Agile = a set of methods and methodologies, and a mindset or attitude that's shared by everyone on the team

- Welcoming change
- Working in small value-added increments
- Using build and feedback loops
- Learning through discovery
- Value-driven development
- Failing fast with learning
- Continuous delivery
- Continuous improvement

Organizational agility

- If just one member of an organization adopts an agile mindset, it can help that person become more effective. However, they will feel continually frustrated that others in the organization don't seem to realize what is important, or are focused on the wrong goals and metrics.
- If one team in the organization adopts agile principles and practices, it can help the team members become more effective at delivering their project work. However, they will feel inhibited or misunderstood by other groups or systems in the organization, such as the PMO or functional silos.
- If the entire organization adopts the agile way of thinking, then everyone will be working together to improve agility and the delivery of value. By adopting common goals and values, everyone's effectiveness will be enhanced.

Business value = total sum of tangible (monetary assets, stockholder equity, etc) and intangible (brand, good will, trademarks, etc) elements

Scope creep = the uncontrolled expansion to product or project scope without adjustment to time, cost, and resources

Gold plating = additional features/functions added without stakeholder noticing

Agile project constraints = scope (variable), time (fixed), cost (fixed), risks, quality, resources, customer satisfaction

Agile Manifesto = a set of values to get to an agile mindset

- Individuals and interactions over processes and tools = projects are undertaken by people, not tools, and problems get solved by people, not processes
- Working software over comprehensive documentation = need to deliver; documentation is just enough (barely sufficient), just in time (last responsible moment), and sometimes just because (avoid facing consequences)
- Customer collaboration over contract negotiation = be flexible and accommodating, rather than fixed and uncooperative
- Responding to change over following a plan = acknowledge things will change instead of suppressing changes and tracking static plans

Format of A over B addresses intention, focus, and effort; apply more of our focus, emphasis and intention to A than to B; consider projects from a value-based perspective

Agile Manifesto principles

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software. (Satisfy customers with great systems in timely manner)

- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage. (Welcome change)
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale. (Deliver frequently to get feedback early)
- Working software is the primary measure of progress. (Completed and delivered work is done and measured, partially completed work get no recognition)
- Business people and developers must work together daily throughout the project. (Engage with business regularly)
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done. (Motivate people)
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. (Face-to-face communications)
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely. (Maintain sustainable pace/ have a work-life balance)
- Continuous attention to technical excellence and good design enhances agility. (Maintain design)
- Simplicity – the art of maximizing the amount of work not done – is essential. (Keep it simple)
- The best architectures, requirements, and designs emerge from self-organizing teams. (Team creates architecture)
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly. (Reflect and adjust on the spot as it happens)

Common agile approaches and methodologies:

- Scrum
- Extreme Programming (XP)
- Lean Product Development
- Kanban
- Feature-Driven Development (FDD)
- Dynamic Systems Development Method (DSDM)
- Crystal

Process tailoring = when agile methodologies are customized to suit the team, not used “as is” or “out of the box”

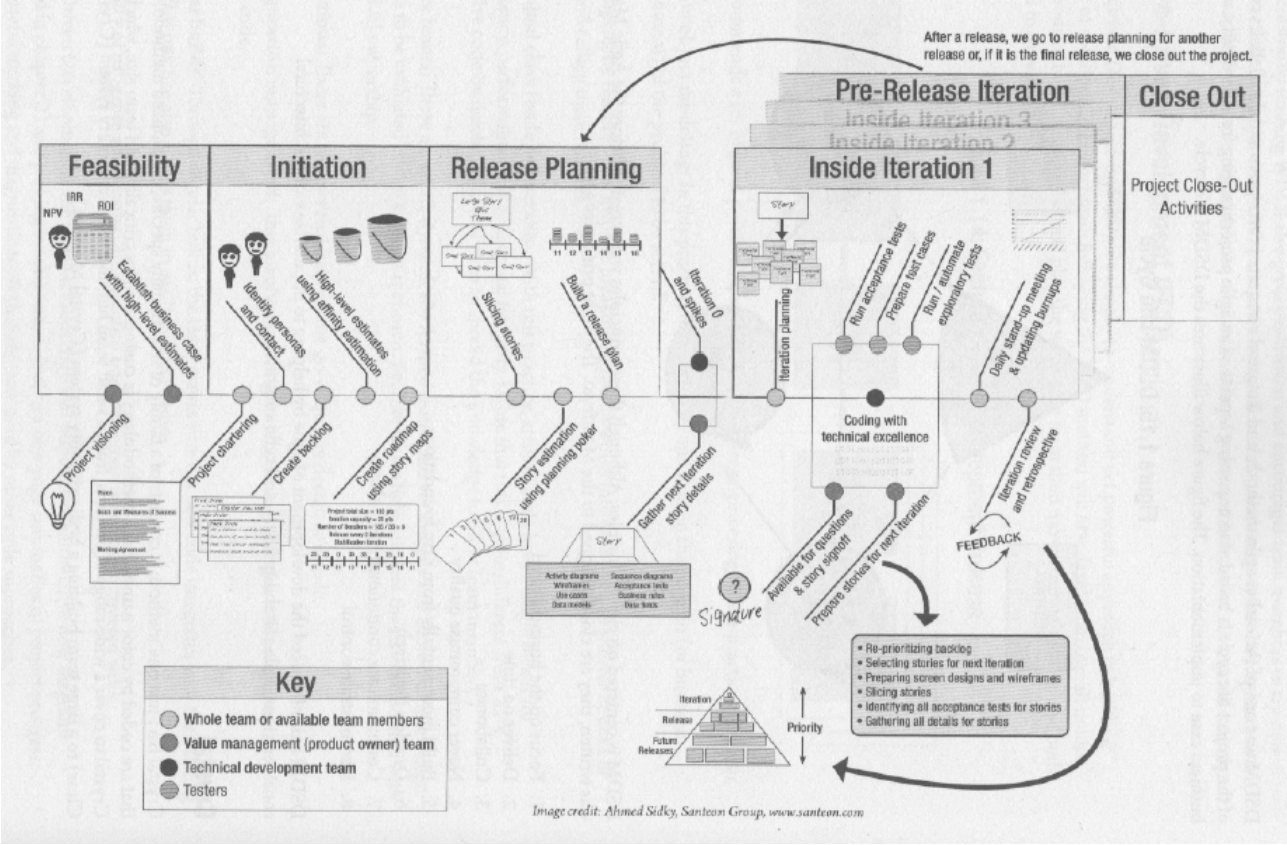
Sprint/ Iteration = time-boxed (time-limited) iteration; only product owner can cancel a sprint

Planning sprint = velocity-driven or commitment-driven; determine sprint goal and sprint backlog

Release = group of iterations that results a completion of a valuable deliverable

Increment = sum of all backlog items completed and delivered at the end of each sprint/ iteration

Agile process



Feature-Driven Development FDD

Practices

- Domain object modeling = break down problems into domain objects then model accordingly
- Developing by features = breaking down requirements into features
- Individual class (code) ownership = have a subject matter expert to oversee and own
- Feature teams = small teams to design and develop product
- Inspection = ensure quality
- Configuration management = tracking and monitoring changes in code and other areas
- Regular builds = integrate code and deliver frequently
- Visibility of progress and results = tracking of completed work

Dynamic Systems Development Method DSDM

Principles

- Focus on the business need
- Deliver on time
- Collaborate
- Never compromise quality
- Build incrementally from firm foundations
- Develop iteratively
- Communicate continuously and clearly
- Demonstrate control

Crystal = uses factors of criticality and team size to classify projects

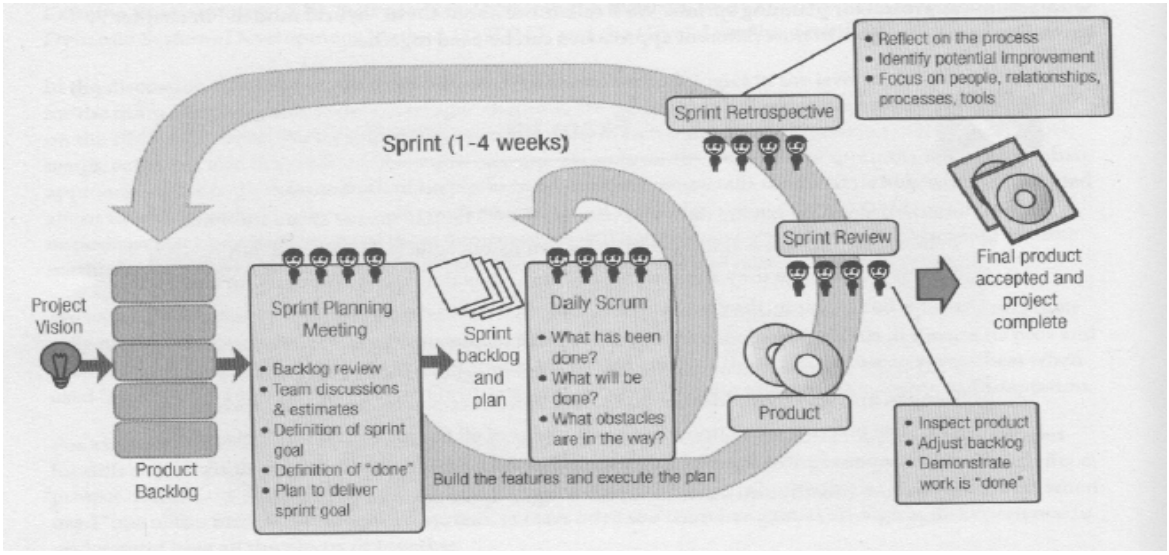
Methods

- Crystal Clear = team size 1 to 6
- Crystal Yellow = team size 7 to 20
- Crystal Orange = team size 21 to 40
- Crystal Red = team size 41 to 100
- Crystal Magenta = team size 101 to 200

Scrum

Pillars
- Transparency = giving visibility to those responsible for the outcome
- Inspection = doing timely checks
- Adaptation = adjusting to team's process

Values
- Focus
- Courage
- Openness
- Commitment
- Respect



Team Roles
- Development team = self-organizing, cross-functional
- Product owner = maximize product value, prioritize product backlog
- Scrum master = act as servant leader to development team, ensure Scrum methodology is used effectively

Activities
- Product backlog refinement = grooming/ refining the backlog
- Sprint planning meeting = determine what to deliver in coming sprint
- Daily scrum = 15 minute time-boxed meeting
- Sprint review = development team with scrum master demo the current product to product owner
- Sprint retrospective = development team and scrum master gather lessons learned and look for opportunities for improvement

Artifacts
- Product increment = development team and product owner determine what is acceptable, definition of “done”
- Product backlog = product owner's prioritized list of product requirements
- Sprint backlog = subset of product backlog, items to deliver for current sprint, updated by development team

Daily scrum aims to answer following questions for each team member:

1. What have I done since the last daily scrum?
2. What do I plan to do today?
3. Are there any impediments (obstacles) to my progress?

Scrum of scrums (multiple scrum teams with representatives for (daily) scrum meetings)

1. What has your team done since we last met?
2. What will your team do before we meet again?
3. Is anything in your team's way?
4. Are you about to put something in another team's way?

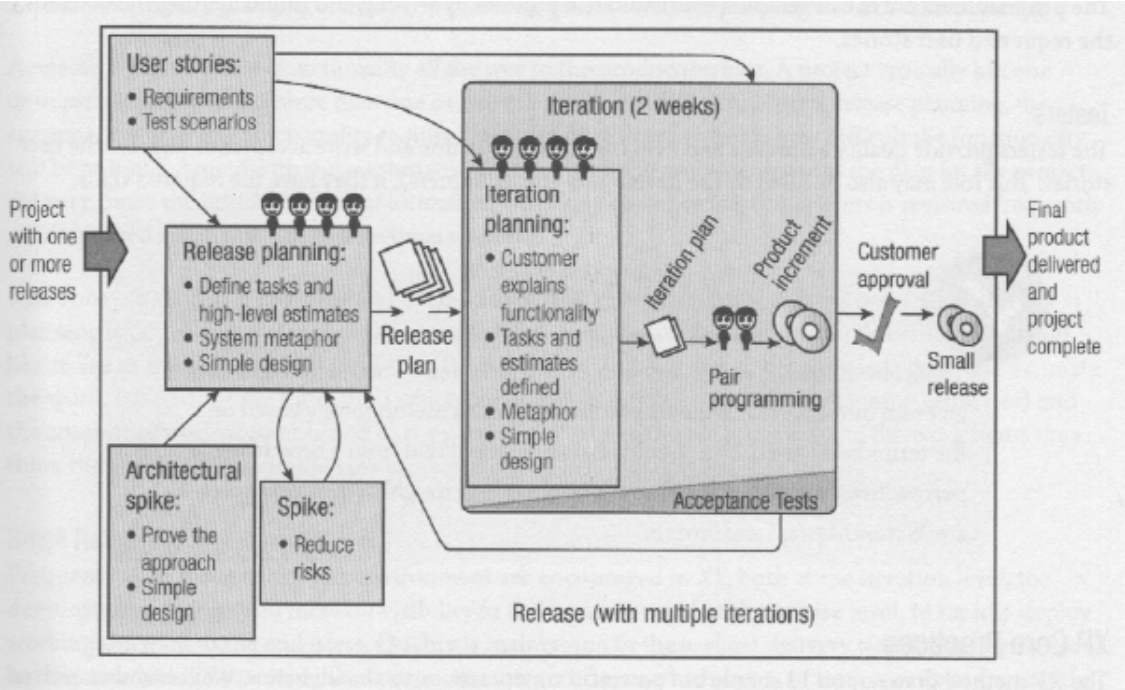
Extreme Programming XP

Values
- Simplicity
- Communication
- Feedback
- Courage
- Respect

Spike = work period to reduce threats and issues

Architectural spike = work period proof of concept

Risk-based spike = work period investigating threats/ risks



Team Roles
- Coach = mentor, facilitator to the team
- Customer = business representative
- Programmer = implement the product
- Tester = provide quality control and assurance

Practices
- Whole team = everyone in team get involved; generalists rather than specialists
- Planning games = release planning (to production) and iteration planning (for current iteration)
- Small releases = get customer feedback often
- Customer tests = perform tests to ensure functionality works
- Collective code ownership = any developer can improve the code base, shared knowledge
- Code standards = consistent approach to writing code
- Sustainable pace = maintain a work-life balance
- Metaphor = easy to understand terminologies and processes

- Continuous integration (CI) = integrate code frequently
- Test-driven development (TDD) = test cases are prepared/ built before coding
- Refactoring = reorganize code for better maintenance / enhancement
- Simple design = use simple structures to express the solution
- Pair programming = 2 developers work on same code base at the same time

Slack = XP practice such that optional or minor items are included in iteration

Lean Product Development

Concepts

- Eliminate waste = reduce partially done work, handoffs, unnecessary features, etc
- Empower the team = prevent micromanaging the team
- Deliver fast = release frequently
- Optimize the whole = align to organization's goals
- Build quality in = do testing and quality assurance throughout
- Defer decisions = make decisions and/or commitment as late as possible
- Amplify learning = communicate and feedback often, share knowledge

Waste	**Description**	**Example**
Partially done work	Work started, but not complete	Code waiting for testing Specifications waiting for development
Extra processes	Extra work that does not add value	Unused documentation Unnecessary approvals
Extra features	Features that are not required or are thought of “nice-to-haves”	Gold plating Technology features
Task switching	Multitasking between several different projects	People assigned to multiple projects
Waiting	Delays waiting for reviews and approvals	Waiting for document/ specification reviews and approvals
Motion	Effort required to communicate or move information or deliverables from one group to another	Distributed (not co-located) teams Handoffs
Defects	Defective documentation and software that need correction	Requirement defects Software bugs

Kanban

Principles
- Visualize the workflow = making the intangible items tangible through visuals
- Limit work in progress (WIP) = restricting the amount of work at any given time so that productivity can be maximized
- Manage flow = tracking the progress
- Make process policies explicit = communicate processes transparently and clearly to team
- Improve collaboratively = work together as a whole

Task (Kanban) board = a board showing the “to do”, “doing” and “done” tasks

Pull system = not using iterations but move tasks from left to right on the Kanban (task) board

WIP limit = increase productivity, reinforce iterations are not needed

Work in progress (WIP) problems
- Consumes investment capital and delivers no return on investment
- Hides bottlenecks in processes that slow overall workflow/ throughput and masks efficiency issues
- Represents risks of potential rework

Servant leadership

Duties
- Shield the team from interruptions = protect team from external distractions/ diversion
- Remove impediments (obstacles) to progress
- Communicate (and re-communicate) the project vision
- Carry food and water = provide resources

Principles
- Learn the team member's needs
- Learn the project's requirements
- Act for the simultaneous welfare of the team and the project
- Create an environment of functional accountability
- Have a vision of the completed project
- Use the project vision to drive your own behavior
- Serve as the central figure in successful project team development
- Recognize team conflict as a positive step
- Manage with an eye towards ethics
- Remember that ethics is not an afterthought, but an integral part of our thinking
- Take time to reflect on the project
- Develop the trick of thinking backwards

Practices
- Model desired behavior = demonstrate values such as honesty, forward-looking, competent and inspiring
- Communicate the project vision
- Enable others to act = create collaborative environment
- Be willing to challenge the status quo

Tasks
- Practice transparency through visualization = both progress and achievements as well as issues and setbacks
- Create a safe environment for experimentation = don't criticize for failure
- Experiment with new techniques and processes
- Share knowledge through collaboration
- Encourage emergent leadership via a safe environment

Emergent leadership = when a team member takes initiative and tries new approaches after gaining team approval

Responsibilities

Development Team/ Delivery Team

- Build the product increments, using agile practices and processes
- Regularly update information radiators to share their progress with stakeholders
- Self-organize and self-direct their working progress within an iteration
- Share their progress with each other in daily stand up meetings
- Write acceptance tests for the product increments
- Test and revise the product increments until they pass the acceptance tests
- Demonstrate the completed product increment to the customer in the iteration review meeting
- Hold iteration retrospectives to reflect on their process and continually improve it
- Perform release and iteration planning, including estimating the stories and tasks

Product Owner/ Customer/ Proxy Customer/ Value Management Team/ Business Representative

- Maximize the value of the product by choosing and prioritizing the product features
- Manage the product backlog, making sure that it is accurate, up to date, and prioritized by business value
- Make sure the team has a shared understanding of the backlog items and the value they are supposed to deliver
- Provide the acceptance criteria that the delivery team will use to prepare acceptance tests
- Determine whether each completed product increment is working as intended, and either accept it or request changes (in the iteration review meeting)
- May change the product features and their priority at any time
- Facilitate the engagement of external project stakeholders and manage their expectations
- Provide the due dates for the project and/or its release
- Attend planning meetings, reviews, and retrospectives

Scrum Master/ Coach/ Team Leader

- Act as servant leader to the delivery team, helping them improve and removing barriers to their progress
- Help the delivery team self-govern and self-organize, instead of governing and organizing them
- Serve as a facilitator and conduit for communication within the delivery team and with other stakeholders
- Make sure the delivery team's plan is visible and its progress is radiated to stakeholders
- Act as a coach and mentor to the delivery team
- Guide the team's agile process and make sure their agile practices are being used properly
- Help the product owner manage the product backlog
- Help the product owner communicate the project vision, goals, and backlog items to the delivery team
- Facilitate meetings (planning, reviews, and retrospectives)
- Follow up on issues raised in stand up meetings to remove impediments so that the team can stay on track

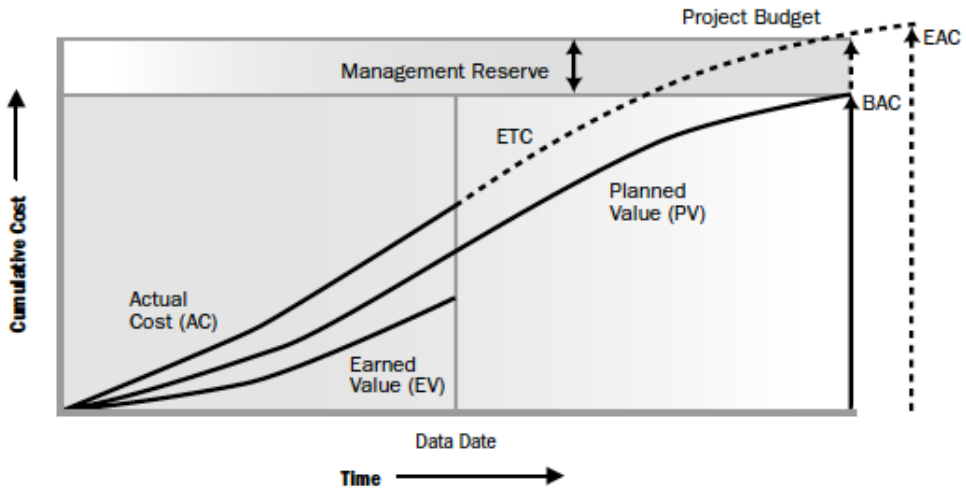
Project Sponsor

- Serve as the project's main advocate within the organization
- Provide direction to the product owner about the organization's overall goals for the project
- Focus on the big picture of whether the project will deliver the expected value on time and on budget
- Get invited to the iteration review meetings to see the product increments as they are completed, but might not attend

Project selection

- ROI = higher the better
- NPV = higher the better
- IRR = higher the better
- Payback period = shorter the better
- Benefit to cost ratio = the higher the better (benefit same as revenue)

Earned Value Management



budget at completion BAC = allocated budget

planned value PV = BAC x planned % complete = EV / SPI = EV - SV

earned vale EV = BAC x actual % complete = SPI x PV = SV + PV = CPI x AC = CV + AC

actual cost AC = EV / CPI = EV - CV

schedule performance index SPI = EV / PV <1 behind schedule, >1 ahead of schedule

schedule variance SV = EV - PV <0 behind schedule, >0 ahead of schedule

cost performance index CPI = EV / AC <1 over budget, >1 within budget

cost variance CV = EV - AC <0 over budget, >0 within budget

to complete performance index TCPI = (BAC - EV) / (BAC - AC) for within budget
TCPI = (BAC - EV) / (EAC - AC) for over budget
high TCPI means tight budget

estimate at completion
EAC = BAC/CPI if CPI is expected to be same for remaining of project
EAC = AC + BAC - EV if future work will be accomplished at planned rate
EAC = AC + bottom up ETC if initial plan no longer valid
EAC = AC + [(BAC-EV) / (CPI x SPI)] if both CPI and SPI influence remaining work

estimate to completion ETC = EAC - AC assuming work is proceeding on plan
variance at completion VAC = BAC - EAC <0 over budget, >0 within budget

Value prioritizing methods

- MoSCoW = Must have, Should have, Could have, Would like to have
- Monopoly money = having Monopoly money as project budget, distribute money to different business features
- 100-point method = for each stakeholder having 100 points, distribute points to his/her own requirements
- Dot voting or multi voting = given a set of dots, stakeholders distribute these dots to different requirements
- Kano analysis = graph or chart categorizing requirements into delighters/exciters, satisfiers, dissatisfiers, indifferent
- Requirements prioritization model = mathematical model to categorize requirements bases on benefit, penalty, cost and risk
- Relative prioritizing/ranking = rank requirements that make up the minimal viable product (MVP) or minimal marketable features (MMF)

Minimal viable product (MVP) or minimal marketable features (MMF) = the requirements or features that make up a working product, containing the must have features

Cumulative flow diagram CFD = diagram aiming to identify project constraints and bottlenecks

Agile Contracts

- Money for nothing and change for free = start with fixed price contract that includes time and material for additional work; “change for free” clause is added such that customers need to work on every iteration; “money for nothing” clause allows customers to terminate project early given project value (eg ROI) is met
- Graduated fixed price contract = both parties (customers/buyers and vendors/sellers) share some of the risks and rewards associated with schedule variance (pay at higher rate if deliver early, pay at lower rate if deliver late, pay at standard rate if deliver on time)
- Fixed price work packages = fixed price contracts split into “work packages” such that each work package has its own contract, allowing re-estimation or reprioritization as project progresses
- Customized contracts = mixing the various contract terms and conditions of the other approaches

Exploratory testing = aim to find edge cases (under extreme conditions), system/product boundaries, unanticipated system/product behavior, etc

Usability testing = aim to find out how end users respond to product under realistic conditions

Gulf of evaluation = when different person envision the end product differently through miscommunication and misinterpretation

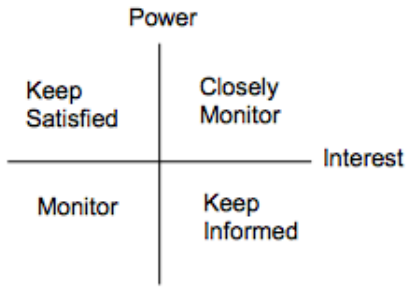
Stakeholder = An individual, group, or organization who may affect, be affected by, or perceive itself to be affected by a decision, activity, or outcome of a project, program, or portfolio

Stakeholder engagement principles

- Get the right stakeholders
- Cement stakeholder involvement
- Actively manage stakeholder interest
- Frequently discuss what “done” looks like
- Show progress and capabilities
- Candidly discuss estimates and projections

Level of stakeholder engagement = Unaware → Resistant → Neutral → Supportive → Leading

Classification Models	**Attributes**
Power/interest grid	Authority and concern
Power/influence grid	Authority and involvement
Influence/impact grid	Involvement and ability to change
Salience model	Power (authority), urgency (attention) and legitimacy (involvement)



Agile project charter = document issued by sponsor that formally authorizes project manager to start work; usually less detail than traditional project charters; contains, but not limited to, the following:

- Project title
- Project description
- Business need
- Project justification
- Pre-assigned resources
- Stakeholders
- Initial stakeholder requirements
- Product description and/or deliverables
- Project manager authority level
- Constraints and assumptions

- Sponsor approval & signature

Agile modeling = any representation of the product usually visual such as use case diagram, sequence diagram, flow chart, data/ domain model, screen design, etc

Wireframe = mock up of the product

Persona = fictional character representing users and audience

Information radiator = highly visible display of information eg charts, graphs, summaries of project data etc

Collaboration (innovation) games = facilitated workshops or exercises to better understand issue and reach consensus on options and solutions; games can include:

- requirement elicitation
- functionality/feature identification
- risk/threat and opportunity identification
- prioritization
- value and cost analysis

Conflict resolution strategies

- Withdrawal (avoid situation)
- Smoothing (play down situation)
- Compromising (give up something)
- Forcing (following someone else's solution)
- Collaborating (get everyone viewpoint)
- Confronting (actually solving problem)

Participatory decision making

- simple voting = for/yes or against/no
- Thumbs up/down/sideways = up means for/yes, down means against/no, sideways means neutral and may need to discuss
- Fist of five = use number of fingers to show the level of support (5 fingers full support) or resistance (1 finger full support)

Shu-Ha-Ri Model

- Shu = obeying the rules
- Ha = consciously moving away from the rules
- Ri = unconsciously finding an individual path

Adult Skill Acquisition = novice → advanced beginner → competent → proficient → expert

Team Building Stages = Forming → Storming → Norming → Performing → Adjourning

Situational leadership = Directing → Coaching → Supporting → Delegating

Co-location = team members are physically (within 33 feet or 10 meters) or virtually grouped together

Team space or war room = area for co-located teams

Cave = an isolated area for team member to work alone

Tacit knowledge = common, unwritten, knowledge of things

Osmotic communication = information flow between team members working in close proximity of each other (allow conversation to be overheard of others in the area)

Burn-down/ burn-up chart = line chart, updated daily, to show the amount of work (story points) left to do (burn-down) or completed (burn-up); burn-up chart has separate line

indicating scope changes

Velocity = measure of team's capacity for work per iteration

Agile planning principles:

- Plan at multiple levels
- Engage the team and the customer in planning
- Manage expectations by frequently demonstrating progress and extrapolate velocity
- Tailor processes to the project's characteristics
- Update the plan based on the project's priorities
- Ensure encompassing estimates that account for risks, distractions, and team availability
- Use appropriate estimate ranges to reflect the level of uncertainty in the estimate
- Base projections on completion rates
- Factor in diversions and outside work

Progressive elaboration = the iterative process of increasing the level of detail in a project management plan as greater amounts of information and more accurate estimates become available

Rolling wave planning = an iterative planning technique in which the work to be accomplished in the near term is planned in detail, while the work in the future is planned at a higher level

Ideal time = uninterrupted time to work on task

Elapsed time = time spent on task including buffers or contingency

Agile planning steps = size (how big is it) → estimate (how long to complete it) → plan (when to do it)

User story = a way to express very specific need of a user, each user story has a story point attached
User story template = As a <role>, I want <functionality>, so that <business benefit>

Story point = a measure of estimating how long a user story should take, rather than using absolute time (days or hours); story points follow the Fibonacci sequence (1,2,3,5,8,13,21,...)

Progression through face-to-face communication = card (documenting user story requirement/ info), conversation (process of eliciting requirements, enhancing card info), confirmation (customer accepting result)

Guideline for writing good user stories INVEST = independent, negotiable, valuable, estimatable, small, testable

Affinity estimating = an approach to compare the estimation of tasks/ story points using a board

T-shirt sizing = high level estimation before breaking down into user stories

Story map = tool for visualizing entire product backlog, listing the backbone (product categories) and walking skeleton (minimal core features)

Product roadmap = visual depiction of product releases

Planning poker = technique estimating how to tackle user stories and assigning story points

Technical debt = the cost of not doing the regular cleanup, maintenance, and standardization while the product is being built or the cost of fixing something

Lead time = how long it takes for the entire process

Throughput = (average) amount of work that can be processed by the delivery team in a given time period

Productivity = rate of efficiency at which the work can be done

Cycle time = how long it takes to complete one component of a process; WIP/ throughput

Common cause = average day-to-day differences of doing work

Special cause = greater degree of variance that are caused by special or new factors

Expected Monetary Value EMV = ABS[SUM(probability x impact)] where impact can be positive or negative

Kaizen = process of continuous improvement

Plan → Do → Check → Act = Plan → Develop → Evaluate → Learn

Value stream mapping = process of listing or mapping the process flow that measures value

Total cycle time = how long it takes to do the value-added tasks & nonvalue-added tasks; value-added time + nonvalue-added time
Value-added time = time during cycles where value is added to the process
Nonvalue-added time = time during cycles where no value is added such as wastes, delays, constraints
Process cycle efficiency = value-added time / total cycle time * 100

Retrospective = session to look back what went good and bad and look for improvements

Team member mood at start of retrospective ESVP = explorers (want to get as much as possible), shoppers (looking for improvements), vacationers (happy to be doing something different), prisoners (don't want to be here)

Retrospective progression = set the stage → gather data → generate insights → decide what to do → closing

Five Whys = 5 questions asking “why” a problem occurred aiming to find the root cause

Fishbone analysis = visual to identify the factors that caused a problem aiming to find the root cause

Short subject = technique to categorize follow-up actions

SMART goals = goals that fulfill the following: specific, measurable, attainable, relevant, timely