**Popular SDLC Models**

1. **V-Model:**

* **Overview:**
  + **Description:** Documentation-intensive model.
  + **Characteristics:** Sequential execution, resembling a V-shape.
  + **Also Known As:** Verification and Validation model, an extension of the waterfall model.
* **Phases:**
  + **Business Requirement Analysis:**
    - Understand product requirements from the customer's perspective.
    - Critical for managing customer expectations.
    - Input for acceptance testing.
  + **System Design:**
    - Design the complete system based on clear and detailed requirements.
    - Develop the system test plan.
    - Early planning allows more time for test execution.
  + **Architectural Design:**
    - Understand and design architectural specifications.
    - Propose multiple technical approaches and make decisions based on feasibility.
    - Break down system design into modules (High-Level Design - HLD).
    - Define data transfer and communication between modules for integration tests.
  + **Module Design:**
    - Specify detailed internal design for all system modules (Low-Level Design - LLD).
    - Ensure compatibility with the system architecture and external systems.
    - Design unit tests to identify faults and errors at an early stage.
* **Execution:**
  + Phases executed sequentially in a disciplined manner.
  + Testing phases (Verification and Validation) run in parallel with corresponding development phases.
  + Coding phase joins the two sides of the V-Model.
* **Illustration:**
* **Key Characteristics:**
  + Each development phase has an associated testing phase.
  + Next phase starts only after the completion of the previous phase.
  + Highly disciplined approach.

2. **(Additional SDLC Models - If Provided):**

* *(If additional SDLC models are provided, you can follow a similar format to describe each model, including its overview, key phases, execution characteristics, and any specific illustrations.)*

**Software Development Life Cycle (SDLC) Models**

1. **Spiral Model:**

* **Overview:**
  + Risk-driven process model combining features of prototyping and waterfall models.
  + Adapts elements of various process models based on project needs.
* **Phases:**
  + **Identification:**
    - Gather business requirements in the baseline spiral.
    - Identify system, subsystem, and unit requirements in subsequent spirals.
    - Continuous communication between customer and system analyst.
  + **Design:**
    - Conceptual design in baseline spiral.
    - Involves architectural, logical, and physical design in subsequent spirals.
  + **Construct or Build:**
    - Production of the actual software product in every spiral.
    - Develop Proof of Concept (POC) in baseline spiral for customer feedback.
    - Subsequent spirals produce working models (builds) with version numbers.
  + **Evaluation and Risk Analysis:**
    - Risk analysis includes identifying, estimating, and monitoring technical feasibility and management risks.
    - Customer evaluates the software and provides feedback at the end of each iteration.
* **Illustration:**
* **Key Characteristics:**
  + Iterative process with repeated phases (spirals).
  + Emphasizes risk analysis and customer feedback.
  + Suitable for large and complex projects with evolving requirements.

2. **Big Bang Model:**

* **Overview:**
  + Focuses on software development with minimal or no planning.
  + Suitable for small projects, academic software development, and scenarios with unknown or changing requirements.
* **Pros and Cons:**
  + **Advantages:**
    - Simple and requires little planning.
    - Easy to manage with no formal procedures.
    - Suitable for small projects and good for learning.
  + **Disadvantages:**
    - High risk and uncertainty.
    - Not suitable for complex and ongoing projects.
    - May become expensive if requirements are misunderstood.

3. **Waterfall Model:**

* **Overview:**
  + Widely accepted SDLC model with a rigid, sequential approach.
  + Divides the software development process into phases, where each phase's outcome is input for the next.
* **Phases:**
  + **Planning and Design:**
    - Long planning and design phases.
    - Outcome serves as input for subsequent phases.
  + **Testing:**
    - Software goes through testing phases after development.
    - Final deployment for use.
* **Advantages:**
  + Simple and easy to understand and use.
  + Easy to manage with a predefined path.
  + Works well for smaller projects with well-understood requirements.
  + Clearly defined stages and milestones.
* **Disadvantages:**
  + Rigid and not adaptable to changing requirements.
  + Lack of feedback throughout the process.
  + Implementation of requirements may not reflect changes that occurred during development.