



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

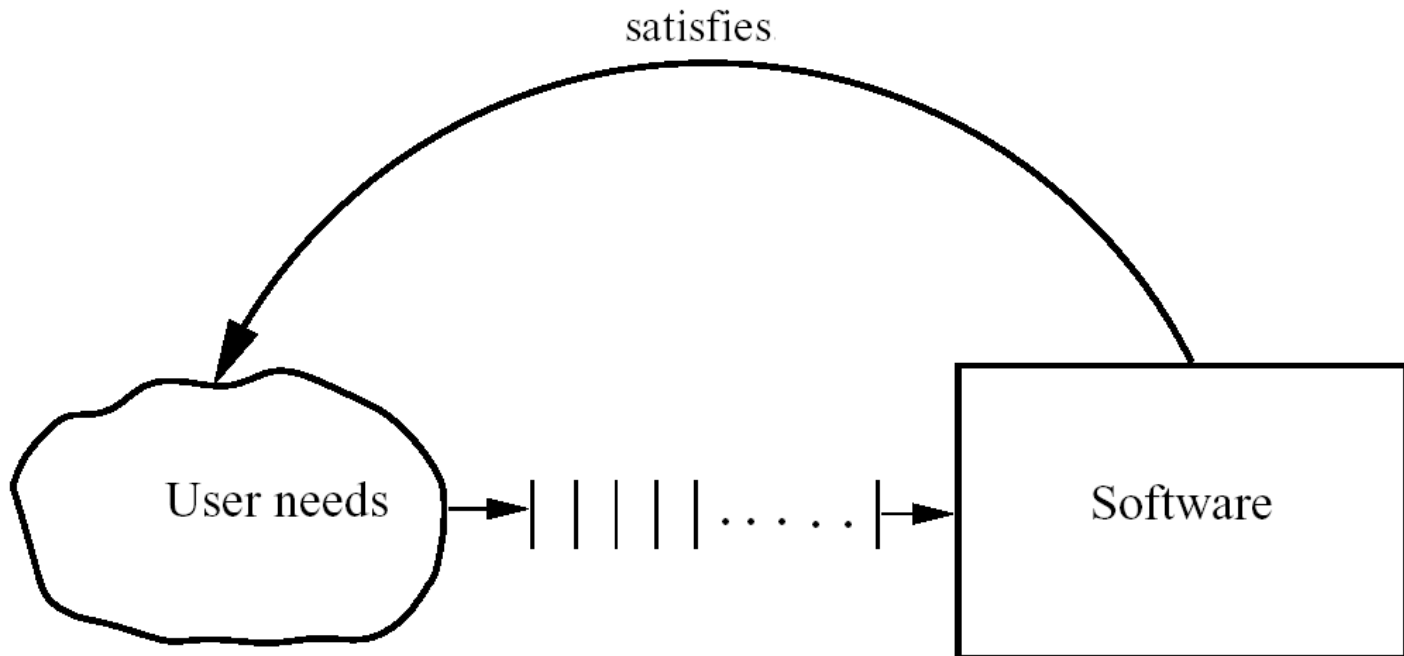
- Q&P is an essential goal
- Q&P depends on people, process, and technology
 - Processes help people become more productive and create fewer errors
 - Tools help people execute some tasks in the process more efficiently and effectively
 - So, process forms the core



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

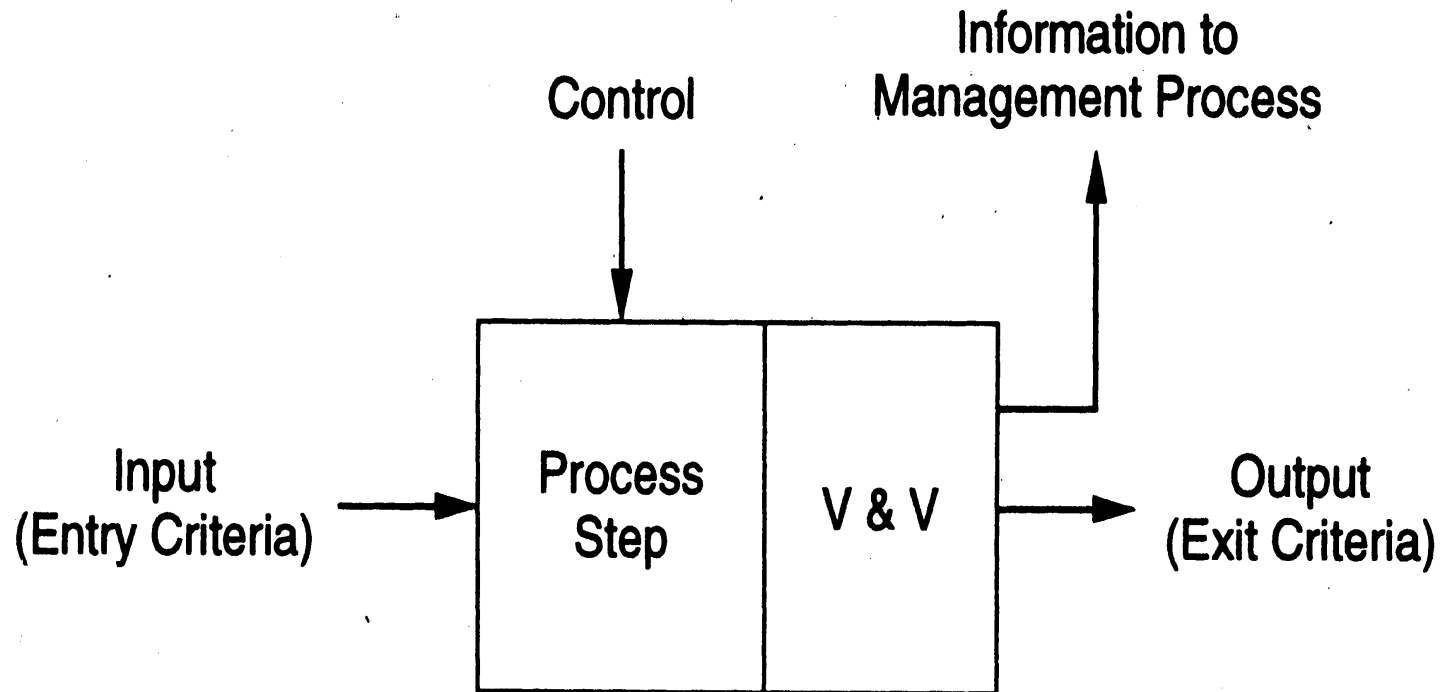
- Process is generally a set of phases
- Each phase performs a well defined task and generally produces an output
- Intermediate outputs – *work products*
- At top level, typically few phases in a process
- How to perform a particular phase – *methodologies* have been proposed



ETVX Specification

- ETVX approach to specify a step
 - Entry criteria: what conditions must be satisfied for initiating this phase
 - Task: what is to be done in this phase
 - Verification: the checks done on the outputs of this phase
 - eXit criteria: when can this phase be considered done successfully
- A phase also produces info for mgmt

ETVX approach



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



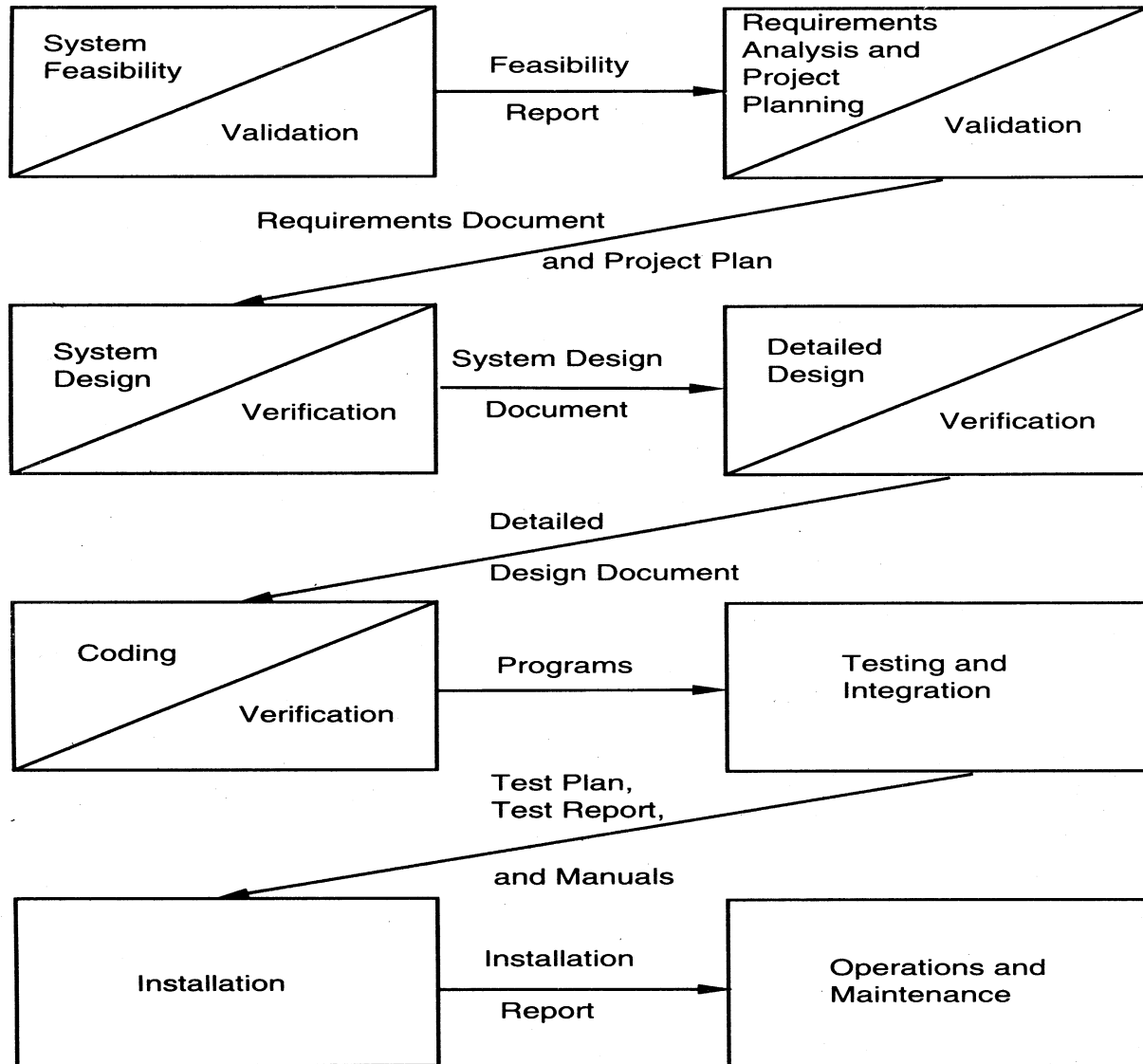
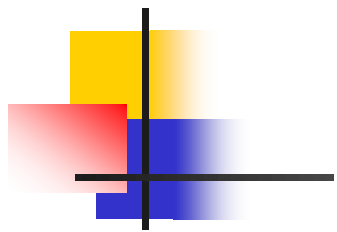
Process Models

- 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





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

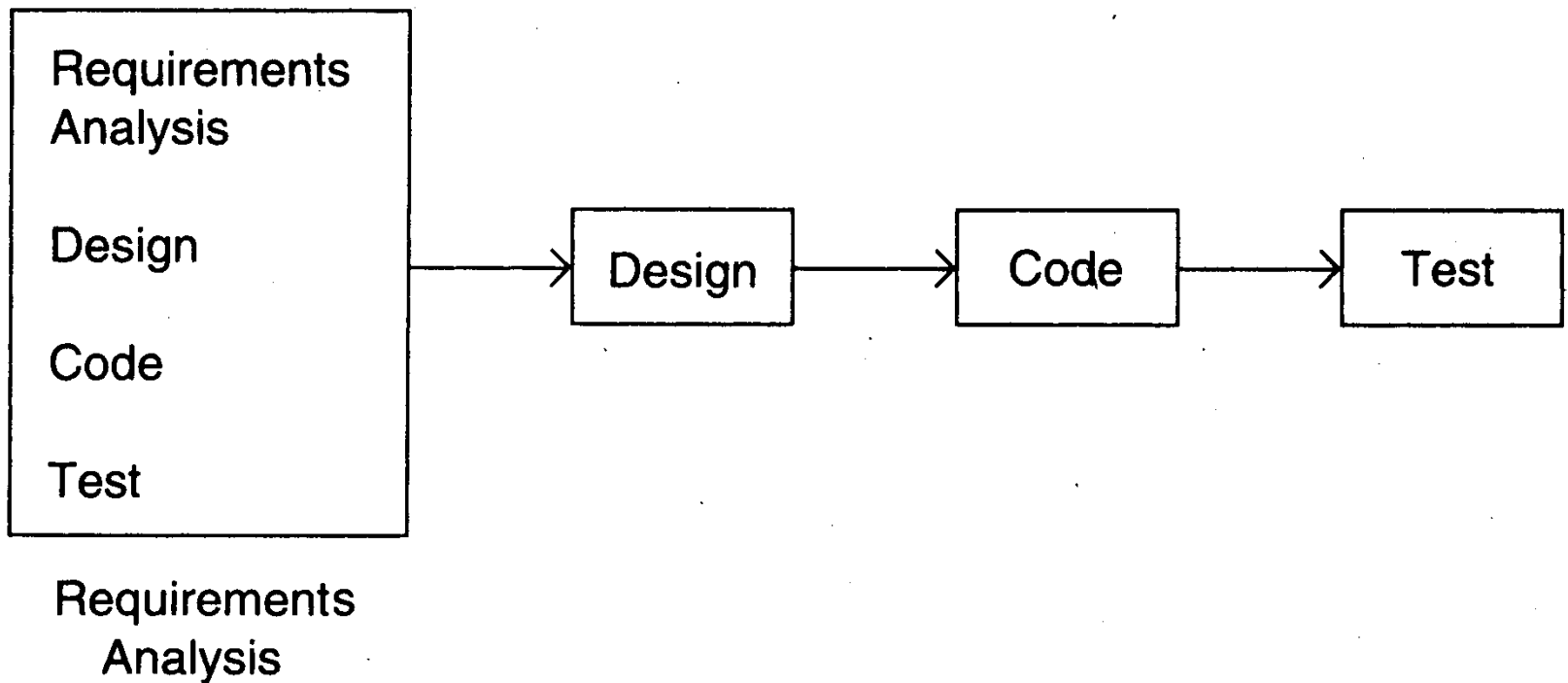
- Has been used widely
- Well suited for projects where requirements can be understood easily and technology decisions are easy
- I.e. for familiar type of projects it still may be the most optimum



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

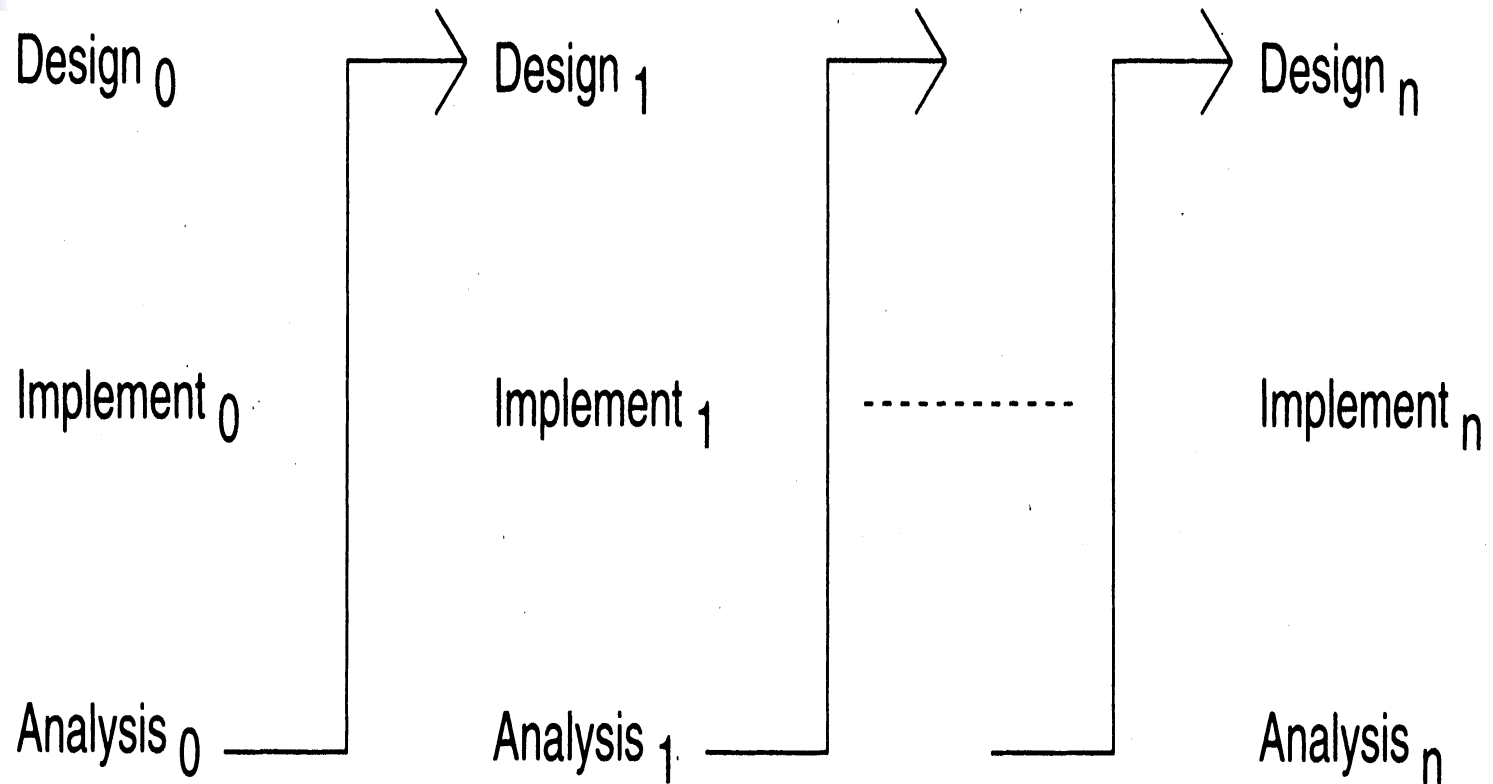
- Advantages: req will be more stable, req frozen later, experience helps in the main development
- Disadvantages: Potential hit on cost and schedule
- Applicability: When req are hard to elicit and confidence in reqs is low; i.e. where reqs are not well understood



Iterative Development

- Counters the “all or nothing” drawback of the waterfall model
- Combines benefit of prototyping and waterfall
- 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 Enhancement





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



Iterative Development

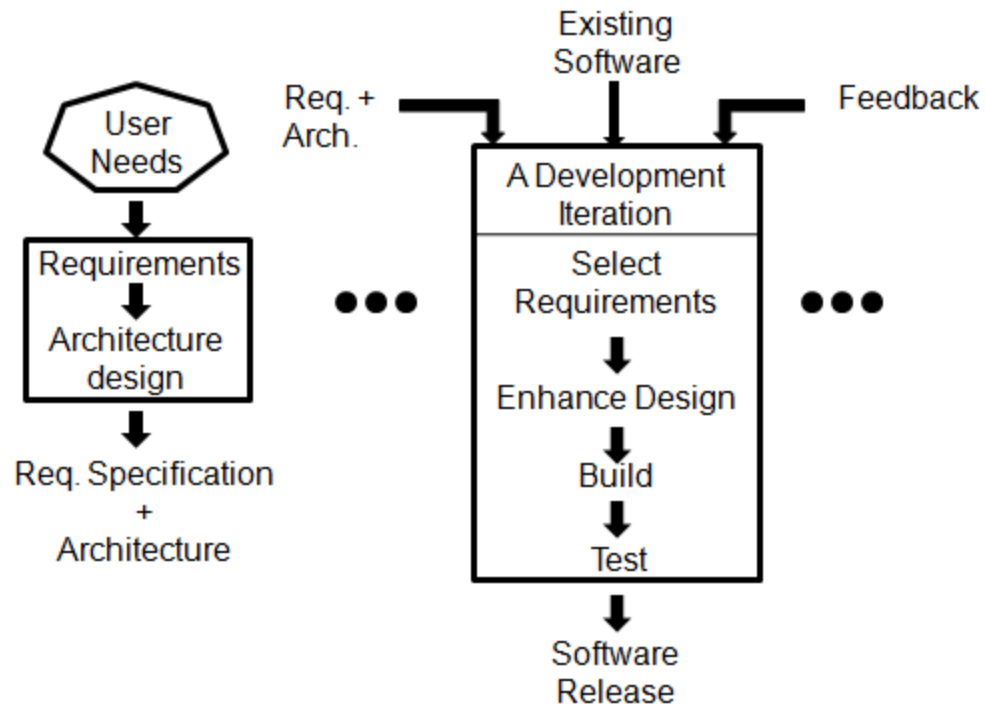
- Benefits: Get-as-you-pay, feedback for improvement,
- Drawbacks: Architecture/design may not be optimal, rework may increase, total cost may be more
- Applicability: where response time is important, risk of long projects cannot be taken, all req not known



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

Another form of Iteration...





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

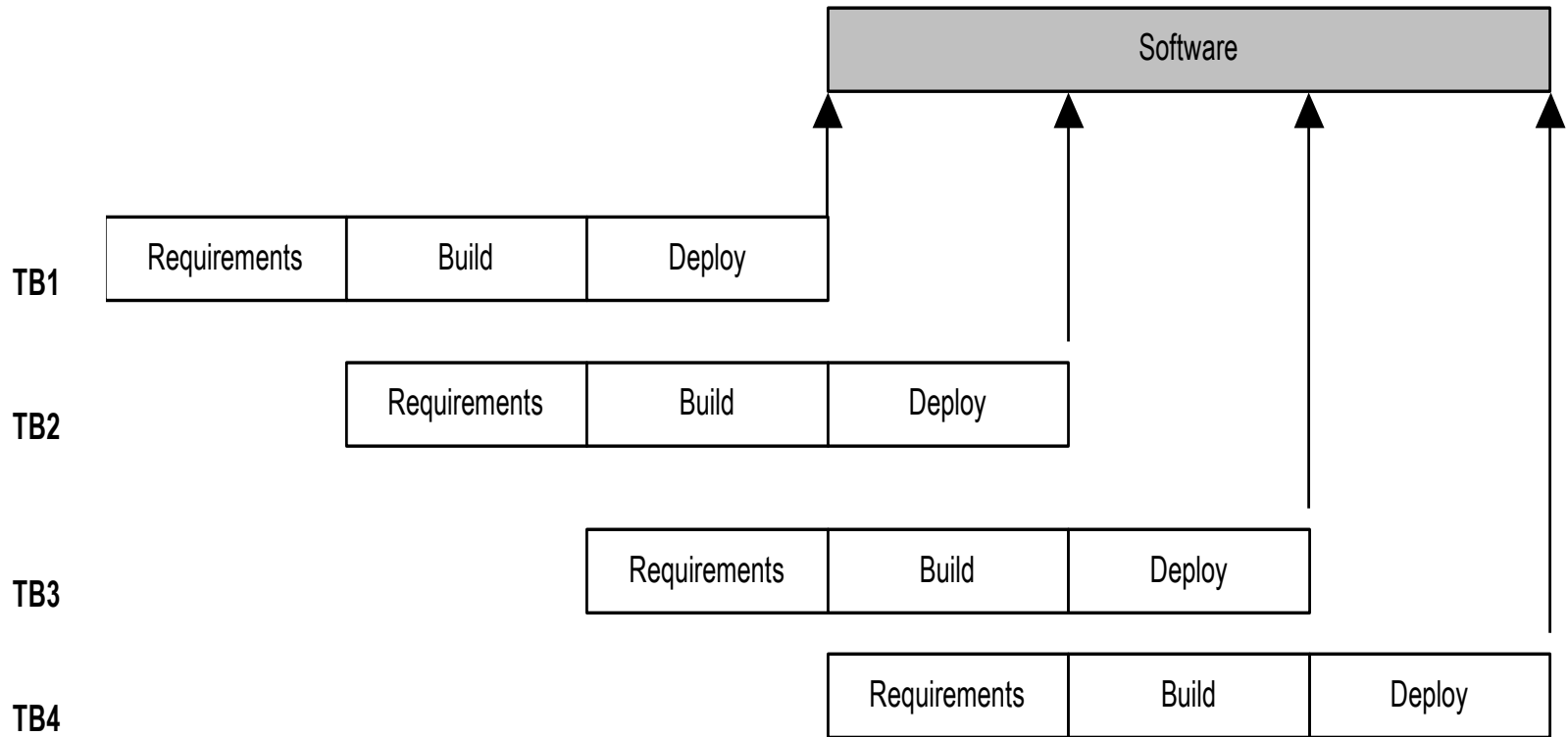
- An iteration with three stages – Analysis, Build, Deploy
 - These stages are appx equal in many situations
 - Can adjust durations by determining the boudaries suitably
 - Can adjust duration by adjusting the team size for each stage
- Have separate teams for A, B, and D



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 $T+2T/3$, 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

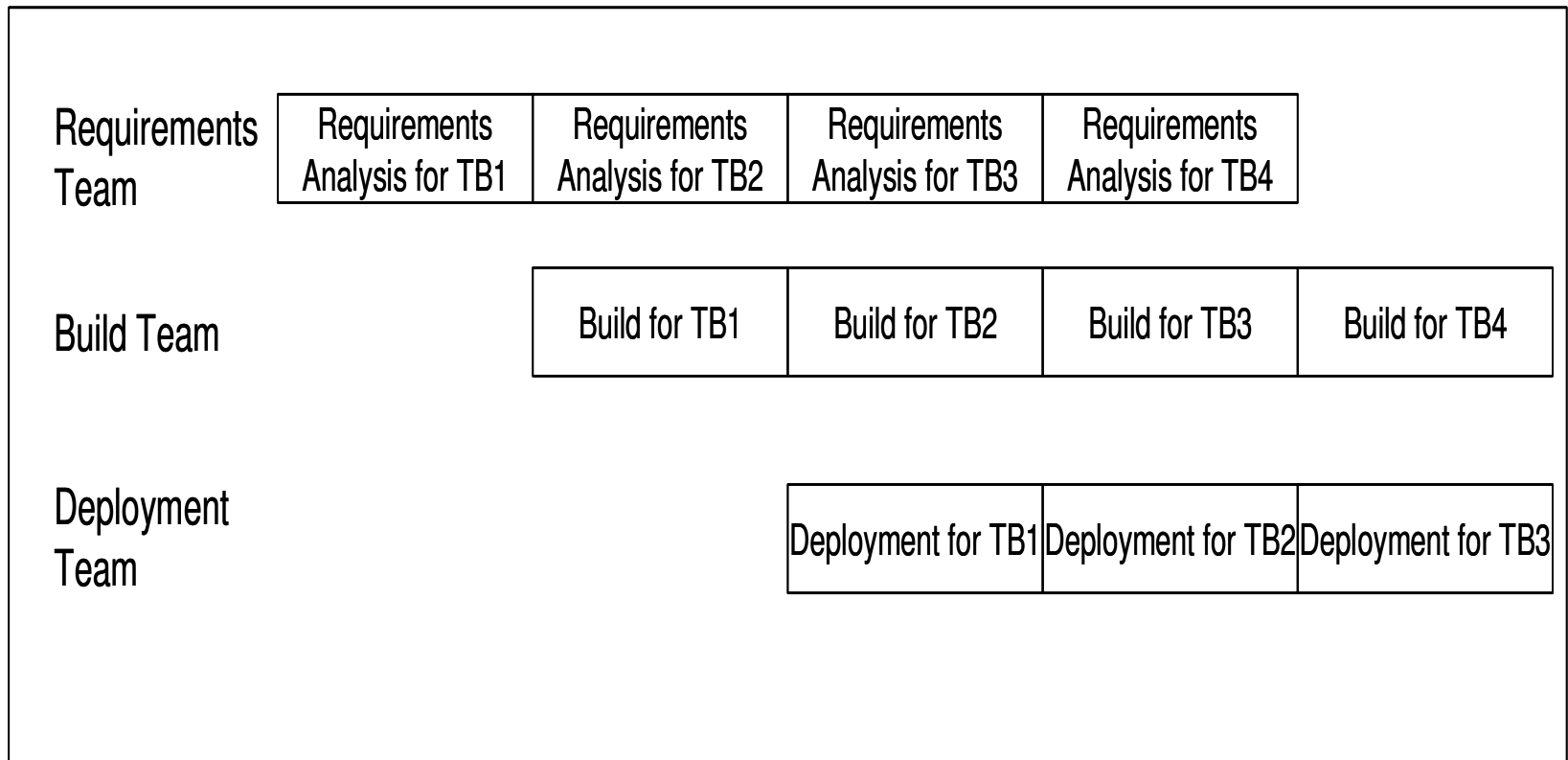


Team Size

- 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



RUP Model

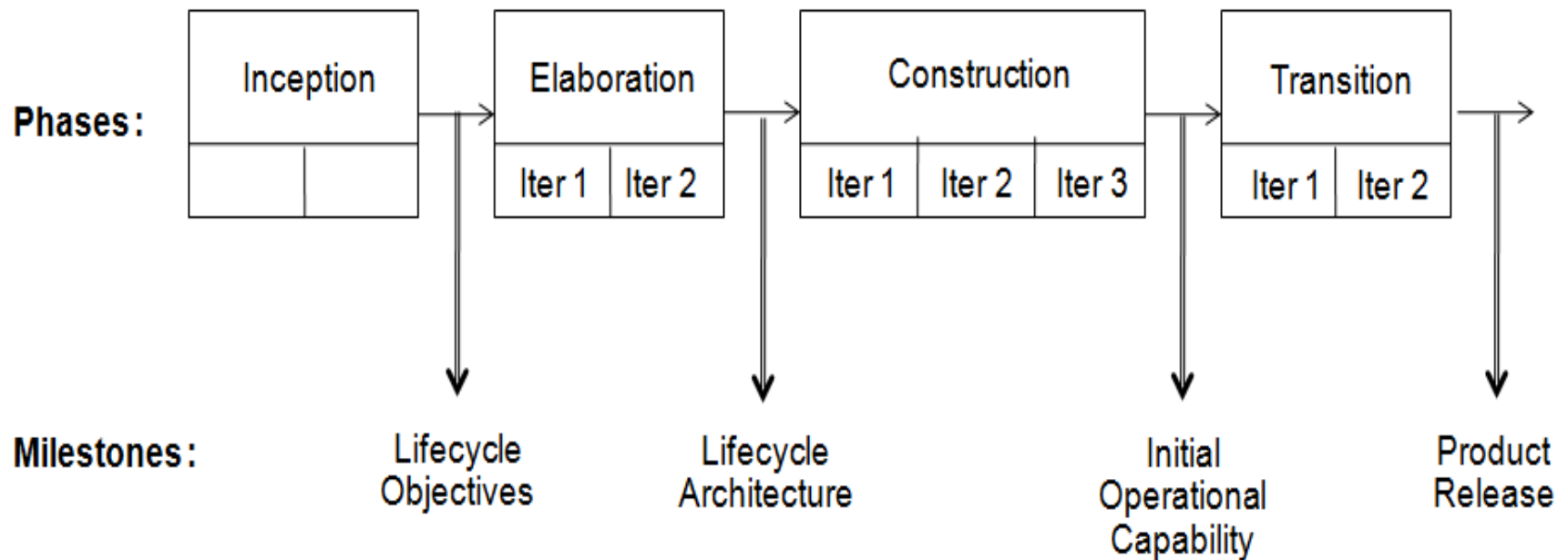
- Rational Unified Process is another iterative model
- Software development is divided into *cycles*, each cycle delivering a fully working system
- Each cycle executed as separate project
- Execution of a cycle is broken into four consecutive phases, each phase ending with a milestone achievement



Phases in a Project

- Phases in a project
 - Inception phase: ends with *Lifecycle Objectives milestone*; vision and high level capability of system defined
 - Elaboration phase: *Lifecycle architecture milestone*; most requirements defined and architecture designed
 - Construction phase: *Initial operational capability milestone*
 - Transition phase: *Product release*; transition product from development to production

Phases and Milestones





Execution of phases

- Each phase itself can be done in multiple iterations, each iteration having an external/internal customer
- Generally construction has multiple iterations; elaboration can also be meaningfully done in multiple iterations



Core workflows and phases

- Engineering tasks are called core process workflows
- These sub processes correspond to tasks of requirements, design, implementation, testing, proj mgmt, etc
- Many sub processes may be active in a phase, the volume of activity generally differs depending on the project



Sub processes and phases

PHASES	Inception	Elaboration	Construction	Transition
Requirements	High	High	Low	Nil
Anal. & Design	Low	High	Medium	Nil
Implementation	Nil	Low	High	Low
Test	Nil	Low	High	Medium
Deployment	Nil	Nil	Medium	High
Proj. Mgmt	Medium	Medium	Medium	Medium
Config. Mgmt	Low	Low	High	High

→ Time



RUP

- Sub processes are active in all phases
- Volume of activity in each phase differs depending on the project
- Hence, a project can use RUP to implement waterfall by having requirements process be active only in the elaboration phase
- Or prototyping by having a lot of construction activity in the elaboration phase
- RUP is therefore a flexible framework

Extreme Programming or Agile Process Model



- Agile approaches developed in 90s as a reaction to document driven approaches
- Most agile approaches have some common principles
 - Working software is the measure of progress
 - Software should be delivered in small increments
 - Even late changes should be allowed
 - Prefer face to face commn over documentation
 - Continuous feedback and customer involvement is necessary
 - Prefer simple design which evolves
 - Delivery dates are decided by the empowered teams
 - ...



XP...

- Many agile methodologies have been proposed; extreme programming (XP) is one of the most popular
- An XP project starts with user stories, which are short descr of user needs
 - Details are ***not*** included
 - Each user story written on a separate card so they can be combined in diff ways



Overall Process

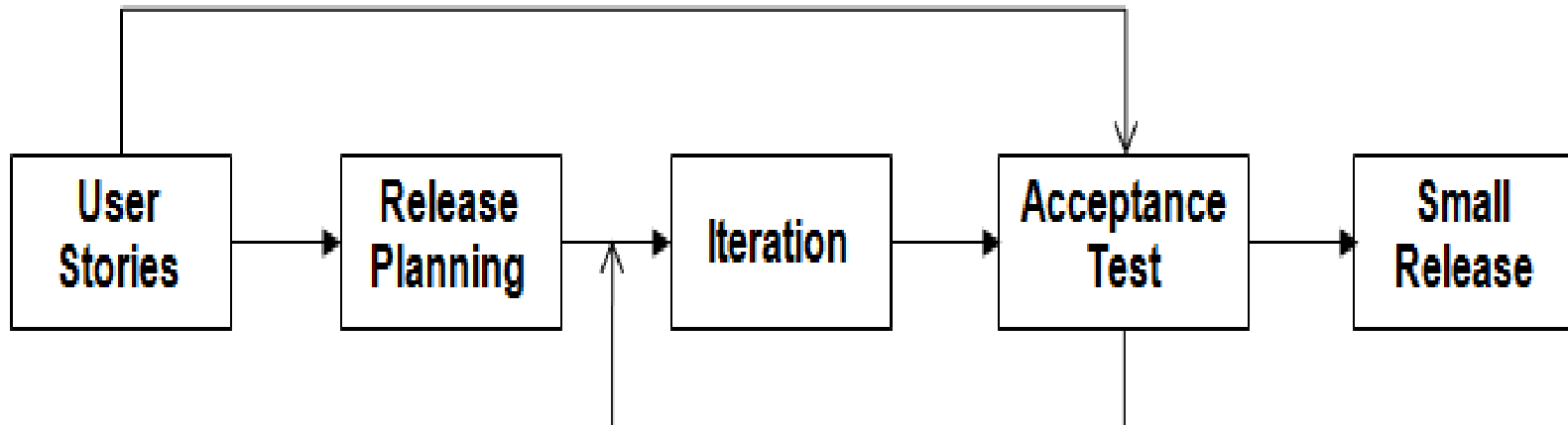
- Team estimates how long it will take to implement a user story
 - Estimates are rough
- Release planning is done
 - Defines which stories are to be built in which release, and dates for release
 - Frequent and small releases encouraged
 - Acceptance tests also built from user stories; used to test before release
 - Bugs found in AT are fixed in next release



Overall Process

- Development done in iterations of a few weeks each
 - Iteration starts with planning, in which stories to be implemented are selected – high risk high value are chosen first
 - Details of stories obtained during the development and implemented
 - Failed AT of previous iteration are also fixed

XP – Overall Process

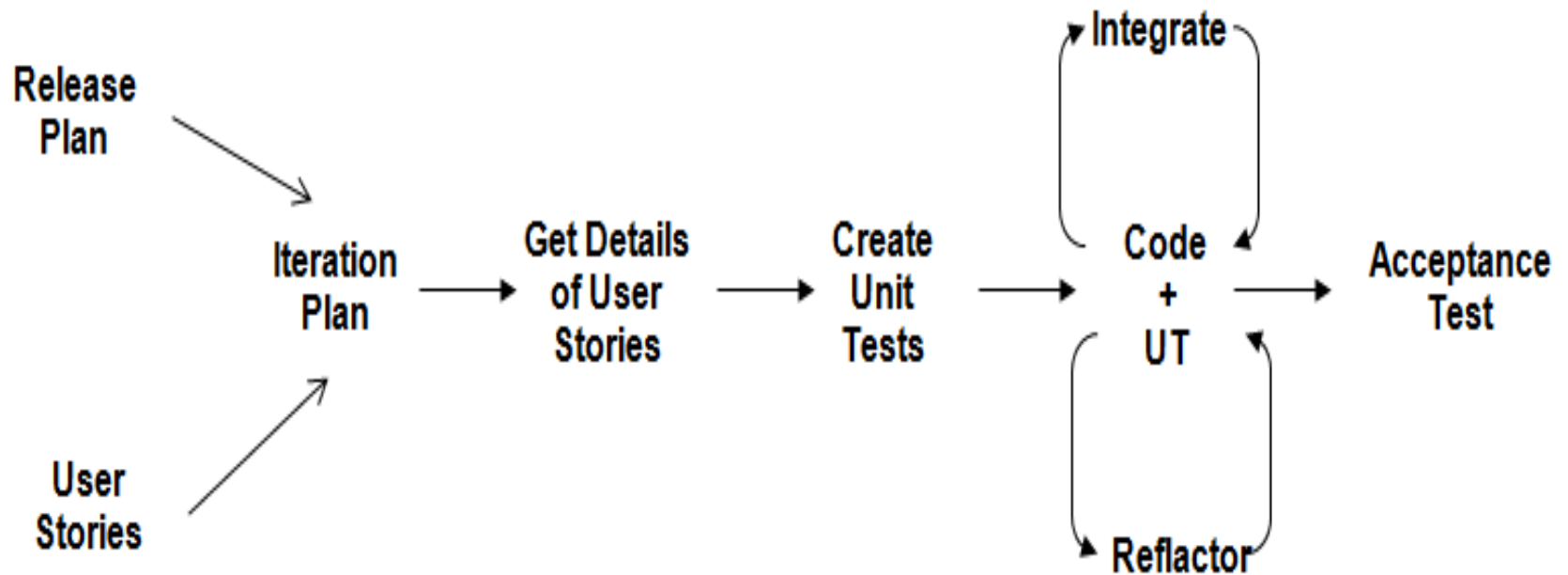




An Iteration

- An iteration execution has some unique practices
 - Pair programming: programming is done in pairs of programmers
 - Test driven development – automated unit tests written before the code
 - Simple solutions, refactoring for improving the design when need arises
 - Frequent integration

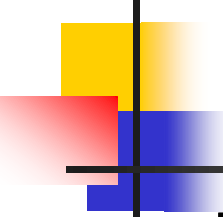
An Iteration





XP - Summary

- Well suited for situations where volume and pace of requirements is high
- Customer is willing to engage heavily with the team
- The team is collocated and is not too large (less than 20 or so)
- Requires strong capability in team members



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 uncertainty. 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 – RUP

Strength	Weakness	Types of Projects
All benefits of iterative Provides a flexible framework for a range of projects	For each project, one has to design the process	Can be applied to a wide range as it allows flexibility



Summary – XP

Strength	Weakness	Types of Projects
Agile and responsive Short delivery cycles Continuous feedback can lead to better acceptance	Can tend to become ad-hoc Lack of documentation can be an issue Continuous code change is risky	Where requirements are changing a lot, customer is deeply engaged in development, and where the size of the project is not too large



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



Summary

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



Project Management Process



Background

- Development process divides development into phases and activities
- To execute it efficiently, must allocate resources, manage them, monitor progress, take corrective actions, ...
- These are all part of the PM process
- Hence, PM process is an essential part of executing a project



PM Process Phases

- There are three broad phases
 - Planning
 - Monitoring and control
 - Termination analysis
- Planning is a key activity that produces a plan, which forms the basis of monitoring



Planning

- Done before project begins
- Key tasks
 - Cost and schedule estimation
 - Staffing
 - Monitoring and risk mgmt plans
 - Quality assurance plans
 - Etc.
- Will discuss planning in detail later



Monitoring and control

- Lasts for the duration of the project and covers the development process
 - Monitors all key parameters like cost, schedule, risks
 - Takes corrective actions when needed
 - Needs information on the dev process – provided by metrics

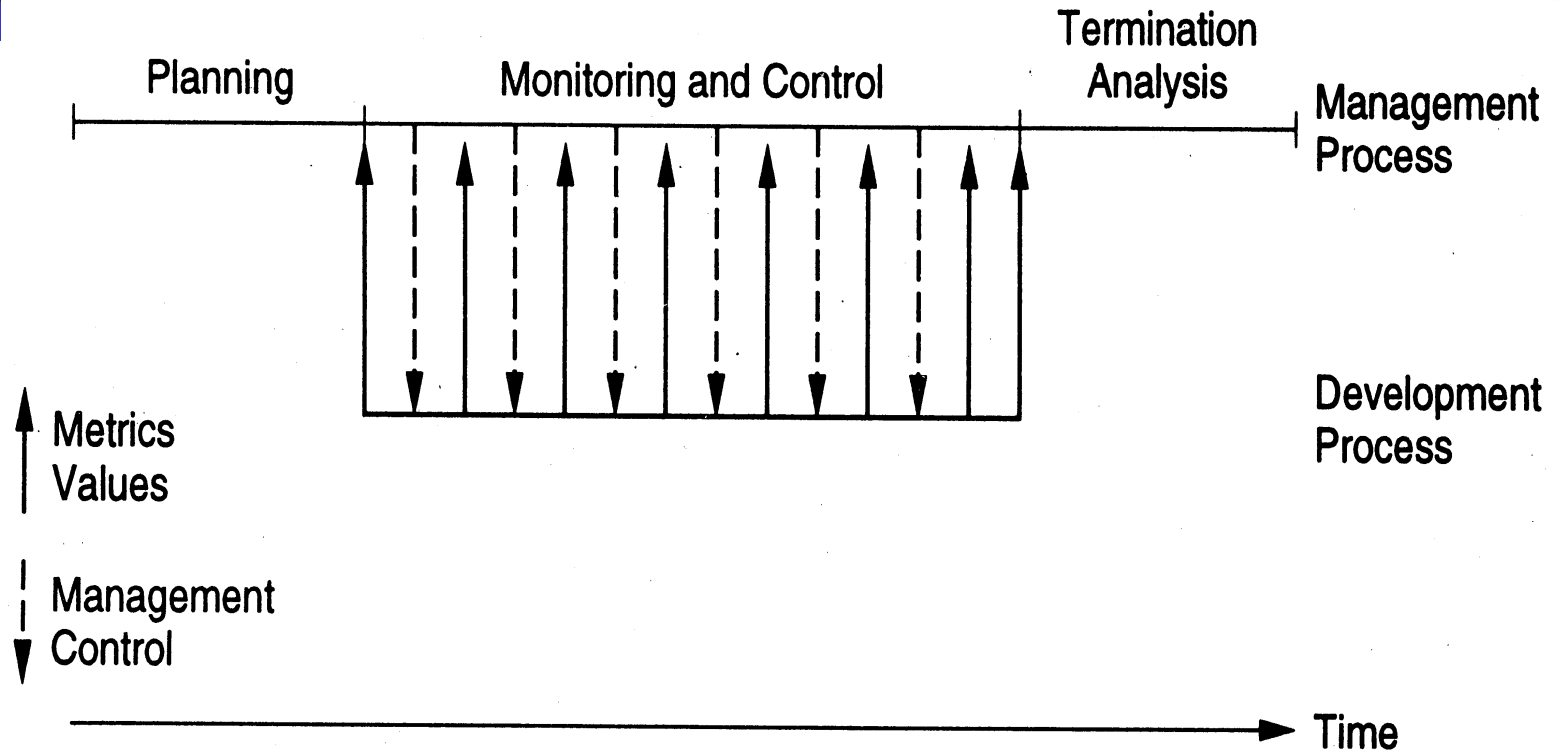


Termination Analysis

- Termination analysis is performed when the development process is over
- Basic purpose: to analyze the perf of the process, and identify lessons learned
- Also called postmortem analysis



Relationship with Dev Process





Summary

- Process has a major impact on the quality and productivity
- Different processes at work in a software project
- We have focused on development process and project management process
- Process models are general process structures, that work well for some types of problems
- A project should select a process model that is best suited for it (and tailor it to meet its requirements)