eXtreme Programming:

Practices:

1. **Write unit test before coding.**
2. **Pair programming.** Disadvantages: Opposite personalities, newbie inefficiency, odd number.
3. **Refactor soon.** Don't refactor if: already has bugs, can't make it simpler, would stop shipment.
4. **Simple design.**
5. **Continuous integration.** Up to 12 people team.
6. **On-site costumer.**
7. **40-hours work week.**
8. **Coding standards.**
9. **User stories.**

User stories: Developers help customer to come up with story. Customer writes. Customer prioritizes. Developers estimate.

XP Idea: Accommodate change, less disagreement with customer.

XP Planning (1 – 7 days):

1. **User stories are written.**
2. **Release planning creates the schedule.**
3. **Iterate (7 – 21 days, total).**
    1) Break down stories.
    2) Add unfinished and new tasks, and bugs.
    3) Use project velocity and prioritization.
    4) Develop.
    5) Repeat until last version.
4. **Get project velocity.**
5. **Do acceptance tests.**
6. **Small release.**
7. **Repeat until finished.**

Tracking: Person that asks about the work done and work left. Used to decide about dropping tasks, adding tasks or reassigning tasks.

Uniqueness:

1) **Stand-up meeting**: Used for risk management and project tracking. Takes 15 min.
2) **Optimistic task and story estimations.**

Scrum:

Idea: Deliver the most value to the customer's product.

Teams: 3 to 9.

Scrum Master: Ensures Scrum principles.

Product owner: Part of your company's team. Represents the customer. Verifies acceptance criteria of user stories. Can stop Sprints. Communicates the changes.

Pigs versus Chickens: Everyone is fully committed (pigs).

Advantages:

1) Feels productive on a daily basis.
2) Minimizes unproductive meetings.
3) Promotes team bonding (pigs).
4) License to tell people to go away.

Disadvantages:

1) Lack of individual ownership.
2) No personal development due to fully commitment to the project.
3) Daily meetings can make team members nervous. So, better focus on tasks.

1. **Pre-game Phase:**
    1) **Planning:** Creation of Product Backlog (requirements), can take several days.
        - **Unlike XP:** *Usually* user stories are used.
        - **Unlike XP:** Team does prioritization.
    2) **Architecture:** High-level design based on backlog. A design review meeting is held to go over the proposals for the implementation decisions.
2. **Game Phase (Development Phase):**
    1) **Broken into a number of Sprints (iterations).**
    2) **Sprint Planning** (1 day)**:**
        - **Scope**: Customers, users and management. Creation of Sprint Backlog to be done in 1 sprint, based on items from Product Backlog.
        - **Plan:** Developers. User stories broken into tasks. Tasks go into the "to do" column of the task board.
    3) **Sprint:**
        - Functionality is developed.
        - Daily Scrum meeting: To track progress.
        - 1 to 4 weeks.
        - Includes requirements, analysis, design, implementation, testing, release.
        - Begins with a Sprint Planning session.
3. **Post-game Phase:**
    1) All requirements have been met, and no more can be invented.
    2) System ready for final release.

Formulas:

1. **Phase Containment Effectiveness:** $PCE_i = E_i / (E_i + D_i)$, goal = 1.
    - **i =** life cycle phase: inception, elaboration…
    - **E_i =** errors introduced and caught in phase i.
    - **D_i =** errors introduced in phase I and caught in phase i+1 or later.
2. **Statistical (beta) estimation:** LOC = (O+P+4R) / 6, confident estimates.
    - **O =** Optimistic
    - **P =** Pessimistic
    - **R =** Realistic
3. **Quality Metrics:**
    - **Modularity:** B/N (lower is better)
    - **Adaptability:** E/N (lower is better)
    - **Maturity:** UT / C0 + C1 (larger the better)
    - **N:** C0 + C1 + C2 (defects)
    - **UT:** Usage time (hours)
    - **E:** Cumulative effort spent fixing N (man-days)
    - **B:** Cumulative broken SLOC due to N (broken SLOC)

Agile

Agile: Set of value and principles used to guide software development.

Characteristics:

- **Don't have commit on time and budget.**
- **Don't support customers that require that all requirements are given in details.**
- **Don't support organization that are unable to relinquish authority.**

Values:

1. **Individuals and interactions over processes and tools:** Don't let wrong rules stop you. People have to understand what they are doing. Reduces bureaucracy.
2. **Working software over comprehensive documentation:** There is some documentation, but discussion is mainly done over the working software (software that adds value to the customer) itself.
3. **Customer collaboration over contract negotiation:** Avoid haven't to constantly CYA, instead, collaborative with the customer to give the most value to his business.
4. **Responding to change over responding a plan:** Agile shouldn't have *fixed* plan.

Principles:

1. Satisfy the customer business.
2. Welcome changes.
3. Frequently deliver (weeks to a few months).
4. Business people and developers must work together daily.
5. Motivate the individuals.
6. Value face-to-face conversation.
7. Working software is the measure of progress.
8. Sustainable development.
9. Technical excellence and good design.

10. Simplicity.
11. Team self-organization.
12. Team self-improvement.

Crystal, FDD and DSDM

Crystal: Family of methodologies which depends on the criticality of the software.

Idea: *Follow* certain rules. After knowing them well, *break* the rules. Finally, *leave* those rules and create your own rules.

Unique to crystal:

1. **Frequent *delivery*:** Not only releases frequently, but also integrate it into the customer's product.
2. **Osmotic communication.**

Common rules:

1. Cycles of 1 to 4 months.
2. Teams up to 80 people.

FDD: Feature Driven Development.

Characteristics:

1. Claims to be suitable for **critical systems**.
2. Focus on developing single **features**.

Uniqueness: Has well-defined roles.

1. **Key roles:** Project manager, chief architect, chief programmer, development manager, class owner, and domain expert.
2. **Supporting roles:** Manager, release manager lawyer/language guru, build engineer, tool smith, and system administrator.
3. **Additional roles:** Tester, deployers, technical writers.

DSDM: Dynamic Systems Development Method.

Uniqueness:

1. **Law of Diminishing Returns**: Use resources on the most valuable features (80% functionality, 20% time).
2. **Customer and developer can do whatever they want**. Managers don't have authority.

Standards for Management, Quality, and Process:

Capability Maturity Model (CMM), ranks the Software Development Process of a firm by using 5 levels of maturity: (1) Initial, (2) Repeatable, (3) Defined, (4) Managed, (5) Optimizing.

1. **Initial**: The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort.
2. **Repeatable**: Basic project management processes are stablished to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications. It has project management processes. You take a few metrics at the end of the project. But during the project, it's a black box.

3. **Defined**: The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software. It has software development processes, like SCRUM. You capture metrics during an iteration and project. But you can't adjust yourself during a project or iteration.
4. **Managed**: Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled. Do a lot of metrics. You capture metrics during an iteration and project. And you can adjust yourself during a project or iteration.
5. **Optimizing**: Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

Comparing CMM to CMMI:

1. CMMI is better documented.
2. CMMI is clearer about how to assess your methodology.
3. CMMI is clearer about how to tailor your methodology.

IEEE standards:

1. Can easily communicate with other companies.
2. Can remind you what is missing in your project by reading the standard.

IBM Agile Development

Core Principles:

1. Just enough, just in time (Kaizen).
2. Deliver a minimum viable product every sprint.

Agile methodology:

1. **Team is divided into squads**.
2. **Each squad** is responsible for **one feature**.
3. **Each squad** has a **scrum master** (guardian of the agile practice) and tech lead.
4. Has a **project owner**, but **unlike Scrum**, he also **oversees all squads**.
5. **Loosely coupled, tightly aligned**: Scrum of Scrums: regular meeting between scrum master to align.

Sprint:

1. **Story (Requirement)**: Description, owner, test owner, acceptance criteria, estimation.
2. **Epic**: Collection of stories that defines a feature.
3. **Sprint:** 2 weeks (most important).
4. **Milestone:** 4 sprints.
5. **Release:** 1-2 milestones.
6. **Sprint Retrospective.**
7. **Sprint Demo:** Only demo completed work (there are exceptions).

Backlog Grooming: Find the epics/stories that have highest priority for the next milestone.

1. **Backlog:** Prioritized list of stories to be completed in a milestone/release.
2. **Grooming:** A sprint (mid-milestone) to refine the backlog.

Coordinating Multi-Squad Teams:

1. **Use regular checkpoints**: Sprint demo, scrum of scrums, backlog grooming.
2. **Transparency of code**.
3. **Regular meetings**.
    a. **Design meeting**: Tech lead, architect, project owner debate about technical issues.