

▶ POLITECNICO DI MILANO

Dipartimento di  
Elettronica e Informazione

Session 3

# Planning

**Emanuele Della Valle**

<http://home.dei.polimi.it/dellavalle>

- This slides are largely based on Prof. John Musser class notes on “Principles of Software Project Management”
- Original slides are available at <http://www.projectreference.com/>
- Reuse and republish permission was granted

## 1. Phases in Detail

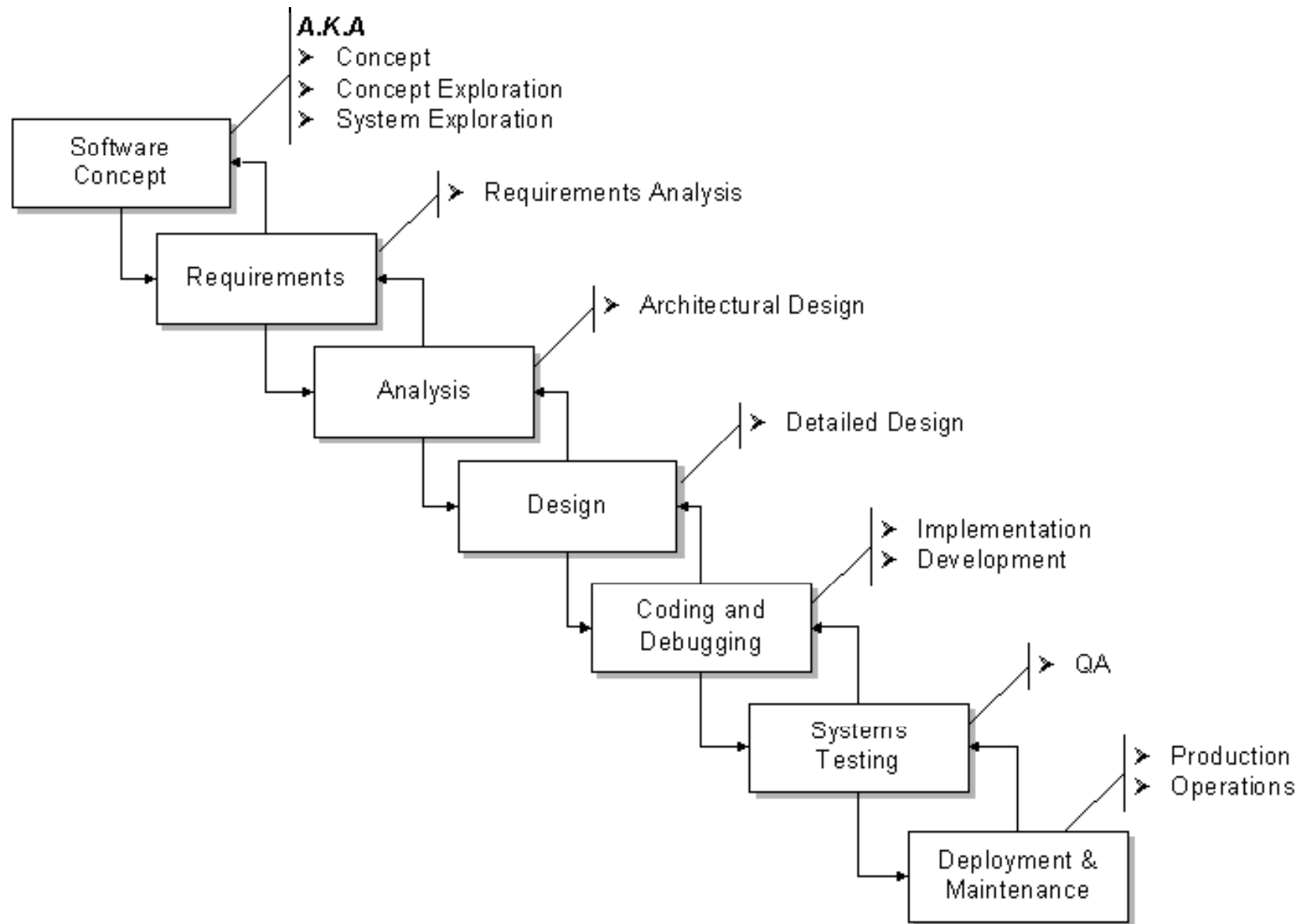
- Step-by-step of typical software project

## 2. Lifecycle Planning

## 3. Project plans

- Next Week: Lots of Project-ish Details: WBS, PERT, CPM, Scheduling & Estimation

- PMI Fundamentals
- PMI Processes
- Project and Organizations
  - Functional, Project, Matrix organizations
  - Role of PM in this type of organizations
- Project Selection
- Project Portfolio Management
- Procurement Management
- Initial documents
  - Statement of Work (SOW)
  - Project Charter



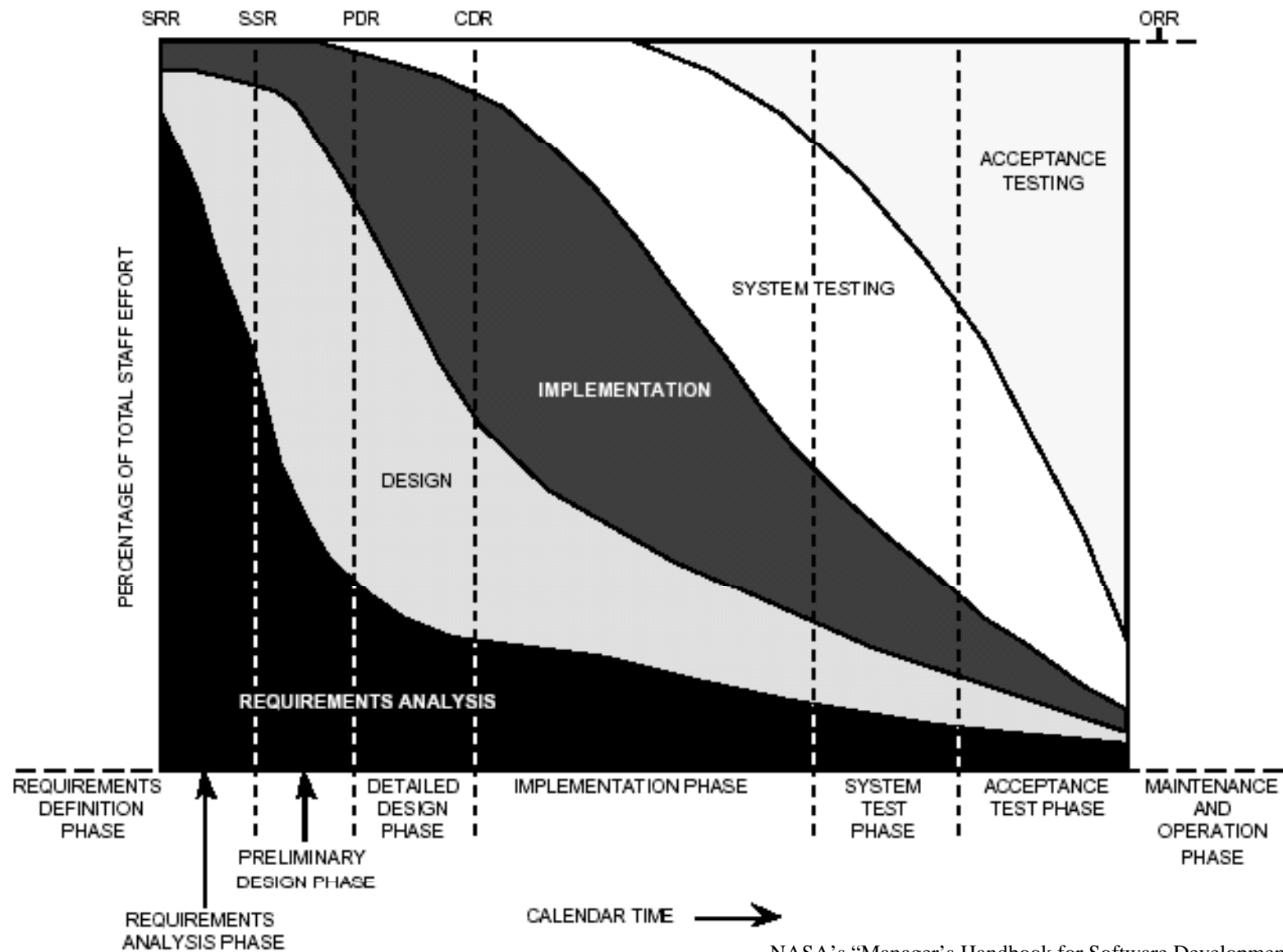
- Remember the 40-20-40 Rule
  - Specification-Implementation-Test

	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%

Bennatan, E.M, "On Time Within Budget"

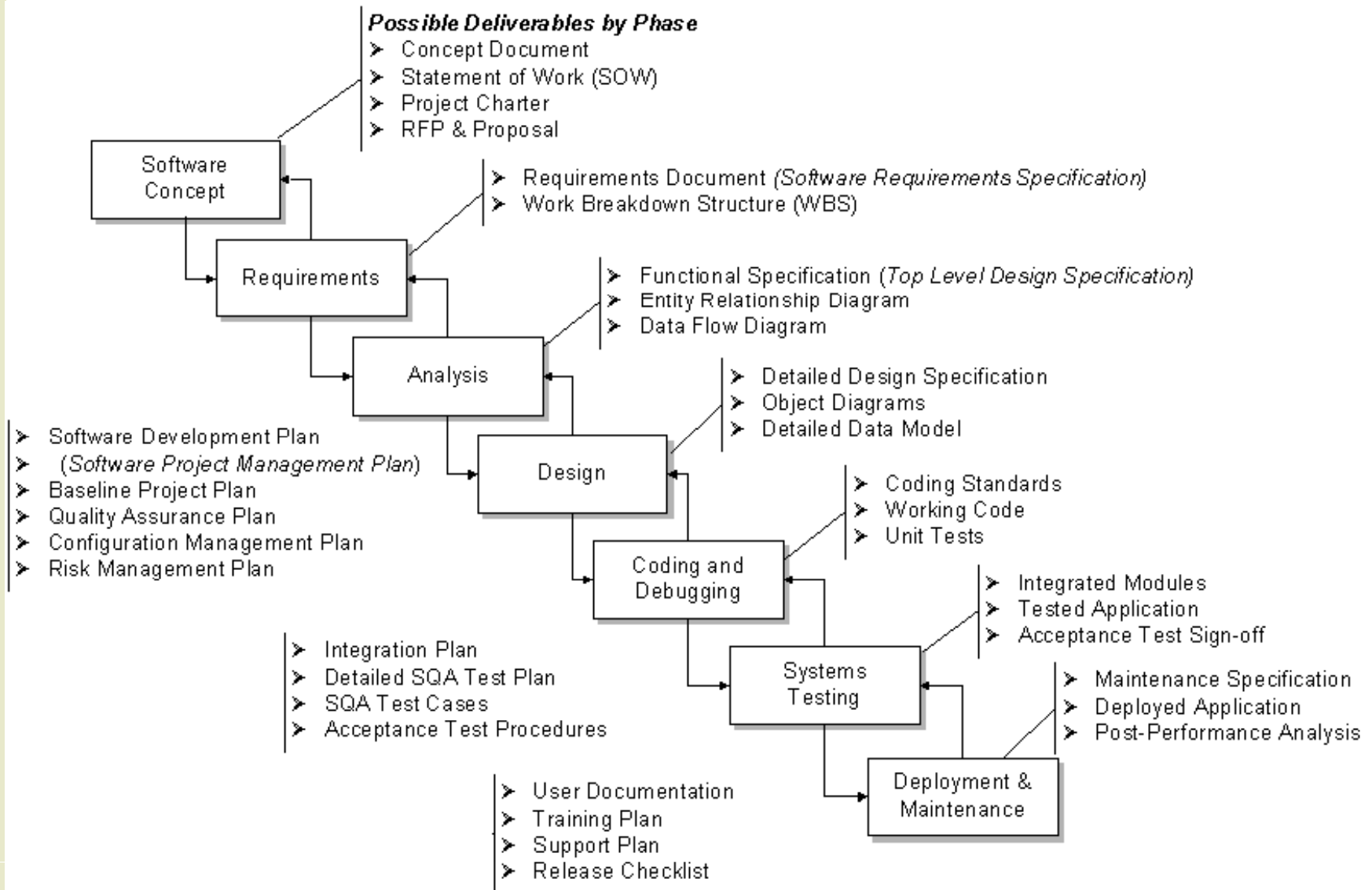
Activity	Small Project (2.5K LOC)	Large Project (500K LOC)
Analysis	10%	30%
Design	20%	20%
Code	25%	10%
Unit Test	20%	5%
Integration	15%	20%
System test	10%	15%

McConnell, Steve, "Rapid Development"



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





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

- 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
- Potential Phase Outputs:
  - Concept Document, Product Description, Proposal, SOW, Project Charter

- Characteristics & Issues
  - Lack of full commitment and leadership
  - Some frustrations:
    - Management only getting rough estimates from development
    - Development not getting enough specifics from customer
    - Finding a balanced team
  - Budget sign-off may be your 1st major task
  - Achieved via:
    - Good concept document or equivalent
    - Demonstration of clear need (justification)
    - Initial estimates

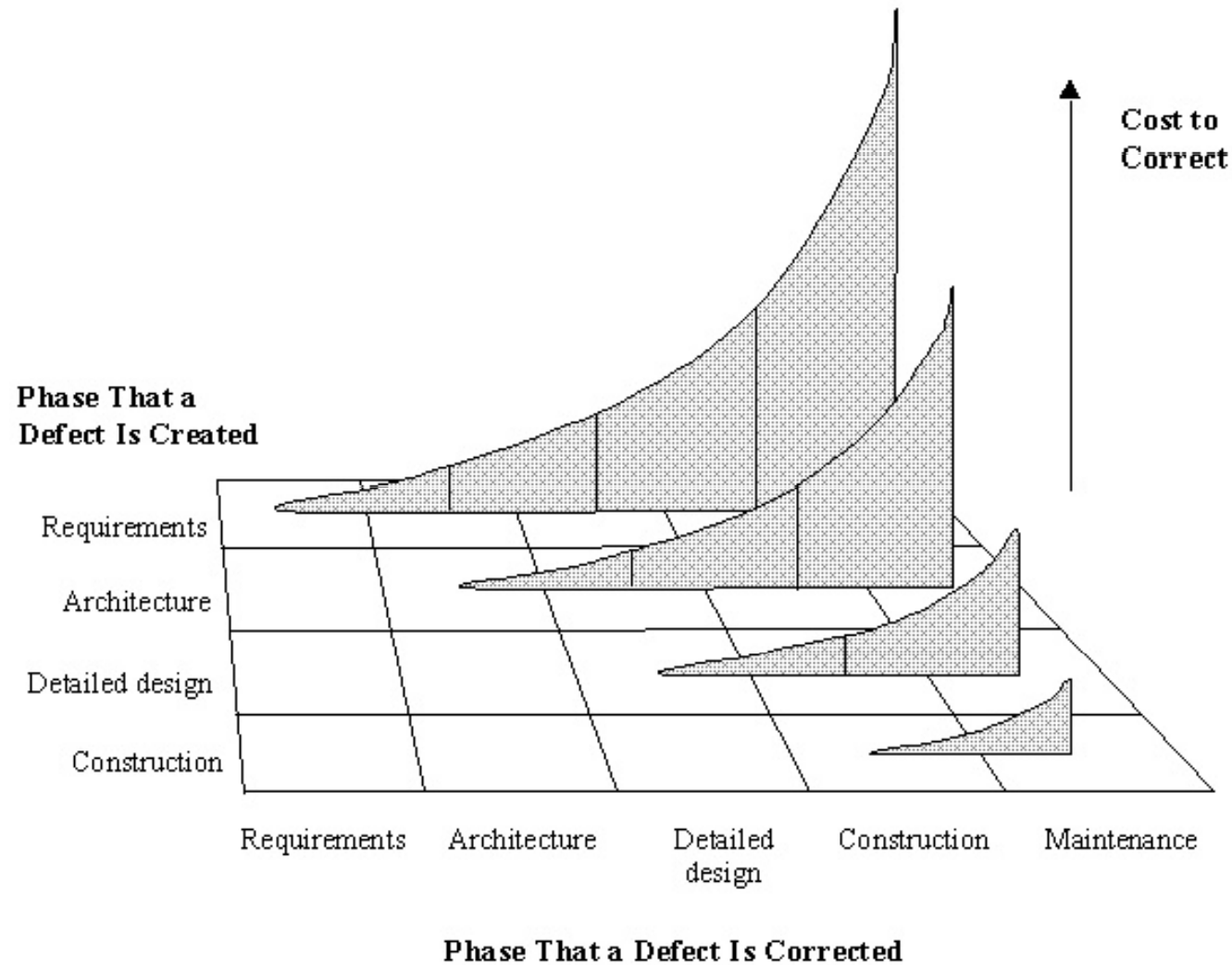
- The “What” phase
- Inputs: SOW, Proposal
- Outputs:
  - Requirements Document (RD)
    - a.k.a. Requirements Specification Document (RSD)
    - Software Requirements Specification (SRS)
  - 1st Project Baseline
  - Software Project Management Plan (SPMP)
  - Requirements Approval & Sign-Off
    - Your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

## Phases in Detail

# Requirements

14

- 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



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

- 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



## Phases in Detail

# Requirements

17

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

## 2 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

- 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
- An example I'm proud of
  - <http://www.service-finder.eu/attachments/D1.1.pdf>

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

- 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)

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

- 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

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

- Evolves from Dev. Phase
- Often done as 2 parallel phases
  - Partial integration & initial test
- Starts with integration of modules
- An initial, incomplete version constructed
- Progressively add more components

- 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

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

- 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
    - Esp. true for fixed-price contracts

- 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

- 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



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

- a.k.a. Lifecycle Management or Systems Development Life Cycle (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

- 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
- 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

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

- Why does the waterfall model “invite risk”?
- Integration and testing occur at the end
  - Often anyone’s 1st chance to “see” the program

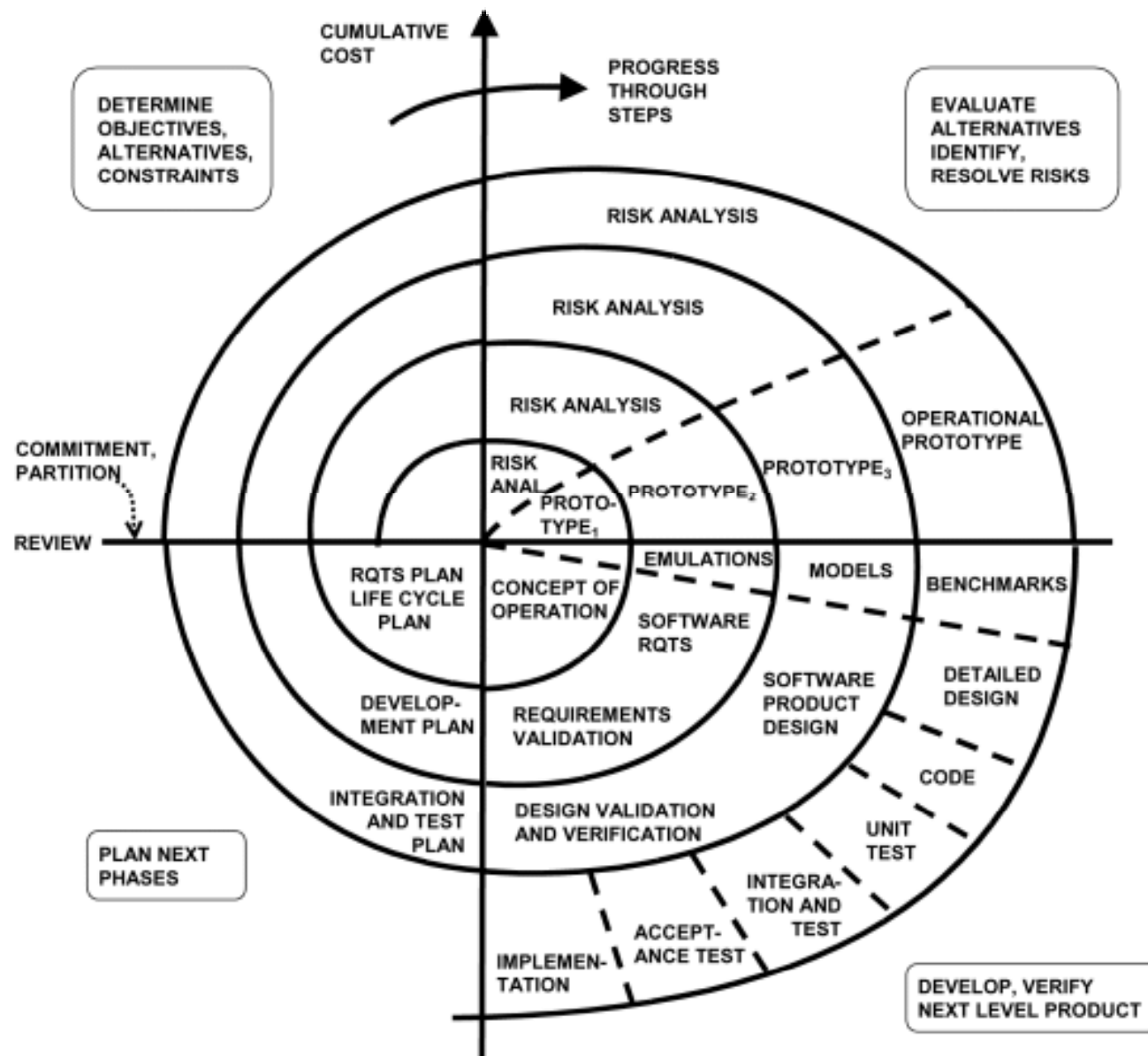
- Works well for projects with
  - Stable product definition
  - Well-understood technologies
  - Quality constraints stronger than cost & schedule
  - Technically weak staff
    - Provides structure
    - Good for overseas projects

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

- “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



# Lifecycle Planning Spiral



- Emphasizes risk analysis & mgmt. 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

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

- 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

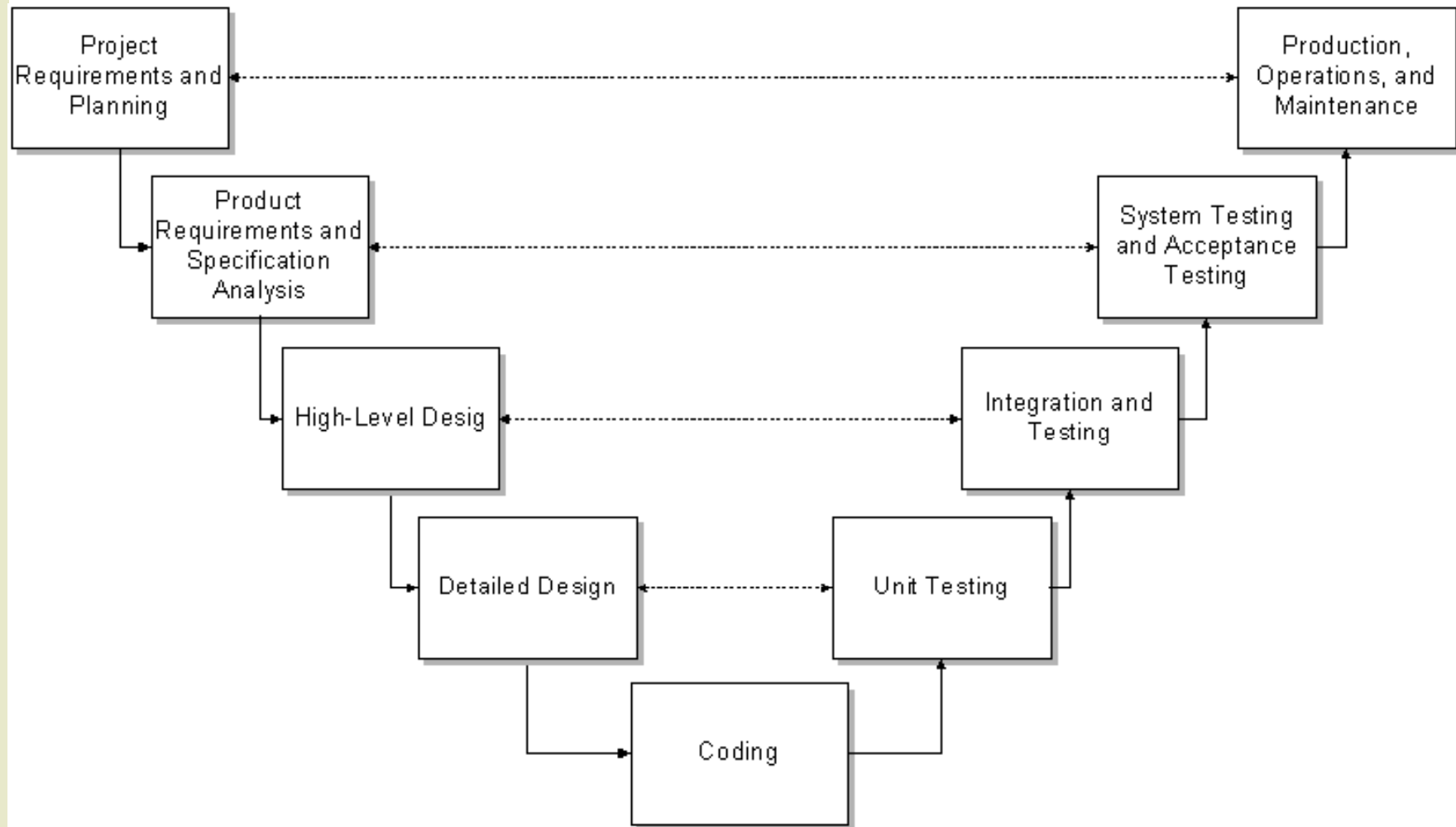
- Design most prominent parts first
  - Usually via a visual prototype
- Good for situations with:
  - Rapidly changing requirements
  - Non-committal customer
  - Vague problem domain
- Provides steady, visible progress
- Disadvantages
  - Time estimation is difficult
  - Project completion date may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

- 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?

# Lifecycle Planning

## V Process Model

47



- 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



- 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

- 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)

- 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?

Model	a	b	c	d	e	f	g	h	i	j	k
Pure Waterfall			x	x		~				~	~
Code-and-Fix							x				x
Spiral	x	x	x	x	x	~	~	~	x	x	
Modified Waterfalls	~	~	x	x	~	~	x	~	~	~	
Evolutionary Prototyping	x		~	x	~		~	x	x	~	
Staged Delivery			x	x	~	~	~		~	x	
Commercial (COTS)	x					x	x				~

Legend:

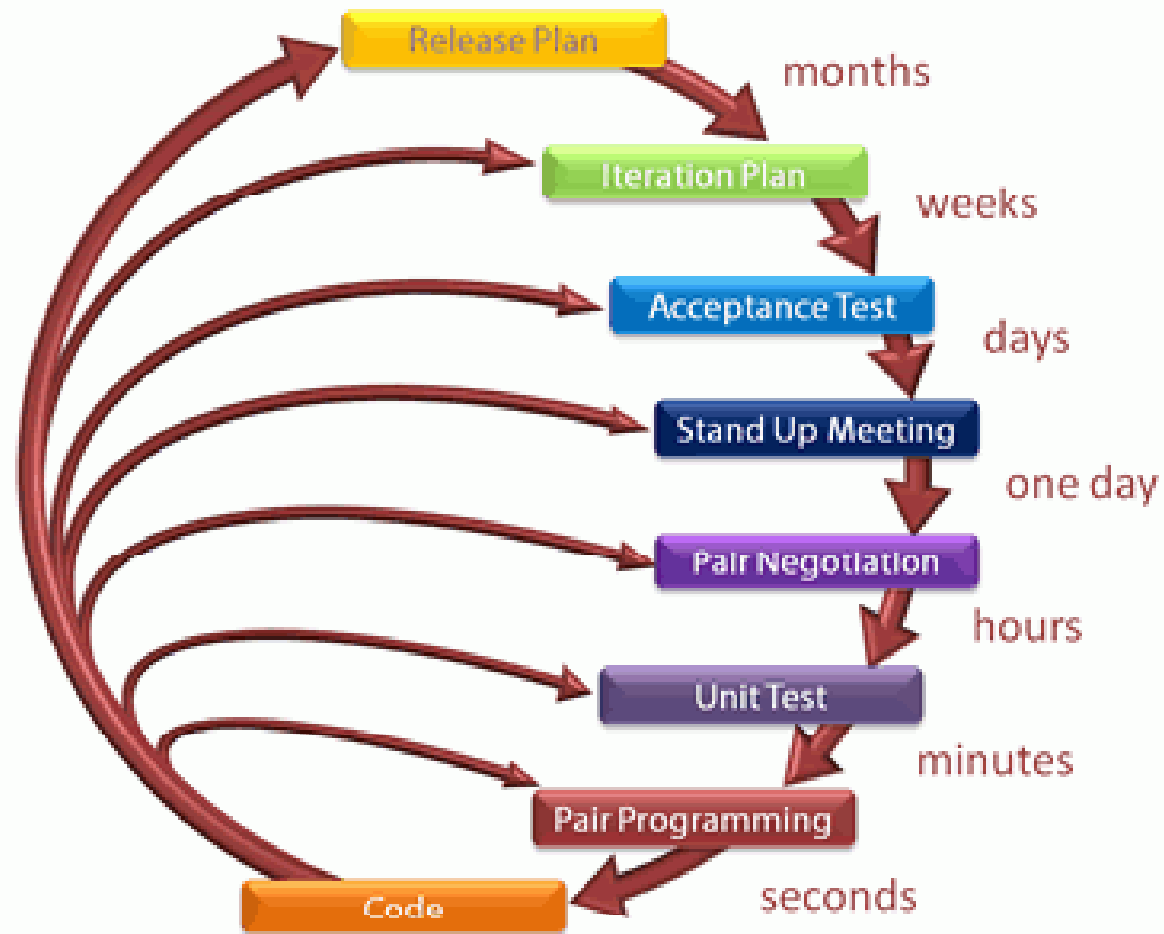
x = excellent,  
 ~ = fair to excellent,  
 empty box = poor.

- a. Works with poorly understood requirements
- b. Works with poorly understood architecture
- c. Produces highly reliable system
- d. Produces system with large growth envelope
- e. Manages risks
- f. Can be constrained to a predefined schedule
- g. Has low overhead
- h. Allows for midcourse corrections
- i. Provides customer with progress visibility
- j. Provides management with progress visibility
- k. Requires little manager or developer sophistication

[source <http://acmesoftware.com/acme/default.asp> ]

- 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 Planning/Feedback Loops



© J. Donovan Wells

## Lifecycle Planning

# eXtreme Programming

55

- 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

- Agile here means “lite”, reduced docs, highly iterative
- Agile Software Development
  - Alliance , <http://www.agilealliance.org/home>
  - their “manifesto”, <http://agilemanifesto.org/>
  - their book  
<http://www.amazon.com/exec/obidos/ASIN/0201699699/104-2402656-5403151>
- SCRUM
  - Features 30-day “Sprint” cycles
  - See <http://jeffsutherland.org/scrum/>
- Feature Driven Development (FDD)
  - XP with more emphasis on docs and process



- Adaptive Software Development (ASD)
  - Book <http://www.amazon.com/exec/obidos/tg/detail/-/0932633404/>
  - Site <http://www.adaptivesd.com/learn.html>
- Dynamic System Development Method (DSDM)
  - Popular in Europe
- Homegrown: developers often hide their “agile adventures” from management

- 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

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

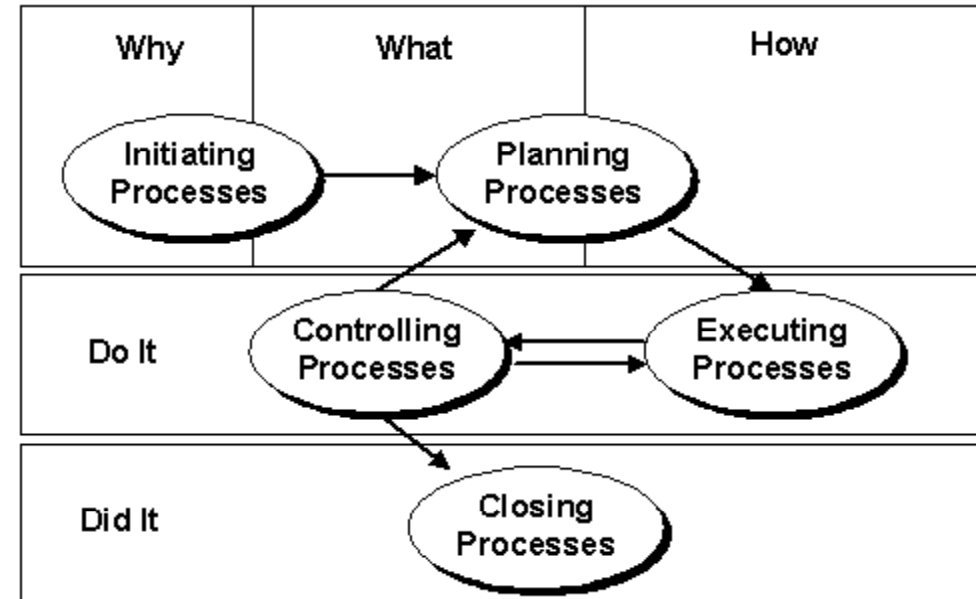
- Preliminary planning starts on day one
- Even in the pre-project phase
- Should not be conducted “in secret”
- Need buy-in and approval
  - Very important step
  - Both from above and below

## Project Plans

# Your PM Process

61

- Why
  - Deliverable: ROI
- What
  - SOW, Requirements
- How
  - Design Specification, Software Development Plan, Lifecycle
- Do it
  - Execution
- Did it
  - Post Project Report



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

- 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

## Project Plans Documents

63

- Planning
- Product

- 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



- 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

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

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

- Another McConnell book
- See construx.com's SPSG section  
<http://www.construx.com/survivalguide/subject.htm>
  - Good content online
  - Documents
  - Schedules
  - Checklists
  - Project web site template
- I tool I've often used
  - Software Project Survival Test
    - <http://www.construx.com/Page.aspx?hid=1229>

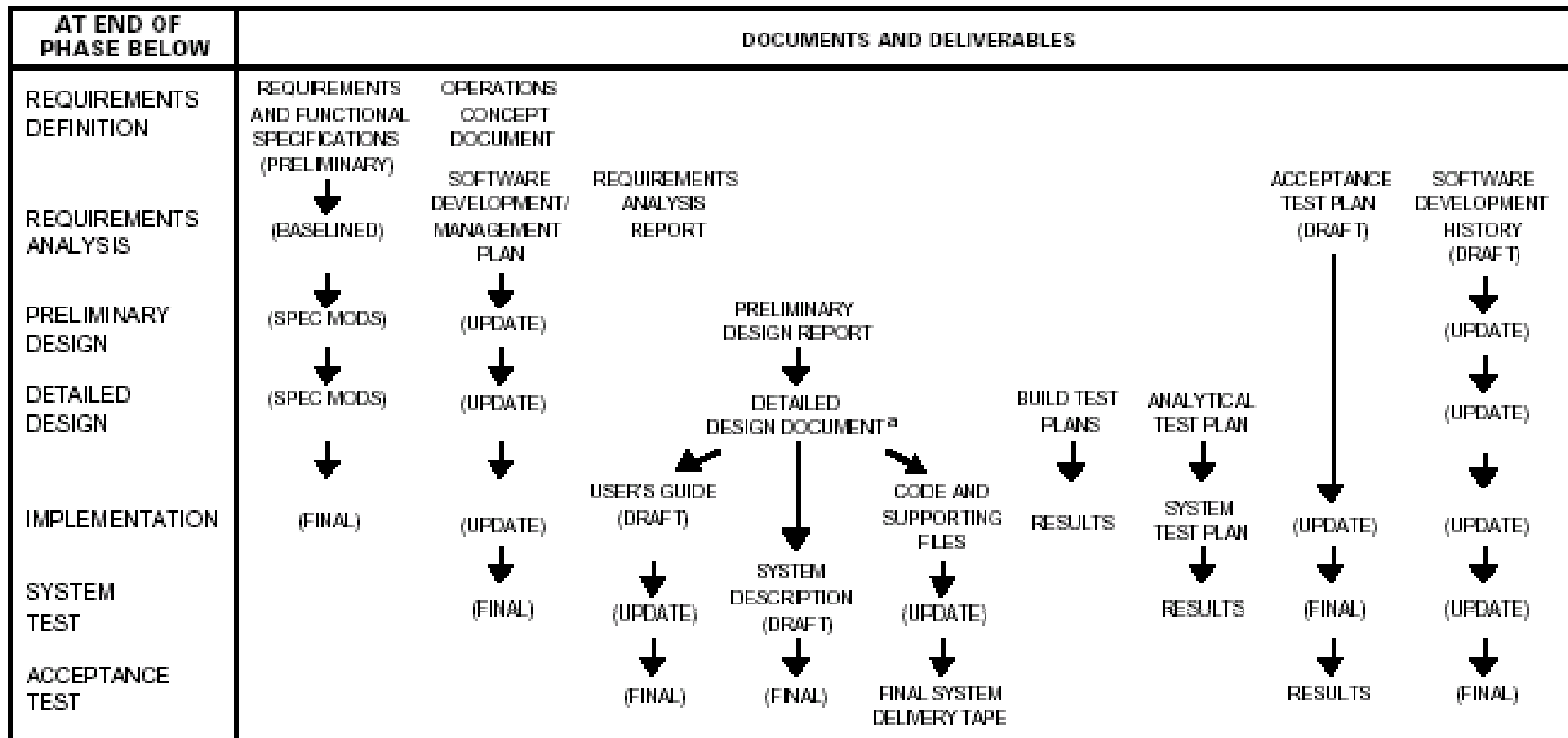
- How much will it cost?
- How long will it take?
- How many people will it take?
- What might go wrong?

- Scoping
- Estimation
- Risk
- Schedule
- Control Strategy

- 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

# Project Plans

## Plans Evolve Over Time



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



- 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
- McConnell's example
  - <http://www.construx.com/Page.aspx?nid=240>

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

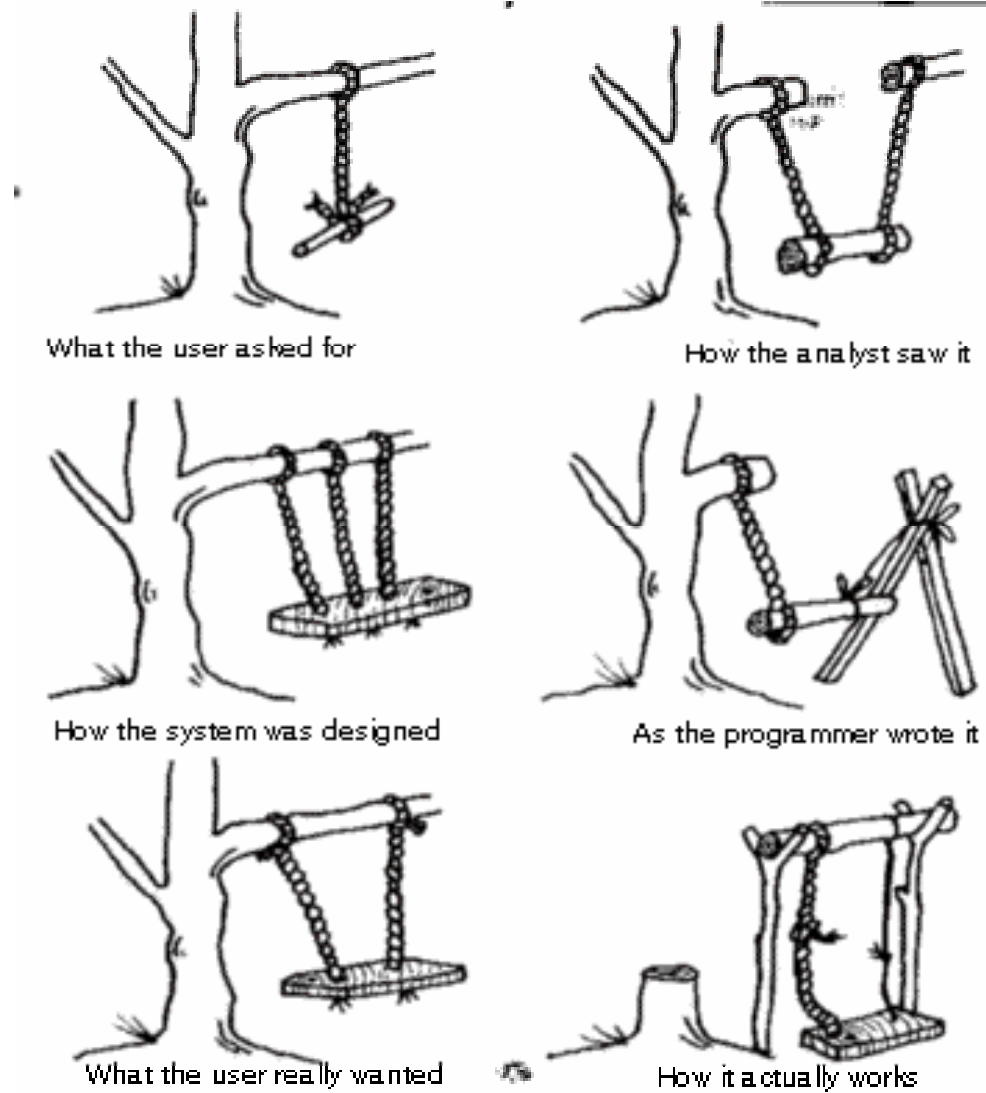
- Often a section of SPMP
- Describes information flow to all parties
  - Gathering and distributing information
- Status meetings
  - Monthly, Weekly, Daily?
  - Status reports are vital

- A great communications tool
- Reference all project resources here
- For instance have a look at portals of my current projects
  - <http://www.service-finder.eu>
  - <http://www.larkc.eu> and <http://wiki.larkc.eu>
  - <http://www.search-computing.it/>

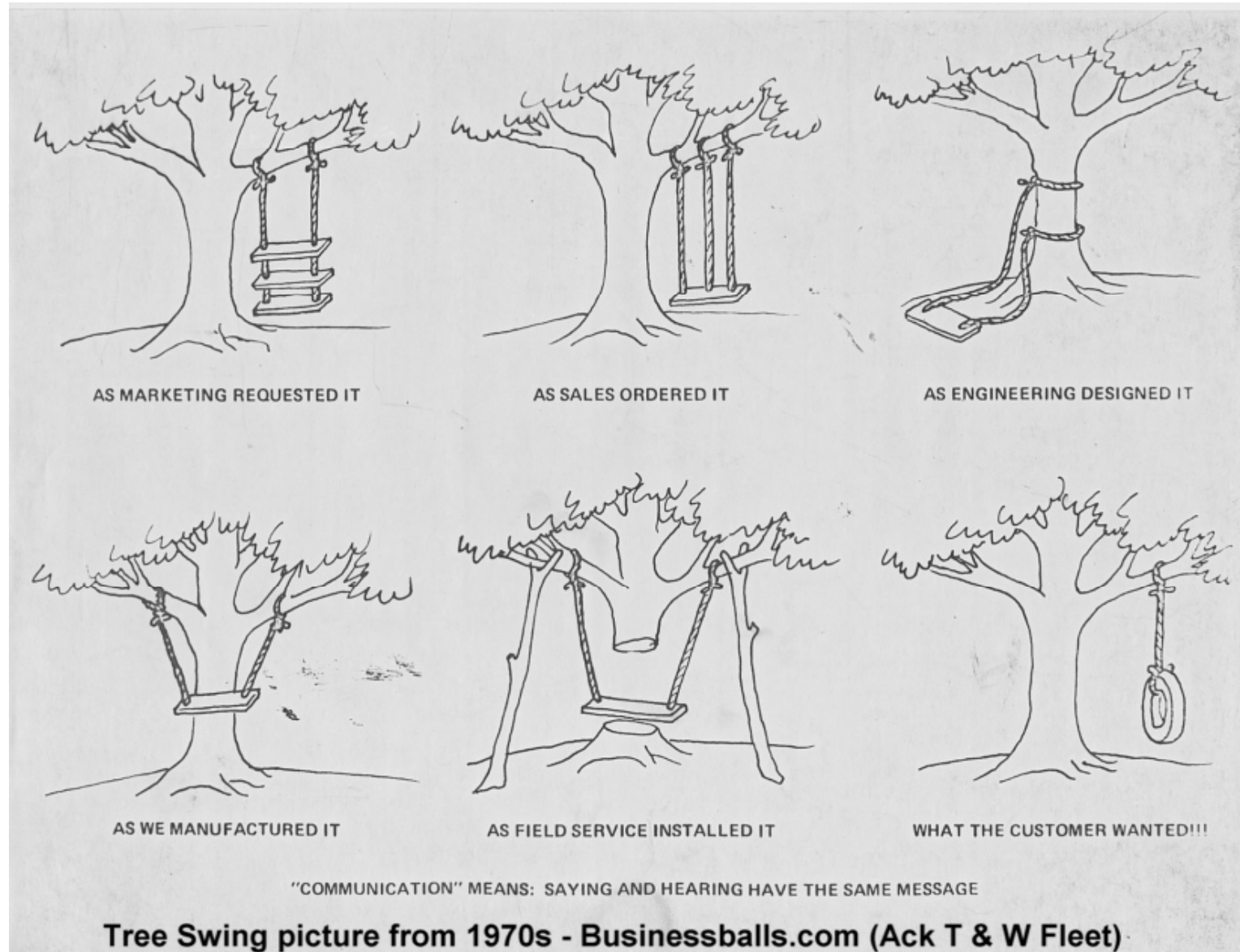
- McConnell: 8 "Estimation", 9 "Scheduling"
- Thayer: Cori pg. 171-182 "Fundamentals of Master Scheduling", Fairley 183-194 "Work Breakdown Structures"
- Review relevant links as appropriate on <http://www.projectreference.com/>

# Questions?

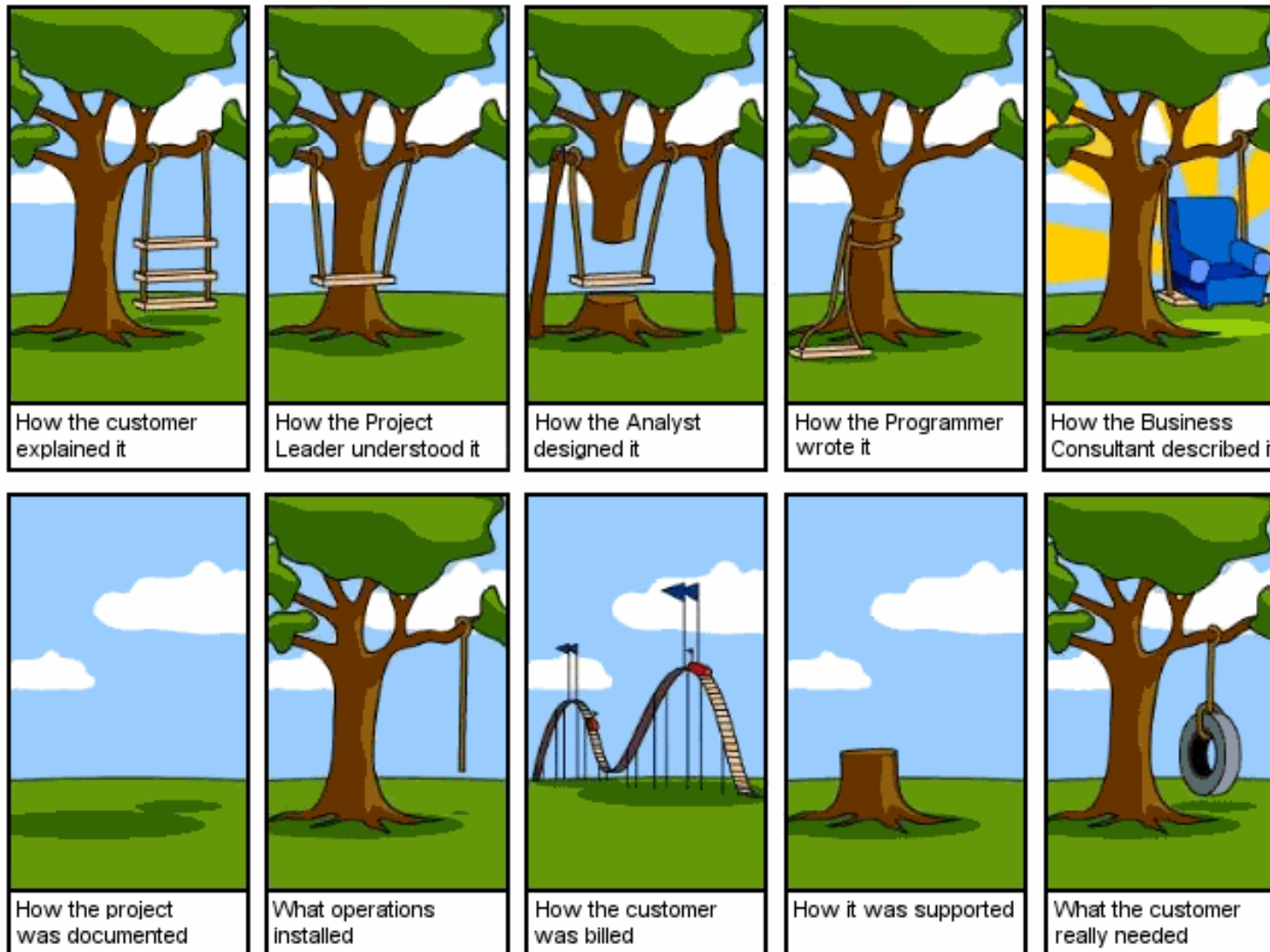
78



[source <http://www.cs.ucl.ac.uk/external/atanu/req.gif> ]







[source <http://www.codinghorror.com/blog/images/software-engineering-explained.png> ]

## Read out more about the tree swing

82

- <http://www.businessballs.com/treeswing.htm>