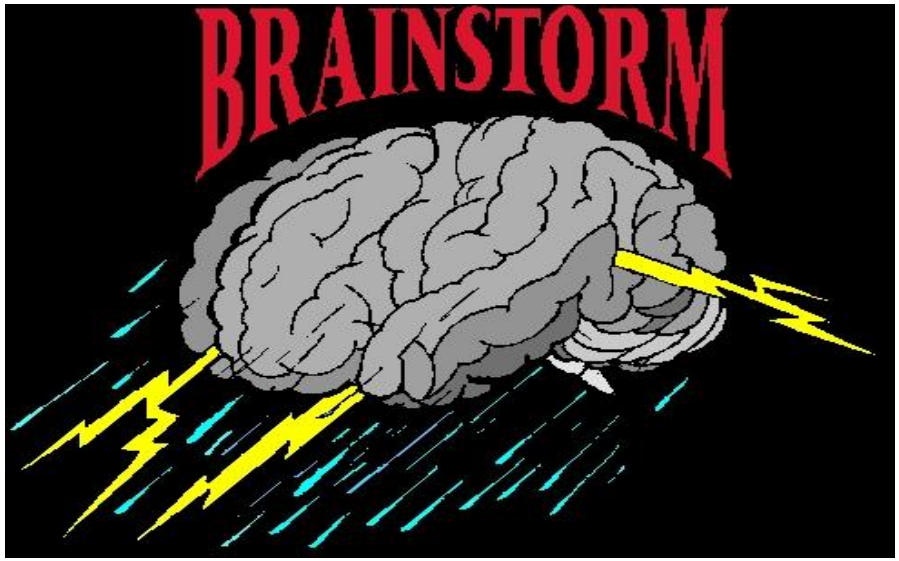
- **Functional** (behavioral)
  - ✓ Features and capabilities
- **Non-functional** (a.k.a. "technical") (everything else)
  - ✓ Usability
    - Human factors, help, documentation
  - ✓ Reliability
    - Failure rates, recoverability, availability
  - ✓ Performance
    - Response times, throughput, resource usage
  - ✓ Supportability
    - Maintainability, portability
  - ✓ Operations: systems management, installation
  - ✓ Interface: integration with other systems

Requirements

- Backward-looking vs. Forward-looking
  - ✓ Backward: address issues with previous version
  - ✓ Forward: Anticipating future needs of customers
- Must be prioritized
  - Must-have
  - Should-have
  - Could-have (Nice-to-have: NTH)
- Must be approved

Early Phase Meetings

- Project Kickoff Meeting [7]



- Project Brainstorming Meeting
  - ✓ Clarify goals, scope, assumptions
  - ✓ Refine estimates
- WBS Meeting

Analysis & Design

- The “How” Phases
- a.k.a. Top-level design & detailed design
- Inputs: Requirements Document
- Outputs:
  - ✓ Functional Specification (Abstract level design specification)
  - ✓ E-R Diagrams
  - ✓ Data Flow Diagrams
  - ✓ Detailed Design Document
  - ✓ Object Diagrams
  - ✓ Data Model
  - ✓ Updated Plan (improved estimates; new baseline)
- Approval & Sign-off

Development

- The “Do It” phase
- Coding & Unit testing
- Often overlaps Design & Integration phases
  - ✓ To shorten the overall schedule
  - ✓ PM needs to coordinate this
- Other concurrent activities
  - ✓ Design completion
  - ✓ Integration begins
  - ✓ Unit testing of individual components
  - ✓ Test bed setup (environment and tools)
  - ✓ Project plans updated
  - ✓ Scope and Risk Management conducted

Development

- Characteristics & Issues
  - ✓ Pressure increases
  - ✓ Last-minute changes
  - ✓ Team coordination (esp. in large projects)
  - ✓ Communication overhead

Integration & Test

- Evolves from Development Phase
- Often done as 2 parallel phases
  - ✓ Partial integration & initial test
- Starts with integration of modules
- Progressively add more components
- Integration primarily a programmer task
- Test primarily a QA team task

Integration & Test

- Integration:
  - ✓ Top-down: Core functionality first, empty shells for incomplete routines (stubs)
  - ✓ Bottom up: gradually bind low-level modules
  - ✓ Prefer top-down generally

- Tests
  - ✓ Integration testing
  - ✓ Black & White-box testing
  - ✓ Load & Stress testing
  - ✓ Alpha & Beta testing
  - ✓ Acceptance testing

Integration & Test

- Characteristics & Issues
  - ✓ Increased pressure
  - ✓ Overtime
  - ✓ Customer conflicts over features
  - ✓ Frustration over last-minute failures
  - ✓ Budget overruns
  - ✓ Motivation problems (such as burnout)
  - ✓ Difficulty in customer acceptance

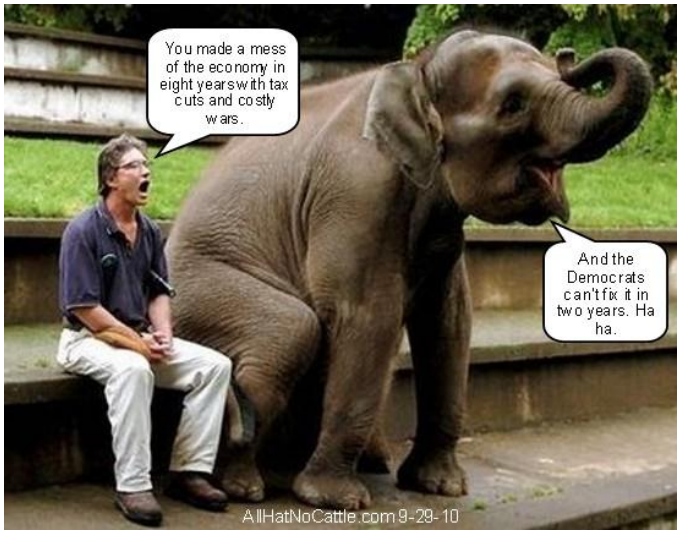
Deployment & Maintenance

- Installation depends on system type
- Maintenance
  - ✓ Fix defects
  - ✓ Add new features
  - ✓ Improve performance

- Configuration control is very important here
- Documents need to be maintained also
- Sometimes a single team maintains multiple products
- Deployment typically in your project plan, maintenance not

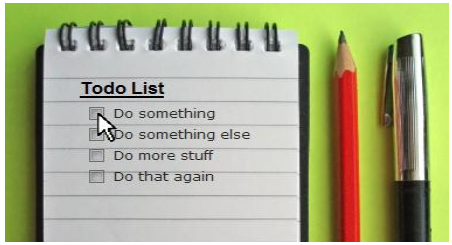
Deployment & Maintenance

- Characteristics & Issues
  - ✓ Lack of enthusiasm
  - ✓ Pressure for quick fixes
  - ✓ Insufficient budget
  - ✓ Too many patches
  - ✓ Personnel turnover
  - ✓ Regression testing is critical



Tracking

- Management Level
  - ✓ Tasks Lists



  - ✓ Status meetings and Reports
  - ✓ Milestones

  - ✓ Reviews

  - ✓ Budget Reports
  - ✓ Management by Walking Around

Rule of Thumb

Most software projects (something like 80 percent) are delivered late, substantially over budget, and incomplete. The more effort you put into managing your project, the more you increase your chances of success.

References

1. Software Engineering by Roger Pressman
2. Software Engineering by Ian Sommerville
3. http://forum.onestoptesting.com/forum_posts.asp?TID=2766
4. http://en.wikipedia.org/wiki/Request_for_proposal
5. http://en.wikipedia.org/wiki/Statement_of_work
6. http://en.wikipedia.org/wiki/Contract_management
7. http://en.wikipedia.org/wiki/Kickoff_meeting

For any query Feel Free to ask