Agile

Shreyata Sugandhi

Corporate Trainer & Consultant

https://www.linkedin.com/in/shreyatasugandhi/

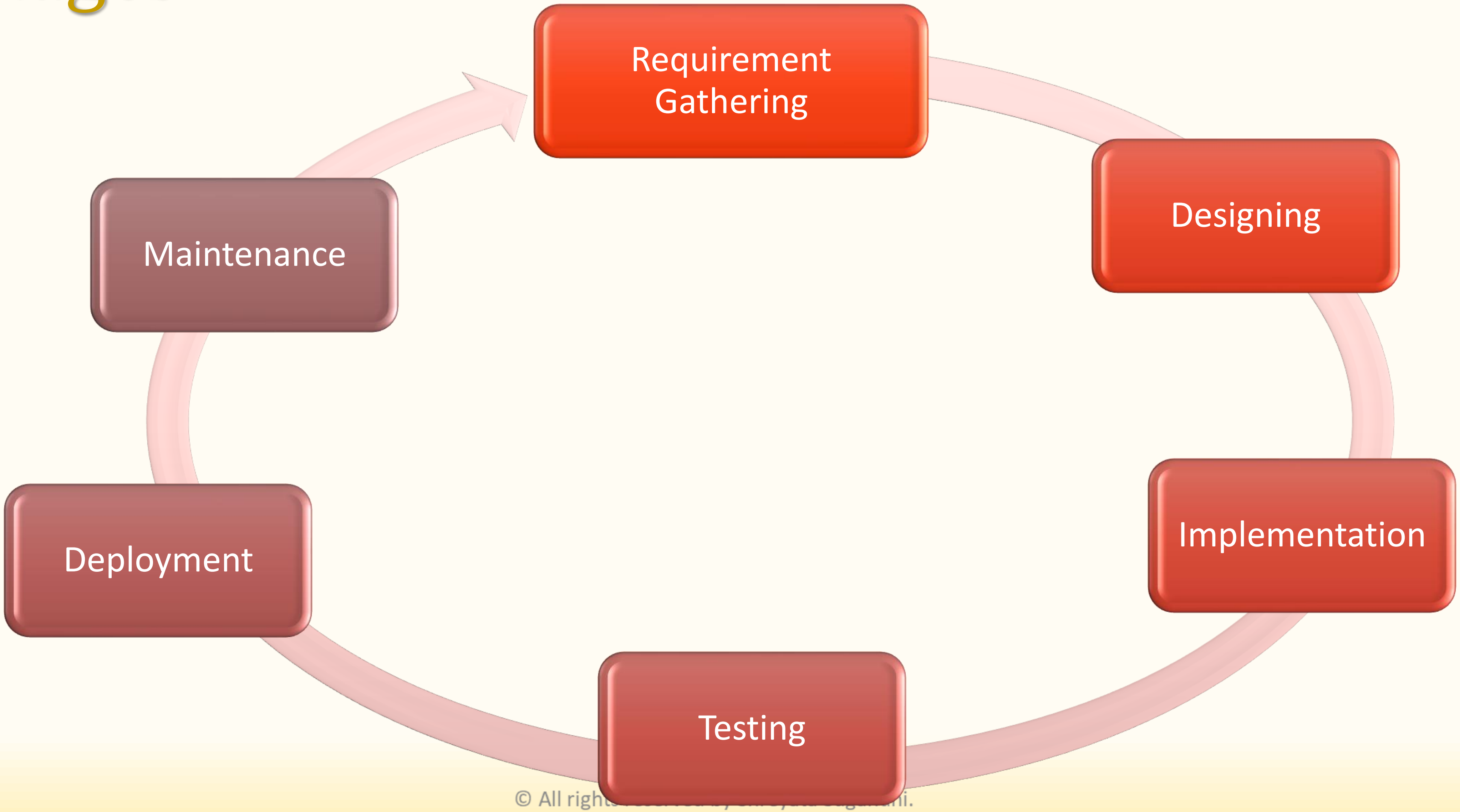
SDLC

- SDLC, Software Development Life Cycle is a process used by software industry to design, develop and test high quality software.

- The SDLC aims to produce a high quality software that **meets or exceeds customer expectations**, reaches completion **within times** and **cost estimates**.

- ISO/IEC 12207, is an international standard for software life-cycle processes.

Stages



Requirement Gathering

Designing

Implementation

Testing

Deployment

Maintenance

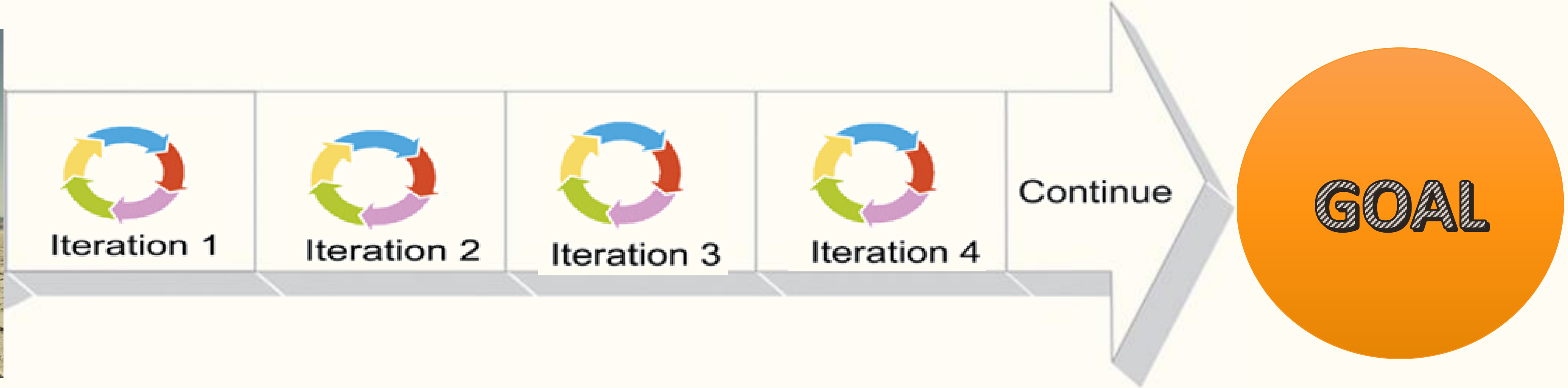
SDLC Models

- Waterfall Model
- V-Model
- Iterative Model
- Spiral Model
- Big Bang Model
- Prototype
- Agile

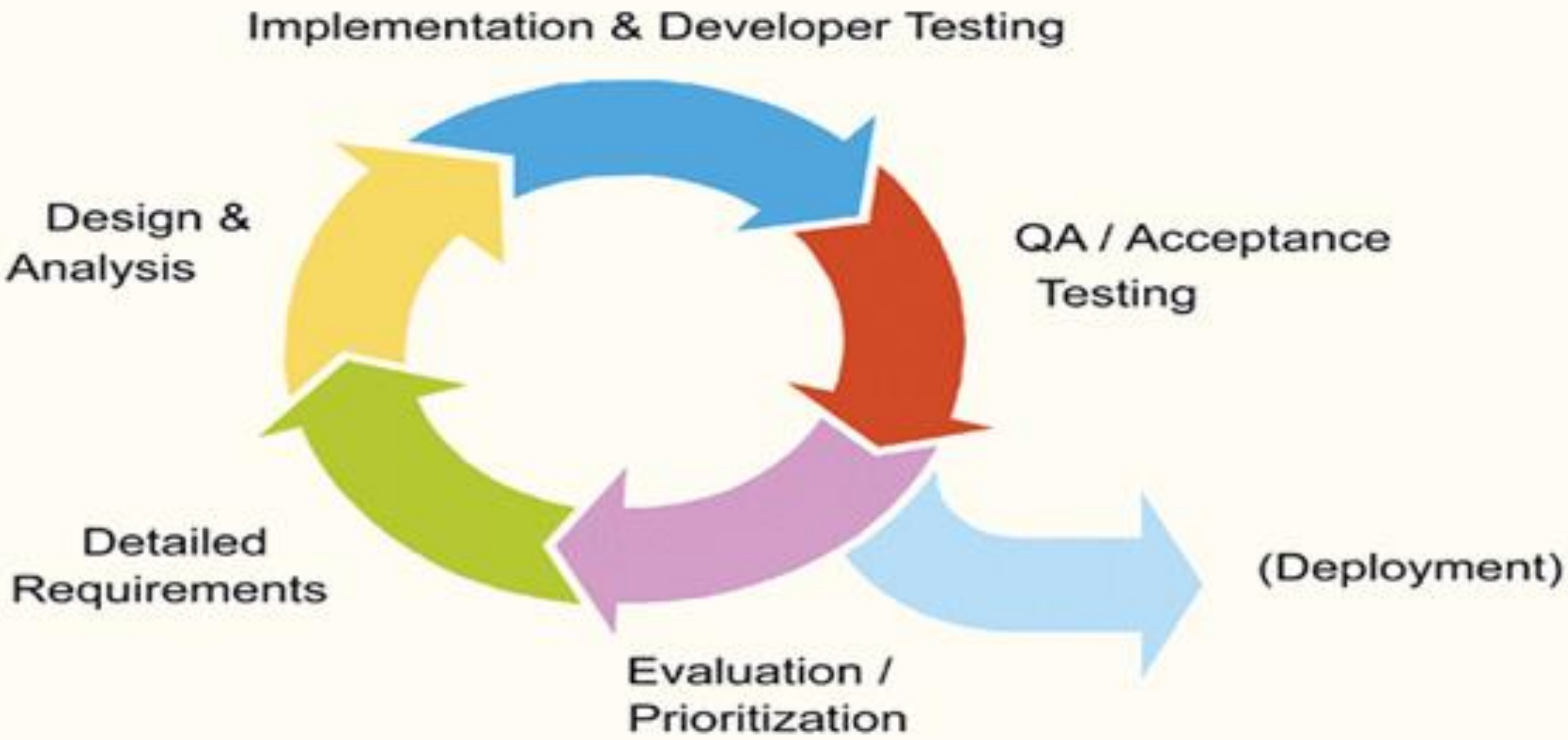
What is Agile

- Agile is all about
  - adaptive planning
  - evolutionary development
  - early delivery
  - continuous improvement
  - it encourages rapid and flexible response to change
- AGILE is a methodology that has **continuous iteration** of development and testing throughout the software development life cycle of the project.

Agile SDLC


Iteration 1
Iteration 2
Iteration 3
Iteration 4
Continue
GOAL

Details of Iterations



Core Values of Agile

- Individual and team interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Agile Principles

- Customer satisfaction by early and continuous delivery of valuable software
- Welcome change - Even late in the development process, changing needs need to be addressed.
- Working software is delivered frequently (weeks rather than months)
- Close, daily co-operation between business people and developers
- Motivated Team - Team members must be motivated and trusted to complete the project successfully and on time.
- Face-to-face conversation is the best form of communication (co-location)
- Working software is the principal measure of progress

Agile Principles

- Sustainable development, able to maintain a constant pace
- Continuous attention to technical excellence and good design
- Simplicity—the art of maximizing the amount of work not done—is essential
- Self-organizing teams
- Reflect and Adjust - The effectiveness of the team can be improved by regularly reflecting on their work and making improvements.

Agile Methods / Frameworks / Philosophies

- Scrum
- Kanban
- Extreme programming (XP)
- Lean software development
- Crystal Clear methods
- Dynamic systems development method (DSDM)
- Feature-driven development (FDD)
- Adaptive software development (ASD)
- Disciplined agile delivery

Scrum

- A flexible product development strategy where a development team works as a unit to reach a common goal
- Specifically on how to manage tasks within a team-based development environment
- Scrum is derived from activity that occurs during a rugby match
- Scrum believes in empowering the development team and advocates working in small teams (say- 7 to 9 members)

Kanban

- Kanban originally came from Japanese word that means, a card containing all the information needed to be done on the product at each stage along its path to completion.
- It is a method that is used to design, manage, and improve the flow of systems.
- Kanban enables organizations to visualize their flow of work and limit the amount of work in progress.
- It is used in situations where work arrives unpredictably, and where it needs to be deployed immediately without waiting for other work items.

Scrum vs Kanban

Scrum	Kanban
Pre-defined roles of Scrum master, Product owner and team member	No prescribed roles
Time boxed sprints	Continuous Delivery
Work is pulled through the system in batches (the sprint backlog)	Work is pulled through the system (single piece flow)
No changes are allowed mid-sprint	Changes can be made anytime
Velocity is the matric	Cycle time is the matric
More appropriate in situations where work can be prioritized in batches that can be left alone	More appropriate in operational environments with a high degree of variability in priority
Stories(tasks) needs to be broken down to achieve completion of the sprint	No particular task size needs to be defined

History of Scrum

- 1986 - The name Scrum is first introduced by management experts Ikujiro Nonaka and Hirotaka Takeuchi.
- 1995 - Jeff Sutherland and Ken Schwaber create the early versions of what would become the Agile methodology.
- 2001 - The Agile Alliance is founded, and the first book on Scrum, the Agile Software Development with Scrum, is published.
- 2002 - Schwaber found the Scrum Alliance, and certifications are added.
- 2006 - Scrum Inc. is created and is in full swing. The certified Scrum courses are taught to users across the world.
- 2009 - Scrum.org is created. It offers the professional Scrum series to users.
- 2010 - The first Scrum guide is published.

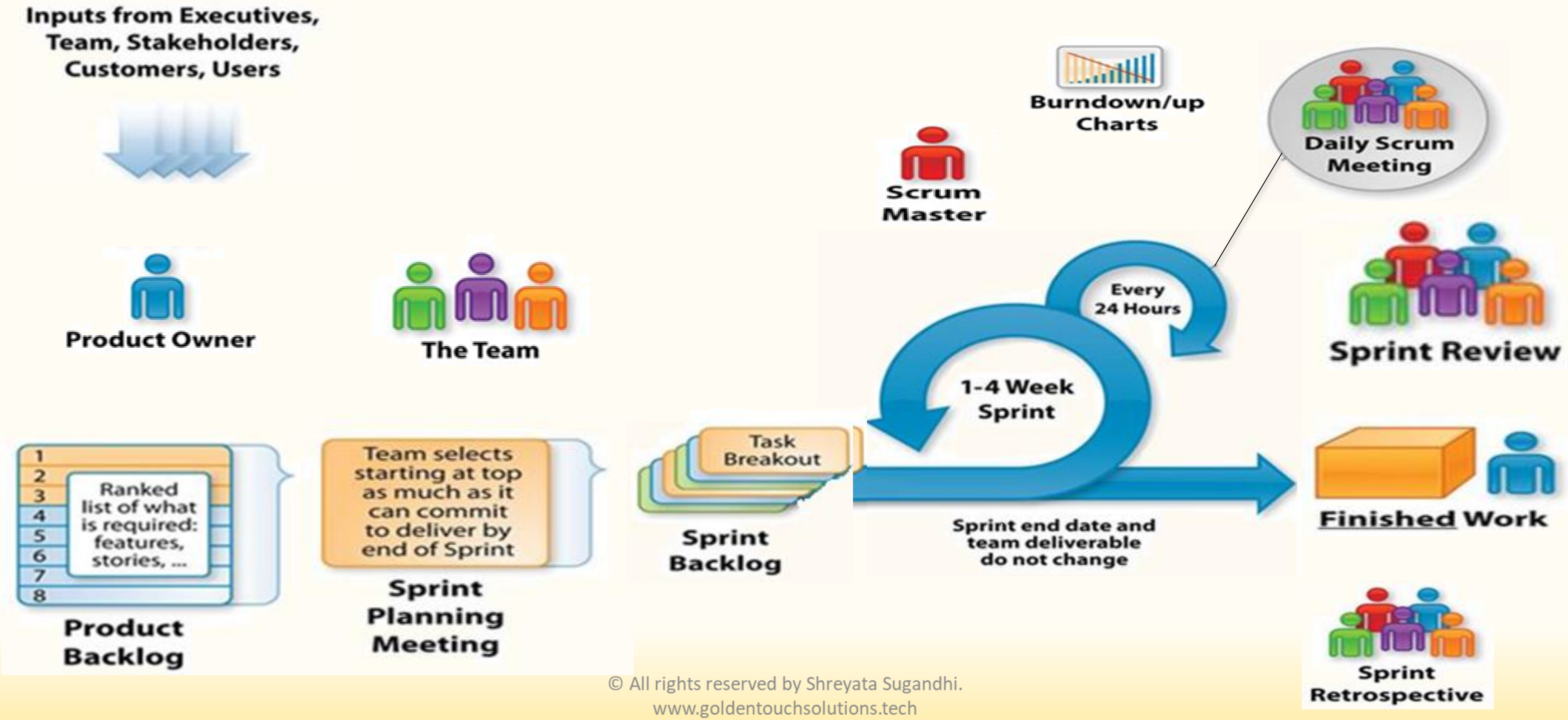
Benefits of Scrum

- Project deliverables are completed quickly and efficiently
- Time and money are used properly
- Projects are manageable since they're divided into smaller units called sprints
- Teams have greater visibility, thanks to scrum meetings and stand-up sessions
- There's constant feedback from customers and clients
- Individual efforts of the team members can be focused on

Scrum players

- Team includes -
  - **Scrum Master**
    - Master is responsible for setting up the team, sprint meeting and removes obstacles to progress
  - **Product owner**
    - The Product Owner creates product backlog, prioritizes the backlog and is responsible for the delivery of the functionality at each iteration
  - **Scrum Team**
    - Team manages its own work and organizes the work to complete the sprint or cycle

Scrum Cycle


Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Burndown/up Charts
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
Sprint Review
1
2
3
4
5
6
7
8
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change
Finished Work
Sprint Retrospective

Scrum Stand Up

Stand up

- Stand up is a status update meeting.
- Meeting should not last more 15 minutes
- Each team member is expected to answer 3 questions
    - What I have accomplished yesterday?
    - What I will accomplish today?
    - What are impediments?

Sprint Planning

- Sprint Planning meeting is conducted for backlog refinement, goal setting for the sprint and respective acceptance criteria
- It is time bound meeting
- Participants in the meeting are product owner, ScrumMaster and the entire Scrum team

Sprint Retrospective

- The retrospective includes three main questions for discussion:
  - What went well during the sprint cycle?
  - What went wrong during the sprint cycle?
  - What could we do differently to improve?

Sprint Review

- At the end of each sprint, a sprint review meeting is held.
- During this meeting, the Scrum team shows what they accomplished during the sprint.
- Participants of the sprint review includes
  - the product owner
  - the Scrum team
  - the Scrum Master
  - management
  - customers
  - developers from other projects.

User Stories

- **What?**
  - User stories are short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a end user or customer of the system.
  - It’s an end goal, not a feature, expressed from the software user’s perspective.
  - A user story is the smallest unit of work in an agile framework.

- **Why?**
  - to articulate how a piece of work will deliver a particular value back to the customer.
  - Stories keep the focus on the user
  - Stories enable collaboration
  - Stories drive creative solutions
  - Stories create momentum
  - Boosts Transparency

User Stories

- **How?**
  - Typically following a simple “Role – Goal – Value” template is used:

  As a < type of user >, I want < some goal > so that < some reason/achieve value >
As a user I want to register on the app so that I can log into.

  OR

  User stories can be written with the following input –

  - Outline subtasks or tasks
  - Definition of Done
  - User person as
  - Ordered Steps
  - Listen to feedback
  - Time

Definition of Done

- The Definition of Done (DoD) represents the organization's formal definition of quality for all Product Backlog Items (PBIs).
- If an organization does not have one, the Scrum team should set its own.
- The Definition of Done is the commitment contained within the Increment artifact.
- It is applied consistently and serves as an official gate separating things from being “in progress” to “done.”
- Typical definition of done consists of a checklist containing items such as:
  - Code is peer-reviewed
  - Code is checked in
  - Code is deployed to test environment
  - Code/feature passes unit test, smoke test, regression test
  - No critical or major defect are open
  - Code is documented
  - Help documentation is updated
  - Feature is OK’d by stakeholders

Agile Scrum Practices

- Pair programming
- Planning poker
- Refactoring (Keep It Small, Business Catalysts, Team Cohesion, Transparency)
- Scrum events (sprint planning, daily scrum, sprint review and retrospective)
- User Story Mapping
- Business analyst designer method (BADM)
- Cross-functional team
- Story-driven modeling
- Retrospective
- Velocity tracking
- Timeboxing

Estimates in Scrum

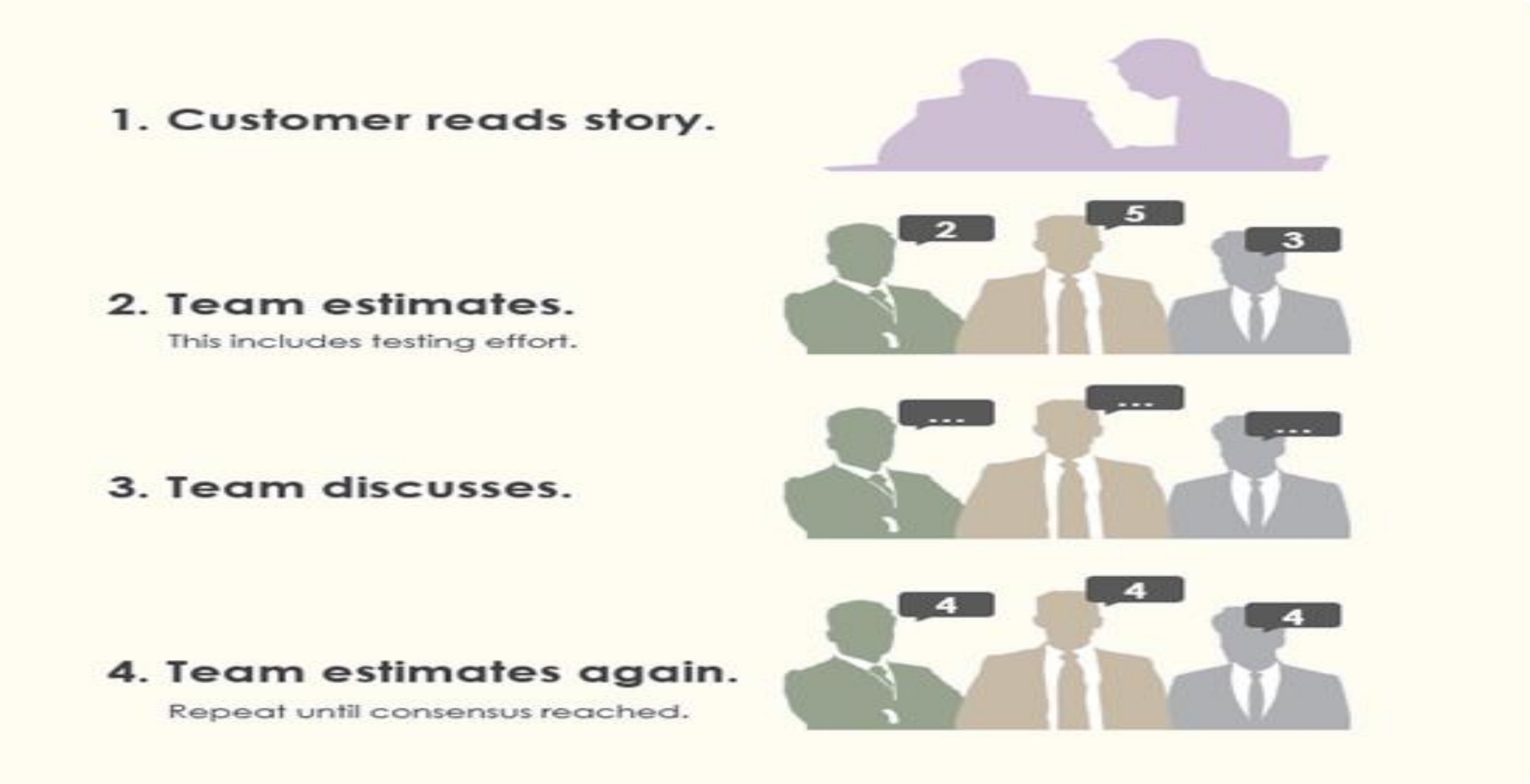
- Estimates in Scum will be done using story points
- Efforts will be measured into Story points as
  - 32
  - 16
  - 8
  - 4
  - 2
    - Whereas, generally story points are mapped to the time as

      2 story point = 0.5 hrs

      Or

      2 story points = 2 hrs

Planning Poker



1. Customer reads story.

2. Team estimates.
   This includes testing effort.

3. Team discusses.

4. Team estimates again.
   Repeat until consensus reached.

Planning Poker

5
5
5
5
Susan
Chris
Vadim
Ann

Planning Poker

- Planning Poker is an agile estimating and planning technique that is consensus based.
- The product owner or customer reads an agile user story or describes a feature to the estimators.
- Team members of the group make estimates by playing numbered cards face-down to the table without revealing their estimate (Fibonacci values: 1,2,3,5,8,13,20,40)
- Cards are simultaneously displayed
- The estimates are then discussed and high and low estimates are explained
- Repeat as needed until estimates converge

Planning Poker

Card(s)	Interpretation
0	Task is already completed.
1/2	The task is tiny.
1, 2, 3	These are used for small tasks.
5, 8, 13	These are used for medium sized tasks.
20, 40	These are used for large tasks.
100	These are used for very large tasks.
<infinity>	The task is huge.
?	No idea how long it takes to complete this task.
<cup of coffee>	I am hungry ☺

Agile Metrics

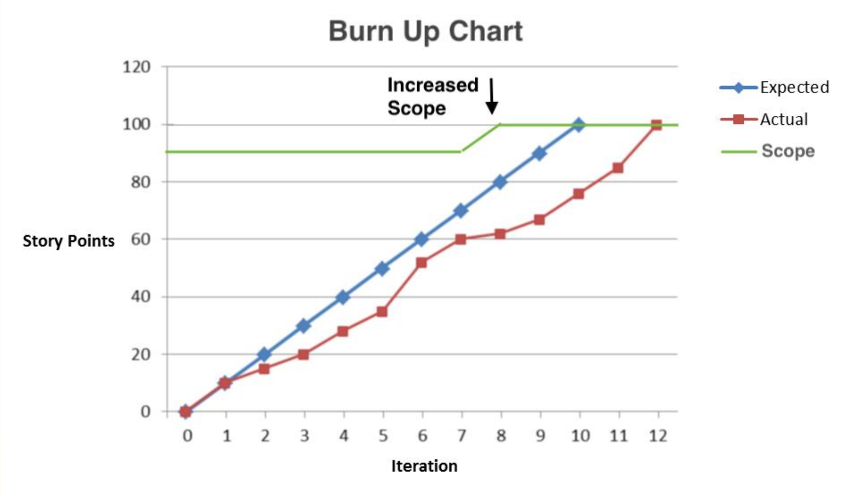
- Metrics that can be collected for effective usage of Agile is:
- Drag Factor
  - Effort in hours which do not contribute to sprint goal
  - Drag factor can be improved by reducing number of shared resources, reducing the amount of non-contributing work
  - New estimates can be increased by percentage of drag factor -New estimate = (Old estimate + drag factor)
- Velocity
  - Amount of backlog(user stories) converted to shippable functionality of sprint
- No of Unit Tests added
- Time interval taken to complete daily build
- Bugs detected in an iteration or in previous iterations
- Production defect leakage

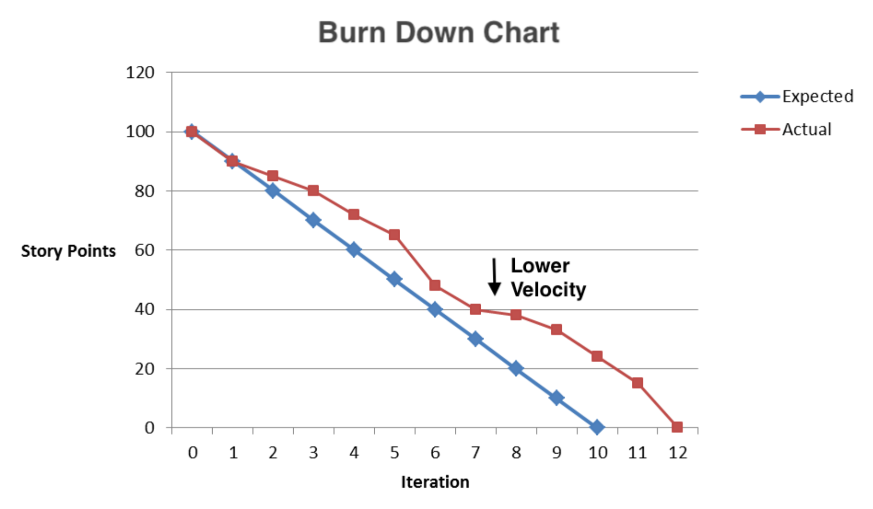
Velocity

- **Velocity** in agile is a measure of how much work an agile team can deliver on average in a sprint.
- Velocity helps to avoid overpromising on deliverables for a client
- Accurately calculated velocity will allow to determine the dates on which product milestones will be achieved or date of completion of milestone
- Estimates can be tracked by teams
- How fast team needs to move though the backlog to achieve sprint.
- Team can look at their progress and strengths and learn how to improve their metrics
- It takes a view at measuring:
  - How much work an agile team has delivered in the past sprints
  - How long it took the team to get the work done

Burn-up and Burn-down charts

- **Burn-up chart** –
  - Shows how much work is done vs story points in the Project
- **Burn-down chart** –
  - Shows how much work is remaining vs story points in the Project





QA Challenges in the form of Risks in Agile

- Minimal documentation, more changes of errors
- Lot of pressure on testers - Quick turn over, fast moving changes and new feature, minimal time
- Testers are often required to play a semi-developer role
- Highly compressed test cycles
- Less time for Test Planning
- Change in Role from being a gate-keeper of quality to being a partner in Quality
- Requirement changes and updates are inherent in an agile
- Overloaded automation and manual testing

Skills Needed in Agile

- Be positive and solution-oriented with team members and stakeholders
- Display critical, quality-oriented, skeptical thinking about the product
- Actively acquire information from stakeholders (rather than relying entirely on written specifications)
- Accurately and timely evaluate and report test results, progress, and product quality
- Sharing and supporting team members
- Work effectively to define testable user stories, especially acceptance criteria, with customer representatives and stakeholders
- Collaborate within the team, working in pairs with programmers and other team members
- Respond to change quickly, including changing, adding, or improving test cases
- Plan and organize own work

Some more critical skills required in Agile

- Cross-functional
- Self-organizing
- Co-located
- Collaborative
- Empowered
- Committed
- Transparent
- Credible
- Open to feedback
- Resilient

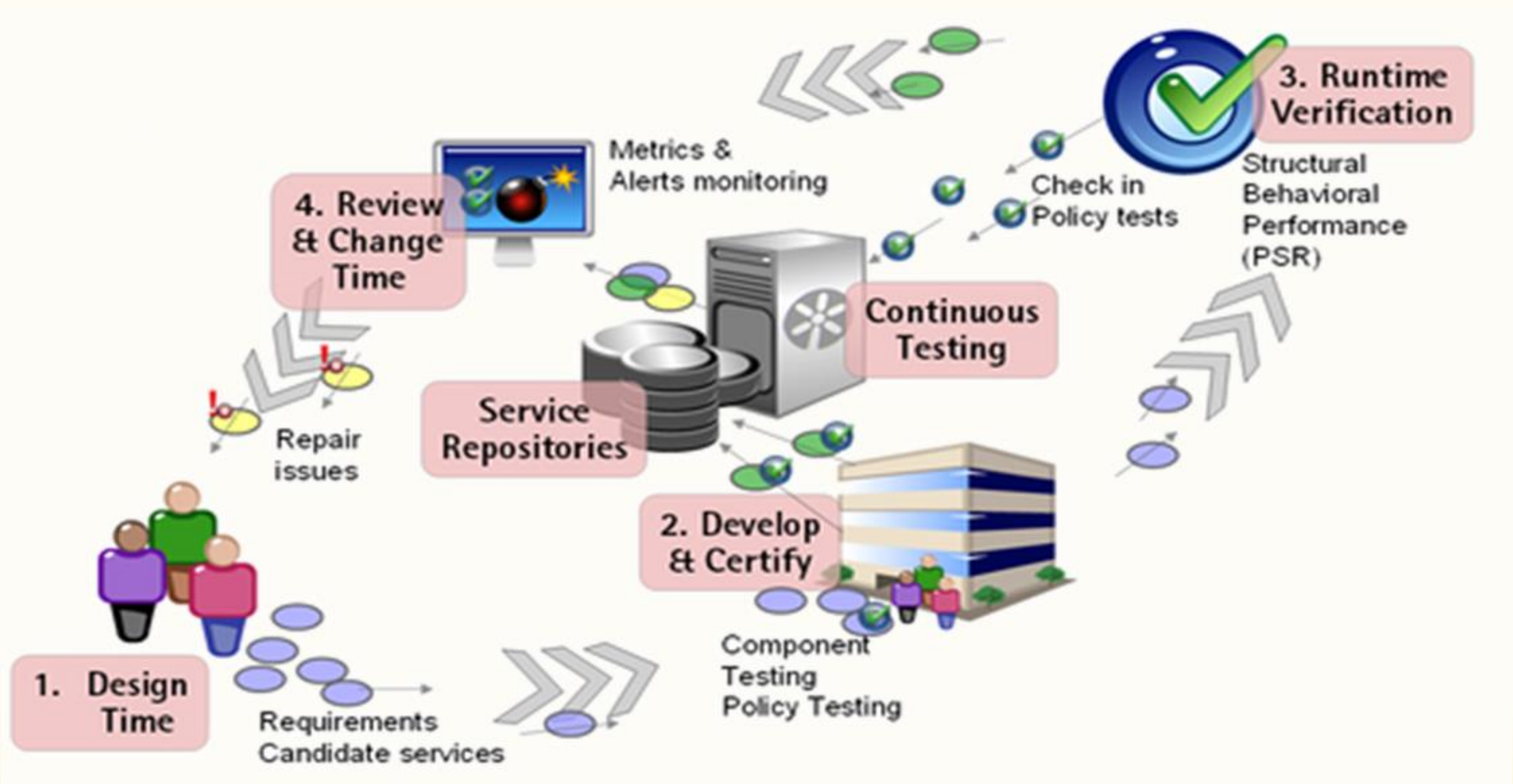
The Role of a Tester in an Agile Team

- Understanding, implementing, and updating the test strategy
- Measuring and reporting test coverage across all applicable coverage dimensions
- Ensuring proper use of testing tools
- Configuring, using, and managing test environments and test data
- Reporting defects and working with the team to resolve them
- Coaching other team members in relevant aspects of testing
- Ensuring the appropriate testing tasks are scheduled during release and iteration planning
- Actively collaborating with developers and business stakeholders to clarify requirements, especially in terms of testability, consistency, and completeness
- Participating proactively in team retrospectives, suggesting and implementing improvements

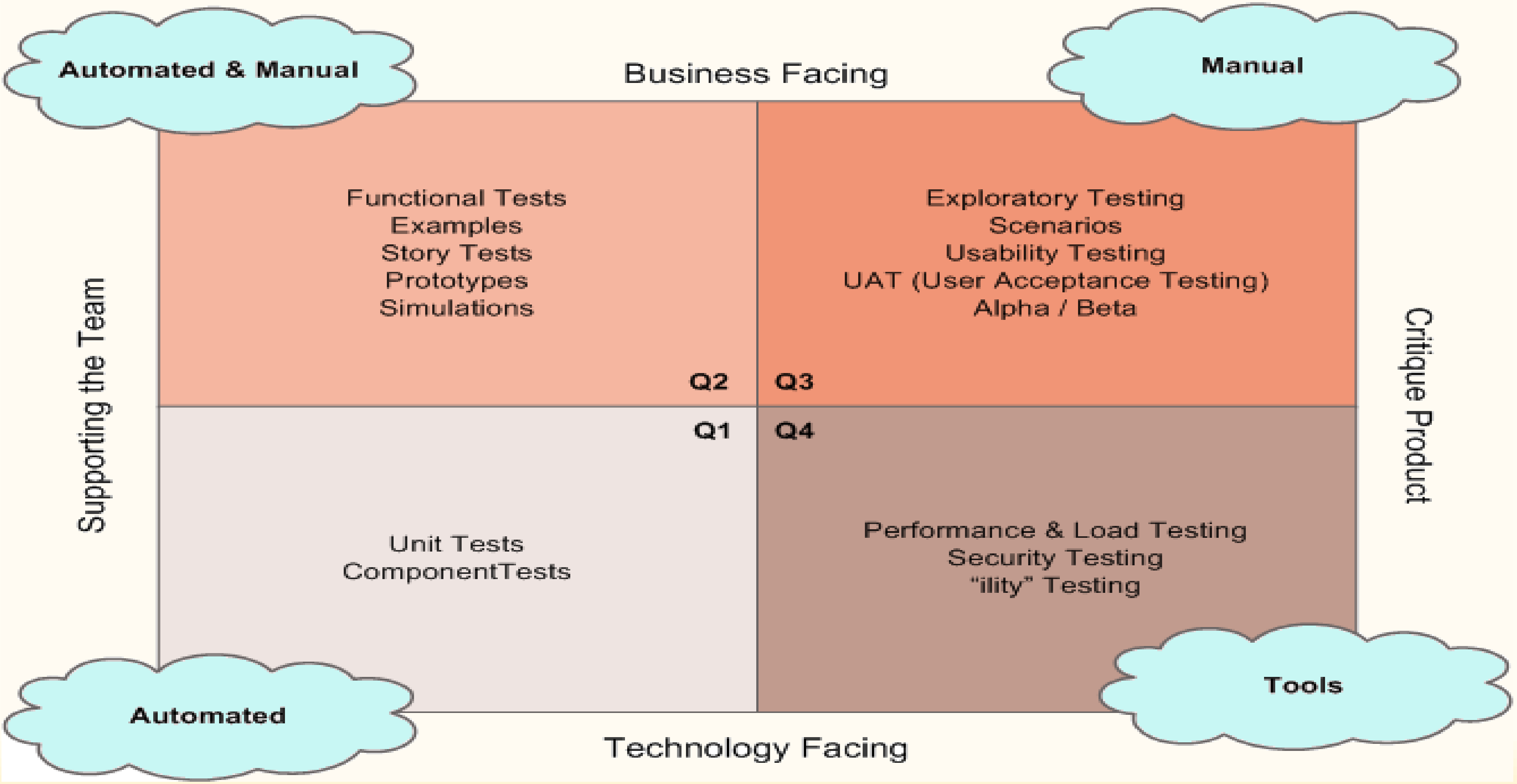
Agile Testing Practices

- Acceptance test-driven development (ATDD)
- Test-driven development (TDD)
- Behavior-driven development (BDD)
- Continuous integration (CI)
- Sprint 0 technique
- Agile testing
- Agile modeling
- User story
- Backlogs (Product and Sprint)
- Domain-driven design (DDD)
- Information radiators (scrum board, task board, visual management board, burndown chart)
- Iterative and incremental development (IID)

Agile Testing Life Cycle



Agile Testing Quadrants



Automated & Manual

Business Facing

Manual

Supporting the Team

Functional Tests
Examples
Story Tests
Prototypes
Simulations

Q2

Exploratory Testing
Scenarios
Usability Testing
UAT (User Acceptance Testing)
Alpha / Beta

Q3

Critique Product

Q1

Unit Tests
ComponentTests

Q4

Performance & Load Testing
Security Testing
"ility" Testing

Automated

Technology Facing

Tools

Risks in Agile

- What risks exist for the client using Agile/Scrum?
  - Budget risks
  - Scope creep risk
  - Not sticking to Agile principles
  - Agile is not suitable for a project
  - Less predictability
- What risks exist for the vendor using Agile/Scrum?
  - Lack of Staff Knowledge
  - A sudden stop of work
  - Technical debt

Thank You