# Lesson 4
## Implementing an Agile Release Train

Day 1

1. Introducing the Scaled Agile Framework
2. Embracing a Lean-Agile Mindset

   Break
3. Understanding SAFe Principles

   Lunch
4. Implementing an Agile Release Train

   Break
5. Experiencing PI Planning

Day 2

6. Executing and Releasing Value

   Break
7. Building an Agile Portfolio

   Lunch
8. Building Really Big Systems

   Break
9. Leading the Lean-Agile Enterprise




## Learning objectives

4.1 Synchronize development with the Agile Release Train

4.2 Organize Agile Teams and implement key Agile Release Train roles

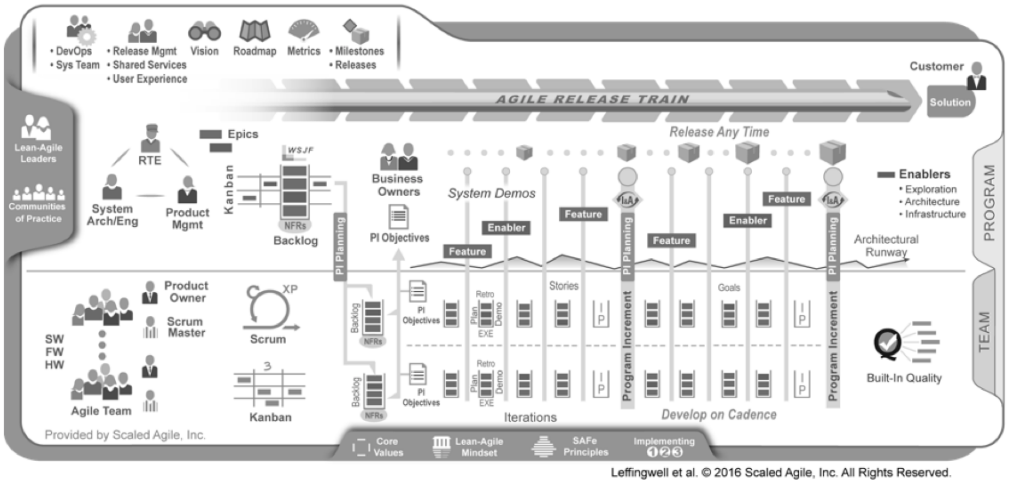
4.3 Prepare to experience PI Planning

# 4.1 Synchronize development with the Agile Release Train



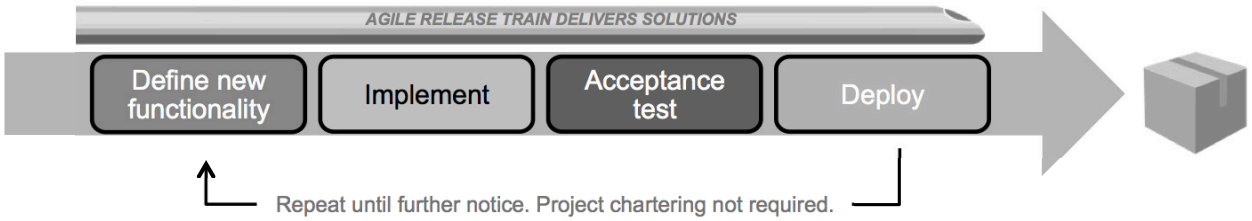
## Agile Release Trains deliver Solutions

A long-lived, self-organizing team of Agile Teams



# The Agile Release Train

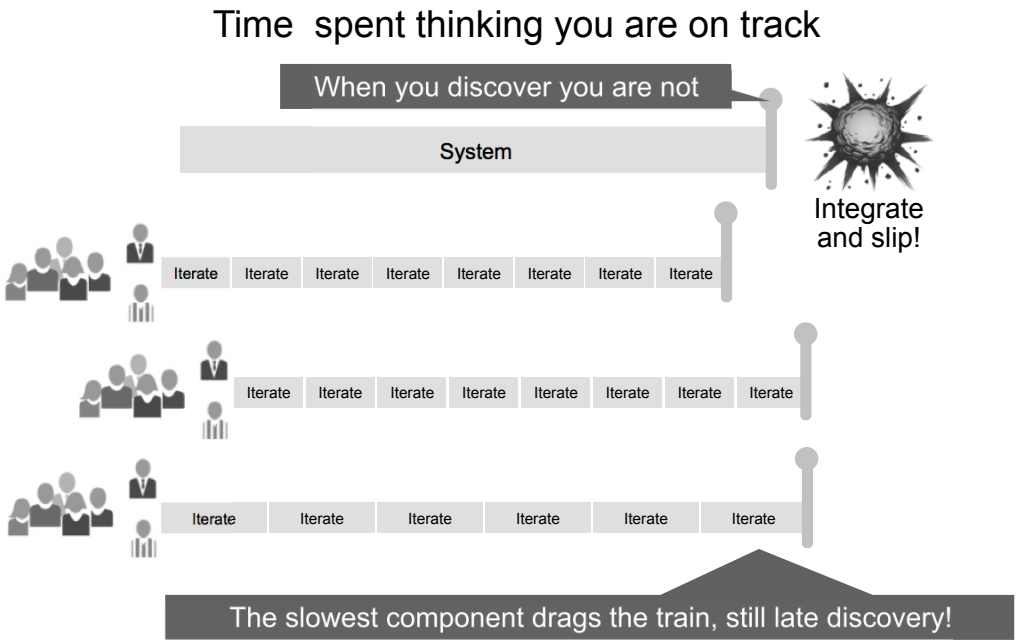
- A virtual organization of 5 – 12 teams (50 – 125+ individuals) that plans, commits, and executes together
- Program Increment (PI) is a fixed timebox; default is 10 weeks
- Synchronized Iterations and PIs
- Aligned to a common mission via a single Program Backlog
- Operates under architectural and UX guidance
- Frequently produces valuable and evaluable system-level Solutions



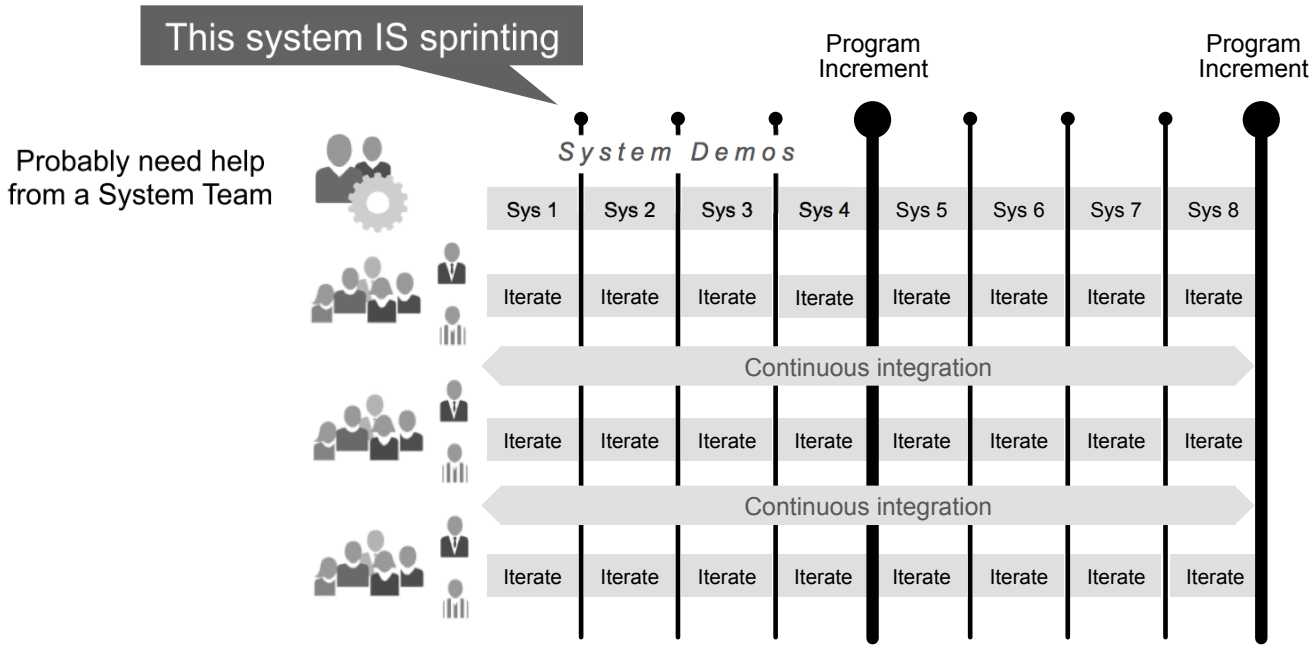




# Cadence without synchronization is not enough



# Synchronize to assure delivery

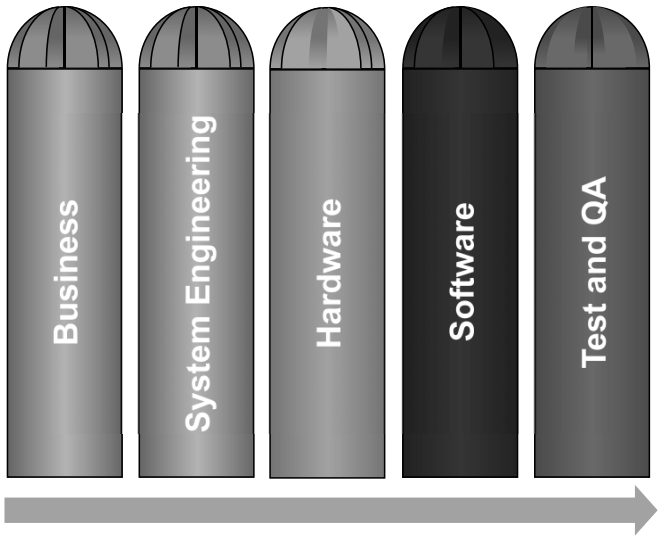







# 4.2 Organize Agile Teams and implement key Agile Release Train roles

# Value doesn't follow silos



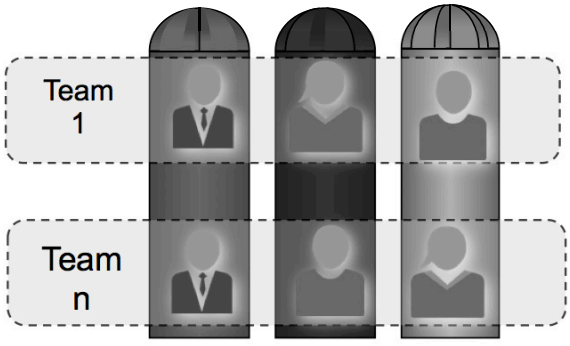

Management challenge:
Connect the silos

- Optimized for vertical communication
- Friction across the silos
- Location via function
- Political boundaries between functions




# Build cross-functional Agile Teams

- Cross-functional, self-organizing entities that can define, build and test a feature or component
- Optimized for communication and delivery of value
- Deliver value every two weeks



# Agile Teams power the train

## Five to nine team members, three Scrum roles

**Scrum Master**

- Runs team meetings, drives agile behavior
- Removes impediments; protects the team from outside influence
- Attends Scrum of Scrum meetings

**Product Owner**

- Defines and accepts stories
- Acts as the customer for developer questions
- Works with product management to plan PIs

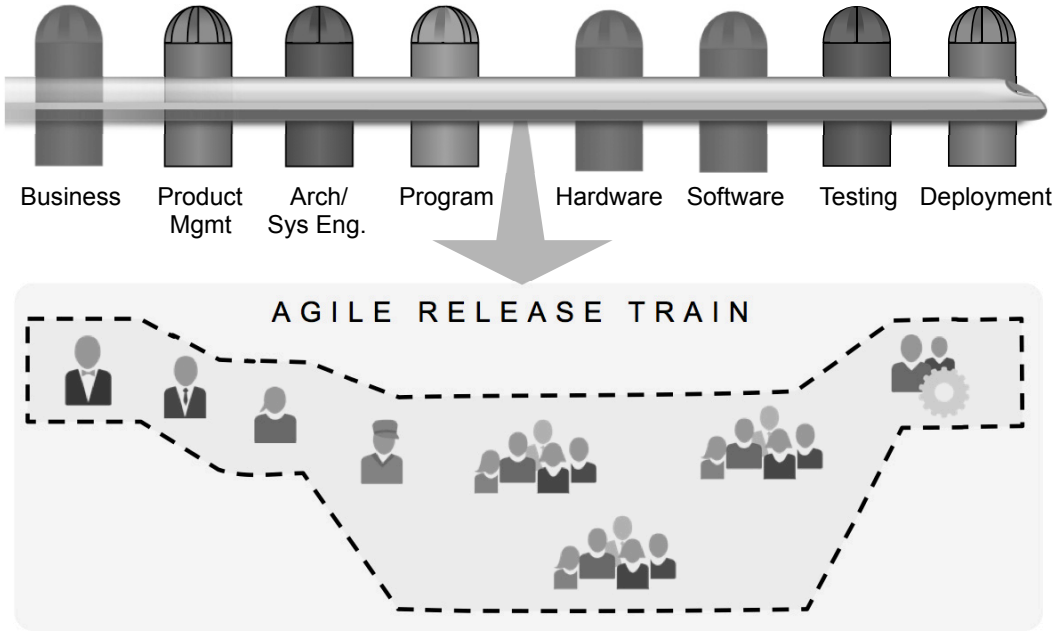
**Agile Team**

- Create and refine user stories and acceptance criteria
- Define/Build/Test/Deliver stories
- Develop and commit to Team PI Objectives and iteration plans





# Build cross-functional Agile Release Trains



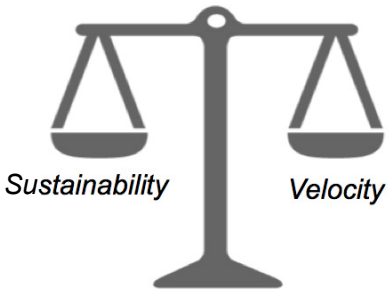
# Organizing teams around value

Organize for the larger purpose

- Maximize velocity by minimizing dependencies and handoffs, while sustaining architectural robustness and system qualities

A team can be organized around

- Features
- Components




Far less desirable

- Architectural layer
  - Platform, middleware, UI, DB, business logic, etc.
- Other
  - Programming language, spoken language, technology, location




# Finding the right trade-off

Most large programs have a mix.

Lean toward Feature teams:

- Fastest velocity
- Minimize dependencies
- Develop T-shaped skills

Use Component teams when:

- High reuse, high technical specialization, critical NFRs
- Create each component as a “potentially replaceable part of the system, with well-defined interfaces”

Generally avoid organizing around architectural layers, as they create team coupling and don’t provide a technical separation of concerns.

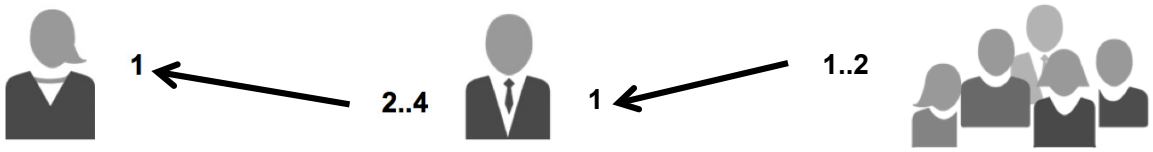
# Other ART roles

| | |
|---|---|
|  | Release Train Engineer acts as the Chief Scrum Master for the train. |
|  | Product Management owns, defines, and prioritizes the Program Backlog. |
|  | System Architect/Engineering provides architectural guidance and technical enablement to the teams on the train. |
|  | The System Team provides processes and tools to integrate and evaluate assets early and often. |
|  | Business Owners are the key stakeholders on the Agile Release Train. |




# The PM/PO team steers the train

At scale, a single person cannot handle product and market strategy while also being dedicated to an Agile Team.




Product Manager owns the Program Backlog

Product Owner owns the Team Backlog

Team implements value

# 4.3 Prepare to experience PI Planning



## PI Planning

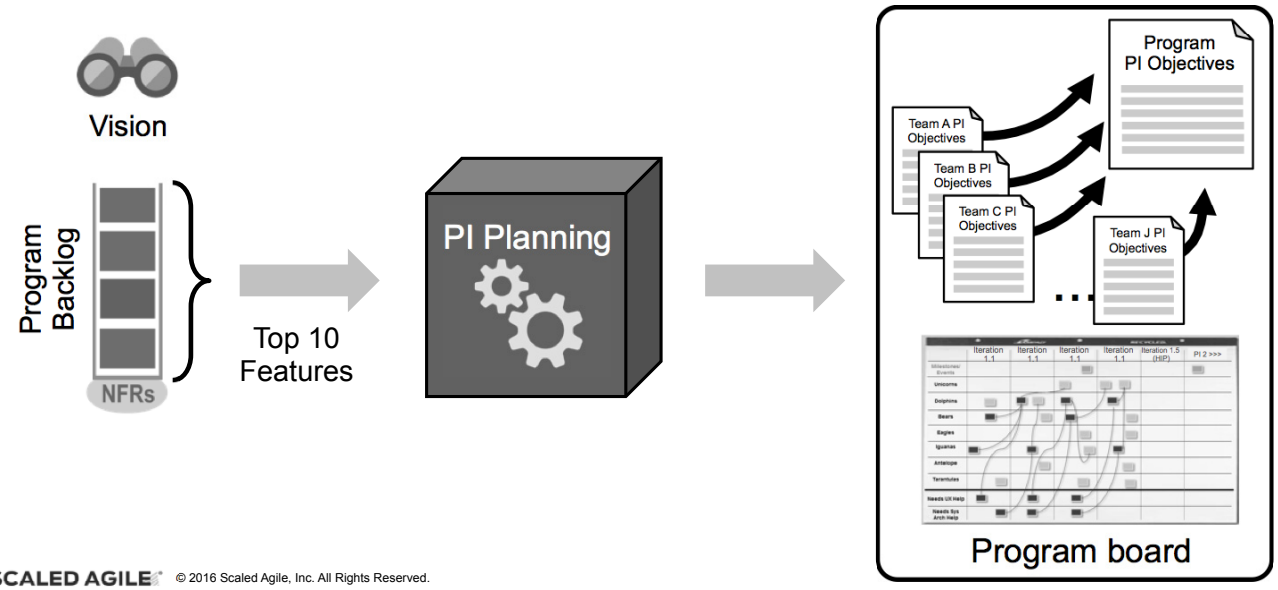
Cadence-based PI Planning meetings are the pacemaker of the Agile Enterprise.

- Two days every 8 – 12 weeks (10 weeks is typical)
- Everyone attends in person if at all possible
- Product Management owns Feature priorities
- Development teams own Story planning and high-level estimates
- Architect/Engineering and UX work as intermediaries for governance, interfaces, and dependencies

**PI Planning example**
https://youtu.be/ZZAtl7nAB1M
1:48

# The PI Planning process

Input: Vision and top 10 Features
Output: Team and Program PI Objectives and program board







# Before PI Planning: the cadence

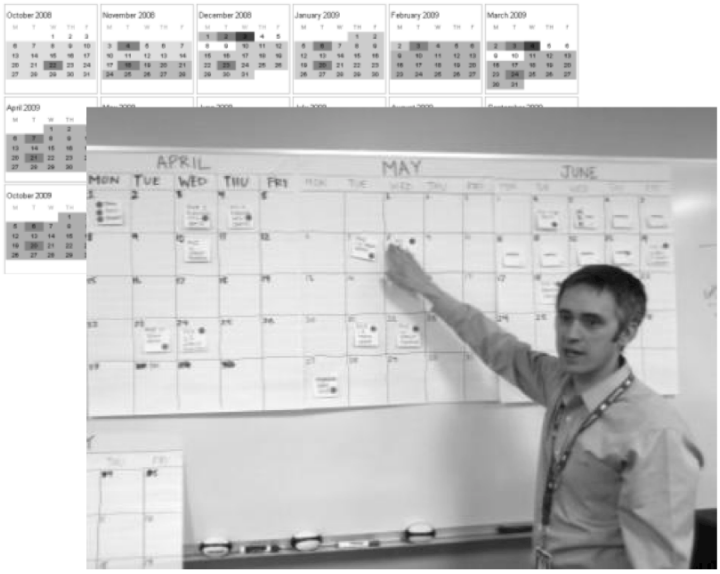
The program planning calendar can be set for a year in advance.

Program Level calendar

- PI Planning meetings
- Demos
- Inspect and Adapt workshops

Team Level calendar

- Iteration Planning meetings
- Team Demos
- Iteration Retrospectives



# Align to a mission with PI Objectives

Objectives are business summaries of what each team intends to deliver in the upcoming PI.

They often map directly to the features in the backlog ... But not always.
For example:

- Aggregation of a set of features, stated in more concise terms
- A milestone like a trade show
- An Enabler Feature needed to support the implementation
- A major refactoring

**Objectives for PI 1** **Business Value**

- Structured location and validation of locations
- Build and demonstrate a proof of concept for context images
- Implement negative triangulation by: tags, companies and people
- Speed up indexing by 50%
- Index 1.2 B more web pages
- Extract and build URL abstracts

**Stretch Objectives for PI 1**

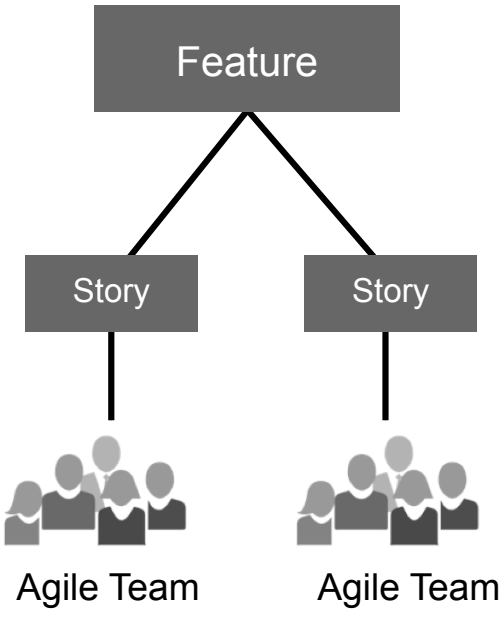
- Fuzzy search by full name
- Improve tag quality to 80% relevance




# Features are implemented by Stories

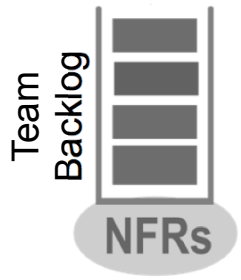
Features are decomposed into Stories by the teams on the train.

- Teams on the train collaborate to deliver features
- Features are implemented incrementally via user Stories
- Teams demonstrate working increments of features by delivering Stories on a regular cadence
- Features fit in one PI for one ART
- Stories fit in one Iteration for one Team



# The Team Backlog organizes the team's work

- It is truly all things. If a thing is in there, it might get done. If it isn't there, there is no chance that it will be done.
- It represents opportunities, not commitments—a list of what we want to do
- Items may be estimated (preferable), but estimates do not imply committed delivery
- It has a single owner: the team's Product Owner, who is also part of a larger PO/PM team
- It is largely driven by Program priorities




Product Owner

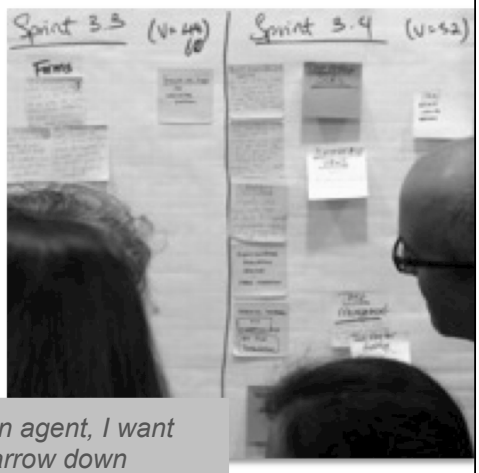




# The Backlog contains User Stories

User Stories replace traditional requirements, and describe intended system behavior.

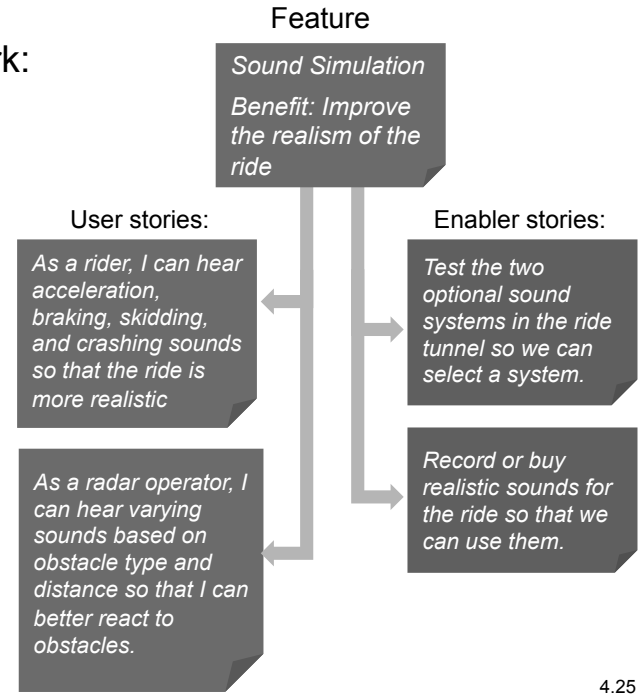
- Small increments of value that can be developed in days
- Relatively easy to estimate
- Elaborated on a just-in-time basis
- No large, unwieldy documents
- In many circumstances, may be safely discarded after implementation




*As an agent, I want to narrow down people-search results by location so I can find the right person more quickly.*

# Enabler Stories support value

- They can represent different types of work:
  - Exploration
  - Architecture
  - Infrastructure
- Enabler stories are demonstrated like any other story







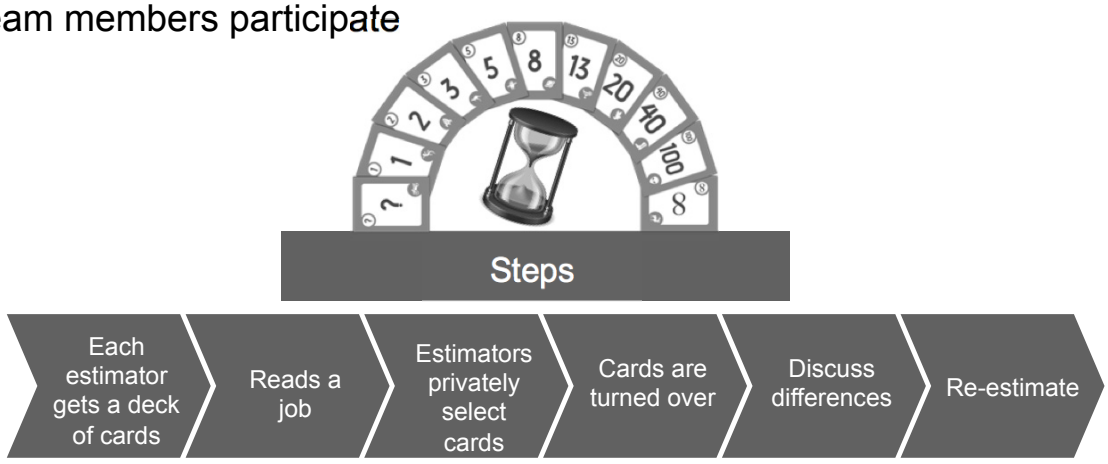
# Estimate Stories with relative Story points

- A Story point is a singular number that represents:
  - Volume: how much is there?
  - Complexity: how hard is it?
  - Knowledge: what do we know?
  - Uncertainty: what's not known?
- Story points are relative; they are not connected to any specific unit of measure
- Compare with other stories (an 8-point story should take 4X longer than a 2-point story)

How *big* is it?

# Apply Estimating Poker for fast, relative estimating

- Estimating Poker combines expert opinion, analogy, and disaggregation for quick but reliable estimates
- All team members participate




Mike Cohn, *Agile Estimating and Planning,* 2005




# Estimation is a whole team exercise

- Increases accuracy by including all perspectives
- Builds understanding
- Creates shared commitment

*Estimation performed by a manager, architect, or select group negates these benefits*

## Exercise: Relative size estimating

Use Estimating Poker to relatively estimate the mass of a set of animals

- As a team at your table, identify the smallest animal and mark it as 1
- Estimate the remaining animals using values 1, 2, 3, 5, 8, 13, 20, 40, 100


Giraffe


Horse


Crocodile


Gorilla


Hyena


Elephant


Chicken

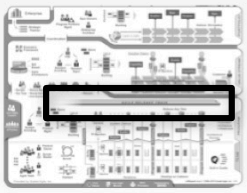





## Lesson summary

In this lesson, you:

- Examined synchronizing development with the Agile Release Train
- Discussed organizing Agile Teams and implementing key Agile Release Train roles
- Prepared for a PI Planning Exercise



*Suggested Scaled Agile Framework reading:*
*"Agile Release Train" article*