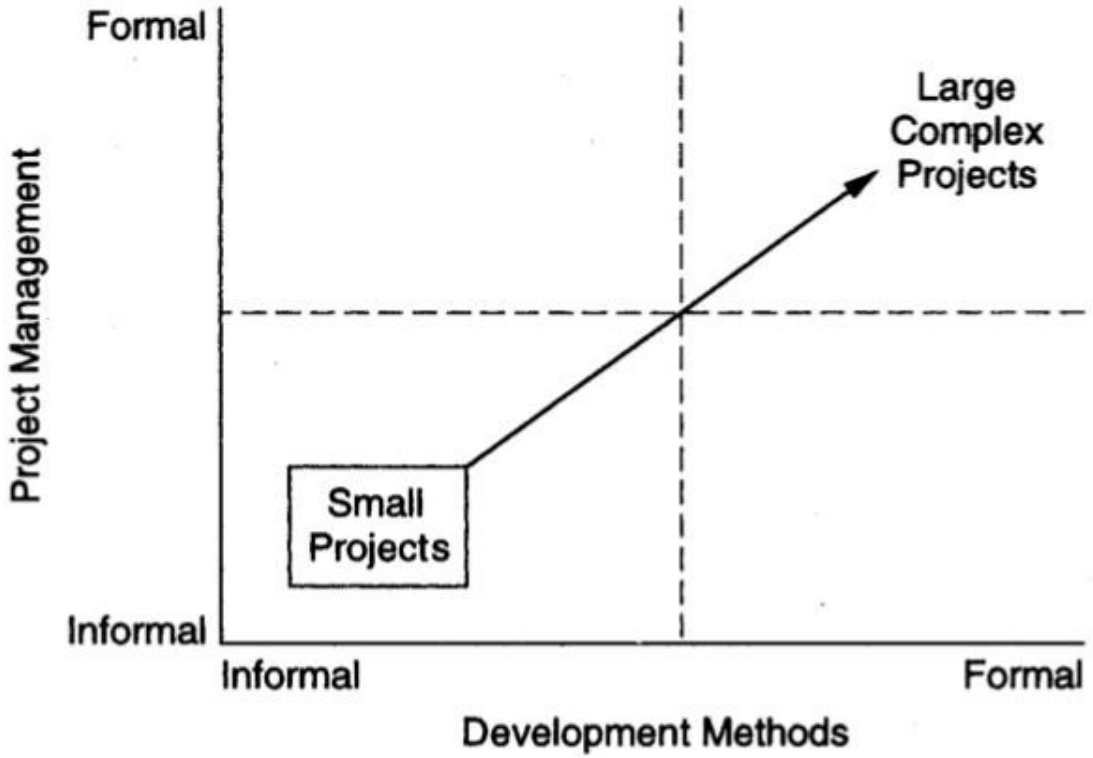
Introduction

Scale…

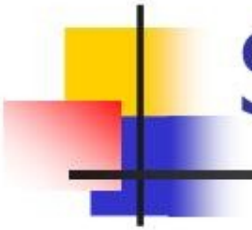

Formal
Project Management
Large
Complex
Projects
Small
Projects
Informal
Informal
Formal
Development Methods

Software

- Q : If you have to write a 10,000 line program in C to solve a problem, how long will it take?
- Answers: generally range from 2-4 months
- Let us analyze the productivity
  - Productivity = output/input resources
  - In SW output is considered as LOC
  - Input resources is effort - person months; overhead cost modeled in rate for person month
  - Though not perfect, some productivity measure is needed, as project has to keep it high

Software …

- The productivity is 2.5-5 KLOC/PM
- Q: What is the productivity in a typical commercial SW organization ?
- A: Between 100 to 1000 LOC/PM
- Q: Why is it low, when your productivity is so high? (people like you work in the industry)
- A: What the student is building and what the industry builds are two different things



Software…

- In a univ a student system is built while the commercial org builds industrial strength sw
- What is the difference between a student program and industrial strength sw for the same problem?
- Software (IEEE): collection of programs, procedures, rules, and associated documentation and data

Software…

Student

- Developer is the user
  - bugs are tolerable
  - UI not important
  - No documentation

Industrial Strength

- Others are the users
  - bugs not tolerated
  - UI v. imp. issue
  - Documents needed for the user as well as for the organization and the project

Software…

Student

- SW not in critical use
- Reliability, robustness not important
- No investment
- Don't care about portability

Industrial Strength

- Supports important functions / business
- Reliability , robustness are very important
- Heavy investment
- Portability is a key issue here

Industrial strength software

- Student programs for a problem & industrial strength software are two different things
- Key difference is in quality (including usability, reliability, portability, etc.)
- Brooks thumb-rule: Industrial strength sw costs 10 time more than student sw
- In this course, software means industrial strength software
- This software has some characteristics

Requires tight Schedule

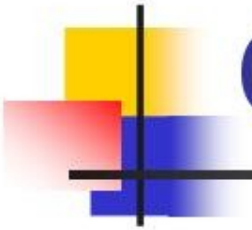
- Business requirements today demand short delivery times for software
- In the past, software products have often failed to be completed in time
- Along with cost, cycle time is a fundamental driving force

Is Expensive

- Let us look at costs involved
  - Productivity = Appx 1000 LOC/PM
  - Cost = $3K to $10K/PM
  - Cost per LOC = $5 to $15
  - I.e, each line of delivered code costs many $s
- A simple application for a business may have 20KLOC to 50KLOC
  - Cost = $100K to $2.25Million
  - Can easily run on $10K-$20K hardware
  - So HW costs in an IT solution are small as compared to SW costs.

Productivity – for cost and schedule

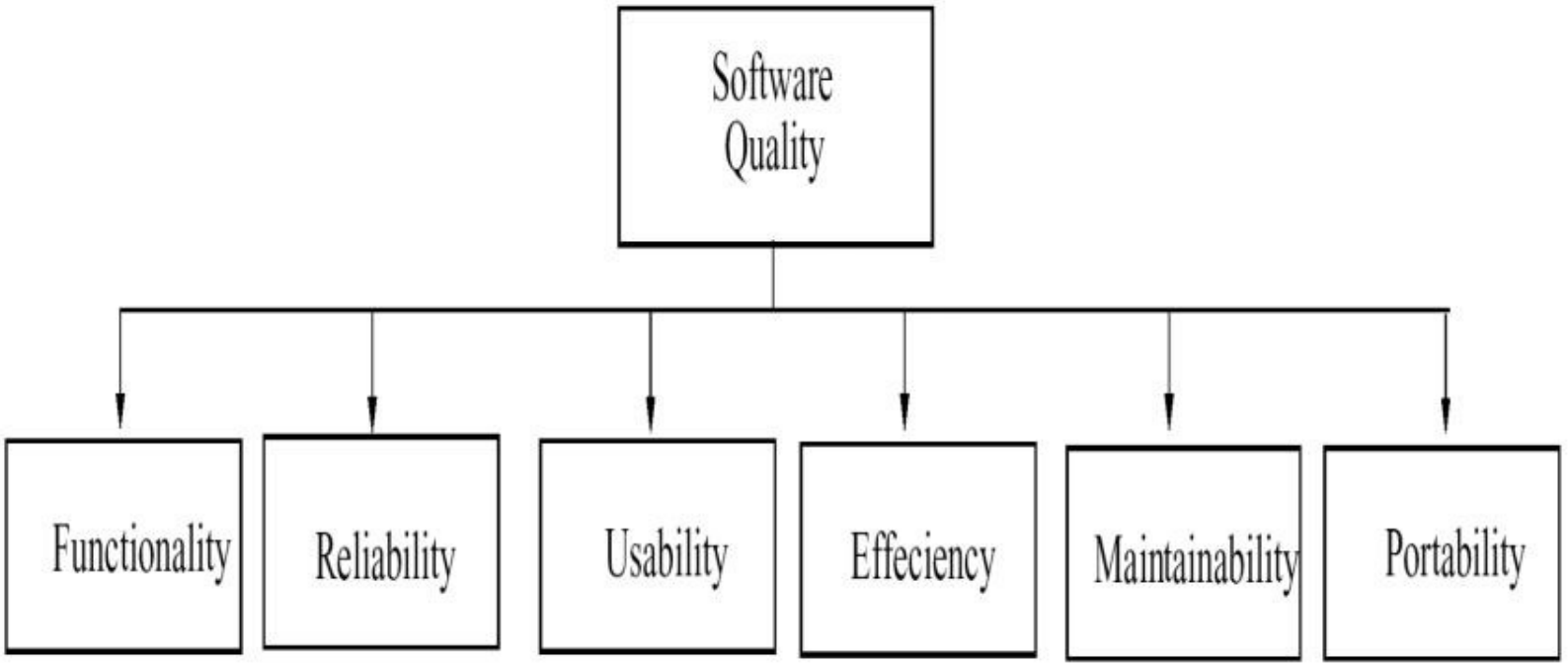
- An industrial strength software project is driven by cost and schedule
- Both can be modeled by productivity, measured in terms of output per unit effort (e.g. LOC per person month)
  - Higher productivity leads to lower cost
  - Higher productivity leads to lower cycle time
- Hence, for projects (to deliver software), quality and productivity are basic drivers

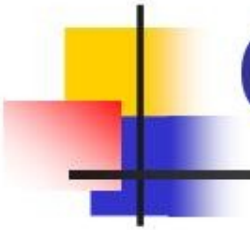


Quality

- Along with productivity, quality is the other major driving factor
- Developing high Q sw is a basic goal
- Quality of sw is harder to define

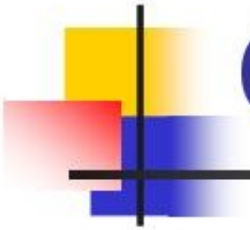
Quality – ISO standard





Quality – ISO std…

- ISO std has six attributes
  - Functionality:Capability to provide functions specified.
  - Reliability:Capability to maintain a specified level of performance.
  - Usability:Capability to be understood ,learned and used.
  - Efficiency:Capability to provide performance relative to the amount of resources used.
  - Maintainability:The capability to be modified for making corrections,improvements or adaptatation.
  - Portability:Capability to be adopted for different specified environments without applying actions or means.

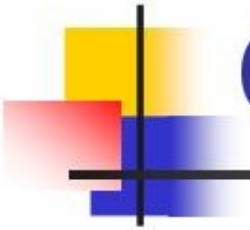


Quality…

- Multiple dimensions mean that not easy to reduce Q to a single number
- Concept of Q is project specific
  - For some reliability is most important
  - For others usability may be more important(games s/w)
- Reliability is generally considered the main Q criterion

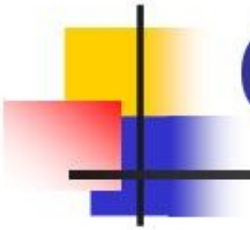
Quality…

- Reliability = Probability of failure
  - hard to measure
  - approximated by no. of defects in software
- To normalize Quality = Defect density
  - Quality = No. of defects delivered / Size
- Defects delivered - approximated with no. of defects found in operation
- Current practices: less than 1 def/KLOC
- What is a defect? Project specific!



Quality – Maintainability

- Once sw delivered, it enters the maintenance phase, in which
  - Residual errors are fixed – this is corrective maintenance
  - Upgrades and environment changes are done – this is adaptive maintenance
- Maintenance can consume more effort than development over the life of the software (can even be 20:80 ratio!)
- Hence maintainability is another quality attribute of great interest



Quality and Productivity

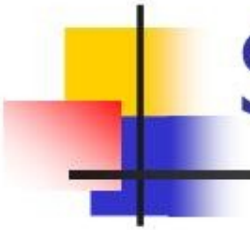
- Hence, quality and productivity (Q&P) are the basic drivers in a sw project
- The aim of most methodologies is to deliver software with a high Q&P
- Besides the need to achieve high Q&P there are some other needs

Change

- Only constant in business is change!
- Requirements change, even while the project is in progress
- In a project, up to 40% of development effort may go in implementing changes
- Practices used for developing software must accommodate change

Scale

- Most industrial strength software tend to be large and complex
- Methods for solving small problems do not often scale up for large problems
- Two clear dimensions in a project
  - engineering
  - project management
- For small, both can be done informally, for large both have to be formalized

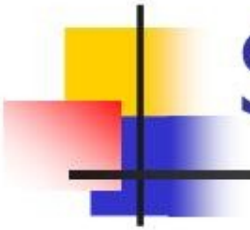


Scale…

- An illustration of issue of scale is counting the number of people in a room vs taking a census
  - Both are counting problems
  - Methods used in first **not** useful for census
  - For large scale counting problem, must use different techniques and models
  - Management will become critical

Scale: Examples

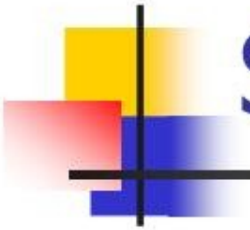
Gcc	980KLOC	C, C++, yacc
Perl	320 KLOC	C, perl, sh
Appache	100 KLOC	C, sh
Linux	30,000 KLOC	C, c++
Windows XP	40,000 KLOC	C, C++



Scale…

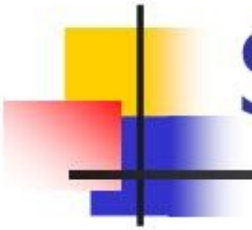
- As industry strength software tends to be large, hence methods used for building these must be able to scale up
- For much of the discussion, we will high Q&P as the basic objective

Software Process



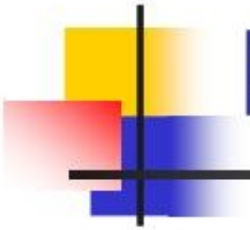
Summary

- The problem domain for SE is industrial strength software
- SE aims to provide methods for systematically developing (industrial strength) software
- Besides developing software the goal is to achieve high quality and productivity (Q&P)
- Methods used must accommodate changes, and must be able to handle large problems



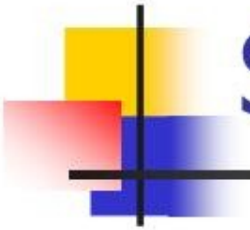
Software Engineering

- We have specified the problem domain – industrial strength software
  - Besides delivering the software, cost, quality, and schedule are drivers
- Software engineering is defined as the systematic approach for development of (industrial strength) software



Process, People, Technology

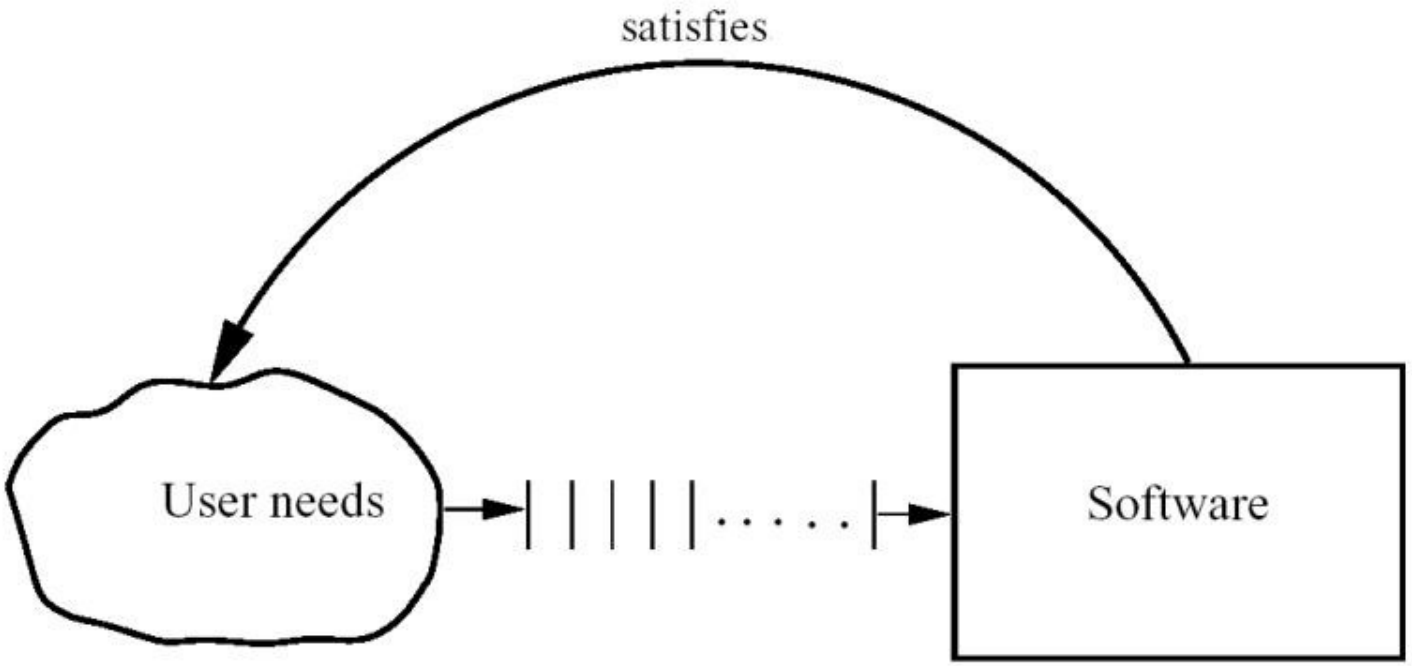
- Q&P is an essential goal
- Q&P depends on people, process, and technology
  - Processes help people become more productive and create fewer errors
  - Tools help people execute some tasks in the process more efficiently and effectively
  - So, process forms the core

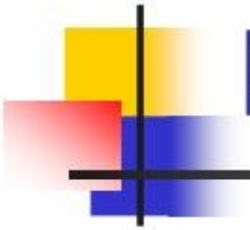


Software Process

- Process is distinct from product – products are outcomes of executing a process on a project
- SW Engg. focuses on process
- Premise: Proper processes will help achieve project objectives of high QP

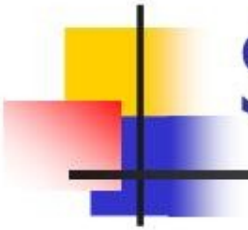
The software Development Problem





Project and Process

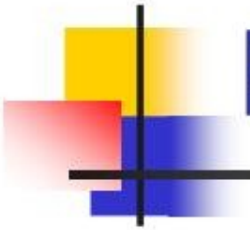
- A software project is one instance of the development problem
- Development process takes the project from user needs to software
- There are other goals of cost schedule and quality, besides delivering software
- Need other processes



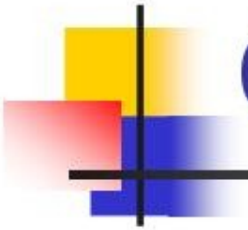
Software Process…

- Process: A sequence of steps performed to achieve some goal
- Software Process: The sequence of steps performed to produce software with high quality, within budget and schedule
- Many types of activities performed by diff people in a software project
- Better to view software process as comprising of many component processes

Component Software Processes



- Two major processes
  - Development – focuses on development and quality steps needed to engineer the software
  - Project management – focuses on planning and controlling the development process
- Development process is the heart of software process; other processes revolve around it
- These are executed by different people
  - developers execute engg. Process
  - project manager executes the mgmt proces

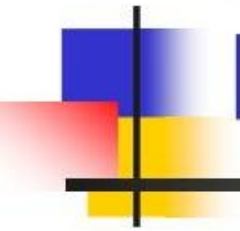


Component Processes…

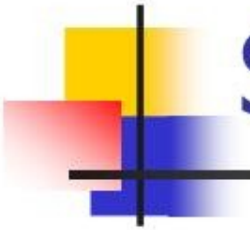
- Other processes
  - Configuration management process: manages the evolution of artifacts
  - Change management process: how changes are incorporated
  - Process management process: management of processes themselves
  - Inspection process: How inspections are conducted on artifacts

Process Specification

- Process is generally a set of phases
- Each phase performs a well defined task and generally produces an output
- Intermediate outputs – *work products*
- At top level, typically few phases in a process
- How to perform a particular phase – *methodologies* have been proposed



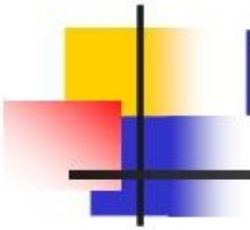
Development Process and Process Models



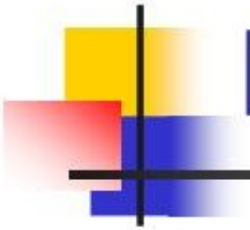
Software Project

- Project – to build a sw system within cost and schedule and with high quality which satisfies the customer
- Suitable process needed to reach goals
- Process should not just help produce the software but help achieve the highest Q&P

Project' s process and Process Models

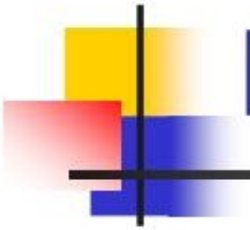


- For a project, the project's process to be followed is specified during planning
- A process model specifies a general process that is optimal for a class of problems
- A project may select its process using one of the process models



Development Process

- A set of phases and each phase being a sequence of steps
- Sequence of steps for a phase - methodologies for that phase.
- Why have phases
  - To employ divide and conquer
  - each phase handles a different part of the problem
  - helps in continuous validation

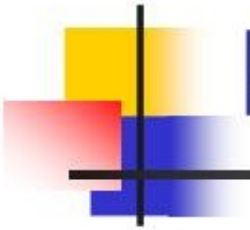


Development Process

- Commonly has these activities: Requirements analysis, architecture, design, coding, testing, delivery
- Different models perform them in different manner

Desired chareteristics of s/w process

- Predictability
- Support testability & maintainability
- Support change
- Early defect removal
- Process improvement & feddback

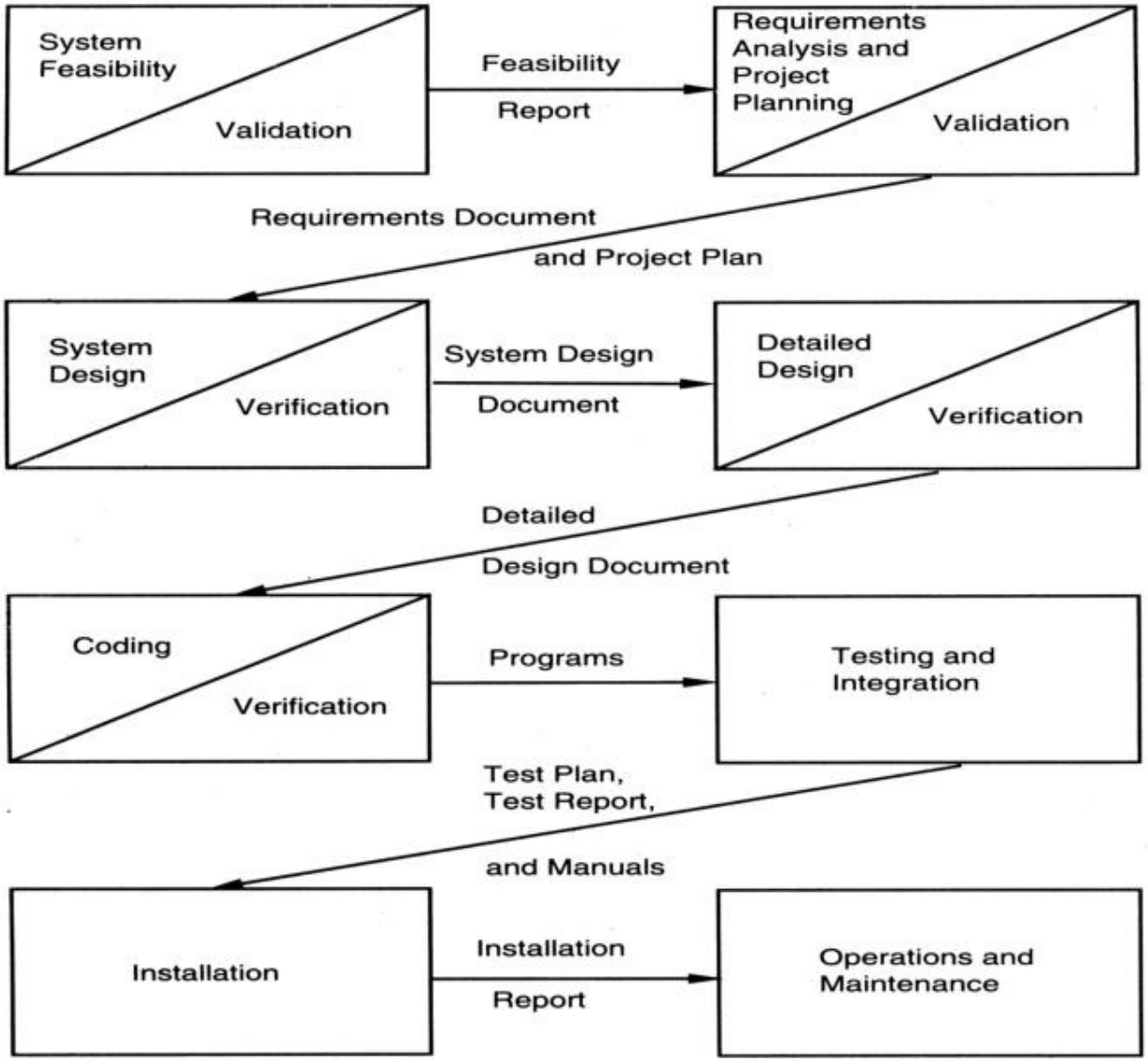


Process Models

- A process model specifies a general process, usually as a set of stages
- This model will be suitable for a class of projects
- I.e. a model provides generic structure of the process that can be followed by some projects to achieve their goals

Waterfall Model

- Linear sequence of stages/phases
- Requirements – HLD – DD – Code – Test – Deploy
- A phase starts only when the previous has completed; no feedback
- The phases partition the project, each addressing a separate concern


System Feasibility
Validation
Feasibility Report
Requirements Analysis and Project Planning
Validation
Requirements Document and Project Plan
System Design
Verification
System Design Document
Detailed Design
Verification
Detailed Design Document
Coding
Verification
Programs
Testing and Integration
Test Plan, Test Report, and Manuals
Installation
Installation Report
Operations and Maintenance

Waterfall…

- Linear ordering implies each phase should have some output
- The output must be validated/certified
- Outputs of earlier phases: work products
- Common outputs of a waterfall: SRS, project plan, design docs, test plan and reports, final code, supporting docs

Waterfall Advantages

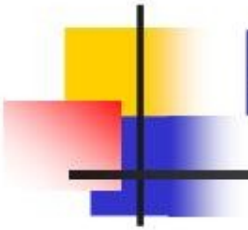
- Conceptually simple, cleanly divides the problem into distinct phases that can be performed independently
- Natural approach for problem solving
- Easy to administer in a contractual setup – each phase is a milestone

Waterfall disadvantages

- Assumes that requirements can be specified and frozen early
- May fix hardware and other technologies too early
- Follows the "big bang" approach – all or nothing delivery; too risky
- Very document oriented, requiring docs at the end of each phase

Waterfall Usage

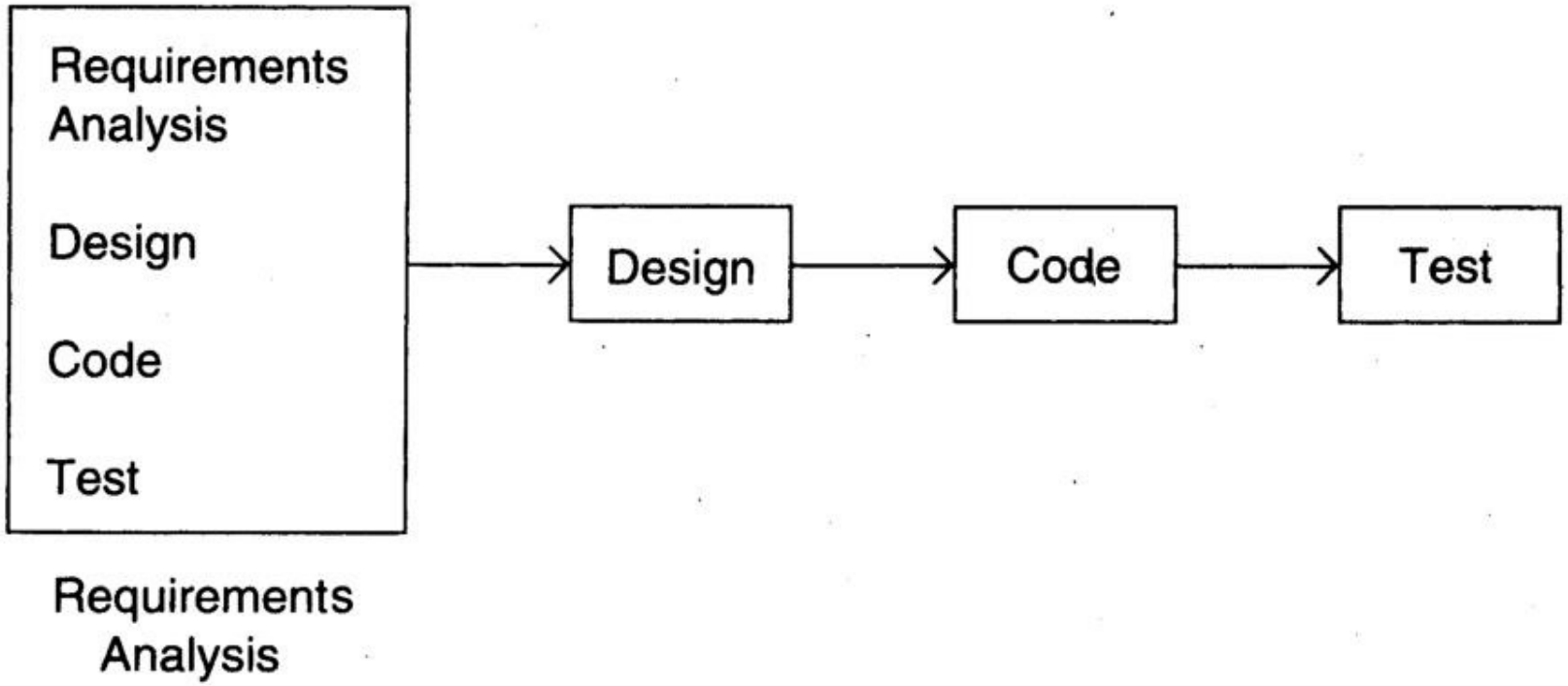
- Has been used widely
- Well suited for projects where requirements can be understood easily and technology decisions are easy
- I.e. for familiar type of projects it still may be the most optimum

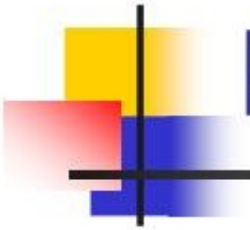


Prototyping

- Prototyping addresses the requirement specification limitation of waterfall
- Instead of freezing requirements only by discussions, a prototype is built to understand the requirements
- Helps alleviate the requirements risk
- A small waterfall model replaces the requirements stage

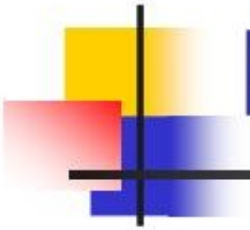
Prototyping





Prototyping

- Development of prototype
  - Starts with initial requirements
  - Only key features which need better understanding are included in prototype
  - No point in including those features that are well understood
  - Feedback from users taken to improve the understanding of the requirements



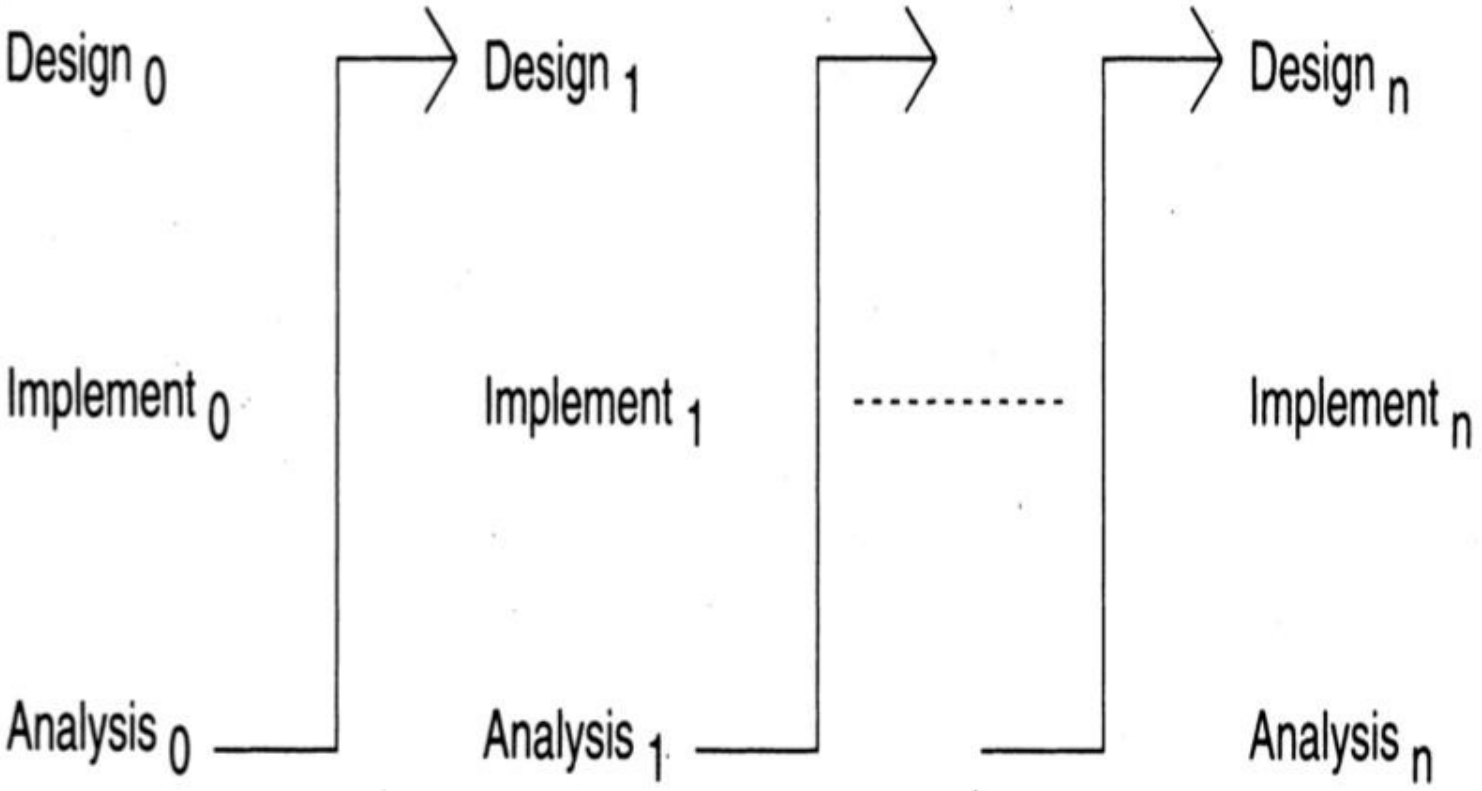
Prototyping

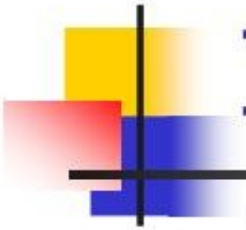
- Cost can be kept low
  - Build only features needing clarification
  - "quick and dirty" – quality not important, scripting etc can be used
  - Things like exception handling, recovery, standards are omitted
  - Cost can be a few % of the total
  - Learning in prototype building will help in building, besides improved requirements

Prototyping

- Advantages: req will be more stable, req frozen later, experience helps in the main development
- Disadvantages: Potential hit on cost and schedule
- Applicability: When req are hard to elicit and confidence in reqs is low; i.e. where reqs are not well understood
- Reduction in testing cost,training and no documentation

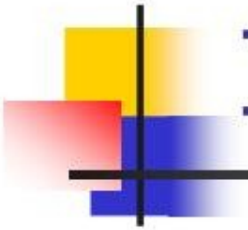
Iterative Enhancement





Iterative Development

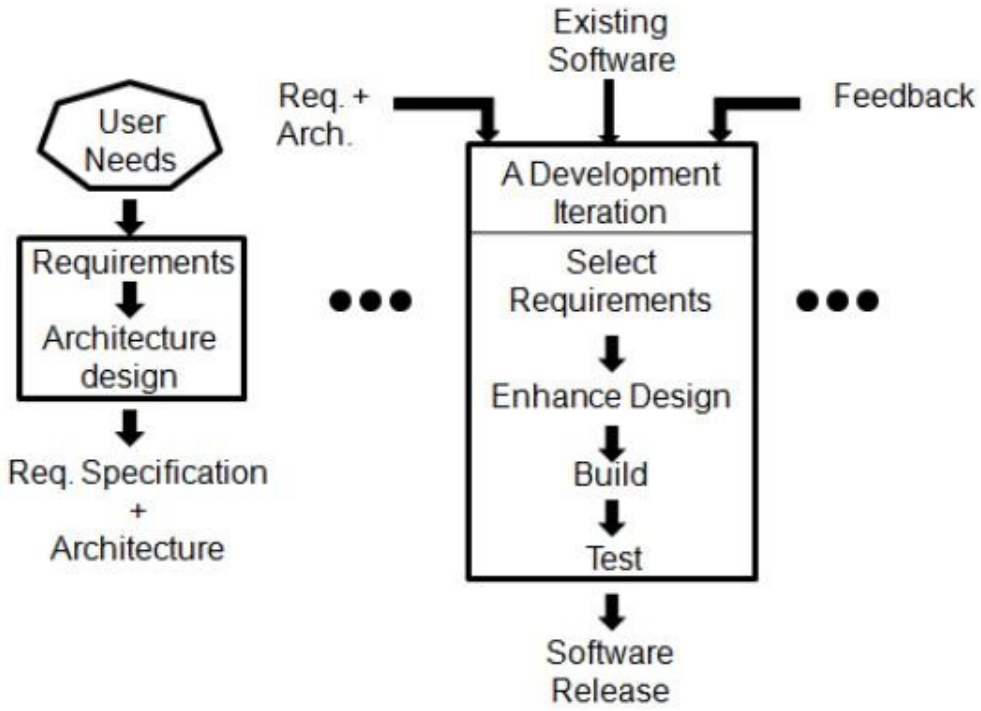
- Counters the "all or nothing" drawback of the waterfall model
- Combines benefit of prototyping and waterfall
- Project control list contains all tasks that must be performed.
- Develop and deliver software in increments
- Each increment is complete in itself
- Can be viewed as a sequence of waterfalls
- Feedback from one iteration is used in the future iterations

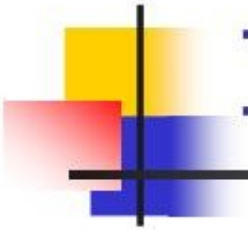


Iterative Development

- Products almost always follow it
- Used commonly in customized development also
  - Businesses want quick response for sw
  - Cannot afford the risk of all-or-nothing
- Newer approaches like XP, Agile,… all rely on iterative development

Another form of Iteration…



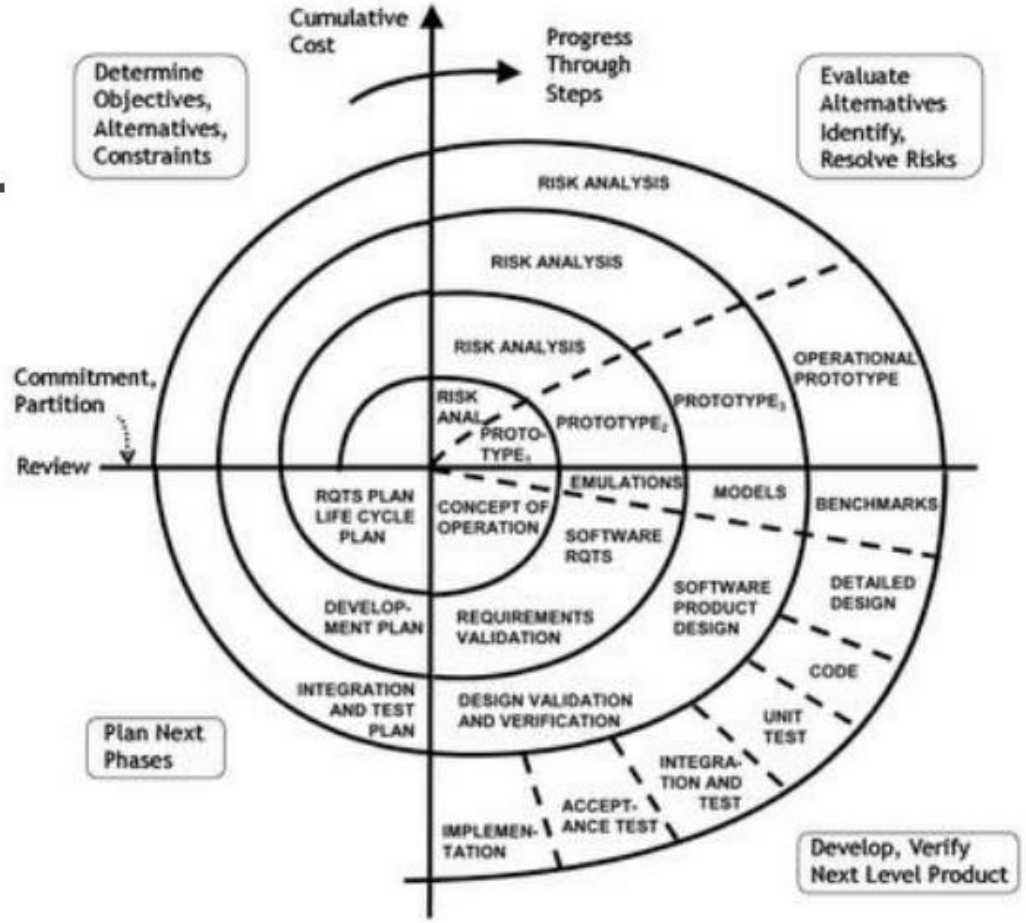


Iterative Development

- Benefits: Get-as-you-pay, feedback for improvement,
- Drawbacks: Architecture/design may not be optimal, rework may increase, total cost may be more
- Applicability: where response time is important, risk of long projects cannot be taken, all req not known

Another Form of Iterative

- The first iteration does the requirements and architecture in the waterfall way
- The development and delivery is done incrementally in iterations


Cumulative Cost
Progress Through Steps
Determine Objectives, Alternatives, Constraints
Evaluate Alternatives Identify, Resolve Risks
RISK ANALYSIS
RISK ANALYSIS
RISK ANALYSIS
Commitment, Partition
Review
RISK ANAL
PROTO-TYPE1
PROTOTYPE2
PROTOTYPE3
OPERATIONAL PROTOTYPE
RQTS PLAN LIFE CYCLE PLAN
CONCEPT OF OPERATION
EMULATIONS
MODELS
BENCHMARKS
SOFTWARE RQTS
DEVELOP-MENT PLAN
REQUIREMENTS VALIDATION
SOFTWARE PRODUCT DESIGN
DETAILED DESIGN
CODE
INTEGRATION AND TEST PLAN
DESIGN VALIDATION AND VERIFICATION
UNIT TEST
INTEGRA-TION AND TEST
ACCEPT-ANCE TEST
IMPLEMEN-TATION
Plan Next Phases
Develop, Verify Next Level Product

Timeboxing

- Iterative is linear sequence of iterations
- Each iteration is a mini waterfall – decide the specs, then plan the iteration
- Time boxing – fix an iteration duration, then determine the specs
- Divide iteration in a few equal stages
- Use pipelining concepts to execute iterations in parallel

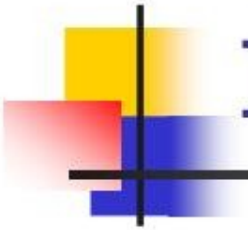
Time Boxed Iterations

- General iterative development – fix the functionality for each iteration, then plan and execute it
- In time boxed iterations – fix the duration of iteration and adjust the functionality to fit it
- Completion time is fixed, the functionality to be delivered is flexible

Time boxed Iteration

- This itself very useful in many situations
- Has predictable delivery times
- Overall product release and marketing can be better planned
- Makes time a non-negotiable parameter and helps focus attention on schedule
- Prevents requirements bloating
- Overall dev time is still unchanged

Timeboxing – Taking Time Boxed Iterations Further



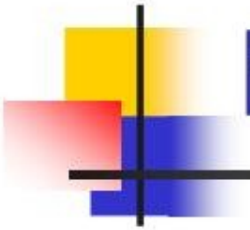
- What if we have multiple iterations executing in parallel
- Can reduce the average completion time by exploiting parallelism
- For parallel execution, can borrow pipelining concepts from hardware
- This leads to Timeboxing Process Model

Timeboxing Model – Basics

- Development is done iteratively in fixed duration time boxes
- Each time box divided in fixed stages
- Each stage performs a clearly defined task that can be done independently
- Each stage approximately equal in duration
- There is a dedicated team for each stage
- When one stage team finishes, it hands over the project to the next team

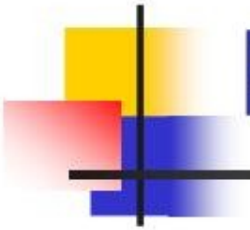
Timeboxing

- With this type of time boxes, can use pipelining to reduce cycle time
- Like hardware pipelining – view each iteration as an instruction
- As stages have dedicated teams, simultaneous execution of different iterations is possible



Example

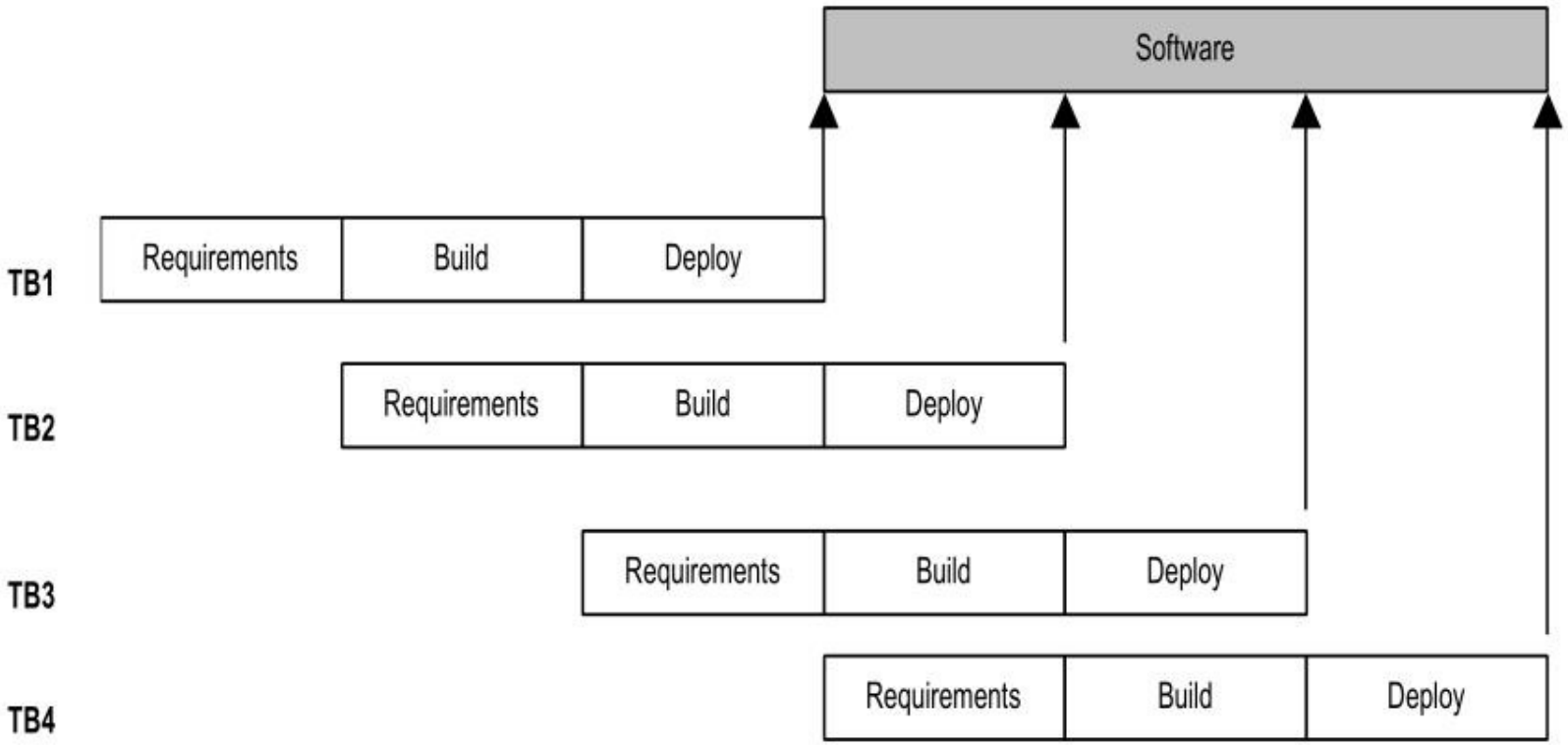
- An iteration with three stages – Analysis, Build, Deploy
  - These stages are appx equal in many situations
  - Can adjust durations by determining the boudaries suitably
  - Can adjust duration by adjusting the team size for each stage
- Have separate teams for A, B, and D



Pipelined Execution

- AT starts executing it-1
- AT finishes, hands over it-1 to BT, starts executing it-2
- AT finishes it-2, hands over to BT; BT finishes it-1, hands over to DT; AT starts it-3, BT starts it-2 (and DT, it-1)
- …

Timeboxing Execution



Timeboxing execution

- First iteration finishes at time T
- Second finishes at T+T/3; third at T+2 T/3, and so on
- In steady state, delivery every T/3 time
- If T is 3 weeks, first delivery after 3 wks, 2nd after 4 wks, 3rd after 5 wks,…
- In linear execution, delivery times will be 3 wks, 6 wks, 9 wks,…

Timeboxing execution

- Duration of each iteration still the same
- Total work done in a time box is also the same
- Productivity of a time box is same
- Yet, average cycle time or delivery time has reduced to a third

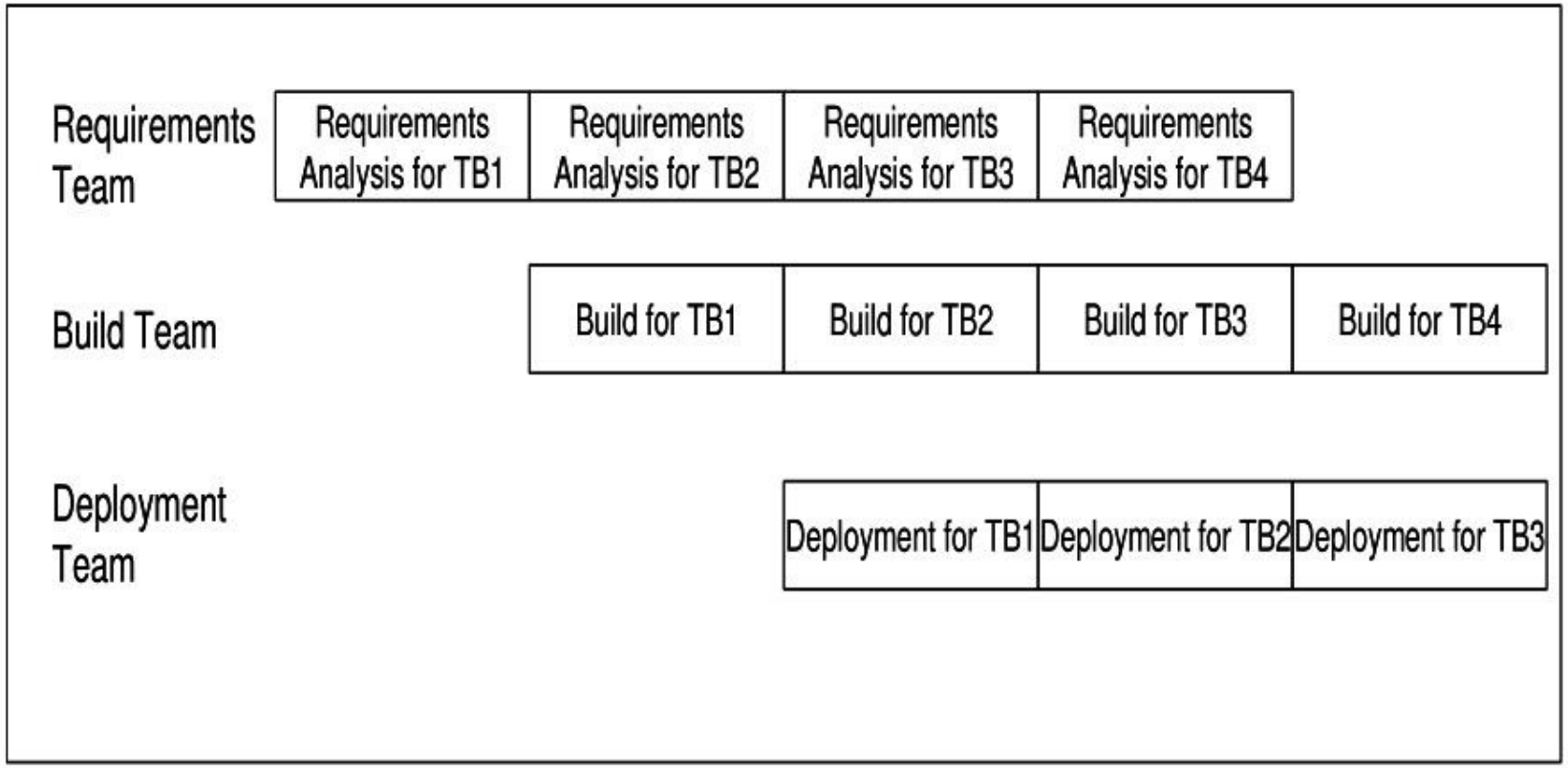
Team Size

- In linear execution of iterations, the same team performs all stages
- If each stage has a team of S, in linear execution the team size is S
- In pipelined execution, the team size is three times (one for each stage)
- I.e. the total team size in timeboxing is larger; and this reduces cycle time

Team Size

- Merely by increasing the team size we cannot reduce cycle time - Brook's law
- Timeboxing allows structured way to add manpower to reduce cycle time
- Note that we cannot change the time of an iteration – Brook's law still holds
- Work allocation different to allow larger team to function properly

Work Allocation of Teams



Timeboxing

- Advantages: Shortened delivery times, other adv of iterative, distr. execution
- Disadvantages: Larger teams, proj mgmt is harder, high synchronization needed, CM is harder
- Applicability: When short delivery times v. imp.; architecture is stable; flexibility in feature grouping

Summary – waterfall

Strength	Weakness	Types of Projects
Simple Easy to execute Intuitive and logical Easy contractually	All or nothing – too risky Req frozen early May chose outdated hardware/tech Disallows changes No feedback from users Encourages req bloating	Well understood problems, short duration projects, automation of existing manual systems

Summary – Prototyping

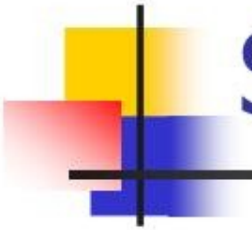
Strength	Weakness	Types of Projects
Helps req elicitation Reduces risk Better and more stable final system	Front heavy Possibly higher cost and schedule Encourages req bloating Disallows later change	Systems with novice users; or areas with req uncertainity. Heavy reporting based systems can benefit from UI proto

Summary – Iterative

Strength	Weakness	Types of Projects
Regular deliveries, leading to biz benefit Can accommodate changes naturally Allows user feedback Avoids req bloating Naturally prioritizes req Allows reasonable exit points Reduces risks	Overhead of planning each iteration Total cost may increase System arch and design may suffer Rework may increase	For businesses where time is imp; risk of long projects cannot be taken; req not known and evolve with time

Summary – Timeboxing

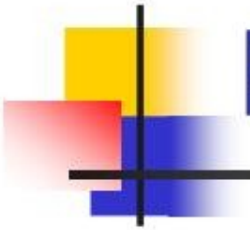
Strength	Weakness	Types of Projects
All benefits of iterative Planning for iterations somewhat easier Very short delivery times	PM becomes more complex Team size is larger Complicated – lapses can lead to losses	Where very short delivery times are very important Where flexibility in grouping features Arch is stable



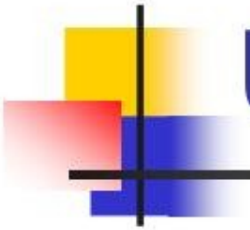
Summary

- Development process models discussed
  - Waterfall
  - Prototyping
  - Iterative
  - Timeboxing
- Each has its strengths and weaknesses and will work well for some types of projects

Using Process Model in a Project

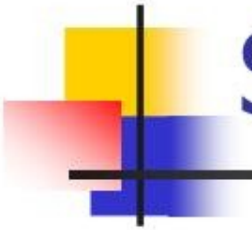


- Model to be used should be selected based on the nature of the problem
- Example: Build a small auction system for a Univ, tight schedule, some core req, customer time only in start,…
- Suitable model: Iterative delivery – do req in 1st iter; and two rounds of delivery; minimizes risk,…



Using Process Models..

- Example: Highly competitive product; req change rapidly; outsourcing is desired for reducing cost,…
- Model: XP not OK as collocated team needed; iterative may not deliver rapidly enough; timeboxing best suited



Summary

- Process is a means to achieve project objectives of high QP
- Process models define generic process, which can form basis of project process
- Process typically has stages, each stage focusing on an identifiable task
- Many models for development process have been proposed