# Process Models

Lecturer: Adel Vahdati

# Generic Process Model - Framework Activities

- Communication
- Planning
- Modeling
- Construction
- Deployment

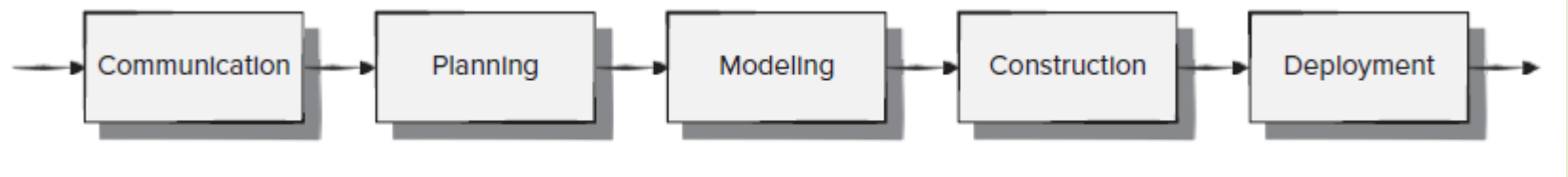
# Generic Process Model - Umbrella Activities

- Project tracking and control
- Risk management
- Quality assurance
- Technical reviews
- Measurement
- Configuration management
- Reusability management
- Work product preparation and production

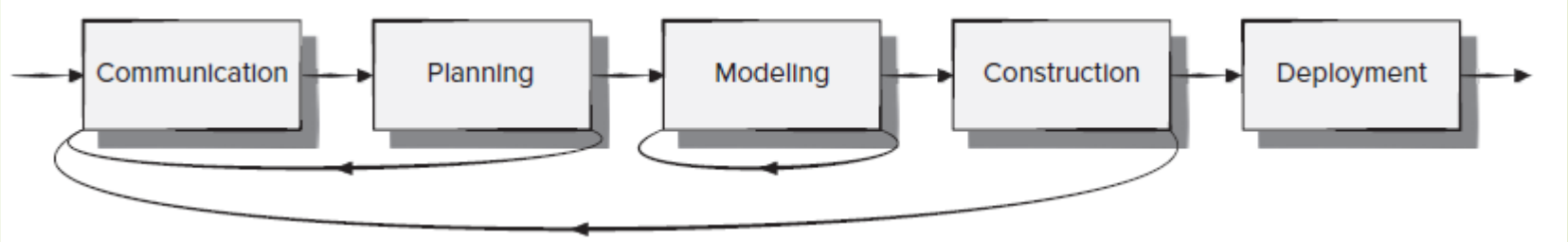
# Generic Process Model - Process Flows

- **Linear**
  - Sequential execution of activities from communication to deployment.
- **Iterative**
  - Repeats one or more activities before moving to the next.
- **Evolutionary**
  - Circular execution, leading to progressively complete software versions.
- **Parallel**
  - Executes activities in parallel (e.g., modeling and construction).

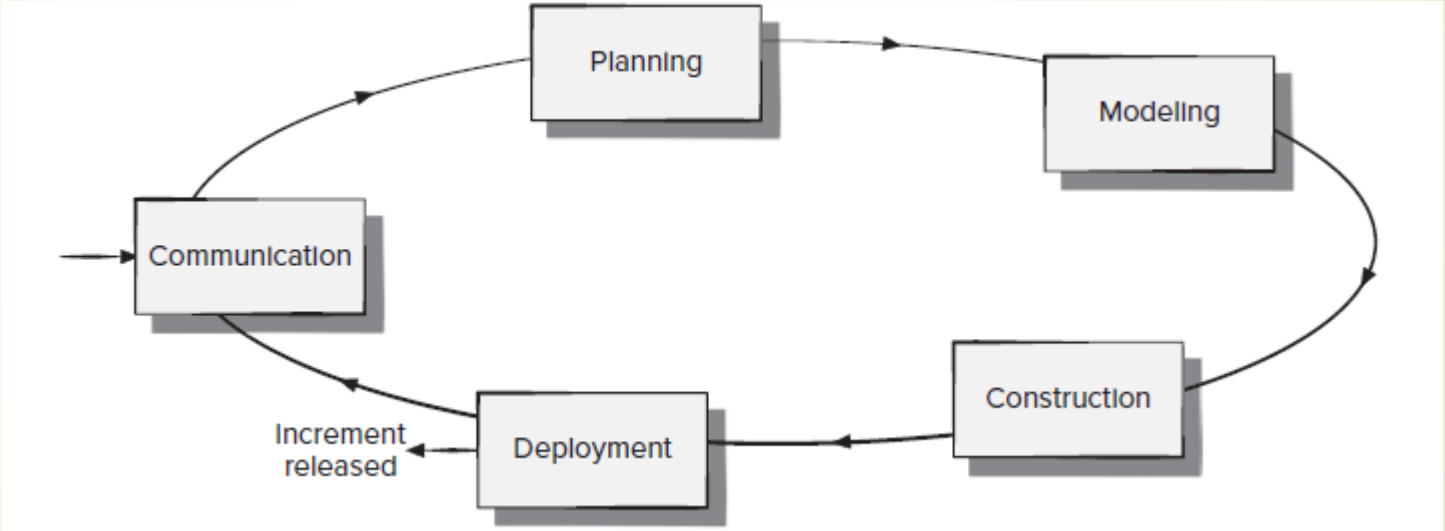
# Linear Process Flow



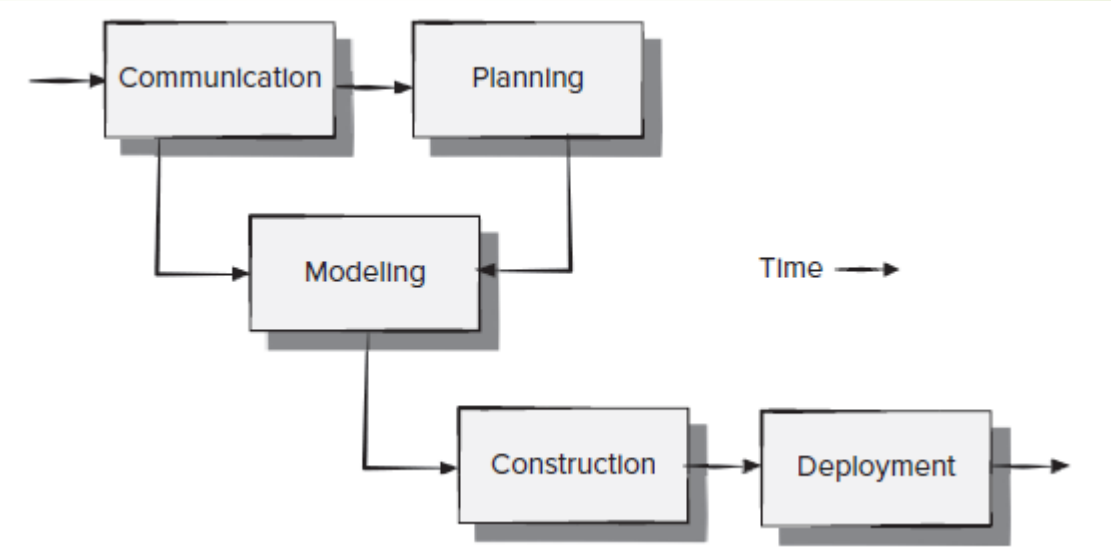
# Iterative Process Flow



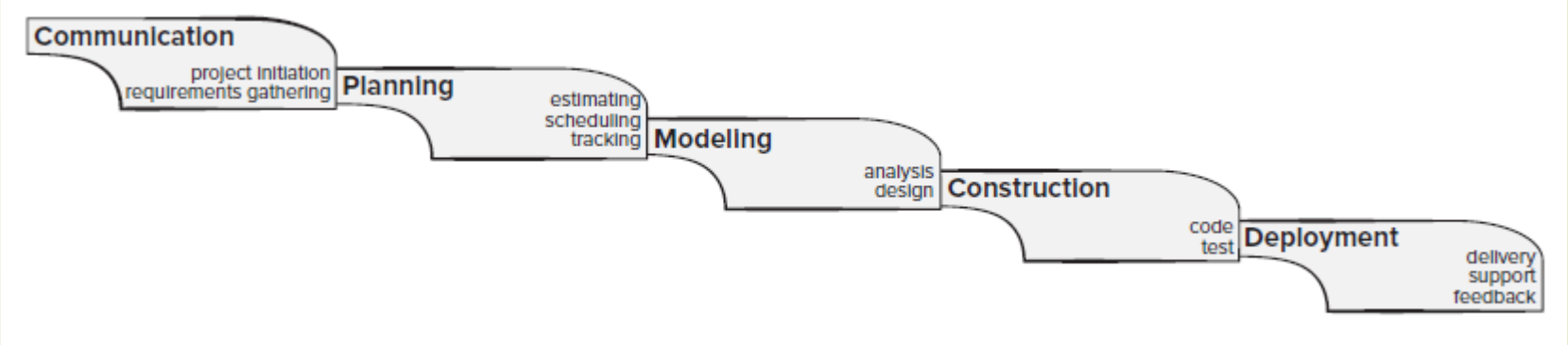
# Evolutionary Process Flow



# Parallel Process Flow



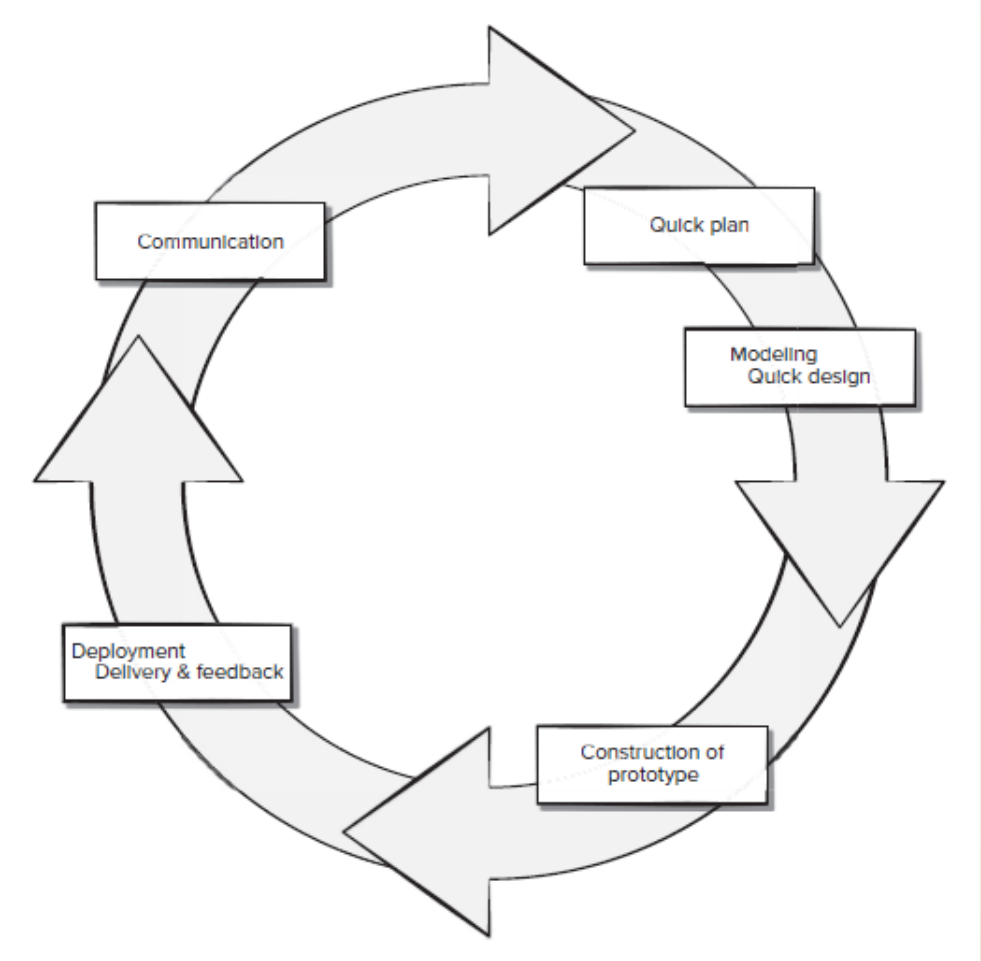
# Waterfall Model



# Waterfall Model

- **Characteristics**
  - Linear, sequential approach
  - Begins with customer requirements, progresses through planning, modeling, construction, deployment, and ongoing support.
- **Challenges**
  - Rarely follows sequential workflow in real projects.
  - Difficult to state all requirements at the beginning.
  - Customer must wait for a working version until late in the project.
  - Major issues may not be detected until the end.

# Prototyping Process Model



# Prototyping Process Model

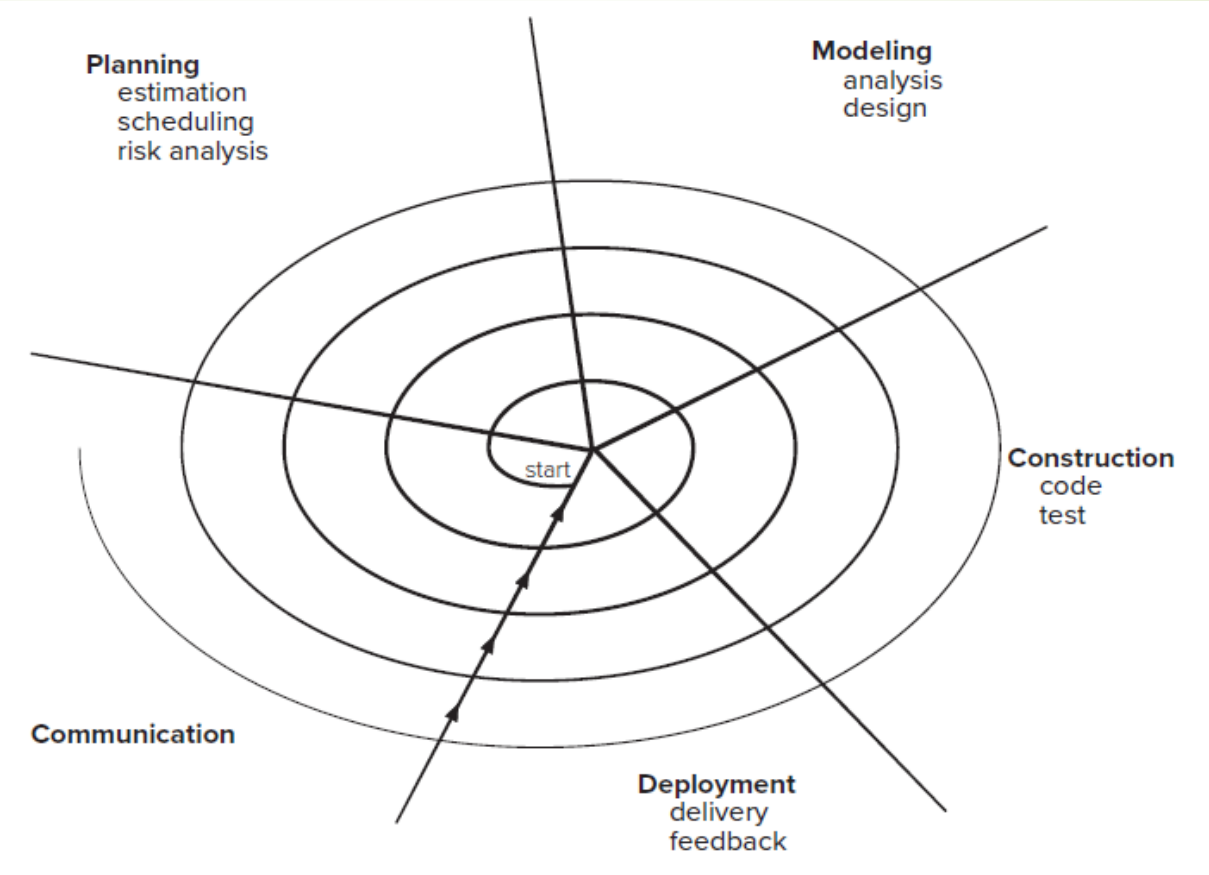
- **Characteristics**
  - Suitable when requirements are unclear or evolving.
  - Begins with communication to define objectives and known requirements.
  - Quick design and construction of a prototype.
  - Prototype evaluated by stakeholders, feedback used to refine requirements.
  - Iterative tuning of the prototype to meet stakeholder needs.

- **Challenges**
  - Stakeholders may mistake the prototype for the final product, unaware of evolving architecture and potential quality issues.
  - Developers might make quick, less-than-ideal implementation choices that become part of the final system

# Evolutionary Process Model

- **Nature of Software**
  - Software evolves over time.
  - Business and product requirements change during development.
  - Tight deadlines may necessitate limited initial versions.
- **Spiral Model**
  - Proposed by Barry Boehm.
  - Combines iterative prototyping with systematic waterfall aspects.
  - Enables rapid development of increasingly complete software versions.

# Evolutionary Process Model - Spiral



# Spiral Model

- **Development Process**
  - Series of evolutionary releases.
  - Early iterations: models or prototypes.
  - Later iterations: progressively complete software versions.
  - Each spiral circuit adjusts project plans based on customer feedback.
  - Cost and schedule are updated iteratively.
- **Lifecycle Adaptation**
  - Can be applied throughout the software's life.
  - Suitable for large-scale systems.
  - Uses prototyping to reduce risks.
  - Directly addresses technical risks at all stages.
- **Challenges**
  - Convincing customers of the controllability of the evolutionary approach.
  - Time to market is critical; missing market windows can render projects meaningless.

# Unified Process Model

- Combines best features of traditional models with agile principles.
- Emphasizes customer communication and use cases.
- Focuses on software architecture for understandability, adaptability, and reuse.
- Iterative and incremental process flow.
- Uses UML for modeling and development of object-oriented systems.

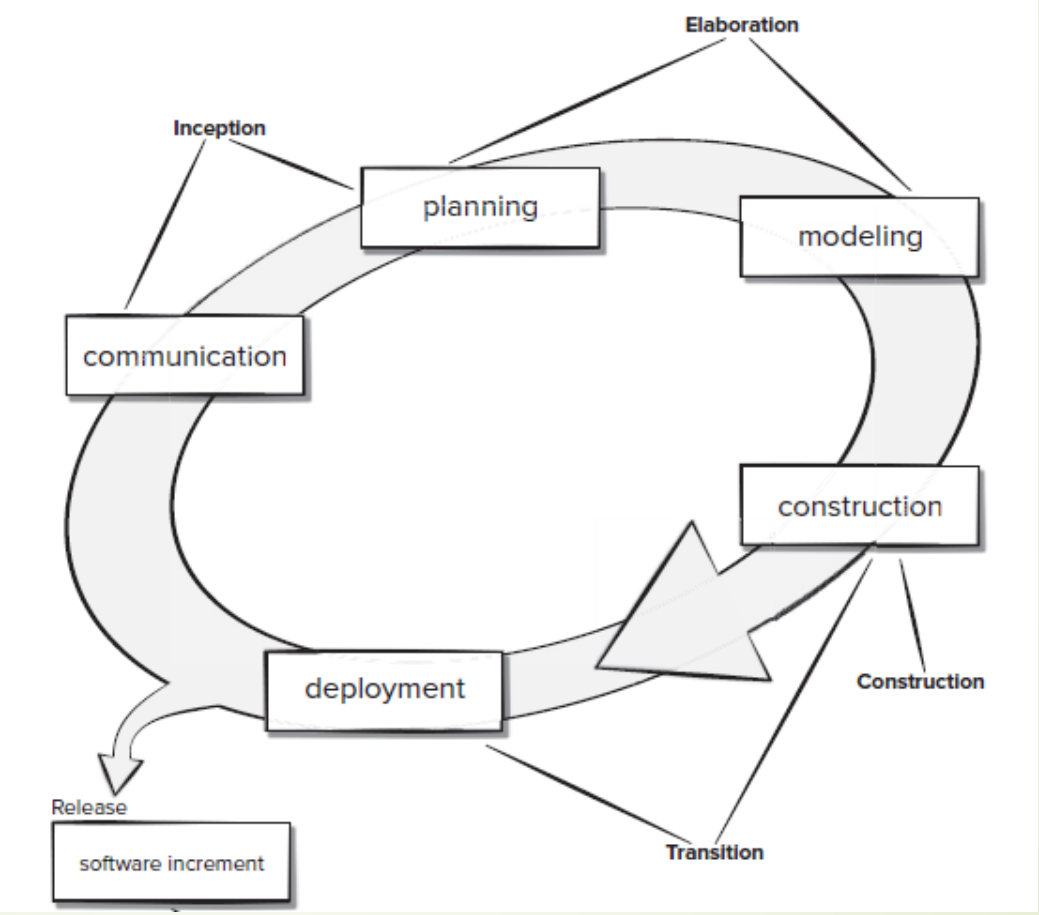
# Unified Process Model - Phases

- **Inception:**
  - Customer communication and planning.
  - Preliminary use cases and business requirements.
  - Resource planning, risk assessment, and preliminary scheduling.
- **Elaboration:**
  - Refines and expands use cases.
  - Creates an architectural baseline with five views
    - use case, analysis, design, implementation, and deployment models.
  - Modifies the plan as needed.
- **Construction:**
  - Implements features and functions in source code.
  - Conducts unit and integration testing.
  - Uses acceptance tests derived from use cases.

# Unified Process Model – Phases (2)

- **Transition:**
  - Beta testing and user feedback.
  - Addresses defects and necessary changes.
  - Results in a usable software release.
  - Monitors ongoing software use.
  - Provides support for the operating environment.
  - Evaluates defect reports and change requests.

# Unified Process Model



# Unified Process Model - Adaptability

- Not all tasks are conducted for every project.
- The process is adapted to meet the team's needs.

# References


- R. S. Pressman and B. R. Maxim. Software Engineering: A Practitioner's Approach. 9th Edition, McGraw-Hill, 2019.