Comparison of Software Development Models and Methods

Here is a structured comparison of **all key concepts** from the **11 lectures**, categorized into **development models, Agile methodologies, estimation techniques, testing approaches, and lean principles**.

📌 Software Development Models

Model	Best Used When...	Not Suitable When...
Waterfall	Requirements are **fixed & well-defined**. Large projects with **low uncertainty**.	Requirements are **likely to change**. Late discovery of issues.
V-Model	**Strict quality control** is needed. **Safety-critical systems** (e.g., medical, aviation).	No flexibility for changes. Slow iteration speed.
Unified Process (UP)	Large projects needing **structured yet iterative** development.	Small, simple projects (too complex).
Spiral Model	High-risk projects requiring **continuous risk evaluation**.	Simple, small projects.
Prototyping	Exploring new ideas or **validating requirements** before full development.	Long-term projects needing structured workflow.
Agile (Scrum/Kanban/XP)	Requirements **change frequently**. Need **fast iterations & customer feedback**.	Large, highly regulated projects needing strict documentation.
RAD (Rapid Application Development)	Fast-paced projects needing **quick user feedback**.	Complex, long-term projects with multiple dependencies.

📌 Agile Methodologies

Method	Best For...	Not Suitable For...
Scrum	**Product development**, structured sprints, predictable work cycles.	Support/maintenance work (rigid sprint cycles).
Kanban	**Continuous workflow** (e.g., bug fixes, DevOps, customer support).	Projects needing **fixed-length iterations**.
Scrumban	Teams transitioning from Scrum to Kanban or **long-term projects with evolving scope**.	Teams needing **strict sprints** (Kanban is more flexible).
Extreme Programming (XP)	High-quality code & collaboration, **Test-Driven Development (TDD)**.	Large teams with **low developer interaction**.
Lean Development	**Optimizing workflows** to remove waste and speed up delivery.	Bureaucratic or **heavily regulated environments**.

📌 Software Estimation Techniques

Estimation Method	Best Used When...	Not Suitable When...
COCOMO (LOC-based)	Large **Waterfall projects**, legacy systems.	Agile projects (LOC is unpredictable).
Function Point Analysis (FPA)	**Business applications**, cross-language comparisons.	Systems where functionality is unclear.
Planning Poker	**Agile teams** estimating effort collaboratively.	Large projects needing historical data.
T-Shirt Sizing	Quick high-level effort estimates.	Highly detailed cost estimations.
Wideband Delphi	Expert-based estimation **reducing bias**.	Teams lacking experienced members.

📌 Testing Approaches

Testing Type	Best For...	Not Suitable When...

Unit Testing	Testing **individual functions** (TDD, XP).	End-to-end system validation.
Integration Testing	Ensuring **multiple modules work together**.	Standalone function validation.
System Testing	Checking if the full system **meets requirements**.	Isolated component testing.
Acceptance Testing	**Client validation** before deployment.	Internal development testing.
Blackbox Testing	Ensuring functionality **without knowing internal code**.	Debugging internal logic.
Whitebox Testing	**Code-level testing** (ensuring logic correctness).	User experience or system validation.
Model-Based Testing (MBT)	**Early testing using system models**.	Rapid development without formal models.

📌 Test Automation & Mocking

Technique	Best For…	Not Suitable When…
Test Doubles (Mocking, Stubbing, Fakes)	Isolating system components **during unit testing**.	Full-system integration tests.
Robot Framework	**Automated testing** for acceptance and regression testing.	Small manual test cases.

📌 Lean Development: Waste Elimination Strategies

Lean Principle	Best Used When…	Not Suitable When…
Eliminate Waste	Identifying & removing **non-value activities**.	Strict compliance projects needing documentation.
Amplify Learning	Fast-paced teams needing **continuous improvement**.	Stable projects with **fixed, unchanging scope**.

Decide Late	Systems with **high uncertainty & evolving requirements**.	Fixed, **fully defined** projects.
Deliver Fast	**Competitive industries** where speed matters.	Government or **heavily regulated projects**.
Empower the Team	**Agile teams** where self-organization is encouraged.	Bureaucratic **hierarchical teams**.
Build Integrity In	Systems needing **long-term maintainability**.	One-time use projects.
See the Whole	Large systems with **multiple teams & integrations**.	Small projects with **few dependencies**.

📌 When to Use Scrum vs. Kanban vs. XP vs. Lean

Method	Best For…	When to Avoid
Scrum	**Product teams, structured work**, predictable sprints.	Support work, high uncertainty.
Kanban	**Continuous workflows**, DevOps, customer support.	When fixed-length iterations are needed.
XP (Extreme Programming)	**High-quality code, pair programming, TDD**.	Large teams where collaboration is low.
Lean	**Maximizing efficiency, reducing waste**.	Compliance-heavy, **regulated industries**.

📌 Final Takeaways: How to Choose the Right Method?

1️⃣ **Use Waterfall/V-Model/UP** → When requirements are **fixed, structured, & documented**.
2️⃣ **Use Agile (Scrum, Kanban, XP)** → When requirements **change frequently & fast iterations are needed**.
3️⃣ **Use COCOMO/FPA for estimation** → When project scope is **predictable**. **Use Planning Poker/T-Shirt Sizing** for Agile.
4️⃣ **Use Lean** → When you need to **maximize efficiency & remove bottlenecks**.
5️⃣ **Use XP/TDD** → When **high-quality code & test coverage are crucial**.
6️⃣ **Use Kanban** → When **tasks flow continuously (support, bug fixing, operations, DevOps)**.
7️⃣ **Use Scrum** → When work needs **structured planning & sprint-based delivery**.

This table should help you quickly compare **when to use each approach** based on your project's needs! 🚀