BMT CONSULTING
A PROJECT MANAGEMENT TRAINING AND CURRICULUM DEVELOPMENT COMPANY

4/27/2025

# Choosing Your Agile Path: A Guide to Popular Models

***Understanding Different Agile Approaches for Your Project Needs***

Agile isn't one-size-fits-all! Different models offer unique structures, practices, and focuses. Let's explore some popular ones to see which might be the best fit for your team and project

---

## <u>Model: Scrum</u>

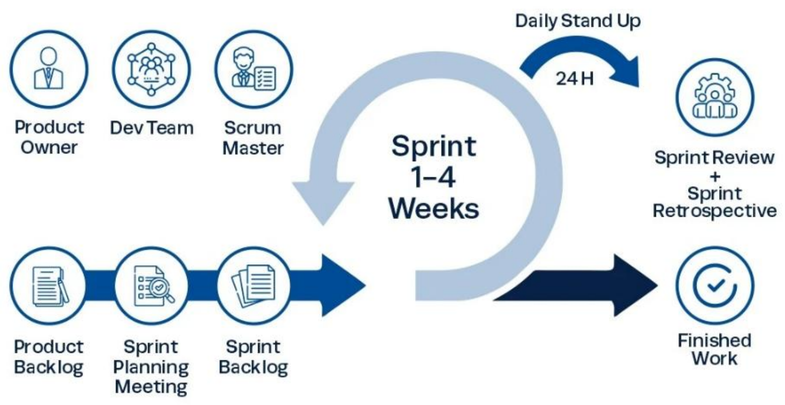

Figure 1 – Scrum Model

- **Core Idea:** Deliver value iteratively in fixed-length **Sprints**, guided by specific roles, events, and artifacts. Focuses on collaboration and adaptation.
- **Key Characteristics:**
  - Fixed Timeboxes (Sprints: 1-4 weeks)
  - Defined Roles (Product Owner, Scrum Master, Developers)
  - Prescribed Events (Planning, Daily Scrum, Review, Retrospective)
  - Artifacts (Product Backlog, Sprint Backlog, Increment)
  - Focus on delivering a usable Increment each Sprint.
- ⭐ **When to Use Scrum:**
  - Complex product development with evolving requirements.
  - Need for a regular, predictable delivery cadence.
  - Cross-functional teams working on a single product.

- Situations where frequent feedback loops are valuable.

---

## Model: Kanban

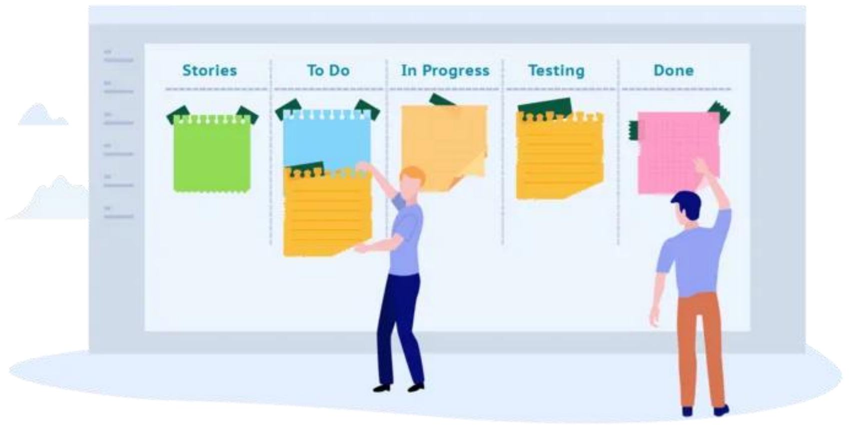

Figure 2 – Kanban Model

- **Core Idea:** Visualize your workflow, **limit Work-In-Progress (WIP)**, and manage flow to improve efficiency and predictability. It focuses on continuous delivery.
- **Key Characteristics:**
  - Visual Board (Kanban Board)
  - WIP Limits (Crucial for preventing bottlenecks)
  - Manage Flow (Measure cycle time, lead time)
  - Explicit Policies (How work moves)
  - Continuous Flow (No prescribed iterations)
  - Focus on optimizing the existing process.
- ⭐ **When to Use Kanban:**
  - Operational or support teams (Help Desk, Maintenance).
  - Work arrives unpredictably or priorities shift frequently.
  - Need to improve workflow visibility and reduce bottlenecks.
  - Situations where fixed Sprints feel unnatural or restrictive.

---

## Model: Extreme Programming (XP)

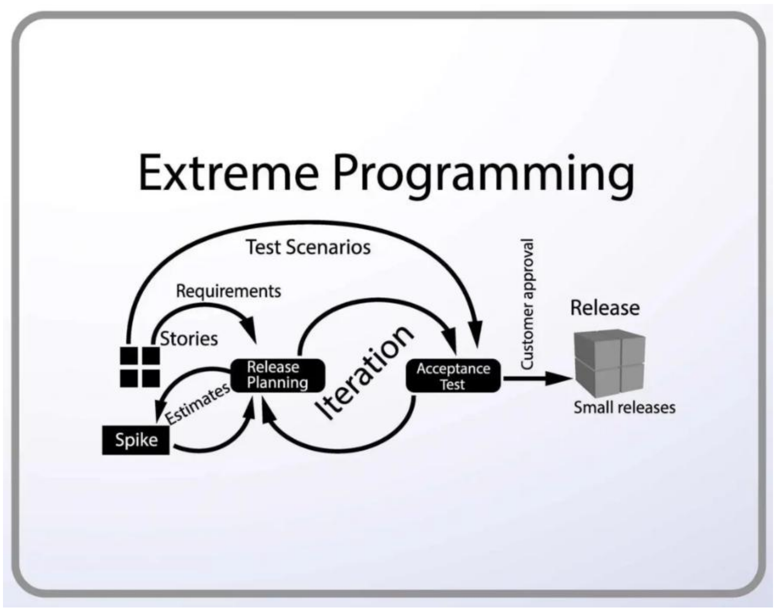

Figure 3 – Extreme Programing Model

- **Core Idea:** Achieve high software quality and responsiveness to changing customer needs through specific **engineering practices**.
- **Key Characteristics:**
    - Values: Communication, Simplicity, Feedback, Courage, Respect
    - Key Practices: Pair Programming, Test-Driven Development (TDD), Continuous Integration (CI), Refactoring, Simple Design, Collective Code Ownership.
    - Short Iterations (Often 1-2 weeks)
    - On-Site Customer involvement.
- ⭐ **When to Use XP:**
    - The highest technical quality is paramount.
    - Requirements are expected to change significantly.
    - Need to manage technical debt effectively.
    - Projects where automated testing and continuous integration are feasible and desired.
    - Often used *alongside* Scrum.

---

## Model: Lean Software Development

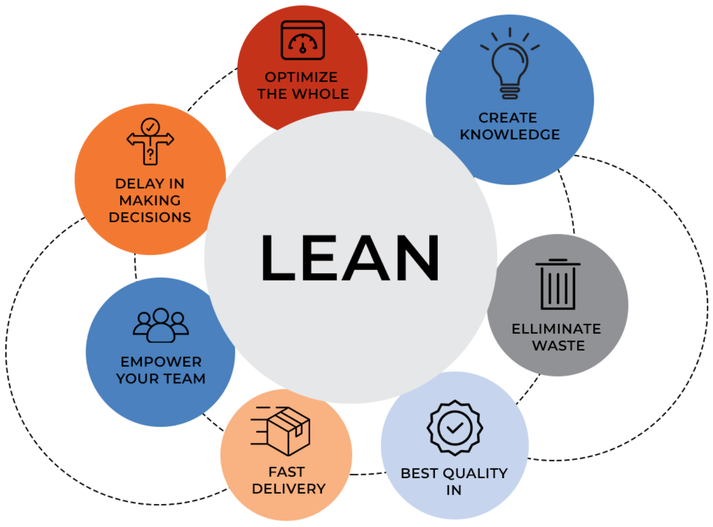

Figure 4 – Lean Software Development Model

- **Core Idea:** Optimize the whole value stream by **eliminating waste**, amplifying learning, delivering fast, and empowering the team. More of a philosophy/set of principles.
- **Key Characteristics:**
  - 7 Principles: Eliminate Waste, Build Quality In, Create Knowledge (Amplify Learning), Defer Commitment, Deliver Fast, Respect People, Optimize the Whole.
  - Focus on Value Stream Mapping.
  - Often underpins other methods (like Kanban).
- ⭐ **When to Use Lean Principles:**
  - Focusing on process efficiency and speed.
  - Identifying and removing bottlenecks across the entire delivery process.
  - Making strategic decisions about what *not* to build (defer commitment).
  - Fostering a culture of continuous improvement (Kaizen).
  - *Can be applied within any other framework.*

## Model: Crystal

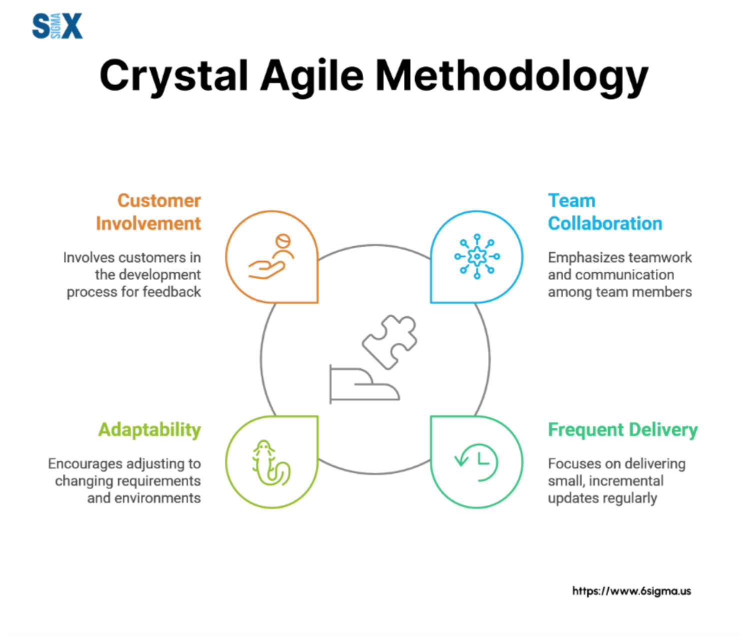
SIX SIGMA
Crystal Agile Methodology
Customer Involvement
Involves customers in the development process for feedback
Team Collaboration
Emphasizes teamwork and communication among team members
Adaptability
Encourages adjusting to changing requirements and environments
Frequent Delivery
Focuses on delivering small, incremental updates regularly
https://www.6sigma.us

Figure 5 – Crystal Agile Method

- **Core Idea:** A family of **adaptive methodologies** focusing on people, interaction, community, skills, and communication. Tailored based on team size and project criticality.
- **Key Characteristics:**
  - Focus on People & Communication (e.g., Osmotic Communication for co-located teams).
  - Frequent Delivery.
  - Reflective Improvement.
  - Configurable based on context (Crystal Clear, Yellow, Orange, Red, etc.).
  - Lighter on prescribed processes than Scrum.
- ⭐ **When to Use Crystal:**
  - Team communication and collaboration are critical success factors.
  - Project needs a methodology tailored to its specific size and risk profile.
  - Emphasis on building team effectiveness and trust.
  - Situations where a less rigid framework is preferred.

## Model: Dynamic Systems Development (DSDM)

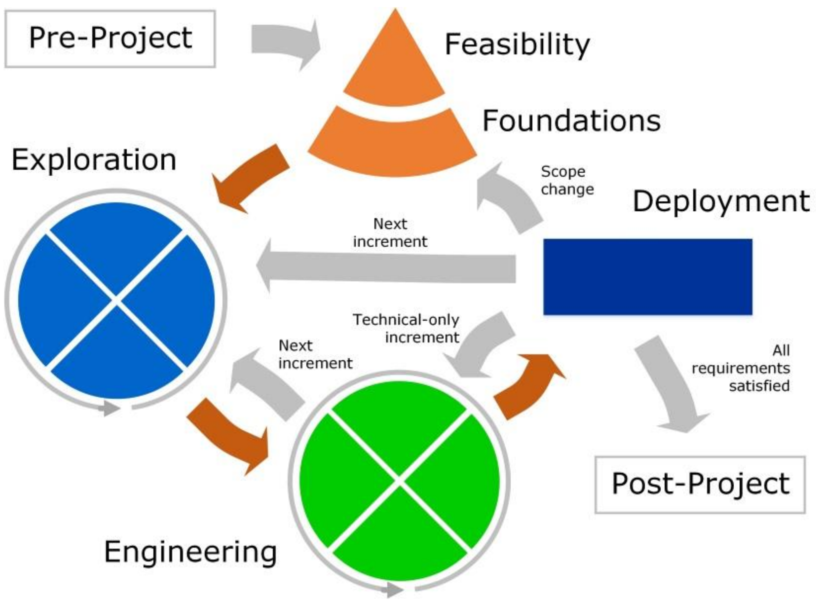

Figure 6 – DSDM Framework

- **Core Idea:** Fix time, cost, and quality, and use **prioritization (MoSCoW)** to guarantee delivery of the minimum usable subset of features. Strong focus on business needs.
- **Key Characteristics:**
    - Timeboxed and Iterative.
    - MoSCoW Prioritization (Must Have, Should Have, Could Have, Won't Have this time).
    - Focus on Fitness for Business Purpose.
    - Collaborative and Empowered Teams.
    - More defined roles and governance than Scrum.
- ⭐ **When to Use DSDM:**
    - Projects with tight, fixed deadlines and/or budgets.
    - Clear business case and need for strong governance.
    - Stakeholder buy-in for scope negotiation via prioritization is possible.
    - Often used in corporate or governmental environments.

---

# Model: Feature-Driven Development (FDD)

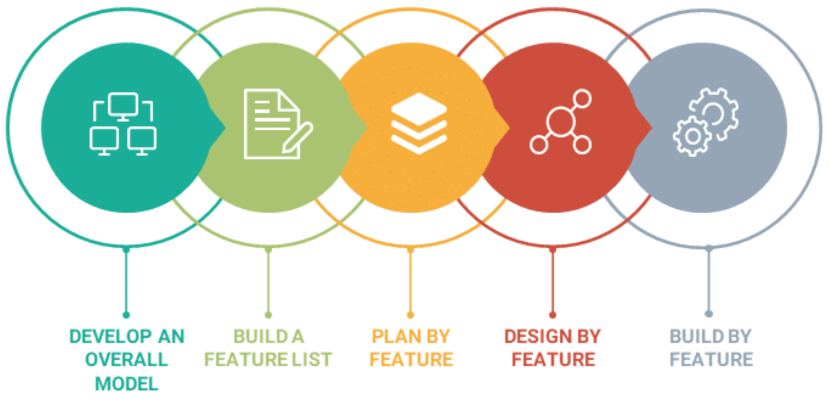

Figure 7 – FDD Model

- **Core Idea:** Design and build software based on **client-valued features**, using domain modeling and short, feature-focused iterations.
- **Key Characteristics:**
  - Emphasis on Domain Object Modeling upfront.
  - Feature Lists are central.
  - Develop by Feature (Design by Feature, Build by Feature).
  - Specific Roles (e.g., Chief Programmer, Class Owner).
  - Regular Builds & Progress tracking by feature.
  - Short Iterations (Often 2 weeks per feature set).
- ⭐ **When to Use FDD:**
  - Larger projects need more structure and modeling.
  - Situations where a clear understanding of the problem domain is crucial.
  - Need for detailed progress tracking at the feature level.
  - Teams are comfortable with modeling techniques.

---

**Final Though -** The best approach often involves understanding these models and tailoring practices to your specific context. Sometimes, teams even blend elements from different models! The key is to remain **Agile** – adaptive, collaborative, and focused on delivering value.
**(Attribution):** RMC Learning Solutions - Agile Concepts.