# -Scrum Overview-

Roles:

* Product owner
  + Represents stakeholders
  + Writes user stories, prioritized goals/product backlog
  + Defines requirements and verifies product matches said requirements
  + Set requirement priorities
  + Keeps stakeholders posted on project status
  + Demo’s project
  + Plans and announces releases
  + Add changes to next iteration or cancel current one
* Team member
  + Build application
  + Analysis
  + Designing
  + Programming
  + Testing
  + Documentation
* Scrum Master
  + Ensures Scrum practice is being followed
  + Challenges team to improve
  + Leads meetings
  + Fixes obstacles

Sprints:

* Incremental iterations
* 2-4 weeks long
* Result of sprint is a fully tested and approved piece of software
* Before each sprint team holds *sprint planning meeting*
  + Product owner decides which user stories should be selected
  + Team members ask for clarification and concerns
  + Selected stories are moved from backlog to *sprint backlog*
  + Team members break stories into tasks
* 10–15 minute *daily scrum meeting*  where Team members answer three questions
  + What did you do since the last scrum meeting?
  + What do you hope to accomplish before the next scrum?
  + What obstacles do you see in your way?
  + Scrum masters looks into obstacles and tries to fix them
* After sprint is completed end with *sprint review meeting*
  + Team members present PSIs(Potentially shippable increment) to Product owner
  + Product owner verifies that PSI matches goals
  + If the application can’t handle part of user stories, Product owner can flag that story as incomplete
* After *sprint review meeting,* hold *retrospective meeting*
  + Scrum master and Team members discuss recent sprint
  + What went well and how can we make it happen again?
  + What went poorly and how can we avoid that in the future?
  + How can we improve the next sprint?
* Planning Poker
  + Used to determine difficulty of tasks
  + Everyone gets a hand of cards with an ace, 2, 3, 5, 8, and king to represent hours estimated for the task’s completion, a king represents a task to big or complicated to talk about presently.
  + Product owner picks a task, and everyone lays a card face down for how long they think the task will take.
  + Everyone flips over their cards at the same time and those with the smallest and largest numbers are given time to explain why. After both sides have spoken everyone votes again and then points are tallied up and assigned to that task. This is done with every task.
* Burndown Chart
  + Burndown chart shows amount of work remaining over time
  + *Sprint* burndown chart shows amount of work for a sprint
  + *Product* burndown chart shows the amount of work remaining for the whole project
  + Measure amount of work in story points
* Velocity
  + A projects velocity represents the amount of work the team can perform during a sprint
  + To calculate velocity, add up the number of features the sprint delivered by story points
* Summary
  + Scrum is a project management and not development methodology
  + Scrum focuses on breaking large problems into smaller tasks and completing tasks with speed and efficiency.
  + Scrum focuses on creating a small workable product soon and building new and more complex versions of that product every sprint.
  + Scrum is flexible, allowing constant changes throughout development to meet customers everchanging requirements.
  + Possible methodologies to combine with Scrum
    - XP(Extreme Programming)-Development
    - Crystal Clear-Development
    - AUP-Management
    - DAD-Management
    - DSDM-Management
    - KANBAN-Management

# -Extreme Programming Overview-

Pair Programming:

* Two programmers sit in front of the same monitor(screen share can work for online work)
* One programmer is the driver/pilot who controls the keyboard and writes the code keeping up a steady monologue of what he is doing
* The other programmer is the observer/navigator who watches and reviews each line of code as it is being typed.
* Both programmers take turns switching roles

Roles:

* Customer
  + Defines requirements
  + Verifies that the application meets the users’ needs
  + Provides frequent feedback to keep development on track
* Tracker
  + Monitors the team members’ progress
* Programmer
  + Defines the application’s architecture and writes the code
* Coach
  + Helps the team work effectively, self-organize, and use good XP practices
* Tester
  + Helps the customer write and perform acceptance tests for use cases
  + Looks for missing requirements and holes in the design
* Administrator
  + Sets up and maintains the team members’ development tools

Values:

* Communication
  + Requirements must be communicated from the customers to the developers
  + Extensive collaboration, frequent interaction
* Simplicity
  + Simple design
  + Start with simplest application possible and add on features later
* Feedback
  + Frequent unit and integration tests
  + Customers give feedback through periodic reviews
  + Pair programmers give each other constant feedback
* Courage
  + Refactor code when necessary
  + Throw away code when necessary

XP practices:

* Have customer on site
  + Keep customer on site so that he can answer any questions that arise
  + Customer should have authority to make decisions without management approval
* Play planning game
  + Release planning
    - Team focuses on the next customer release
    - Team shuffles cards around with written stories on them and tries to determine how many of the cards can be implemented in time for the next customer release
  + Iteration planning
    - At the beginning of each iteration(1-3 weeks) team gets together to develop plan for that iteration
    - Team selects user stories from the current release plan starting with the most important ones
    - Developers pick the tasks that they will perform.
    - Each Developer estimates the amount of time for the task
    - Each task shouldn’t take over 3 days
* Use standup meetings
  + Brief meeting 15 minutes or less
  + All team members must attend
  + What did you complete last time
  + What are you going to complete this time
* Make frequent small releases
  + Get customers useful software as soon as possible
  + Get frequent feedback from customers
* Use intuitive metaphors
  + Use common metaphors to describe features
  + Example: shopping cart for websites
* Keep designs simple
  + Use simplest design you can come up with
  + May need to modify design later
* Defer optimization
  + Don’t complicate code to make it optimized if it is not necessary
  + Optimize later if necessary
* Refactor when necessary
  + Because of simple design you may need to redesign things later on
* Give everyone ownership of code
  + Anyone can change the code as necessary
* Use coding standards
  + Team should adopt coding standards and conventions
* Promote generalization
  + Everyone should understand every part of the project to a general extent
* Use pair programming
  + Constant code reviews
* Test constantly Integrate continuously
  + Test everything
  + Automate as much testing as possible
* Work sustainably
  + Set a working pace that everyone can keep up indefinitely
* Use test-driven and test-first development
  + Create the test for a specific function that you want to create before you write the code for the said function
  + Code should fail the first test and if not create a better test
  + Write the simplest code possible to past the test
  + Run the test again and any previous test
  + If code passes all tests, refactor if necessary to clean up code, then move onto then next function starting with creating the test
* Summary
  + XP management methodologies align closely with that of SCRUM
  + Focuses on quality code through pair programming and test-driven development
  + Uses quick and simple meetings and having a customer on site to increase speed of development
  + Pros:
    - Increased confidence in code
    - Reduced number of bugs
    - Increase communication
  + Cons:
    - Less programmers creating code at a time
    - Refactoring

# -Crystal Clear Overview-

Project criticality:

* Comfort
* Discretionary money
* Essential money
* Life

Seven common features of Crystal methods:

* Frequent iterations
  + Frequent iterations result in releases
  + In critical projects, only some of the releases might be delivered to the users
* Constant feedback
  + Team meets regularly to discuss development and ways it can be improved
  + Team meets with customer to keep the project on track
* Constant communication
  + Team members should be located at the same location so that they can communicate easily and frequently. For small projects they should be in the same room.
* Safety
  + Team members can express their opinions safely without fear of blame
  + Should finish on time and within budget
  + Consider a project’s criticality
* Focus
  + Team members should be given enough time to focus on their key items without interruptions
  + Focus of project should be clearly stated
  + Project leader should prioritize tasks
* Easy access to expert users
  + The developers should be able to talk to the expert users to ask questions and request feedback
  + Expert should be available to meet at least two hours per week
* Testing support
  + Automated testing
  + Continuous integration
  + Code management

Key assumption:

* Larger project needs for formalization
* Smaller project
  + Can get away with verbal understanding of customers’ needs
  + Informal developer meetings
  + Cross-functional team where everyone pitches in on everything

Roles:

* Sponsor
  + The person for whom the software is developed
  + Who will sign of on the finished product
* Senior designer
  + Knows how to design the software
  + Makes necessary technical decisions
* Programmer
  + Writes the code

Summary:

* “XP is more productive, Crystal Clear more likely to get followed”-Alistair Cockburn
* Much more informal methodology
* Expected to deliver production every 2-3 weeks, possibly shorter with non-released development iterations
* Take into consideration what the program will be used for when deciding how strict certain requirements such as testing need to be
* Emphasis on communication and in person collaboration whenever possible
* Pros
  + More likely to be followed than other stricter methodologies
  + Flexible
  + Promotes communication
* Cons
  + In person collaboration
  + Not as effective as other methodologies

# -Agile Unified Process Overview-

Seven disciplines:

* Model
  + Attempt to understand the problem being addressed, its restrictions, assumptions, and possible solutions
  + Result should be model of solution
* Implementation
  + Converts the model into an executable program
  + Basic testing such as unit testing
  + Code needs to work
* Test
  + Testing to find bugs
  + Validates implementation
* Deployment
  + Delivers the result
  + Setting up user’s work environment
  + Training and installation
* Configuration management
  + Manages the items produced by the project
    - Requirement documents
    - Designs
    - Source code(versions and changes)
* Project management
  + Identifying and mitigating risks
  + Assigning tasks
  + Tracking progress
  + Interacting with forces outside of project team
* Environment
  + Supports project by making sure it has everything
    - Hardware
    - Software
    - Training manuals
    - Administrative support

Summary:

* Significantly simpler than SCRUM
* Can be used to build on SCRUM as a management Methodology, not a development one
* Similar to UP in that multiple of the disciplines will be happening simultaneously just to different degrees
* Produces multiple releases with more iterations