**Agile Model**

**Overview:**

* **Definition:**
  + Agile methodology promotes continuous interaction between development and testing throughout the SDLC process.
  + Divides the entire project into small incremental builds, delivered in iterations lasting one to three weeks.
  + Emphasizes adaptability, teamwork, prototyping, and feedback loops.
* **Manifesto for Agile Software Development:**
  + Drafted and signed by software developers in 2001.
  + Highlights contrasts with the Waterfall model, emphasizing flexibility and adaptability.

**Agile Manifesto Principles:**

1. **Individuals and interactions:**
   * Emphasizes self-organization, motivation, and interactions like co-location and pair programming.
2. **Working software:**
   * Demo working software as the best means of communication with customers.
   * Prioritizes actual software over documentation.
3. **Customer collaboration:**
   * Continuous customer interaction due to evolving requirements.
   * Ensures proper understanding of product requirements.
4. **Responding to change:**
   * Focused on quick responses to change and continuous development.

**Popular Agile Methods:**

* **Examples:**
  + Rational Unified Process, Scrum, Crystal Clear, Extreme Programming (XP), Adaptive Software Development, Feature Driven Development, Dynamic Systems Development Method (DSDM).
  + These collectively form Agile Methodologies post the Agile Manifesto publication in 2001.

**Pros and Cons:**

* **Advantages:**
  + Realistic approach to software development.
  + Promotes teamwork and cross-training.
  + Rapid development and demonstration of functionality.
  + Minimum resource requirements.
  + Suitable for changing requirements.
  + Delivers early partial working solutions.
* **Disadvantages:**
  + Not suitable for handling complex dependencies.
  + More risk of sustainability, maintainability, and extensibility.
  + Requires an overall plan, agile leadership, and project management practices.
  + Strict delivery management dictates scope and adjustments to meet deadlines.
  + Heavy dependence on customer interaction may lead to wrong directions.

**Summary on Software Development Life Cycle (SDLC)**

The Software Development Life Cycle (SDLC) is a systematic process designed for building software to ensure its quality and correctness. Here are key points about SDLC:

1. **Definition:**
   * SDLC stands for Software Development Lifecycle.
   * It is a systematic process that guides the development, testing, deployment, and maintenance of software.
2. **Framework and Stages:**
   * SDLC provides a framework consisting of a standard set of activities and deliverables.
   * The process involves seven key stages:
     1. Requirement Collection and Analysis
     2. Feasibility Study
     3. Design
     4. Coding
     5. Testing
     6. Installation/Deployment
     7. Maintenance
3. **Responsibilities in Stages:**
   * Senior team members typically lead the requirement analysis phase.
   * The feasibility study stage includes planning for the entire project life cycle.
   * Design phase involves preparing system and software design documents based on requirement specifications.
4. **Development Phases:**
   * Coding phase involves writing code using the chosen programming language to build the entire system.
   * Testing phase verifies that the application works according to customer requirements.
5. **Deployment and Maintenance:**
   * Installation and deployment begin once the testing phase is complete with no remaining bugs or errors.
   * Maintenance phase includes bug fixing, upgrades, and ongoing engagement to ensure the system's effectiveness.
6. **SDLC Models:**
   * Popular SDLC models include Waterfall, Incremental, Agile, V model, Spiral, and Big Bang.
   * Each model follows a specific approach to planning, building, and maintaining software.
7. **Detailed Plan:**
   * SDLC consists of a detailed plan that guides the entire software development process.
   * The plan includes steps for planning, building, and maintaining specific software.

In summary, SDLC is a comprehensive approach that ensures the successful development and maintenance of high-quality software by following a structured and systematic process.