Agile

Voronezh, 2018

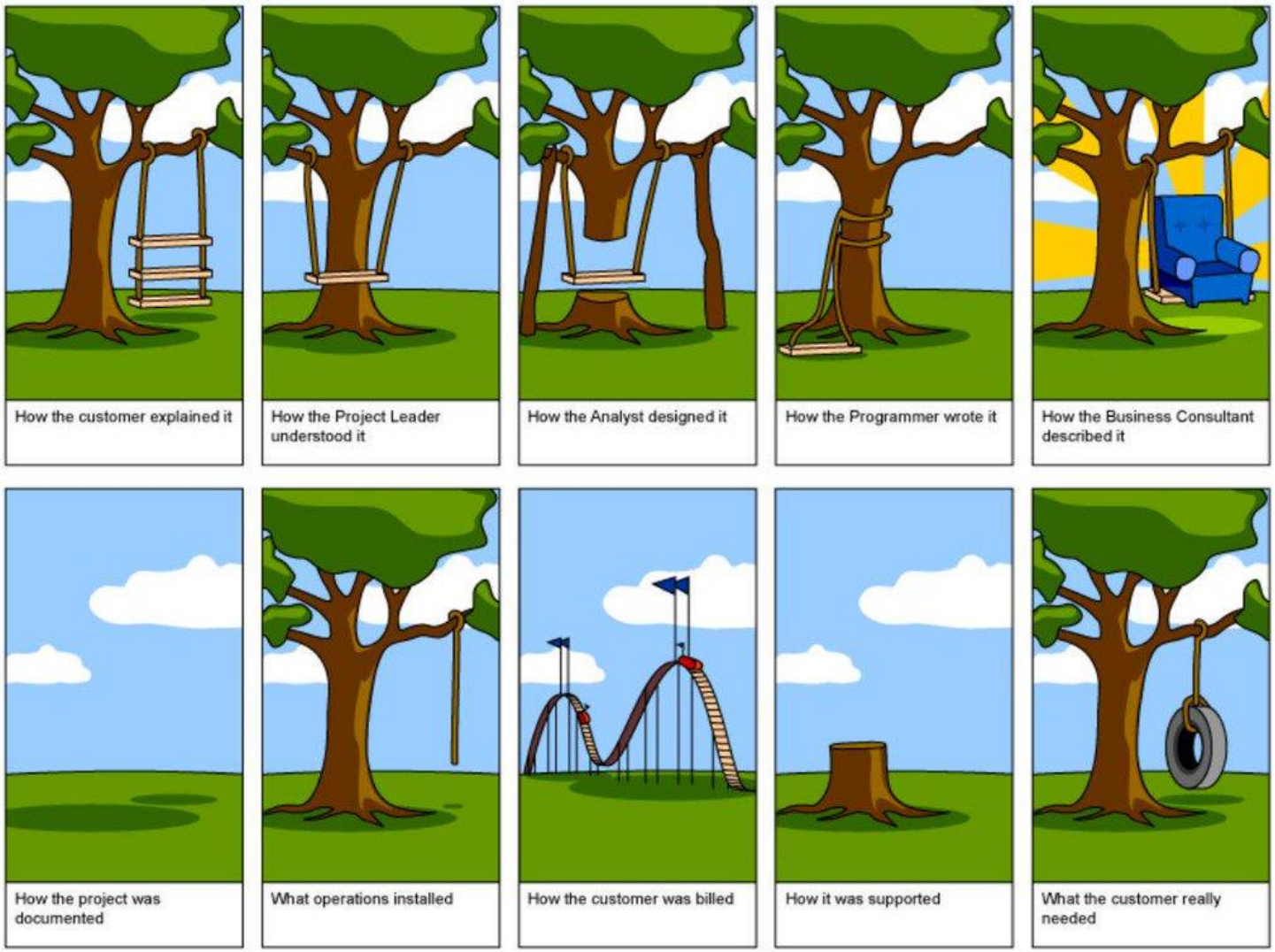
T · · Systems

Agenda

- Waterfall
- Agile
- Agile Methodologies
- Kanban
- Scrum

T · · Systems

Real Life Case



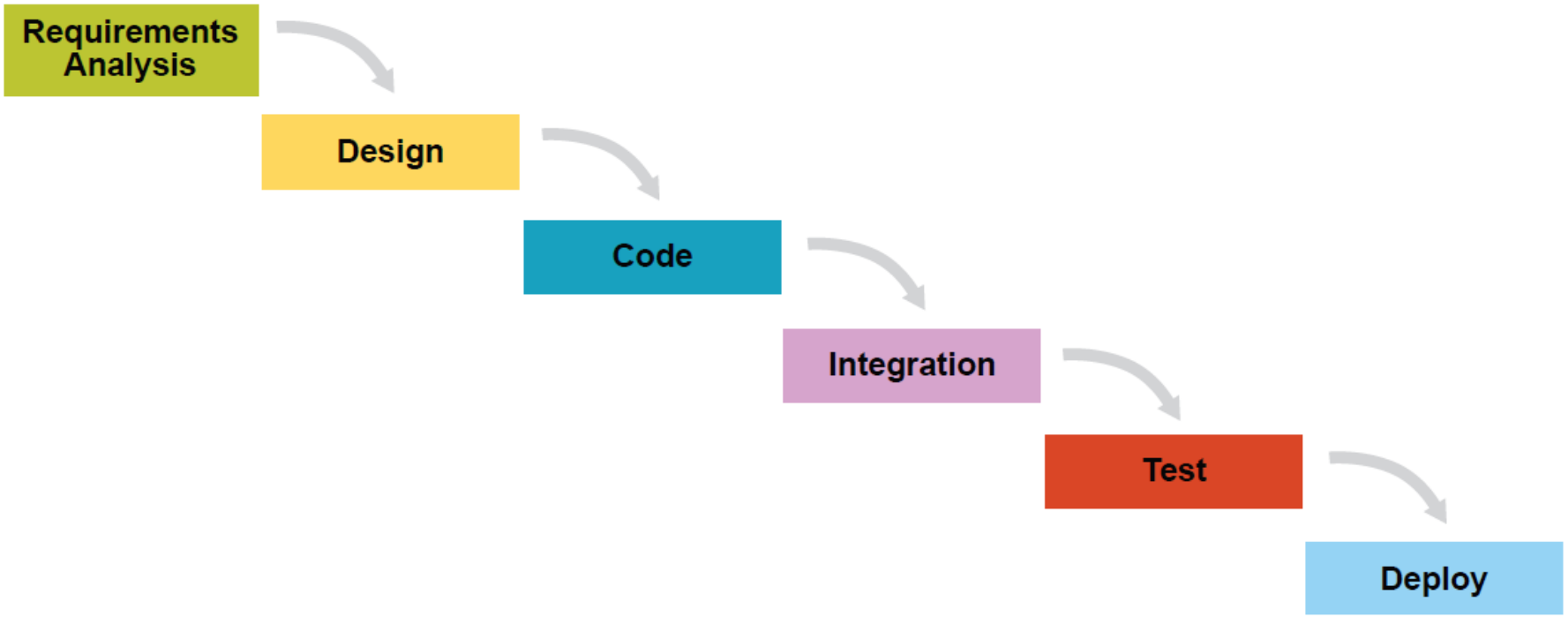
T · · Systems

Waterfall

- **Waterfall** methodology follows a **sequential**, **linear process** and is the most popular version of the systems development life cycle for software engineering and IT projects.
- Once one of the eight stages are complete, the development team moves onto the next step.
- The team can't go back to a previous stage without starting the whole process from the beginning.
- And, before the team can move to the next stage, requirements may need to be reviewed and approved by the customer.
- The Waterfall model originated in the manufacturing and construction industries.
- The first formal description of Waterfall is attributed to Winston W. Royce in a 1970 article where he described a flawed software model.

T · · Systems

Waterfall: Stages



T · · Systems

Waterfall: Pros & Cons

- **Advantages**:
  - Easy to use and manage.
  - Discipline is enforced.
  - Requires a well documented approach.
- **Disadvantages**:
  - Changes can't be easily accommodated.
  - Software isn't delivered until late.
  - Gathering accurate requirements can be challenging.

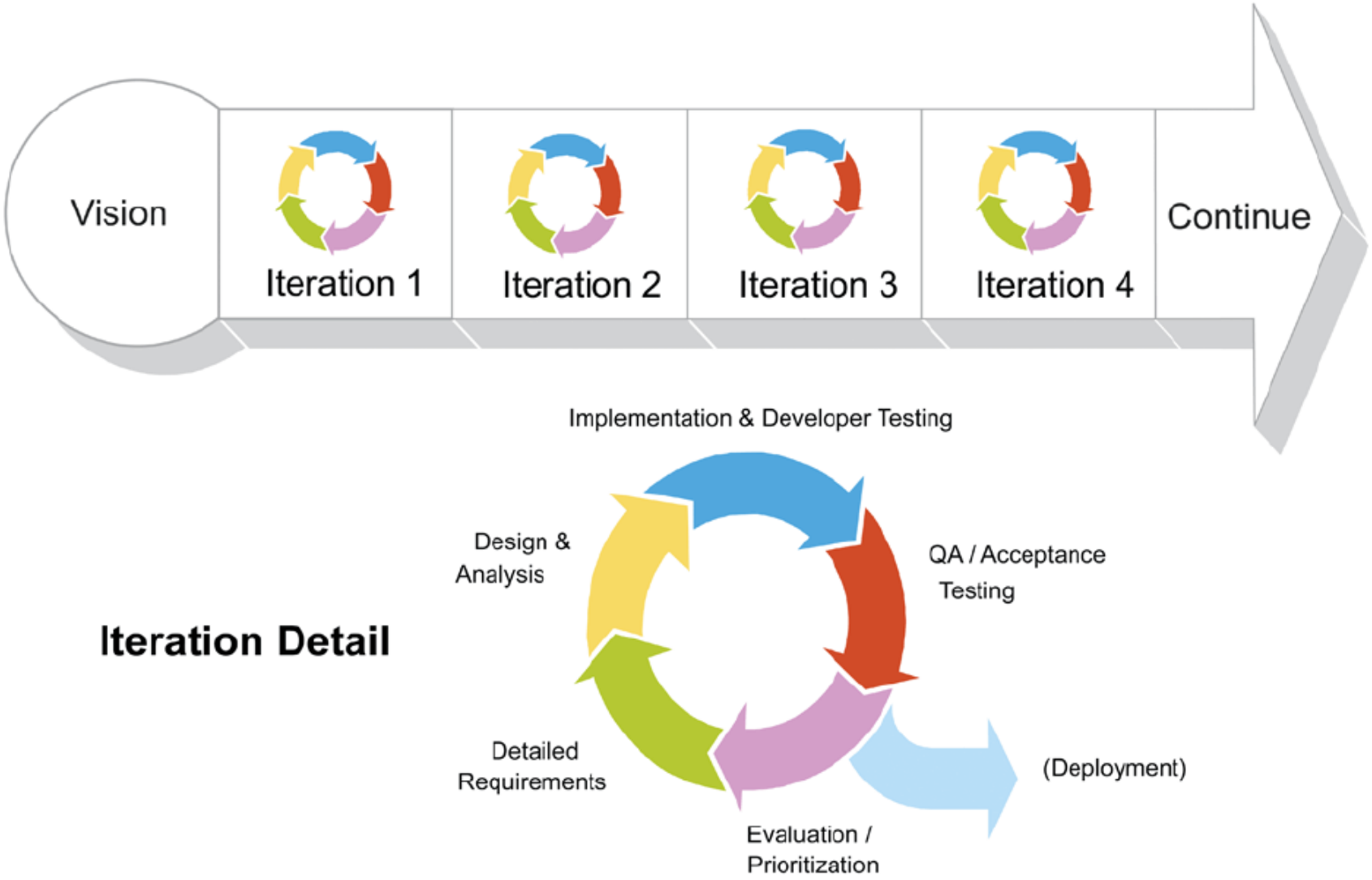
T · · Systems

Agile

- **Agile** software development is based on an **incremental, iterative approach**.
- Agile methodologies **are open to changing requirements** over time and **encourages constant feedback** from the end users.
- **Cross-functional teams** work on iterations of a product over a period of time, and this work is organized into a backlog that is prioritized based on business or customer value.
- The **goal** of each **iteration** is to **produce** a **working product**.
- In Agile methodologies, **leadership encourages teamwork**, **accountability**, and **face-to-face communication**.
- **Business** stakeholders and **developers** must **work together** to align the product with customer needs and company goals.

T · · Systems

Agile: Development Cycle



Iteration Detail

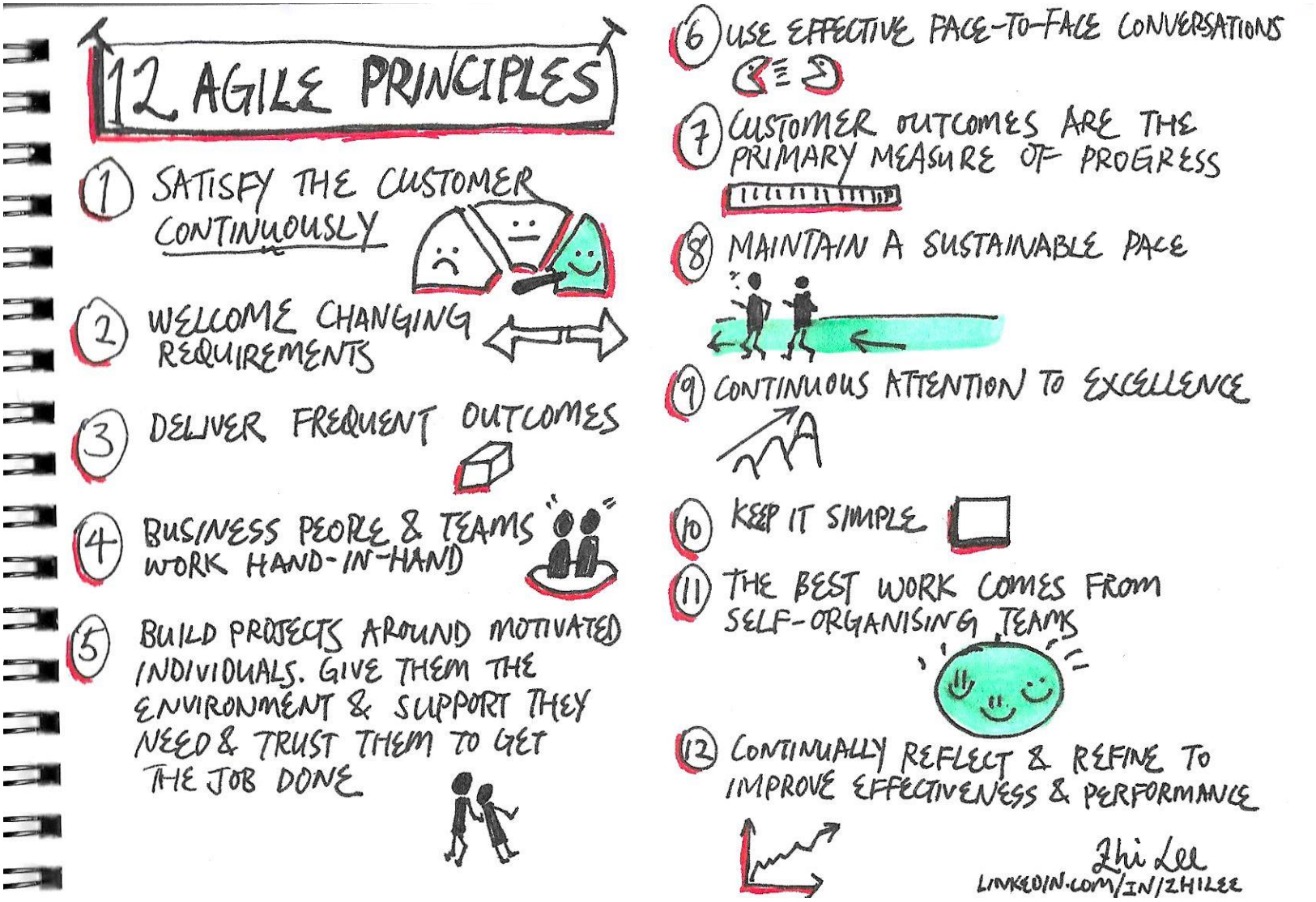
T · · Systems

Agile: Manifesto (1)

- **Agile** refers to **any process** that **aligns with** the concepts of the **Agile Manifesto**.
- In February 2001, 17 software developers met in Utah to discuss lightweight development methods.
- They published the Manifesto for Agile Software Development, which covered how they found “better ways of developing software by doing it and helping others do it” and included four values and 12 principles.

Agile: Manifesto (2)

Agile: Principles

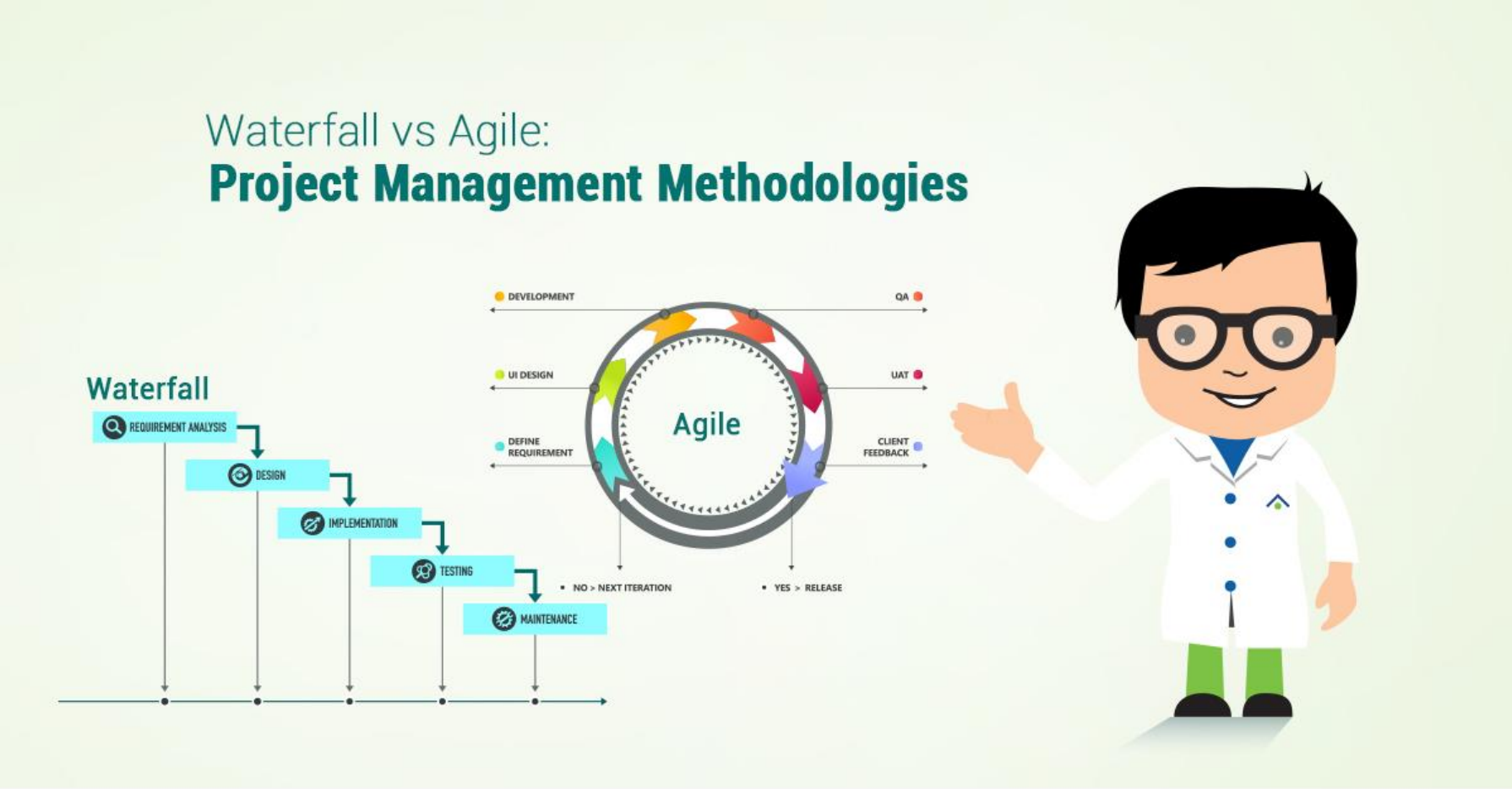


Agile: Pros & Cons

- **Advantages**:
  - Change is embraced.
  - End-goal can be unknown.
  - Faster, high-quality delivery.
  - Strong team interaction.
  - Customers are heard.
  - Continuous improvement.

- **Disadvantages:**
  - Planning can be less concrete.
  - Team must be knowledgeable.
  - Time commitment from developers.
  - Documentation can be neglected.
  - Final product can be very different.

Waterfall vs. Agile (1)



Waterfall vs. Agile (2)

	Waterfall	Agile
Sequential	X	
Flexible		X
Accommodates change		X
Defined requirements	X	
Deliver quality products	X	X
Continually evolving		X
Rigid process	X	

T · · Systems

When to Use Waterfall or Agile?

- If you anticipate or expect any changes throughout the project, go with Agile. If you know the project is fixed, unchanging, and predictable, Waterfall may be a better choice.

- Waterfall if:
  - You don't expect changes in scope and you're working with fixed-price contracts
  - The project is very simple or you've done it many times before
  - Requirements are very well known and fixed
  - Customers know exactly what they want in advance
  - You're working with orderly and predictable projects

- Agile if:
  - The final product isn't clearly defined
  - The clients/stakeholders need the ability to modify the scope
  - You anticipate any kind of changes during the project
  - Rapid deployment is the goal

T · · Systems

Agile Methodologies (1)

- **Extreme Programming (XP):**
  - Also known as XP, Extreme Programming is a type of software development intended to improve quality and responsiveness to evolving customer requirements.
  - The principles of XP include feedback, assuming simplicity, and embracing change.

- **Pair programming:**
  - Also known as “pairing” is part of the Extreme Programming (XP) practices.
  - It is when two programmers share a single workstation, which includes sharing one screen, keyboard, and mouse.
  - The purpose of this technique is to encourage better communication, clarification of the problem, and understanding of the solution.
  - Pairing is often used in Agile projects to quickly deliver high-quality products.

Agile Methodologies (2)

- **Feature-driven development (FDD):**
  - This iterative and incremental software development process blends industry best practices into one approach.
  - There are five basic activities in FDD: develop overall model, build feature list, plan by feature, design by feature, and build by feature.

- **Adaptive system development (ASD):**
  - Adaptive system development represents the idea that projects should always be in a state of continuous adaptation.
  - ASD has a cycle of three repeating series: speculate, collaborate, and learn.

- **Scrum** is one of the most popular ways to implement Agile.
  - It is an iterative software model that follows a set of roles, responsibilities, and meetings that never change.
  - Sprints, usually lasting one to two weeks, allow the team to deliver software on a regular basis.

Agile Methodologies (3)

- **Dynamic Systems Development Method (DSDM):**
  - This Agile project delivery framework is used for developing software and non-IT solutions.
  - It addresses the common failures of IT projects, like going over budget, missing deadlines, and lack of user involvement.
  - The eight principles of DSDM are: focus on the business need, deliver on time, collaborate, never compromise quality, build incrementally from firm foundations, develop iteratively, communicate continuously and clearly, and demonstrate control.

- **Lean Software Development (LSD):**
  - Lean Software Development takes Lean manufacturing and Lean IT principles and applies them to software development.
  - It can be characterized by seven principles: eliminate waste, amplify learning, decide as late as possible, deliver as fast as possible, empower the team, build integrity in, and see the whole.

Agile Methodologies (4)

- **Kanban:**
  - Kanban, meaning “visual sign” or “card” in Japanese, is a visual framework to implement Agile.
  - It promotes small, continuous changes to your current system.
  - Its principles include: visualize the workflow, limit work in progress, manage and enhance the flow, make policies explicit, and continuously improve.

- **Crystal Clear:**
  - Crystal Clear is part of the Crystal family of methodologies.
  - It can be used with teams of six to eight developers and it focuses on the people, not processes or artifacts.
  - Crystal Clear requires the following: frequent delivery of usable code to users, reflective improvement, and osmotic communication preferably by being co-located.

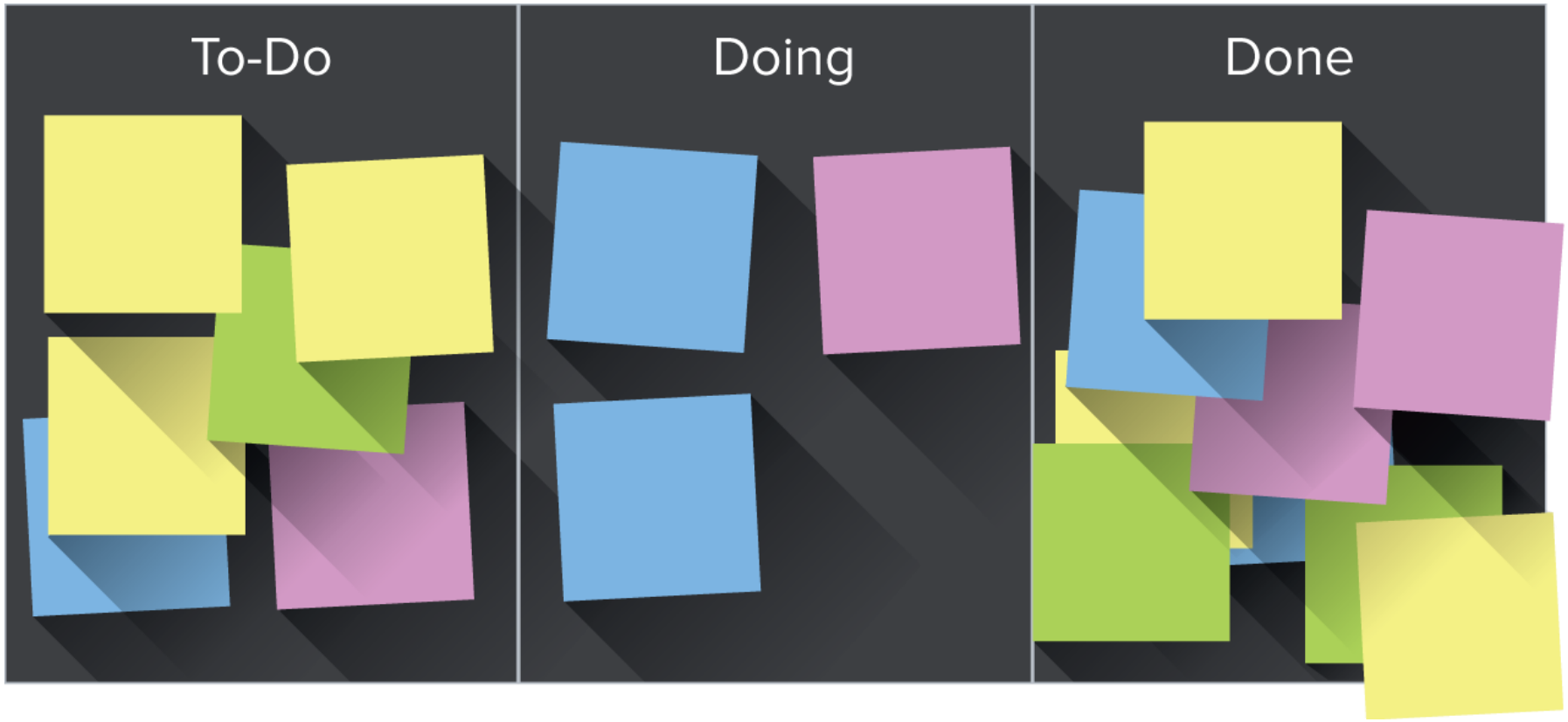
Kanban

- **Kanban** is Japanese for “visual sign” or “card.”
  - It is a visual framework used to implement Agile that shows what to produce, when to produce it, and how much to produce.
  - It encourages small, incremental changes to your current system and does not require a certain set up or procedure.

- Kanban was inspired by the Toyota Production System and Lean Manufacturing in the 1940s. Toyota improved its engineering process by modeling it after how supermarkets stock shelves.

- These same ideas apply to software teams and IT projects today: development work-in-progress (WIP) takes the place of inventory, and new work can only be added when there is an “empty space” on the team’s visual Kanban board.
- Kanban matches the amount of WIP to the team’s capacity, improving flexibility, transparency, and output

Kanban: Board



T · · Systems

Kanban: Pros & Cons

- **Advantages**:
  - Increases flexibility.
  - Reduces waste.
  - Easy to understand.
  - Improves delivery flow.
  - Minimizes cycle time.

- **Disadvantages**:
  - Outdated board can lead to issues.
  - Teams can overcomplicate the board.
  - Lack of timing.

Kanban: When to Use?

- Your project doesn't require iterations
- You want the ability to release at any time
- Your team prefers incremental change
- Your team works well with visuals
- You want to improve delivery flow
- You're looking for an easy-to-understand system

T · · Systems

Kanban vs. Agile

	Kanban	Agile
Continuous flow	X	
Iterations		X
Visualization	X	
Cross-functional teams		X
Equally beneficial for all industries	X	
Philosophy		X
Continuous improvement	X	X
Transparency	X	X
Don't require upfront planning	X	X
Faster delivery	X	X
Breaks projects into smaller chunks	X	X

T · · Systems

Scrum

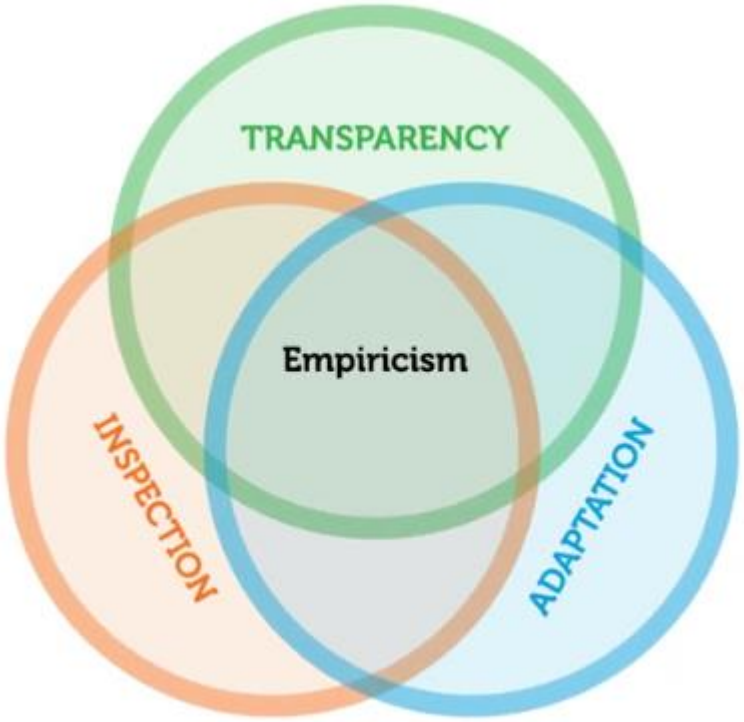
- **Scrum** (n): A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.

- **Scrum is**:
  - Lightweight
  - Simple to understand
  - Difficult to master

- The main document is **The Scrum Guide™**
  - https://www.scrumguides.org/scrum-guide.html

- Jeff Sutherland created the Scrum process in 1993, taking the term “Scrum” from an analogy in a 1986 study by Takeuchi and Nonaka published in the Harvard Business Review.
- In the study, Takeuchi and Nonaka compare high-performing, cross-functional teams to the Scrum formation used by Rugby teams.

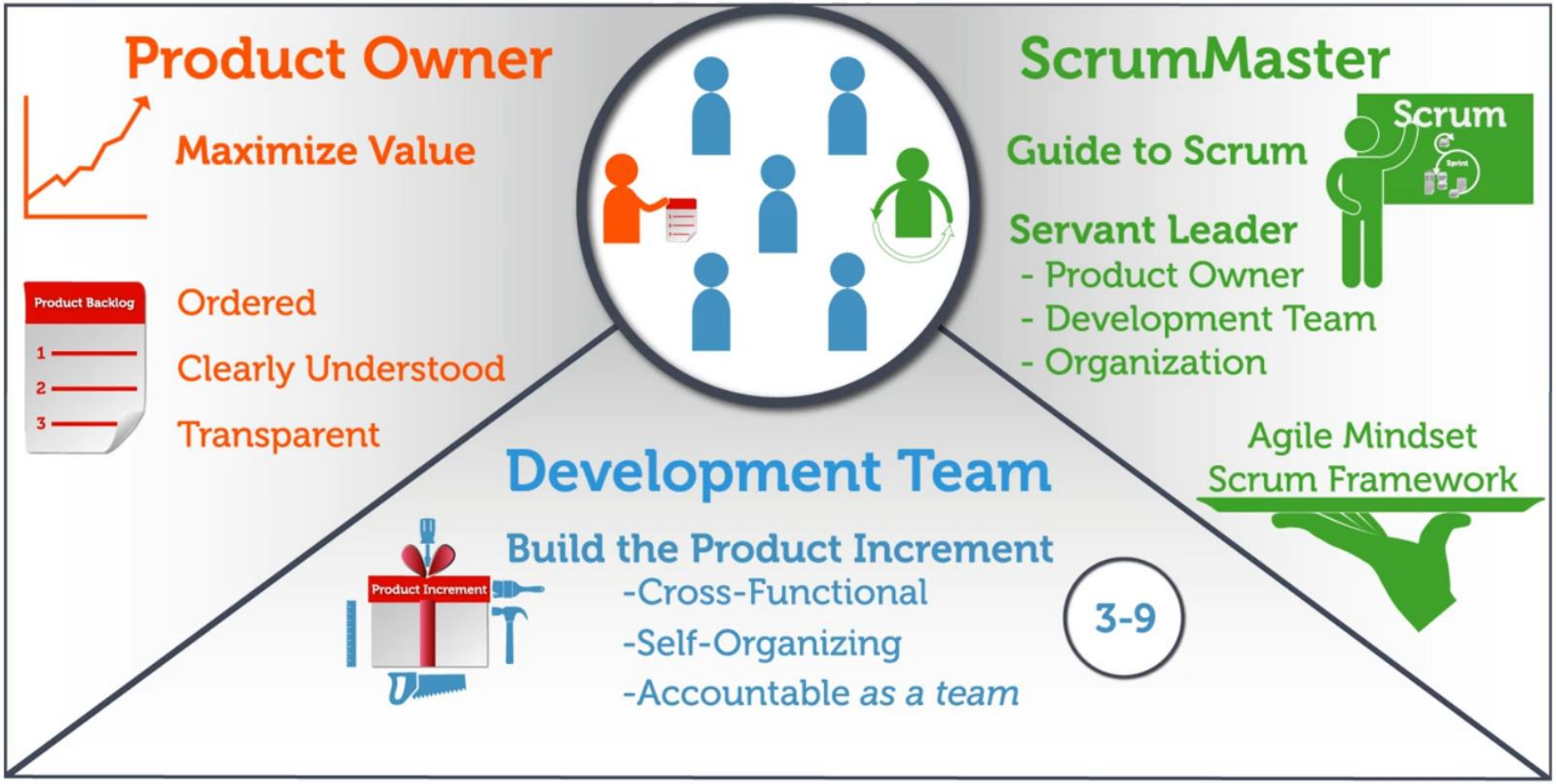
Scrum: Pillars



T · · Systems

Scrum: Values

Scrum: Roles



T · · Systems

Scrum: Artifacts

T · · Systems

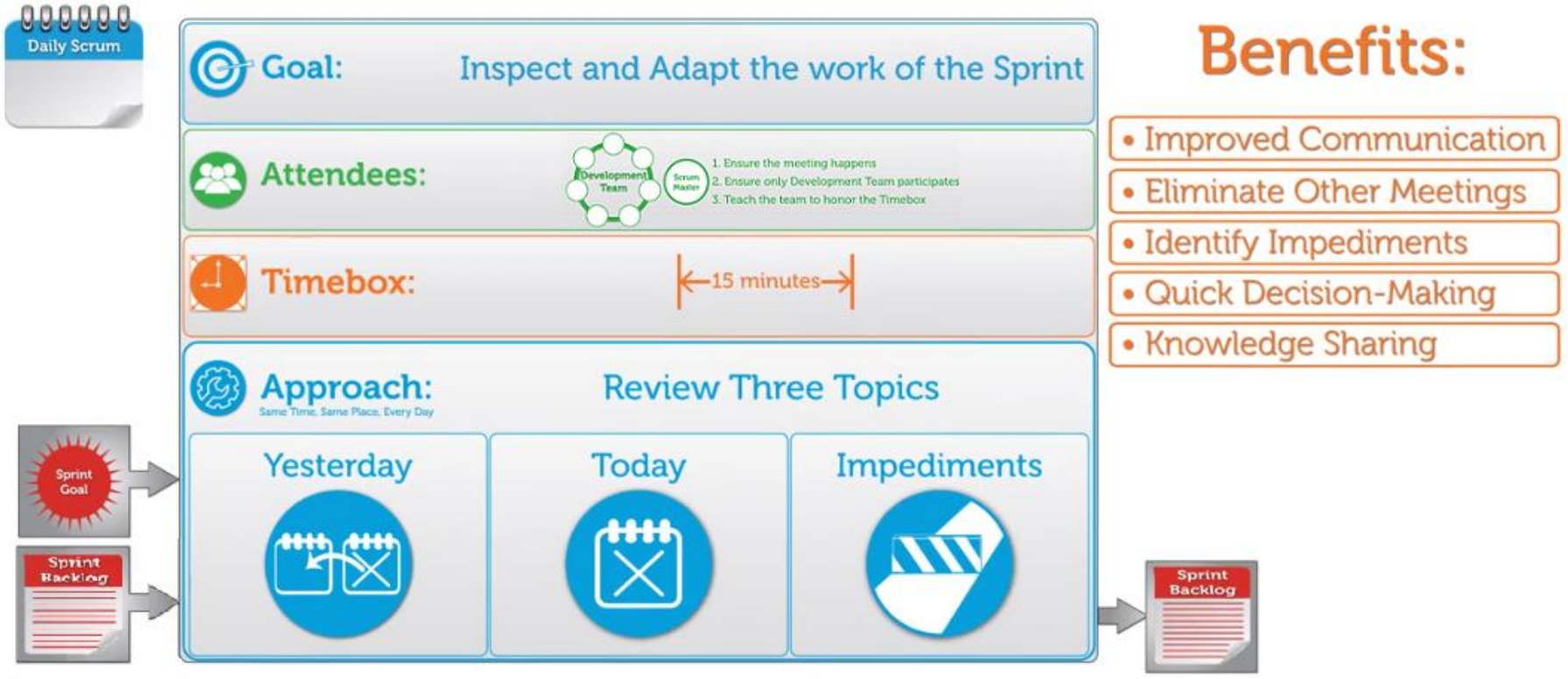
Scrum: Sprint

T · · Systems

Scrum: Sprint Planning

T · · Systems

Scrum: Daily

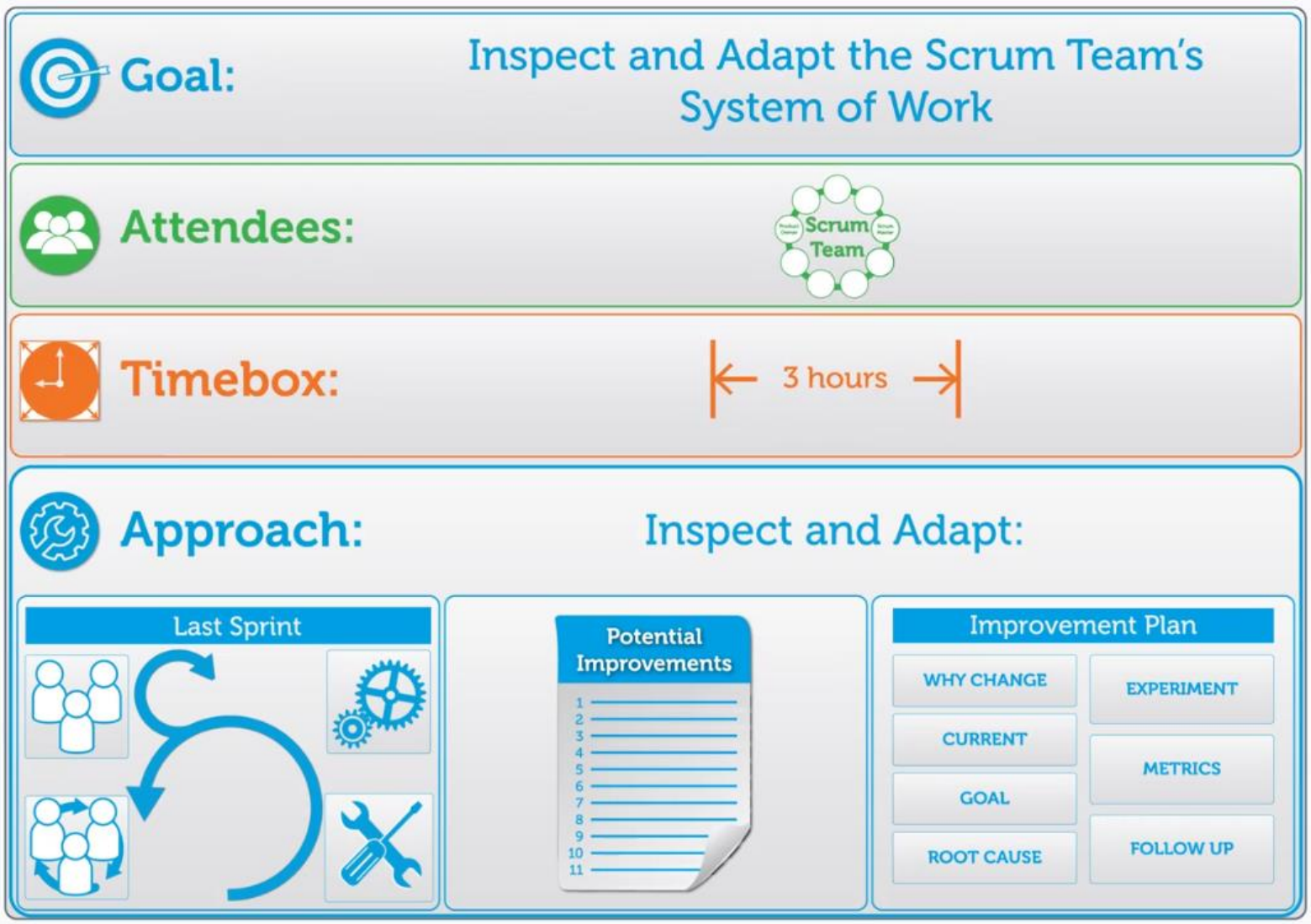


T · · Systems

Scrum: Sprint Review

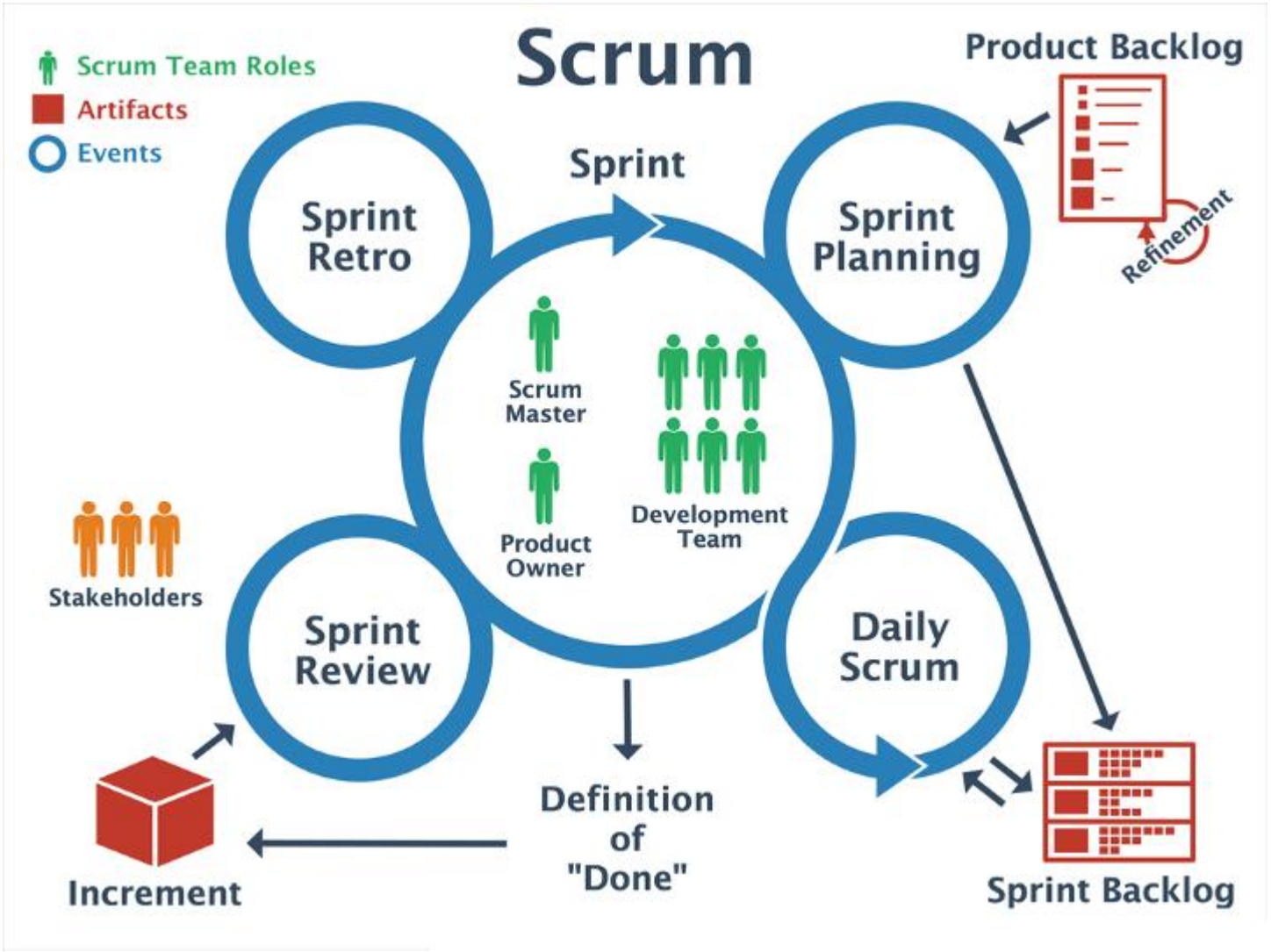
T · · Systems

Scrum: Sprint Retrospective



T · · Systems

Scrum



T · · Systems

Scrum: Pros & Cons

- **Advantages**:
  - Change is embraced.
  - End-goal can be unknown.
  - Faster, high-quality delivery.
  - Strong team interaction.
  - Customers are heard.
  - Continuous improvement.

- **Disadvantages**:
  - Planning can be less concrete.
  - Team must be knowledgeable.
  - Time commitment from developers.
  - Documentation can be neglected.
  - Final product can be very different.

Scrum: When to Use?

- Scrum works well for projects that have a lot of unknowns or that evolve over time.
- Scrum deals with these changes very effectively, so you can easily accommodate new information or features throughout the process.
  - The project requirements will change and evolve
  - Continuous feedback is required
  - You have to figure out how to do a large part of the work because you haven't done it before
  - You don’t need to commit to a fixed release date
  - The project team wants autonomy
  - You need to deliver software on a regular basis

T · · Systems

Scrum vs. Agile

	Scrum	Agile
Philosophy		X
Methodology	X	
Adds process	X	
Transparency	X	X
Deliver software early and often	X	X
Iterative	X	X
Accommodates change	X	X
Continuous improvement	X	X

T · · Systems

Scrum vs. Kanban

	Scrum	Kanban
Specific roles	X	
Timeboxed iterations	X	
Accomodates change		X
Estimation	X	
Empirical	X	X
Lean and agile	X	X
Limits WIP	X	X
Work can be done simulatenously	X	X
Board is continuously used		X
Teams must be cross functional	X	
Pull scheduling	X	X
Transparency	X	X
Deliver software early and often	X	X

T · · Systems

Thank you for your attention!

T Systems