Software Engineering and Project Management
BCOL19209

Class SY. B.TECH

Name of Teacher: Ms. Nivedita Rawte

BCOL19209 Software Engineering and Project Management

Teaching Scheme:	Credit	Examination Scheme
Lectures: 2Hrs/Week **Tutorials:** 0Hr/Week	**02**	**TAE: 10 Marks** **CAE : 10 Marks** **ESE : 30**

Prerequisite(If any): NA

Course Outcome: On completion of this course graduate shall be able to

1. Describe & adapt the life cycle of software development process
2. List requirements & prepare SRS document for a given problem.
3. Apply various modeling techniques to design a system
4. Apply appropriate cost estimation techniques for the development of software.

- What is Software?
- What are the characteristics of Software?
- Nature of Software
- Software Myths
- Software Engineering Practice
- Framework Activities
- Umbrella Activities

What is Software?

- Set of programs
- Data Structures
- Documentation

- **What are the characteristics of Software?**
  - Logical rather than Physical
  - Software is developed or engineered, it is not manufactured (Quality, People, Process, Cost)
  - Software does not wear out (Bathtub Curve)
  - Moving towards component based construction, however most of the software is custom built

Hardware vs. Software

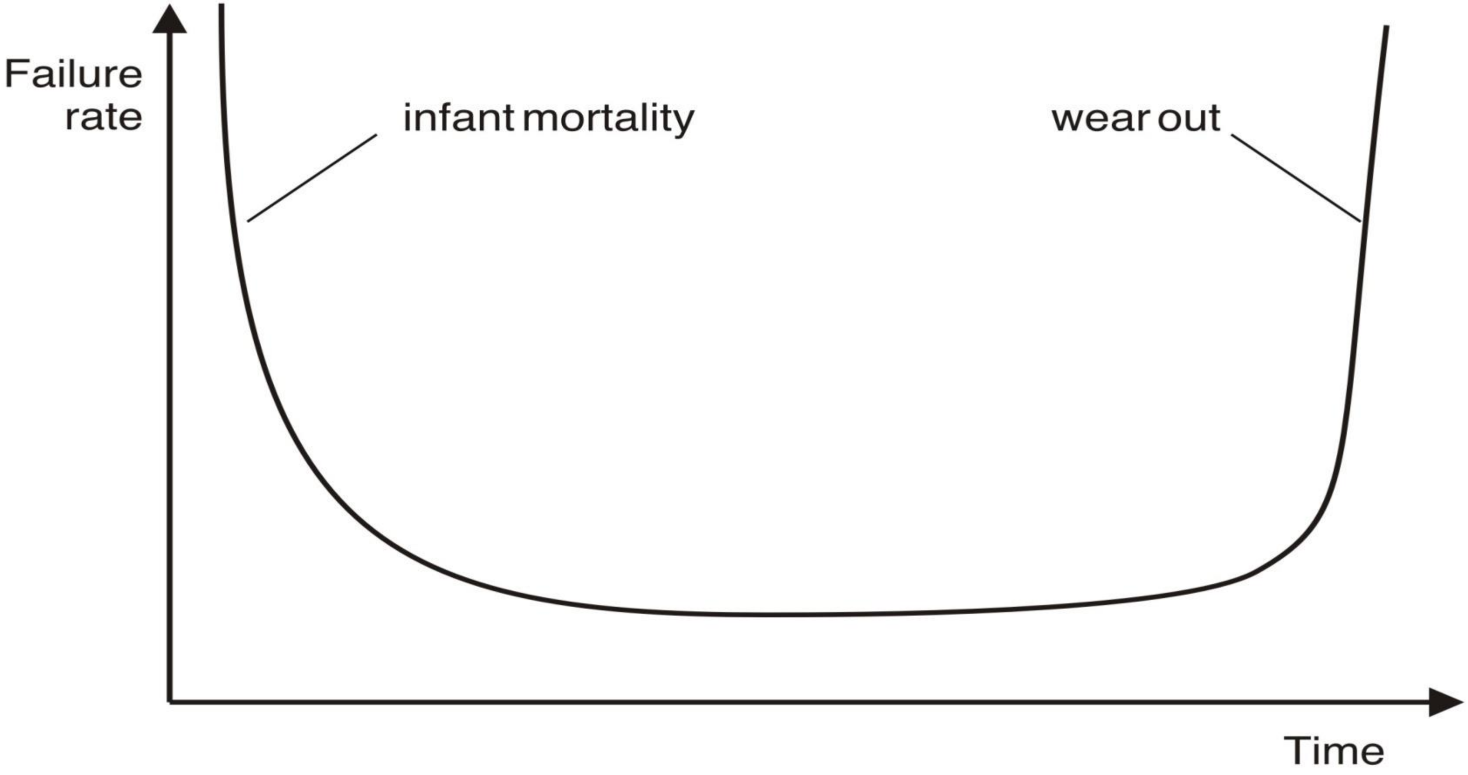
Hardware	Software
▪ Manufactured ▪ Wears out ▪ Built using components ▪ Relatively simple	▪Developed/Engineered ▪ Deteriorates ▪ Custom built ▪ Complex

Manufacturing vs. Development

- Once a hardware product has been manufactured, it is difficult or impossible to modify. In contrast, software products are routinely modified and upgraded.
- In hardware, hiring more people allows you to accomplish more work, but the same does not necessarily hold true in software engineering.
- Unlike hardware, software costs are concentrated in design rather than production.

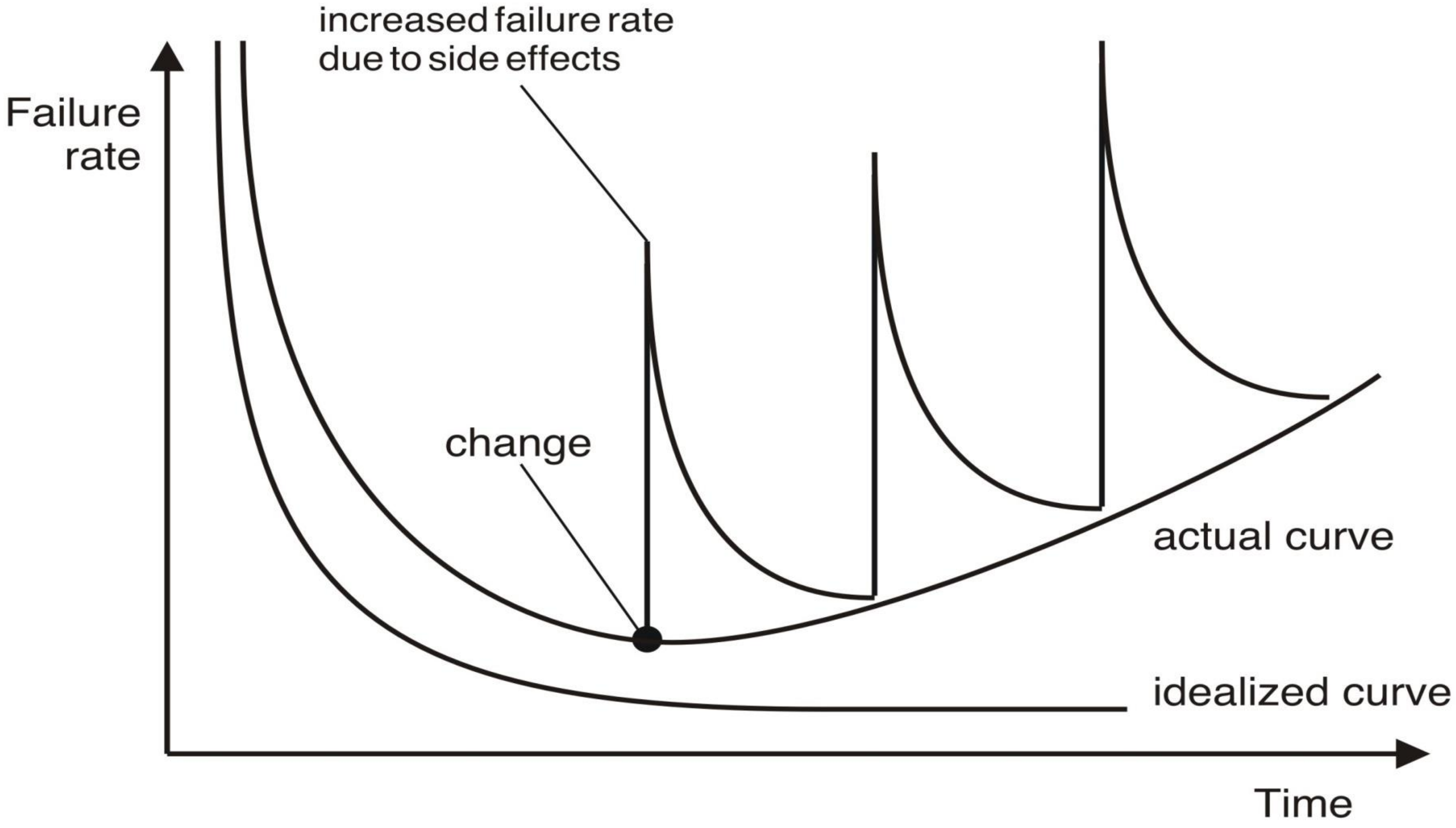
Wear vs. Deterioration

Hardware wears out over time



Wear vs. Deterioration

Software deteriorates over time



Software Complexity

"I believe the hard part of building software to be the specification, design, and testing of this conceptual construct, not the labor of representing it and testing the fidelity of the representation".
If this is true, building software will always be hard. There is inherently no silver bullet.

- Fred Brooks, "No Silver Bullet"

http://www.computer.org/computer/homepage/misc/Brooks/

- **Nature of Software**
  - Systems Software
  - Applications Software
  - Engineering / Scientific software
  - Embedded Software
  - Product Line Software
  - Web Applications
  - Artificial Intelligence Software
  - Ubiquitous Software
  - Net sourcing
  - Open Source

- **Software Myths**
    - Management Myths
    - Customer Myths
    - Practitioners Myth

Software Myths

- **Management Myths**
  - We are already having a book that is full of Standards & Procedures for building software. It will provide my people everything they need to know.

  - If we get behind schedule, we can add more developers & catch up.

  - We can outsource the software project to a third party. I can just relax & let that firm build it.

Software Myths

Customer Myths

 A general statement of objectives is sufficient to begin writing programs. We can fill in the details later

 Project requirements continually change, but the change can be easily accommodated because software is flexible

Software Myths

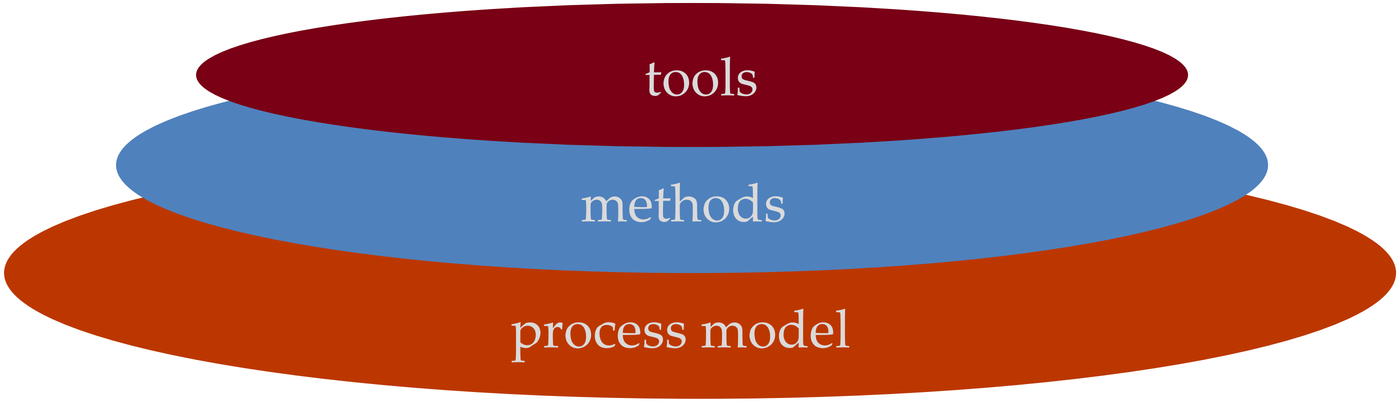
- **Practitioners Myth**
  - Once we write the program and get it to work our job is done
  - Until I get the program running, I have no way of accessing its quality
  - The only deliverable work product for a successful project is the working program
  - Software Engineering will make us create voluminous & unnecessary documentation & invariably slow us down

Software Engineering – A layered Technology

- **Software Engineering** is the application of systematic, disciplined, quantifiable approach to the development, operation and maintenance of software i.e. the application of engineering to software [IEEE definition]

A Layered Technology

Software Engineering



a “quality” focus

Software Engineering Layers

Quality Focus : Organizational commitment to quality (TQM, Six Sigma)

Process : Defines a framework (Foundation for Software Engineering)

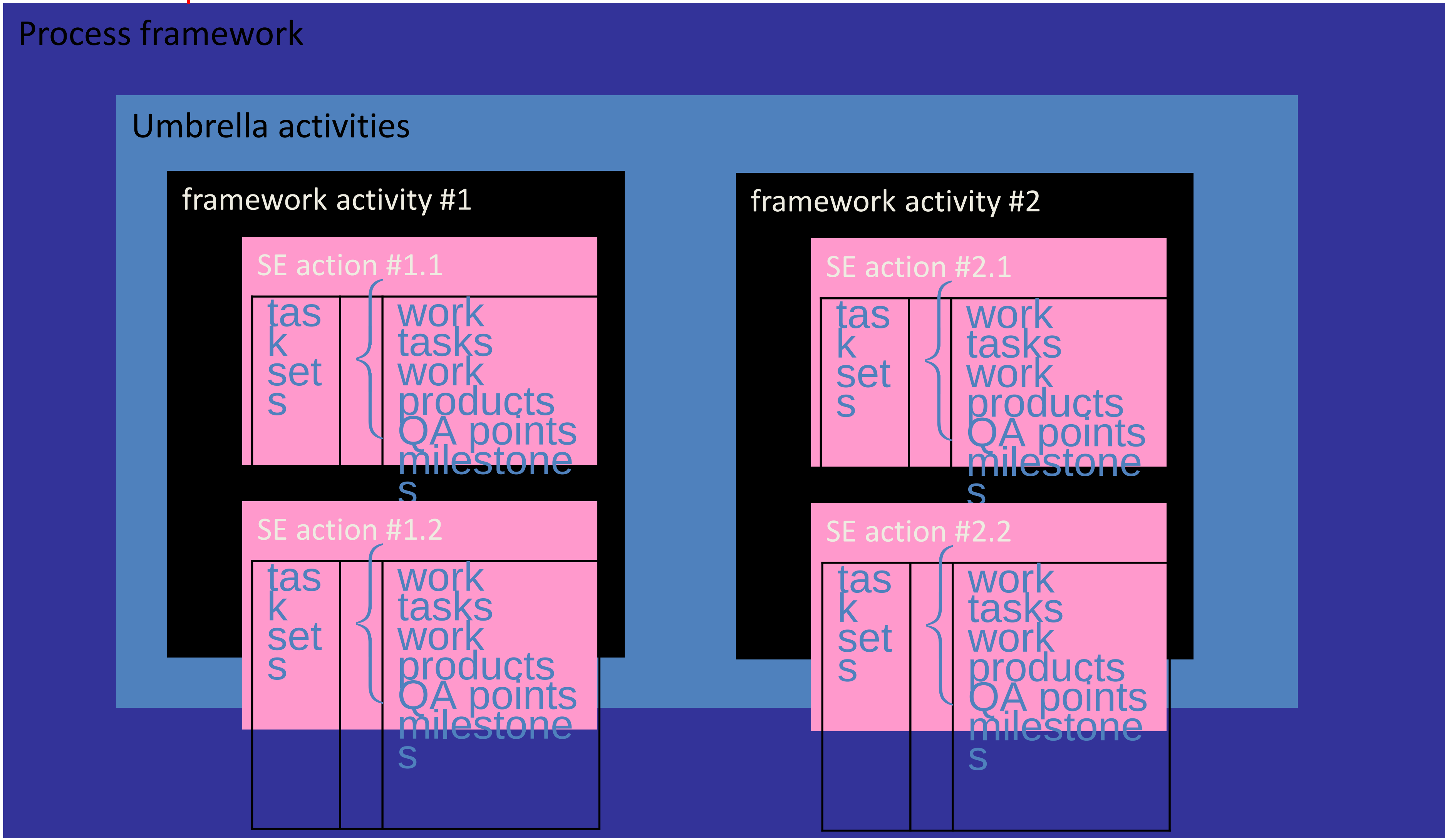
Methods : “How to”s for building software (Tasks)

Tools : Automated or semi-automated support (Rational Rose, CASE tools)

- Framework Activities
  - Communication
  - Planning
  - Modeling
  - Construction
  - Deployment

A Process Framework

Software process



Process framework

Umbrella activities

framework activity #1

SE action #1.1

task sets

work tasks

work products

QA points

milestones

SE action #1.2

task sets

work tasks

work products

QA points

milestones

framework activity #2

SE action #2.1

task sets

work tasks

work products

QA points

milestones

SE action #2.2

task sets

work tasks

work products

QA points

milestones

Umbrella Activities

- Software Project Tracking & Control
- Risk Management
- Software Quality Assurance
- Formal Technical Reviews
- Measurement
- Software Configuration Management
- Reusability Management

Software Engineering Practice

- Practice is collection of Concepts, Principles, Methods & Tools that a software engineer calls upon on a daily basis.
- Practice allows Managers to manage software projects & Software Engineers to build computer programs.

The Essence of SE Practice

1. Understand the Problem – Communication & Analysis
2. Plan a Solution – Modeling & Software Design
3. Carry out the Plan – Code generation
4. Examine the Results – Testing & QA

Core Principles

1. The Reason it all Exists
2. Keep It Simple, Stupid (KISS!)
3. Maintain the Vision
4. What you Produce, others will Consume
5. Be Open to Future
6. Plan Ahead for Reuse
7. THINK

Software process model

- **Attempt to organize the software life cycle by**
  - defining activities involved in software production
  - order of activities and their relationships

- **Goals of a software process**
  - standardization, predictability, productivity, high product quality, ability to plan time and budget requirements

Code & Fix

The earliest approach

- Write code
- Fix it to eliminate any errors that have been detected, to enhance existing functionality, or to add new features
- Source of difficulties and deficiencies
  - impossible to predict
  - impossible to manage

Process Models

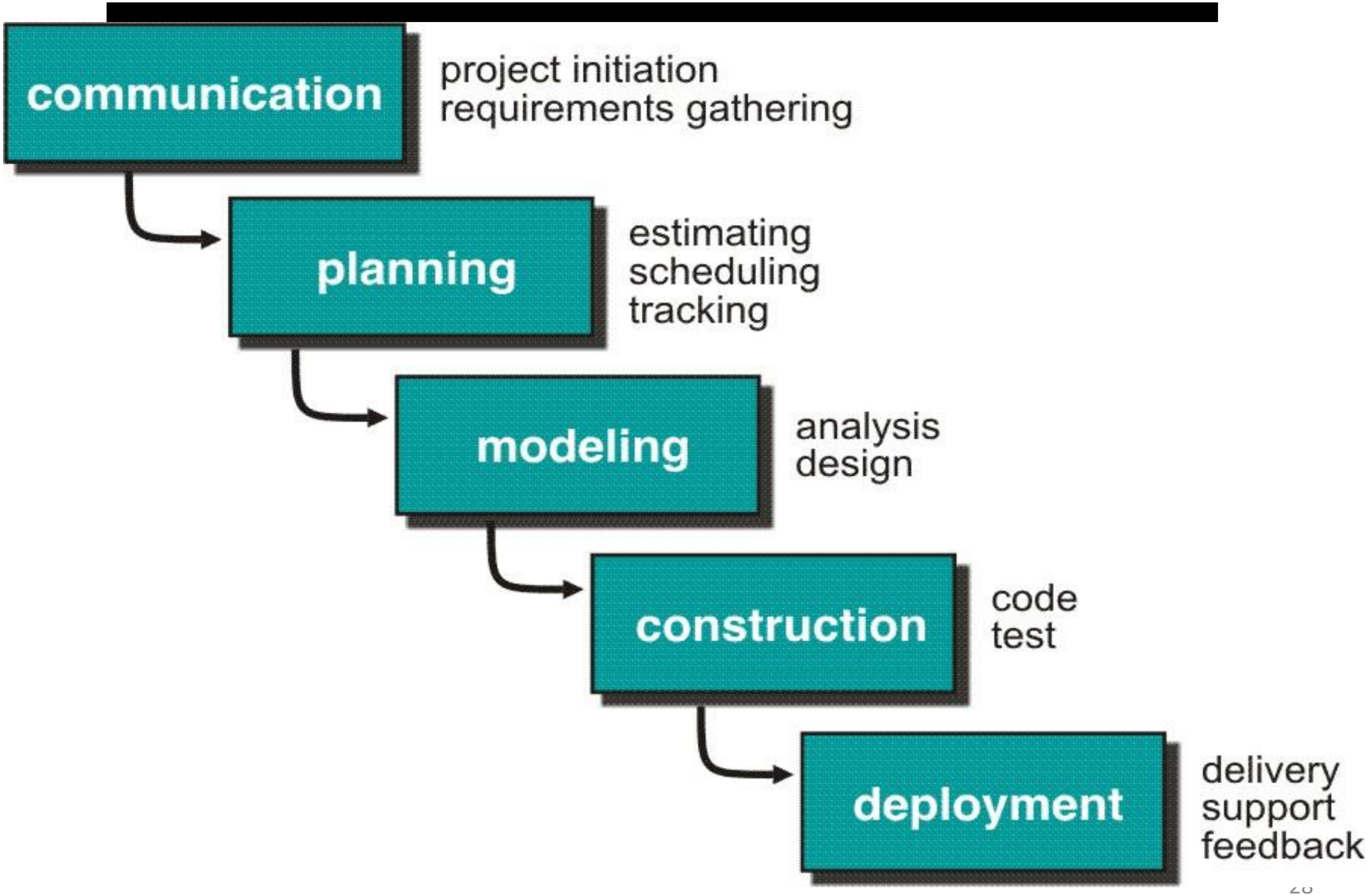
Prescriptive Process Models

- The Waterfall Model
- Incremental Model
- The RAD Model

Evolutionary Process Models

- The Prototyping Model
- The Spiral Model
- The Concurrent Development Model

The Waterfall Model



communication
project initiation
requirements gathering

planning
estimating
scheduling
tracking

modeling
analysis
design

construction
code
test

deployment
delivery
support
feedback

The Waterfall Model: (Payroll System)

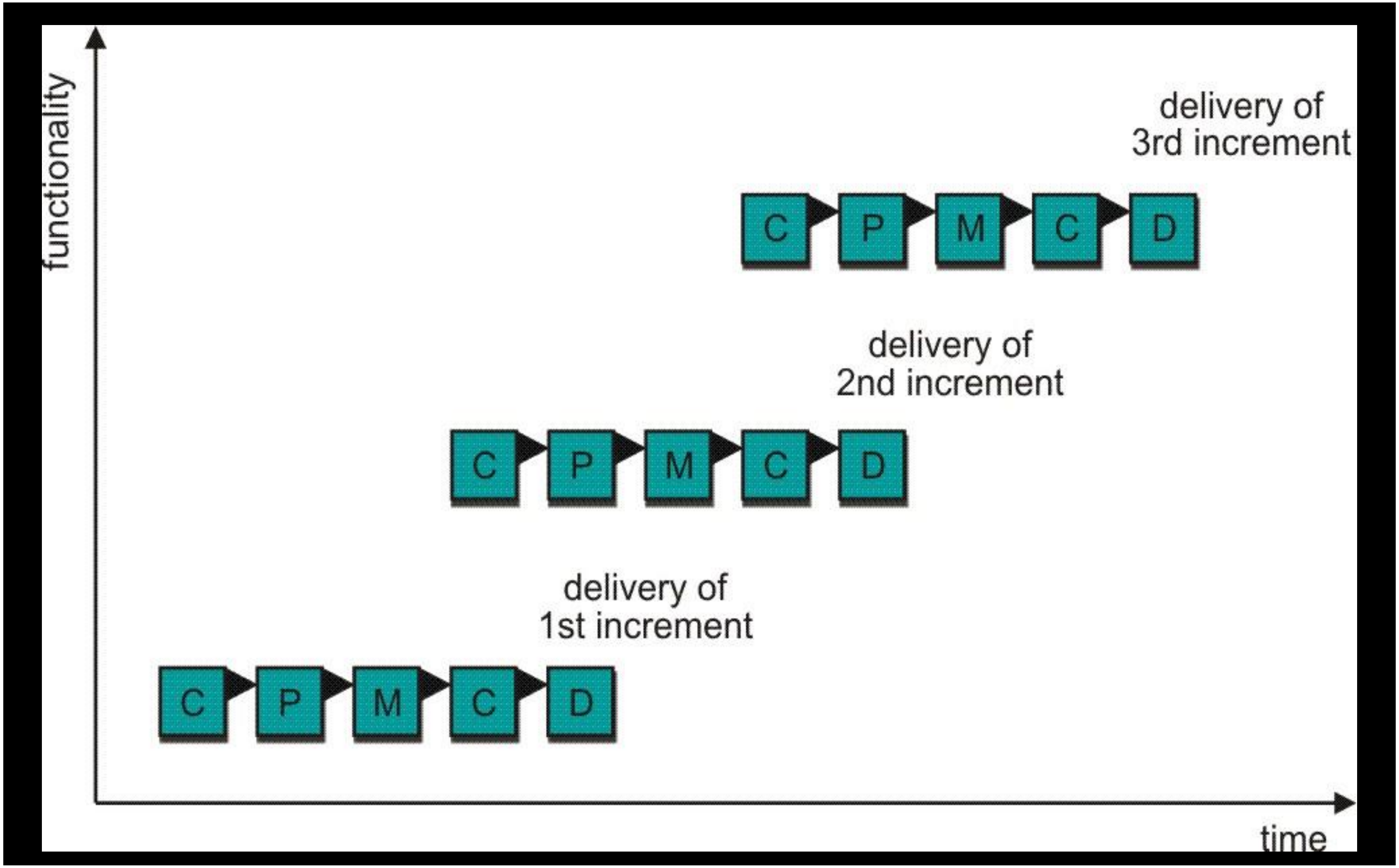
Merits :

- It is systematic sequential approach for Software Development

Demerits

- All Customer Requirements at the start of project may be difficult
- Problems remain uncovered until testing phase
- Customer patience is needed, working version of the software is delivered too late.

Incremental Models: Incremental



Incremental Model: (Word Processor)

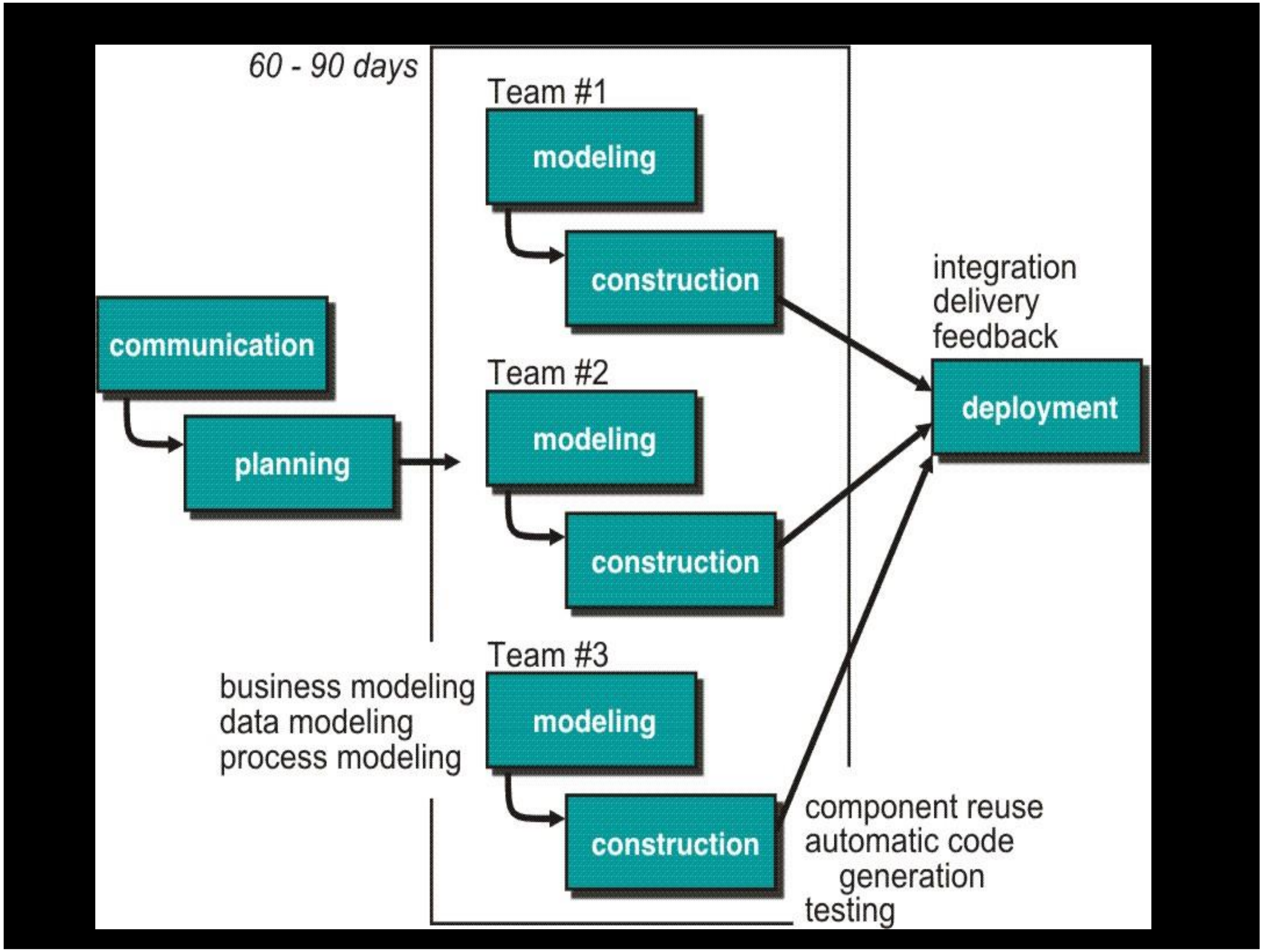
Merits :

- **Less number of developers required**
- **All the requirements need not be known at the beginning of the project**
- **Technical risks can be managed**

Demerits :

- **Problems remain uncovered until testing phase**
- **Customer patience is needed, working version of the software is delivered too late.**

Incremental Models: RAD Model



The RAD Model : (Very Large Projects)

Merits :

- Project cycle time is reduced

Demerits :

- All Customer Requirements at the start of project may be difficult
- For large projects high human resources are required
- Risk of project failure if teams are not committed to rapid fire action
- Problems due to improper modularization of system
- RAD approach may not work if high performance is an issue.
- RAD may not be appropriate if technical risks are very high.

Evolutionary Process Models: Need

- Software like all complex systems evolves over a period of time.

- Target market deadlines make completion of a comprehensive software product impossible, but a limited version must be introduced to meet competitive or business pressure. Some of the core product or system requirements are well understood but the details of product or system extensions have yet to be defined.

- Solution is to adapt Evolutionary Model which is Iterative

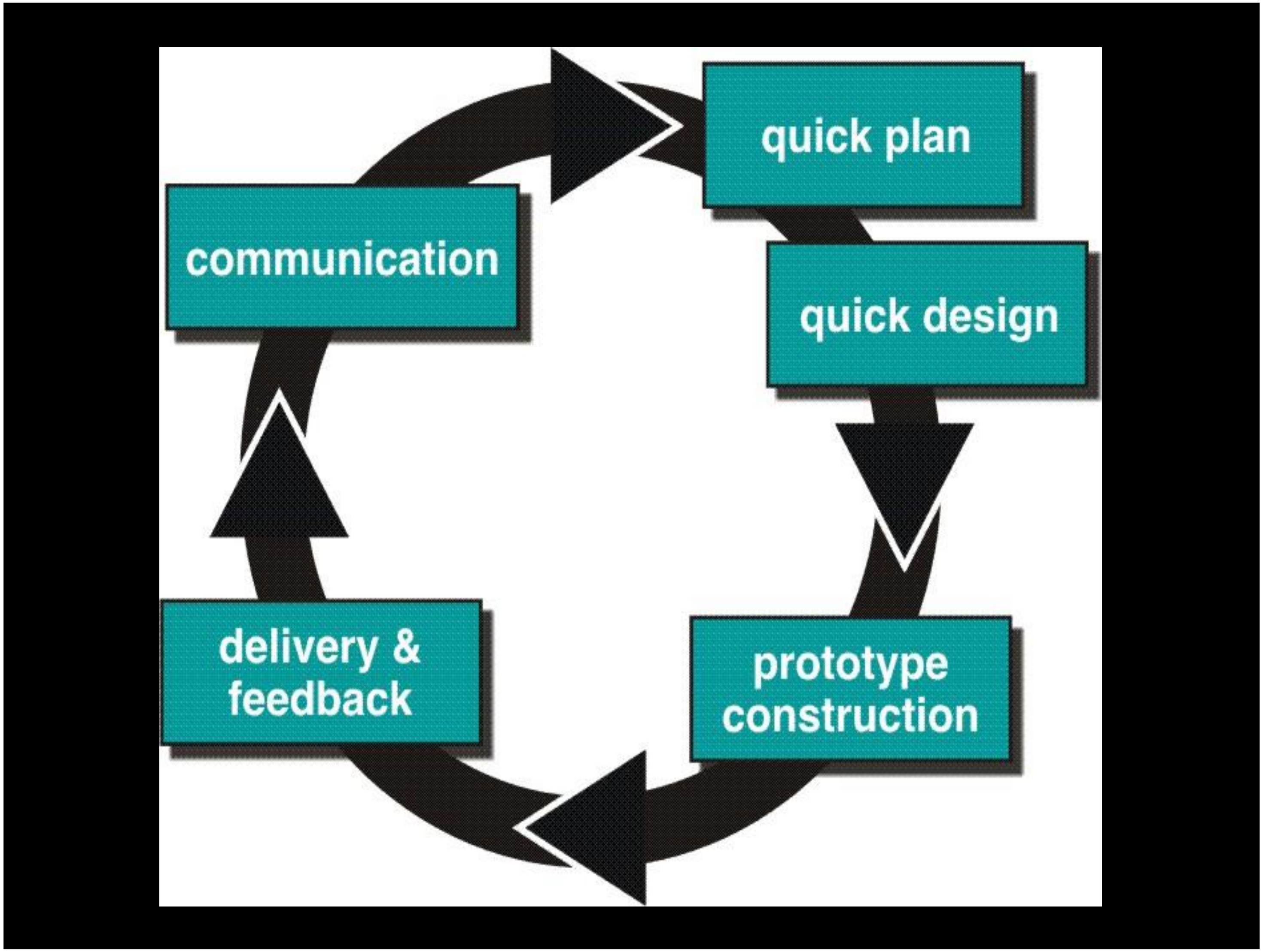
The Prototyping Model: (Need)

- Prototyping is used when customers requirements are fuzzy.

- OR the developer may not be sure of the efficiency of algorithm, the adaptability of an Operating System or the form that Human Computer interaction should take

- But we have to throw away the prototype once the customer requirements are clear & met for better quality. The product must be rebuilt using software engineering practices for long term quality.

Evolutionary Models: Prototyping



The Prototyping Model:

Merits:

- Prototyping helps in requirement gathering & can be applied at any stage of the project.

Demerits:

- Customer insists to convert prototype in working version by applying “few fixes”
- Developer may become comfortable with the compromises done. “The-less-than-ideal-choice” may become integral part of the system

Evolutionary Models: Spiral

Spiral Model is an evolutionary software process model that couples the iterative nature of Prototyping with controlled & systematic aspects of the Waterfall Model

Merits:

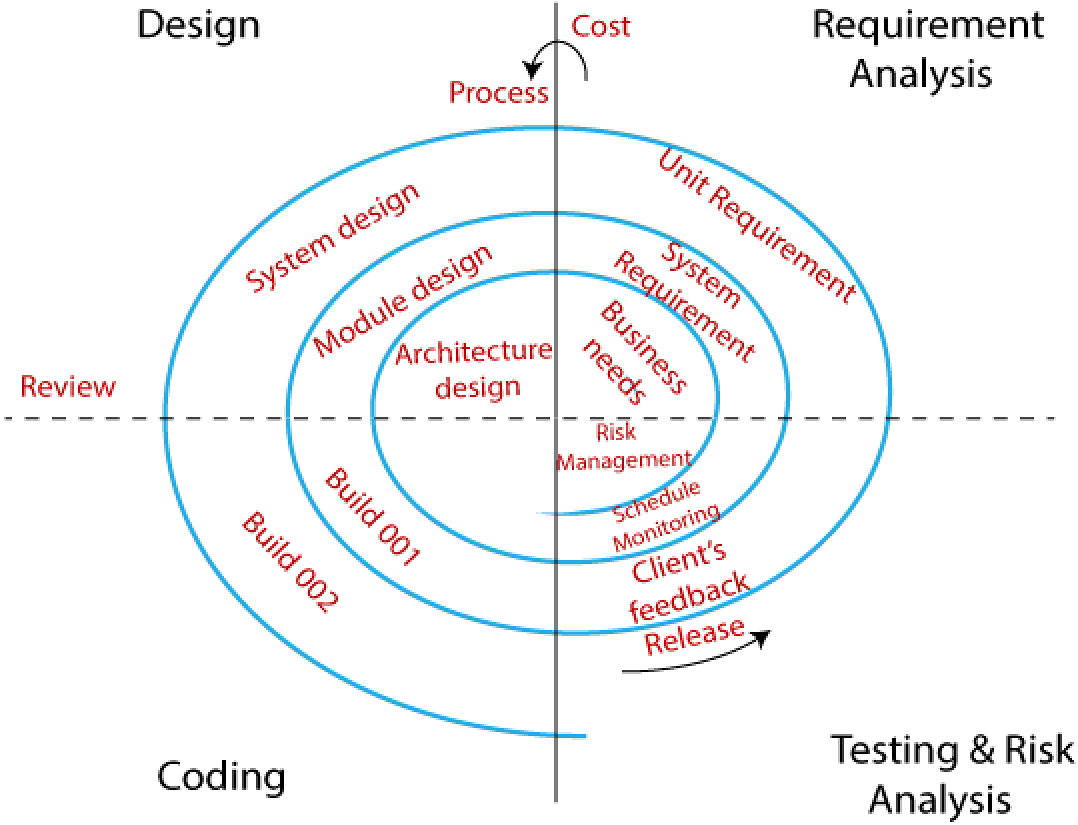
- **Risk is considered as each iteration is made**
- **Spiral Model can be applied throughout the life of the computer software.**

Demerits:

- **It is difficult to convince customers that the evolutionary approach is controllable**
- **Considerable risk assessment expertise required**
- **If major risk is uncovered, problems will occur**

The different phases of the spiral model are as follows:

- **Requirement analysis**
- **Design**
- **Coding**
- **Testing and risk analysis**



Concurrent Development Model:

The concurrent Development Model, sometimes called concurrent engineering can be represented schematically as a series of framework activities, software engineering actions and tasks, & their associated states. All activities exist concurrently.

Modeling activity (Example) :

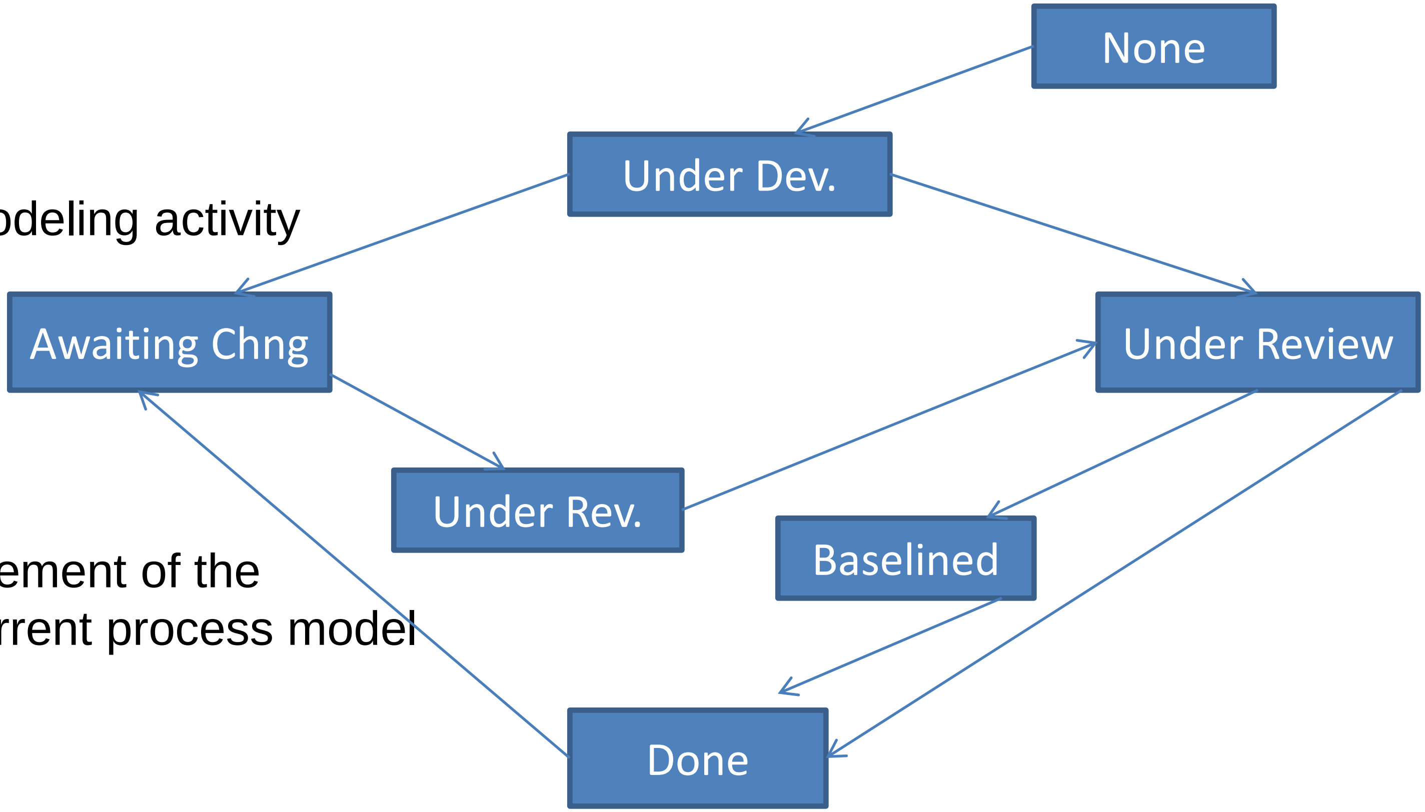


Fig: Modeling activity

One element of the
concurrent process model

Concurrent Development Model: Contd…

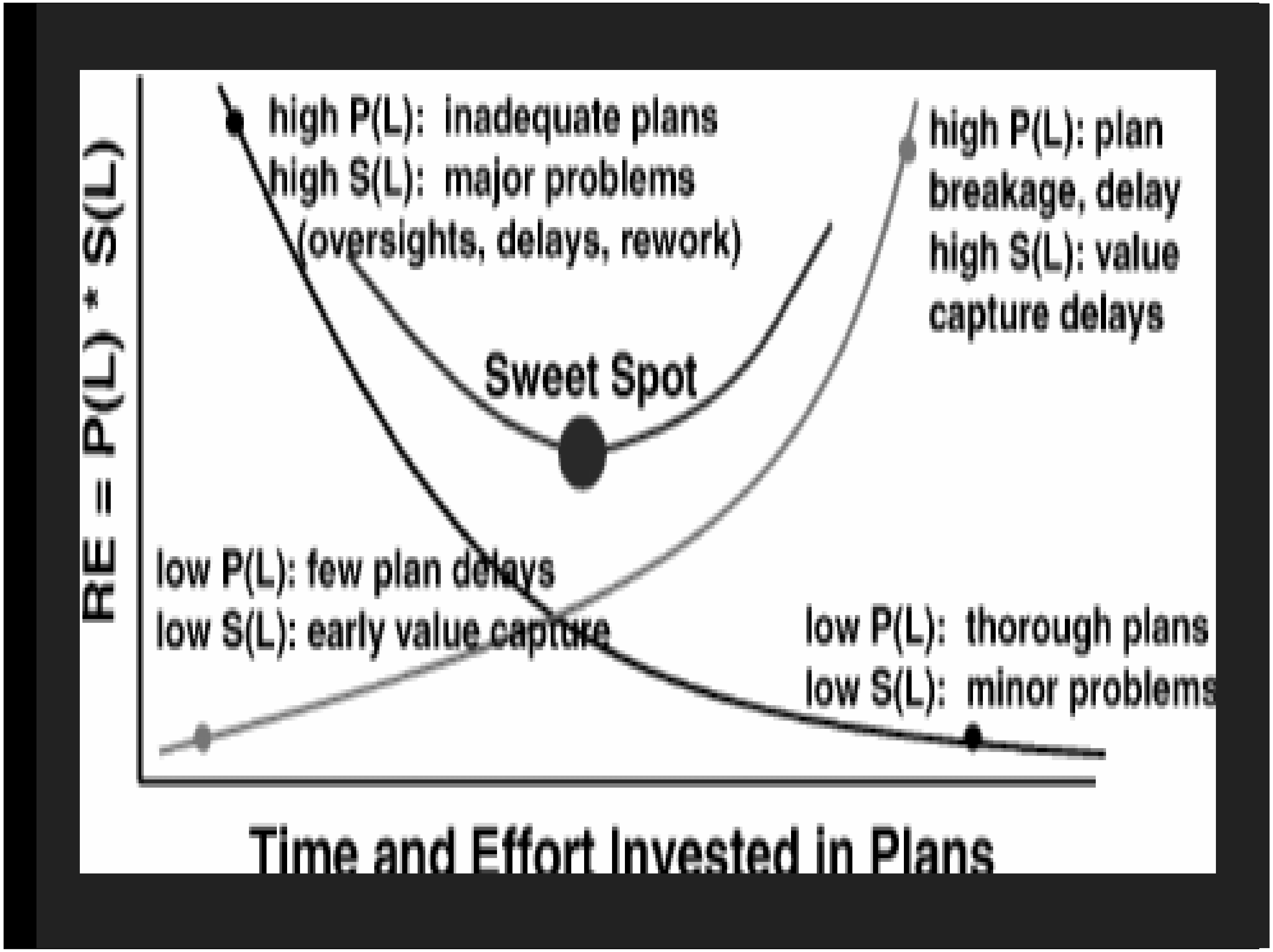
Merits:

- **Applicable to all types of S/W development & provides an accurate picture of the current state of the project.**

Demerits:

- **Problem to Project planning. How many No of iterations are to be planned? Uncertainty…**
- **Process may fall in chaos if the evolutions occurs too fast without a period of relaxation. On the other hand if the speed is too slow productivity could be affected.**
- **S/W processes are focussed on flexibility & extendability, rather than on high quality.**

Risk Exposure



Specialized Process Models:

Specialized process models use many of the characteristics of one or more of the conventional models presented so far, however they tend to be applied when a narrowly defined software engineering approach is chosen. They include,

- **Components based development**
- **The Formal Methods Model**
- **Aspect oriented software development**

Components Based Development :

In this approach, Commercial Off-The-Shelf (COTS) S/W components, developed by vendors who offer them as products are used in the development of software. Characteristics resemble to spiral model.

Following steps are involved in CBD:

1. Available component based products are researched and evaluated for the application domain in question
2. Component issues are considered.
3. A software architecture is designed to accommodate the components.
4. Components are integrated into the architecture.
5. Comprehensive testing is conducted to ensure proper functionality.

Components Based Development

Merits:

➢Leads to software reuse, which provides number of benefits, based on studies of reusability, QSM Associates reports that,

- 70% reduction in development cycle time
- 84 % reduction in project cost
- Productivity index goes up to 26.2 (Norm : 16.9)

Demerits:

➢Component Library must be robust.

➢Performance may degrade

The Formal Methods Model:

- The formal methods model encompasses a set of activities that lead to formal mathematical specification of computer software.
- It consists of specifications, development & verification by applying rigorous mathematical notation. Example, Clean Room S/W Engineering (CRSE)

Merits:

- ➢Removes many of the problems that are difficult to remove using other S/W Engg. Paradigms.
- ➢Ambiguity, Incompleteness & Inconsistency can be discovered & corrected easily by using formal methods of mathematical analysis.

Demerits:

- ➢Development is time consuming & expensive
- ➢ Extensive training is required
- ➢Difficult to use with technically unsophisticated customers

Aspect Oriented Software Development (AOSD):

- A set of localized features, functions & information contents are used while building complex software.
- These localized s/w characteristics are modeled as components (e.g. OO classes) & then constructed within the context of a system architecture.
- Certain “concerns” (Customer required properties or areas of technical interest) span the entire architecture i.e. Cross cutting Concerns like system security, fault tolerance etc.

Merits:

- ➢It is similar to component based development for aspects

Demerits:

- ➢Component Library must be robust.
- ➢Performance may degrade

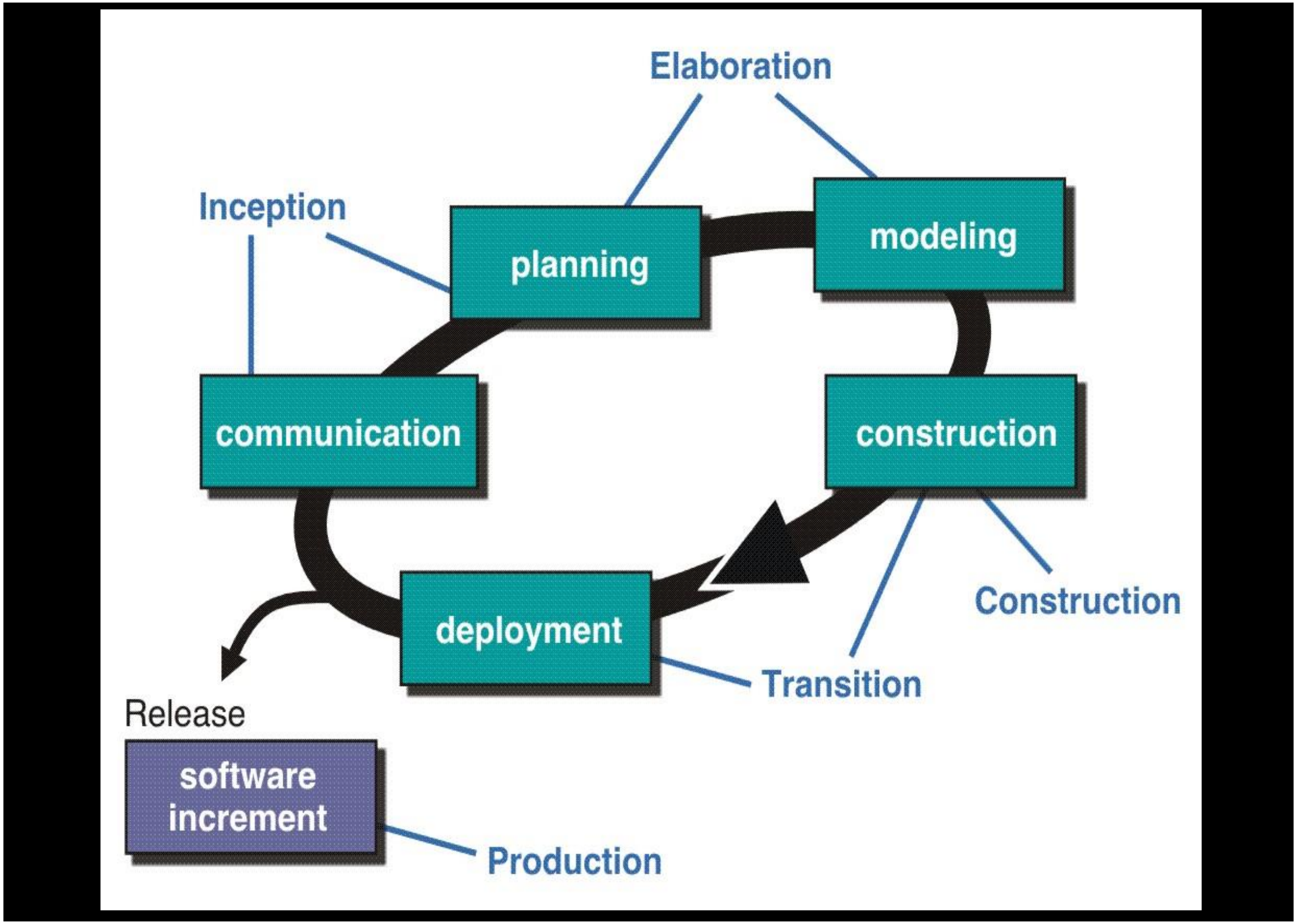
Unified Process Model

A software process that is:

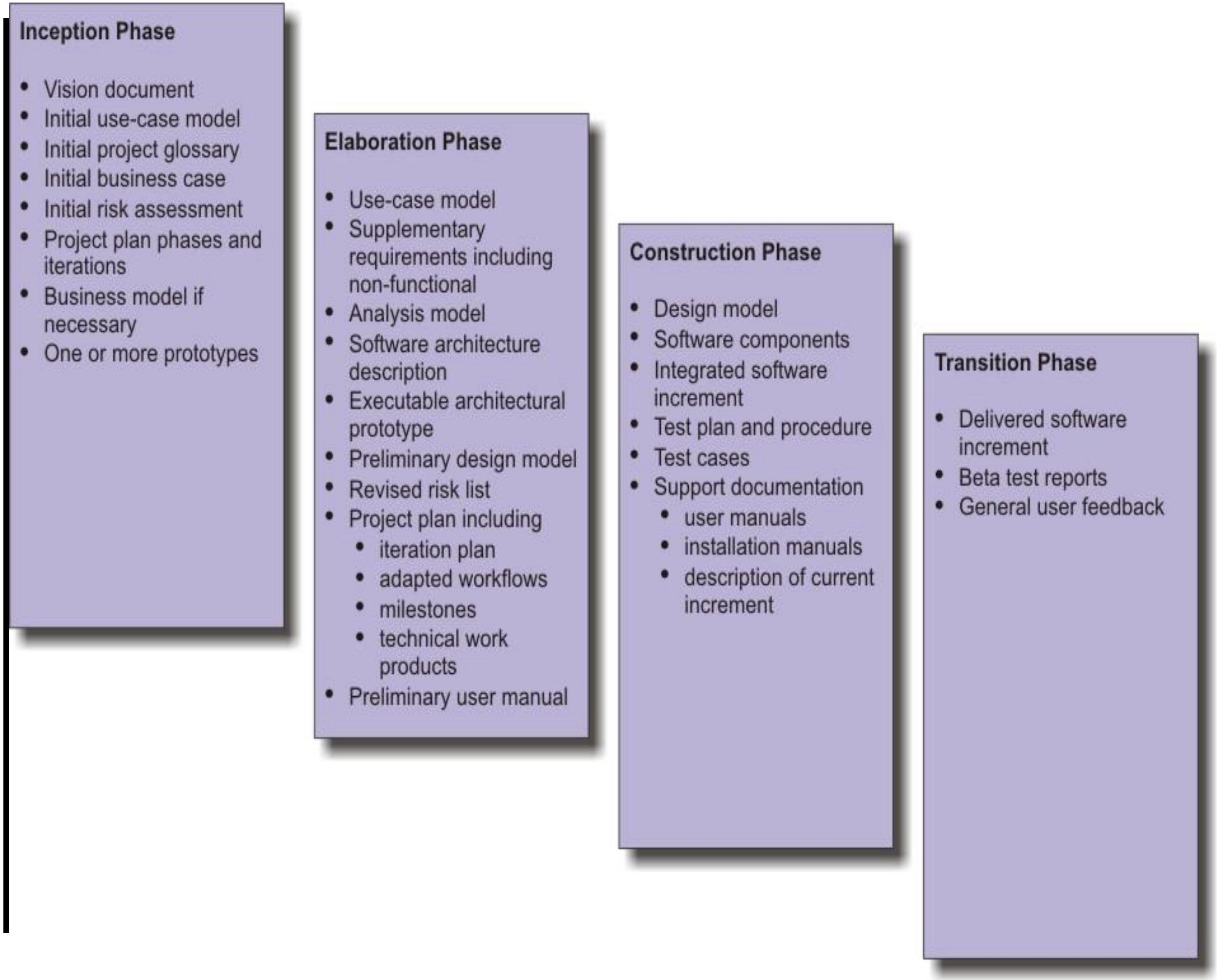
- use-case driven
- architecture-centric
- iterative and incremental

Closely aligned with the
Unified Modeling Language (UML)

The Unified Process (UP)



UP Work Products



Inception Phase

- Vision document
- Initial use-case model
- Initial project glossary
- Initial business case
- Initial risk assessment
- Project plan phases and iterations
- Business model if necessary
- One or more prototypes

Elaboration Phase

- Use-case model
- Supplementary requirements including non-functional
- Analysis model
- Software architecture description
- Executable architectural prototype
- Preliminary design model
- Revised risk list
- Project plan including
  - iteration plan
  - adapted workflows
  - milestones
  - technical work products
- Preliminary user manual

Construction Phase

- Design model
- Software components
- Integrated software increment
- Test plan and procedure
- Test cases
- Support documentation
  - user manuals
  - installation manuals
  - description of current increment

Transition Phase

- Delivered software increment
- Beta test reports
- General user feedback

Common Fears for Developers

- The project will produce the wrong product.
- The project will produce a product of inferior quality.
- The project will be late.
- We’ll have to work 80 hour weeks.
- We’ll have to break commitments.
- We won’t be having fun.

The Manifesto for Agile Software Development

“We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- *Individuals and interactions* over processes and tools
- *Working software* over comprehensive documentation
- *Customer collaboration* over contract negotiation
- *Responding to change* over following a plan

That is, while there is value in the items on the right, we value the items on the left more.”

-- Kent Beck et al.

What is “Agility”?

- Effective (rapid and adaptive) response to change
- Effective communication among all stakeholders
- Drawing the customer onto the team
- Organizing a team so that it is in control of the work performed

Yielding …

- Rapid, incremental delivery of software

An Agile Process

- Is driven by customer descriptions of what is required (scenarios)
- Recognizes that plans are short-lived
- Develops software iteratively with a heavy emphasis on construction activities
- Delivers multiple ‘software increments’
- Adapts as changes occur

Principles of Agility

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time scale.
4. Business people and developers must work together daily throughout the project.

Principles of Agility

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Principles of Agility

9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity - the art of maximizing the amount of work not done - is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

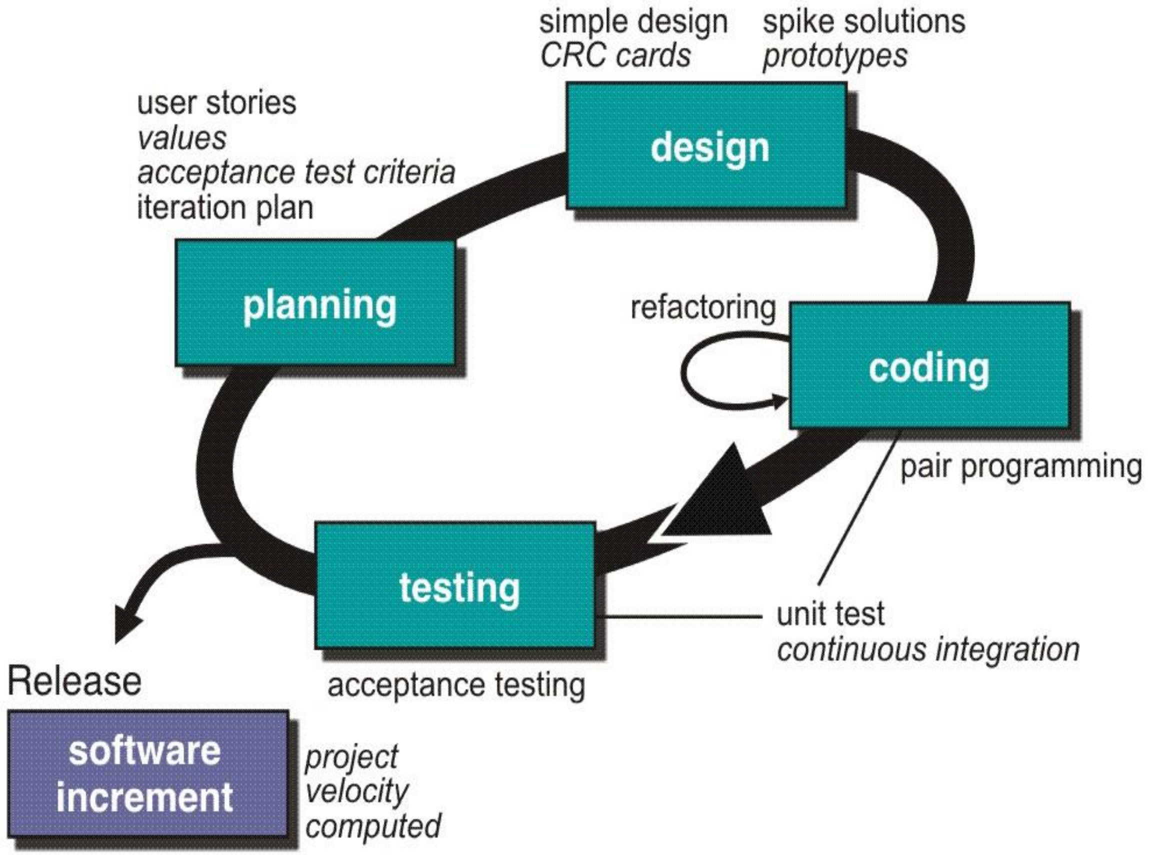
Extreme Programming (XP)

- The most widely used agile process, originally proposed by Kent Beck
- XP Planning
  - Begins with the creation of user stories
  - Agile team assesses each story and assigns a cost
  - Stories are grouped to for a deliverable increment
  - A commitment is made on delivery date
  - After the first increment project velocity is used to help define subsequent delivery dates for other increments

Extreme Programming (XP)

- XP Design
  - Follows the KIS principle
  - Encourage the use of CRC cards
  - For difficult design problems, suggests the creation of spike solutions — a design prototype
  - Encourages refactoring — an iterative refinement of the internal program design
- XP Coding
  - Recommends the construction of a unit test for a store *before* coding commences
  - Encourages pair programming
- XP Testing
  - All unit tests are executed daily
  - Acceptance tests are defined by the customer and executed to assess customer visible functionality

Extreme Programming (XP)



Other Agile Processes

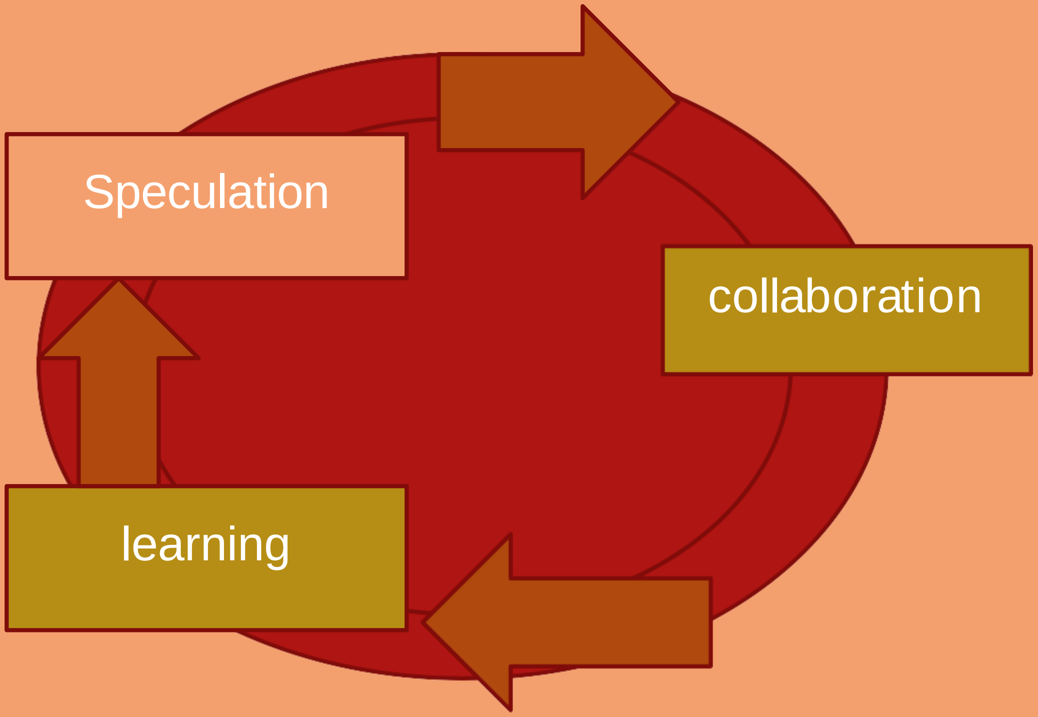
- Adaptive Software Development (ASD)
- Dynamic Systems Development Method (DSDM)
- Scrum
- Crystal
- Feature Driven Development
- Agile Modeling (AM)

Introduction to Adaptive software development

- A software development process that grew out of rapid application development work.
- It has been proposed by Jim Highsmith as a technique for building complex software and system.
- It focus on human collaboration and team self-organization.

Adaptive software development

- Life cycle



Speculation

- During speculation, the project is initiated and adaptive cycle planning is conducted.

- When we speculate, it's not that we don't define a mission to the best of our ability.

Collaboration

- Encompasses communication and teamwork but also emphasizes individualism.
- If we can't predict , then we can't control in the traditional management sense.
- If we can't control, then a significant set of current management practices is no longer operable.

Learning

- Challenges all stakeholders.
- Examine their assumptions and use the results of each development cycle to learn the direction of the next.
- Customer focus groups, technical reviews, beta testing, and postmortems are all practices that expose results to scrutiny.

Speculate-collaborate-learn

- For many project leaders and project teams, adaptive development is a terrifying prospect.
- First, we knock away the foundation pillar of cause and effect, so we can't say for sure what needs to be done next.

Advantages and disadvantages

Advantages

- Software incremental adjustment
- Rapid and complex software development

Disadvantages

- There is lack of emphasis on necessary designing and documentation.
- it is difficult to assess the effort required at the beginning of the software development life cycle.

Scrum

SCRUM is an agile development process focused primarily on ways to manage tasks in team-based development conditions.

There are three roles in it, and their responsibilities are:

Scrum Master: The scrum can set up the master team, arrange the meeting and remove obstacles for the process

Product owner: The product owner makes the product backlog, prioritizes the delay and is responsible for the distribution of functionality on each repetition.

Scrum Team: The team manages its work and organizes the work to complete the sprint or cycle.

Crystal:

There are three concepts of this method-

1. Chartering: Multi activities are involved in this phase such as making a development team, performing feasibility analysis, developing plans, etc.

2.Cyclic delivery: under this, two more cycles consist, these are:

Team updates the release plan.

Integrated product delivers to the users.

3. Wrap up: According to the user environment, this phase performs deployment, post-deployment.

Dynamic Software Development Method(DSDM):

DSDM is a rapid application development strategy for software development and gives an agile project distribution structure. The essential features of DSDM are that users must be actively connected, and teams have been given the right to make decisions. The techniques used in DSDM are:
Time Boxing
MoSCoW Rules
Prototyping

The DSDM project contains seven stages:
Pre-project
Feasibility Study
Business Study
Functional Model Iteration
Design and build Iteration
Implementation
Post-project

Feature Driven Development(FDD):

This method focuses on "Designing and Building" features. In contrast to other smart methods, FDD describes the small steps of the work that should be obtained separately per function.

Lean Software Development:

Lean software development methodology follows the principle "just in time production." The lean method indicates the increasing speed of software development and reducing costs. Lean development can be summarized in seven phases.

1. Eliminating Waste
2. Amplifying learning
3. Defer commitment (deciding as late as possible)
4. Early delivery
5. Empowering the team
6. Building Integrity
7. Optimize the whole

When to use the Agile Model?

- When frequent changes are required.
- When a highly qualified and experienced team is available.
- When a customer is ready to have a meeting with a software team all the time.
- When project size is small.

Advantage(Pros) of Agile Method:

- Frequent Delivery
- Face-to-Face Communication with clients.
- Efficient design and fulfils the business requirement.
- Anytime changes are acceptable.
- It reduces total development time.

Disadvantages(Cons) of Agile Model:

- Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.
- Due to the lack of proper documentation, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.