

Software Project Planning

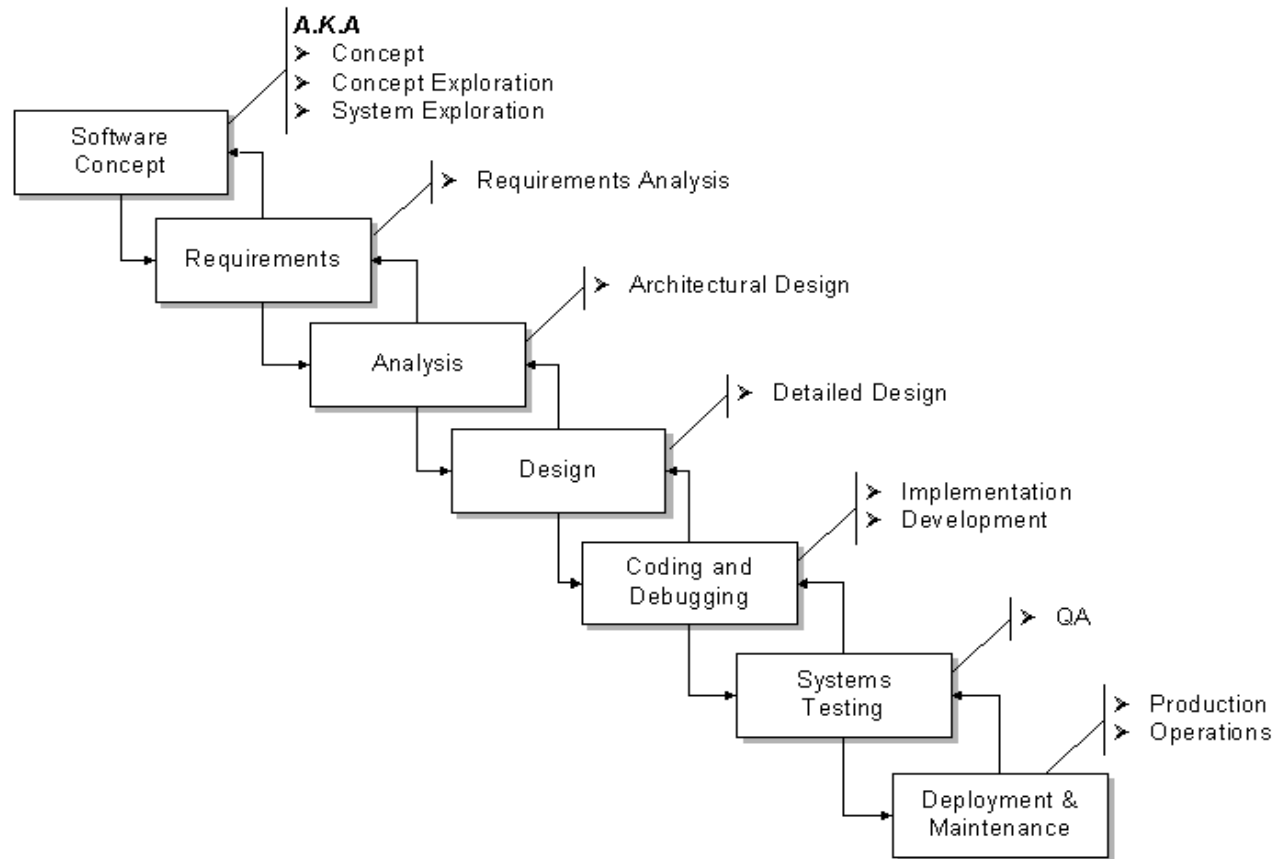
Today

- 1. Phases in Detail
 - Step-by-step of typical software project
- 2. Lifecycle Planning
- 3. Project plans

Review

- PMI Fundamentals
- PMI Processes
- Project Organization
 - Functional, Project, Matrix Orgs.
- Initial documents
 - Statement of Work (SOW)
 - Project Charter

Project Phases



Time Allocation by Phase

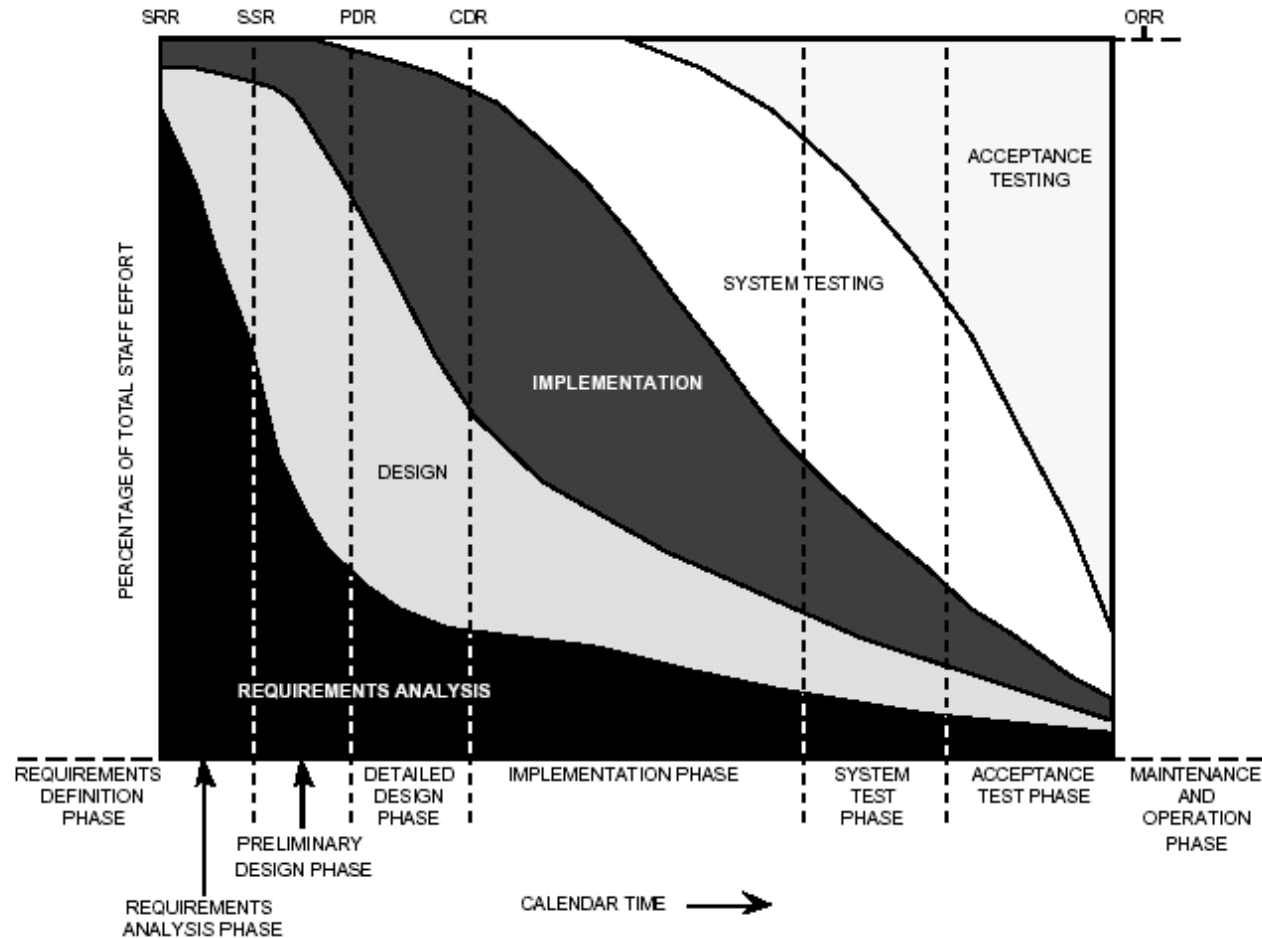
- Remember the 40-20-40 Rule
 - Specification-Implementation-Test

	Planning	Code & Unit Test	Integration & Test
Commercial DP	25%	40%	35%
Internet Systems	55%	15%	30%
Real-time Systems	35%	25%	40%
Defense Systems	40%	20%	40%

Time Allocation by Phase

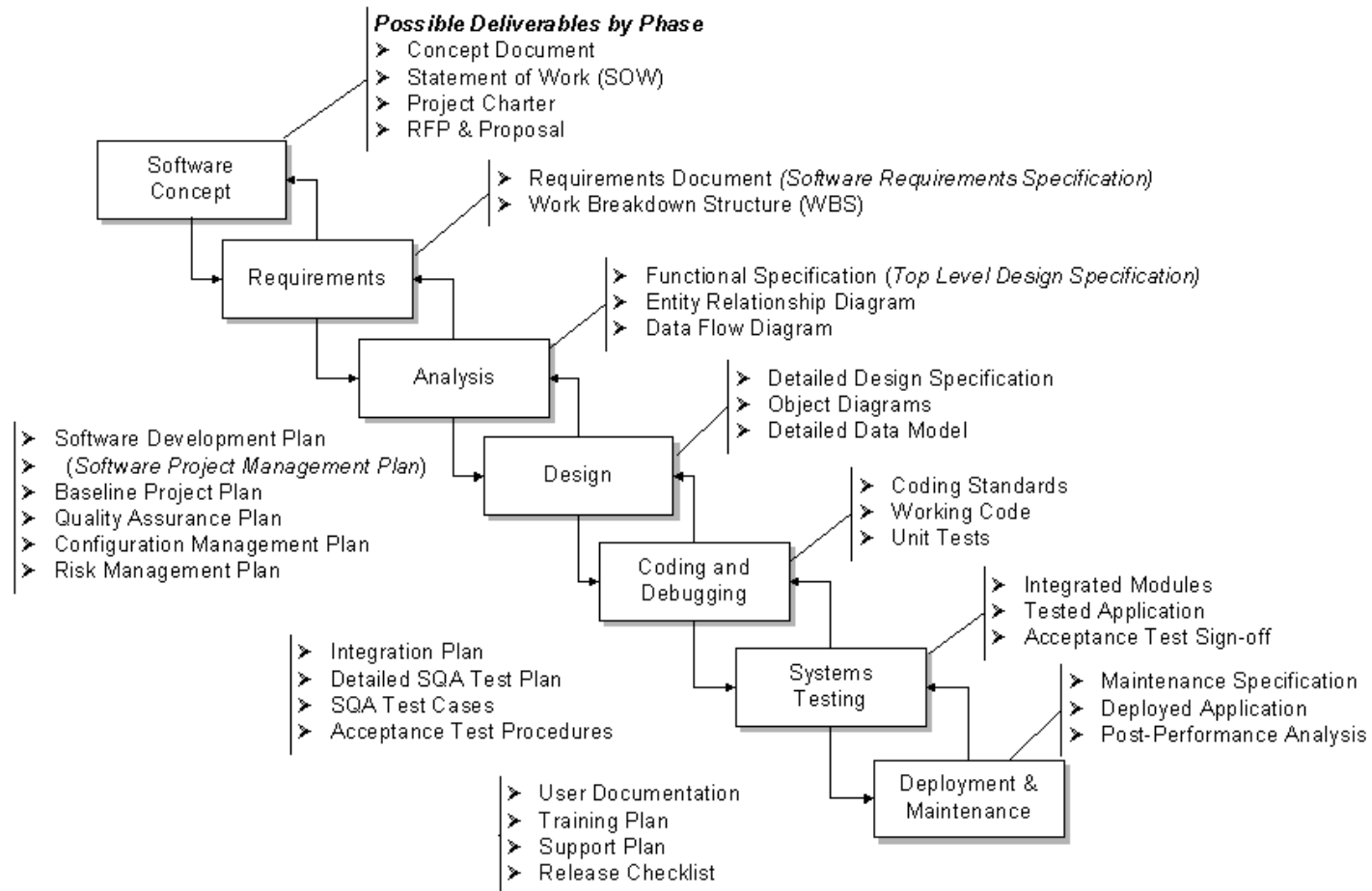
Activity	Small Project (2.5K LOC)	Large Project (500K LOC)
Analysis	10%	30%
Design	20%	20%
Code	25%	10%
Unit Test	20%	5%
Integration	15%	20%
System test	10%	15%

Activities by % of Total Effort



NASA's "Manager's Handbook for Software Development"

Deliverables by Phase



Concept Exploration

- The “Why” phase
- Not a “mandatory formal” phase
- Collecting and funneling project ideas
- Project Justification
 - ROI
 - Cost-benefit analysis
 - Project Portfolio Matrix
- Initial planning and estimates

Concept Exploration ..2

- Possibly includes Procurement Management:
 - RFP Process
 - Vendor selection
 - Contract management
- Gathering the initial team
 - Including PM if not already on-board
- Identify the project sponsor
 - Primary contact for approval and decision making

Concept Exploration ..3

- Potential Phase Outputs:
 - Concept Document, Product Description, Proposal, SOW, Project Charter
- Characteristics & Issues
 - Lack of full commitment and leadership

Concept Exploration ..4

- Some frustrations:
 - Management getting rough estimates from development
 - Development not getting specifics from customer
 - Finding a balanced team
- Budget sign-off may be your first major task
- Achieved via:
 - Good concept document or equivalent
 - Demonstration of clear need (justification)
 - Initial estimates

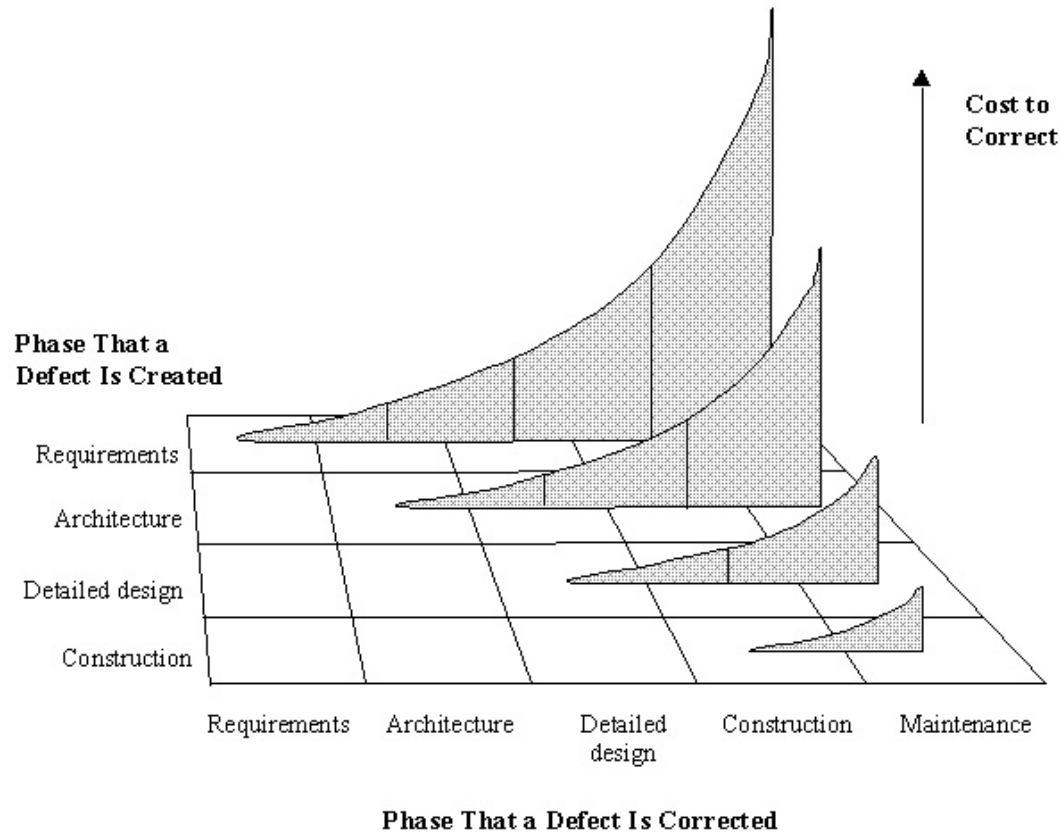
Requirements

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

Requirements

- Perhaps most important & difficult phase
- Shortchanging it is a ‘classic mistake’
- Can begin with a Project Kickoff Meeting
- Can end with a Software Requirements Review (SRR)
 - For Sponsor and/or customer(s) approval

Why are Requirements so Important?



Copyright 1998 Steven C. McConnell. Reprinted with permission from *Software Project Survival Guide* (Microsoft Press, 1998).

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
- Project planning occurs in parallel

Requirements

- Requirements are capabilities and condition to which the system – more broadly, the project – must conform

Two Types of Requirements

- **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, internationalization
 - Operations: systems management, installation
 - Interface: integration with other systems
 - Other: legal, packaging, hardware

Requirements

- Other ways of categorizing
 - Go-Ahead vs. Catch-up
 - Relative to competition
 - 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
- Project Brainstorming Meeting
 - Clarify goals, scope, assumptions
 - Refine estimates
- WBS Meeting

Analysis & Design

- The “How” Phases
- Inputs: Requirements Document
- Outputs:
 - Functional Specification
 - Detailed Design Document
 - User Interface Specification
 - Data Model
 - Prototype (can also be done with requirements)
 - Updated Plan (improved estimates; new baseline)

Analysis & Design

- a.k.a. Top-level design & detailed design
- Continues process from Requirements Document
- Ends with Critical Design Review (CDR)
 - Formal sign-off
 - Can also include earlier Preliminary Design Review (PDR) for high level design

Analysis & Design

- Characteristics & Issues
 - Enthusiasm via momentum
 - Team structure and assignments finalized
 - Delays due to requirements changes, new information or late ideas
 - Issues around personnel responsibilities
 - Unfeasible requirements (technical complexity)
 - Resource Issues
 - Including inter-project contention

Development

- The “Do It” phase
- Coding & Unit testing
- Often overlaps Design & Integration phases
 - To shorten the overall schedule
 - PM needs to coordinate this

Development

- 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
 - Pressure increases
 - Staffing at highest levels
 - Often a “heads-down” operation
- Issues
 - Last-minute changes
 - Team coordination (especially in large projects)
 - Communication overhead
 - Management of sub-contractors

Integration & Test

- Evolves from Development Phase
- Often done as two parallel phases
 - Partial integration & initial test
- Starts with integration of modules
- An initial, incomplete version constructed
- Progressively add more components

Integration & Test ..2

- Integration primarily a programmer task
- Test primarily a QA team task
- Integration:
 - Top-down: Core functionality first, empty shells for incomplete routines (stubs)
 - Bottom up: gradually bind low-level modules
 - Prefer top-down generally

Integration & Test ..3

- Tests
 - Integration testing
 - Black & White-box testing
 - Load & Stress testing
 - Alpha & Beta testing
 - Acceptance testing
- Other activities
 - Final budgeting; risk management; training; installation preparation; team reduced

Integration & Test ..4

- 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
 - Especially true for fixed-price contracts

Deployment & Maintenance

- Installation depends on system type
 - Web-based, CD-ROM, in-house, etc.
- Migration strategy
- How to get customers up on the system
 - Parallel operation
- Deployment typically in your project plan, maintenance not

Deployment & Maintenance .2

- 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 & Maintenance .3

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

Lifecycle Planning

- a.k.a. Lifecycle Management or SDLC
- Greatly influences your chance of success
- Not choosing a lifecycle is a bad option
- Three primary lifecycle model components
 - Phases and their order
 - Intermediate products of each phase
 - Reviews used in each phase

Lifecycle Planning

- Different projects require different approaches
- You do not need to know all models by name
- You should know which SDLC would be appropriate in various situations
- There are more than covered here
- A lifecycle is not a design, modeling or diagramming technique
 - The same technique (UML, DFD, etc) can be used with multiple lifecycles

Pure Waterfall

- The “granddaddy” of models
- Linear sequence of phases
 - “Pure” model: no phases overlap
- Document driven
- All planning done up-front

Waterfall Risk

- Why does the waterfall model “invite risk”?
- Integration and testing occur at the end
 - Often customer’s first chance to “see” the program

Pure Waterfall

- Works well for projects with
 - Stable product definition
 - Well-understood technologies
 - Quality constraints stronger than cost & schedule
 - Technically weak staff
 - Provides structure
 - Good for overseas projects

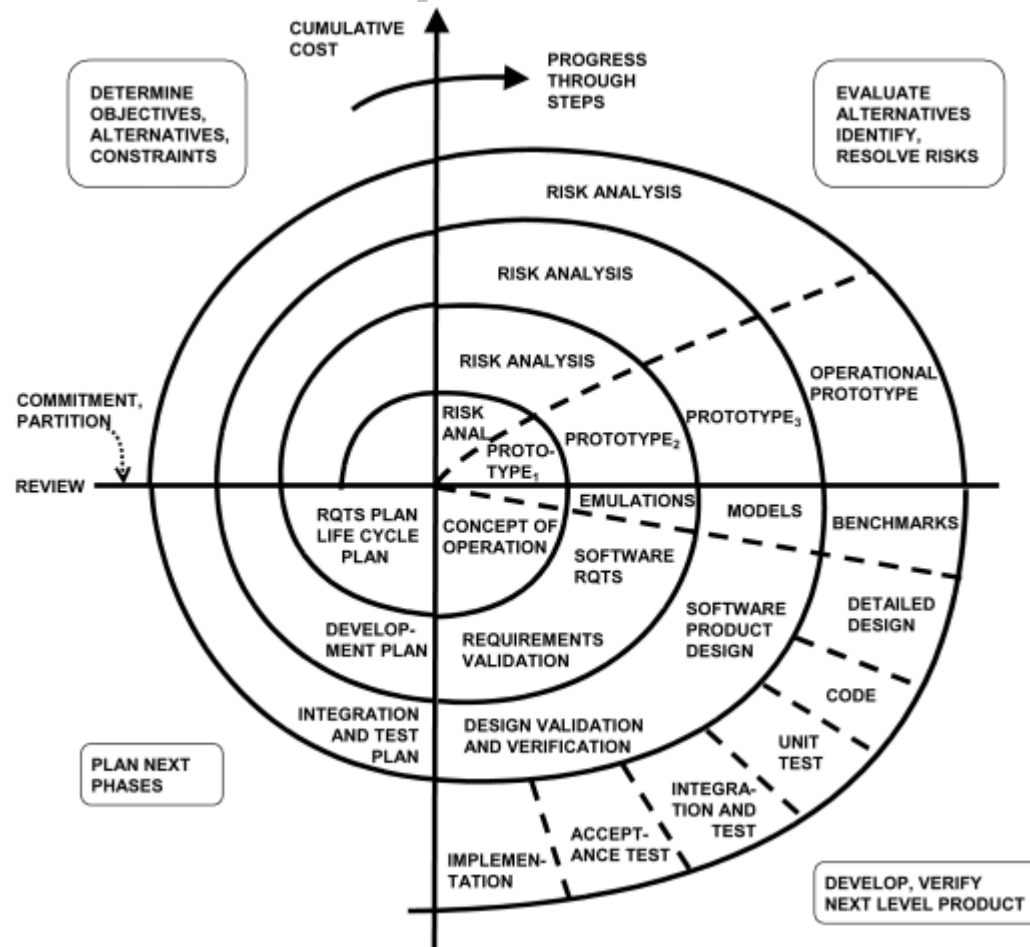
Pure Waterfall

- Disadvantages
 - Not flexible
 - Rigid march from start to finish
 - Difficult to fully define requirements up front
 - Can produce excessive documentation
 - Few visible signs of progress until the end

Code-and-Fix

- “Code-like-Hell”
- Specification (maybe), Code (yes), Release (maybe)
- Advantages
 - No overhead
 - Requires little expertise
- Disadvantages
 - No process, quality control, etc.
 - Highly risky
- Suitable for prototypes or throwaways

Spiral



Spiral

- Emphasizes risk analysis & management in each phase
- A Series of Mini-projects
- Each addresses a set of “risks”
 - Start small, explore risks, prototype, plan, repeat
- Early iterations are “cheapest”
- Number of spirals is variable
 - Last set of steps are waterfall-like

Spiral

- Advantages
 - Can be combined with other models
 - As costs increase, risks decrease
 - Risk orientation provides early warning
- Disadvantages
 - More complex
 - Requires more management

Modified Waterfall – Sashimi

- Overlapping phases
- Advantages
 - Reduces overall schedule
 - Reduces documentation
 - Works well if personnel continuity
- Disadvantages
 - Milestones more ambiguous
 - Progress tracking more difficult
 - Communication can be more difficult

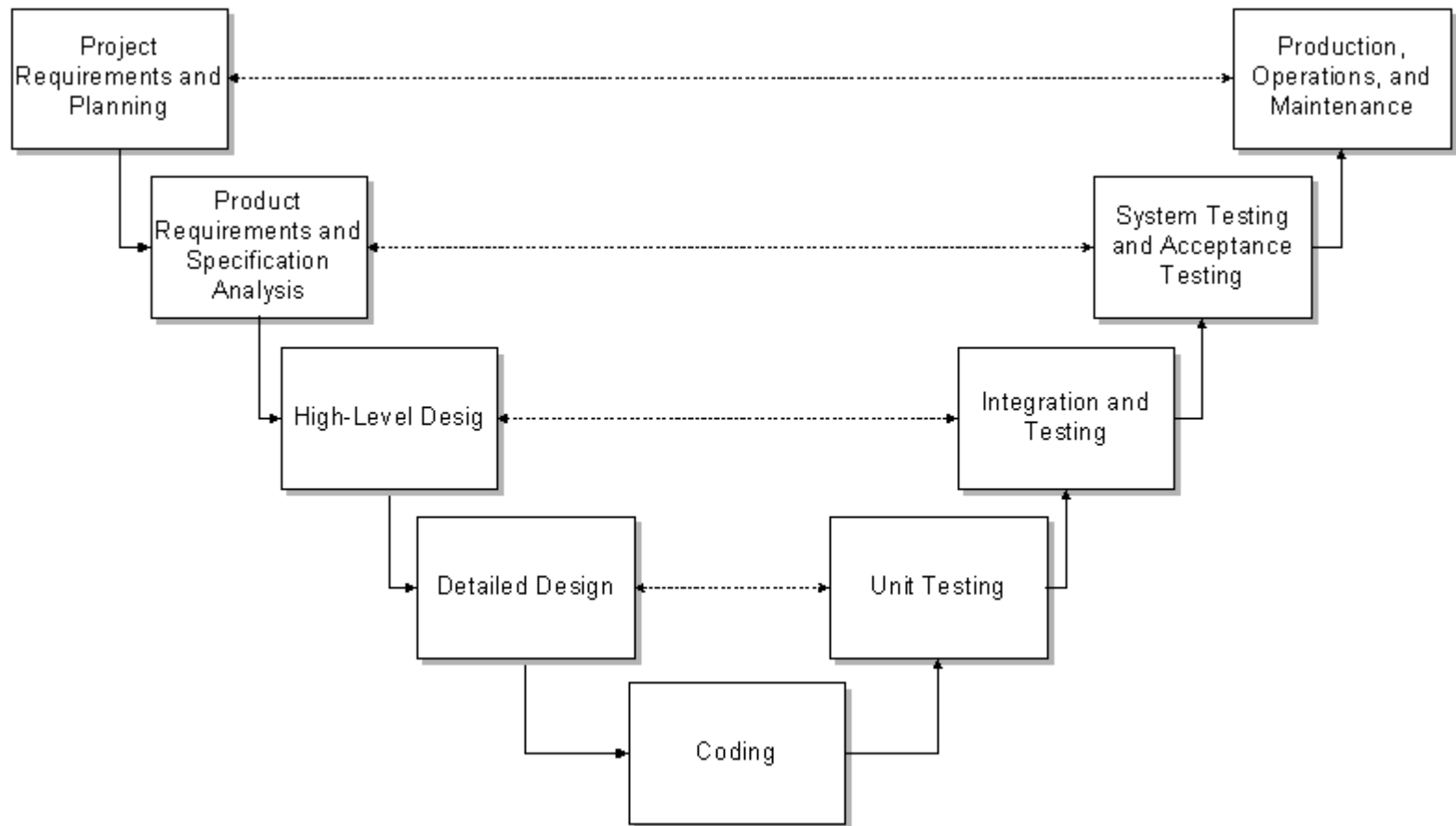
Evolutionary Prototyping

- Design most prominent parts first
 - Usually via a visual prototype
- Good for situations with:
 - Rapidly changing requirements
 - Non-committal customer
 - Vague problem domain
- Provides steady, visible progress
- Disadvantages
 - Time estimation is difficult
 - Project completion date may be unknown
 - An excuse to do “code-and-fix”

Staged Delivery

- Waterfall steps through architectural design
- Then detailed design, code, test, deliver in stages
- Advantages
 - Customers get product much sooner
 - Tangible signs of progress sooner
 - Problems discovered earlier
 - Increases flexibility
 - Reduces: status reporting overhead & estimation error
- Disadvantages
 - Requires more planning (for you, the PM)
 - More releases increase effort (and possible feature creep)
- How's this differ from Evolutionary Prototyping?

V Process Model



V Process Model

- Designed for testability
 - Emphasizes Verification & Validation
- Variation of Waterfall
- Strengths
 - Encourages V&V at all phases
- Weaknesses
 - Does not handle iterations
 - Changes can be more difficult to handle
- Good choice for systems that require high reliability such as patient control systems

RAD

- Rapid Application Development
- Popular in the 80's
 - 1. Joint Requirements Planning (JRP)
 - 2. Joint Application Design (JAD)
 - 3. Construction
 - Heavy use of tools: code generators
 - Time-boxed; many prototypes
 - 4. Cutover
- Good for systems with extensive user input available

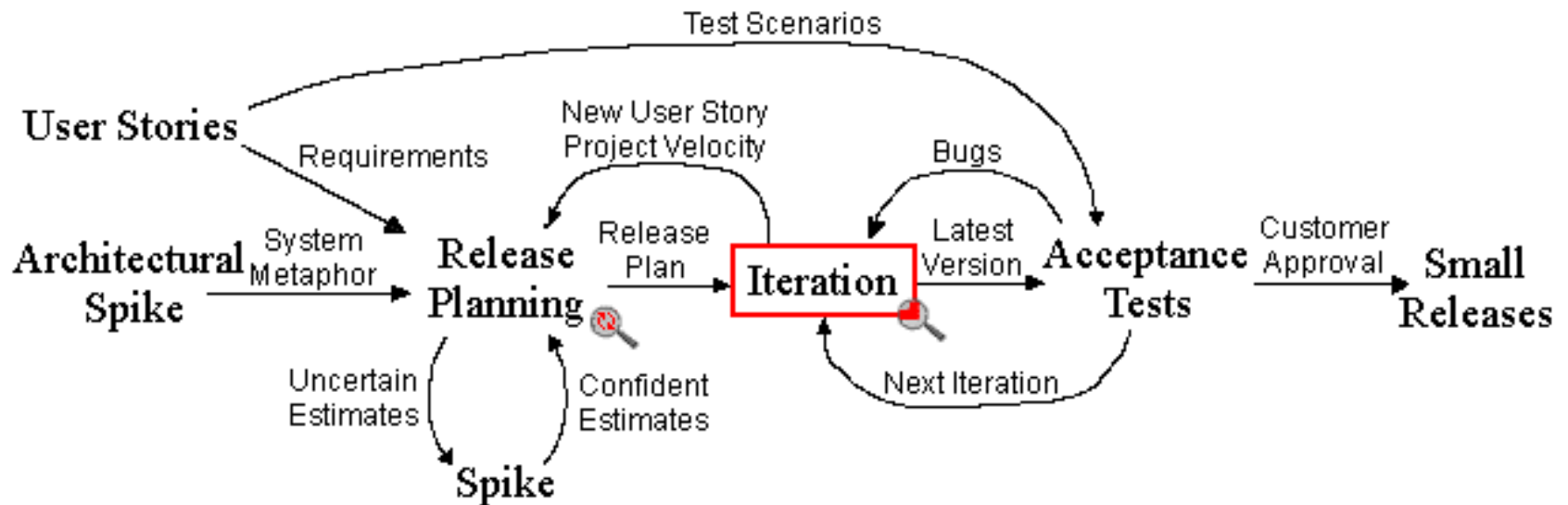
COTS

- Commercial Off-The-Shelf software
- Build-vs.-buy decision
- Advantages
 - Available immediately
 - Potentially lower cost
- Disadvantages
 - Not as tailored to your requirements
- Remember: custom software rarely meets its ideal (so compare that reality to COTS option)

XP: eXtreme Programming

- Not a Microsoft product
- Part of movement called “Agile Development”
- A “Lightweight” methodology
- A bit counter-culture
- Currently in vogue
- Motto: “Embrace Change”
- Highly Incremental / Iterative

eXtreme Programming



eXtreme Programming

- Suitable for small groups
- Attempts to minimize unnecessary work
- Uses an “on-site” customer
- Small releases
- Pair programming
- Refactoring
- Stories as requirements
- You want good developers if you use this

Other “Agile” Methodologies

- Agile here means “lite”, reduced docs, highly iterative
- Agile Software Development
 - Alliance, their “manifesto”, their book
- SCRUM
 - Features 30-day “Sprint” cycles
- Feature Driven Development (FDD)
 - XP with more emphasis on docs and process

Other “Agile” Methodologies

- Adaptive Software Development (ASD)
 - Book, site
- Dynamic System Development Method (DSDM)
 - Popular in Europe
- Homegrown: developers often hide their “agile adventures” from management

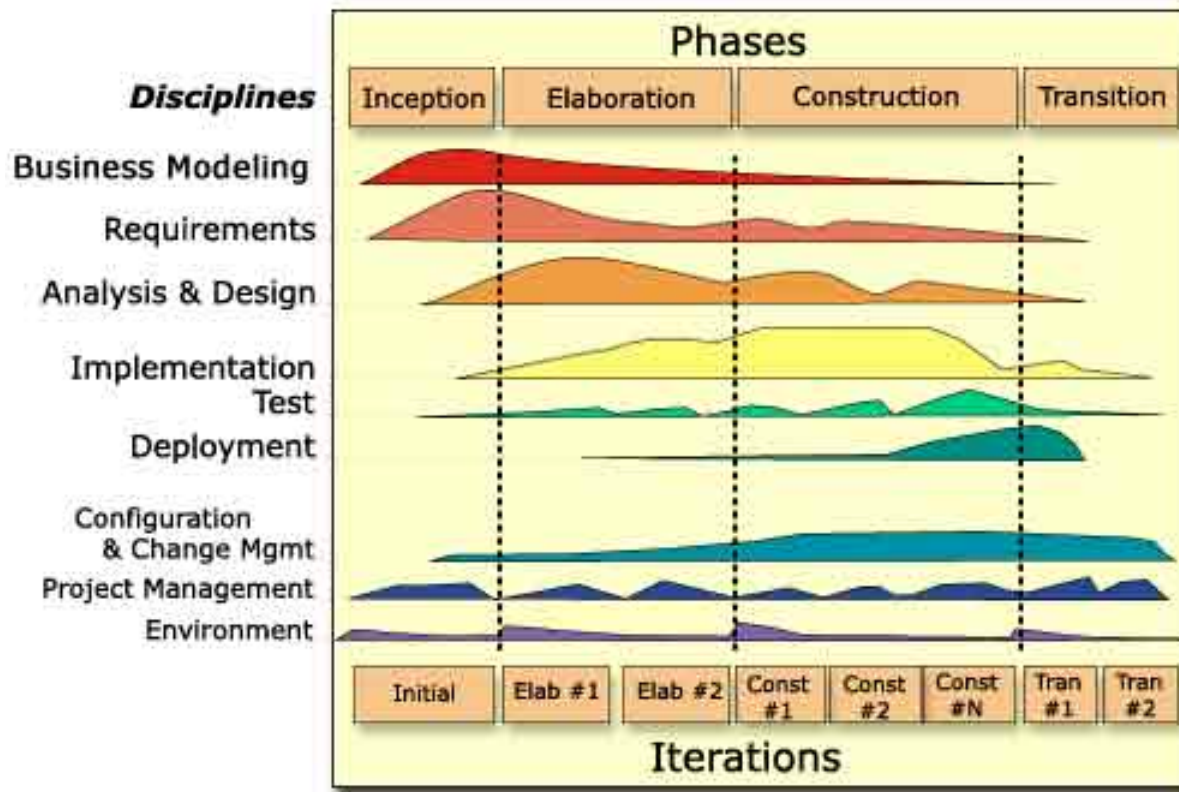
Other “Agile” Methodologies

- Pros
 - Similar to XP, can reduce process overhead
 - Responsive to user feedback
 - Amenable to change
- Cons
 - Requires close monitoring by PM
 - May not “scale” to large projects
 - Often requires better quality developers

Rational Unified Process

- RUP
- From Rational Corporation
- “Generic” version is the Unified Process
- Commercial
- Extensive tool support (expensive)
- Object-oriented
- Incremental
- Newer

Rational Unified Process



Rational Unified Process

- Develop Iteratively
- Manage Requirements
- Uses UML (Unified Modeling Language)
- Produces “artifacts”
- Use component-based architecture
- Visually model software
- Complex process
- A “framework”
- Suitable for large scale systems

Choosing Your Lifecycle

- Varies by project
- Opt for “iterative” or “incremental”
- How well are requirements understood?
- What are the risks?
- Is there a fixed deadline?
- How experienced is the team or customer?

IEEE 1074

- A standard for developing software processes
 - Lifecycle model selection
 - Project management process
 - Predevelopment processes
 - Development processes
 - Post-development processes
 - Integral process

Planning

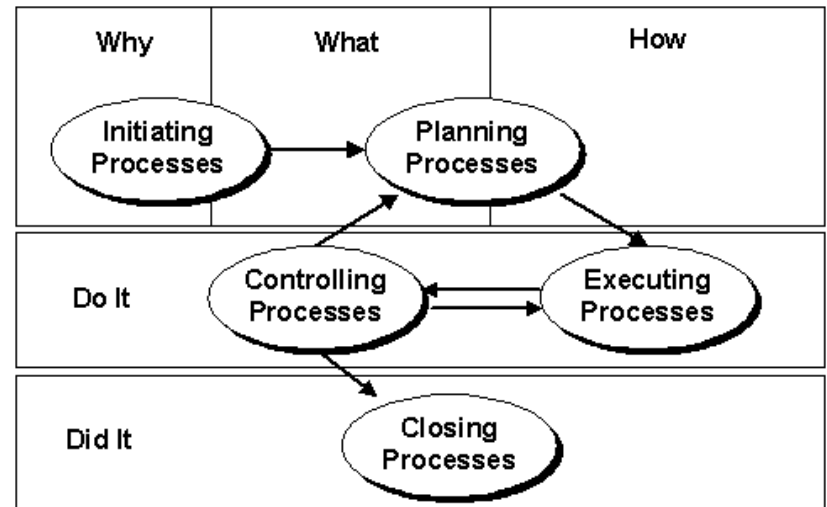
- “Plans are nothing. But planning is everything.” Gen. Dwight Eisenhower

Planning

- Preliminary planning starts on day one
- Even in the pre-project phase
- Should not be conducted “in secret”
- Need buy-in and approval
 - Very important step
 - Both from above and below

Your PM Process

- Why
 - Deliverable: ROI
- What
 - SOW, Requirements
- How
 - Design Specification, SDP, Lifecycle
- Do
 - Execution
- Done
 - PPR



Futrell, Shafer, Shafer, "Quality Software Project Management"

Primary Planning Steps

- Identify project scope and objectives
- Identify project organizational environment
- Analyze project characteristics
- Identify project products and activities
- Estimate effort for each activity
- Identify risk
- Allocate resources
- Review and communicate plan

Documents

- Planning
- Product

Planning Documents

- Software Development Plan (SDP)
- Software Quality Assurance Plan (SQAP)
- Software Configuration Management Plan (SCMP)
- Risk Management Plan
- Software Process Improvement Plan
- Communications Management Plan
- Migration Plan
- Operations Plan

Planning Documents

- You (the PM) need to choose which documents are appropriate
- Docs do not *have* to be lengthy
- Small Set:
 - Software Development Plan
 - Risk Management Plan
 - Software Quality Assurance Plan
 - Software Configuration Management Plan

Planning Documents

- Project ROI Analysis
- Statement of Work (SOW)
- Project Charter
- Software Project Management Plan (SPMP)
- Budget
- Responsibility Assignment Matrix (RAM)
- Risk Management Plan

Product Documents

- Statement of Need
- System Interface Specification
- Software Requirements Specification
- Software Design Specification
- Software Validation & Verification Plan
- User Documentation
- Support Plan
- Maintenance Documentation

Software Project Survival Guide

- Another McConnell book
- See construx.com's SPSG section
 - Good content online
 - Documents
 - Schedules
 - Checklists
 - Project web site template

Planning

- How much will it cost?
- How long will it take?
- How many people will it take?
- What might go wrong?

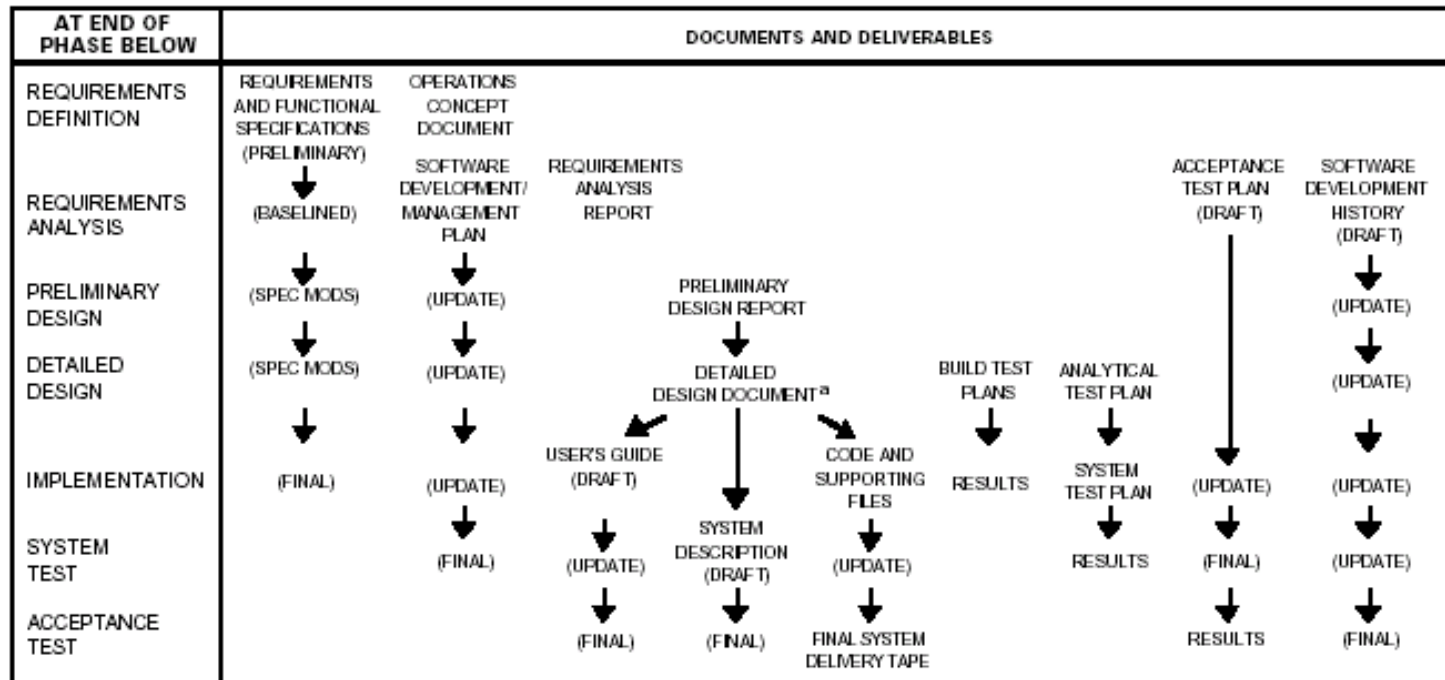
Planning

- Scoping
- Estimation
- Risk
- Schedule
- Control Strategy

Process Issues

- You want a fairly sophisticated process without incurring much overhead
- Remember, projects are often larger than they first appear
- Easier to loosen too much process than add later

Plans Evolve Over Time



NASA's "Manager's Handbook for Software Development"

Software Development Plan

- Software Project Management Plan (SPMP)
- Some consider it the most important document in the project (along with SRS)
 - Can be seen as an aggregation of other core documents
- Evolves over time as pieces come together

SDP / SPMP

- Fundamental Sections
 - Project overview
 - Deliverables
 - Project organization
 - Managerial processes
 - Technical processes
 - Budget
 - Schedule

Communications Management Plan

- Often a section of SPMP
- Describes information flow to all parties
 - Gathering and distributing information
- Status meetings
 - Monthly, Weekly, Daily?
 - Status reports are vital

Create a Project Intranet

- A great communications tool
- Reference all project resources here