Agile Methodologies

WELCOME

Program – Ground Rules

Phone on Silent Mode

Ask if in Doubt

Be Punctual

Notify Facilitator if leaving

Have FUN!

Acknowledgement

The information and slides have been reproduced from multiple sources…

I hereby acknowledge their direct/indirect contribution

The images and pictures used in this presentation are only for representation purposes and are not a reflection of any political or social views of these individuals

Agile Methodologies

Program Agenda

Day 1

- Waterfall Method
- Introduction To Agile
- History Of Agile
- Agile Manifesto
- Key Agile Principles, Terminology
- Agile Project Characteristics
- Introduction To SCRUM, Key SCRUM Roles, Ceremonies
- SCRUM Projects :- Day 1 To Release
- SCRUM – Lifecycle
- KANBAN

Day 2

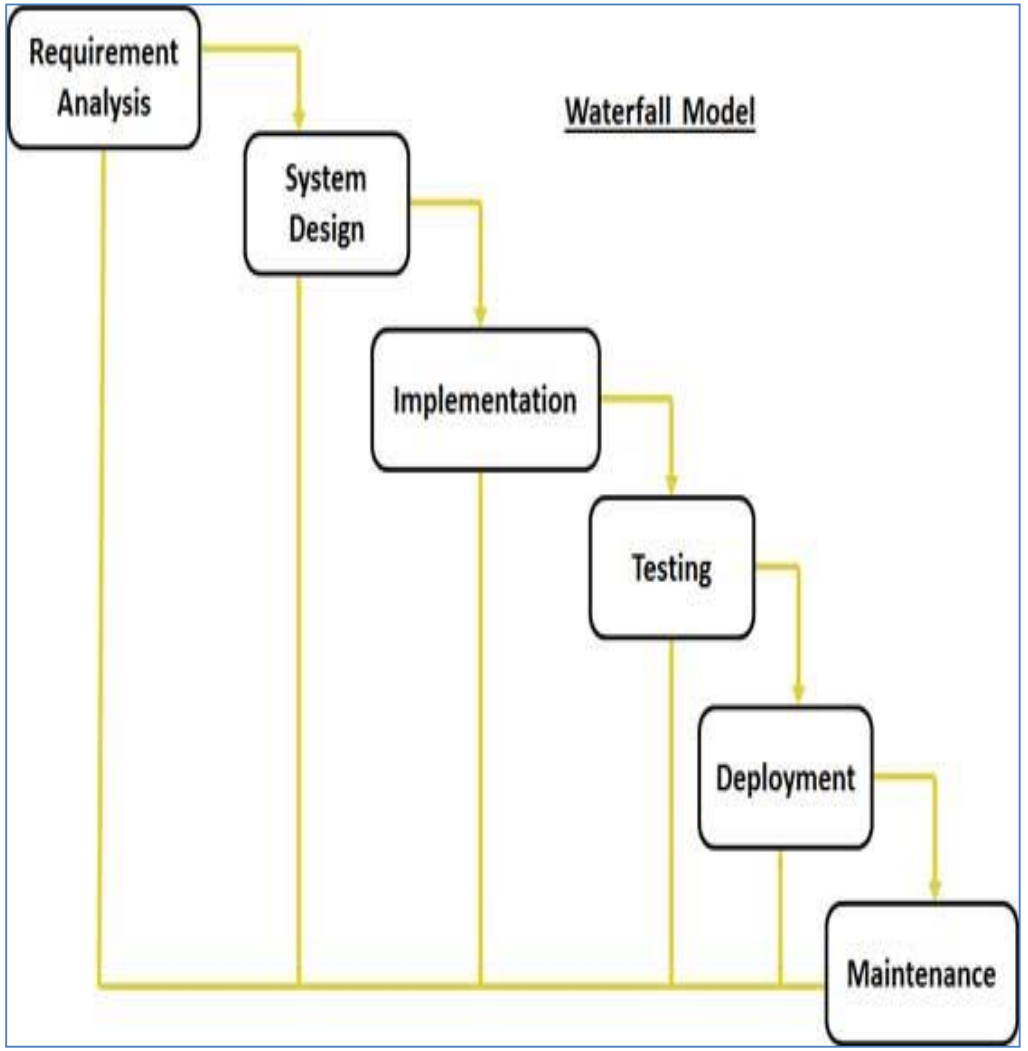
- User Stories : 3C's , User Story Model, Persona
- Estimation in SCRUM Projects :- Planning Poker, User Stories, Story Points
- Communication In Agile Project
- Leadership Styles, Servant Leadership
- Team Dynamics In an Agile Project
- Testing In Agile Projects, Test Driven Development

Waterfall / SDLC Lifecycle

Traditionally software development has been carried out using the classical waterfall methodology (also called SDLC)

Key features of this model are:-

- ✓ Sequential process in which progress is seen as flowing steadily downwards (like a waterfall)
- ✓ Logical and easy to implement model.. If requirements are frozen at start
- ✓ Clients may not know exactly what requirements they need before reviewing a working prototype
- ✓ Changes, especially in the later lifecycle are costly to implement



Waterfall / SDLC Lifecycle – Who Invented It ??

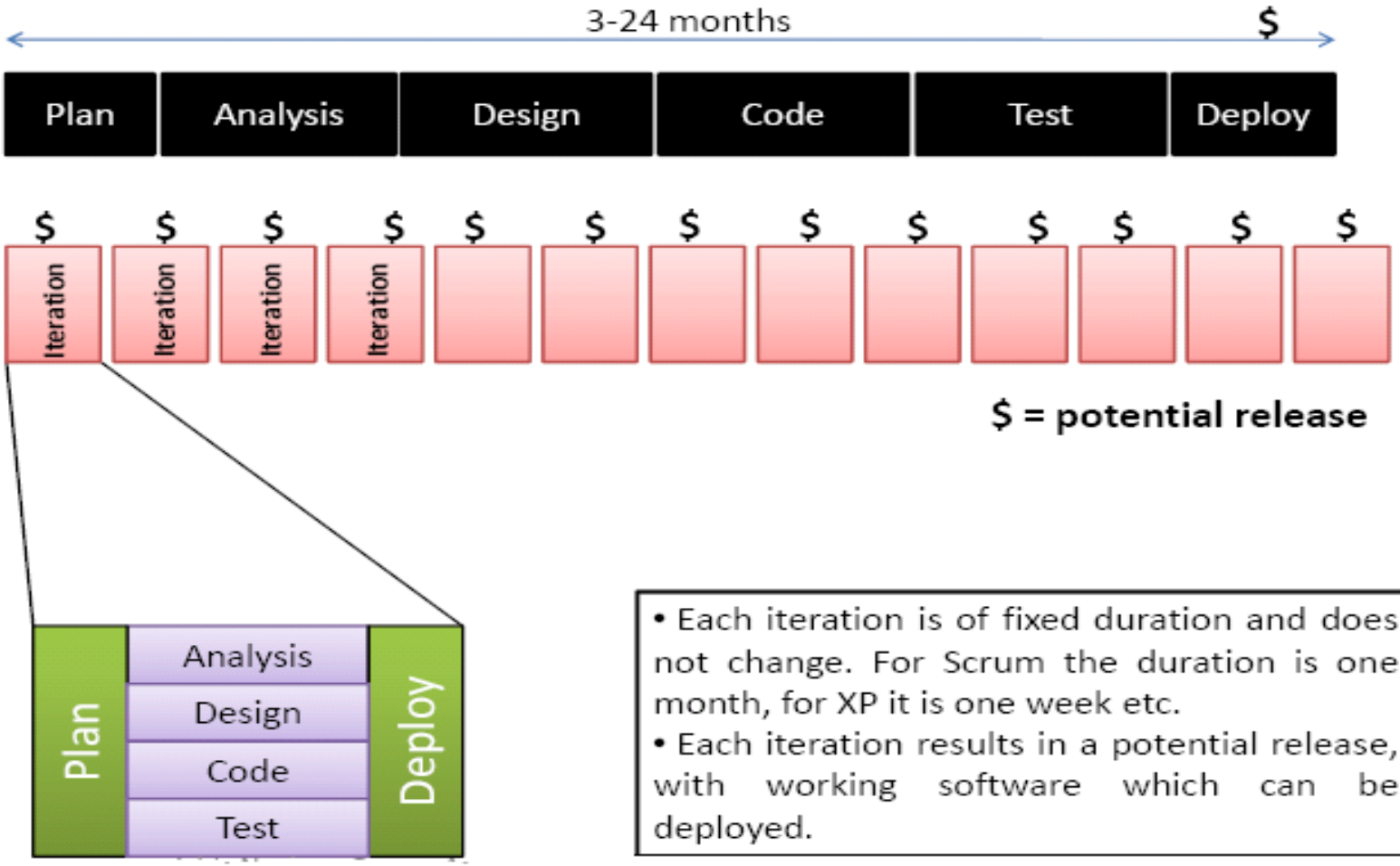
Lets go back in time to see the true story of waterfall

Waterfall / SDLC Lifecycle – Key Disadvantages

Disadvantages Of Waterfall Method

- ✓ Works excellent for projects where the requirements are frozen at the start
- ✓ However for a dynamic industry like software, where changes to requirements are inherent, managing these changes is a nightmare
- ✓ The later a change is introduced, the more costlier it is to implement
- ✓ Customer Involvement is only at the time of requirement gathering and during implementation time

How Agile Compares With Waterfall?



- Each iteration is of fixed duration and does not change. For Scrum the duration is one month, for XP it is one week etc.
- Each iteration results in a potential release, with working software which can be deployed.

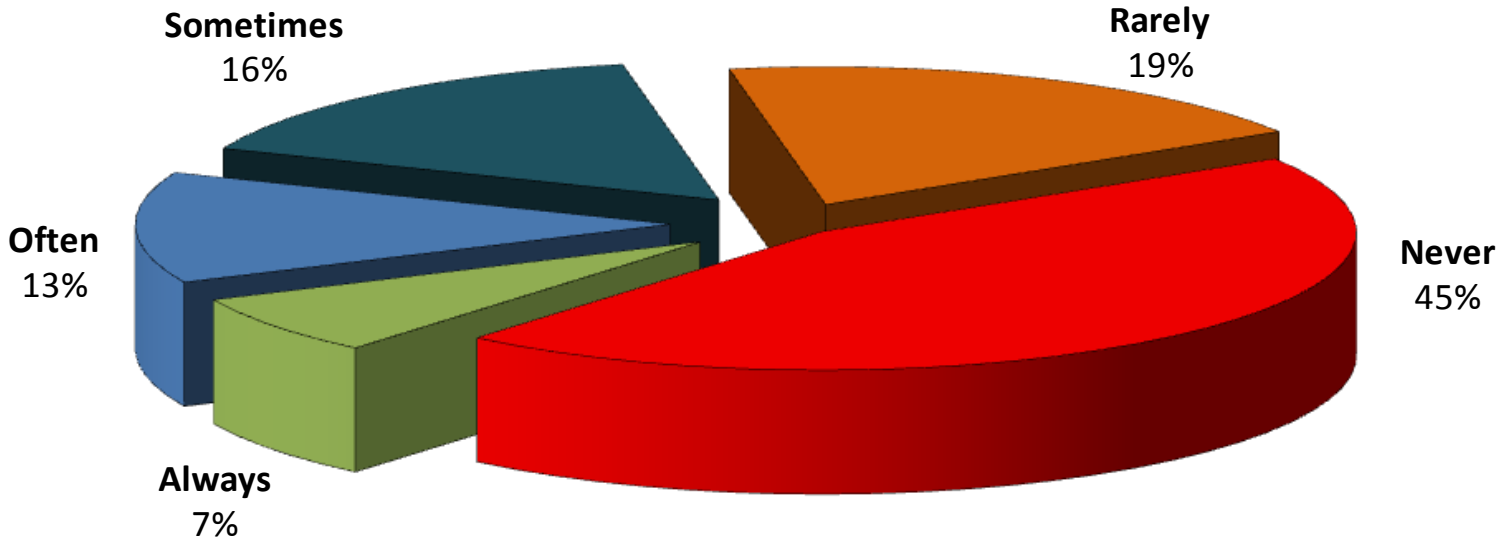
Agile Methodologies

Agile Vs. Waterfall

Key Agile Principles	Traditional Waterfall
Focus on Highest Value First Align project, product and team visions by prioritizing by business needs, and using well architected code, to deliver better quality products faster and cheaper.	**All or nothing** Tight coupling & bias toward building out internals in a breadth first fashion means that nothing can be delivered in isolation, even if it's valuable.
Responding to Change Acknowledge uncertainty & adapt to both external (market) and internal changes, by modifying plans & approach. Use engineering principles to make code base easy to modify.	**Baseline & Change Control** Constrain or even completely eliminate any significant change other than dropping features. Work to initial plans, even when they are proven to be invalid.
Iterative Use short time boxes to provide a rhythm, continually flow of value to the customer, and refine deliverables over time.	**Phased** No rhythm as project is executed using long phases.
Feedback & Continuous Improvement Use continual feedback to inform plans and modify approach.	**Lessons Learned at the End** Feedback is rarely given in time for it to be applied towards improving processes and project execution.
Small, Integrated Teams A small team size, and overlap in skills sets, simplifies communications, allows everyone to see the big picture, creates self discipline and provides flexibility.	**Silo Teams with Handoffs** Staff works in functional oriented groups, throwing documentation and code over the wall.
Technical Excellence / Bake Quality In Use TDD , ATDD, refactoring, and other strong engineering principles to ensure quality.	**Inspection** Attempt to ensure quality by after the fact inspection.

Agile Challenges

Rates of Feature Usage in Software Projects:



Always or Often Used
20%

Never or Rarely Used
64%

Standish Group Study Reported at XP2002 by Jim Johnson, Chairman

What Do You Think ??

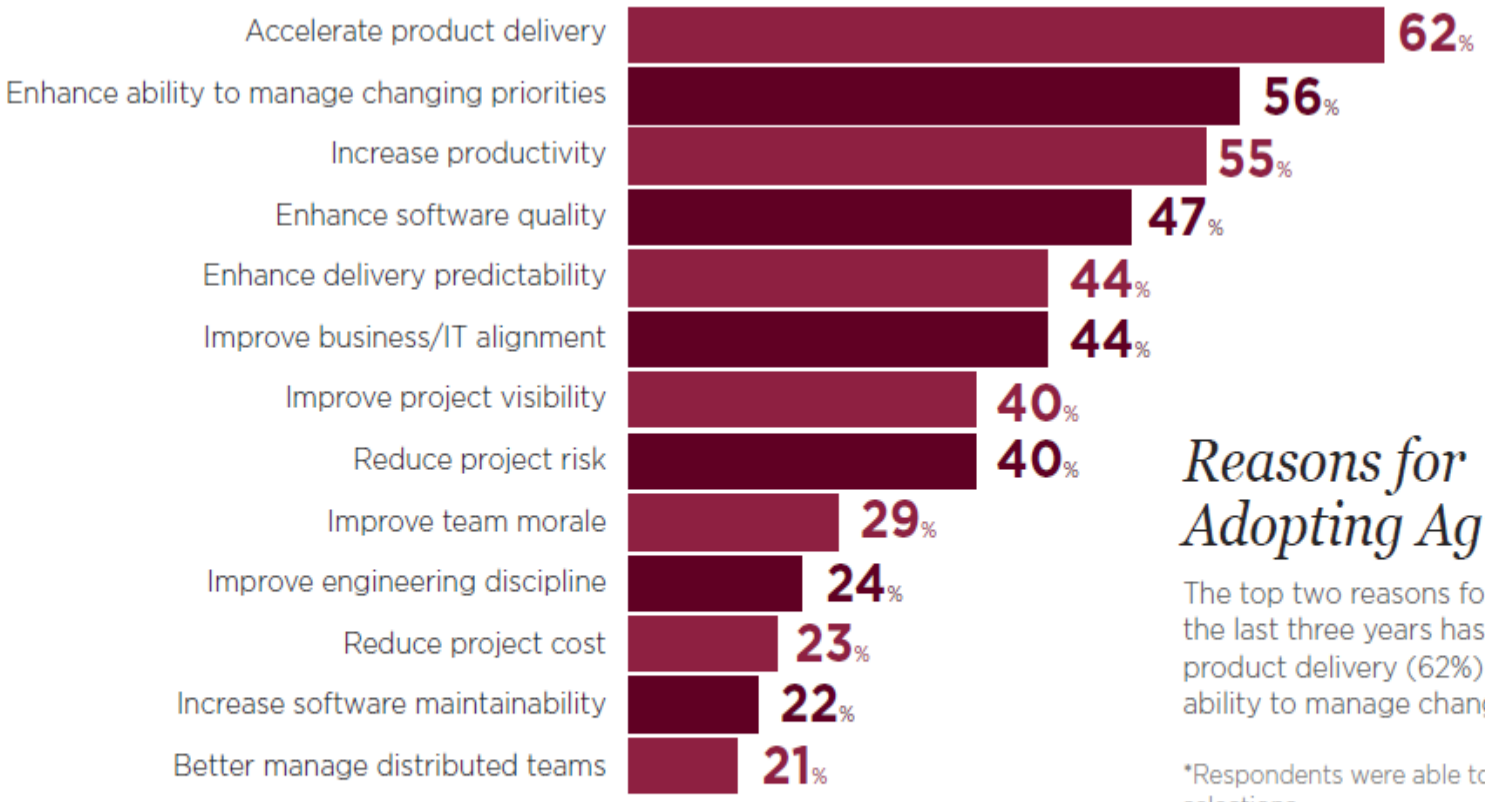
What Do You Think ??

What Do You Think ??

To be successful in projects, you must know all the requirements in advance!

That's not realistic. In projects, you only really know all the requirements at the end of the project!

Why Agile ?



Reasons for Adopting Agile

The top two reasons for adopting agile for the last three years has been to accelerate product delivery (62%) and enhance their ability to manage changing priorities (56%).

*Respondents were able to make multiple selections.

VERSIONONE® Agile Made Easier VERSIONONE.COM

Source: 2016 ***The State of Agile Development* Survey**, VersionOne

Why Agile ?

Top 3 Benefits of Agile

87%

Ability to manage changing priorities

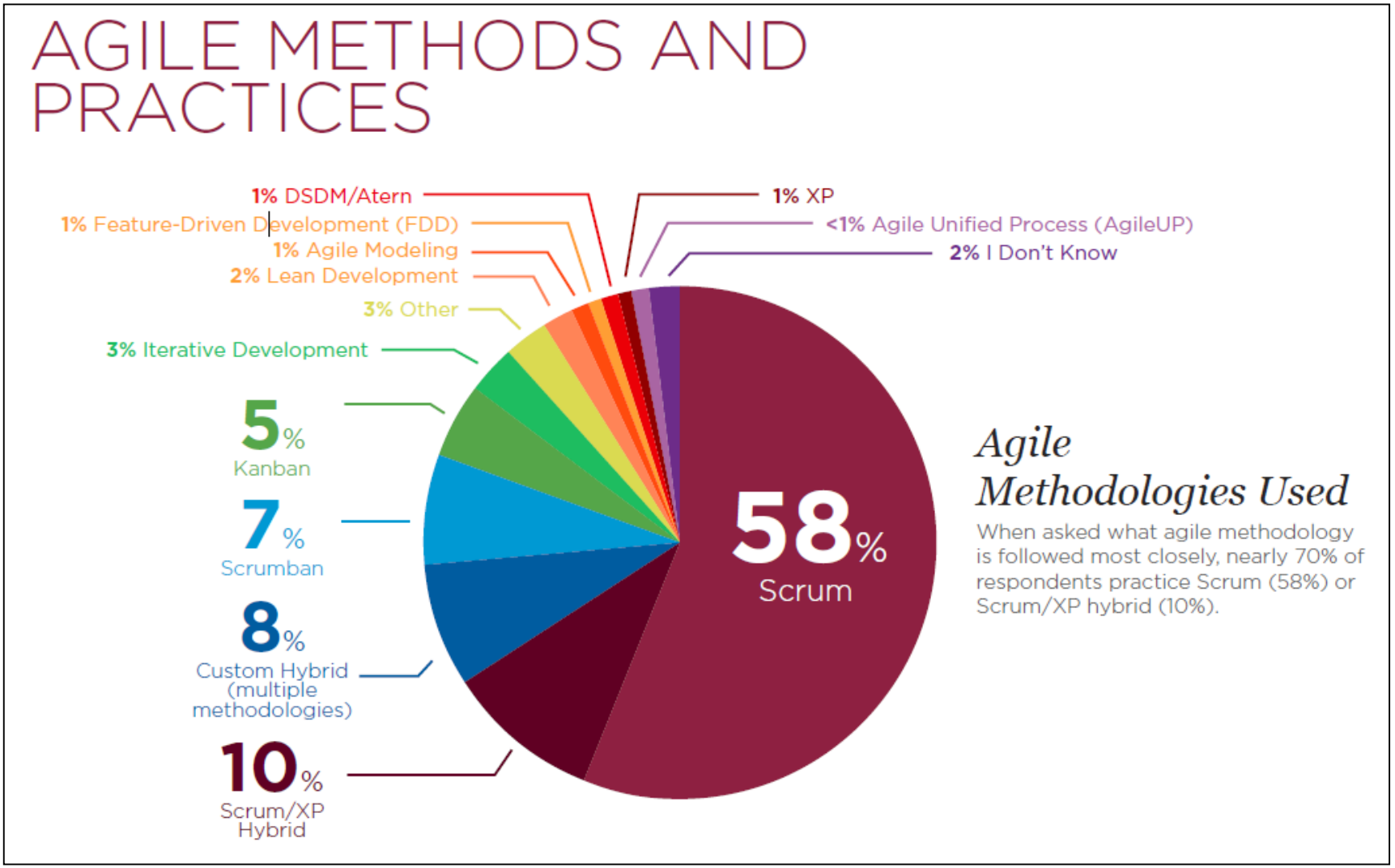
85%

Increased team productivity

84%

Improved project visibility

Why Agile ?


AGILE METHODS AND PRACTICES
1% DSDM/Atern
1% Feature-Driven Development (FDD)
1% Agile Modeling
2% Lean Development
3% Other
3% Iterative Development
5% Kanban
7% Scrumban
8% Custom Hybrid (multiple methodologies)
10% Scrum/XP Hybrid
1% XP
<1% Agile Unified Process (AgileUP)
2% I Don't Know
58% Scrum
Agile Methodologies Used
When asked what agile methodology is followed most closely, nearly 70% of respondents practice Scrum (58%) or Scrum/XP hybrid (10%).

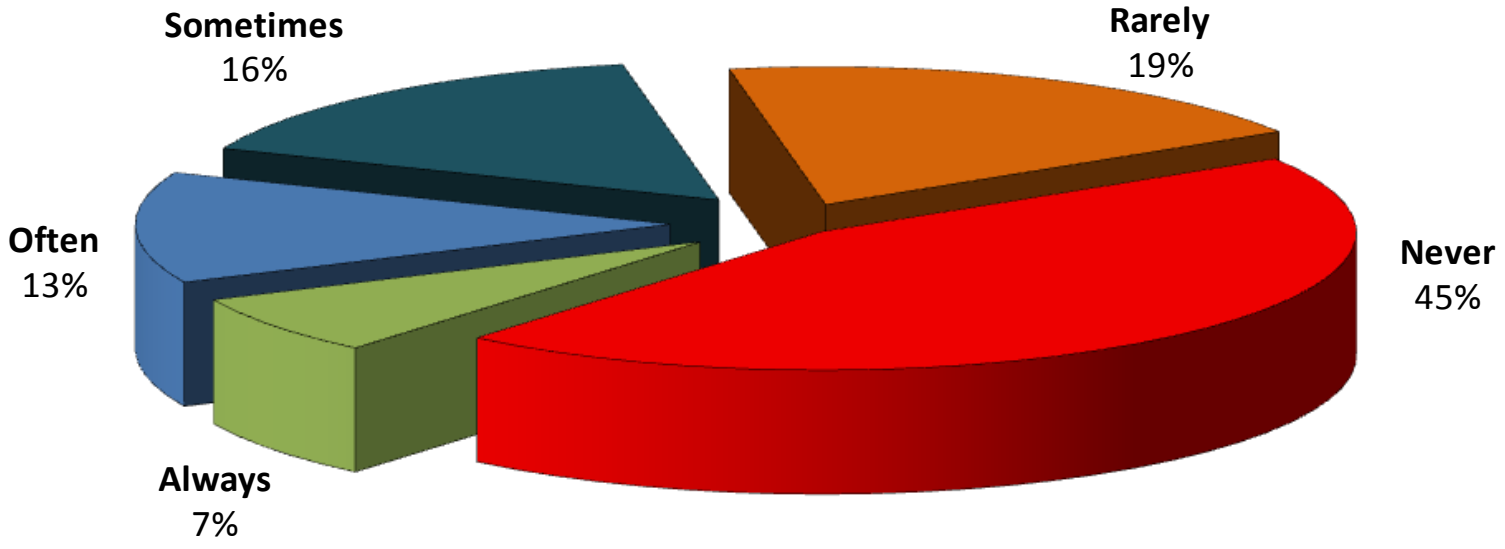
Agile Challenges

Barriers to Agile adoption

As with any significant process change, the biggest barrier seen to the adoption of Agile development is the ability to change organizational culture.

Agile Challenges

Rates of Feature Usage in Software Projects:



Always or Often Used
20%

Never or Rarely Used
64%

Standish Group Study Reported at XP2002 by Jim Johnson, Chairman

What is Agile?

Agile software development is a group of software development methodologies….

- ✓ Based on iterative and incremental development
- ✓ Where requirements and solutions evolve through, collaboration between self-organizing and cross-functional teams
- ✓ Promotes adaptive planning, evolutionary development, early delivery, continuous improvement and encourages rapid and flexible response to change

History of Agile

✓In February 2001, 17 software developers met at a ski resort in Snowbird, Utah, to discuss lightweight development methods

✓They published the "**Manifesto for Agile Software Development**" to define the approach now known as agile software development

✓ The Manifesto for Agile Software Development, also known as the Agile Manifesto, which first laid out the underlying concepts of Agile development, introduced the term in 2001

Agile Manifesto

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 and interactions	over	Process and 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.

Source: www.agilemanifesto.org

Agile Methodologies

The Agile Manifesto is based on 12 principles

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.
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.
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
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

Flavors of Agile

Dynamic System Development Method (DSDM)
Dane Faulkner

Extreme Programming (XP)
Kent Beck

Feature Driven Development (FDD)
Jeff DeLuca

Scrum
Ken Schwaber

Lean Software Development
Mary Poppendieck

Adaptive Software Development (ASD)
Jim Highsmith

Crystal Clear
Allistair Cockburn

Behavior driven development (BDD)

Agile Methodologies

Agile Flavors

Scrum	• Scrum teams typically work in iterations (called sprints) that are from two weeks to one month long. • During each sprint, teams pull from a prioritized list of customer requirements, called user stories , so that the features that are developed first are of the highest value to the customer. • At the end of each sprint, a potentially shippable product is delivered.
XP	• XP methodology was created by Kent Beck. The XP improves a software project in four essential ways which are communication, simplicity, feedback and courage. • It is introduced twelve best practices with which XP programmers are able to courageously respond to changing requirements and technology
ASD	• Adaptive Software Development is a software development process that grew out of rapid application development work by Jim Highsmith and Sam Bayer. • ASD embodies the principle that continuous adaptation of the process to the work at hand is the normal state of affairs

Agile Methodologies

Agile Flavors

Kanban	Kanban is a method for managing knowledge work with an emphasis on just-in-time delivery while not overloading the team members. In this approach, the process, from definition of a task to its delivery to the customer, is displayed for participants to see and team members pull work from a queue.
Lean Software Development	Lean is a translation of Lean manufacturing and Lean IT principles and practices to the software development domain. Adapted from the Toyota Production System, a pro-lean subculture is emerging from within the Agile community.
FDD	Feature-driven development (FDD) is an iterative and incremental software development process. It is one of a number of lightweight or Agile methods for developing software. FDD blends a number of industry-recognized best practices into a cohesive whole. These practices are all driven from a client-valued functionality (feature) perspective. Its main purpose is to deliver tangible, working software repeatedly in a timely manner.

Agile Project – Key Characteristics

Key Characteristics of Agile Project
Adaptability rather than predictability
"Fits just right" process
Ongoing customer involvement
Consistent team collaboration
Rapid response to change
Frequent delivery of working software
Continuous testing and validation
People rather than development process

Agile Team
Smaller in Size
A good guideline is to have a maximum of 12 members.
Assigning partial work is not recommended.
Multi-tasking is not recommended in Agile methods as it decreases productivity and hinders team work

Agile - Important Terms

Refactoring	Changing the code structure (rephrasing) without changing its behavior to improve code quality and make it easy to add new features
Technical Debt	Total amount of "less than perfect" design and implementation decisions in the project. Quick and dirty hacks made to make something work right now. This usually increases the maintenance cost and cost of adding new features. Refactoring is done to reduce technical debt so that code can be reused.
Time Boxing	Work invariably stretches to use the existing time (Student syndrome). Time boxing prevents this by ensuring a work stops when it is time up.
Product Backlog	Scrum term for the prioritized list of all the functionality desired in the Product
Done Done	Release ready feature is considered as **"Done Done".** Partially finished stories destabilize release planning efforts and hence all stores need to be ensured to be **"Done Done"** by the end of iteration.

Agile - Important Terms

User Story	A short description of the functionality told from the perspective of the end user or customer. User Stories are typically documented on a index card and are typically written in a single line
Iterations/ Sprints	An iteration is the full cycle of design-code-verify-release of an Agile team. Each iteration will involve customer deciding what stories to implement in the iteration and results in an software which the customer can use.
Burndown Charts	A burn down chart is a graphical representation of work left to do versus time. The outstanding work (or backlog) is often on the vertical axis, with time along the horizontal axis
Last Responsible Moment	Make decisions during the moment at which failing to make a decision eliminates an important alternative. Any delay beyond this time, the decisions are made by default which is NOT good. By delaying the decision till this point you improve accuracy of decisions, decrease your workload and decrease impact of changes due to more elaborate and clear information available at the time of decision

Agile - Important Terms

Velocity	A measure to be able to determine the amount of work which can be done in an iteration.
Theory Of Constraints	Every system has a single constraint that determines the overall throughput of the system. One must find the constraints in the system (developers, programmers, testers) and eliminate the constraint.
Collocation	Agile methods rely on high bandwidth and fast communication and this is better done if the team is in same location
Pair Programming	Two members share the same keyboard. One member does the coding (and the tactical issues concerned) and the other members thinks about the big picture (Driver & Navigator)
Energized Work	When you are energized you work better. One of the best ways to feel energized is to take care of yourself. Go home on time every day, spend time with family & friends and do engage in activities that take your mind off of work

Agile - Important Terms

Informative Work Space	An informative work space brings information into the team. It allows the team to sense the state of project when they take breaks. An informative work space also provides for people to communicate easily
Root Cause Analysis	A method of problem solving that tries to identify the root causes of faults or problems. A root cause is a cause that once removed from the problem fault sequence, prevents the final undesirable event from recurring
Daily Standup	A short 15 minute meeting where all the members stand up and face each other while talking. Objective is to ensure that the team is in line with their commitments
Retrospective	Retrospectives (Introspective lessons learned meetings) are a great way to continually improve a process. Iteration retrospective happens at the end of each sprint.

Agile - Important Terms

Customer Involvement	Customer involvement improves the success of project. In house customer development: Try turning real customers into onsite customers Outsourced customer development: May be easier to have your team to customer's offices. Try meeting for iteration demo or retrospectives or planning sessions
Trust	Trust is essential for a team to improve productivity.
Coding Standard	Creating of coding standards will help increase communication and collaborative work.

Agile Benefits

Why Agile Software Development Pays?

SCRUM

(Rugby Union Scrummage)

What Is SCRUM?

Scrum is an iterative and incremental agile software development framework for managing product development

Scrum defines a flexible, holistic product development strategy where a development team works as a unit to

- Reach a common goal
- Challenges assumptions of the "traditional, sequential approach" to product development
- Enables teams to self-organize by encouraging physical co-location or close online collaboration of all team members
- Daily face-to-face communication among all team members and disciplines in the project.

- In 1995, Ken Schwaber and Jeff Sutherland jointly presented a paper describing Scrum in Austin, Texas

- In 2001, Ken Schwaber teamed up with Mike Beedle to describe the method in the book "Agile Software Development with Scrum"

What Is SCRUM?

- Although the word Scrum is NOT an acronym, some companies implementing the process have been known to spell it with capital letters as SCRUM

- This may be due to one of Ken Schwaber's early papers, which capitalized SCRUM in the title

SCRUM Roles

PIGS	Pigs are the core roles in a Scrum team and these people are c**ommitted!!** • Product Owner • The Team • SCRUMMaster
Chickens	These are individuals performing other roles in a SCRUM team and are basically **involved** and **not committed** • Functional Managers • Network Administrators/Sys Admin • SMEs, DBAs

Agile Methodologies

Key SCRUM Roles

Product Owner	• Represents the voice of the customer and is accountable for ensuring that the Team delivers value to the business • The Product Owner writes customer-centric items (in form of user stories), prioritizes them, and adds them to the product backlog (priority list of requirements)
Team	• The Team is responsible for delivering the product. • SCRUM team sizes are between 5–9 people with cross-functional skills who do the actual work • Team is self led and self organizing
SCRUM Master	• Is accountable for removing impediments to the ability of the team to deliver the sprint goal/deliverables • A key part of the Scrum Master's role is to protect the team and keep them focused on the tasks at hand. • Scrum Master is not the team leader but acts as a buffer between the team and any distracting influences

Agile Methodologies

SCRUM Features

3 Roles

1	Product Owner
2	The Team
3	SCRUM Master

4 Ceremonies

1	Sprint Planning
2	Daily SCRUM
3	Sprint Review
4	Sprint Retrospective

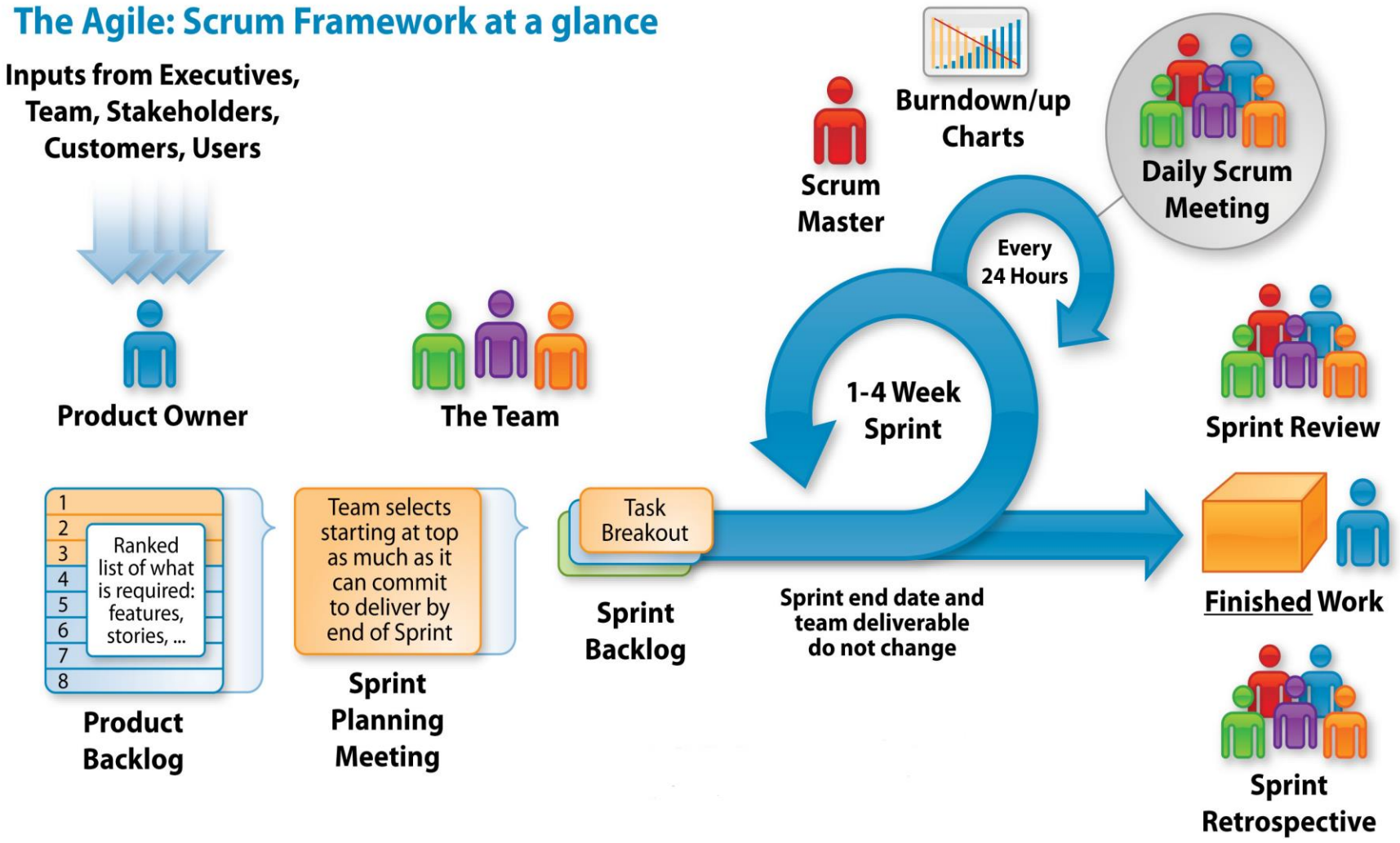
3 Artifacts

1	Product Backlog
2	Sprint Backlog
3	Burndown Chart

Sprint Planning	Team meets with the product owner to choose a set of work to deliver during a sprint
Daily SCRUM	Team meets each day to share struggles and progress
Sprint Review	Team demonstrates to the product owner what it has completed during the sprint
Sprint Retrospective	Team looks for ways to improve the product and the process

Product Backlog	Prioritized list of desired project features
Sprint Backlog	Set of work from the product backlog that the team agrees to complete in a sprint, broken into tasks
Burndown Chart	A graphical representation of the work remaining (can be of 2 types - one for the sprint and one for the overall project)

SCRUM at a Glance



SCRUM Lifecycle

- ✓ A product owner creates a prioritized wish list called a product backlog based on the inputs from the key stakeholders
- ✓ During sprint planning, the team pulls a small chunk from the top of that wish list, a sprint backlog, and decides how to implement those pieces
- ✓ The team has a certain amount of time, a sprint, to complete its work - usually two to four weeks - but meets each day to assess its progress (daily scrum)
- ✓ Along the way, the Scrum Master keeps the team focused on its goal
- ✓ At the end of the sprint, the work should be potentially shippable, as in ready to hand to a customer, put on a store shelf, or show to a stakeholder. The sprint ends with a sprint review and retrospective
- ✓ As the next sprint begins, the team chooses another chunk of the product backlog and begins working again

SCRUM Basics

Let us revisit Scrum Basics

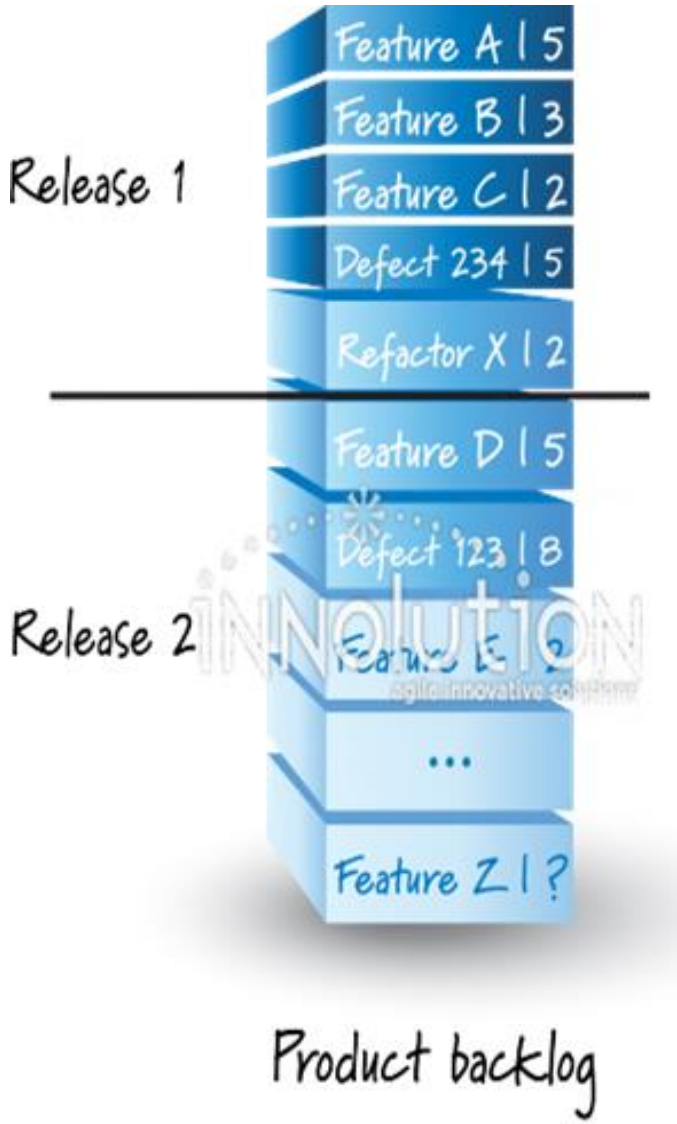
Scrum Project
From Day 1 to Release

Product Backlog

- ✓ A product backlog is a prioritized list of features and is created by the Product Owner (PO)
- ✓ Product Owner (PO) articulates the product vision in any way that is clear and sustainable, though either Use Cases or "user stories"
- ✓ Product Backlog is continuously updated by the Product Owner to reflect changes in the needs of the customer and new ideas or insights
- ✓ There exists only a single Product Backlog for a project

Release Backlog

- ✓ A subset of the Product Backlog that is intended for the current release is known as the Release Backlog
- ✓ The goal of a given release is to deliver a subset of the product backlog, known as the release backlog
- ✓ Team provides the Product Owner with estimates of the effort required for each item on the Product Backlog
- ✓ After identifying which user stories will go into a particular release, the user stories become part of a release backlog…



Sprint Planning Meeting

- ✓ During the sprint planning meeting, the product owner describes the highest priority features to the team.
- ✓ Sprint Planning meeting takes place at the start of each sprint
- ✓ There are two defined artifacts that result from a sprint planning meeting:
  - A sprint goal
  - A sprint backlog
- ✓ The team asks enough questions that they can turn a high-level user story of the product backlog into the more detailed tasks of the sprint backlog

Sprint Planning Meeting

- ✓ Product Owner and Team also review the "Definition of Done"
- ✓ Team does a detailed task planning on how to implement items decided earlier
- ✓ Team decides how much work it will commit to complete, rather than having it assigned to them by the Product Owner or Scrum Master
- ✓ Common technique used to determine the work commitment is done by using "Work Time Available" /Available Capacity method
- ✓ Typical duration of Sprint Planning Meeting is from 4-8 hours for a four-week Sprint

Capacity Based Planning / Work Time Available

Name	Days Available Per Sprint	Hours / Day	Total
Bill Gates	10	6	60
Shilpa	10	6	60
Amitabh	8	6	48
Shahrukh	9	6	54
		Total	222
		Buffer (10%)	-22
		Time for Backlog Refinement (5%)	-10
		Available Total	189

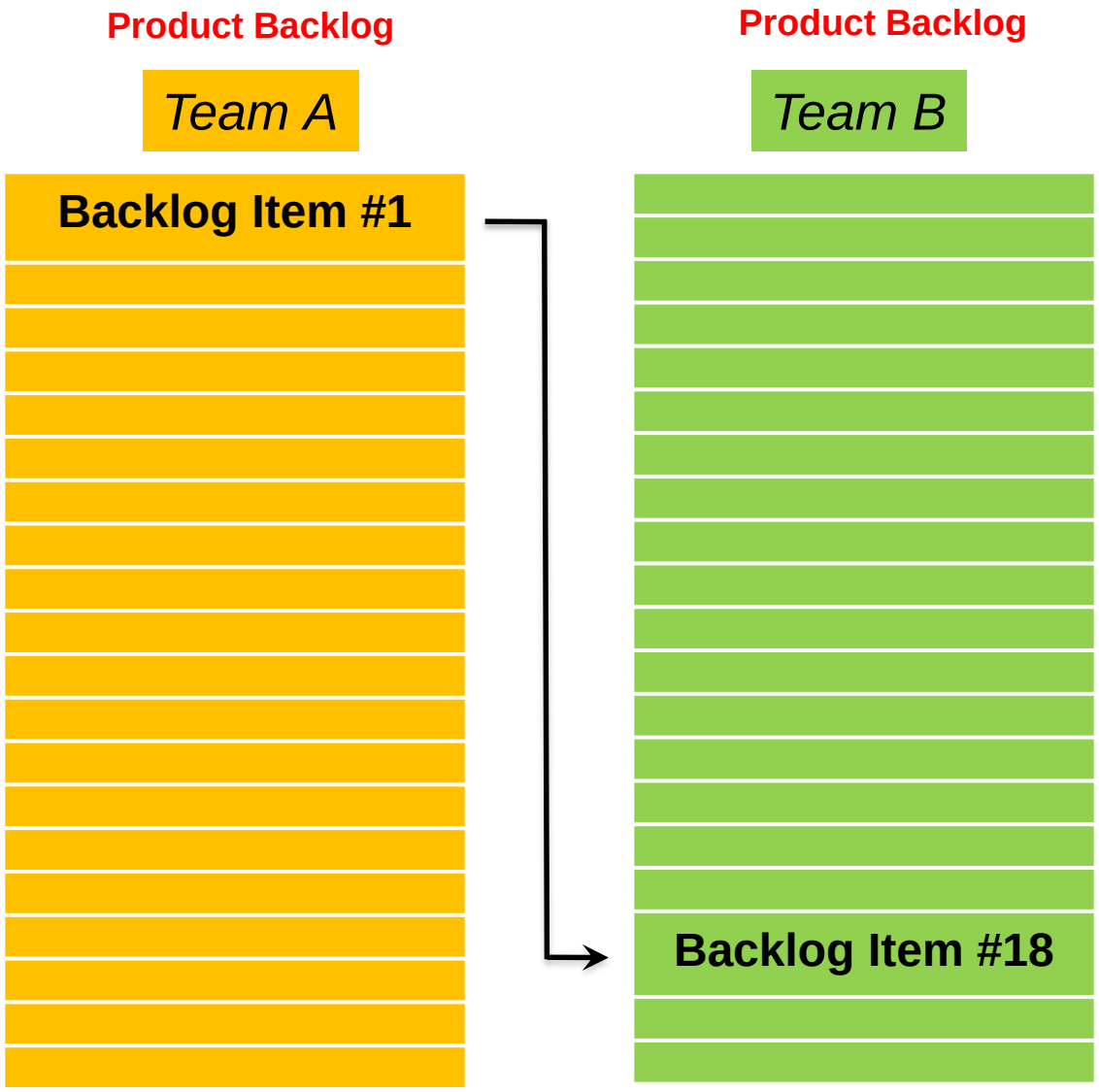
- ✓Sprint Duration Assumed : 10 Days
- ✓Per Day effective work hours = 6

- ✓Based on this analysis,Team will decide to pick up approx **189** hrs of work from the product backlog for the sprint.

Capacity Based Planning / Work Time Available

- ✓ Once capacity is determined, team would decide upon how many items from the product backlog can be picked up in this sprint
- ✓ Team picks up the 1st item from product backlog, breaks down into details into a document called as a Sprint Backlog
- ✓ Team will move sequentially down the Product Backlog in this way, until it's used up all its estimated capacity
- ✓ One of main pillars of Scrum is that once the Team makes its commitment, any additions or changes must be deferred until the next Sprint

Dependency Determination



- Identify the dependency before Sprint Commitment is made

Then, either…

- Product Owner A reduces priority of #1
  or
- Product Owner B increases priority of #18
  or
- Product Owner A shifts #18 to Team A Product Backlog and Team A builds it
  or
- Team A uses mock object in place of #18, and replaces with actual #18 later

Sprint Backlog

Sprint Backlog

				New Estimates of Effort Remaining as of Day...					
Product Backlog Item	**Sprint Task**	**Volunteer**	**Initial Estimate of Effort**	1	2	3	4	5	6
As a buyer, I want to place a book in a shopping cart	modify database		5						
	create webpage (UI)		8						
	create webpage (Javascript logic)		13						
	write automated acceptance tests		13						
	update buyer help webpage		3						
	. . .								
Improve transaction processing performance	merge DCP code and complete layer-level tests		5						
	complete machine order for pRank		8						
	change DCP and reader to use pRank http API		13						

Goal of Sprint….

The Team goal at the end of each Sprint is to be "Done" with the Product Backlog Items the team committed to

But what does "Done" mean?

Coded?

Coded and tested?

Coded and tested with no defects remaining?

To avoid any ambiguity or confusion, the Product Owner and the Team must agree on a "*Definition of Done*"

DEFINITION OF "DONE"

FOR FUNCTIONALITY TO BE "DONE" AT THE END OF A SPRINT,
IT MUST HAVE COMPLETED THE FOLLOWING STEPS:

1. Code complete
2. Code reviewed
3. Unit tests written and executed
4. Integration tested
5. Documented
6.
7.
8.
9.
10.
11.
12.
13.
14.
15.

OF DEFECTS PERMITTED?

Sprint Start

- ✓ Team members pick up tasks from the sprint backlog and start working on the same
- ✓ Teams also make use of a visual task-tracking tool, in the form of a wall-sized task board where tasks migrate during the Sprint across columns labeled “Not Yet Started,” “In Progress,” and “Completed.”
- ✓ ScrumMaster protects the team from outside influences and is responsible for removing any roadblocks
- ✓ Product Owner is not able to make changes to the selected items under development during the current Sprint, he/she can reprioritize the product backlog for the remaining items

Sprint – Daily Standup

Goal

- ✓ Enable Team to inspect and adapt daily
- ✓ Enable Team to share progress with each other
- ✓ Find out what's blocking or slowing down the Team

Team stands in a circle and reports 3 things

- ✓ What did I do since the last time we met?
- ✓ What will I try to do by the next time we meet?
- ✓ What is blocking me (or slowing me down)?

RECOMMENDED No discussion or debate: listening only / Max 15 minutes

RECOMMENDED ScrumMaster notes the blocks (and then later helps Team resolve them)

RECOMMENDED Team and ScrumMaster only – Team can invite PO if they wish, but it's the Team's decision

Agile Game – Who Wants To be a ScrumMaster?

Game: Daily Standup Meeting

Volunteers:

1 Scrum master , 8 Team Members, 1 Product Owner

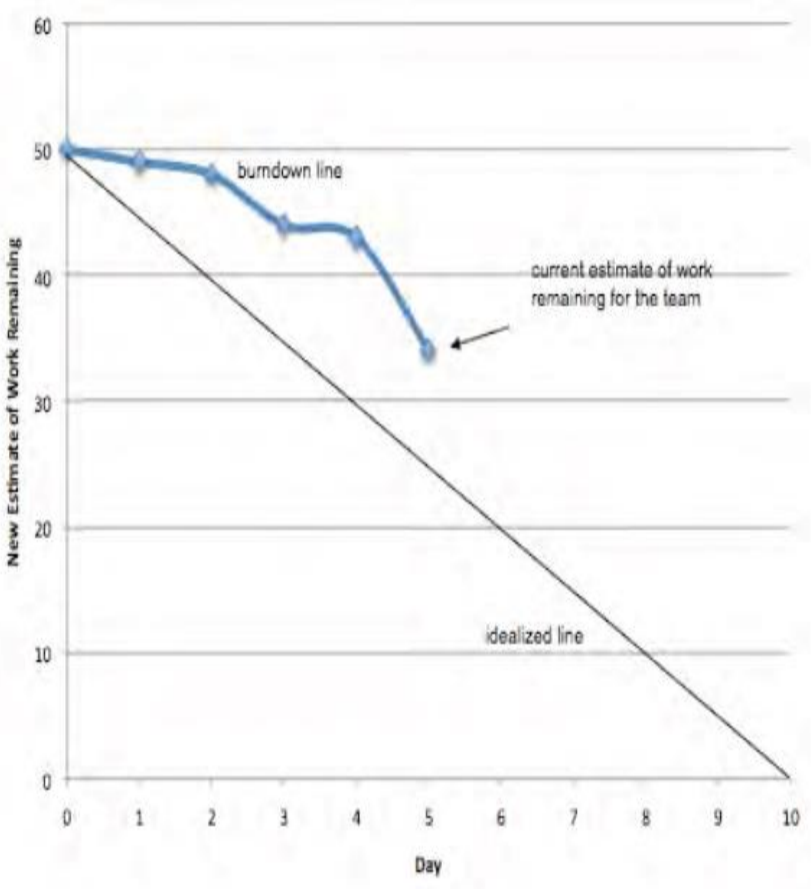
Project Online Travel Company :

Team is working on developing a application which will allow users to book bus tickets from web and mobile

Today is 5th day of Sprint 1 (Total sprint duration is of 30 days)
Conduct the Daily Scrum Meeting for 15 minutes

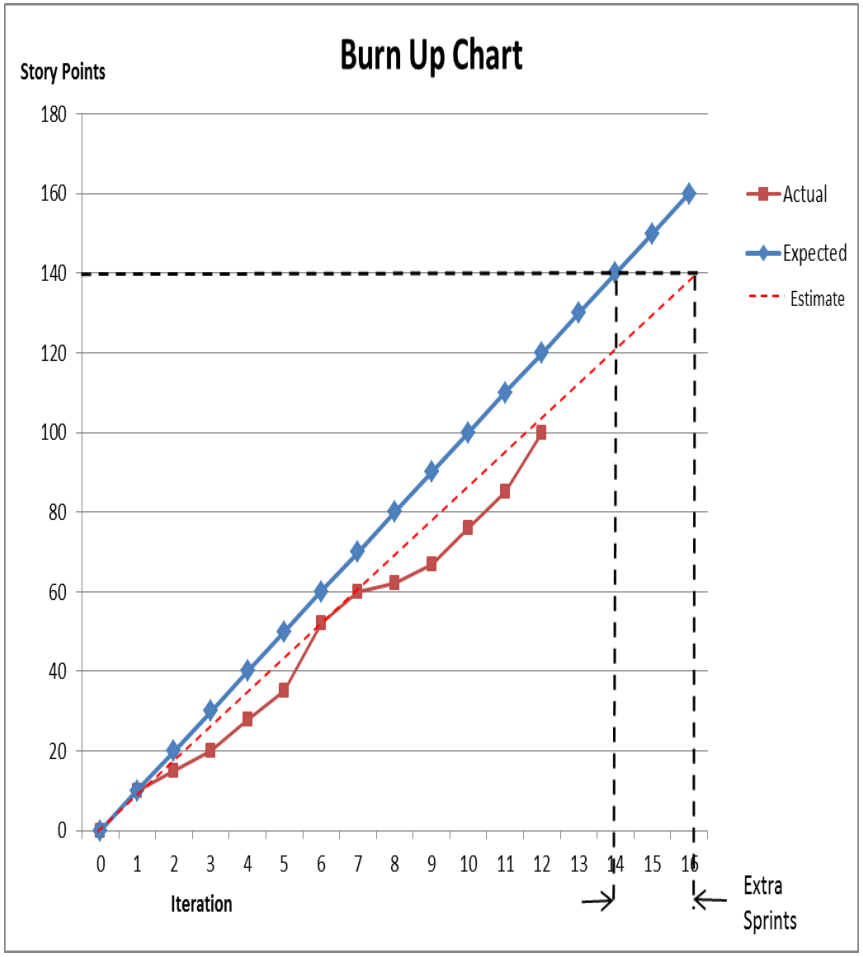
Sprint – Burndown Chart

- ✓ A burn down chart provides a graphical representation of the work left plotted against time.
- ✓ It is useful for predicting when all of the work will be completed.
- ✓ One of the main chart used by agile teams to team tracks its progress against a release plan or to track the progress of a sprint
- ✓ X – Axis: Plots the days in a sprint(Sprint Burndown chart) or number of iterations (Release Burndown chart)
- ✓ Y –Axis: Plots the Total work remaining
- ✓ At the end of each sprint/day , the Work Remaining is updated



Sprint – Burnup Chart

- ✓ A burnup chart is opposite to the Burndown chart
- ✓ Instead of showing the work remaining, it shows the work that has already been completed
- ✓ The burn up chart is read the opposite way to the burn down chart- i.e. graph lines should converge in the upper right corner instead of the lower right
- ✓ It is useful for predicting when all of the work will be completed



Product Backlog Refinement

- ✓ In each sprint 5-10% of time needs to be devoted to Product Backlog refinement
- ✓ Key tasks carried as a part of this include a detailed requirements analysis, splitting large items into smaller ones, estimation of new items, and re-estimation of existing items
- ✓ Refinement activity is not done for items selected for the current Sprint but is for items for the future, most likely in the next one or two Sprints

Sprint – Review

- ✓ Takes place at the end of the sprint
- ✓ Objective is to inspect and adapt the shippable product developed
- ✓ Involves in-depth conversation between the Team and Product Owner to learn the situation, to get advice, and so forth and also includes demo of the product
- ✓ ScrumMaster is responsible for reiterating the "Definition Of Done" to everyone
- ✓ Actual demo of the product … No Slides …No Presentation

Sprint – Retrospective

- ✓ Sprint Retrospective, follows the Review, involves inspect and adapt regarding the process
- ✓ Provides opportunity for the Team to discuss what's working and what's not working, and what can be tried
- ✓ Team and ScrumMaster will attend, and the Product Owner is welcome but not required to attend
- ✓ Usually these meetings are conducted by a neutral person (Peer ScrumMaster)

Sprint – Retrospective

RECOMMENDED

Product Owner should give the ScrumMaster and Team some time to talk alone

RECOMMENDED

The ScrumMaster should invite feedback on their effectiveness from the Team and PO

RECOMMENDED

Invite another team's ScrumMaster to facilitate the discussion

RECOMMENDED

Spend at least 60 minutes on the Retrospective

Sprint – Closure

- ✓ Each sprint is timeboxed and cannot be extended under any circumstances
- ✓ Team typically over-commits in its first few Sprints and fails to accomplish its commitments
- ✓ Sometime teams also overcompensates and under-commits, and finishes early (in which case it can ask the Product Owner for more Product Backlog items to work on)
- ✓ Sprint end results in a potentially shippable product

SCRUM – Important Points

- Not a Silver Bullet
- Is not a solution to every problem
- Is Fast and flexible
- Implementation takes time (No BIG BANG)
- Is Common Sense
- Is Not to manage the software development team
- Not to overrule the informed decisions taken by the team members

Let us do a quick recap of SCRUM

SPRINT "0"

- Initial Sprint to carry out setup activities

- Also called as Iteration Zero, Pre Sprint or Inception Sprint
- The following activities are mainly done in Sprint zero
  - ✓ Get some quality items on the Product Backlog
  - ✓ Provide a minimal environment that enables the writing of quality code
  - ✓ Write a piece of *real code*, no matter how small it is

Reference :

Peter Stevens, an Agile Coach in Switzerland, used a Zero Sprint to estimate the most important features, agree on a definition of done and rebuild confidence with the customer.

Ken Schawber, co-creator of Scrum : "Sprint 0 has become a phrase misused to describe the planning that occurs prior to the first sprint".

Agile Game - Flipping Coins

Rules Of The Game

- ✓ 7 Persons sit around a table in a close circle
- ✓ The persons are named as A, B, C, D, E, F,G
- ✓ Person A has to pick up a coin flip it and then pass to B
- ✓ Person B on receipt flip its again and passes to C
- ✓ The above cycle continues till all the coins reach F and he/she completes the flipping
- ✓ Time keepers need to record the time to complete the exercise for each sprint

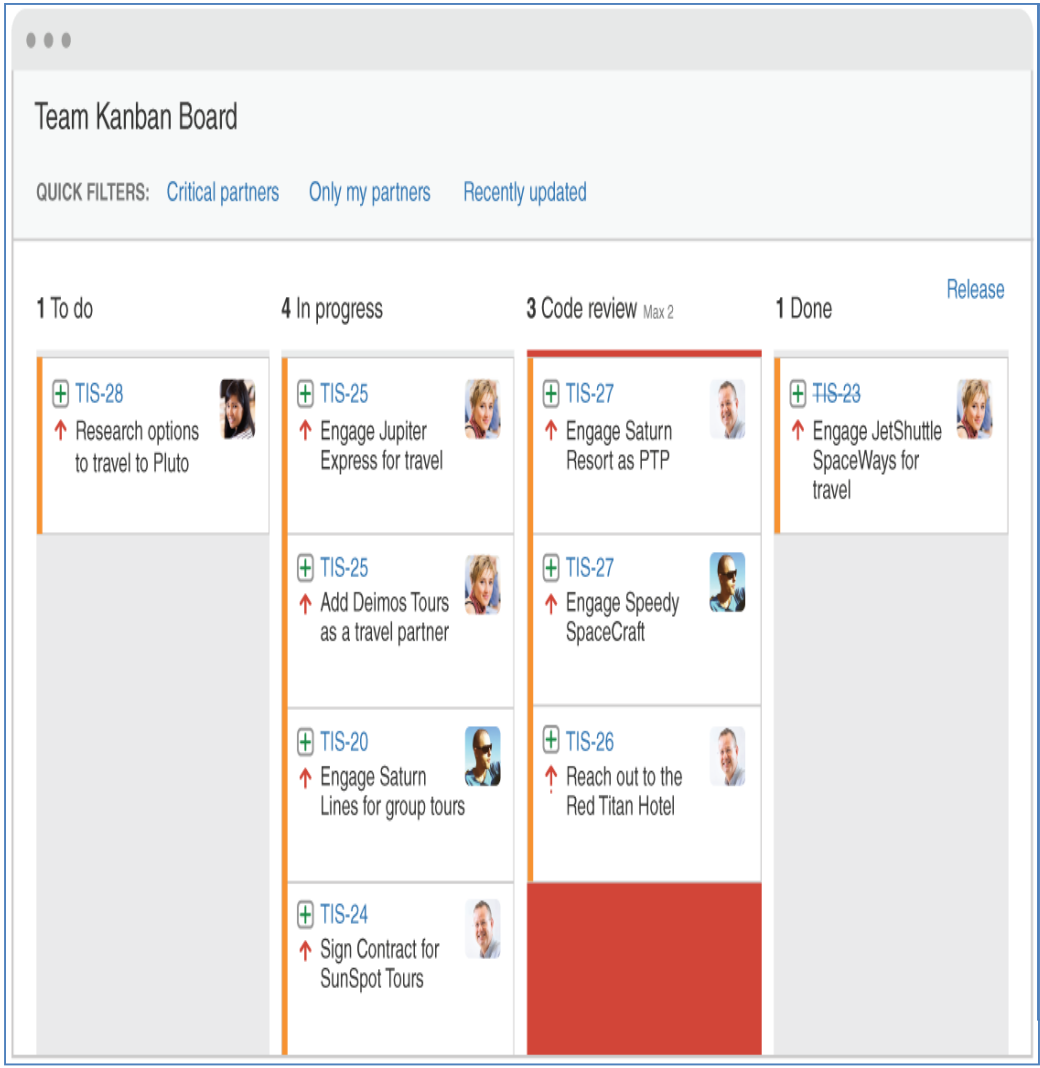
Flipping Coins - Results

Sprint	Lot Size	A	B	C	D	E	F	Total Time
1	20							
2	15							
3	10							
4	5							
5	2							

Process cycle time is directly related to batch size but inversely related to batch frequency

KANBAN

KANBAN



The columns represent the different states of a work item, while rows represent the work items.

Kanban as applied to software development is a pull-based planning and execution method.

Rather than planning work items up front and pushing them into the work queue of a team, the team signals when they are ready for more work and pulls it into their queue.

Kanban historically uses cards to signal the need for an item.

Work in progress (WIP) limits determine the minimum and maximum amount of work that lives in each status of a workflow.

Limiting the amount of WIP improves throughput and reduces the amount of work "nearly done" by forcing the team to focus on a smaller set of tasks

KANBAN

- Kanban matches the amount of work in progress (WIP) to the team's capacity
- Kanban gives teams more flexible planning options, faster output, clear focus, and transparency throughout the development cycle
- A Kanban team is only focused on the work that's actively in progress. Once the team completes a work item, they pluck the next work item off the top of the backlog
- Cycle time is a key metric for kanban teams.
- Cycle time is the amount of time it takes for a unit of work to travel through the team’s workflow–from the moment work starts to the moment it ships.
- By optimizing cycle time, the team confidently forecast the delivery of future work.

Scrum Vs. KANBAN

	SCRUM	KANBAN
Cadence	Regular fixed length sprints (ie, 2 weeks)	Continuous flow
Release methodology	At the end of each sprint if approved by the product owner	Continuous delivery or at the team's discretion
Roles	Product owner, scrum master, development team	No existing roles. Some teams enlist the help of an agile coach.
Key metrics	Velocity	Cycle time
Change philosophy	Teams should strive to not make changes to the sprint forecast during the sprint. Doing so compromises learnings around estimation.	Change can happen at any time

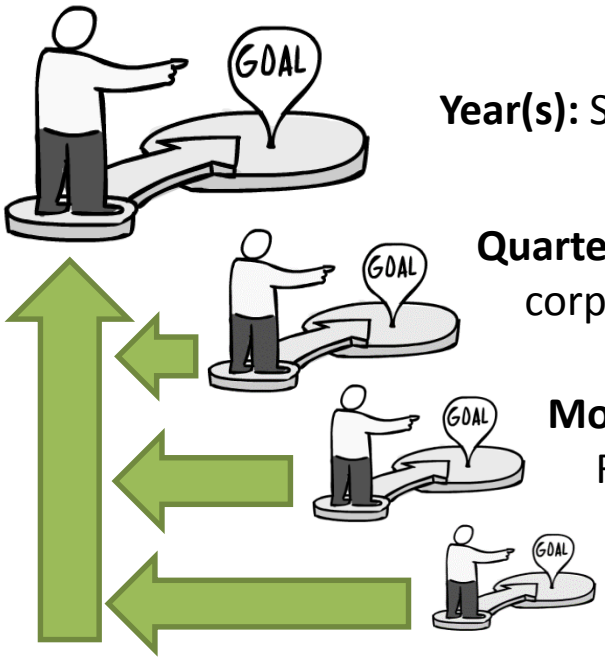
LEAN

Lean

"A production practice that considers the expenditure of resources for any goal other than the creation of value for the end customer to be wasteful, and thus a target for elimination."

Lean

- Corporate strategy operates over years with direct line of sight to priorities
- Optimize the whole
- Manage internal and external dependencies
- Focus on quickly delivering minimal marketable features
- Clear feedback mechanism between business and development
- Creates alignment beyond single teams



Year(s): Set corporate goals and strategies

Quarter(s): Discover and create innovative product strategies from corporate goals

Month(s): Update Release Plans, Product Backlogs and Roadmaps

Week(s): Decompose features from Product Backlog into tasks and deliver working code

The 7 Principles of Lean

1. Eliminate Waste
2. Create Knowledge
3. Build Quality In
4. Defer Commitment
5. Optimize the Whole
6. Deliver Fast
7. Respect People

Source: Mary and Tom Poppendieck

Lean Principles

1. Eliminate Waste

•***Provide market and technical leadership*** - your company can be successful by producing innovative and technologically advanced products but you must understand what your customers value and you know what technology you're using can deliver

•***Create nothing but value*** - you have to be careful with all the processes you follow i.e. be sure that all of them are required and they are focused on creating value

•***Write less code*** - the more code you have the more tests you need thus it requires more work and if you're writing tests for features that are not needed you are simply wasting time

•***Have less meetings*** – the single most wasteful thing you can do is have unnecessary meetings

2. Create Knowledge

- ***Create design-build teams -*** leader of the development team has to listen to his/her members and ask smart questions encouraging them to look for the answers and to get back with encountered problems or invented solutions as soon as possible

- ***Maintain a culture of constant improvement -*** create environment in which people will be constantly improving what they are working on - they should know that they are not and should not be perfect - they always have a field to improve and they should do it

- ***Teach problem-solving methods -*** development team should behave like small research institute, they should establish hypotheses and conduct many rapid experiments in order to verify them

Principle – **Lean**

3. Build Quality In

- ***Synchronize*** - in order to achieve high quality in your software you should start worrying about it before you write single line of working code - don't wait with synchronization because it will hurt
- ***Automate*** - automate testing, building, installations, anything that is routine, but do it smartly, do it in a way people can improve the process and change anything they want without worrying that after the change is done the software will stop working
- ***Refactor*** - eliminate code duplication to ZERO - every time it shows up refactor the code, the tests, and the documentation to minimize the complexity. Resolve **technical debt** as it happens.

Principle – **Lean**

4. Defer Commitment

- ***Schedule Irreversible Decisions to the Last Responsible Moment*** - you should know where you want to go but you don't know the road very well, you will be discovering it day after day - the most important thing is to keep the right direction
- ***Break Dependencies*** - components should be coupled as loosely as possible to enable implementation in any order
- ***Maintain Options*** - develop multiple solutions for all critical decisions and see which one works best

80

Principle – **Lean**

5. Optimize the Whole

- ***Focus on the Entire Value Stream*** - focus on winning the whole race which is the software - don't optimize local inefficiencies, see the whole and optimize the whole organization
- ***Deliver a Complete Product*** - teams need to have great leaders as well as great engineers, sales, marketing specialists, secretaries, etc. - they together can deliver great final products to their customers

Principle – **Lean**

6. Deliver Fast

- ***Work in small batches*** - reduce projects size, shorten release cycles, stabilize work environment (listen to what your velocity tells you), repeat what's good and eradicate practices that creates obstacles
- ***Limit work to capacity*** - limit tasks queue to minimum (one or two iterations ahead is enough), don't be afraid of removing items from the queue - reject any work until you have an empty slot in your queue
- ***Focus on cycle time, not utilization*** - put in your queue small tasks that cannot clog the process for a long time - reduce cycle time and have fewer things to process in your queue

Principle – **Lean**

7. Respect People

- ***Train team leaders/supervisors*** - give team leaders the training, the guidance and some free space to implement lean thinking in their environment
- ***Move responsibility and decision making to the lowest possible level*** - let your people think and decide on their own - they know better how to implement difficult algorithms and apply state-of-the-art software frameworks
- ***Foster pride in workmanship*** - encourage passionate involvement of your team members to what and how they do

History of Lean

"Lean" is a term coined to describe the Toyota Production System. *The core tenets of Lean are:*

Lean Principles	Description
Value	What the customer wants/needs
Waste	Activities that don't add Value
Pull	Downstream activities "pull" work according to their capacity
Flow	Value is delivered continuously through minimal gaps between Value-added activities
Perfection	Inspect and adapt continuously

8 Wastes

Original Wastes	Software Development Wastes
Defects	Bugs
Overproduction	Extra/unneeded features
Waiting	Delays between activities
Not Using Available Skills	Pigeon-holing performers
Transportation	Handoffs between performers
Inventory	Partially done work
Motion	Task switching
Excess Processing	Reinventing the wheel, goldplating

The 7 Wastes

MUDA is the Japanese word for WASTE.

Seek it out and get rid!

An 8th waste is the wasted potential of people

Overproduction

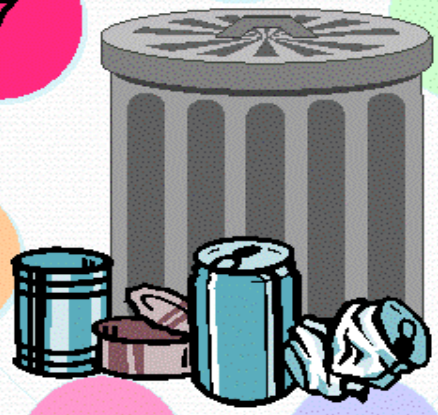
To produce sooner, faster or in greater quantities than customer demand.

Inventory

Raw material, work in progress or finished goods which is not having value added to it.

Over Processing

Processing beyond the standard required by the customer.



Rework

Non right first time. Repetition or correction of a process.

Waiting

People or parts that wait for a work cycle to be completed.

Transportation

Unnecessary movement of people or parts between processes.

Motion

Unnecessary movement of people, parts or machines within a process.

Common Sources of Waste

What are some common sources of waste?

- ✓ Chaotic work environment, constant interruptions
- ✓ Lack of available resources, resource bottlenecks
- ✓ Lack of clear prioritization of projects/tasks
- ✓ Poor communication across functional barriers
- ✓ Poorly defined product requirements
- ✓ Disruptive changes to product requirements
- ✓ Lack of early consideration of manufacturability
- ✓ Overdesigning, analysis paralysis, gold-plating
- ✓ Too many @!%&* meetings
- ✓ E-mail overload

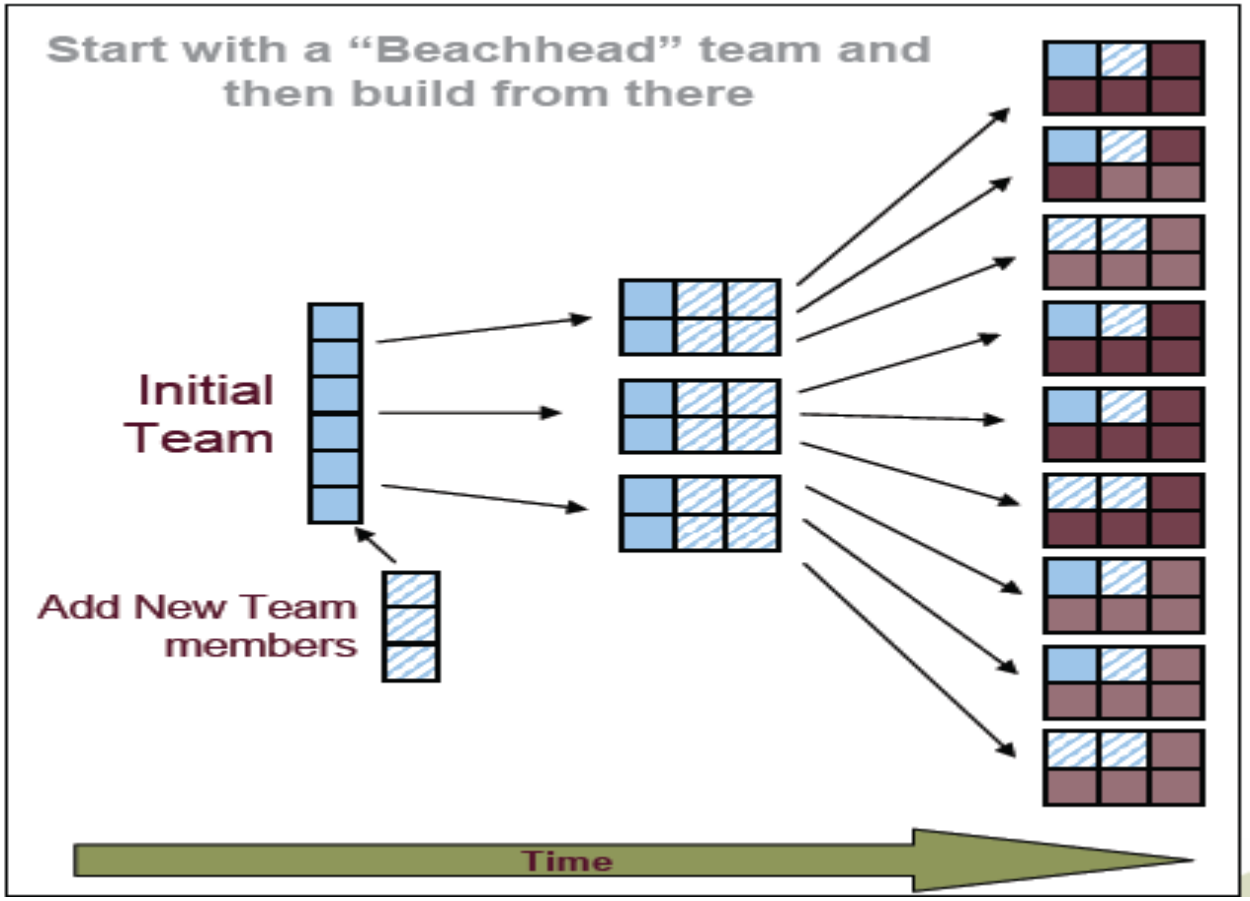
SCALING SCRUM

SCALING SCRUM

A Scrum team consists of 7 ± 2 people

- ✓ How to use Scrum in case of a mega project/program ?
- ✓ One solution is to scale SCRUM – SCRUM of SCRUMS teams
- ✓ Individual teams have their own Daily Standup meetings
- ✓ Scrum masters from individual teams have another meeting at an above tier called as Daily SCRUM of SCRUMS meeting
- ✓ These meetings are analogous to the Daily Scrum Meeting

SCALING SCRUM – Beachhead Model

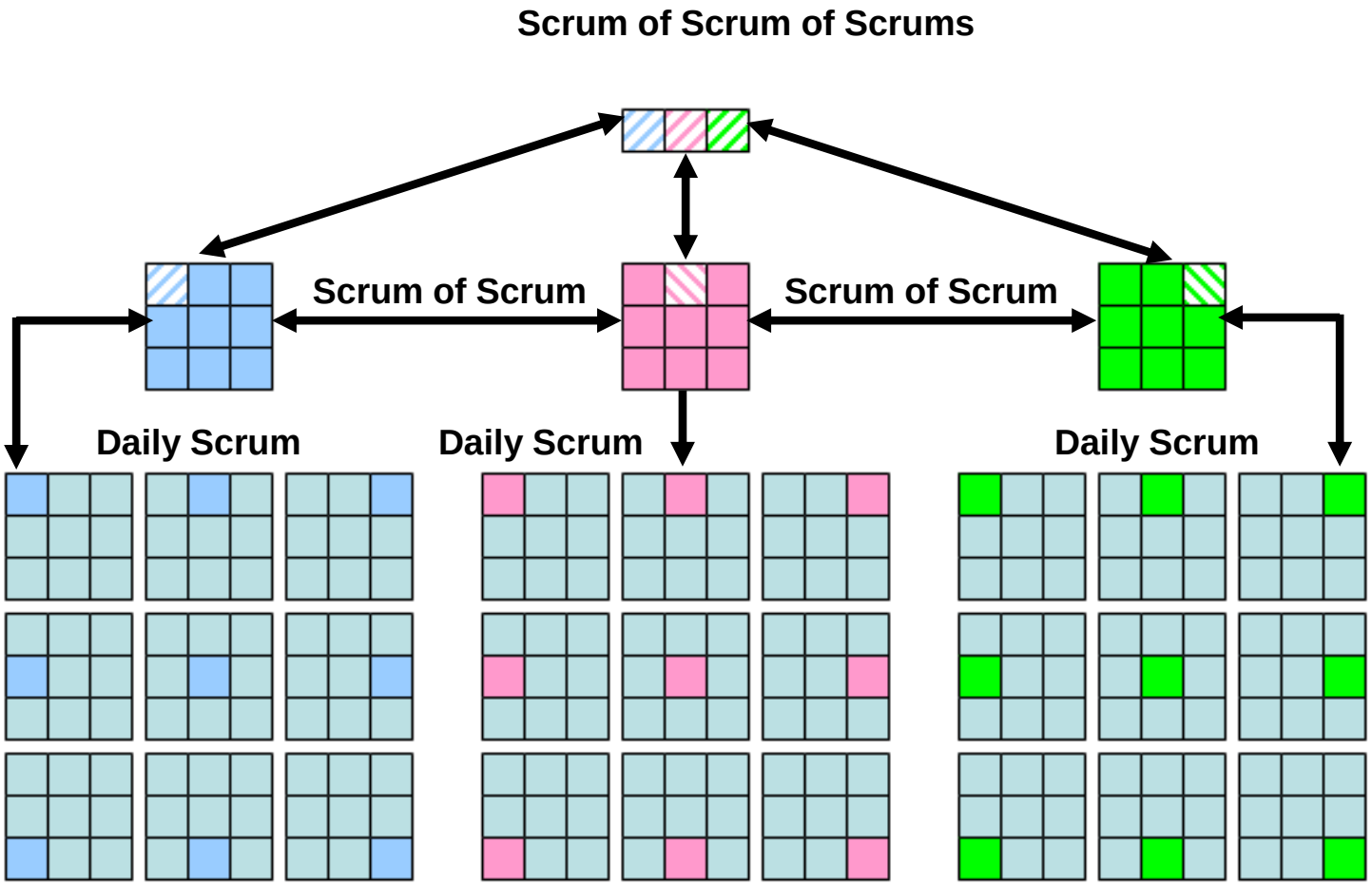


SCRUM of SCRUM (SoS) Meetings

Create a master team above the scrum teams

- ✓ Subdivide into Scrum teams and hold "scrum of scrums" meeting
- ✓ Objective is to allow these clusters of teams to discuss their work, focusing especially on areas of overlap and integration.
- ✓ Attended by a representative from each team
- ✓ Agenda is not restricted to the standard 3 question format
- ✓ More focus on issues of integration and overlap areas

SCRUM of SCRUM (SoS) Structure



SCRUM of SCRUM (SoS) - Challenges

In creating SCRUM of SCRUMS the traditional roles following roles is challenged

Product Owner

- ✓ Increase In Workload
- ✓ Present at all the meetings

Scrum Master

- ✓ More inter-team issues
- ✓ Dependencies

Scrum Team

- ✓ Less empowered
- ✓ Large or distributed

Distributes Team - Challenges

Co-location is ideal however not always possible

If the teams have to be distributed–minimize the communications gap

- ✓ Arrange the teams to match geographical and time zone
- ✓ Use Of Instant Messenger
- ✓ VOIP enables low cost voice communication
- ✓ Web Conferencing for team presentations
- ✓ Good quality phone conference facilities
- ✓ Face to Face meetings/working –swap team members
- ✓ Conduct "Daily Stand-up"meeting at the same times
- ✓ Ensure the teams have an integrated development infrastructure
- ✓ Common source control system and structure
- ✓ Ensure regular integration of all teams output

USER STORY

EPIC

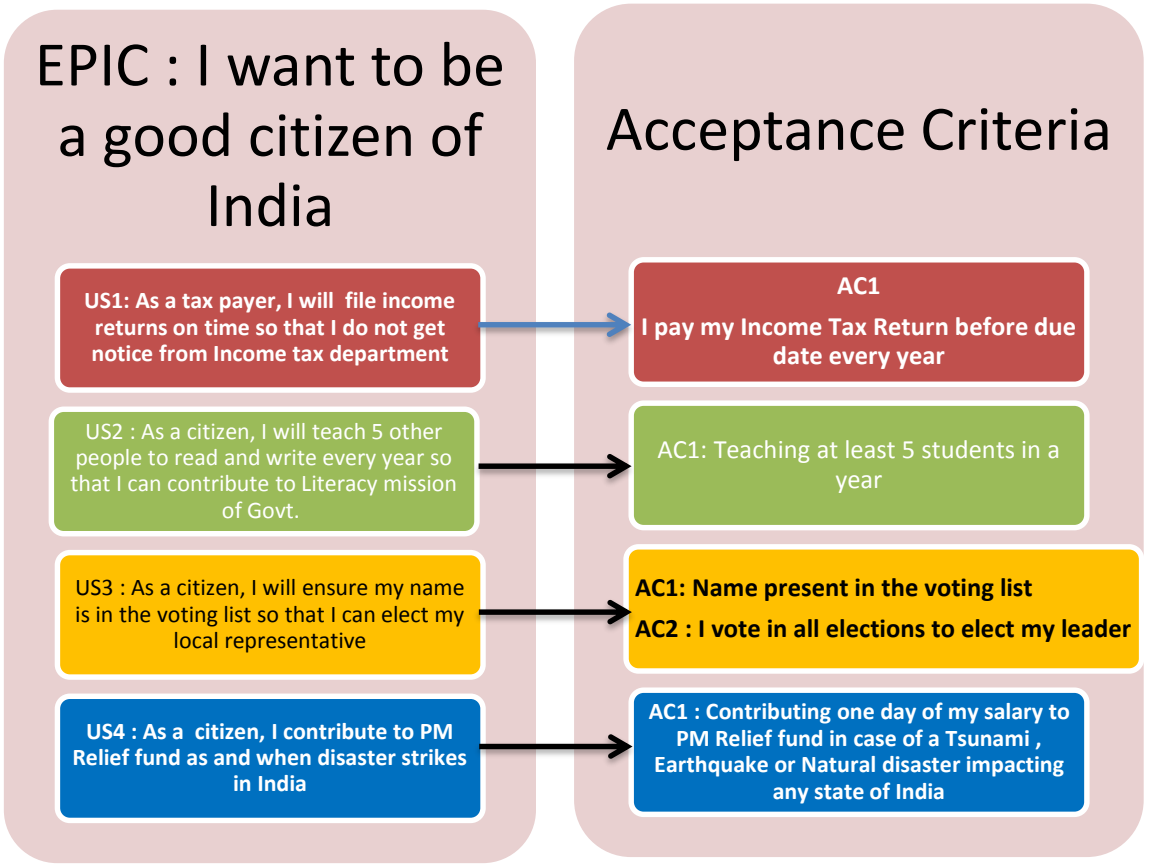
A epic is a large user story

There's no magic threshold at which we call a particular story an epic

---

EPIC gets broken down into multiple user stories

Each story has got acceptance criteria



96

USER STORY

Key Principles

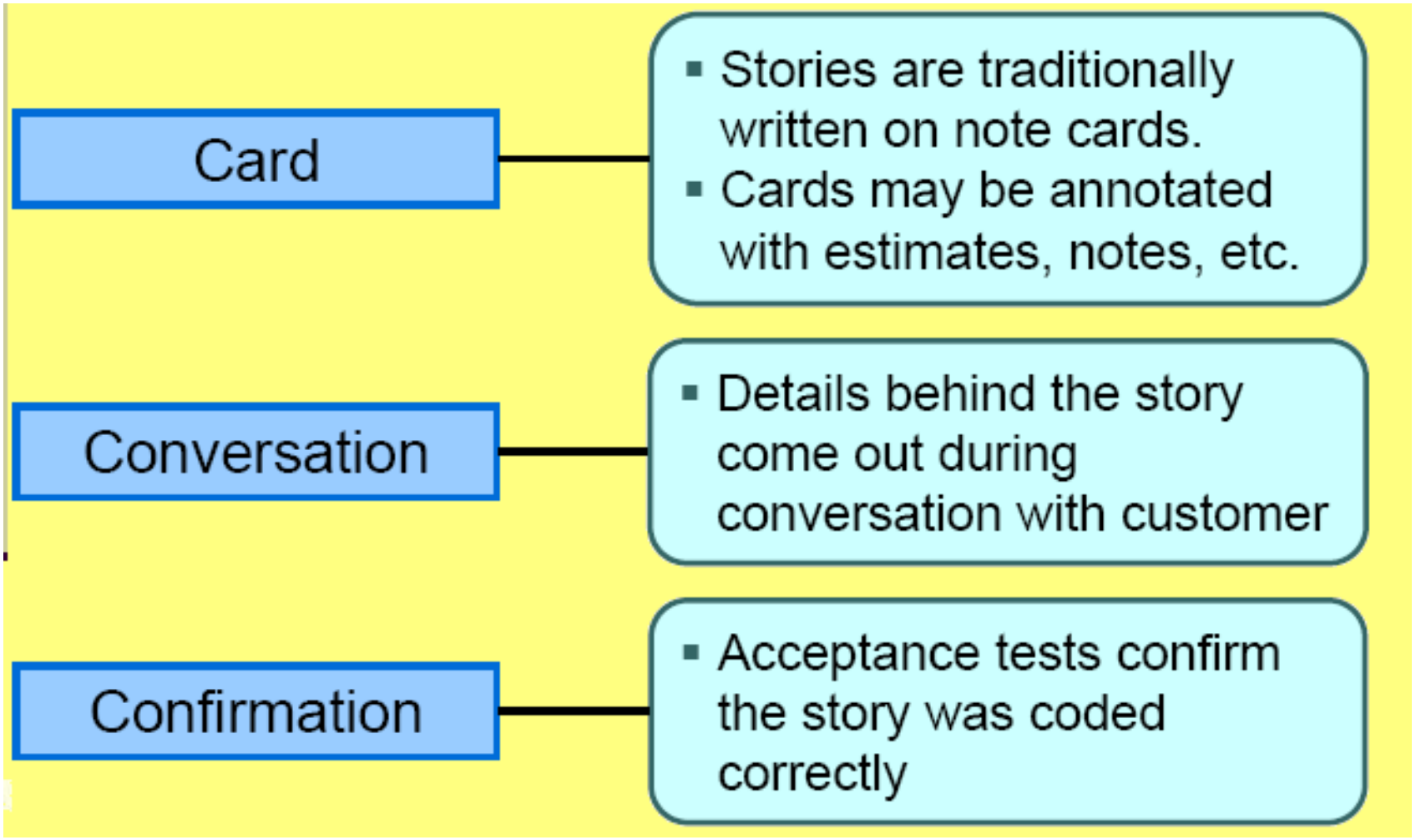
- ✓ Simple means to capture user requirements instead of lengthy requirement documents and are small enough to estimate
- ✓ Brief statement of intent which describes something that the system needs to do for the user
- ✓ Usually written by the customer/business person, testers assist in writing the same
- ✓ Keep It Short and Sweet … written on a card for ease in implementation
- ✓ Consists of 3C's Card, Conversation and Confirmation
- ✓ FORMAT: As a <Role>, I want <Goal/Desire> so that <Benefit>

User Stories – Sample Format

As a Marketing Manager, I would like the system to generate multiple product plans as an option for the customer

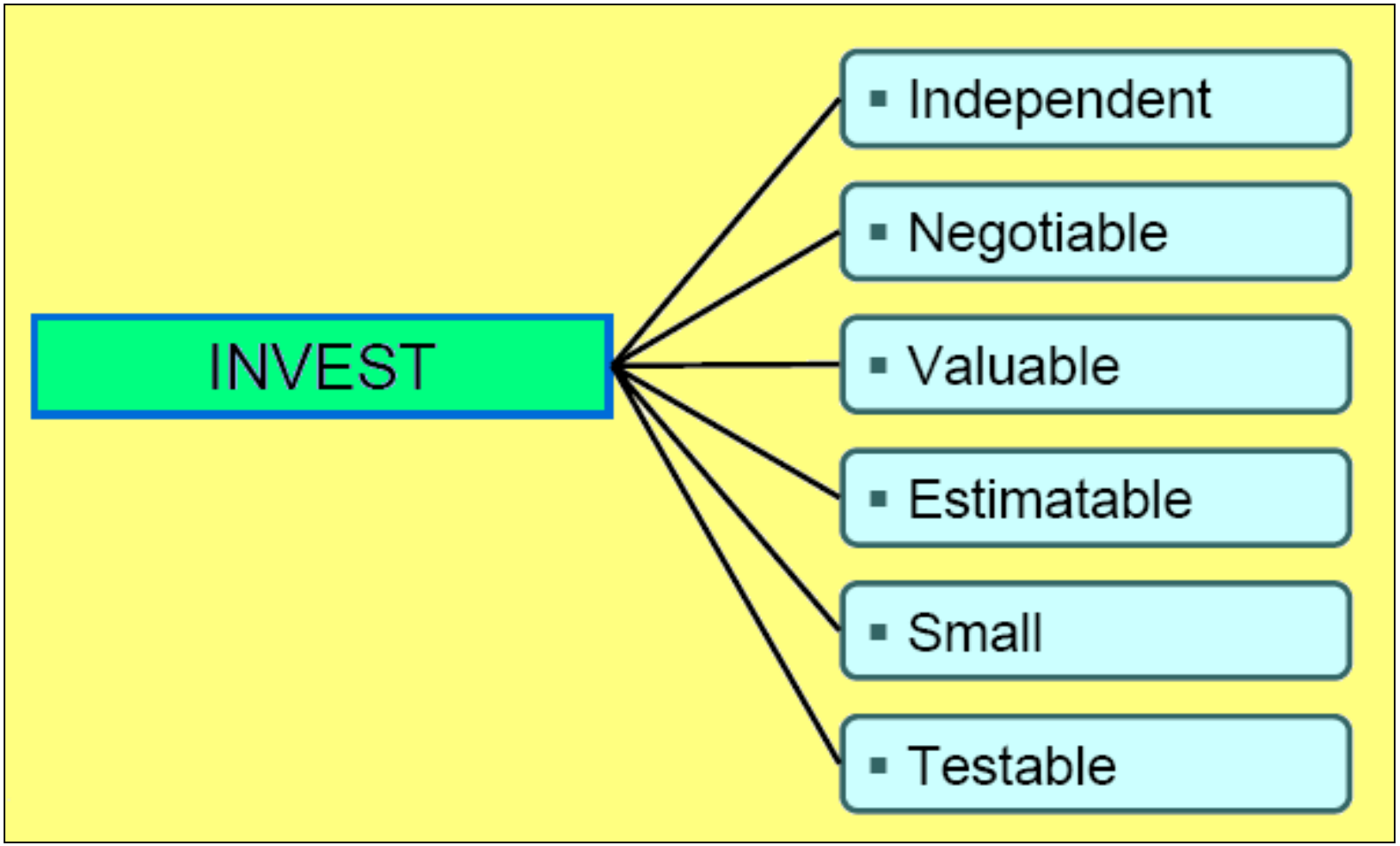
As a Finance professional, I would like the payroll to process salary for all employees

User Stories – 3C Model



User Stories – INVEST Principle

- To create a good user story always INVEST



PERSONA

- ✓ Persona defines an archetypical user of a system, an example of the kind of person who would interact with it
- ✓ Personas offer a great way to capture the users and the customers with their needs.
- ✓ They are fictional characters that have a name and picture; relevant characteristics such as a role, activities, behaviours, and attitudes; and a goal, which is the problem that has to be addressed or the benefit that should be provided
- ✓ Idea is that if you want to design an effective software, then it needs to be designed for a specific person

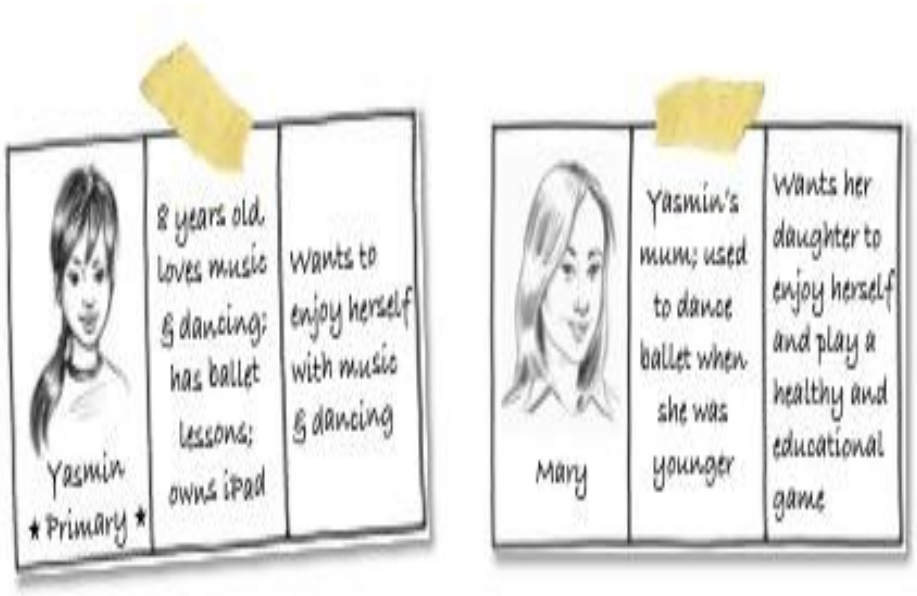
  e.g. For banking software , potential personas could be named Rajeev Deshpande and Ms Manisha Chopra

PERSONA To USER STORIES

1. Write Personas
2. Derive Epics From Personas
3. Decompose Epics To User Stories
4. Get The Stories Ready

PERSONA To USER STORIES

1. Write Personas



- Understand The Target users and customers
- Describe the characteristics exhibited by personas in relation to the proposed product
- It is important to know who the users are and what problem we want to solve else it's impossible to write the *right* stories

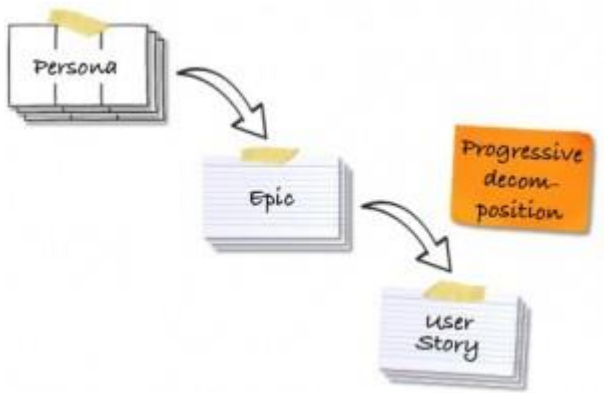
PERSONA To USER STORIES

2. Derive Epics From Personas

- Use the goals of the personas to identify the product functionality
- Document how the product addresses the personas' problems and benefits them
- Start with your primary persona and capture the functionality as epics, as coarse-grained, high-level stories
- Write all the epics necessary to meet the persona goals but keep them rough and sketchy at this stage

PERSONA To USER STORIES

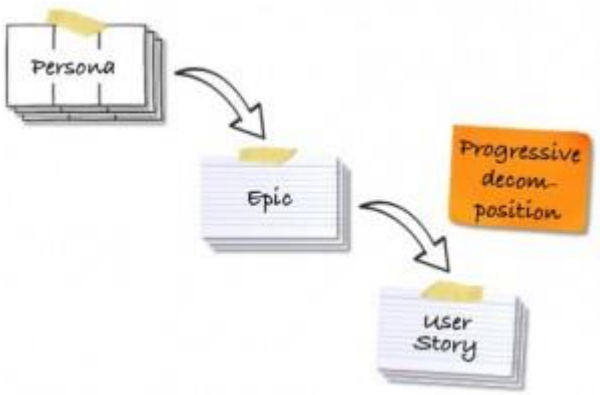
3. Decompose Epics Into Stories



- With a holistic but coarse-grained description of your product in place start progressively decomposing your epics
- Derive *just enough* user stories *just in time* for the next sprint
- Use your sprint goal or hypothesis to determine which epics to decompose and which stories to write

PERSONA To USER STORIES

4. Get The Stories Ready



- Before the development team starts working on the stories, check that each user story is *ready*: clear, feasible, and testable
- Check that the story meets INVEST criteria
- A story is feasible if it can be delivered in the next sprint according to the Definition of Done

Agile Methodologies

Vertical Slice cuts across all the layers. Horizontal layers focus on each individual component (e.g. Database, Persistence layer etc.)

Traditional Method : Build each horizontal layer separately sequentially and then integrate by stacking them

---

Horizontal layer development is not an effective way to deliver quick business value

---

Difficult to prioritize horizontal stories/requirements and potentially release the most important ones or the most required features first

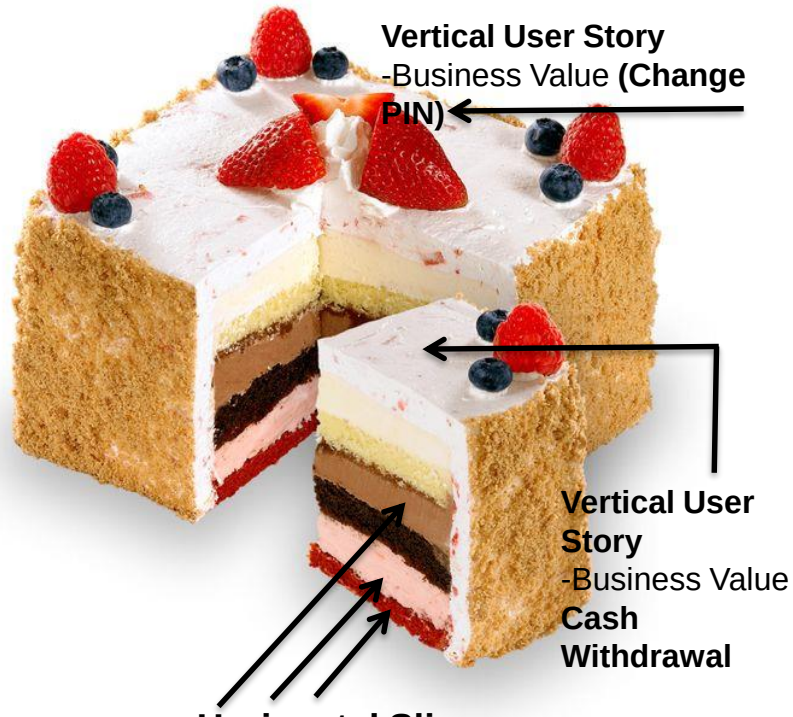
---

The feedback loop is long, which significantly increases the risk. The entire cake must be baked before the customer realizes that the cake they got is not the one they wanted

---

Focus on developing a vertical slice (Business Value) cutting across horizontal layers

---



107

PERSONA

Mr Rajeev Deshpande: Sr Vice President IT

Rajeev is aged 42 and is based out of Gurgaon. He has a busy schedule and works nearly 60-70 hrs a week. He is in top 5 earners in his industry and usually travels 15-18 days a month. He is a gadget freak and makes sure is up to date with technology in terms of s/w, mobile phones and tablets.

Ms Manisha Chopra: Socialite/ Page 3 Celebrity

Manisha is aged about 40 and comes from a family of industrialists. She is usually seen in parties and events. She has a dedicated assistant who takes care of her logistics.

She is not a tech savvy but flaunts new mobiles to her friends without knowing the functionality. She travels 3 days in a month in remote villages for CSR activity.

Let us see a recap of User Story

Prioritization Techniques

Agile Methodologies

The **MoSCoW** method is a prioritization technique used in multiple management fields to reach a consensus on what's more important to stakeholders and customers.

Requirements are thus classified as:

Must have — these are critical and must be included into the product. If even one isn't included, the release is considered a failure. These can be downgraded if there's agreement among stakeholders.

Should have — these requirements are important but not crucial for the release. They're the first level of "Nice to have", and generally share the importance of *MUST* requirements, without being so time-sensitive.

Could have — these requirements are desirable but not necessary for the release. They're usually low-cost enhancements to the product. Due to their lower importance, they're the second level of "Nice to have" features.

Won't have — these are considered to be the least-critical or even not aligned with the product strategy. They are to be definitely dropped or to be reconsidered for future releases.

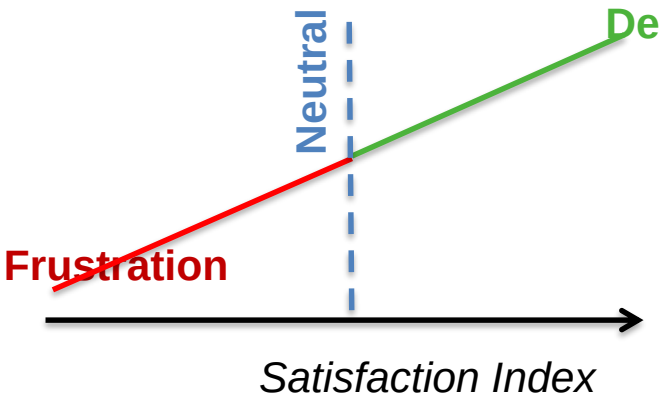
Leave Application	
M	UI to apply Leave
S	Leave Reports for a Manager
C	Leave App on Smart Phone
W	Apply Leave through Whats App /SMS

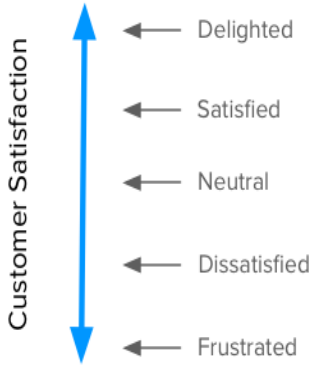
KANO Model

Developed by Prof Noriaki Kano, a set of ideas and techniques that help us determine our customers' (and prospects') satisfaction with product features /functionality. There 2 dimensions to measure Satisfaction and Functionality/Features

Customers' **Satisfaction** with our product's features depends on the **level of** that is provided (how much or how are they implemented)

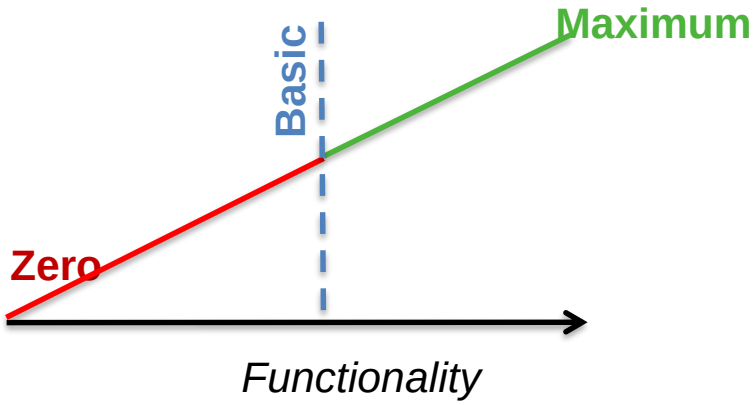
Plot on Y - Axis

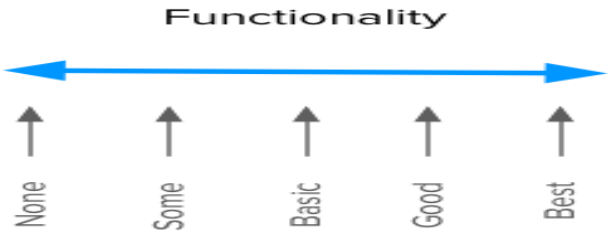




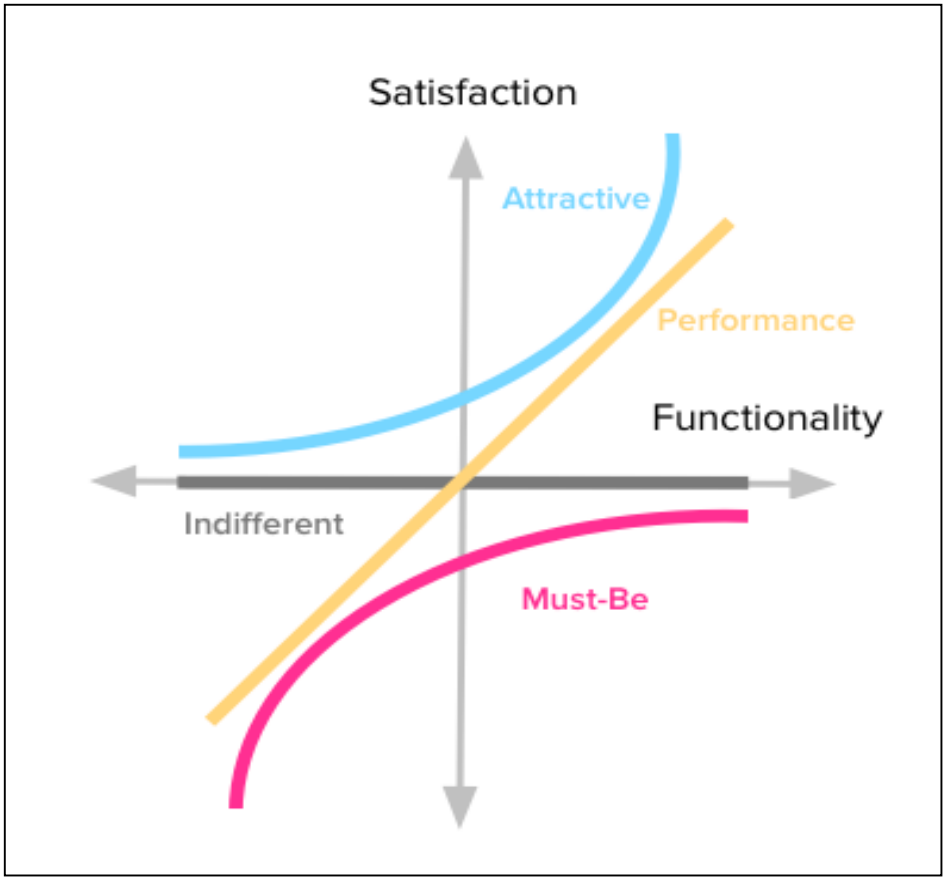
Product Functionality/Features that is provided (how much or how well they're implemented)

Plot on X-Axis





Agile Methodologies



Functionality / Features Categories	
Must Be	**Basic : Bare minimum expected functionality**
Performance	**More The Better : The more we provide, the more satisfied our customers are**
Attractive	**Delighters/Exciters : Adds/Enhances Value To The Product**
Indifferent	**Irrelevant As of Now :** Presence (or absence) doesn't make a real difference in reaction towards the product

Relative Weighting Technique

Relative Weighting Technique was created by Karl Wiegers. This technique is based on the premise that the features that provide the highest benefits after adjustment for costs, risks, and penalties, should have the highest priority.

Key aspects of this techniques are:

- A feature's priority is directly proportional to the value it provides, and inversely proportional to its cost and the technical risk associated with its implementation.
- Each category uses a scale of 1 – 9.
- Benefits reflect the value a feature will provide, while penalties reflect the negative effect a customer will experience if the feature is not included.
- Further, risks reflect the challenges of implementing the feature, and costs reflect the actual costs of implementing the feature.

Relative Weighting Technique

Each feature is prioritized based on its relative weighting for Benefits, Penalties, Costs, and Risks. Each feature uses a relative scale of 1–9 to determine its rating.

Feature	Benefit	Penalty	Total Value	Value %	Relative Cost	Cost %	Relative Risk	Risk %	Priority
Feature 1	8	2	10	27%	2	20%	3	17%	.73
Feature 2	8	3	11	30%	4	40%	6	33%	.39
Feature 3	7	4	11	30%	3	30%	7	39%	.43
Feature 4	3	2	5	13%	1	10%	2	11%	.62
TOTAL	**26**	**11**	**37**	**100%**	**10**	**100%**	**18**	**100%**	

$$\text{Priority} = \frac{\text{Value\%}}{\text{Cost\%} + \text{Risk\%}}$$

Estimation - Agile Projects

What is Estimation?

Estimation

"The process of finding a rough guess or an approximation, which is a value that is usable for some purpose even if input data may be incomplete, uncertain, or unstable"

Using Estimation we get a
-Rough idea about the project size
-Understand the overall Project feasibility
-Get a Baseline to measure performance
-Help us separate Risks from estimation
-Get everyone's buy-in
-Highlight any unstated assumptions, risks & uncertainties

An "exact/accurate estimate" is an oxymoron ("credible estimate" is a better phrase)

Estimate how long will it take you to place to another place

- On what basis did you do that?
- Experience right?
- Likely as an "average" probability
- For most software projects there is no such 'average'
- Most estimations are off by 25-100%

Estimation Challenges

The following are the key challenges faced by teams during the estimation process :-

- Inexperienced people involved in estimation
- Lack of training
- Lack of historical data
- Absence of tools
- Over-estimation and under-estimation
- Inadequate time allocated for estimation
- Lack of review of the estimation
- Business Pressures to ignore genuine estimation
- Less priority for estimation

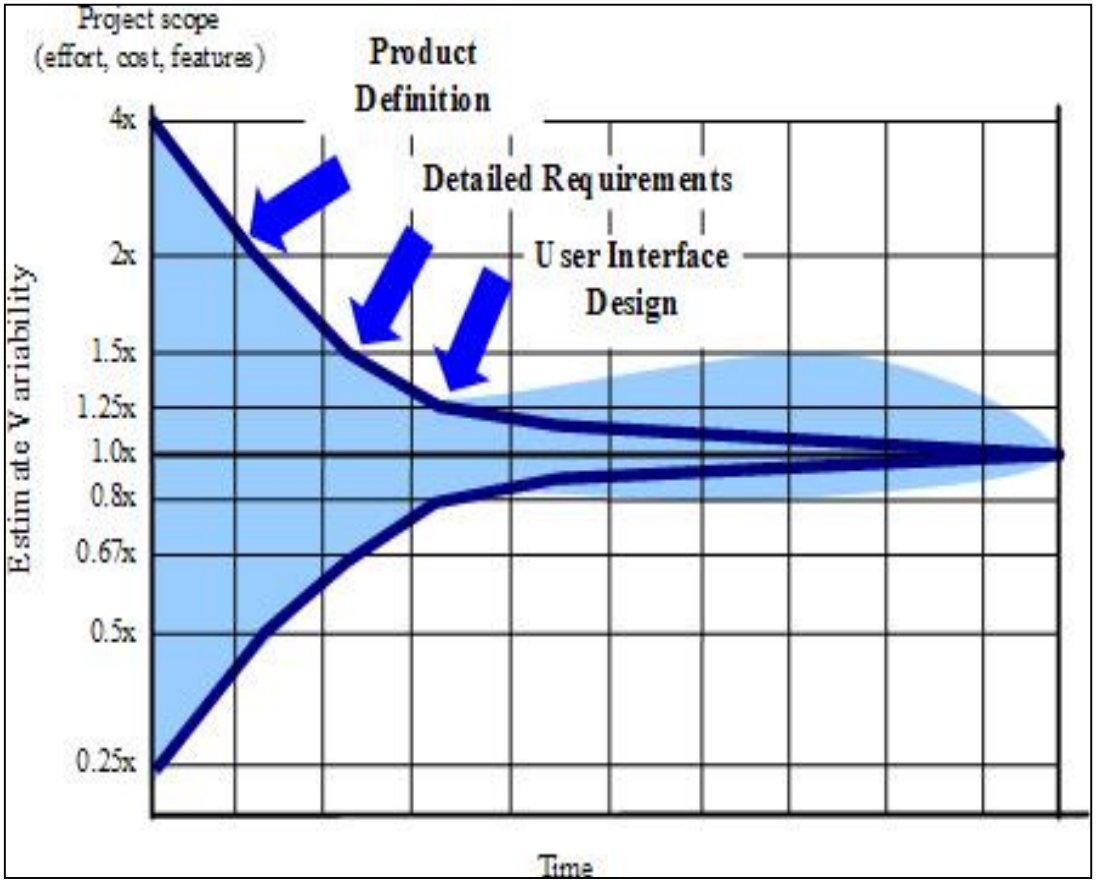
Issues - Under Estimation

- Quality Issues
- Inability to meet deadlines
- Team Burnout / Morale
- Impact on Profitability

Issues- Over Estimation

- Project Loss
- Conservative estimates which are guaranteeing a success is a winning probability of 0
- Parkinson's Law / Students Syndrome
- Feature/ Scope creep
- "Double-Padding":

Estimation Challenges



Initial Phases :

- Unclear on the specific details of the nature of the software to be built,
- Details of specific requirements /Solution unknown
- No clarity on the project plan, staffing, and other project variables are unclear.
- Variation can be from 4x on the either side

Progressive Elaboration

- As we progress, the cone of uncertainty narrows down

Estimation – Traditional Methods

Traditional Estimation methods used for s/w development are:

KLOC / LOC

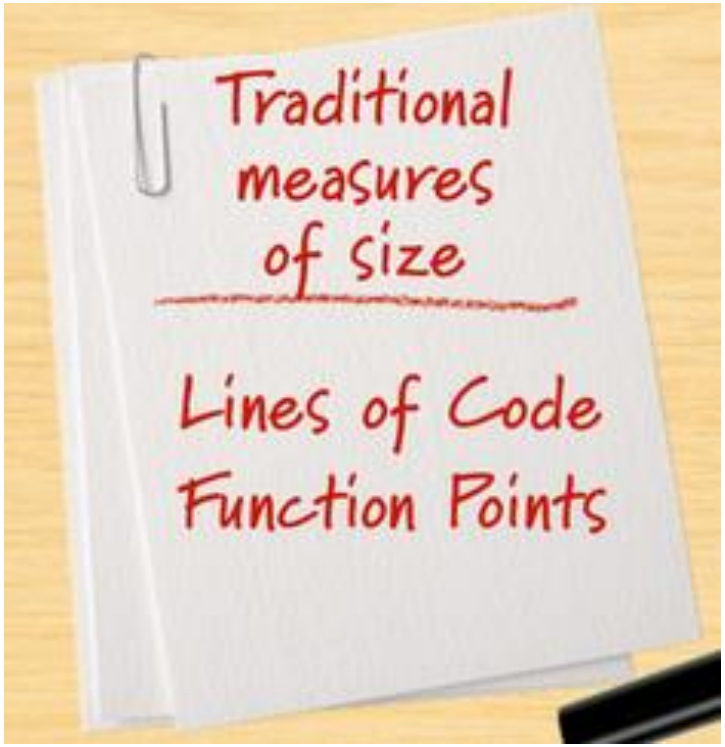
- Measuring Lines Of Code

Function Point

- Measuring software size

Heuristic

- Historical Project Data



Estimation – Agile Projects

Estimation methods used in Agile Projects include

Ideal Days

- Measuring Ideal Days to complete work without break or interuptions
- Assuming everything what is required for the story is available

Story Points

- Like FP calculates story points for a work

Agile Estimation – Key Characteristics

Agile Estimates are relative estimates

- ✓ Advantage is that it is easier to reach a consensus on the relative size within the team
- ✓ Main disadvantage of this method is that its difficult to explain and use
- ✓ Story points are abstract representation of size, which includes complexity, effort, risk (constraints) etc.

Estimation Scales predominantly used include

- ✓ Fibonacci Scale: 1,2,3,5,8,13,…
- ✓ Linear Scale: 1,2,3,4,5,6,7,8,9,10
- ✓ T-Shirt Size: XS, S, M, L, XL

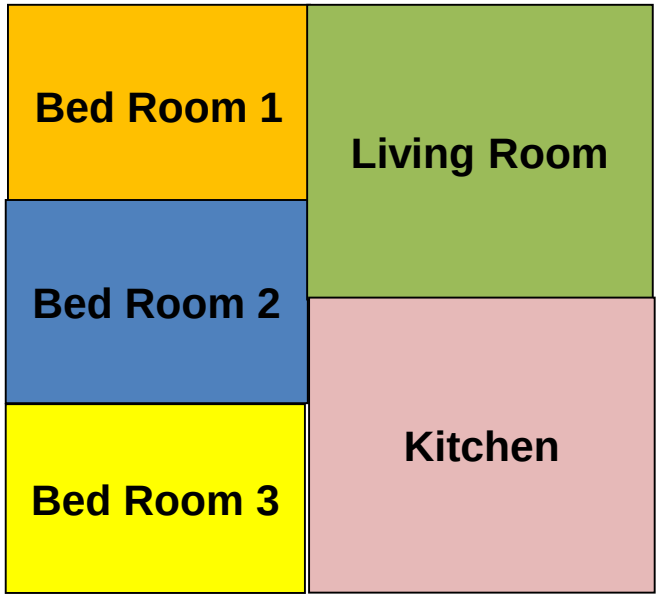
Relative Estimates – Key Features

Key features of relative estimates are

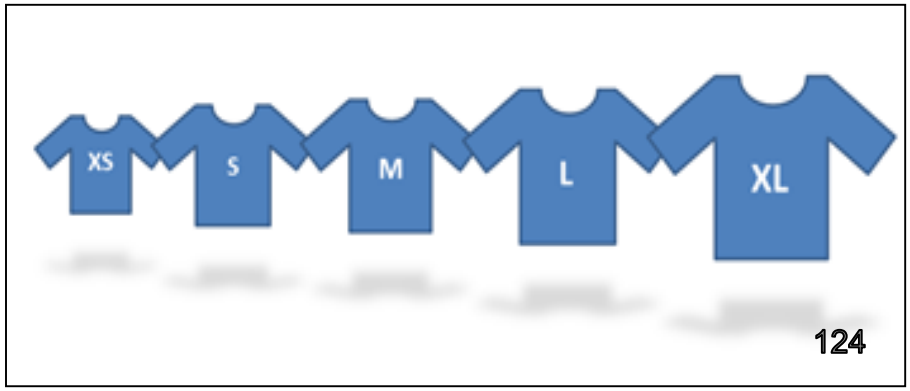
- ✓ Points do not have units and are relative
- ✓ Points cannot be converted easily into duration (hours, days etc.)
- ✓ How to then convert the points (abstract) into duration (tangible) ??
- ✓ Comparing points between projects or teams has very less relevance and use (although it can be done)
- ✓ The whole team is involved in estimation

Agile Estimation – Size

How long will it take to paint this apartment?



- ✓ Determine what is the size of work
- ✓ Size depends on dimensions of the rooms, obstructions etc.
- ✓ So first one needs to know how large it is before answering "How long will it take"



Agile Estimation – Size

First step is to identify the smallest size component and use that as a baseline for further components

Room	Points/Size
Bed Room 1	2
Bed Room 2	2
Bed Room 3	2
Living Room	3
Kitchen	5

- ✓ Kitchen & Bedroom are of same size, so why is kitchen given size of 4 ?
- ✓ Reason: Kitchen would generally be complex to paint due to cabinets, appliances etc.

Velocity

- ✓ Velocity is the measure of how much work the team can complete in a given amount of time

- ✓ In agile terms it's the amount of work a team can complete in an iteration

Velocity can be calculated by

- ✓ Using historical data of the team
- ✓ Running along the iteration
- ✓ Making a forecast

Estimation Conversions

Iteration #	Backlog Item	Estimated Size
1	Install the test environment	10
1	Develop ATM functionality	30
1	Deploy the ATM machine	30
2	Develop Cash Deposit functionality	30
2	Regression testing	20
2	Linking Of ATM's	30

Iteration 1 Size: 70

Iteration 2 Size: 80

- ✓ Based on the above data, the team has a approx. velocity of 75 story points/iteration

Velocity Calculation

Some rough indicators to convert points to duration are

Size	Ideal Days
X Small	0.5 To 1
Small	1 To 3
Medium	3 To 5
Large	5 To 7
X Large	7 To 9

Size	Fibonacci Story Point
X Small	1
Small	2
Medium	3
Large	5
X Large	8

Estimation Technique - Planning Poker

- ✓ One of the most popular and widely used estimation technique for agile projects
- ✓ Entire team conducts this exercise
- ✓ The card values are based on the modified Fibonacci series
- ✓ Similar to the Wide Band Delphi method
- ✓ www.planningpoker.com

Estimation Technique - Planning Poker

- ✓ All participants read the product backlog item
- ✓ Each participant then individually estimates the size of the item (using the principle of relative estimates)
- ✓ The leader of the meeting then ask everyone to show their cards
- ✓ If estimates vary significantly, the participants with extreme value cards are asked to explain the rationale
- ✓ Repeat the above up to 3 times or to a point where the estimates are converged. Pick up a number which everyone can live with

Estimation Technique - Planning Poker

Planning Poker®–an example

Estimator	Round 1	Round 2
Ana	5	8
Trond	5	8
Maria	8	8
Johannes	20	13

Estimation Technique - Planning Poker

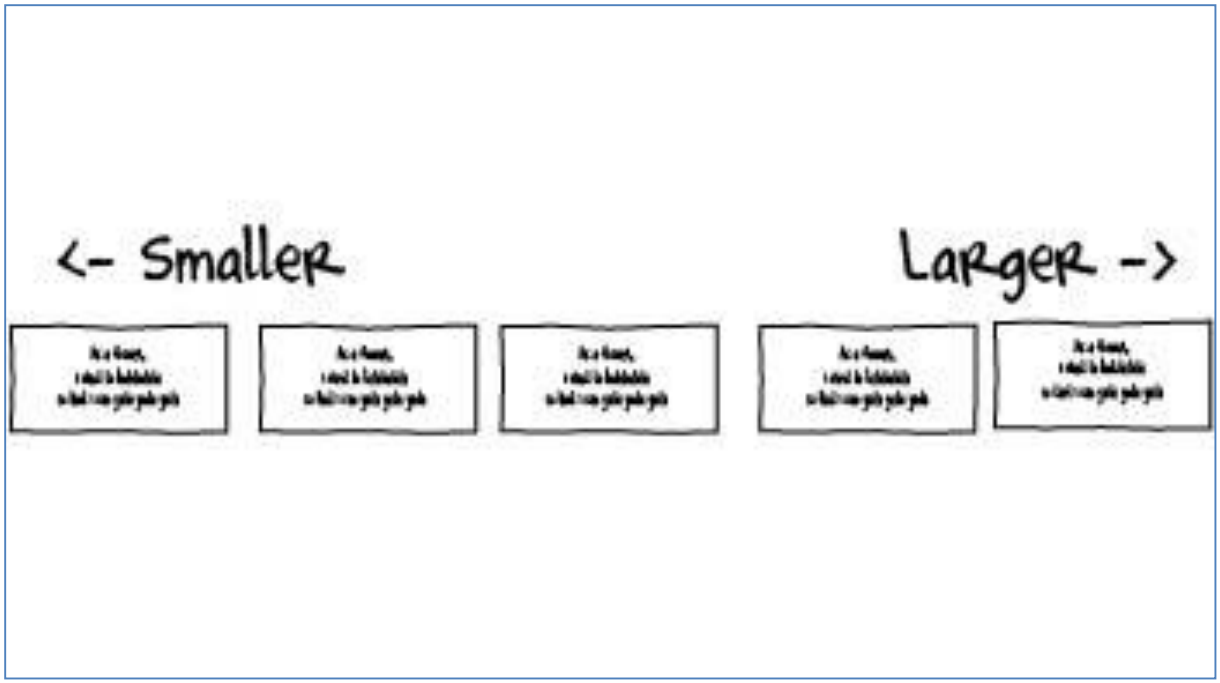
Success Factors of Planning Poker

- ✓ Planning poker brings together multiple expert opinions to do the estimating. As these experts form a cross-functional team from all disciplines on a software project they are better suited to the estimation task than anyone else

- ✓ A lively dialogue ensues during planning poker and estimators are called upon by their to justify their estimates. This has been found to improve the accuracy of the estimate,

- ✓ Group discussion is the basis of planning poker and those discussions lead to an averaging of sorts of the individual estimates

Estimation Technique – Team Estimation Technique

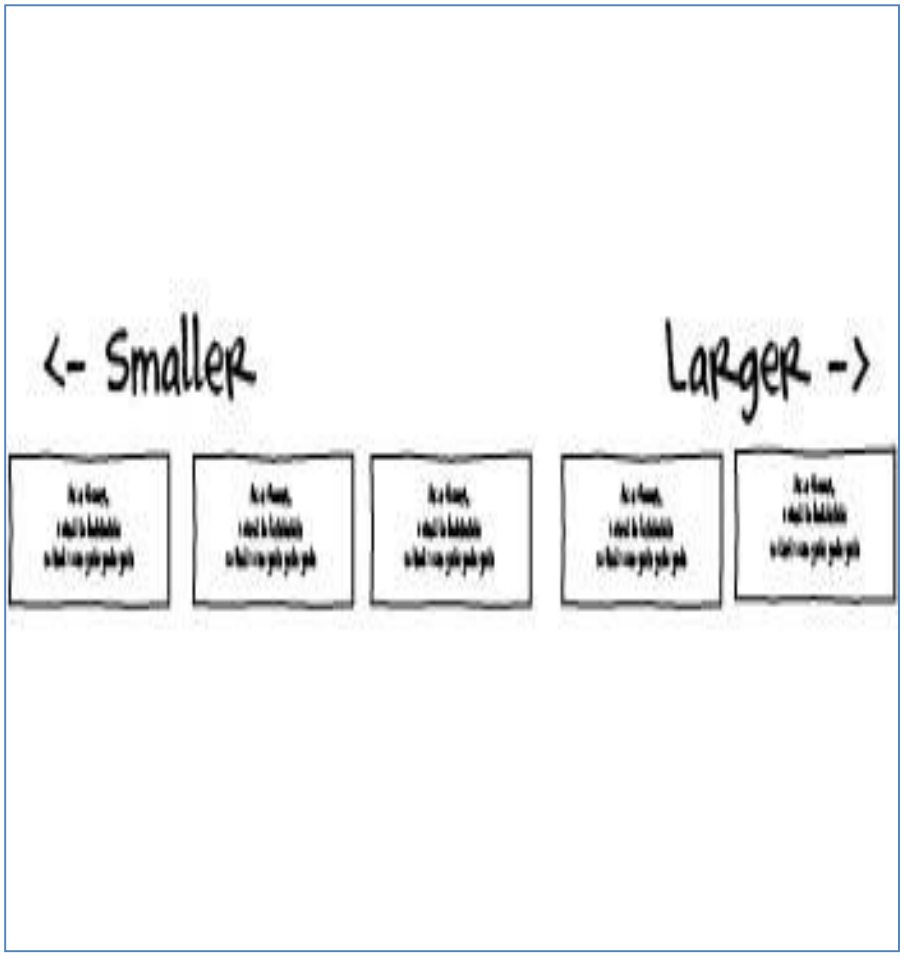


The Team Estimation Game is a technique to get a scrum team up-and-running with useful estimates.

It is a game used to :-

- Accomplishes valuable work
- Assigning story point estimates to user stories
- Estimate stories quickly (Approx. 40-60 stories Per Hour)

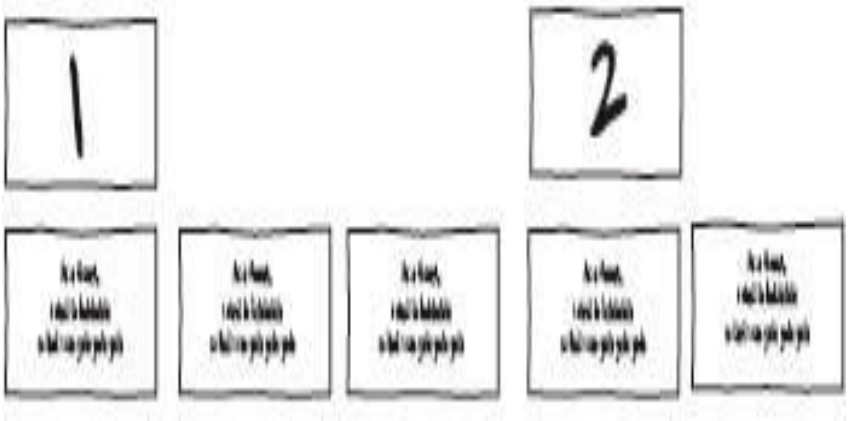
Estimation Technique – Team Estimation Technique



Step 1 : The Big Line Up

- Entire Team assembles in the front of the wall in a circle
- Wall has 2 sections Smaller and Larger
- Scrum Master takes about 60 cards in hand. He/She reads aloud the story and places in on the wall
- The cards are then handed to the second person.
- The second person picks another story reads it aloud and places it relative to the first story
- The cycle continues till all decks are over
- Multiple movements of stories happen

Estimation Technique – Team Estimation Technique

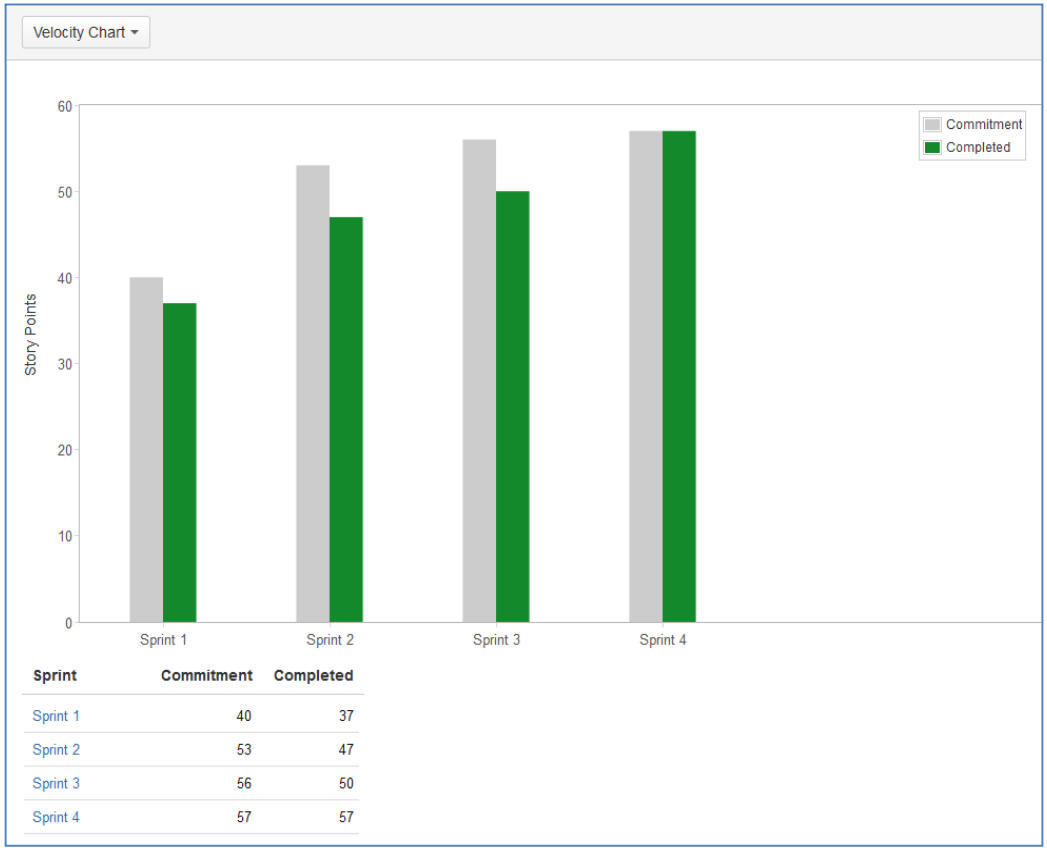


Step 2 : The Numbers

- Post sorting stories teams start assigning story points to the stories
- This again goes in a round robin fashion

Agile Metrics

Agile Methodologies



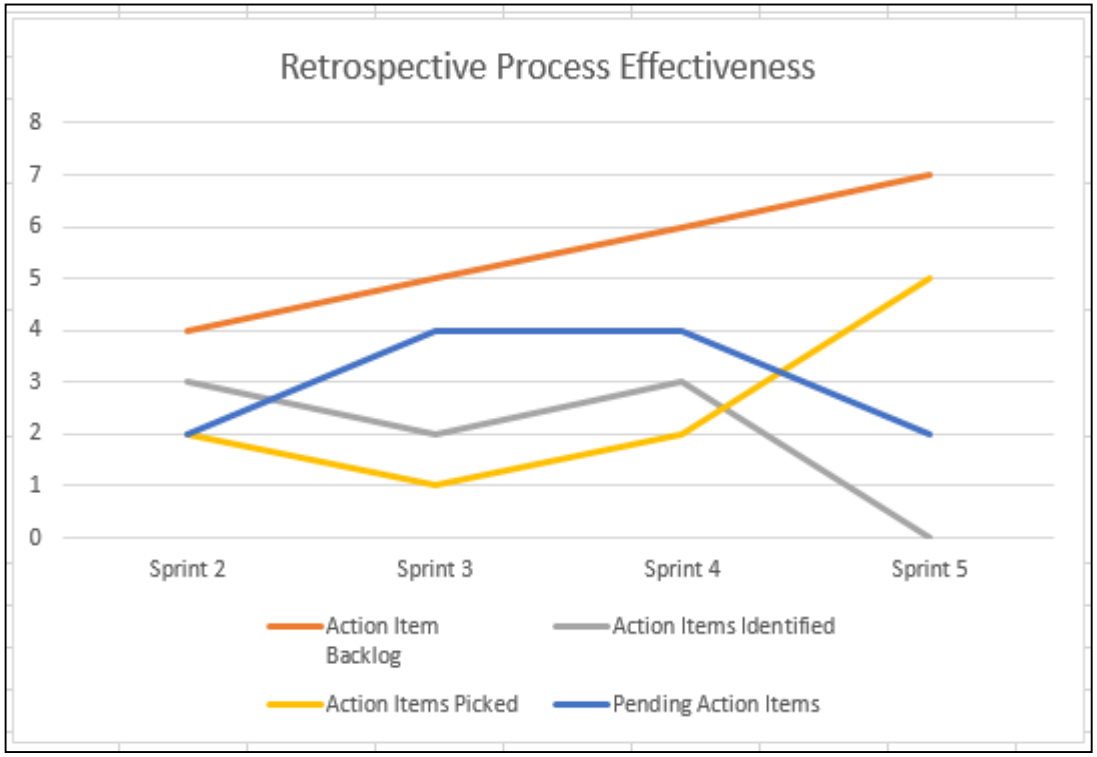
Sprint	Commitment	Completed
Sprint 1	40	37
Sprint 2	53	47
Sprint 3	56	50
Sprint 4	57	57

Velocity is the volume of work completed in a sprint by a team

Typically, this is measured as story points accomplished per sprint.

This measure is sometimes called "yesterday's weather" by agile practitioners

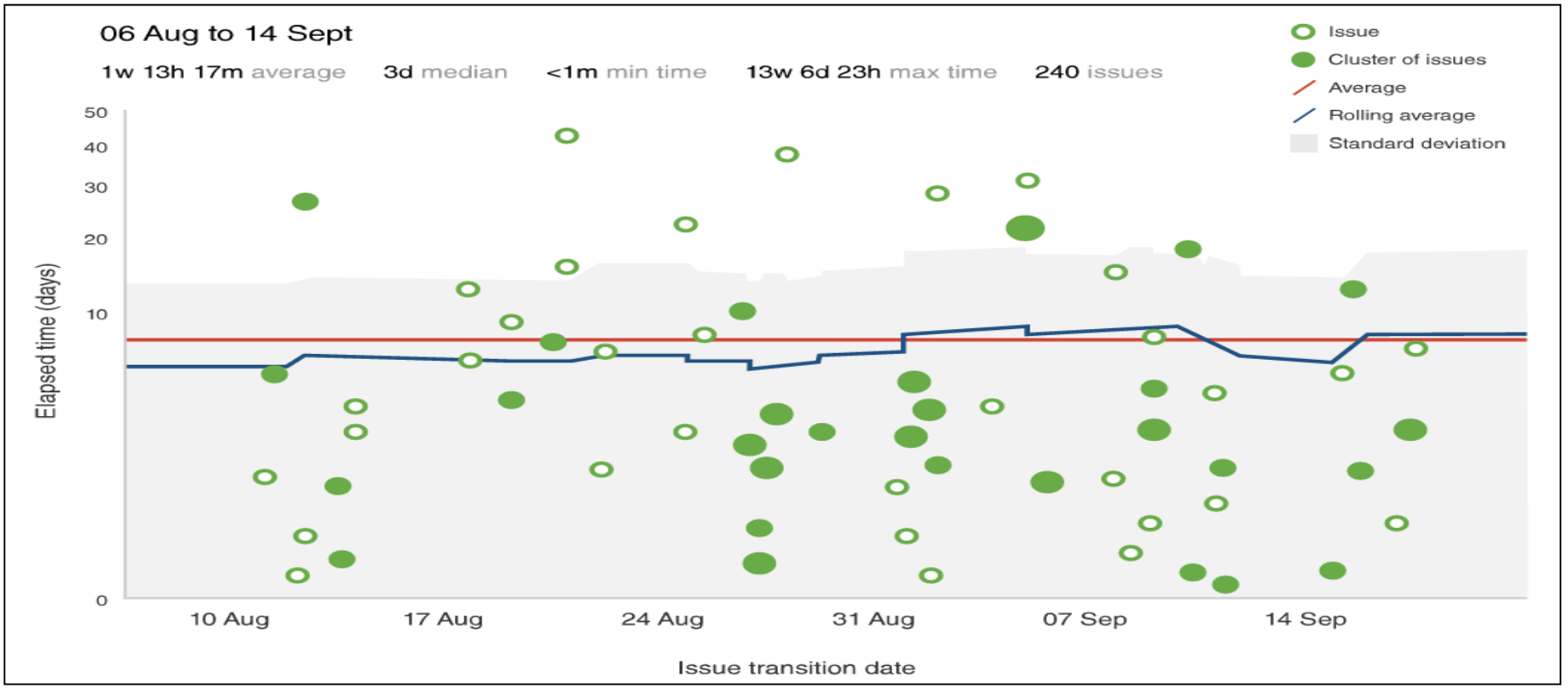
A team's recent velocity can be useful in helping to predict how much work can be completed by the team in a future sprint.



Retrospective Process Improvement

The scrum team's ability to revise its development process to make it more effective and enjoyable for the next sprint.

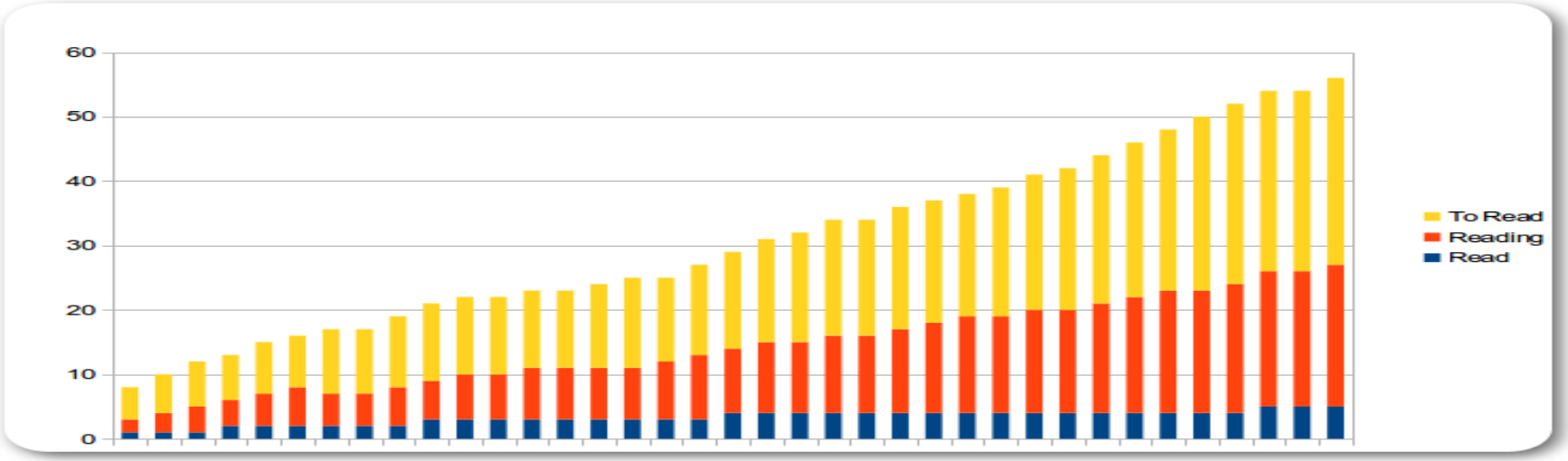
Parameter	Sprint 2	Sprint 3	Sprint 4	Sprint 5
Action Item Backlog	4	5	6	7
Action Items Identified	3	2	3	0
Action Items Picked	2	1	2	5
Pending Action Items	2	4	4	2



Control Chart

Control charts focus on the cycle time of individual issues–the total time from "in progress" to "done".

Teams with *shorter* cycle times are likely to have higher throughput, and teams with consistent cycle times across many issues are more predictable in delivering work.

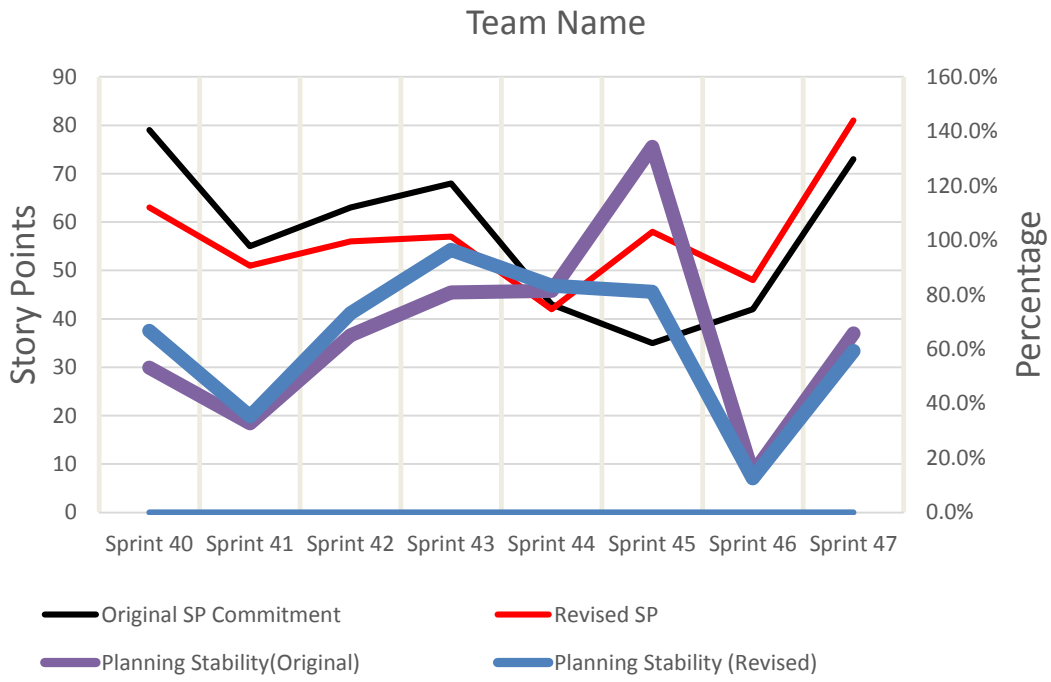


Cumulative Flow Diagram (CFD)

CFD is an area chart that shows the various statuses of work items for a particular time interval.
-Horizontal x-axis in a CFD indicates time
-Vertical y-axis indicates cards (issues)

Each colored area of the chart equates to a workflow status (i.e. a column on your board).

A CFD can be useful for identifying bottlenecks. If your chart contains an area that is widening vertically over time, the column that equates to the widening area will generally be a bottleneck.



Original Story Points (OSP) Story Points committed by team during Sprint Planning	Left Y Axis
Revised Story Points (SP) Story Points revised by team post sprint planning	Left Y Axis
Planning Stability (Original) Ratio of Actual Story points completed versus Original Story Points	Right Y Axis
Planning Stability (Revised) Ratio of Actual Story points completed versus Revised Story Points	Right Y Axis

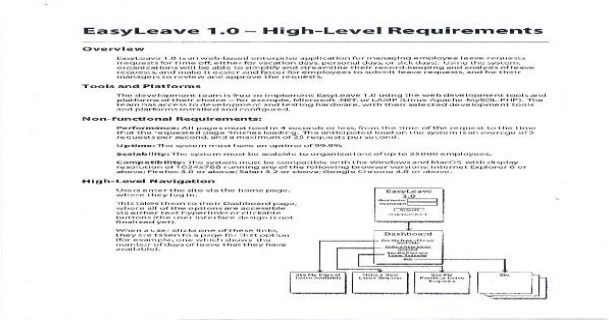
Scope Adherence Metrics

Let us see a Planning Poker in Action

Exercise – Easy Leave

Case Study – Develop a Leave Application Module

- ✓ Works in teams to complete this exercise
- ✓ Complete at least stories from 1- 10
- ✓ Teams to come up and present their estimates
- ✓ Use Planning Poker cards to arrive at estimates


EasyLeave 1.0 – High-Level Requirements

Overview

Tools and Platforms

Non-functional Requirements:

High-Level Navigation

Information Radiators

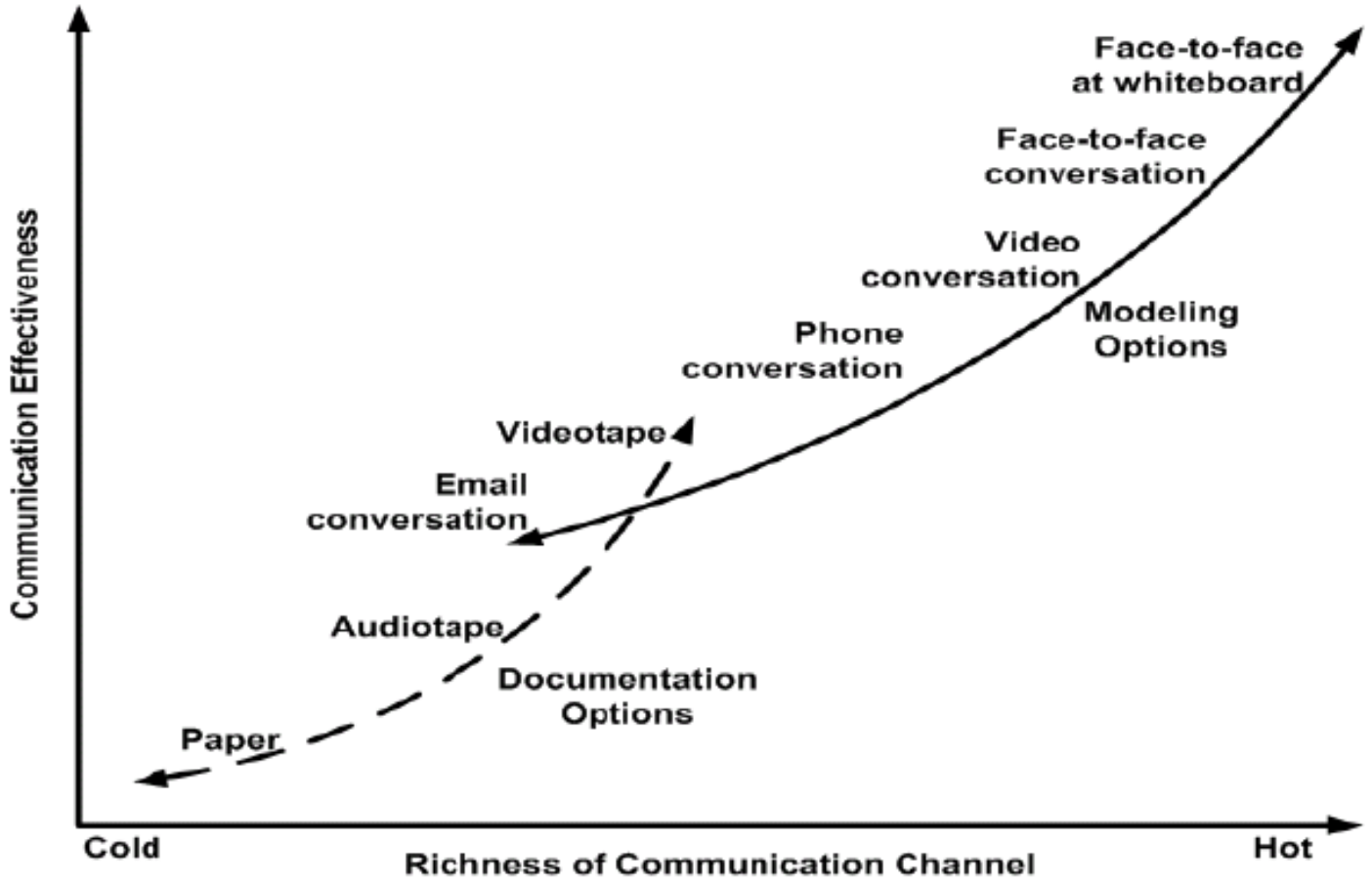
Information Radiators, also known as “Big Visible Charts”, are an effective way to communicate project status, issues, or metrics without too much effort.

The premise is that these displays make critical, changing information about a project accessible to anyone with enough ambition to walk over to the team area and take a look.

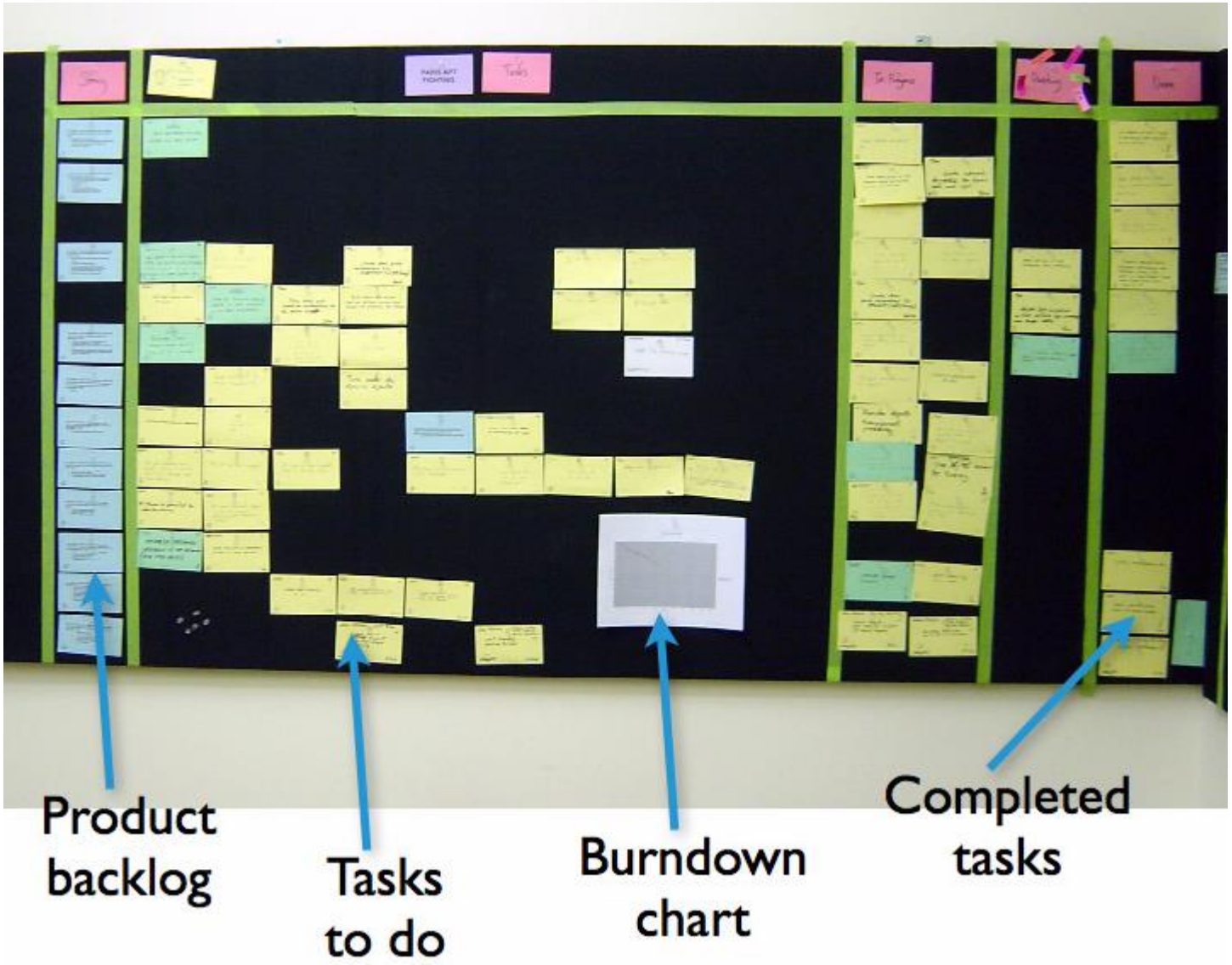
Information Radiators

- ✓ Task Board, also called as information radiator, is the primary communication tool in Agile projects
- ✓ Task board visually represents the work that is being done by the team
- ✓ Task board is a living entity and needs to be updated periodically
- ✓ Task board needs to be placed at a central location so that every team member can have a look at it.
- ✓ Task boards consist of sections as per the workflow status with the backlog items placed in the respective sections (using post it slips)

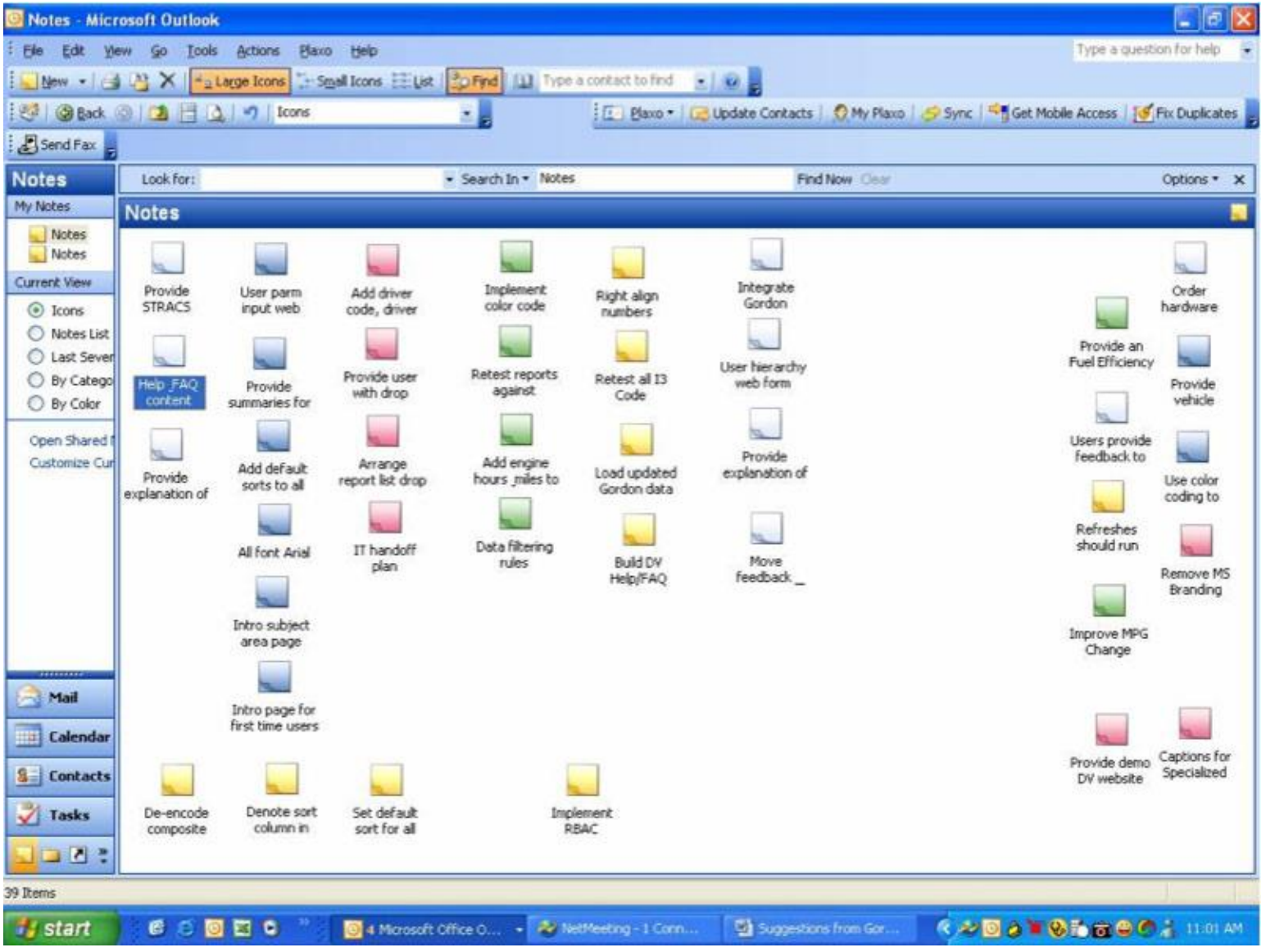
Agile Communication Channels



Agile Tools


Product backlog
Tasks to do
Burndown chart
Completed tasks

Agile Tools


Notes - Microsoft Outlook
File Edit View Go Tools Actions Plaxo Help
Type a question for help
New Large Icons Small Icons List Find Type a contact to find
Back Icons
Plaxo Update Contacts My Plaxo Sync Get Mobile Access Fix Duplicates
Send Fax
Notes
Look for: Search In Notes Find Now Clear Options
My Notes
Notes
Notes
Current View
Icons
Notes List
Last Sever
By Catego
By Color
Open Shared
Customize Cur
Mail
Calendar
Contacts
Tasks
Notes
Provide STRACS
User parm input web
Add driver code, driver
Implement color code
Right align numbers
Integrate Gordon
Order hardware
Help FAQ content
Provide summaries for
Provide user with drop
Retest reports against
Retest all I3 Code
User hierarchy web form
Provide an Fuel Efficiency
Provide vehicle
Provide explanation of
Add default sorts to all
Arrange report list drop
Add engine hours miles to
Load updated Gordon data
Provide explanation of
Users provide feedback to
Use color coding to
All font Arial
IT handoff plan
Data filtering rules
Build DV Help/FAQ
Move feedback
Refreshes should run
Remove MS Branding
Intro subject area page
Improve MPG Change
Intro page for first time users
Provide demo DV website
Captions for Specialized
De-encode composite
Denote sort column in
Set default sort for all
Implement RBAC
39 Items
start
4 Microsoft Office O...
NetMeeting - 1 Conn...
Suggestions from Gor...
11:01 AM

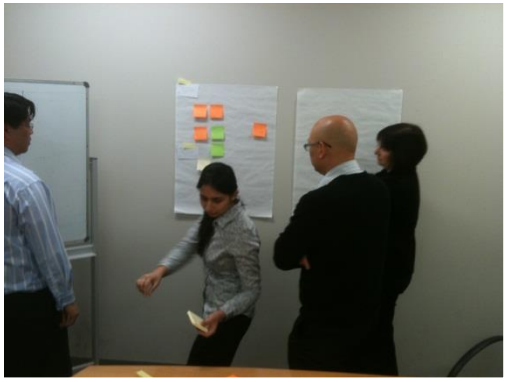
Agile Tools

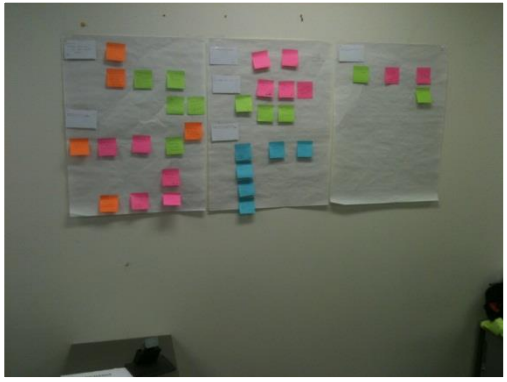
Agile teams generally prefer to use

- ✓ Physical task board
- ✓ Physical user story cards
- ✓ Paper Sprint Backlog and Burndown Chart

Software for scrum projects are

- ✓ VersionOne, ScrumWorks, RTC, JIRA etc.





Other Agile Tools

Open Source Tools

M-ASE

Story Server

TWiki XPTrackerPlugin

UserStory.NET

Xplanner

XpCgi

XPWeb

XPSwiki

Commercial Tools

Rally

AgilePlan

VersionOne

Iterate

Scope Manager

ScrumWorks

DevPlanner

XpPlanIt

Requirements Emerge & Evolve

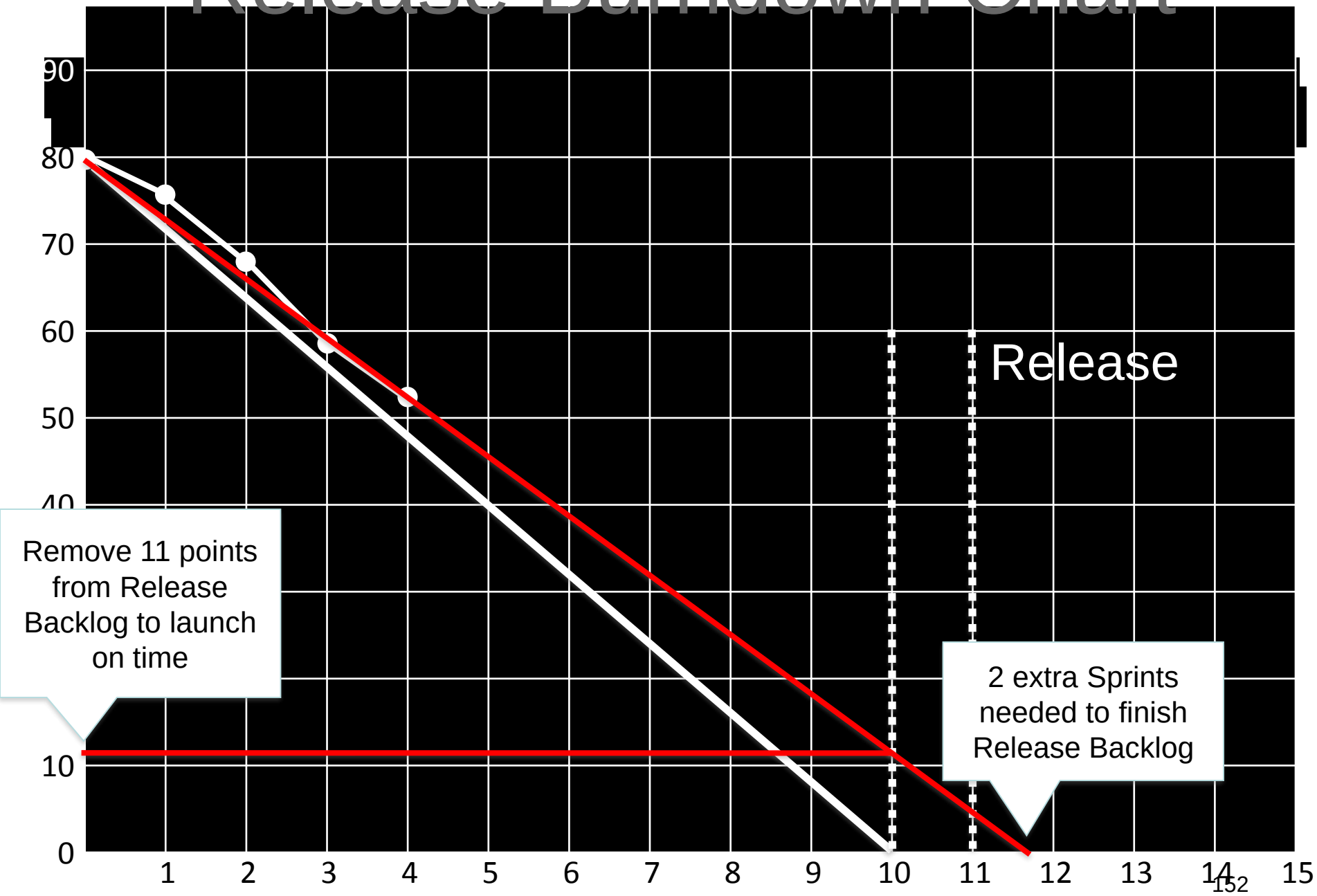
Good ideas for features emerge throughout the project, NOT just at the beginning when requirements are gathered

When customers see working software, they identify lots of changes and improvements they'd like to make

Our customer's business is constantly evolving, and this causes their requirements to change

There is a lot of miscommunication and misunderstanding – between the team and their customer, or between the customer and their stakeholder – and often this is only spotted when we see working software

Release Burndown Chart



	Relative Size	Sprint 1 Plan
As an Administrator or Employee, I want to go to the Home Page and type my login name and password, so that I can gain access to the Dashboard Page.	2	
As an Employee, I want to submit a leave request to my manager, so I can find out whether I have permission to take one or more days of leave.	3	
As an employee who has other employees reporting to me, I want to receive an email to inform me that a leave request has been filed, so that I can approve or deny it in a timely manner.	2	
As an employee, I want to be able to see the unused vacation, personal, and sick leave I currently have, so that I can plan when to take leave and how much I'm entitled to.	3	
As an employee, I want to be able to review my pending leave requests, so that I can see which requests have been approved, and which are pending review by my manager.	5	
As an Administrator or Employee, I want to click the "Forgot Password" link on the home page, enter my email address, and have a new password generated and sent to me, so I can log into the system.	1	
As an Administrator, I want to be able to modify the personal details (for example, name, or manager, or date of hire) for a particular employee, so that I can keep their information up-to-date and correct any errors.	3	
As an Administrator, I want to have a backup of the system made nightly, so that in the event of a crash or other problem, the recent days' worth of data can be restored.	8	
As an Administrator, I want to create a new Administrator account for another person, so that they have the necessary permissions within the system.	3	
As an Administrator, I want to set up a new Employee account for another person, so they can use the system to manage their leave requests and approvals.	3	
As an Administrator, I want to be able to modify the system-wide defaults for company name, company email, and number of days of leave per year.	2	
As an Administrator, I want to set up new Employee accounts for multiple other people at once by copying and pasting their information into a text field, so they can use the system to manage their leave requests and approvals.	3	
As an employee, I want to be able to review my pending leave requests, and be able to cancel any that are no longer required, so that I can keep my request up-to-date and accurate.	2	
As an employee, I want to be able to review my pending leave requests, and be able to modify any of the details that are no longer correct, so that I can keep my request up-to-date and accurate.	2	
As an employee who has other employees reporting to me, I want to be able to see a table that lists all the employees who report to me (as well as the employees that report to them), and for each a summary of their leave requests and days of leave remaining, so I can have a high-level overview of my departmental leave metrics.	5	
As an employee, I want to be able to see all my current and past leave requests (including approved, denied, and pending) since I was hired, so I can quickly get information I need regarding one particular leave request.	3	
As an employee who has other employees reporting to me, I want to be able to see all the leave requests that are currently pending my approval on a single page, and be able to approve or deny them from that page, so that I can quickly and easily take care of all my pending approvals at once.	5	
As an Administrator, I want to be able to select a subset of employees and export their data as either a tab-delimited text file, or as an excel spreadsheet.	13	
As an Administrator or Employee, I want to click "Change Password" choice on the Dashboard Page, and be able to type in a new password.	2	
As an Administrator or Employee, I want to click "Change Username" choice on the Dashboard Page, and be able to type in a new username.	3	
As an employee who has other employees reporting to me, I want to receive an email reminder once a day if there are any leave requests that have been waiting for my review for more than 72 hours, so that I can approve or deny it without further delay.	5	
As an employee, I want to receive an email notification when my manager either approves or denies one of my requests.	2	

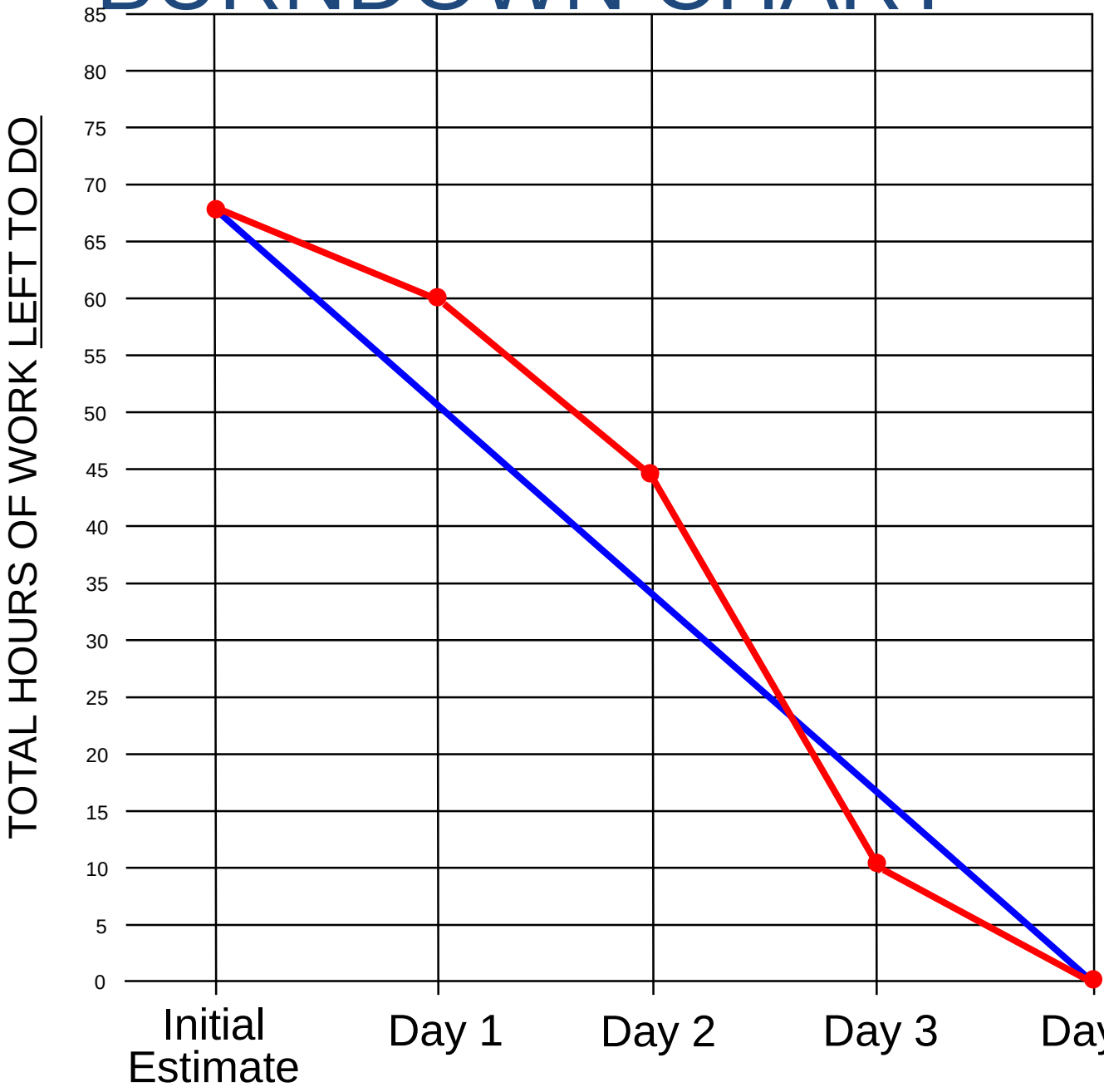
Estimated Velocity	10
Total Points in Project	80
Total Sprints Required	80/10 = 8
Uncertainty Buffer	15%
Polishing Buffer	10% = 10
Pre-Release Sprint	+1
Total Sprints Required	**= 11**

Estimating Release Date and Budget

- The team has completed high-level relative estimation using Planning Poker
- The team has estimated their velocity using past history, or by doing a Sprint Planning Meeting
- Using this data, estimate the release date and budget for EasyLeave 1.0
- Don’t forget to include:
  - Uncertainty Buffer (15% minimum recommended)
  - “Polishing” Buffer (for improvement & rework, 10% minimum recommended)
  - Pre-Release Sprint

Tasks to Complete	Owner	Initial Est.	Day 1	Day 2	Day 3	Day 4
Hkdfs jhfk sjdhf jsdhkf sdjh	Amitabh	10				
Dijsf iosdjf oisdjfoidsjfi ods		5 →				
Sijofoisjf iosjdiofj sdoijfio sd		3 →				
Dsjhk fkjsdhfjk shdjkfhsdjkf kjds		2 →				
Djkhs djhfjksdhfkjsdf jksdhjkf s		3 →				
Dosih shkjdhfjk sdfjkdsjk		2 →				
Dkjshfdjk f dshdfjk sdhfjk sdjkf	Aishwarya	5				
Jkh fjksdhdkjf dhjskfjs h	Shahrukh	4				
Kjldsfhjkldjflk sdjklf jskdljf l		1 →				
Djkfhsk djhfsjk dhfkjs dfjkskjds	Shilpa	15				
Kldjfslkd jlfkjsdkl fjskld fjl		5 →				
Sdijhkfi sdhiufhsdiu fhsduihfiudshfiu s		3 →				
Djkfhskj dhfkjsd fhdjks fjk sdkjhfk jsd		5 →				
Ldjkfsid jflskljflk sdjflk sdjkf sdkljflskfj		5 →				
	TOTAL	68				

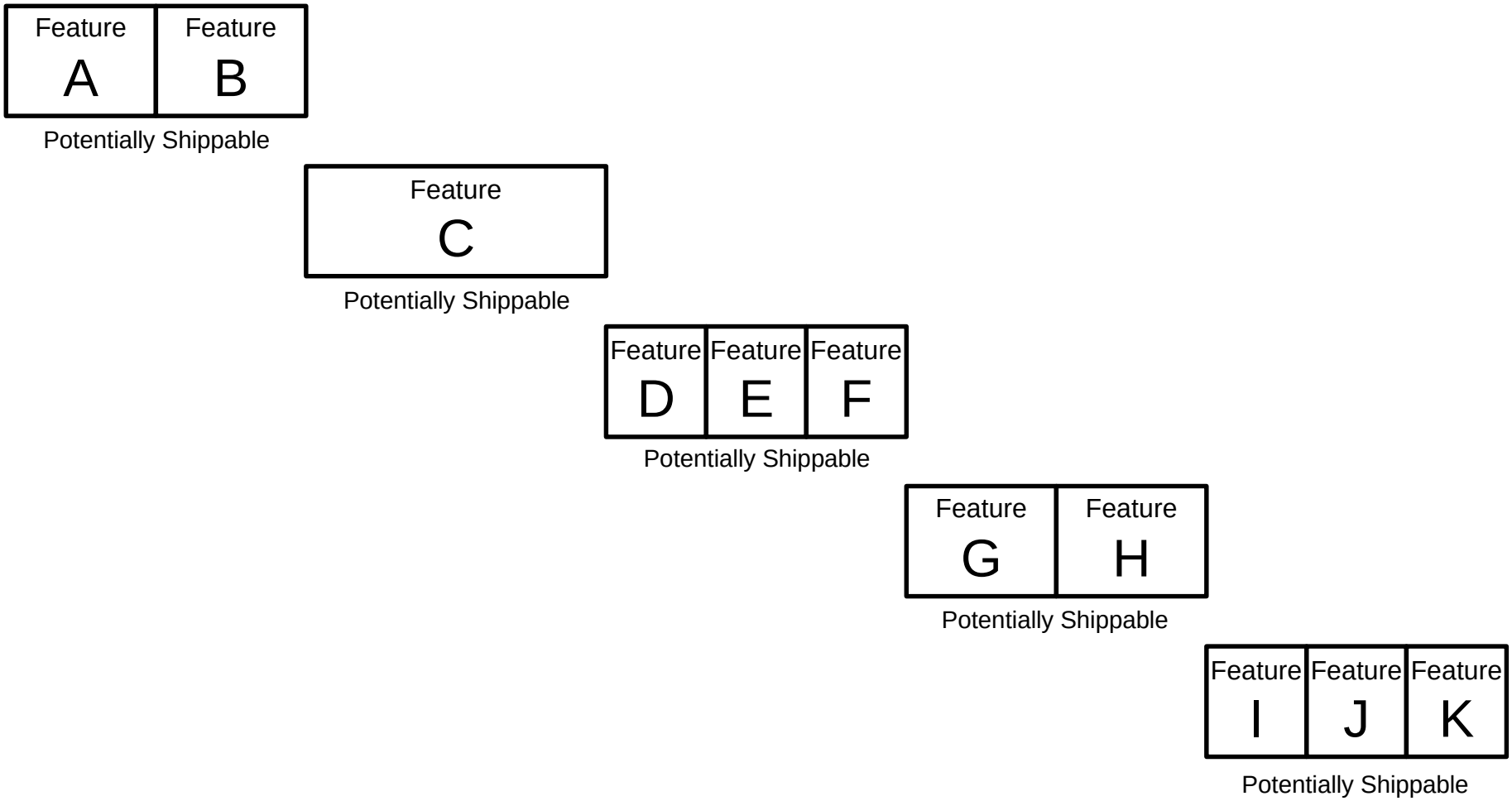
BURNDOWN CHART



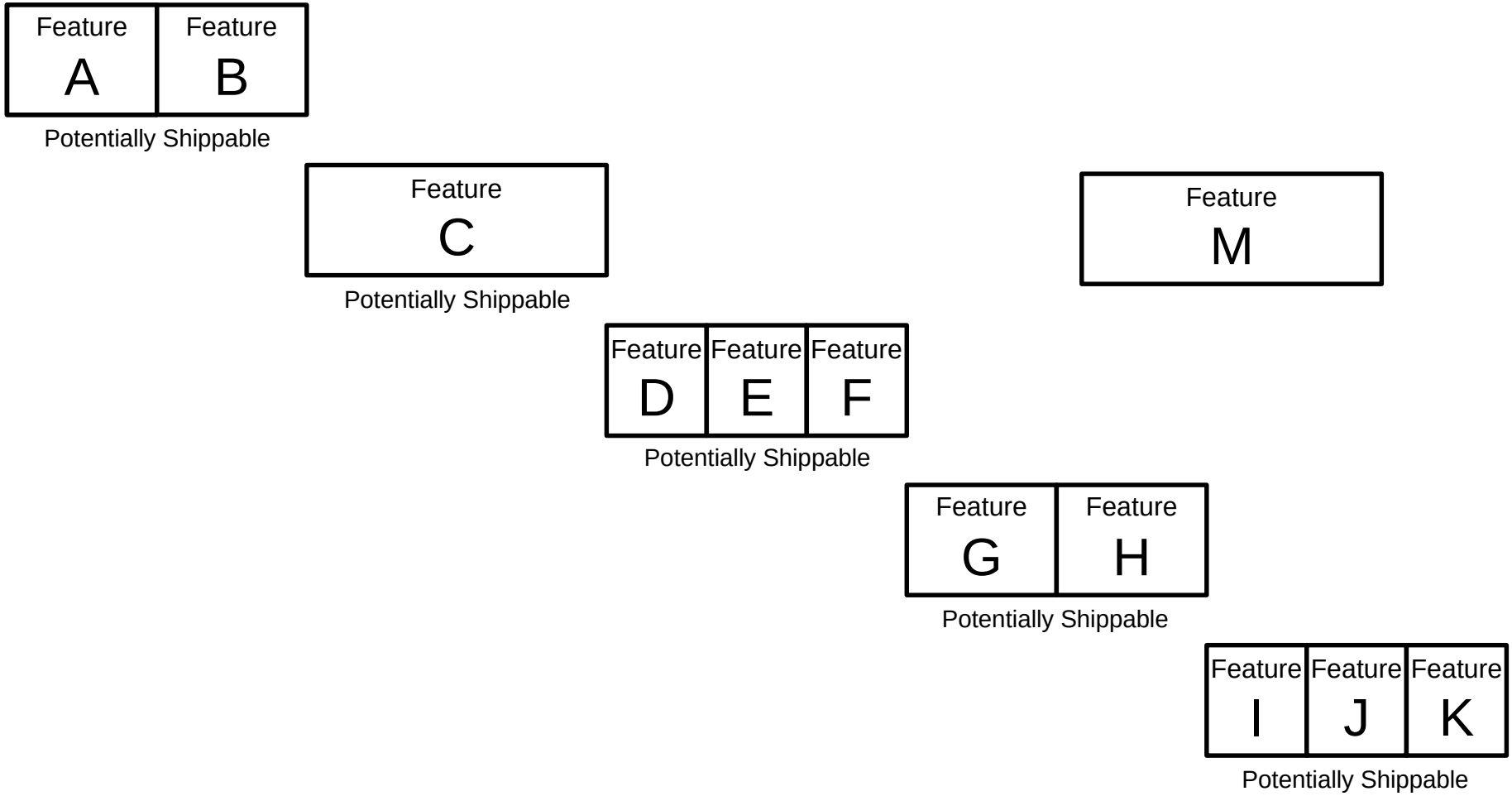
Requirements Emerge and Evolve

- Good ideas for features emerge throughout the project, **NOT** just at the beginning when requirements are gathered
- When customers see working software, they identify lots of changes and improvements they'd like to make
- Our customer's business is constantly evolving, and this causes their requirements to change
- There is a lot of miscommunication and misunderstanding – between the team and their customer, or between the customer and their stakeholder – and often this is only spotted when we see working software
- What we design can't always be built, and what we build often doesn't work the first time, so changes must be made
- We are human, and we often forget or overlook things

Cross-Functional Team (Design, Coding, Testing, Doc, UI, and other req'd skills in team)



Iterative Incremental Development



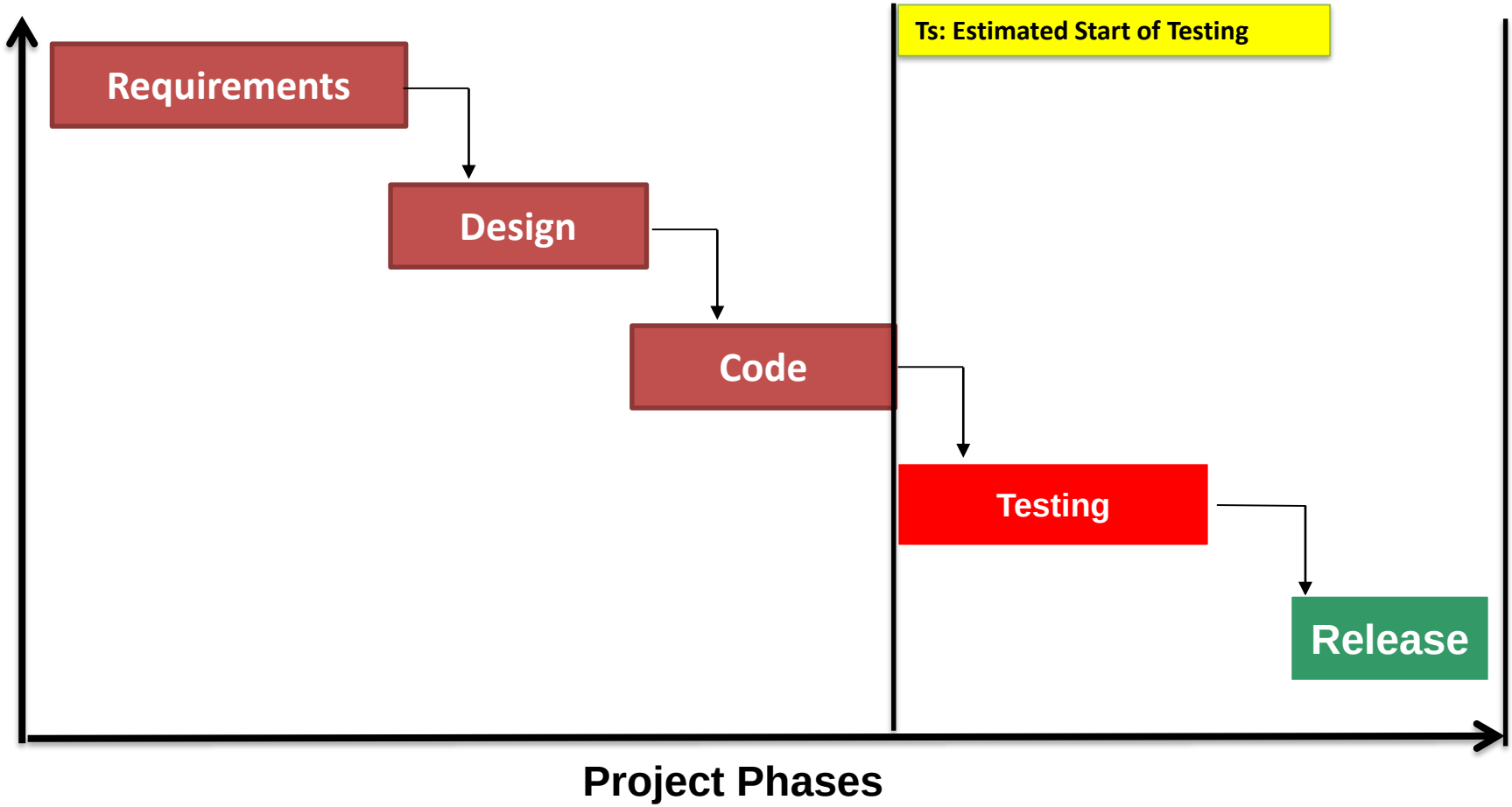
Agile Testing

Agile Testing

- ✓ Testing activities performed in Agile projects
- ✓ Is tester a Tester , developer a Developer or something else!!
- ✓ Agile team = Cross functional team, so there is a role overlap
- ✓ “Quality Is Baked In” Mike Cohn
- ✓ Production Ready/Deployable Code
- ✓ Test Driven Development/ Write Test First, Code Later

Traditional SDLC Cycle – Testing At End

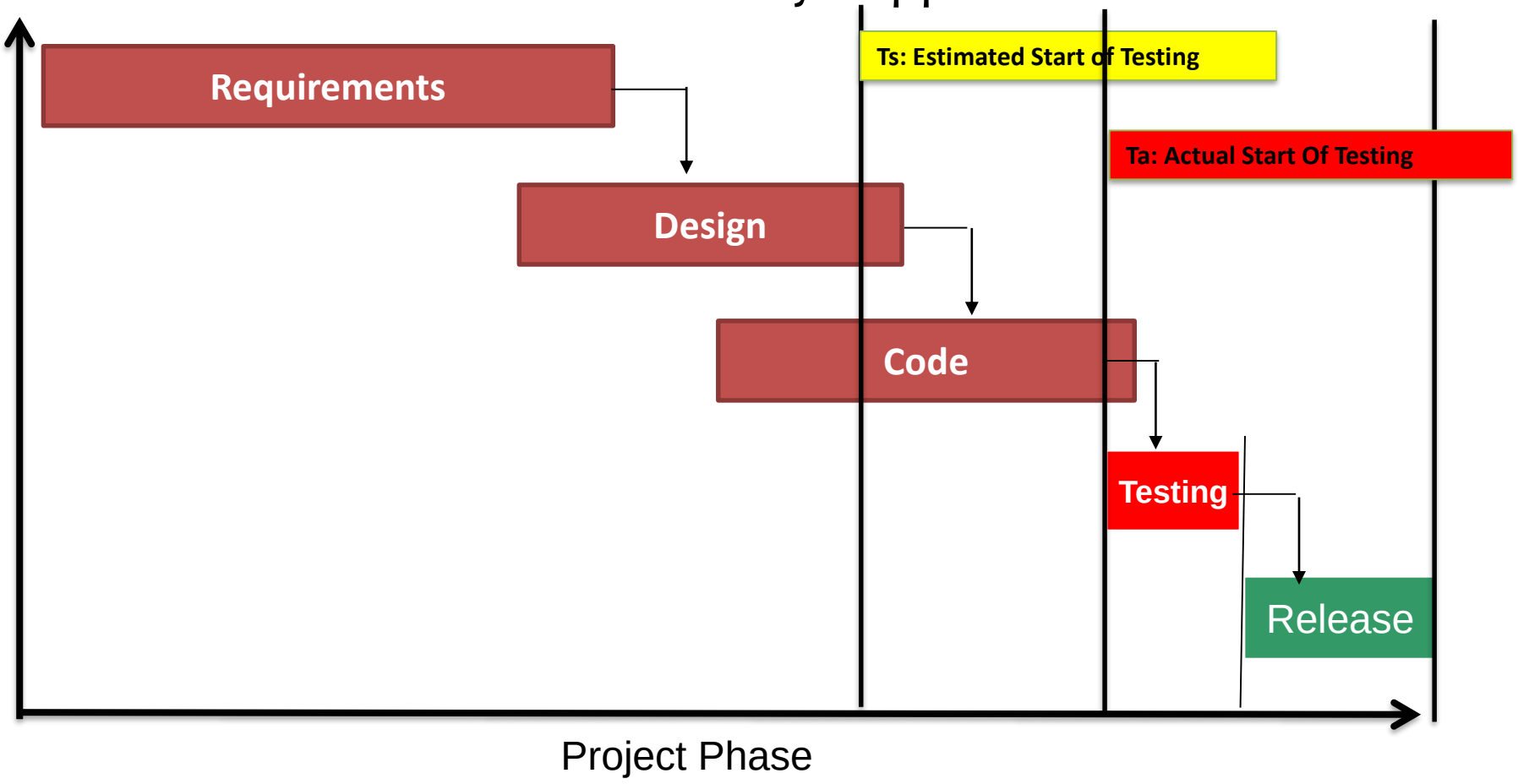
Traditional SDLC Model: What is Planned!!!



Testing is usually carried out in the later stages

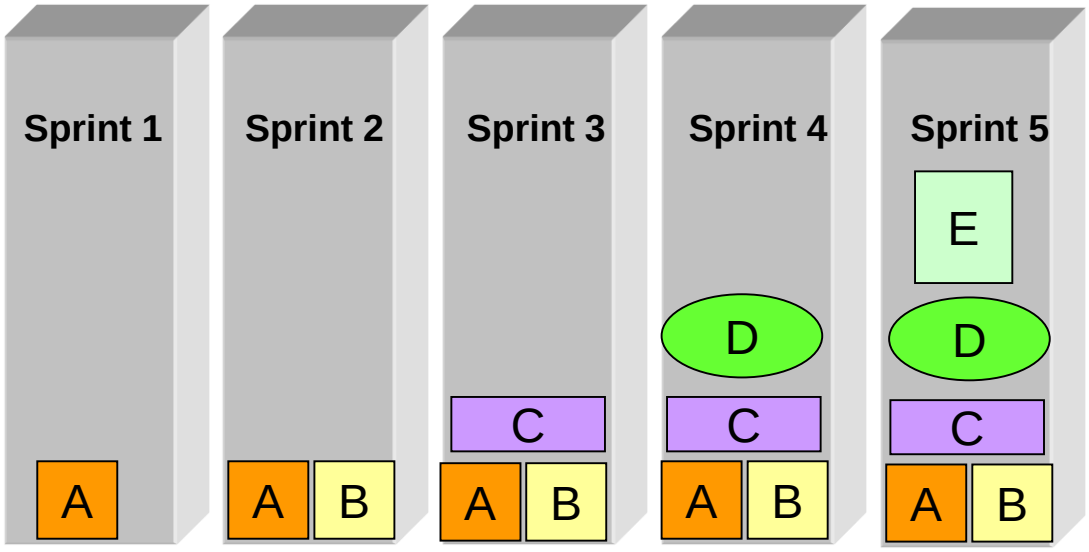
Traditional SDLC Cycle – Testing At End

Traditional SDLC Model: What Really Happens??



Testing activity is the one likely to be squeezed….

How is Agile Testing Different?

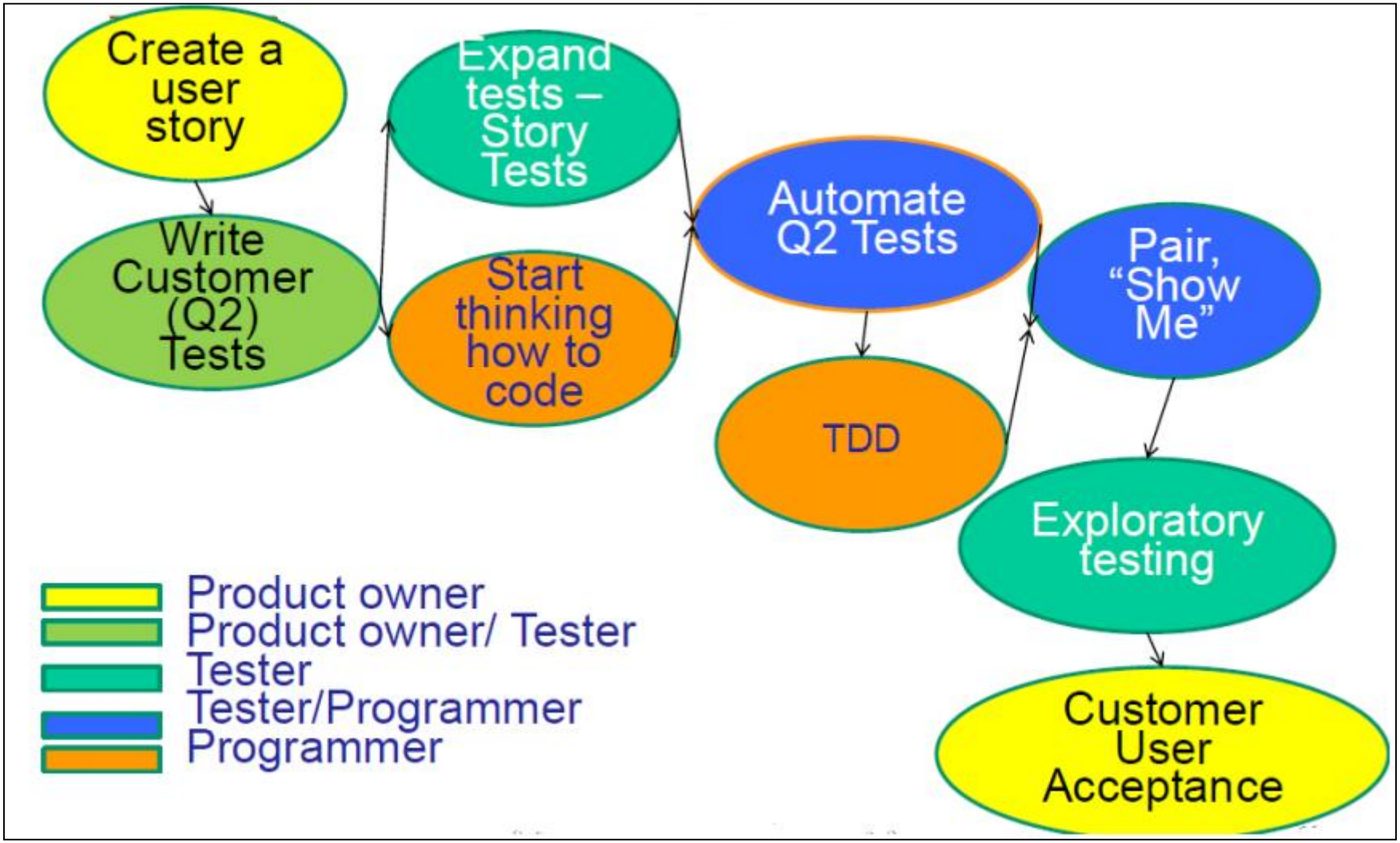


- ✓ Agile Testing is incremental & iterative
- ✓ Production ready/ shippable product at end of each iteration
- ✓ Each iteration involves Regression Testing
- ✓ Each iteration involves various Test Levels

How is Agile Testing Different?

- ✓ Agile Project Teams work closely with business
- ✓ Involves testing at each increment of the code
- ✓ Involves Test Driven development
- ✓ Not only Testers but Everyone owns Quality
- ✓ Main objective is to provide value to business by delivering workable code
- ✓ No Push "Over The Wall"
- ✓ Development and testing carried out in Parallel

Whole Team Involvement



Test Driven Development (TDD)

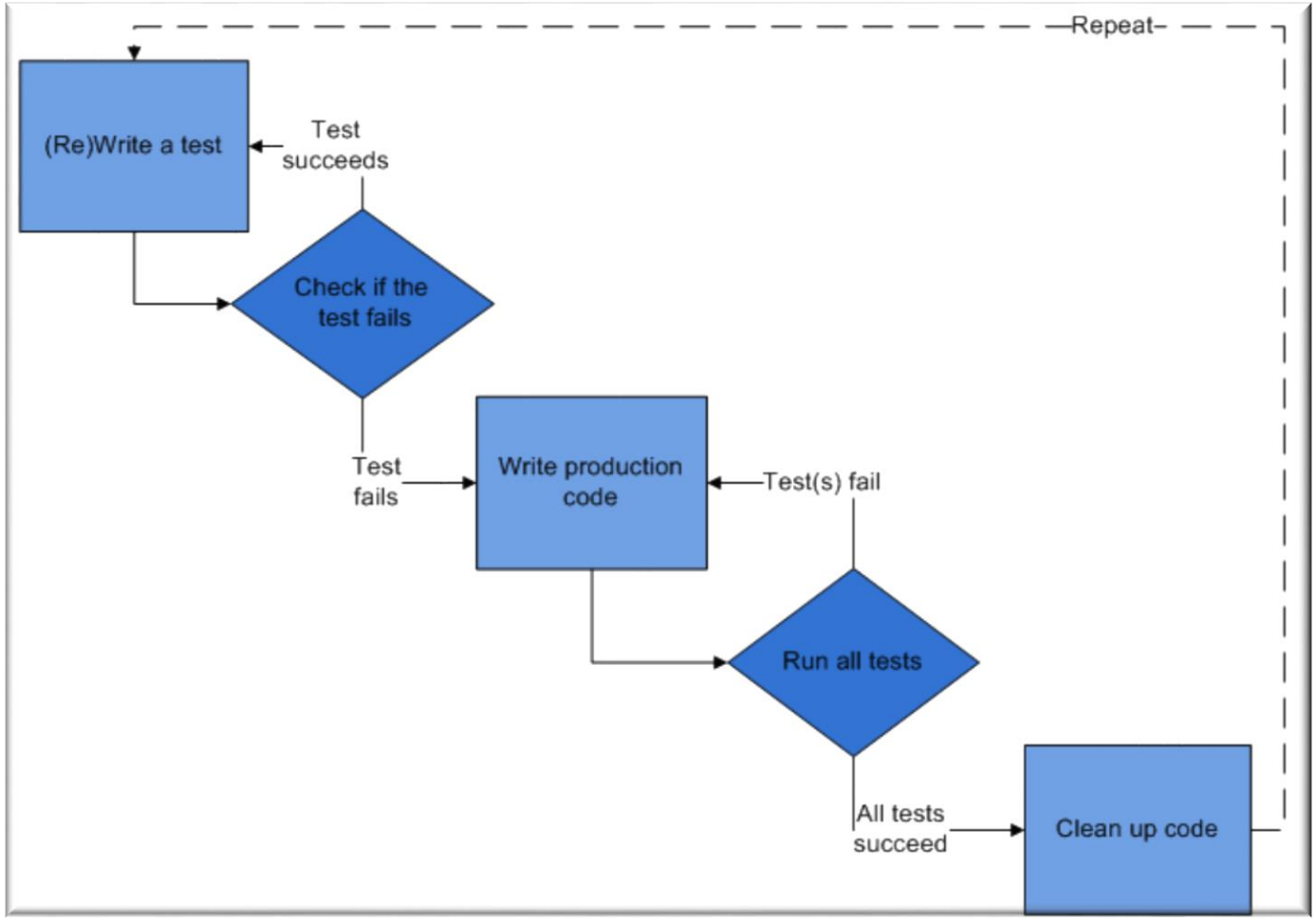
TDD is an evolutionary approach to development which combines test-first development and refactoring

- ✓ Test First Development (TFD)
  Tests are written in advance of the corresponding production code but not necessarily made to work one test at a time

- ✓ Test Driven Development (TDD)
  The programmer writes and automates a small unit test before writing the piece of code that will make the test pass. The production code is made to Work one test at a time

- ✓ Code refactoring
  Disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior

Test Driven Development (TDD)



Flowchart Of Test Driven Development (TDD)

Test Driven Development (TDD)

- ✓ TDD = TFD + Refactoring

- ✓ Steps Involved In TDD
  1. Create a Test Case
  2. Run the Test Case and ensure it fails (as the Code is not ready)
  3. Create /Develop code and make sure it passes the test case created
  4. Refactor the developed code

- ✓ TDD is primarily a specification technique with a side effect of ensuring that your source code is thoroughly tested at a confirmatory level

Role of Tester

- ✓ Is Tester role obsolete in Agile Environment??
- ✓ Agile Teams are cross functional teams, so testing is not limited to testers, developers can also perform UAT
- ✓ All testing phases are repeated in a sprint/iteration
- ✓ Testers are involved early in the project lifecycle
- ✓ Testers co-locate with developers and analysts
- ✓ Possess wide range of skills and not just limited to Testing
- ✓ Testers test each increment of the code once ready

Role of Tester

Involved continuously from start…

- Facilitate communication between the technical & business stakeholders
- Support early validation of requirements
- Help the customer/business stakeholders define acceptance criteria

Support creation of automated acceptance tests

- Define for developers to script
- Expand scope of ‘acceptance’ tests
- Advise the team about overall risks and trends
- Perform manual/exploratory tests
- Needs ‘technical awareness’

Role of Developer

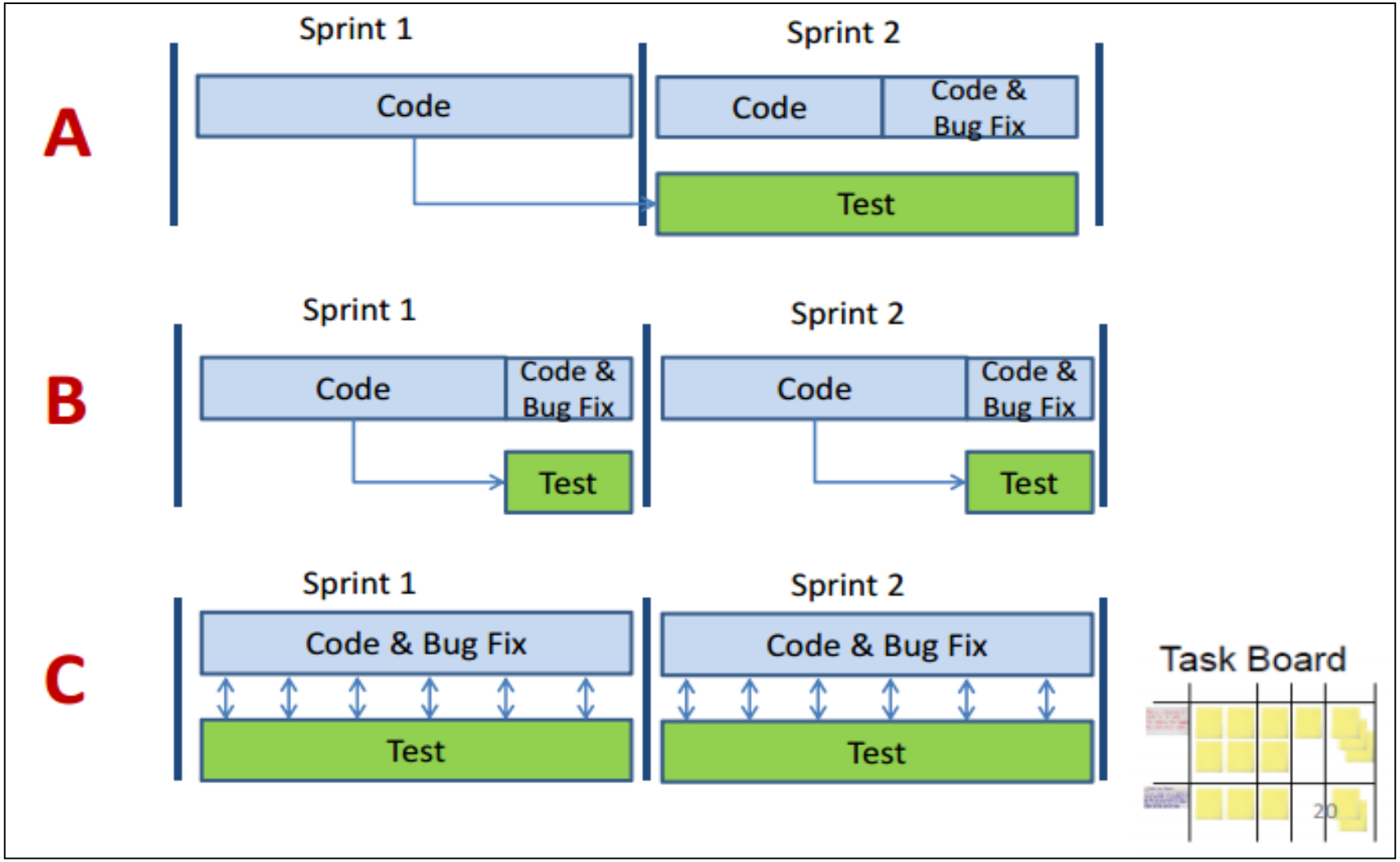
Perform code reviews

- Use static analysis tools
- Perform effective unit testing
- Automated ideally using TDD achieving structural coverage
- Perform component/component integration testing (Ideally automated, API/Service level)

Support testers in Acceptance/System testing

- Frameworks for automated API/Service level testing –
- GUI based automation

Evolving From Sequential To Iterative Development

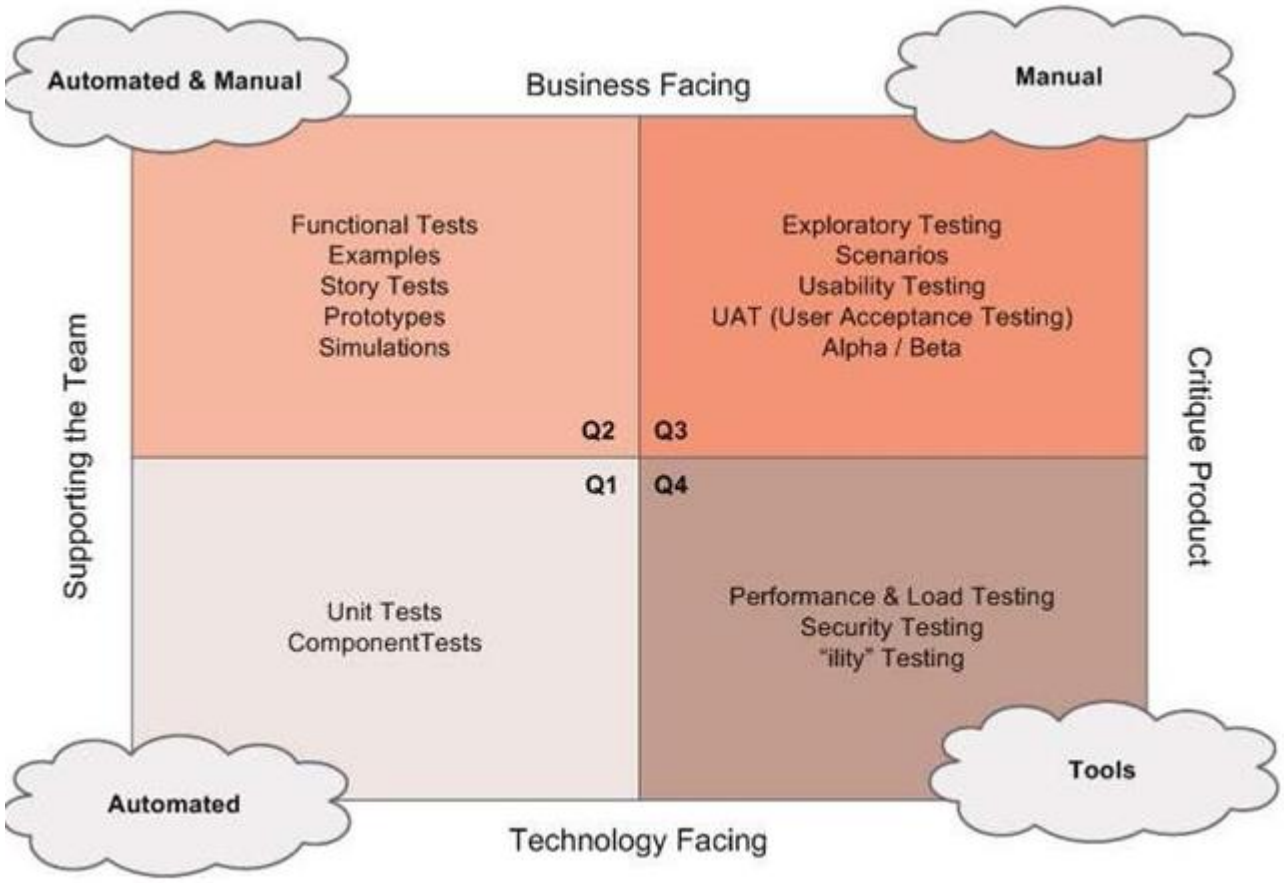


Agile Testing Quadrants

Introduction

- ✓ Agile Testing :- Tests are distributed across 4 quadrants
- ✓ Different types of testing carried out for different reason
- ✓ Each quadrant represents different reasons for testing
- ✓ Tests categorized into 2 major axis
- ✓ X-Axis Bottom: Technology Support Tests
- ✓ Y-Axis Left : Tests That Support the Team /Support Programming
- ✓ X-Axis Top : Business Facing Tests
- ✓ Y-Axis Right : Tests That Critique The Product

Agile Testing Quadrants



Agile Testing Strategy

- ✓ Tests in all 4 quadrants need to be executed
- ✓ Not every story needs all the tests to be done
- ✓ However combination of tests from all the four quadrants will determine if each feature has met customers requirement
- ✓ Testing is a shared responsibility between developers, testers , BA and ensure tests cover all the quadrants
- ✓ SMEs can be sought for specialized testing e.g. are Security Test, Performance Test, Stress test etc.
- ✓ Customers are closely involved in Acceptance test

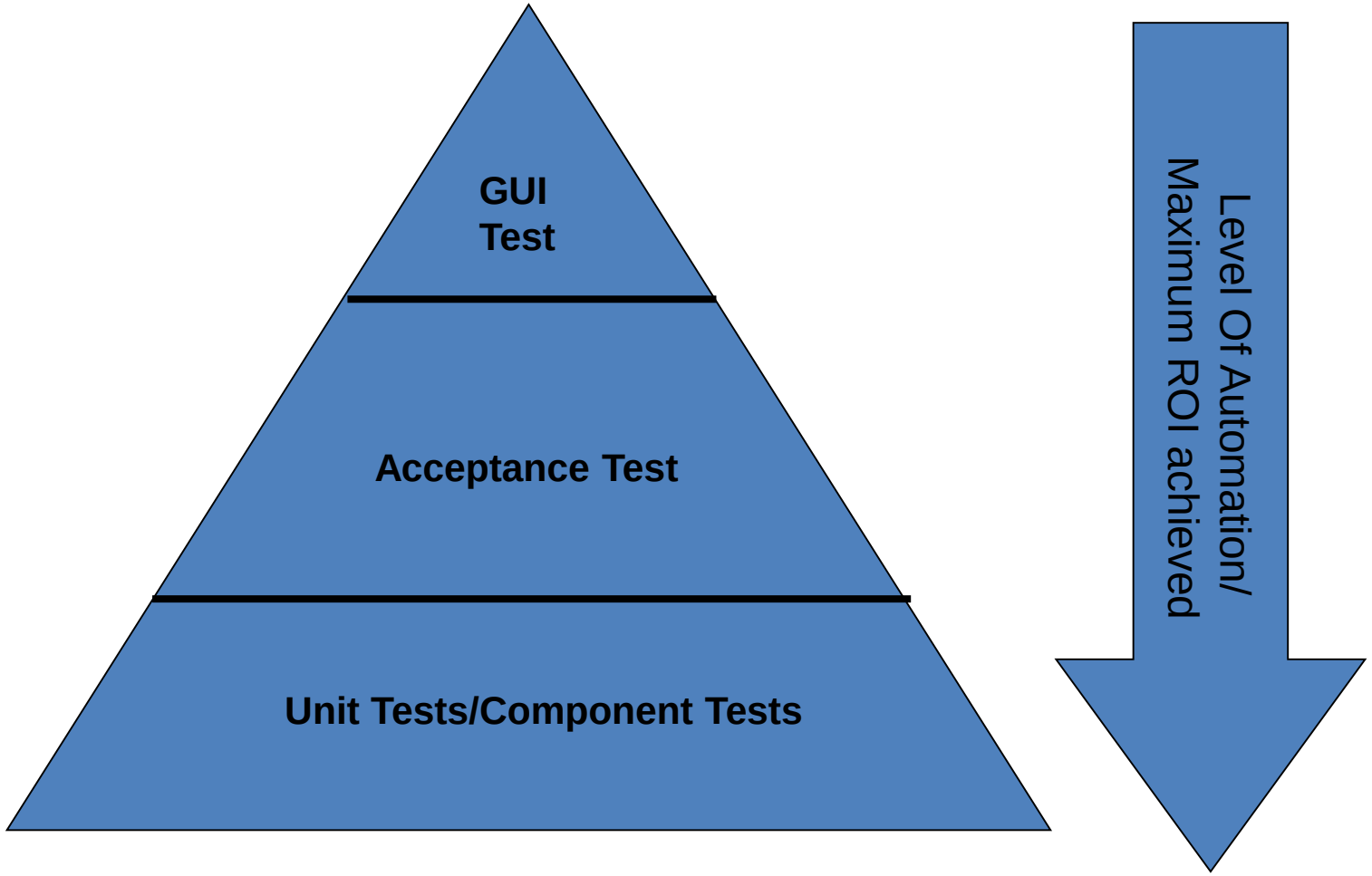
Why Test Automation?

- ✓ Manual process is cumbersome and takes time
- ✓ Manual process is error prone
- ✓ Automation is faster for tasks which are repetitive in nature
- ✓ If ensured that coverage is sufficient, automation testing can provide a safety net for regression testing
- ✓ Automation tests provide feedback early and often
- ✓ If implemented properly they provide a huge ROI and payback

Barriers To Automation

- ✓ Attitude- Why go for automation
- ✓ “The Hump OF Pain” syndrome
- ✓ Huge Initial investment
- ✓ Volatile code or code which is always in a state of flux
- ✓ Legacy systems
- ✓ Fear
- ✓ Old Habits die hard

Test Automation Pyramid



Test Automation Pyramid

Lower Layer

- ✓ Foundation or base layer which supports the top layers
- ✓ Consists of Technology facing and supporting the team tests
- ✓ Consists of unit and component tests
- ✓ As a test strategy ensure maximum automation happens in this zone
- ✓ Tests written in same language as that of development
- ✓ Most common example is JUnit framework
- ✓ Post mastering TDD, these tests are quicker and less expensive to develop and execute

Test Automation Pyramid

Middle Layer

- ✓ Consists of automated business facing tests to support the team
- ✓ These tests certify that "We are building the right thing"
- ✓ Mostly consists of story tests, acceptance tests and cover the functionality over and above unit tests
- ✓ Unlike bottom layer, tests are written in a language in which customers can comprehend
- ✓ Tests operate at the API level without GUI intervention
- ✓ Involves setting up environment and test data for verifying input and output conditions and use data sheet feature of tools

Test Automation Pyramid

Top Layer

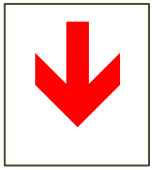
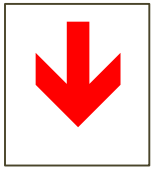
- ✓ Consist of Capture and Playback tests
- ✓ Provide the lowest ROI from automation
- ✓ More expensive to write and maintain
- ✓ Ensure that automation efforts are kept at the minimum at this level
- ✓ Written once the GUI code is ready and completed

“The Product Owner Changes His Mind”

In the middle of the Sprint, the Product Owner comes to you and says he wants to make a substitution in the current Sprint. He wants to drop one of the Product Backlog Items, and replace it with a new one. The new feature is very high priority, and the Product Owner wants the team to start on it as soon as possible.

Scenario…

What if we say "Yes" to the Product Owner's request to change the Team's commitment in the middle of the Sprint?

Current Sprint			Future Sprints				
Satisfaction of Product Owner	Team's ability to deliver its commitment	Team's focus and sense of commitment	PO's willingness to not request changes during Sprint	PO's time and thought in preparing the Product Backlog	Team's focus and sense of commitment in future Sprints	Team's discipline in following the other rules of Scrum	Other teams' discipline in following the rules of Scrum
							

Management Styles…

The following are the key management styles

- ✓ Directing- Telling what needs to be done
- ✓ Facilitating – Co-ordinate inputs of others
- ✓ Coaching – Instructing others
- ✓ Supporting – Provide assistance to others
- ✓ Autocratic- Manager retains power and decision making authority
- ✓ Consultative-Invite ideas from others
- ✓ Consultative-autocratic
- ✓ Consensus-Make decisions based on group agreement

Management Styles

- ✓ Delegating-Teach and let a sub-ordinate work
- ✓ Bureaucratic- emphasize procedures and historical ways
- ✓ Charismatic-create a powerful self image to draw people to you
- ✓ Democratic/participative
- ✓ Laissez-faire – Hands off style, provides little or no direction and gives freedom
- ✓ Analytical- Leadership driven by data, facts and numbers
- ✓ Influencing-Change others thinking by advice or training or by example

Management Styles...

Servant Leadership

Servant leadership is a leadership theory that focuses more on the followers than the leader him/herself

Proposed by Robert K. Greenleaf in 1970

Servant-leaders achieve results for their organizations by giving priority attention to the needs of their colleagues and those they serve

Servant-leaders are often seen as humble stewards of their organization's resources like human, financial and physical

Management Styles

Servant Leadership

Qualities needed for a servant leader are :-
listening, empathy, healing, awareness, persuasion, conceptualization, foresight, stewardship, growth and building community

Acquiring these qualities tend to give a person authority versus power

Servant leadership emphasizes collaboration, trust, empathy, and the ethical use of power which is in contrast to the traditional top down leadership approach

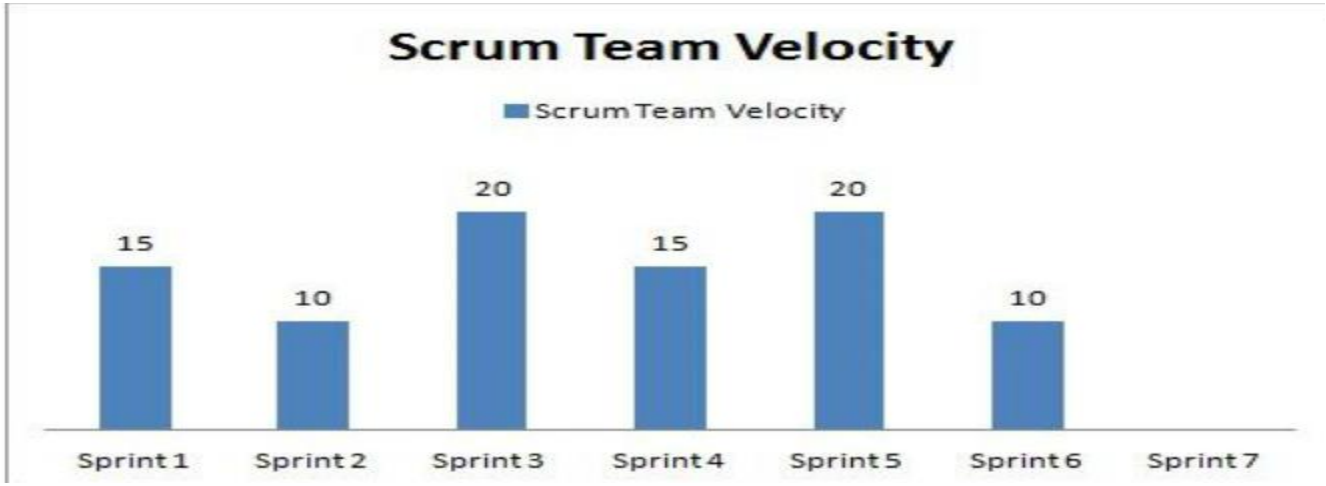
Is Agile Right for my Project

Let us see if Agile is right for us??

QUIZ

QUIZ

Question 1:



Which of the below story points should the team pick up for Sprint 7 ?

A. Story 1 **(6 SP)**, Story 3 **(8 SP)**, Story 4 **(5 SP)**
B. Story 1 **(6 SP)**, Story 2 **(2 SP)** , Story 3 **(8 SP)**
C. Story 1 **(6 SP)**, Story 3 **(8 SP)**
D. Story 1 **(6 SP)**, Story 3 **(8 SP)**, Story 5 **(1SP)**

Answer: Option D (Average Velocity from Sprint 1 -6 = 15 SP)

QUIZ

Question 2:

When forming an Agile team which of the following you would pick up ?

A. Highly Technical skilled Resources
B. Top Management Professionals
C. Generalist Specialist
D. All Of The Above

Answer: Option C (Agile expects resources to be multi skilled)

QUIZ

Question 3:

What is the key point of difference between Agile Planning and Traditional Planning?

A. No Difference
B. Agile Planning is done only Once
C. Agile Planning is non iterative
D. Agile planning places emphasis on plan and is iterative

Answer: Option D

QUIZ

Question 4:

The product backlog is maintained by:

A. Consulting Experts
B. The Scrum Master
C. The Product Owner
D. The Scrum Team

Answer: Option C

QUIZ

Question 5:

The person or persons in charge of the tracking and the updates for the scrum (equivalent to a project manager) is called the:

A. Scrum Master
B. Product Owner
C. Project Leader
D. Consulting Expert

Answer: Option A

QUIZ

Question 6:

A Persona used in a user story is ?

A. A person involved in UAT of User Story
B. Product Owner
C. A person who is never going to use the application
D. Imaginary Role

Answer: Option D (Persona is imaginary representation of the User Role)

QUIZ

Question 7:

Which of the following is a characteristic of an Agile leader?

A. Task focused
B. Process oriented
C. Supportive
D. Disinterested

Answer: Option C

QUIZ

Question 8:

What are the advantages of maintaining consistent iteration /sprint (timebox) lengths throughout the project?

A. They help to establish a consistent pattern of delivery
B. They help the team to objectively measure progress
C. They provide a consistent means of measuring team velocity
D. All of the above

Answer: Option D

QUIZ

Question 9:

Why is it important to trust the team?

A. High trust teams do not have to be accountable to each other
B. High trust teams do not require a user representative
C. The Project Manager does not have to keep a project schedule
D. The presence of trust is positively correlated with the team performance

Answer: Option D

QUIZ

Question 10:

Who should be the main judge of the business value?

A. Product Owner
B. Scrum Master
C. The Team
D. All of the above

Answer: Option A (PO is the business SPOC and is the best judge for Business Value

QUIZ

Question 11:

If a timebox (iteration) plan needs to be reprioritized in a hurry, who should re-prioritize?

A. Only The Team
B. Only Product Owner
C. Only Scrum Master
D. The whole team including PO and the Scrum Master

Answer: Option D (Together they can consider both business value and practicality)

QUIZ

Question 12:

What is the effect of having large visible project plan on a wall?

A. It is a fire risk and a health hazard
B. It communicates progress to the team and other stakeholders
C. It is dangerous, as management will misinterpret what the team is doing
D. It is useless, as it does not allow the team to innovate

Answer: Option B(Agile stresses on openness in communication)

QUIZ

Question 13:

How should work be allocated to the team in an Agile project?

A. The Scrum Master should give tasks to individuals to create challenges for them
B. Tasks should be randomly allocated using Planning Poker
C. Team members should self-select tasks
D. The biggest tasks should be done by the Scrum Master themselves

Answer: Option C(Agile teams are self organizing and motivated)

QUIZ

Question 14:

What should the team do if the customer representative (Product Owner) is too busy to be available?

A. Carry on with the work without customer input. It will probably be faster without customer interference anyway and Team is self organizing and motivated
B. Ask SCRUM Master to send Product Owner an email warning that the end product will be completed on time, but will not meet their needs
C. Allow a developer to take on the role of customer representative
D. Draw the problem to the attention of Scrum master

Answer: Option D (Scrum Master is the interface between PO and team)

QUIZ

Question 15:

Which one of the following is a key feature that you would expect to find in an Agile project?

A. A large number of written progress reports
B. No project documentation, as all communication is tacit information
C. Good face-to-face communication, supplemented by lean but sufficient documentation
D. All documentation done in previous projects would be replicated for the Agile project.

Answer: Option C

QUIZ

Question 16:

When handling team dynamics, the Agile Leader (Scrum Master) should …

A. Work to build trust between the team members
B. Encourage an environment of competition and personal advantage
C. Stand for no nonsense and show who is the boss
D. Expect team members to sort out all of their own problems, and not come to him/her for any help

Answer: Option A

QUIZ

Question 17:

Which one of the following statements is correct regarding acceptance of any deliverables on an Agile Project?

A. The team should allow only senior managers to sign off deliverables
B. The team should get acceptance of project deliverables from the appropriate stakeholders at least at the end of every timebox / iteration
C. The team should get acceptance of project deliverables from the users during a UAT phase at the end of the project
D. Acceptance of any particular deliverable on the project is gained from all stakeholders at the same time.

Answer: Option B

QUIZ

Question 18:

What is the Agile approach to doing design early in a project?

A. A big design up front is always a good idea
B. Just enough design up front gives a good foundation to start from and helps to mitigate risk, without unnecessarily wasting time
C. No design up front is the best approach as most of the fun of a project is discovery of the unexpected
D. Design has no place in an Agile project

Answer: Option B

QUIZ

Question 19:

An Agile approach advocates which of the following approaches?

A. Get something quick and dirty thrown together to save time
B. Get something simple up and working as quickly as possible
C. Get something business-valuable delivered as quickly as possible, consistent with the right level of quality
D. Get something delivered once it has been fully documented and the documentation signed off as complete and unchangeable

Answer: Option C

QUIZ

Question 20:

Which one of the following is an important feature of the stand-up / wash up meeting?

A. Everyone must stand for the whole time
B. The meeting must be short and well structured
C. The Scrum Master must ensure it is clear to all which team members are not performing
D. No-one is allowed to leave the stand-up meeting until all problems raised have been solved

Answer: Option B

Famous Quotes

"We cannot build our own future without helping others to build theirs"

"Everybody counts, everybody deserves a chance, everybody has a responsible role to play and we all do better when we work together."
--Bill Clinton

"As we look ahead into the next century, leaders will be those who empower others"
---Bill Gates

References…

Agile Alliance – www.agilealliance.org

Agile Manifesto – www.AgileManifesto.org

Introduction to Agile Software Development
http://www.danube.com/docs/Intro_to_Agile.pdf

SCRUM Alliance - www.scrumalliance.org

Thank You