

# Agile Principles and Mindset

---

DOMAIN I

A solid blue horizontal bar spanning the width of the slide, located at the bottom.

# What is Agile

---

- Developed for Software projects, but it is a methodology that can be used on all Projects types
- Agile is an umbrella term that is used to refer to different types of iterative development
- Scrum is the most common method of agile, there are others such as extreme programming (XP), lean development, and Kanban.

# Agile vs. Traditional Project Management

---

- Agile builds in increments vs. as a whole
- Agile does planning throughout vs. done all at once
- Agile delivers products over time vs. all at once
- Customers sees value faster vs. at the end
- Agile wants changes vs. discouraging changes

# Agile Benefits

---

- Customer involved throughout the life cycle
- Greater Customer Interaction with all stakeholders
- Constant Feedback is required to stay current and successful
- Greater Value up front
- Change is welcomed by all stakeholders

# Agile Concurrent Development

---

- Fund incrementally – opt to extend, redirect or cancel at a very granular level
- Deliver & realize value steadily
- Validate designs with users & customers
- Continuously adapt to risk and change
- Integrate early & often

# Agile *Declaration of Interdependence (DOI)*

---

*Agile and adaptive approaches for linking people, projects and value*

*We are a community of project leaders that are highly successful at delivering results. To achieve these results:*

*We **increase return on investment** by making continuous flow of value our focus.*

*We **deliver reliable results** by engaging customers in frequent interactions and shared ownership.*

*We **expect uncertainty** and manage for it through iterations, anticipation, and adaptation.*

*We **unleash creativity and innovation** by recognizing that individuals are the ultimate source of value, and creating an environment where they can make a difference.*

*We **boost performance** through group accountability for results and shared responsibility for team effectiveness.*

*We **improve effectiveness and reliability** through situationally specific strategies, processes and practices.*

©2005 David Anderson, Sanjiv Augustine, Christopher Avery, Alistair Cockburn, Mike Cohn, Doug DeCarlo, Donna Fitzgerald, Jim Highsmith, Ole Jepsen, Lowell Lindstrom, Todd Little, Kent McDonald, Pollyanna Pixton, Preston Smith and Robert Wysocki.

# Agile Mindset

---

Welcoming change

Working in small value increments

Using build and feedback loops

Learning through discovery

Value -driven development

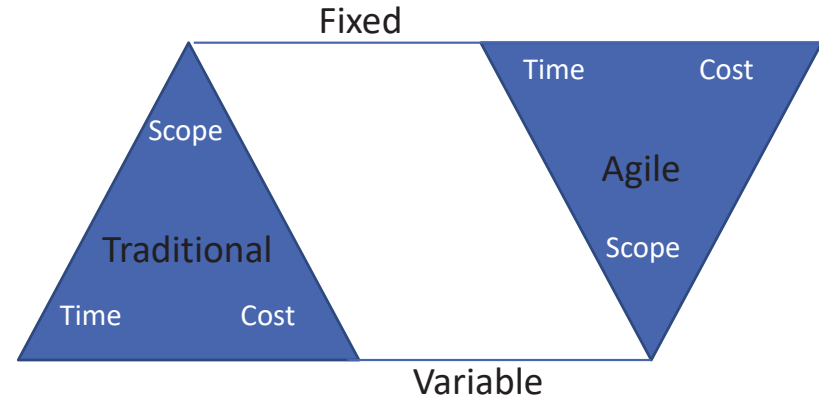
Failing fast with learning

Continuous delivery

Continuous improvement

# Inverting the Triangle

---





# Agile Manifesto

---

Create in 2001

Contains:

- 4 values
- 12 guiding principles

<https://agilemanifesto.org/>

# The Agile Manifesto Values

---

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals & interactions	over	Processes & tools
Working software	over	Comprehensive documentation
Customer collaboration	over	Contract negotiation
Responding to change	over	Following a plan

That is, while there is value in the items on the **right**, we value the items on the **left** more.

[www.agilemanifesto.org](http://www.agilemanifesto.org)

# Individuals and interactions over processes and tools

---

- While processes and tools will likely be necessary on our projects, we should focus the team's attention on the individuals and interactions involved.
- Projects are undertaken by people, not tools
- Problems get solved by people, not processes
- Projects are ultimately about people

## Working software over comprehensive documentation

---

- Focus on the delivering value vs. paperwork.
- Agile documents should be barely sufficient
- Done just in time
- Done just because
- Delivering software that does what it should comes first, before creating documentation.
- Agile dramatically simplify the administrative paperwork relating to time, cost, and scope control

## Customer collaboration over contract negotiation

---

- Be flexible and accommodating, instead of fixed and uncooperative
- Manage change, don't suppress change
- Shared definition of "done"
- Requires trusting relationship

## Responding to change over following a plan

---

- Spend effort and energy responding to changes
- Software projects tend to have high rates of change

# Agile Guiding Principles 1-3

---

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

# Agile Guiding Principles 4-6

---

4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.



# Agile Guiding Principles 7-9

---

- 7. Working software is the primary measure of progress.
- 8. Agile processes promote sustainable development. The sponsors, developers)and users should be able to maintain a constant pace indefinitely.
- 9. Continuous attention to technical excellence and good design enhances agility.

# Agile Guiding Principles 10-12

---

- 10. Simplicity; the art of maximizing the amount of work not done is essential.
- 11. The best architectures, requirements and designs emerge from self-organizing teams.
- 12. At regular intervals, the team reflects on how to become more effective then tunes and adjusts its behavior accordingly.

# Agile Methods

---

- Over 12 agile methodologies
- Scrum
- Extreme Programming (XP)
- Kanban Development
- Lean Software Development

# Agile Terms

---

**Product Owner** - Designated person that represents the customer on the project

**Agile Project Manager/Scrum Master** – Manages the agile project

**Product Backlog** - Project requirements from the stakeholders

**Sprint Planning Meeting**- Meeting done by the agile team to determine what features will be done in the next sprint

**Sprint Backlog** – Work the team selects to get done in the next sprint

**Sprint** - A short iteration where the project teams work to complete the work in the sprint backlog, (1-4 weeks typical)

**Daily Stand Up Meeting** - A quick meeting each day to discuss project statuses, led by the Scrum Master. Usually 15 minutes

**Sprint Review** – An inspection done at the end of the sprint by the customers

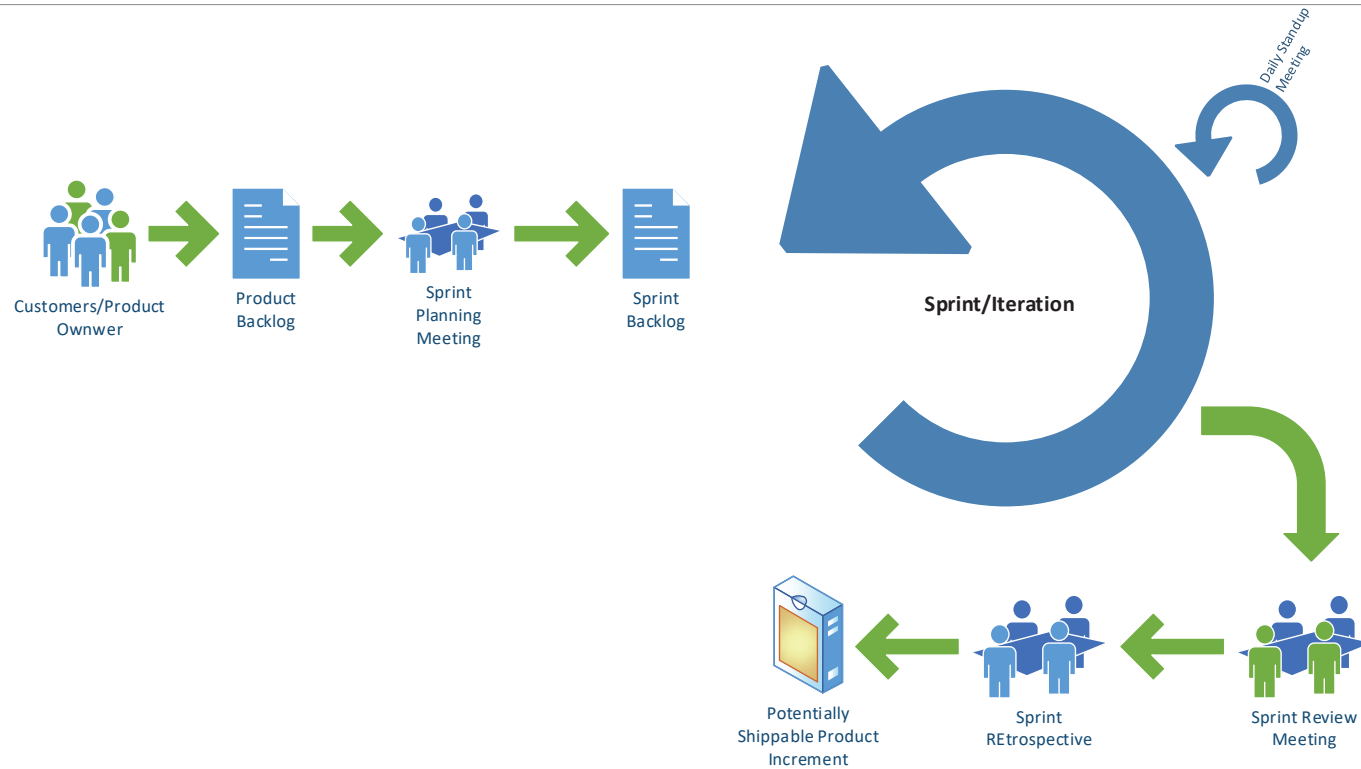
**Retrospective** – Meeting done to determine what went wrong during the sprint and what when right. Lesson learned for the sprint.

**Partial Completed Product** - Customers Demo the product and provides feedback. This feedback adjust the next Sprint priorities

**Release** - Several Sprints worth of work directed to operations for possible rollout and testing

**Sprint = Iteration**

# Agile Process



# Scrum

---

Set of team guidance practices, roles, events, artifacts, and rules

Based on three pillars of Transparency, Inspection, and Adaptation:

- Transparency
  - Visibility to those responsible for the outcome
- Inspection
  - Timely checks on how well a project is progressing toward goals
  - Looks for problematic deviations or differences from goals
- Adaptation
  - Adjusting a process to minimize further issues if an inspection shows a problem or undesirable trend

# Scrum Roles & Responsibilities

---

## Product Owner

- Owns Product vision
- Defines features, decides on release date and content
- Responsible for market success
- Prioritizes features according to market value
- Can change features and priorities every Sprint

## ScrumMaster

- Responsible for facilitating process
- Focuses Team and protects them from external interruption
- Looks for ways to enhance productivity
- Assists Product Owner in leveraging Scrum

# Scrum Roles & Responsibilities

---

## Development Team

- Small group containing all necessary project skills
- Focuses on steady delivery of high quality features
- Generates options for delivery
- Manages own work within Sprints



# Scrum Activities

---

The Scrum methodology refers to several different types of activities:

1. sprint planning meeting
2. sprints
  - Daily stand-up meeting
3. sprint review meeting
4. sprint retrospectives.

# Sprint Planning Meeting

---

- ✓ Used to determine what work will be done in that sprint and how the work will be achieved.
- ✓ The development team predicts what can be delivered based on estimates, projected capacity, and past performance to define the sprint goal.
- ✓ The development team then determines how this functionality will be built and how the team will organize to deliver the sprint goal.
- ✓ Output of this will be the sprint backlog. The work to get done in the next sprint.

# Sprints

---

- ✓ A sprint is a **timeboxed** (time-limited) iteration of 1-4 weeks to build a potentially releasable product
- ✓ Each sprint includes a sprint planning meeting, daily Scrum, the actual work, a sprint review meeting, and the sprint retrospective
- ✓ During the sprint, no changes are made that would affect the sprint
- ✓ The development team members are kept the same throughout the sprint

# Daily Scrum (or Standup)

---

- ✓ A 15-minute time-boxed activity for the Development Team to synchronize activities and create a plan for the next 24 hours
- ✓ Should be held at the same time and place each day
- ✓ Each team member should answer 3 questions:
  1. What did you do yesterday?
  2. What will you do today?
  3. Are there any impediments in your way?

# Sprint Review

---

- ✓ Takes place at the end of the Sprint
- ✓ Designed to gather feedback from stakeholders on what the Team has completed in the sprint
- ✓ Team demonstrates work that was completed during the sprint
- ✓ To create a conversation between the Team and the stakeholders about how to make the product better
- ✓ should be time boxed to no more than an hour per week of Sprint

# Sprint Retrospective

---

- ✓ Opportunity for the Team to inspect and create a plan for improvements to be done during the next Sprint.
- ✓ Team discusses:
  - ✓ What went well
  - ✓ What went wrong
  - ✓ What to do more of
  - ✓ What to do less of

# Scrum Artifacts

---

- ✓ Product increment
  - ✓ Part of the product that is complete after each sprint
- ✓ Product Backlog
  - ✓ Prioritized list of valuable items to deliver
- ✓ Sprint Backlog
  - ✓ List of committed items to be addressed within Sprint

# Product Backlog

---

- ✓ Prioritized list of all work that needs to be done to complete the product
- ✓ List is dynamic, it evolves as the more work is added and prioritized
- ✓ Items in it are prioritized by the product owner and are sorted by value
- ✓ Most valuable items are listed first
- ✓ Constantly being refined as more work is added to it.
- ✓ Team and product owner will “groom the backlog”.



# Product Increment

---

- ✓ Part of the product that is done after each sprint
- ✓ Done to get feedback after each sprint
- ✓ The product owner and team needs to agree upon the “definition of done” before the team starts working on the product

# Sprint Backlog

---

- ✓ The sprint backlog is the set of items from the product backlog that were selected for a specific sprint.
- ✓ The sprint backlog is accompanied by a plan of how to achieve the sprint goal, so it serves as the development team's forecast for the functionality that will be part of the sprint.
- ✓ It is a highly visible view of the work being undertaken and may only be updated by the development team.

# Definition of Done (DoD)

---

Definition of Done (DoD) is a shared understanding of what it means when work is considered done, it should be defined at the beginning of the project, and it applies globally to the project.

Definition of Done (DoD) is a crucial element of a successful scrum software development

Might include things such as:

- DoD for Unit & functional tests.
- DoD Documentation.
- DoD for a Writing code.

# Extreme Programming (XP)

---

Software development centric agile method

Focus software development good practices

Scrum at the project management level focuses on prioritizing work and getting feedback

# XP Core Values

---

## Simplicity

- Reduce complexity, extra features, and waste
- “ Find the simplest thing that could possibly work”

## Communication

- Team members know what is expected of them and what other people are working on
- Daily stand-up meeting is key communication component

## Feedback

- Get impressions of correctness early
- Failing fast allows for faster improvement

# XP Core Values

---

## Courage

- Allow our work to be entirely visible to others

## Respect

- People work together as a team and everyone is accountable for the success or failure of the project
- Recognize people work differently and respect those differences

# XP Roles

---

## Coach

- Acts as a mentor, guiding the process and helping the team stay on track. Is a facilitator helping the team become effective.

## Customer:

- Business representative who provides the requirements, priorities, and drives the business direction for the project.

## Programmers

- Developers who build the product. Writes the codes.

## Testers

- Helps the customer define and write the acceptance tests for the user stories.

Product Owner and Customer are equivalent  
ScrumMaster and Coach are equivalent

# XP Practices

---

## Planning Activities (Games):

- Release Planning:
  - Push of new functionality all the way to the production user
  - Customer outlines the functionality required
  - Developers estimate difficult build
- Iteration Planning:
  - Short development cycles within a release (Scrum calls "sprints")
  - Conducted at start of every iteration, or every two weeks
  - Customer explains functionality they would like in iteration
  - Developers break functionality into tasks and estimate work
  - Based on estimates and amount of work accomplished in previous iteration,



# XP Practices

---

## Small Releases:

- Frequent, small releases to test environments
- Demonstrate progress and increase visibility for the customer
- Quality is maintained: Rigorous testing or Continuous integration

## Customer Tests:

- Customer describes one or more tests to show software is working
- Team builds automated tests to prove software is working.

## Collective Code Ownership:

- Any pair of developers can improve or amend any code
- Multiple people work on all code, which results in increased visibility and knowledge of code base
- Leads to a higher level of quality; with more people looking at the code, there is a greater chance defects will be discovered.
- Less risk if programmer leaves, since knowledge is shared

# XP Practices

---

## Code Standards:

- Follow consistent coding standard
- Code looks as if it has been written by a single, knowledgeable programmer

## Sustainable Pace:

- While periods of overtime might be necessary, repeated long hours of work are unsustainable and counterproductive
- The practice of maintaining a sustainable pace of development optimizes the delivery of long-term value

# XP Practices

---

## Metaphor:

- XP uses metaphors and similes to explain designs and create a shared technical vision.
- These descriptions establish comparisons that all the stakeholders can understand to help explain how the system should work.
- For example, “The invoicing module is like an Accounts receivable personnel who makes sure money collected from our customers”.

## Continuous Integration:

- Integration involves bringing the code together and making sure it all compiles and works together.
- This practice is critical, because it brings problems to the surface before more code is built on top of faulty or incompatible designs.

# XP Practices

---

## Test -Driven Development (TDD):

- The team writes tests prior to developing the new code.
- If the tests are working correctly, the initial code that is entered will fail the tests
- The code will pass the test once it is written correctly.

## Pair Programming:

- In XP, production code is written by two developers working as a pair to write and provide real-time reviews of the software as it emerges.
- Working in pairs also helps spread knowledge about the system through the team.

# XP Practices

---

## Simple Design:

- Code is always testable, browsable, understandable, and explainable
- Do the simplest thing that could possibly work next. Complex design is replaced with simpler design
- The best architectures, requirements, and designs emerge from self-organizing teams

## Refactoring:

- Remove redundancy, eliminate unused functionality, and rejuvenate obsolete designs
- Refactoring throughout the entire project life cycle saves time and increases quality
- Code is kept clean and concise so it is easier to understand, modify, and extend

# Some Basic Terminology Review

---

Scrum	Extreme Programming (XP)	Definition
Sprint	Iteration	Fixed-length period of time (timebox)
Release	Small Release	Release to production
Sprint/Release Planning	Planning Game	Agile planning meetings
Product Owner	Customer	Business representative to project
Retrospective	Reflection	“Lessons learned”-style meeting
ScrumMaster	Coach	Agile project manager
Development Team	Team	Empowered Cross-Functional team
Daily Scrum	Daily Standup	Brief daily status meeting

# Lean Software Development

---

Lean was started by Toyota as manufacturing method that was applied to software development.

## Principles:

- Using visual management tools
- Identifying customer-defined value
- Building in learning and continuous improvement

# Lean Software Development

---





# Lean Software Development

---

## **Eliminate waste:**

- To maximize value, we must minimize waste. For software systems, waste can take the form of **partially done work, delays, handoffs, unnecessary features.**

## **Empower the team:**

- Rather than taking a micro-management approach, we should respect team member's superior knowledge of the technical steps required on the project and let them

## **Deliver fast:**

- Quickly delivering valuable software and iterating through designs.

## **Optimize the whole:**

- We aim to see the system as more than the sum of its parts.

# Lean Software Development

---

## **Build quality in:**

- Build quality into the product and continually assure quality throughout the development process

## **Defer decisions:**

- Balance early planning with making decisions and committing to things as late as possible.

## **Amplify learning:**

- This concept involves facilitating communication early and often, getting feedback as soon as possible, and building on what we learn.

# Seven Wastes of Lean

---





1. Partially done work
2. Extra Processes
3. Extra features
4. Task switching
5. Waiting
6. Motion
7. Defects

# Kanban Development

---

Kanban development is derived from the lean production system used at Toyota.

"**Kanban**" is a Japanese word meaning "**signboard**."

Items	In Progress	Testing	Done
			
	6 cards	4 cards	

# Kanban five core principles:

---

## **Visualize the workflow:**

- Software projects, by definition, manipulate knowledge, which is intangible and invisible.

## **Limit WIP:**

- Keeping the amount of work in progress low increases the visibility of issues and bottlenecks

## **Manage flow:**

- By tracking the flow of work through a system, issues can be identified and changes can be measured for effectiveness

## **Make process policies explicit:**

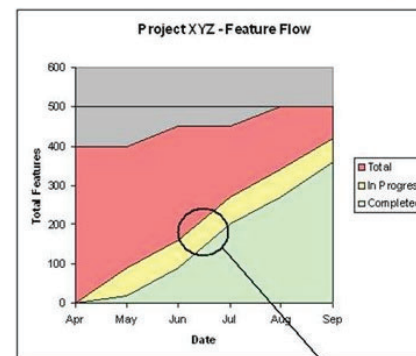
- It is important to clearly explain how things work so the team can have open discussions about improvements

## **Improve collaboration:**

- Through scientific measurement and experimentation, the team should collectively own and improve the processes it uses.

# Kanban Limit Work in Progress

## Little's Law

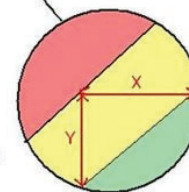


### Little's Law:

Cycle times are proportional to queue lengths.

(We can predict completion times based on queue size)

Y = Queue Length (units)  
X = Queue Duration (time)



[https://herdingcats.typepad.com/my\\_weblog/2014/08/the-use-and-misuse-of-littles-law.html](https://herdingcats.typepad.com/my_weblog/2014/08/the-use-and-misuse-of-littles-law.html)

# Other agile methods

---

## Feature-Driven Development

- Team will first develop an overall model for the product then build a list, and plan the work.

## Dynamic Systems Development

- One of the first agile methods and follows eight principles.

## Crystal

- It's a customized methodologies that are coded by color names.

# Leading Effectively

---

Tap into people's intrinsic motivations

- Discover why team members want to do something and what motivates and then align that to the project goals

Management vs Leadership

- Management → Mechanical Focus
- Leadership → Humanistic Focus (on people and purpose)

Management Focus	Leadership Focus
Task/things	People
Control	Empowerment
Command	Communication



# Leading Effectively

---

## Servant Leadership

- Leader provides what the team needs
  1. Shield team from interruptions
  2. Remove impediments to progress
  3. (Re)Communicate project vision
  4. Carry food and water

# Twelve Principles for Leading Agile Projects

---

1. Learn the team members needs
2. Learn the project requirements
3. Act for the simultaneous welfare of the team and the project
4. Create an environment of functional accountability
5. Have a vision of the completed project
6. Use the project vision to drive your own behavior

©2002 Jeffery Pinto, Project Leadership from Theory to Practice

# Twelve Principles for Leading Agile Projects

---

7. Serve as the central figure in successful project team development
8. Recognize team conflict as a positive step
9. Manage with an eye toward ethics
10. Remember that ethics is not an afterthought, but an integral part of our thinking
11. Take time to reflect on the project
12. Develop the trick of thinking backwards

# Leadership Tools and Techniques

---

Using these tools still need soft-skills approach

Modeling Desired Behavior

- Honesty
- Forward-Looking
- Competent
- Inspiring

Communicating project vision

Enabling others to act

- Switch from exclusive tools to inclusive tools

Being willing to change the status quo

# Leadership Task

---

Practice Transparency through Visualization

Create a safe environment for experimentation

Experiment with new techniques and processes

Share knowledge through collaboration

Encourage emergent leadership via a safe environment

# Value-Driven Delivery

---

# Value-Driven Delivery

---

Projects undertaken to generate business value

- Produce Benefit
- Improve Service
- Market Demand
- Safety Compliance
- Regulatory Compliance

# Early Value Delivery

---

Agile promote early and often delivery

Aim to deliver highest value early in project

- Deliver as many high-value components as soon as possible
  - Reduces risk
- Stakeholder satisfaction → Project success
  - Shows understanding of stakeholders' needs
  - Stakeholders are engaged
  - Builds confidence of stakeholders in team



# Reduce Waste

---

Minimize Waste, E.g:

- Partially done work
- Extra processes
- Extra features
- Waiting
- Defects

# Assessing Value - Financial Metrics

---

## Return on investment (ROI)

- The ratio of the benefits received from an investment to the money invested. Usually a percentage

## Internal rate of return (IRR)

- Interest rate you will need to get in today's money to receive a certain amount of money in the future

## Present Value/Net Present value (NPV)

- Value of future money in today's terms

# Assessing Value - Financial Metrics

---

## Earned Value Management

- Formulas that monitor the value of the project as its progressing.

# Accounting on Agile Projects

---

Refers to how the different economic models of agile works

Agile accounting is different than traditional accounting

Agile looks to deliver value as quickly as possible

Uses minimal viable product (MVP)

This leads to more opportunity for incremental funding

# Key Performance Indicators (KPI's)

---

Uses as a way to measure the project progress

- Rate of progress: How much points has been completed
- Remaining work: How much work is yet to be done from the backlog
- Likely completion date
- Likely Cost remaining

# Regulatory Compliance

---

Mandated requirements usually by government agencies

Must be implement into the project work as regular development work

Doing it after the project work is done

# Risk Management

---

Risk is closely related to value

Considered as anti-value

Usually has the potential to remove, erode or reduce value with threats

# Managing Risks Process

---





# Tools to Manage Risk

---

Risk-adjusted backlog

Risk burndown chart

# How Customers Conduct Value Prioritization

---

Valued based prioritization is the one of core practices in agile planning

Features are prioritized on the basis of business value, risk and dependencies

Some of prioritization techniques used:

- Simple Scheme
- MoSCoW prioritization
- Monopoly Money
- 100-point method
- Dot Voting or Multi-voting
- Kano Analysis
- Requirements Prioritization Model

# Prioritization Techniques

---

## Simple Scheme

- Priority 1, Priority 2, Priority 3, etc.
- Could be problematic as many items might become the first priority.

## MoSCoW prioritization

- Must have
- Should have
- Could have
- Would like to have, but not this time

# Prioritization Techniques

---

## Dot Voting or Multi-voting

- Each person gets a certain number of dots to distribute to the requirements

## Monopoly Money

- Give everyone equal monopoly money
- They then distribute the funds to what they value the most

## 100-point method

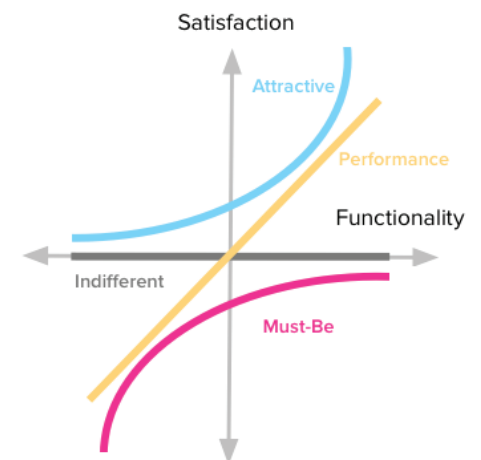
- Each person is given 100 points
- They then use that to distribute to individual requirements

# Prioritization Techniques

---

## Kano Analysis

- Helps to understand the customers satisfaction
  - Delighters/Exciters
  - Satisfiers
  - Dissatisfiers
  - Indifferent



<https://foldingburritos.com/kano-model/>

# Prioritization / Ranking is Relative

---

Doesn't matter what techniques the customers uses priority, the end results should be a list of prioritized features.

# Delivering Value Incrementally

---

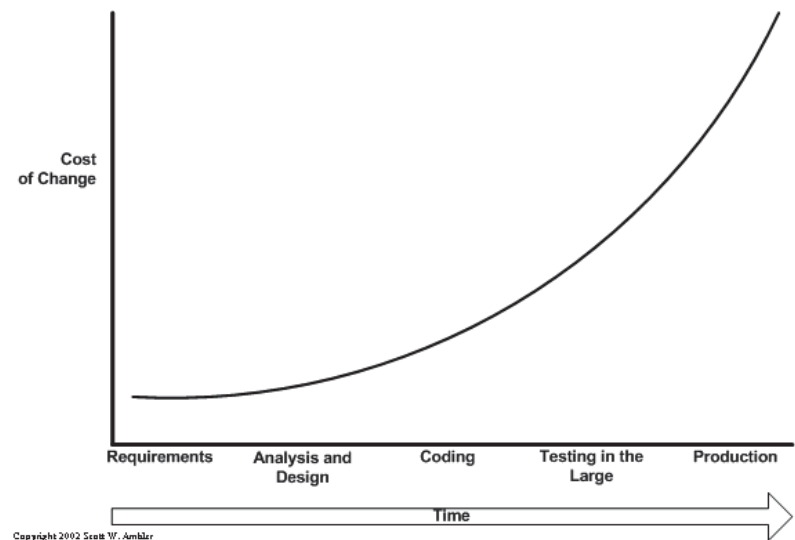
Incremental delivery is about deploying working parts of a product over the life of the project

In software development, its first delivered to a testing environment then to production

This will reduce the amount of rework by discovering issues early and fixing them

# Delivering Value Incrementally

---



<http://www.agilemodeling.com/essays/costOfChange.htm>



# Minimal Viable Product (MVP)

---

Refers to a set of functionality that is complete to be useful, but small enough not to be an entire project

Usually a module in a software

# Tools for Agile Projects

---

Low-tech, high-touch over computer models

When using computer models problems could arise such as:

- Data accuracy perception increases
- No stakeholder interaction. Only a few people would update them

# Low-Tech, High-Touch Tools

---

Use card, charts, whiteboards, and walls

Promotes communication and collaboration

Skip using a computer Gantt chart to a Kanban board

# Kanban/Task Board

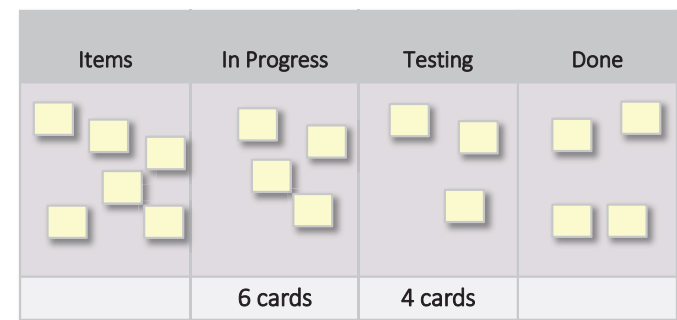
---

An "information radiator" - ensures efficient diffusion of information

Can be drawn on a whiteboard or even a section of wall

Makes iteration backlog visible

Serves as a focal point for the daily meeting



# Limit WIP (Work in Progress)

---

Includes work that has been started but not completed yet

Represents money spent with no return

Hides process bottlenecks that slow the processes

Represents risk in form of potential risk

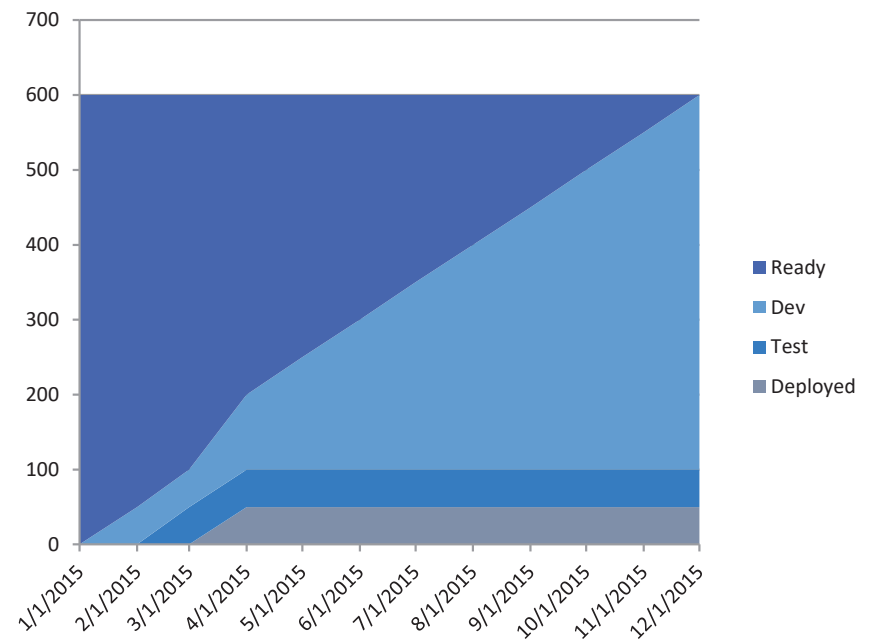
Agile processes aim to Limit and optimize WIP

Optimal WIP makes processes efficient

# Cumulative Flow Diagrams (CFD's)

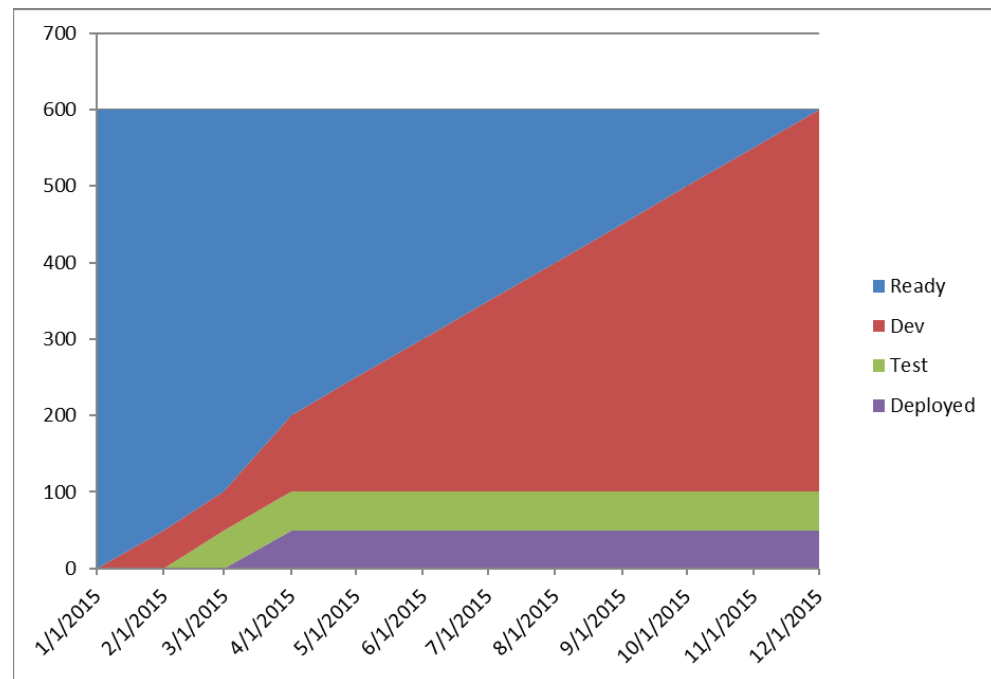
---

Stack graphs that show how work is progressing



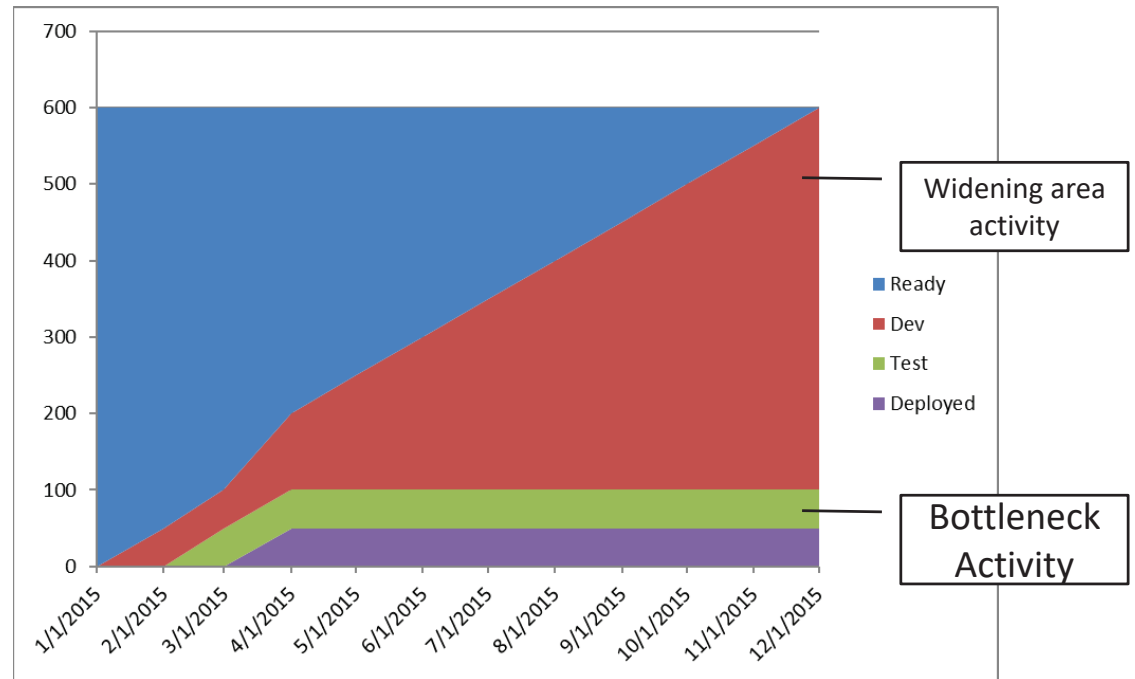
# Cumulative Flow Diagrams (CFD's)

## Bottlenecks and Theory of Constraints



# Cumulative Flow Diagrams (CFD's)

Bottlenecks and Theory of Constraints





# Agile Contracting

---

Agile's flexibility creates difficulty in outlining contract acceptance criteria

- Agile attempts to fix resources and time (cost) and vary functionality

“Customer collaboration over contract negotiation”

- Close cooperation
- Active participation
- Timely and often feedback

Money for nothing and change for free

# Agile Contracting

---

## Graduated Fixed Price Contract

- Buyer / Seller share in risks and rewards
- Different hourly rates based on:
  - Finish early, Finish on time, Finish late

## Fixed Price Work Packages

- Mitigate risks of under/over estimating

# Verifying and Validating Value

---

## “Gulf of Evaluation”

- What one person describes is often different from how another interprets

# Frequent Verification and Validation

Resolve problems as soon as possible

Don't let little problems grow over time



# Stakeholder Engagement

---

DOMAIN 3



# Stakeholder Stewardship

---

Looking after everyone involved on the project

Ensuring everyone has everything they need to complete the project successfully

Starts with identifying the stakeholders

# Educating People about Agile

---

Teach all the stakeholders about the benefits of agile

Concerns about agile can Include:

- Senior management and sponsor: They are worried about the risk of failing
- Managers: fear the loss of control
- Project team: resist agile methods
- Users: will not get all features

# Engaging Stakeholders

---

Short iterations and release keeps them engage

Keeping then engage can lead to stakeholders being more involved and getting more change request

This helps us to identify risk and issues early

If some stakeholders are causing problems, the agile PM will need to use their interpersonal skills to resolve issues

Need to have a process for escalating stakeholders issues

Why such a big focus on stakeholders?

- Projects and done by people for people



# Methods of Stakeholder Engagement

---

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

# Set a Shared Vision

---

Important to ensure customers and agile project team has the same vision

Methods include:

- Agile Charter
- Definition of “Done”
- Agile Modeling
  - Use case diagram
  - Data models
  - Screen design
- Wireframes
- Personas

# Agile Chartering

---

High-level (uses the W5H)

Agreement

Authority to proceed

Focuses on *how* project will be conducted

- Allows for flexibility and ability to deal with change

Project specific processes outlined

May use project Tweet – Describes project goal in 140 Characters or less.

# Definition of “Done”

---

Creating a shared vision of what done looks like

Should be done for:

- User stories
- Releases
- Final project deliverables

# Agile Modeling

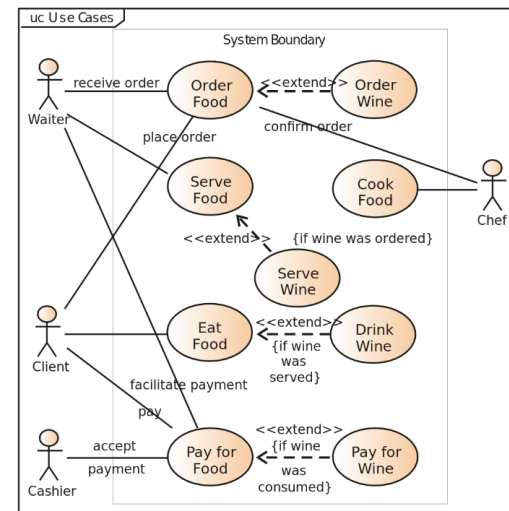
---

Different modeling techniques that are used to help establish the shared vision

Should be lightweight or “barely sufficient”

# Agile Modeling

- Use case diagrams
  - Visually shows how users would use an application

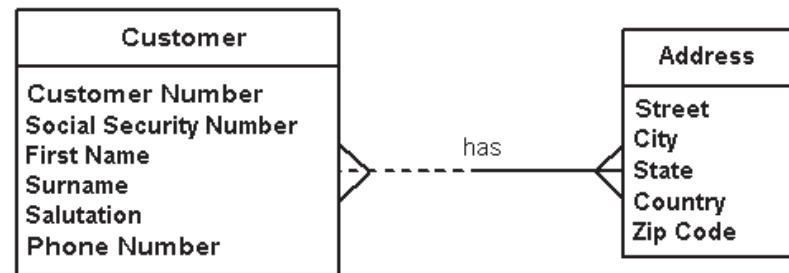


[https://en.wikipedia.org/wiki/Use\\_case\\_diagram](https://en.wikipedia.org/wiki/Use_case_diagram)

# Agile Modeling

---

- Data models
  - How the data are structured in tables and their relationships



Copyright 2002-2006 Scott W. Ambler

<http://www.agiledata.org/essays/dataModeling101.html>

# Agile Modeling

- Screen designs
- Simple screen shots

The image shows two hand-drawn wireframes for software screens. The left screen is titled 'Student Information' and contains fields for 'Student Number: 787-507-234', 'First Name: Scott', 'Middle: C', 'Surname: P', 'Salutation: Mr.', and 'Date first edit: June 11 2003'. Below these is a table for 'Semesters' with columns for 'Semester', 'Term', 'Date', and 'Status'. The table contains three rows of data. At the bottom are buttons for 'Add', 'Delete', 'Transcript', and 'Close'. The right screen is titled 'All a server' and contains fields for 'Server Number: CSC-8' and 'Name: C'. Below these is a table for 'Results' with columns for 'Server', 'Term', 'Grade', and 'Status'. The table contains three rows of data. Below the table is a text area for 'Course description:' followed by a paragraph of text. At the bottom right is a button for 'Close'.

Semester	Term	Date	Status
CSC 100 Intro to CS	Fall 2003	11	Passed
CSC 200 Intro to AI	Fall 2003	11	Passed
CSC 300 Advanced AI	Spring 2004	11	Enrolled

Server	Term	Grade	Status
CSC 250 Agile Design	Fall 2003	4	Success
CSC 300 Agile Exp	Spring 2004	17	Success
CSC 300 Agile Design	Spring 2004	18	Success

<http://agilemodeling.com/artifacts/uiPrototype.htm>



# Wireframes

---

## Wireframes

- Quick mock-up of product
- “low-fidelity prototyping”
- Clarify what “done” looks like
- Validate approach prior to execution

# Personas

---

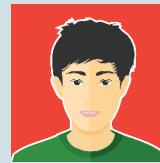
## Personas

- Quick guides or reminders of key stakeholders and interests
  - Provide description of users
  - Be grounded in reality
  - Be goal-oriented, specific, and relevant
  - Be tangible and actionable
  - Generate focus
- Help team focus on valuable features to users

# Personas

---

## Name: Andrew Jones– Certified Accountant



### Description:

Andrew has been an Accountant for over 10 years and has worked at many large accounting firms.

He likes to be organized and get his work done on time.

### Value:

Andrew would like to ensure all company bills are paid on time while using online auto payments.

He would like to ensure customers are reminded automatically of outstanding balances.

He is looking to print the receivables and payable reports on a weekly basis to check on bills and invoices.

# Communicating with Stakeholders

---

Face to face communication

Two-way communication

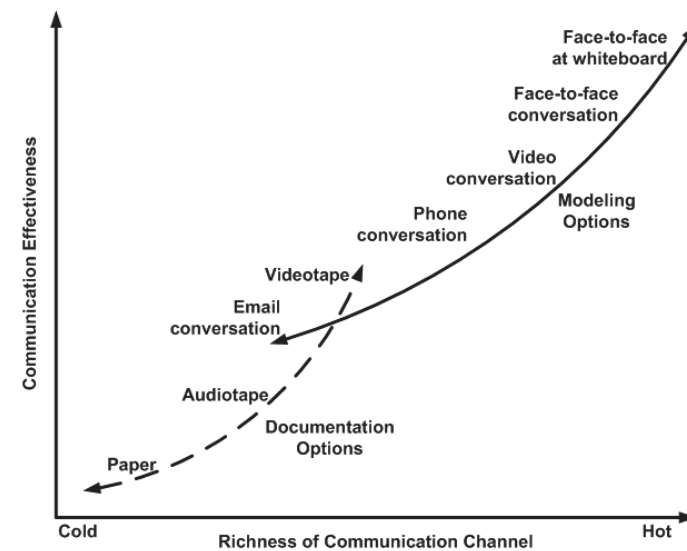
Knowledge sharing

Information Radiators

Social Media

# Face-to-face Communication

Face to face communication



Copyright 2002-2005 Scott W. Ambler  
Original Diagram Copyright 2002 Alistair Cockburn

<http://www.agilemodeling.com/essays/communication.htm>

# Communicating with Stakeholders

---

## Two-way communication

- Just don't ask for confirmation or concerns, but actually listen to what they have to say

## Knowledge sharing

- Agile teams work closely with each other such as with pair-programming.
- Using Kanban boards or wireframes are ways to share information
- Use of low-tech tools like a whiteboard will allow all to see the work and understand it
- We must encourage it

# Communicating with Stakeholders

---

## Information Radiators

- Things that are highly visible
- Used to display information
- Usually includes chats, graphs and boards

## Social Media

- Use to communicate
- Can include twitter or Instagram

# Green Zone/Red Zone

---

## Red Zone:

- Blames others for everything
- Responds defensively
- Feels threatened
- Triggers defensiveness
- Doesn't let go or forgive
- Uses shame and blame
- Focus on short-term advantage
- Doesn't seek or value feedback
- Sees conflict as a battle and only seeks to win
- Communicates high level of disapproval
- Sees others as the problem or enemy
- Does not listen effectively



# Green Zone/Red Zone

---

## Green Zone:

- Take responsibility
- Seeks to respond nondefensively
- Is not easily threatened psychologically
- Attempts to build success
- Uses persuasion rather than force
- Thinks both short and long term
- Welcomes feedback
- Sees conflict as a natural part of life
- Seeks excellence rather than victory
- Listens well

# Using Workshops

---

Meeting when work gets done

Retrospectives are a type of workshops

Ways to make them more effective:

- Diverse groups has a larger perspective
- Use methods such as round-robin to ensure no one dominates
- Try to get everyone to participate in the first few minutes

User story workshops are where we write the user stories and keep stakeholders engage

# Brainstorming

---

## Brainstorming

- Quite Writing
  - Give people about 5 minutes to write down their ideas
- Round-Robin
  - Pass a token around to ensure everyone will speak
- Free-for-all
  - People shout out their thoughts. May only work in a supportive environment

# Collaboration Games

---

Remember the future

Prune the product tree

Speedboat(Sailboat)

# Remember the future

---

Ask stakeholders to imagine that an upcoming release was successfully and to look back

Gets a better understanding of how a stakeholder would define success

Outlines how we can accomplish that success for them

# Prune the Product Tree

---

Draw a tree and ask stakeholders to add their features to it

Use stick notes to have them place new features on the tree

Group the features on the trunk

Features that are depending on other features would be higher up the tree

Lets everyone understand the priorities of development

# Speedboat(Sailboat)

---

Draw a waterline and a boat moving

Explain the boat is moving toward the goals of the project

Ask them to use sticky notes to show what can make the boat move (wind) and what can stop it (anchors)

Allows stakeholders to identify threats and opportunities

# Using Critical Soft Skills

---

Emotional intelligence

Negotiation

Active Listening

Facilitation

Conflict Resolution

Participatory Decision Models



# Emotional intelligence

---

Our skill to identify, assess, and influence the emotions of ourselves and others around us

We need to recognize our own feeling

Then we can learn how to response to others and how they feel

Understand how we take care of ourselves will impact other around us

As an agile PM we have to know when team members are stuck, angry, or frustrated

# Negotiation

---

This happens all throughout the project

Good negotiation will allow everyone to investigate the options and trade-offs

Most effective when interactions between people are positive and there are room for give and take

# Active Listening

---

Level 1: Internal – how is it going to affect me

Level 2: Focused – put ourselves in the mind of the speaker

Level 3: Global – builds on level with body language

# Facilitation

---

Run effective meeting and workshops.

Have the following:

- Goals
- Rules
- Timing
- Assisting

# Conflict Resolution

---

All projects will have conflicts

While some level of conflicts are good, we need to ensure they don't become a "world war" where people are trying to destroy each other

Levels of conflict(1-5):

- Level 1: Problem to solve – sharing info
- Level 2: Disagreement – Personal Protection
- Level 3: Contest – Must win
- Level 4: Crusade – Protecting one's group
- Level 5: World War – Must destroy the other

# Participatory Decision Models

---

Engage stakeholder in decision making process

- Simple voting
  - Vote “for” or “against” it
- Thumps up/down/sideways
  - People hold their thumps in a way of if the support it or not. Sideway is if they cannot make up their mind
- Fist of five
  - People how up finger based on they support the idea
  - 1 finger: total support – 5 finger: Stop against it

# Team Performance

---

# People Over Processes

---

Projects are done by people, not tools

- Agile manifesto: “Individuals and Interactions over processes and tools”

Focus on the people side of the project

Projects are more about people management than tools management



# People Over Processes

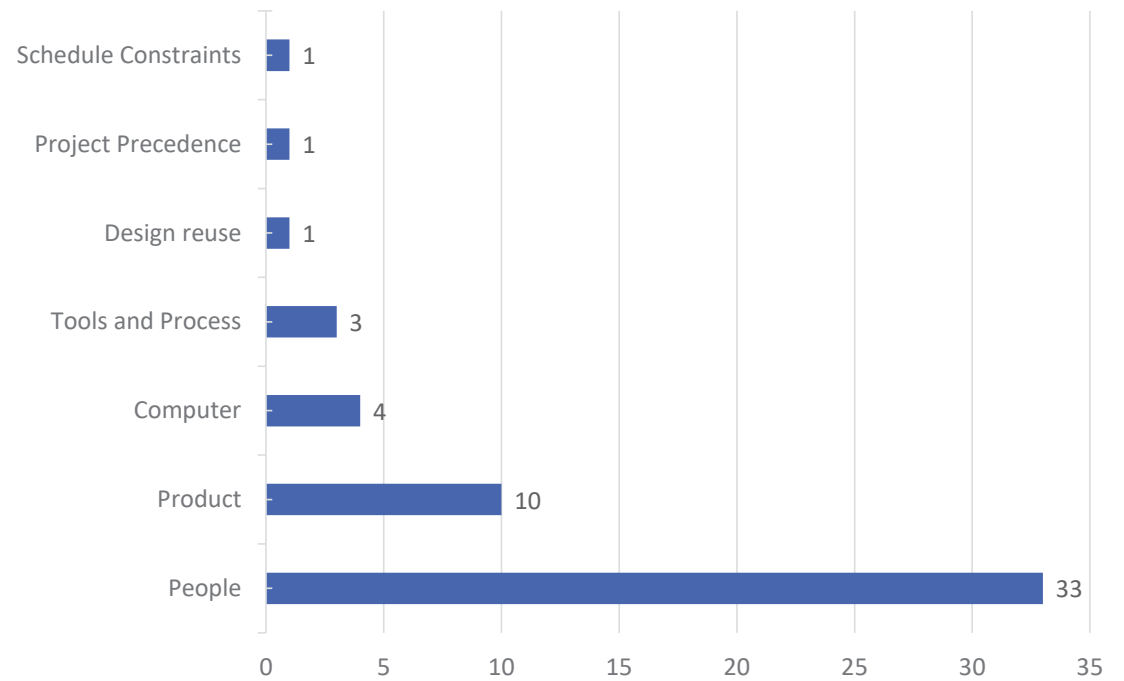
---

## COCOMO

- Constructive Cost Model
- To determine correlation between project input variables and final cost to use to estimate future projects
- People factors has a score of 33...11 times more significant than tools and processes

# COCOMO II

---



# Development/Delivery Team

---

Group that build and test the increments of the product

- Build product in increments
- Update information radiators
- Self organize and directing
- Share progress by doing daily stand-up meetings
- Write acceptance tests
- Demo the completed product increments
- Holds retrospectives at the end of sprints
- Does release and sprint planning and estimations

# Product Owner/Customer

---

Prioritizing the product features

Manage the product backlog ensuring its accurate and up to date

Ensures the team has a shared understanding of the backlog items

Defines the acceptance criteria

Provides the due dates for the releases

Attends planning meeting, reviews, and the retrospective.

# Agile Project Manager (ScrumMaster/Coach)

---

Act as a servant leader

Help the team self-organize and direct themselves

Be a facilitator

Ensure the team plan is visible and the progress is known to the stakeholders

Act as a mentor and coach

Work with the product owner to manage the product backlog

Facilitates meeting

Ensure issues are solved

# Building Teams

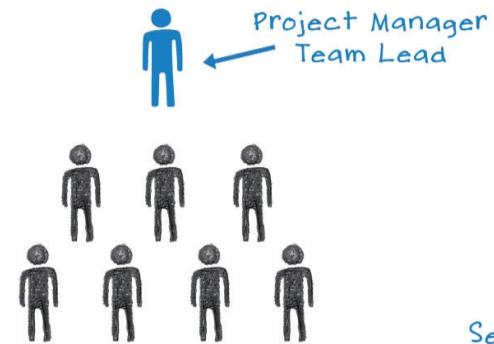
---

Self-Organizing

Self-Directing

Small teams with fewer than 12 members

## Traditional Teams



## Agile Teams



# Generalizing Specialists

---

Have members that can do different tasks

Members skilled in more than one area

Share work reduce bottleneck

# High-Performance Agile Teams

---

Have a shared vision

Realist goals

Fewer than 12 members

Have a sense of team identity

Provide strong leadership



# Experiments (Have a safe place)

---

Establish safe environment for disagreement

Allows team members to build strong commitment to decisions

Encourage people to experiments with new methods

Leads to more engagement

# Welcome Constructive Disagreement

---

Leads to better buy-in and decisions

Avoiding conflicts can lead to conflicts escalating

A safe place for disagreement leads to successful problem solving

# Models of team development

---

## Shu-Ha-Ri Model of Skill Mastery

- Shu- Obey,
- Ha – Moving away,
- Ri – finding individual paths

## Dreyfus Model of Adult Skill Acquisition

- Novice, Advanced Beginner, Competent, Proficient, Expert

# Tuckman's Five Stages of Team Development

---

1. **Forming:** team comes together and starts to get to know each other. There is not much conflict or communication.
2. **Storming:** team members start to have conflicts with each other. They start to learn of each other's ideas and may not agree with them. Most conflicts takes place in this stage.
3. **Norming:** the team members begin to agree with each other on the best methods to build the deliverables. Generally, everyone is coming to a consensus.
4. **Performing:** the team is performing well and is working without conflict.
5. **Adjourning:** In this stage, the project is completed and the team is reassigned.

# Adaptive Leadership

---

Concept of adapting how we lead team based on specific circumstances and how mature team is in formation

Forming → Directing  
Storming → Coaching  
Norming → Supporting  
Performing → Delegating  
Adjourning

# Training, Coaching, and Mentoring

---

## Training

- Teaching of skills or knowledge

## Coaching

- Process that helps a person develop and improve their skills

## Mentoring

- More of a professional relationship that can fix issues on an as-needed basis

Help team stay on track, overcome issues, and continually improve skills

Individual level

Whole-team level

# Team Spaces

---

Co-located Teams

Team Spaces

Osmotic Communication

Global and Cultural Diversity

Distributed teams

# Co-Located Teams

---

All team member work together in the same location

Allows for face-to-face time and interaction

Should be within 33 feet of each other

No physical barriers

Sometimes a virtual co-location



# Team Space

---

Lots of low-tech, high touch

- Whiteboards and task boards
- Sticky notes, flip charts
- Round table
- No barriers to face-to-face communication

Caves and Common

- Caves → space team members can retreat to individually
- Common → space team members can work as group

Osmotic Communication

- Information flows that occur as part of everyday conversations and questions
- 33 feet or 10 meters

Tacit Knowledge

- Information that is not written down; supported through collective group knowledge

# Global and Cultural Diversity

---

Time Zones

Cultures

Native Languages

Styles of communications

# Distributed Teams

---

At least one team member working off-site

Need to find ways to replicate co -location team benefits

## Agile Tools

- Low-Tech, High-Touch Tools
- Digital Tools for distribute teams
  - Video conferencing
  - Interactive whiteboards
  - IM / VoIP
  - Virtual card walls
  - Web cams
  - Digital cams

# Tracking Team Performance

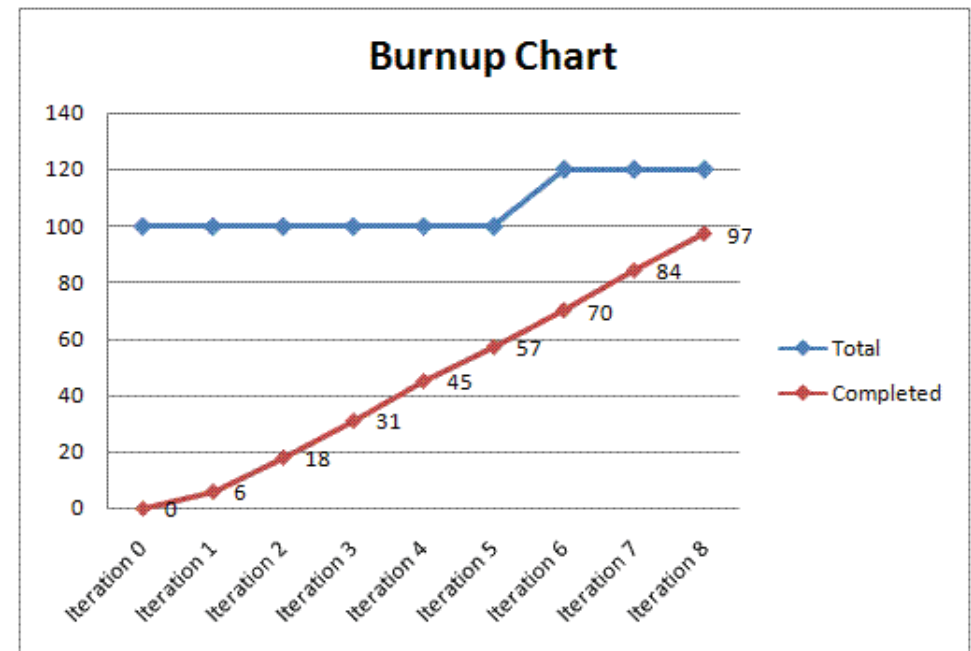
---

## Burn Charts

- Burnup
- Burndown

## Velocity Charts

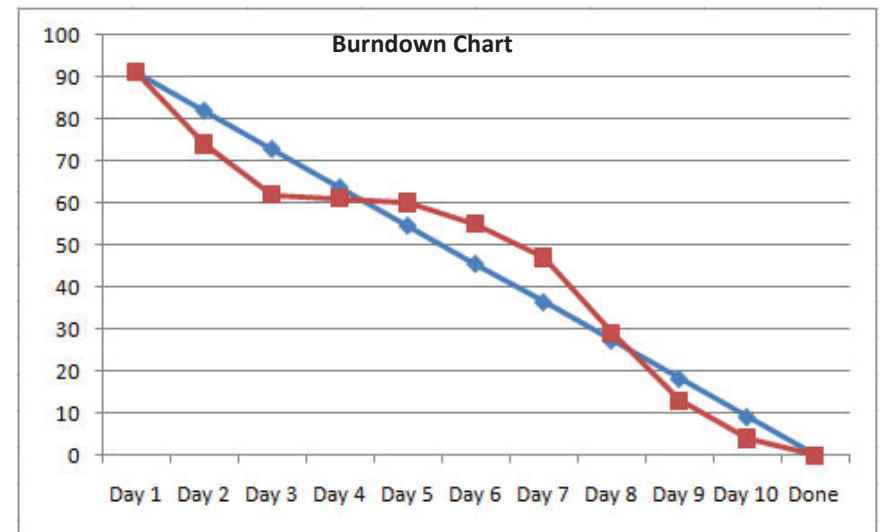
# Burnup Chart



Work that has been done

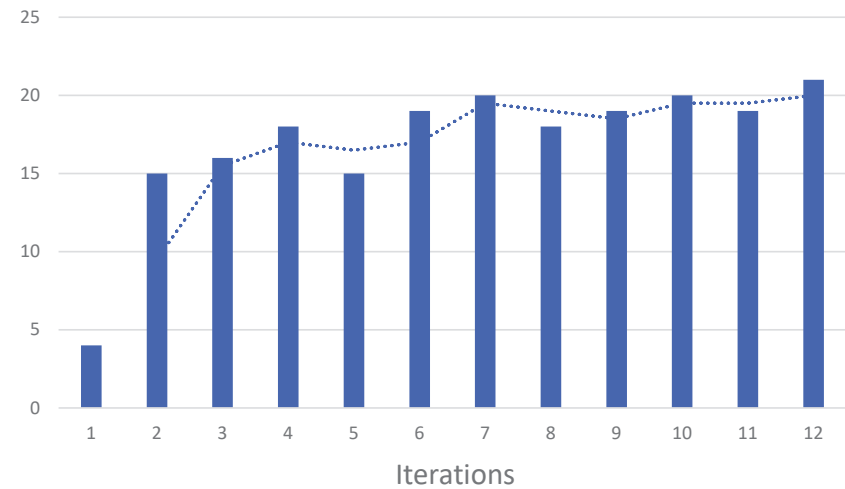
# Burndown Chart

---



Work that remains to be done

# Velocity Charts



Show how the team is performing

# Velocity Charts

---

If a team has complete 3 iterations with the average velocity of 18 points per iteration, how many iterations would it take to complete 250 points of work?

= $250/18$  = About 14 more iterations.



# Adaptive Planning

---

# Adaptive Planning

---

Planning is ongoing process

Multiple mechanisms to proactively update plan

Focus on value delivery and minimize nonvalue-adding work

Uncertainty drives need to replan

Frequently discover issues and experience high rates of change

# Agile Plans

---

Agile planning varies from traditional planning

1. Trial and demonstration uncover true requirements, which then require replanning
2. Agile planning is less of an upfront effort, and instead is done more throughout the project
3. Midcourse adjustments are the norm

# Principles of Agile Planning

---

1. Plan at multiple levels
2. Engage the team and the customer in planning
3. Manage expectations by frequently demonstrating progress
4. Tailor processes to the project's characteristics
5. Update the plan based on the project priorities
6. Ensure encompassing estimates that account for risk, distractions, and team availability
7. Use appropriate estimate ranges to reflect the level of uncertainty in the estimate
8. Base projections on completion rates
9. Factor in diversion and outside work

# Progressive Elaboration

---

Adding more detail as information emerges

Includes:

- Plans
- Estimates
- Designs
- Test scenarios

Rolling wave planning: Planning at multiple points in time as data becomes available

# Value-Base Analysis and Decomposition

---

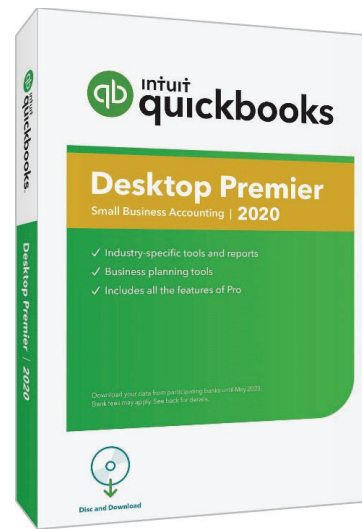
Assessing and prioritizing the business value of work items, and then plan accordingly.

Consider payback frequency and dependencies

## Value -Based Decomposition

- Breaks down requirements and prioritized them
- Design the product box

# Design the Product Box



# “Coarse-Grained” Requirements

---

Keep Requirements “coarse” then progressively refine them

Helps keeps the overall design balanced

Delays decision on implementation until the “last responsible moment”



# Timeboxing

---

Short, fixed-duration periods of time in which activities or work are undertaken

- If work is not completed within time period, move it to another timebox

Daily Stand-up – 15 minutes

Retrospectives – 2 hours

Sprints – 1-4 weeks

Beware of Parkinson's Law

- Work tends to expand to fill the time given

# Agile Estimation

---

*Knowledge* of agile estimation theory & ability to perform simple agile estimating techniques

Why do we estimate?

- Determining which pieces of work can be done within a release or iteration

How are estimates created?

- By progressing through the stages planning.

How should estimates be stated?

- Should be started in ranges

When do we estimate?

- Throughout the project. More detail in the later parts of the project

Who estimates?

- Team members will do their own estimates

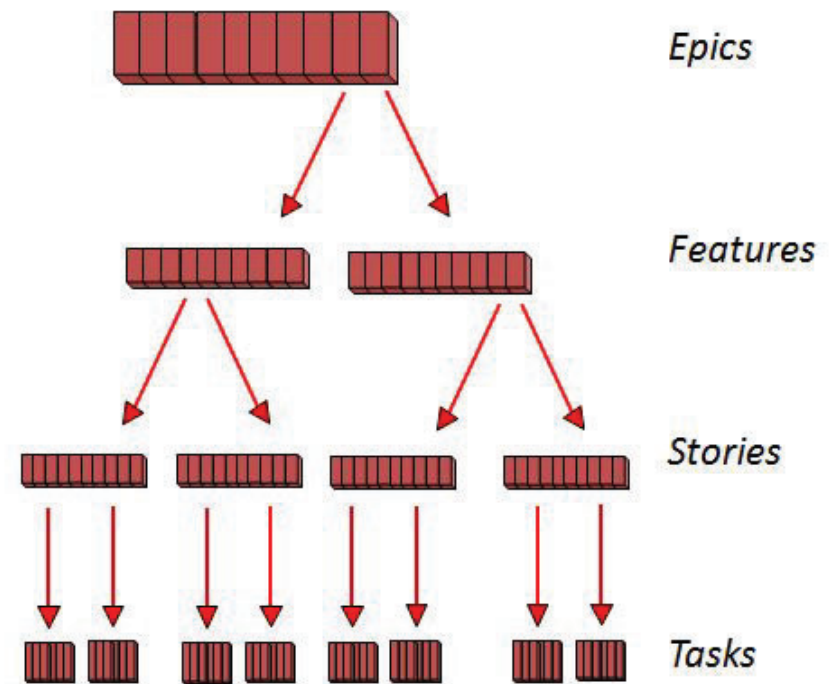
# Ideal Time

---

Refers to the time it would take to complete a given task assuming zero interruptions or unplanned problems

# Decomposing Requirements

---



# User Stories

---

## User Stories / Backlogs

- Business functionality within a feature that involves 1-3 days of work.
- Acts as agreement between customers and development team
- Every requirement is user story
- Every story, including technical stories, has value
- Common structure of a user story

As a <user type>  
I <want to/need, etc.> goal  
So that <**value**>

# User Story Example

---

“As an payroll clerk, I want to be able to view a report of all payroll taxes, so that I can pay them on time”

“As a sales person, I want to be able to see a current list of leads, so that I can call them back quickly”

“As student of this course, I want to be able to understand the requirements of the exam, so that I know if I qualify for it or not”

# Three C's of Stories

---

Have users write the stories on index cards

No details, it's used to help converse

3 Cs:

- Card
- Conversation
- Confirmation

# User Stories - INVEST

---

Effective user stories should be “INVEST”

**Independent**

- Should be independent so it can reprioritize

**Negotiable**

- Should allow for trade-off's based on cost and function

**Valuable**

- Should clearly state the value of it

**Estimatable**

- Should be able to estimate how long to complete

**Small**

- Stories should be between 4-40 hours of work

**Testable**

- Should be testable to ensure it will be accepted once completed

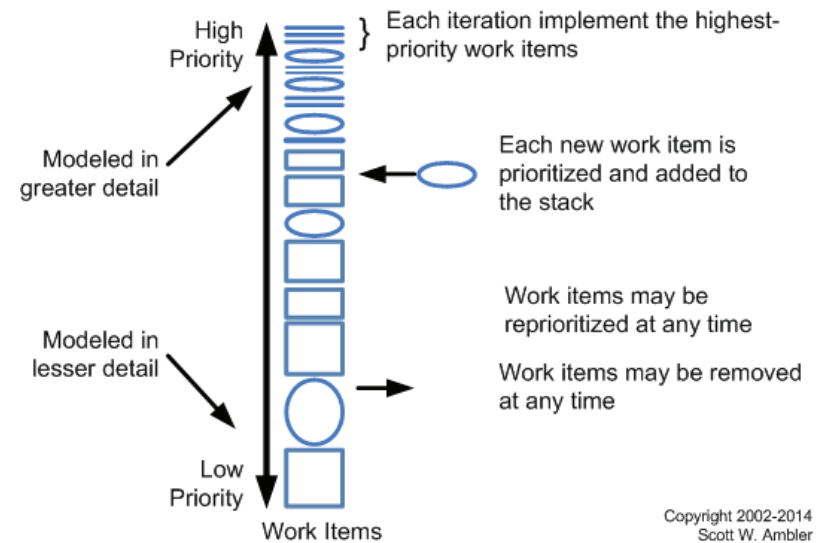


# User Story Backlog (Product Backlog)

Prioritize Requirements

Refining (Grooming) Backlog

- Keeping the backlog updated and accurately prioritized



Copyright 2002-2014  
Scott W. Ambler

# Relative Sizing and Story Points

---

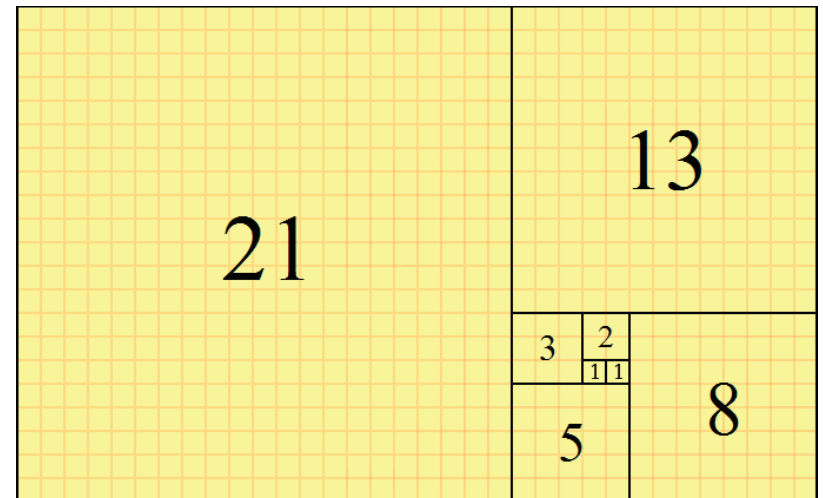
Absolute estimates are difficult for humans to make

Estimates should be relative

Assign points to each story using a relative numbers

# Fibonacci Sequence

---



[https://en.wikipedia.org/wiki/Fibonacci\\_number](https://en.wikipedia.org/wiki/Fibonacci_number)

Fibonacci Sequence: 1, 2, 3, 5, 8, 13, 21

# Guidelines for Using Story Points

---

Team should own the definition of their story points

Story point estimates should be all-inclusive

Point sizes should be relative

Complexity, work effort, and risk should all be included in the estimate

# Affinity Estimating and T-Shirt Sizing

---

## Affinity Estimating

- Group estimates into categories or collections

## T-Shirt Sizing

- Place stories in sizes of t-shirts

# Wideband Delphi

---

## Wideband Delphi

- Group-based estimation approach
- Panel of experts, anonymously

## It's used to prevent:

- Bandwagon effect
- HIPPO decision making (Highest-Paid Person's Opinion)
- Groupthink

# Planning Poker

---

Advantages of Wideband Delphi

Fast, collaborative process

Uses cards with Fibonacci sequence

# Story Maps

---

High-level planning tool

Stakeholders map out what the project priorities early in the planning

Serves as the “product roadmap”

Shows when features will be delivered and what is included in each release



# Product Roadmap

---

Shows when features will be delivered and what is included in each release

Can convert the story map into a product roadmap

# Types of Iterations

---

## Iteration 0

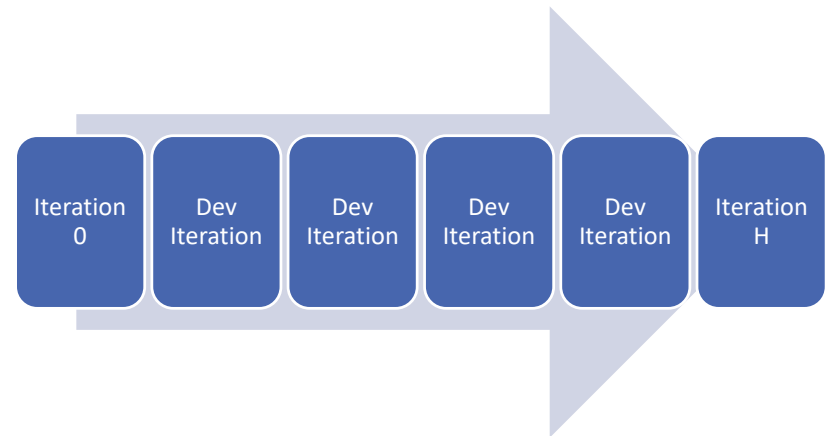
- Set the stage for development efforts
- Doesn't build anything

## Development Iteration

- Build the product increment

## Iteration H (hardening sprint or release)

- Done at the end to clean up codes or producing documentation



# Spikes

---

- Architectural spike
  - Period of time dedicated to proof of concept
- Risk-Based Spike
  - Team investigate to reduce or eliminate risk

# Iteration Planning

---

Meeting run by the delivery team.

Discuss the user stories in the backlog

Select the user stories for the iteration

Define the acceptance criteria

Break down the user stories into task

Estimate the task

# Release Planning

---

Meeting with all stakeholders to determine which stories will be done in which iterations for the upcoming release.

Selecting the user stories for the release

- Using Velocity – points per iteration

Slicing the stories

- Breaking down stories that are too large to be completed in 1 iteration

# Problem Detection and Resolution

---

# Understand How Problems Happen

---

All projects will have problems

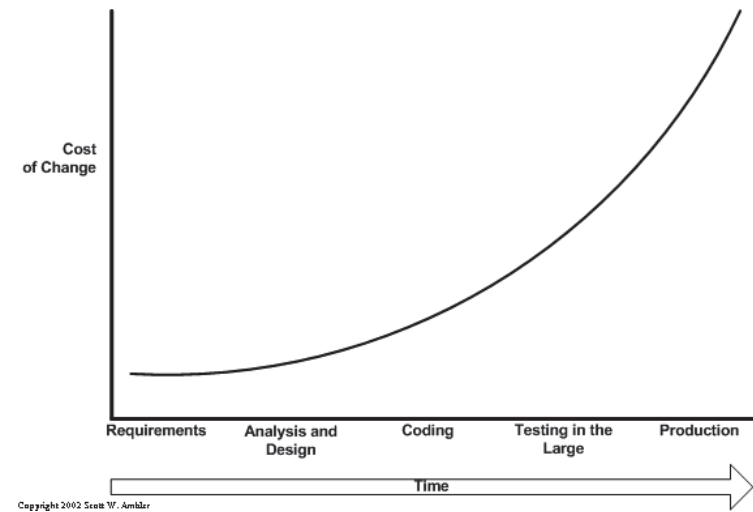
As a project is progressing the agile PM should expect issues to happen

Over time issues can delay or change a project objectives

# Cost of Change

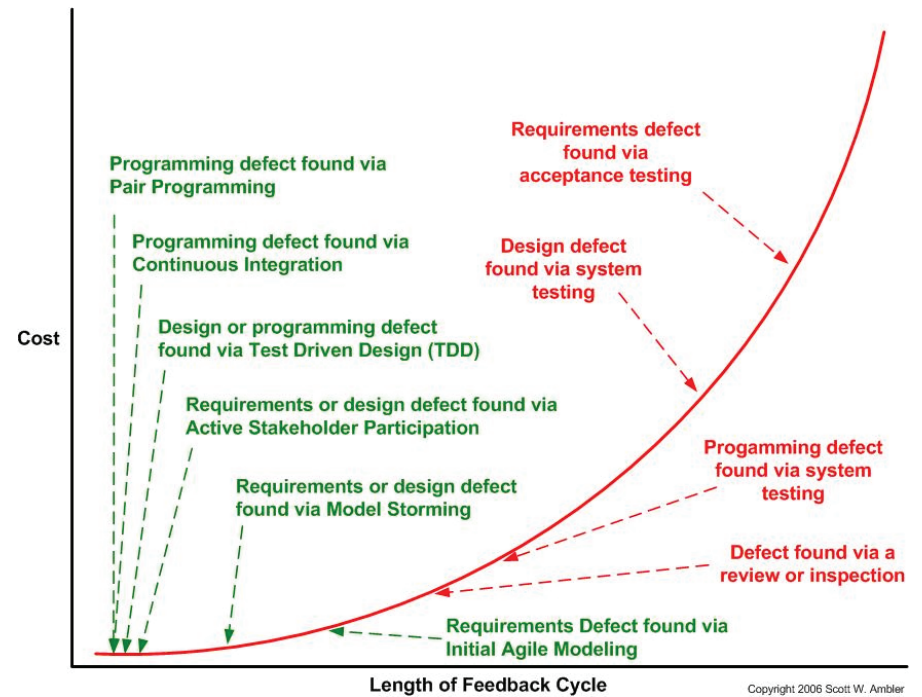
---

Over time the cost of change will increase





# Cost of Change



# Technical Debt

---

Backlog of work caused by not doing regular cleanup

If not done will lead the increase cost of development and make it harder to implement changes

Refactoring is the solution

# Failure Modes

---

Why do people Fail:

1. Making mistakes
2. Preferring to fail conservatively
3. Inventing rather than researching
4. Being creatures of habit
5. Being inconsistent

# Success Modes

---

Why do we succeed:

1. Being good at looking around
2. Being able to learn
3. Being malleable
4. Taking pride in work

# Success Strategies

---

Balance discipline with tolerance

Start with something concrete and tangible

Copy and alter

Watch and listen

Support both concentration and communication

Match work assignment with the person

Retain the best talent

Use rewards that preserve joy and combine rewards

Get feedback

# Lead Time and Cycle Time

---

## Lead/Cycle time

- Lead time: how long something takes to go through the entire process
- Cycle time: how long something takes to go through a part of the process. Part of lead time.

## Cycle Time

- Measure of how long it takes to get things done
- Closely related to work in progress (WIP)
  - Excessive WIP is associated with several problems
    - Represents money invested with no return on investment yet
    - Hides bottlenecks in processes & masks efficiency issues
    - Represents risk in form of potential rework

# Cycle Time

---

Long cycle times lead to increased amounts of WIP

$$\text{Cycle Time} = \frac{WIP}{Throughput}$$

Throughput: Amount of work that can done in a time period

# Cycle Time Question

---

What would be the cycle time of feature A, if it requires 60 points of work and the team can complete 5 points per day?

= $60/5$  points per day = 12 days.



# Defects

---

Longer defects are left, more expensive to fix

More work may have been built on top of bad design, resulting in more work to be undone

Later in development cycle, more stakeholders impacted by defect and more expensive to fix

## Escaped Defects

- Defects that make it to the customer

# Variance and Trend Analysis

---

Variance measure of how far apart things are (or vary)

Trend Analysis measure that provides insight into future issues

- Lagging Metrics provides information on something that has already happened
- Leading Metrics provides information on is or is about to occur

# Control Limits

---

Help diagnose issues before issue occurs

Provide guidelines to operate within

# Risk

---

## Risk Adjusted Backlog

- Adjusting the backlog for risk
- Done after risk response

Expected Monetary Value = Impact(\$) x Probability(%)

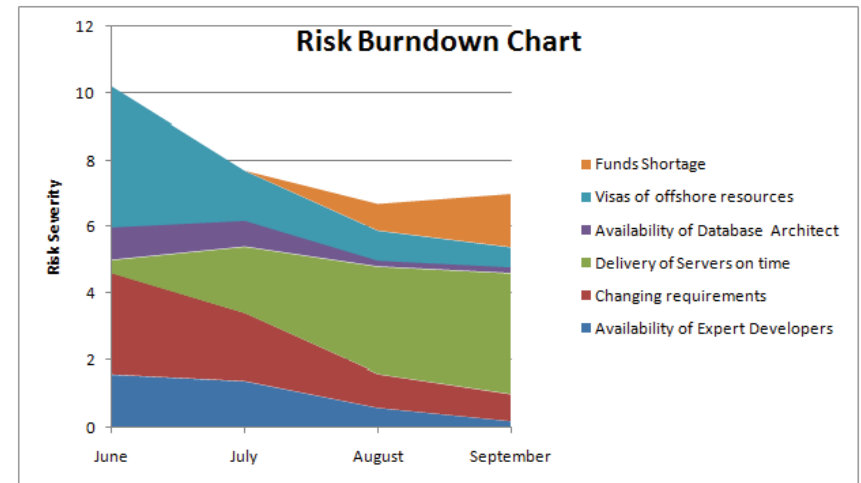
## Risk Severity

- Risk Probability x Risk Impact
- Uses a scale of numbers (E.g 1-5)

# Risk

---

## Risk Burndown Graphs



# Solving Problems

---

Problem Solving as continuous improvement

Engage the team

Some problems can't be solved

# Why Engaging the Team?

---

Team usually produces the best practical solutions

## Benefits

- Get consensus from all members
- Gets a broad knowledge base
- Solutions are practical
- When ask people work hard to produce good ideas
- Asking someone for help shows confidence

## Usage and Cautions

- Solve real problems
- Poor team cohesion
- Team and project changes
- Follow-Through

# Continuous Improvement

---



# Kaizen

---

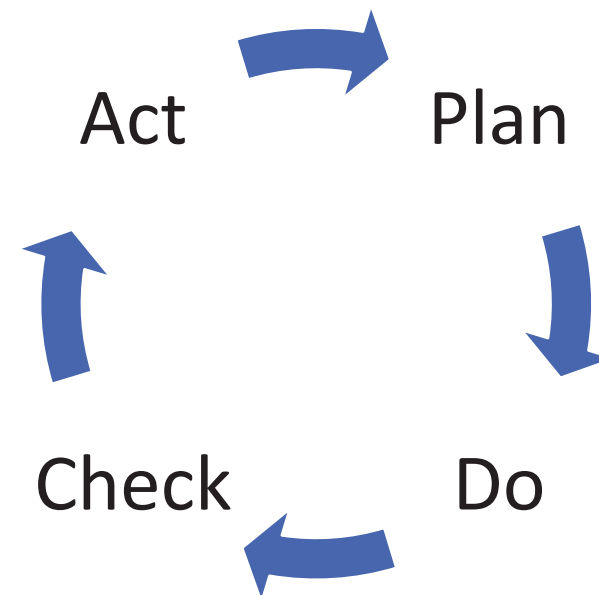
Kaizen is a process for continuous improvement  
name after the Japanese word

Focus on the team to implement small incremental  
improvement

Usually follows the Plan-Do-Check-Act (PDCA) cycle  
by Edwards Deming

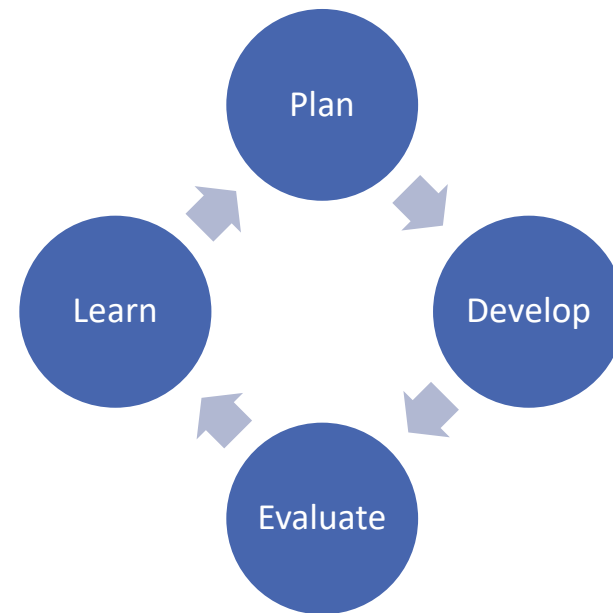
# PDCA

---



# Agile Cycle

---



# Process Analysis

---

Review and diagnose issues

Look for tailoring possibilities

# Process Tailoring

---

Amend methodology to better fit project environment

Change things for good reason, not just for sake of change

Develop a hybrid

# Value Stream Map

---

Optimize the flow of information or materials to complete a process

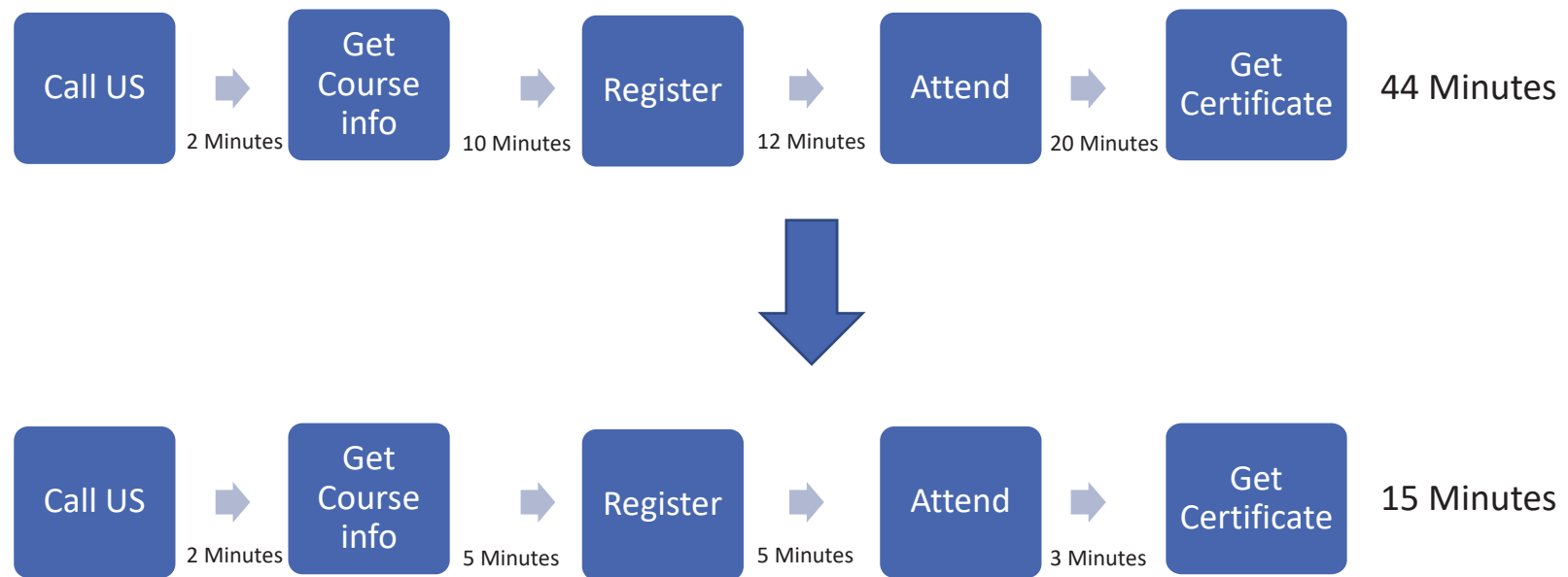
Reduce waste (waiting times) or unnecessary work

Steps to creating:

- Identify the product or service
- Create a value stream map
- Review to find waste
- Create a new map with the desire improvement
- Develop a roadmap to implement the fixes
- Plan to revisit it again

# Value Stream Map Example

---



# Pre-Mortems

---

Team meeting that looks at possible things that can cause failure during a project before they take place

Steps include:

- Think what the failures might be
- Create a list of reasons that can cause the failures
- Review the project plan to determine what can be done to reduce or remove the reasons for failure



# Retrospectives

---

Special meeting that takes place after each iteration

Inspect and improve methods and team work

Offers immediate value

Should have a 2 hour time limit

# Retrospectives Stages

---

About 2 Hours for a typical retrospective

1. Set Stage – 6 Minutes
2. Gather Data – 40 Minutes
3. Generate Insights – 25 Minutes
4. Decide What to Do – 20 Minutes
5. Close Retrospective – 20 Minutes

# 1. Set the Stage

---

Start of the retrospective

Help people to get focus

Encourage participation to ensure everyone start talking early

Outlining the approach and topics for discussion

Get people in mood for contributing information

Activities include:

- Check-In
- Focus On/Focus Off
- ESVP
  - People identify if they are an explorer, shopper, vacationer, or Prisoner

## 2. Gather Data

---

Create a picture of what happened during the sprint

Start to collect information to be used for improvement

Activities:

- Timeline
- Triple Nickels: break the team into 5 groups to spend 5 minutes collecting 5 ideas, 5 time
- Mad, Sad, Glad: what where the team emotion as the sprint was taking place

# 3. Generate Insights

---

Analyze the data

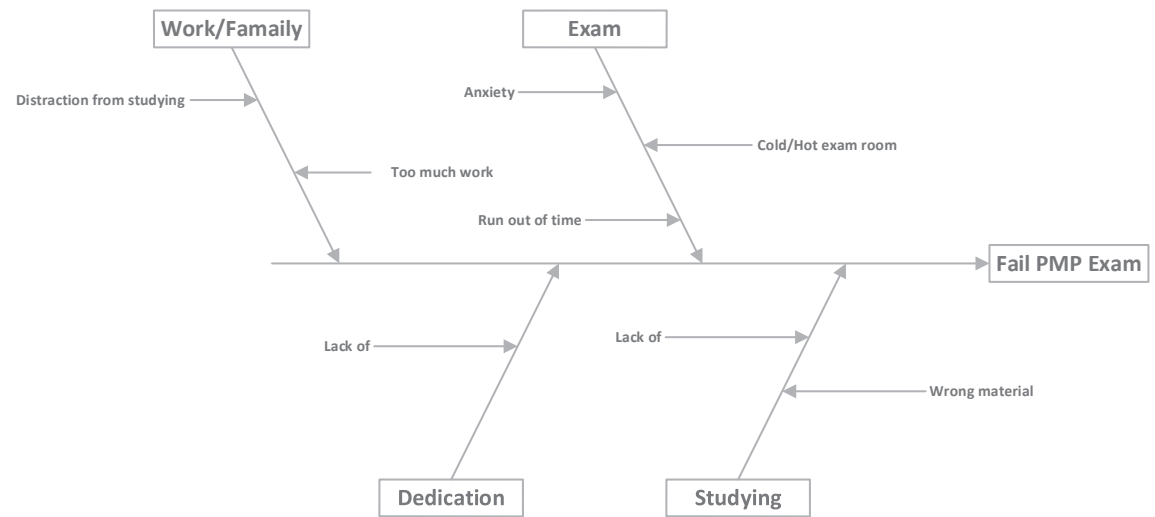
Helps to understand what was found

Activities Include:

- Brainstorming
- Five Whys: asking why five times
- Fishbone analysis
- Prioritize with dots: use a dot voting technique

# Fishbone Analysis

---



## 4. Decide what to do

---

Decide what to do about the problems that was found

How can we improve for the next iteration

Activates include:

- Short Subjects
- Smart Goals

# Short Subjects

---

Team decides what actions to take in the next iteration:

- Start doing
- Stop doing
- Do more of
- Do less of



# SMART Goals

---

Team sets goals that are SMART:

- **S**pecific
- **M**easurable
- **A**ttainable
- **R**elevant
- **T**imely

## 5. Close the Retrospective

---

Opportunity to reflect on what happened during the retrospective

Activities include:

- Plus/Delta: make two column of what the team will do more of and what to do less of

# Team Self-Assessments

---

Uses to evaluate the team as a hold

Things to evaluate can include:

- Self-organization
- Empowered to make decisions
- Belief in vision and success
- Committed team
- Trust each other
- Constructive disagreement