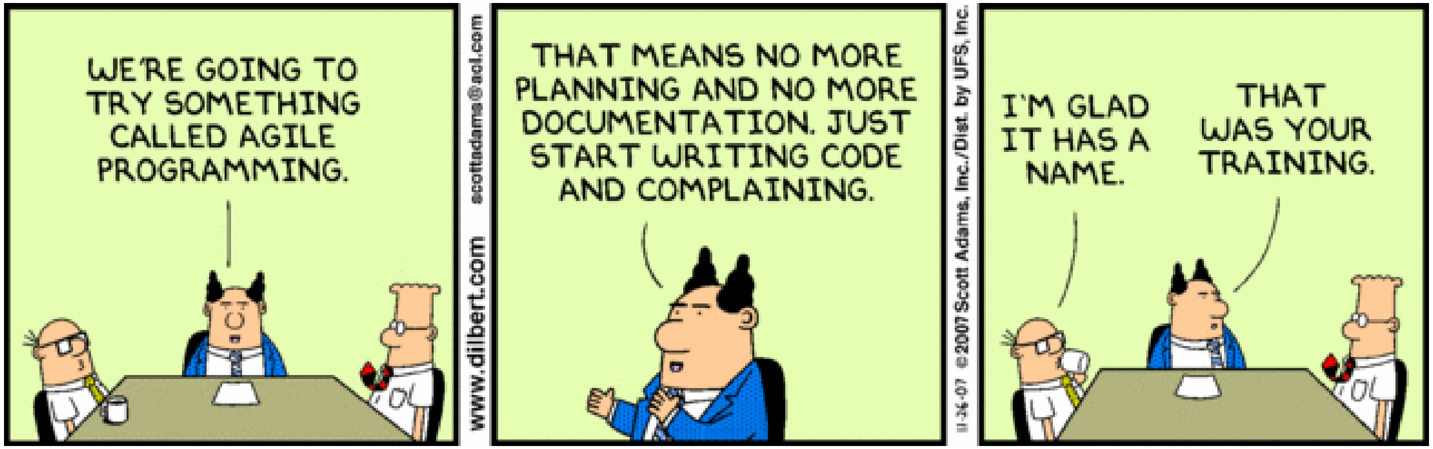
# 敏捷開發
# Agile Development

逢甲大學 人工智慧研究中心
資訊工程學系

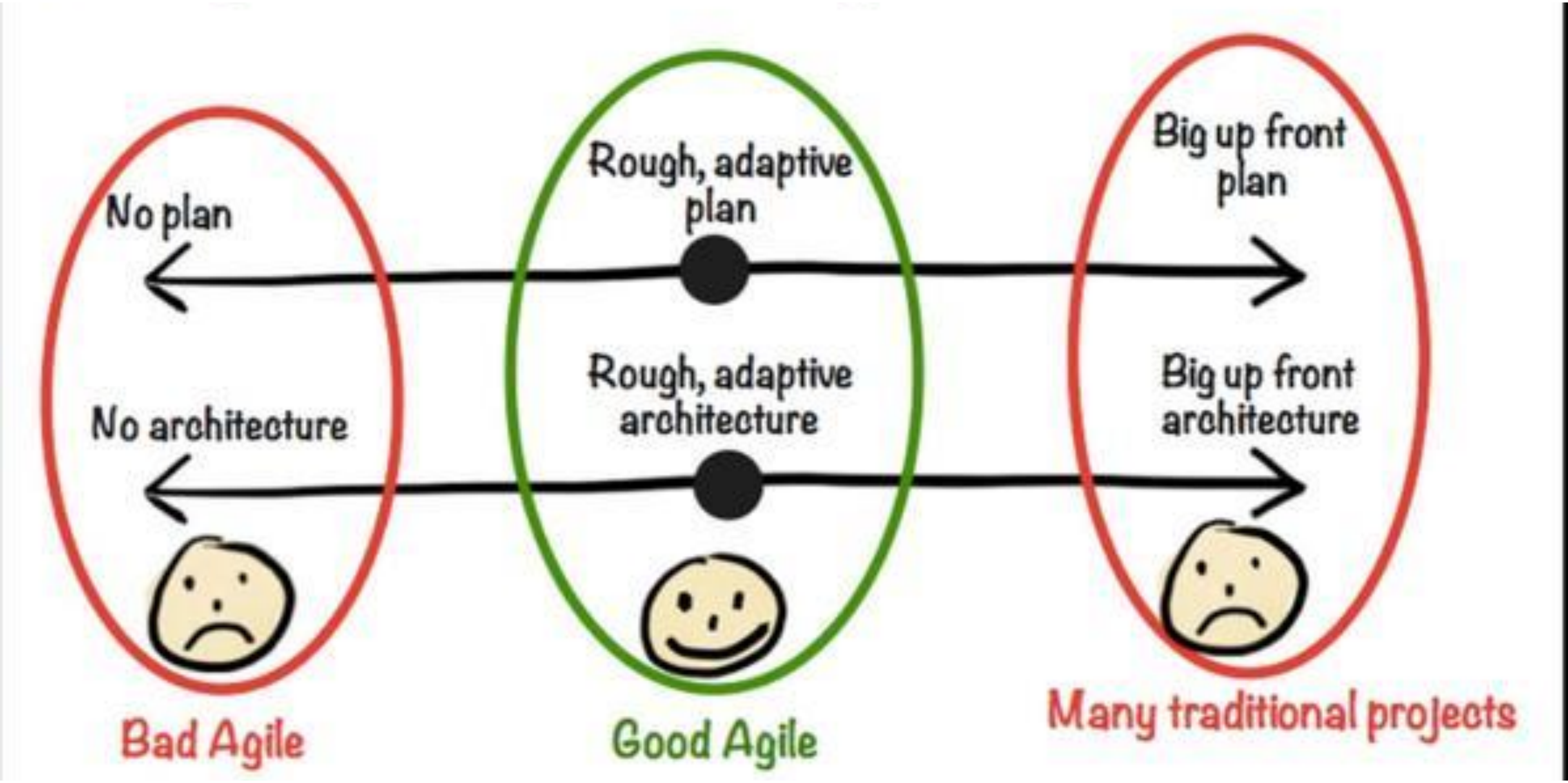
許懷中

# Agile Development

- Agile Development are frameworks and practices of Software Engineering


WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.
scottadams@aol.com
www.dilbert.com
THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.
11-26-07 ©2007 Scott Adams, Inc./Dist. by UFS, Inc.
I'M GLAD IT HAS A NAME.
THAT WAS YOUR TRAINING.

- https://youtu.be/Z9QbYZh1YXY


No plan
Rough, adaptive plan
Big up front plan
No architecture
Rough, adaptive architecture
Big up front architecture
Bad Agile
Good Agile
Many traditional projects

Agile Development

# AGILE MANIFESTO & PRINCIPLES

# 敏捷宣言

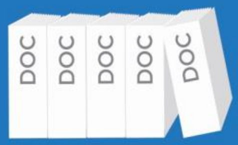
我們在實踐中探尋更好的軟體開發方法，
身體力行的同時也幫助他人，由此我們重視：

個人與互動 勝過 流程與工具

可用的軟體 勝過 詳盡的文件

與客戶合作 勝過 合約協商

回應變化 勝過 遵循計畫



雖然下方事項也有價值，但是我們更重視上方事項的價值

# 敏捷原則 (1)

1. 我們最重要的目標，是透過**持續不斷**地**及早交付有價值的軟體**令客戶滿意
2. **欣然面對需求變化**，既使已經在開發後期，敏捷流程為了客戶的競爭優勢駕馭變化
3. 從一兩個月到幾週，**經常地交付可工作的軟體**，週期越短越好
4. **業務人員**與**開發者**在開發過程的**每一天**都必須互相合作

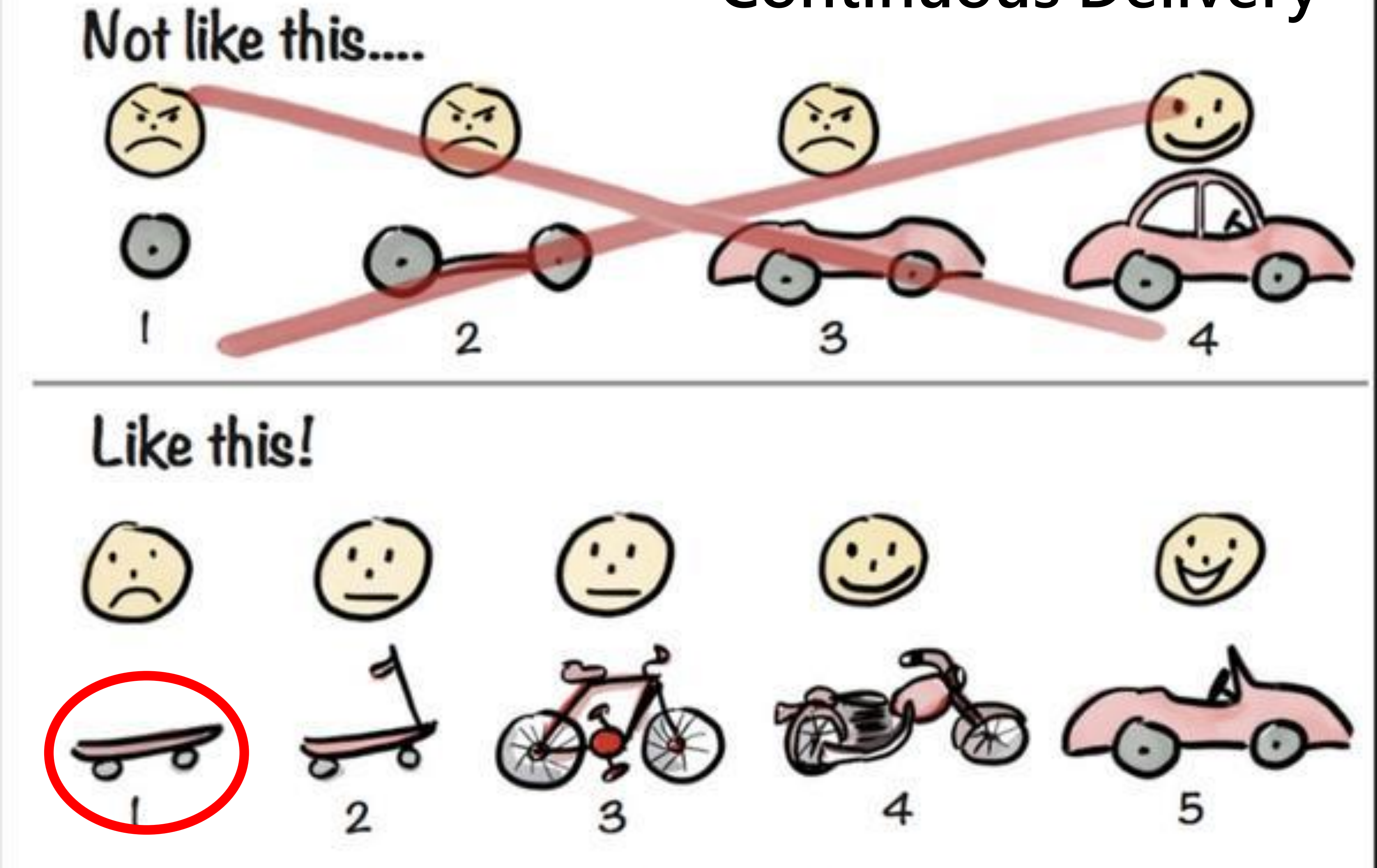
# 敏捷原則 (2)

5. 以**具有動機**的個人為中心開發專案，提供所需的**環境**與**支援**，輔以**信任**，從而達成目標
6. 無論團隊內外，最有效的溝通方式是**面對面的交談**
7. **可工作的軟體**是衡量專案進度的最主要標準
8. 敏捷流程提倡可持續的開發，**利害關係人**、**開發者**與**用戶**必須要**共同**維持穩定而持續的步調

# 敏捷原則 (3)

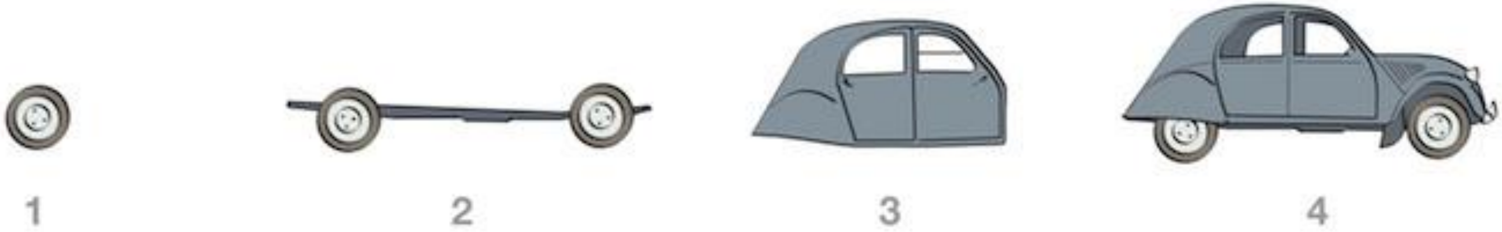
9. **持續**專注於**技術卓越**與**良好設計**令我們愈發敏捷
10. **簡潔**，所謂極力減少不必要工作的藝術，是箇中關鍵
11. 最好的架構、需求與設計出自**自組織團隊**
12. 團隊**定期反思**如何能提高成效，並依此調整自身的行為

# Continuous Delivery

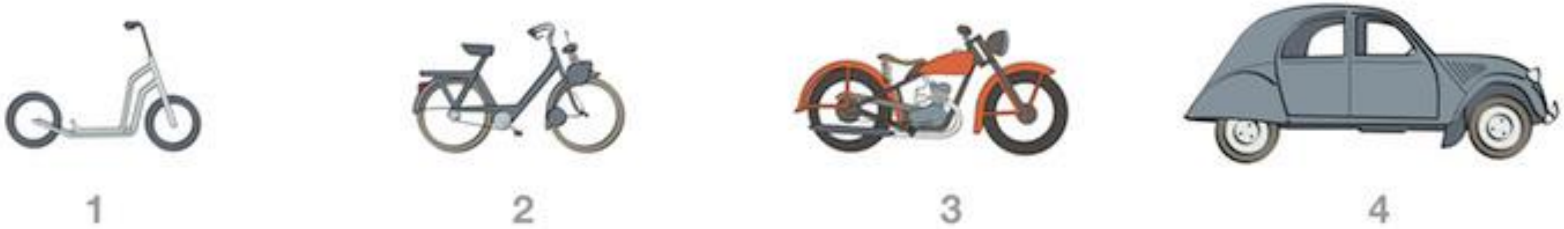


Minimal Viable Product (MVP)

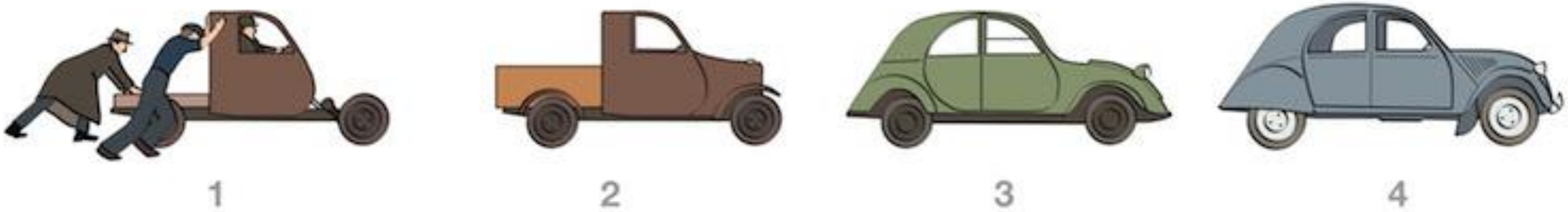
## HOW NOT TO BUILD A MINIMUM VIABLE PRODUCT



## ALSO HOW NOT TO BUILD A MINIMUM VIABLE PRODUCT



## HOW TO BUILD A MINIMUM VIABLE PRODUCT



FRED VOORHORST

WWW.EXPRESSIVEPRODUCTDESIGN.COM

# MVP Practice

- Try to describe one of your mornings
  - Starts from opening your eyes ……

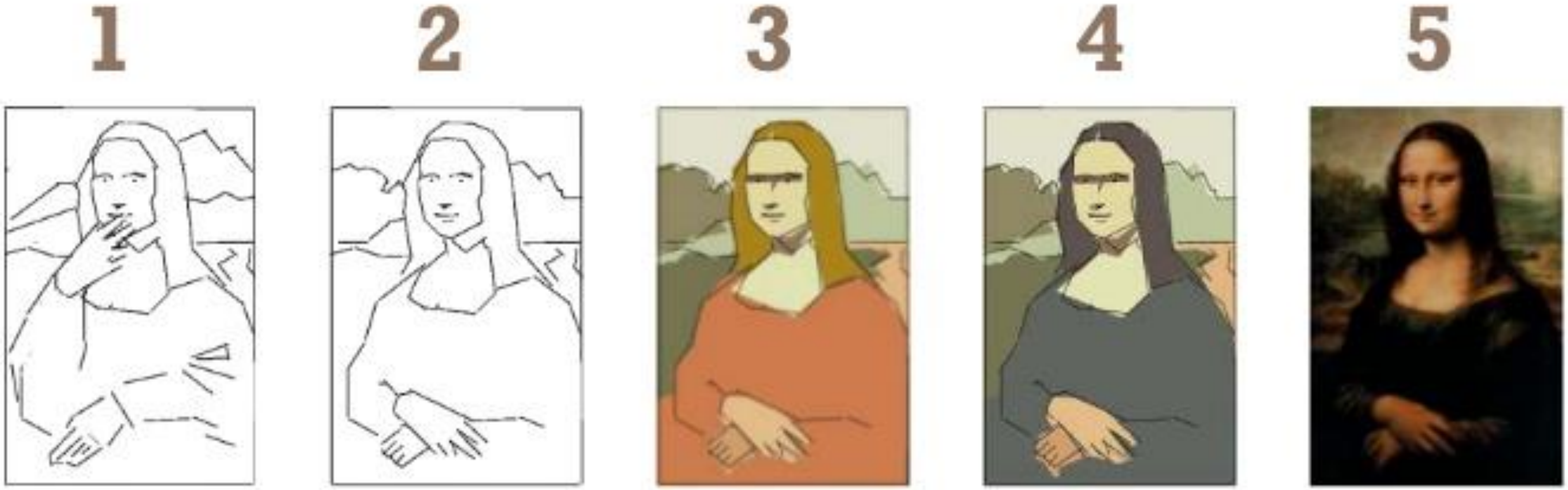
# MVP Practice (cont.)

- Think of the following scenarios
  - What would you do if you are late and you have only 10 minutes to get out of your door?

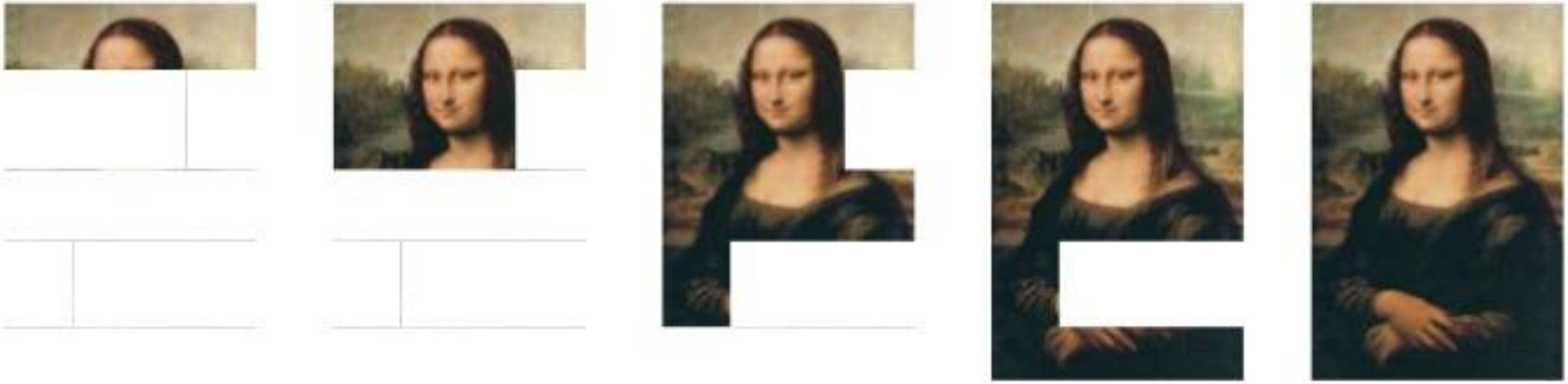
# MVP Practice (cont.)

- Think of the following scenarios
  - What is your ideal morning?

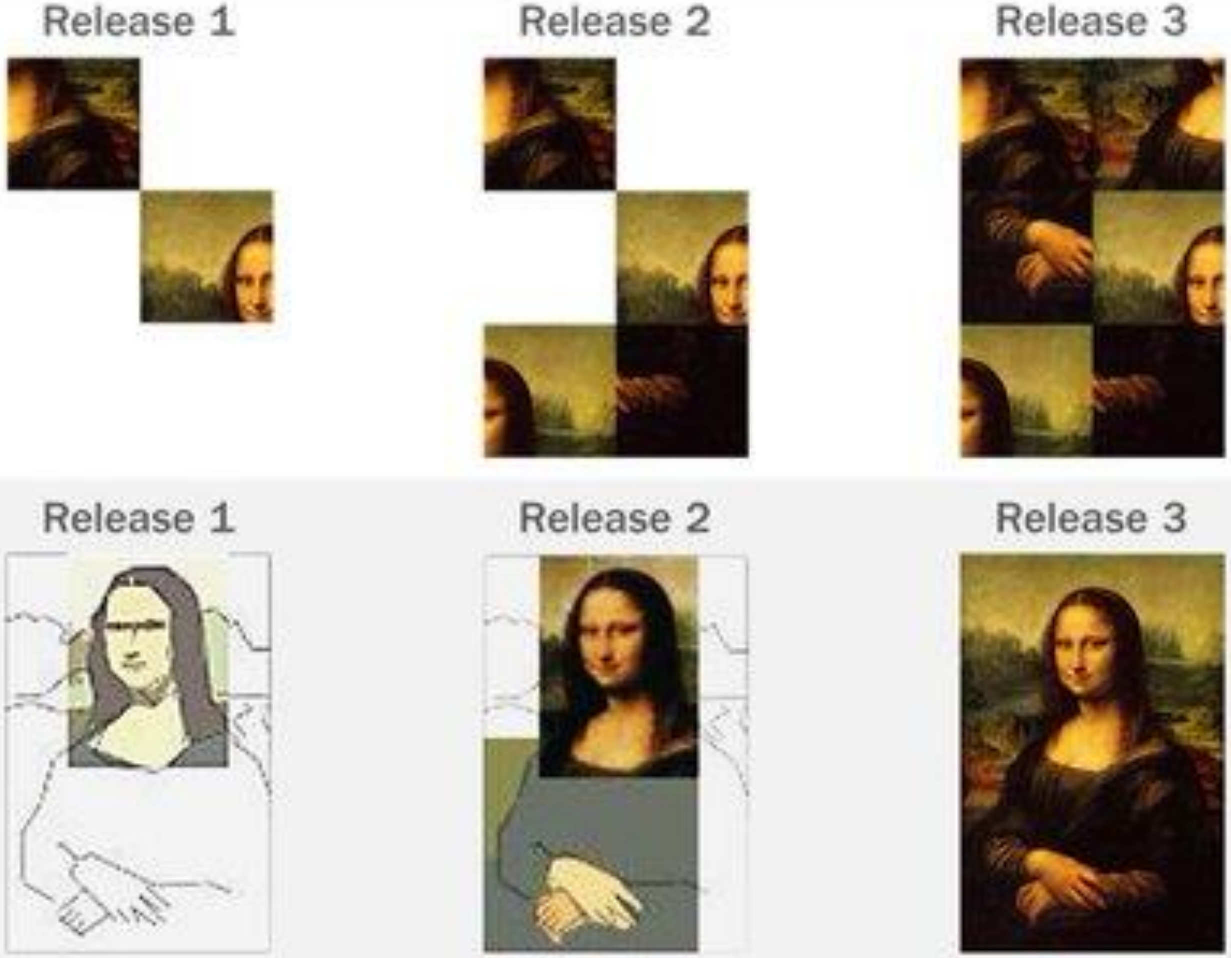
# Iterative



# Incremental



Outcome: Explain the difference between iterative and



Agile as usually practiced

Release 1

Release 2

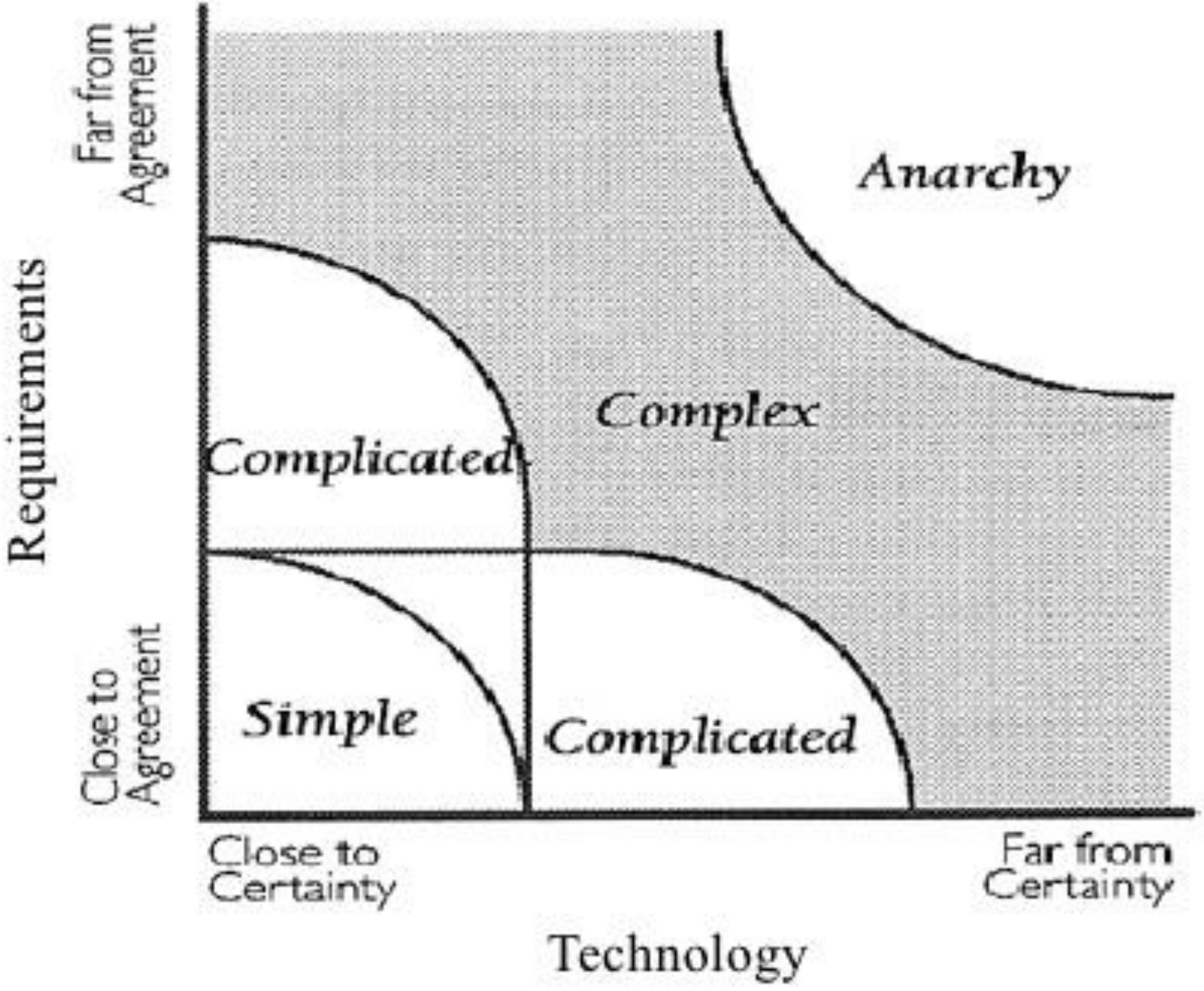
Release 3

Agile UX

Release 1

Release 2

Release 3



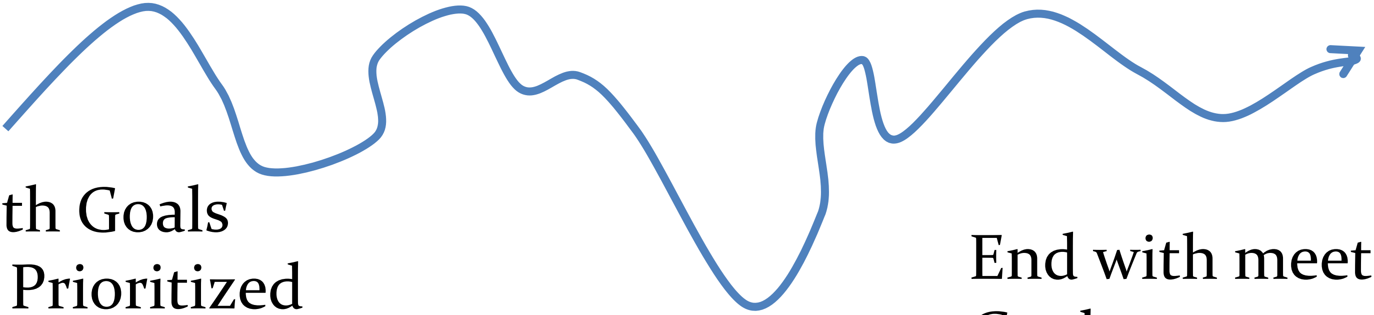
# Categorization of Complexity in Project Development

# Predictive

Start with Plan
& all Requirements

End with completing
all Requirements

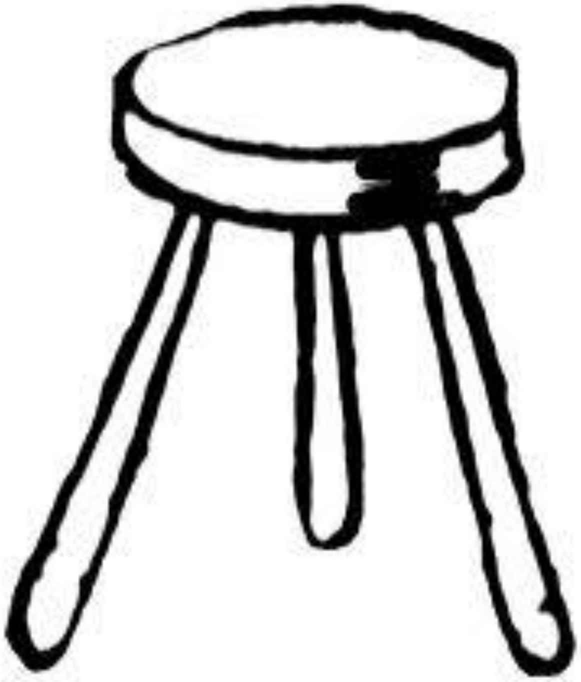
# Agile Empirical



Start with Goals
& some Prioritized
Requirements

End with meeting
Goals

# Agile Empirical



Inspect

Adapt

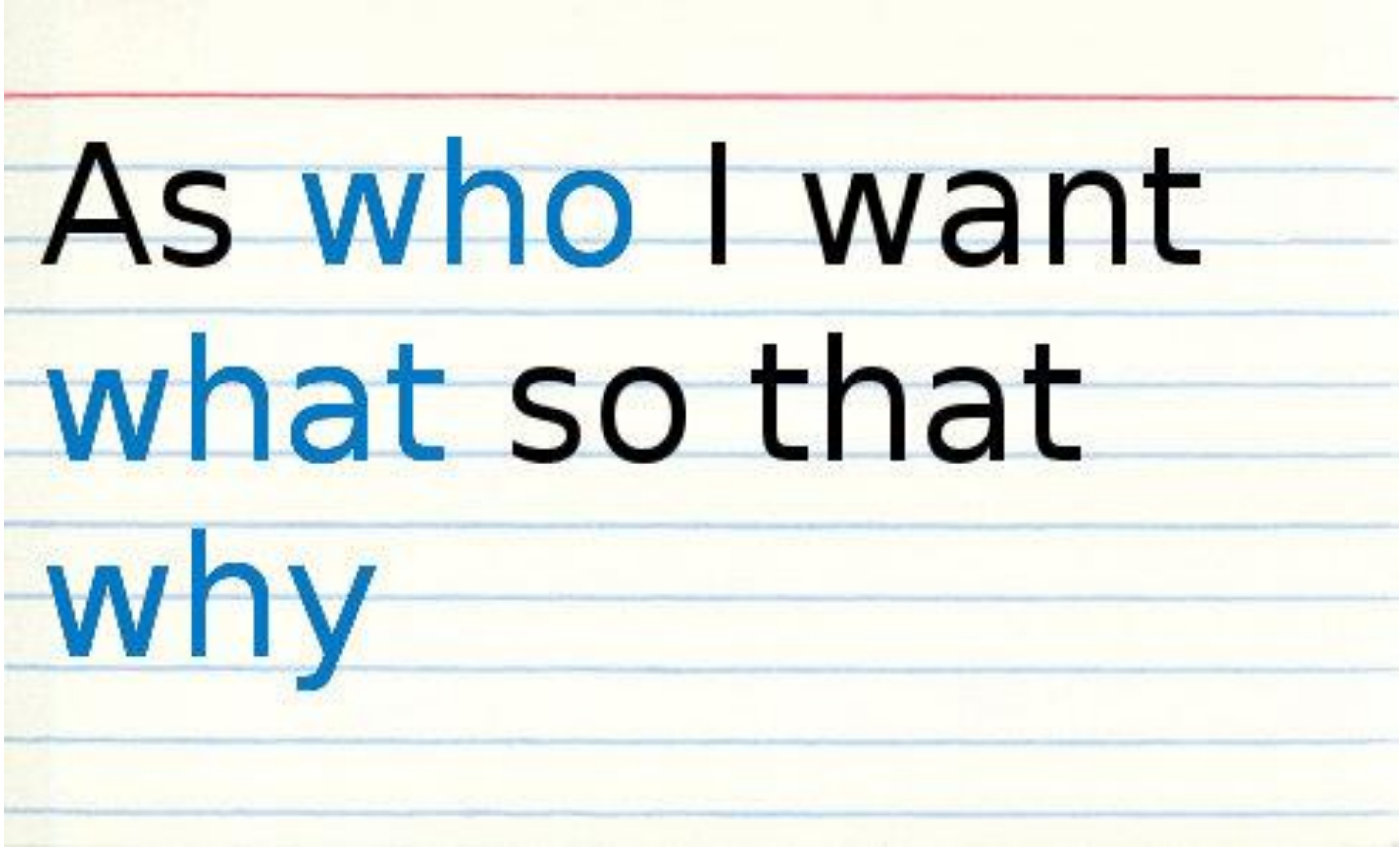
Transparency

# Discussion

Agile Development - Scrum

# USER STORY AND
# USER STORY MAPPING

# User Story



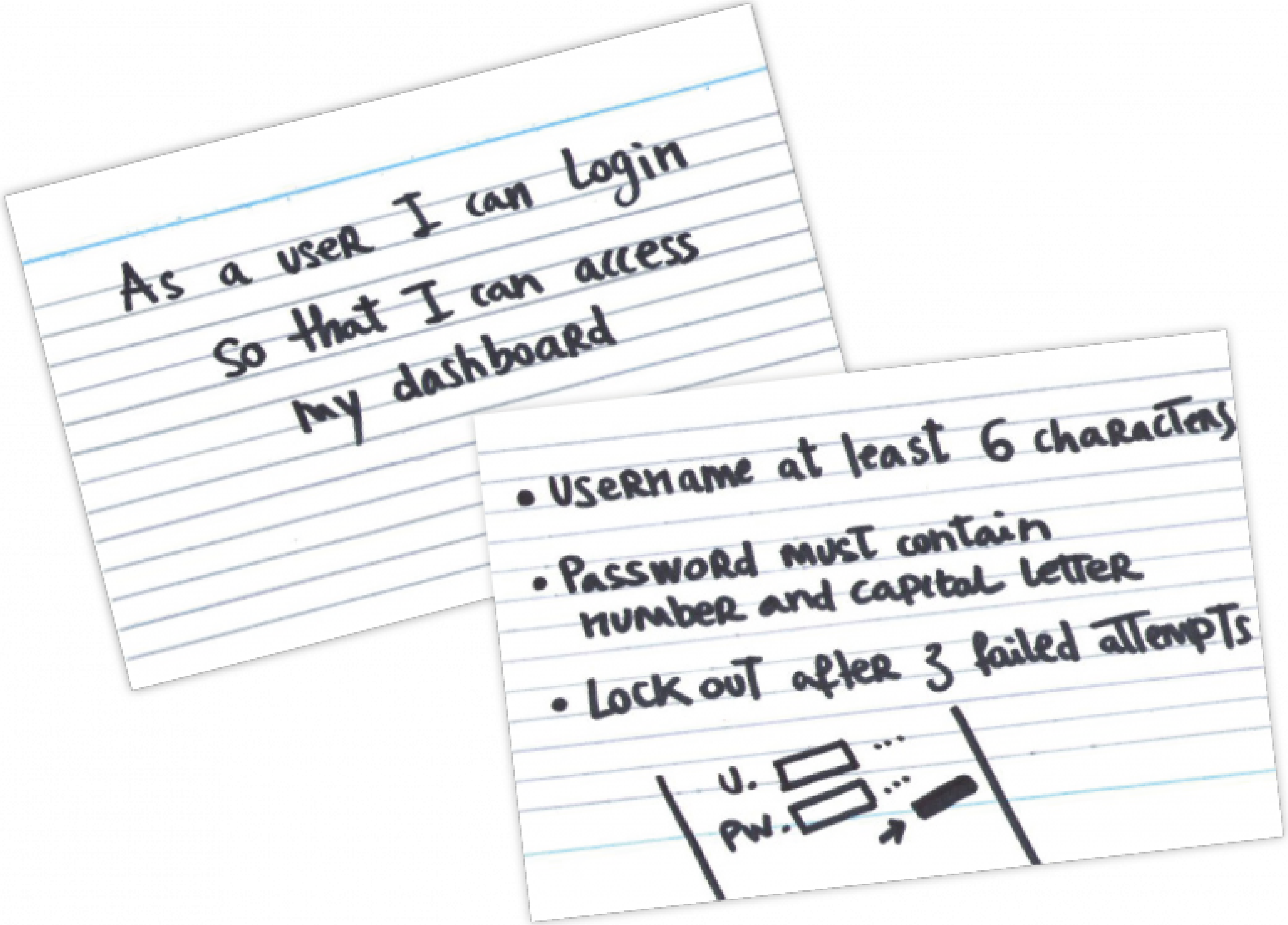
As who I want what so that why

Adapted from: http://www.bloomy.com/support/blog/agile-action-user-stories-part-3-5

# User Story - Example

- Subject Registration System
  - As a user, I want to login the system, so that I can register my subjects for the semester.
  - As a student, I want to see what course is available for me so that I can select subjects into my curriculum.
  - As a student, I want the system always available in the registration duration so that I can register my subjects anytime I want.
  - As a teacher, I want to know how many students choose my course so that I can realize how to arrange the course schedule.

# User Story (cont.)



As a user I can login
So that I can access
my dashboard

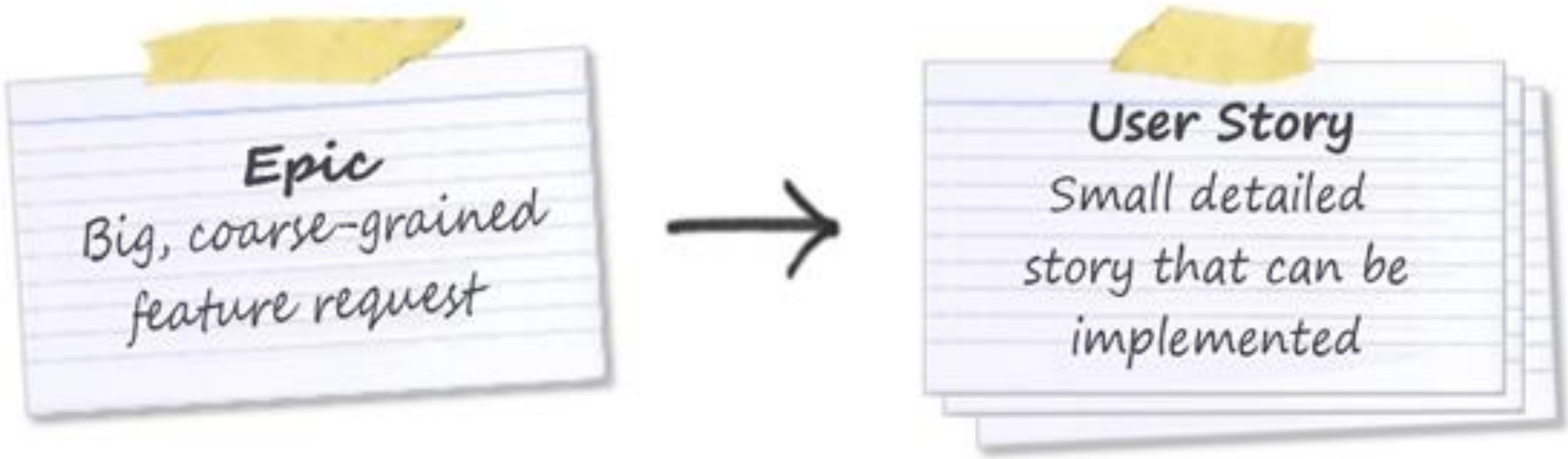
- Username at least 6 characters
- Password must contain number and capital letter
- Lock out after 3 failed attempts

Adapted from: http://www.bloomy.com/support/blog/agile-action-user-stories-part-3-5

# Acceptance Criteria - Example

- As a student, I want to see what course is available for me so that I can select subjects into my curriculum.
  - The required subjects are pre-selected into my curriculum.
  - The subjects selected into my curriculum can not be conflict in time.
  - The final curriculum must satisfy the constraints in credits (9 in minimum, 30 in maximum)
  - ……

# User Story (cont.)


Epic
Big, coarse-grained feature request
User Story
Small detailed story that can be implemented

Adapted from: http://www.bloomy.com/support/blog/agile-action-user-stories-part-3-5

# The Core Concept for User Story

- **Card**
  - Writing stories on cards
  - Prioritize the cards
- **Conversation**
  - Coordination
  - Solving problems
  - Best Solution
- **Confirmation**
  - Acceptance criteria
  - Story tests
- Describe in business language

Agile Development

# SCRUM

Works
INFRASTRUCTURE

# A Brief History of Scrum

The New New Product Development Game

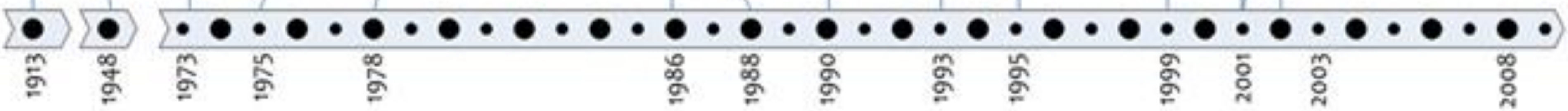
„Moving the Scrum Downfield“

Manifesto for Agile Software Development

Wicked Problems, Righteous Solutions

A Catalogue of Modern Software Engineering Paradigms

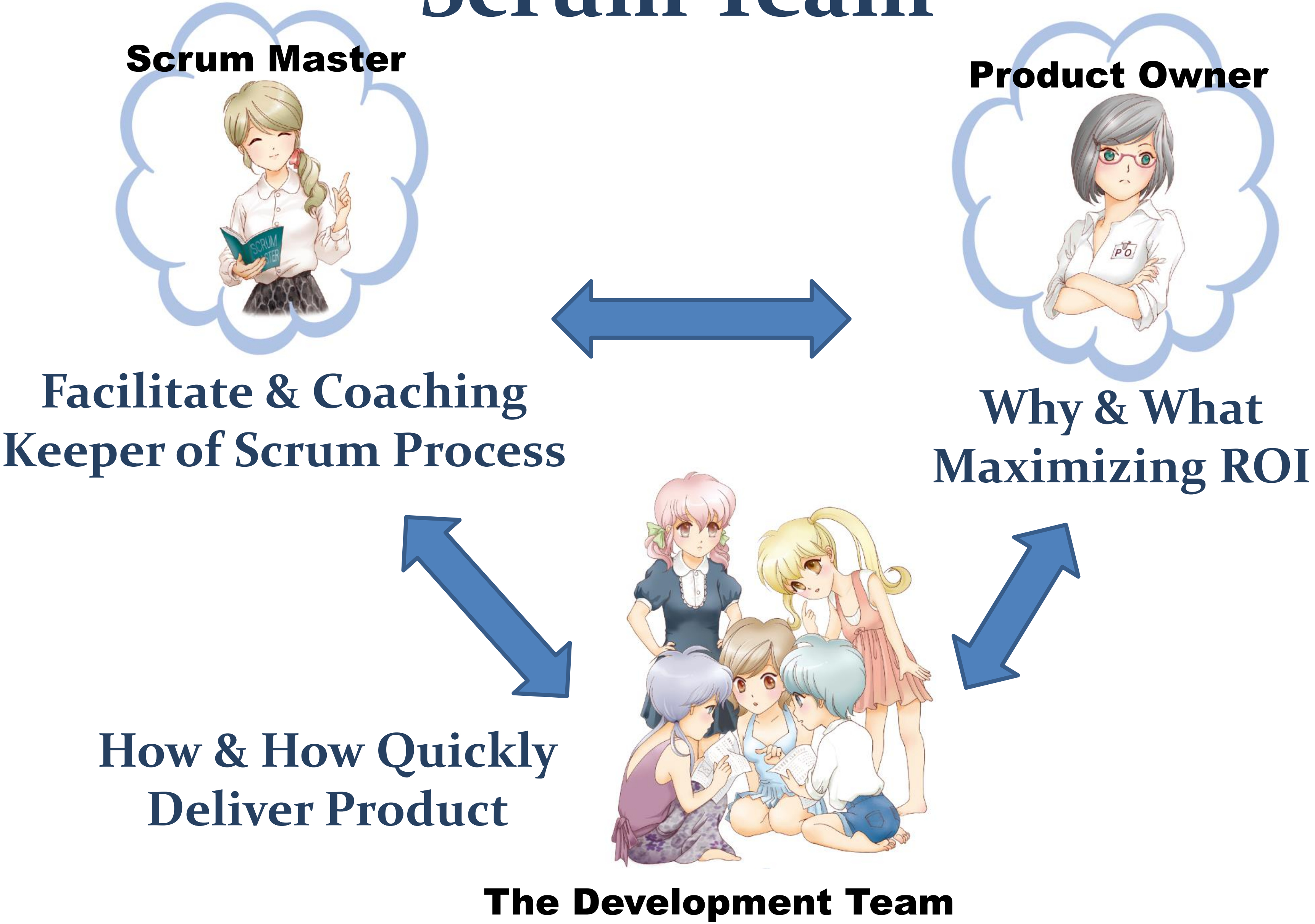
First Scrum at Easel Corp.



# What is Scrum?

- A framework without engineering practices
  - Activities with explicit time-box
  - Scrum roles
  - Iterative & Incremental
- Lightweight
- Inspection & Adaptation
- Transparency
  - Product backlog, Sprint backlog, Task board

# Scrum Team



Scrum Master

Facilitate & Coaching
Keeper of Scrum Process

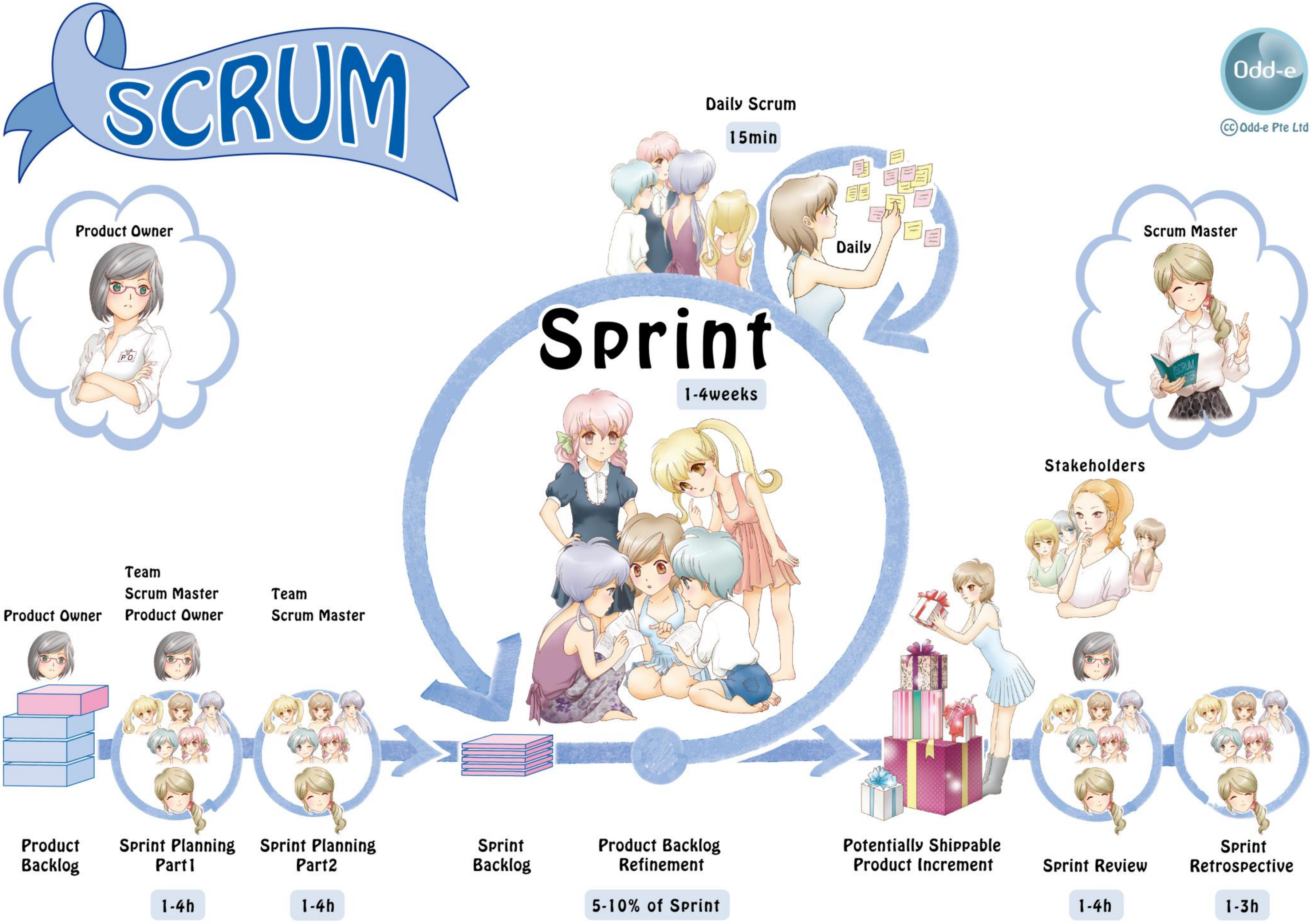
Product Owner

Why & What
Maximizing ROI

How & How Quickly
Deliver Product

The Development Team

# SCRUM



Product Owner

Scrum Master

Daily Scrum

15min

Daily

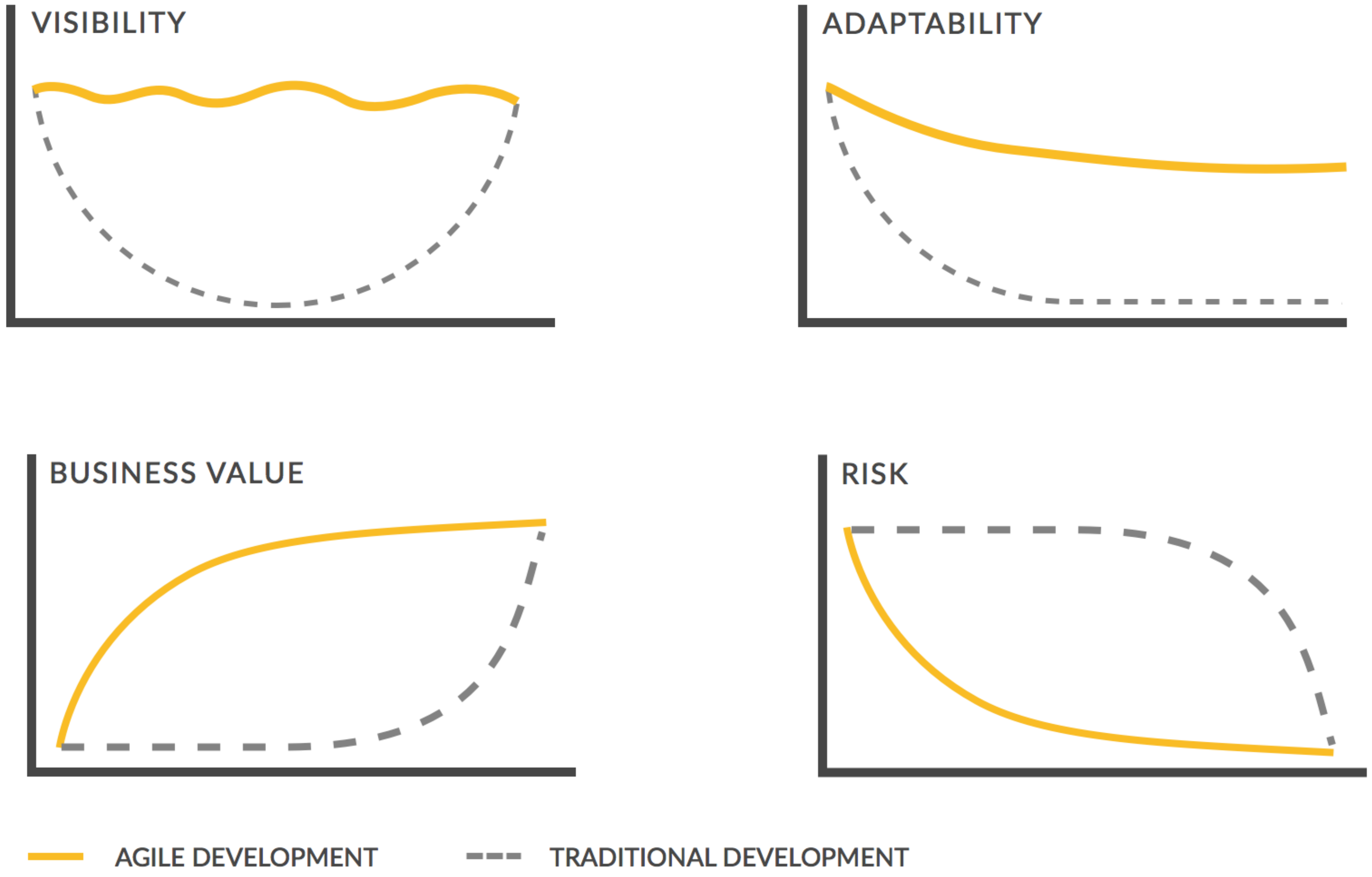
Sprint

1-4weeks

Stakeholders

| Product Owner | Team<br>Scrum Master<br>Product Owner | Team<br>Scrum Master | | | | | |
|---|---|---|---|---|---|---|---|
| Product Backlog | Sprint Planning Part1 | Sprint Planning Part2 | Sprint Backlog | Product Backlog Refinement | Potentially Shippable Product Increment | Sprint Review | Sprint Retrospective |
| | 1-4h | 1-4h | | 5-10% of Sprint | | 1-4h | 1-3h |

# AGILE DEVELOPMENT
VALUE PROPOSITION



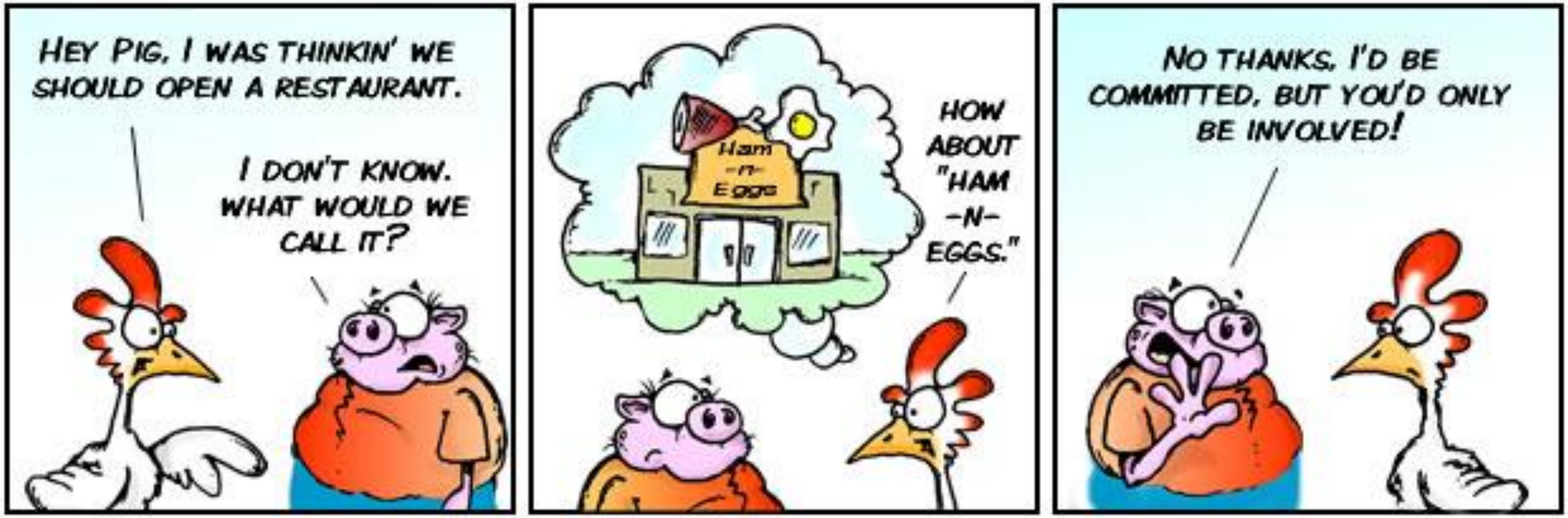
# Product Owner

- Maximizing Return of Investment (ROI)
- Identifying & Prioritizing Features
- Responsible for Profit and Loss of the Product
- Regularly and Actively Interact with the Team and the Stakeholders
- The One and the Only

# The (Development) Team

- Build the Product
- Self-organizing
  - Cross-functional & Autonomous
- **7 ± 2**
- **Pull** instead of **being Pushed**
- Commitment
- Continuous Delivering Working Software

# Chickens and Pigs


HEY PIG, I WAS THINKIN' WE SHOULD OPEN A RESTAURANT.
I DON'T KNOW. WHAT WOULD WE CALL IT?
Ham -n- Eggs
HOW ABOUT "HAM -N- EGGS."
NO THANKS, I'D BE COMMITTED, BUT YOU'D ONLY BE INVOLVED!

By Clark & Vizdos

© 2006 implementingscrum.com

# Scrum Master

- Help the Product Group Learn and Apply Scrum to Achieve Business Value
- Shield the Team from outside interferences
- Coach and Teach both PO and the Team
- Reveal Problems and Remove Barriers
- Ensure the Process is Followed

# What Scrum Master is NOT

- Not a Manager
  - Not a project (product) manager
  - Driven by management or driven by customer?
- Not a Decision Maker
  - Let the team decide
  - Let them learn themselves
  - Are you sure **you** are right?

# Quiz

| Product Owner 的責任 | |
|---|---|
| 澄清需求 | |
| 決定這個發行 (release) 要包含什麼需求 | |
| 決定要做哪些工作 (task) | |
| 拜訪客戶 | |
| 參加每日站立會議 | |
| 隨時能被開發團隊找到 | |
| 決定一個 sprint 要完成多少工作 | |
| 了解客戶如何使用產品 | |
| 為產品盈利負責 | |
| 了解競爭對手 | |

# Quiz (cont.)

| 開發團隊 (The Development Team) 的責任 | |
|---|---|
| 持續改善流程 | |
| 決定工作 (task) 的優先順序 | |
| 管理與其他團隊的相依性 | |
| 維護 Product backlog | |
| 維護 Sprint backlog | |
| 決定產品架構與設計 | |
| 不允許和客戶交流 | |
| 團隊自己做測試 | |
| 自己更新 Task board 與 燃盡圖 | |

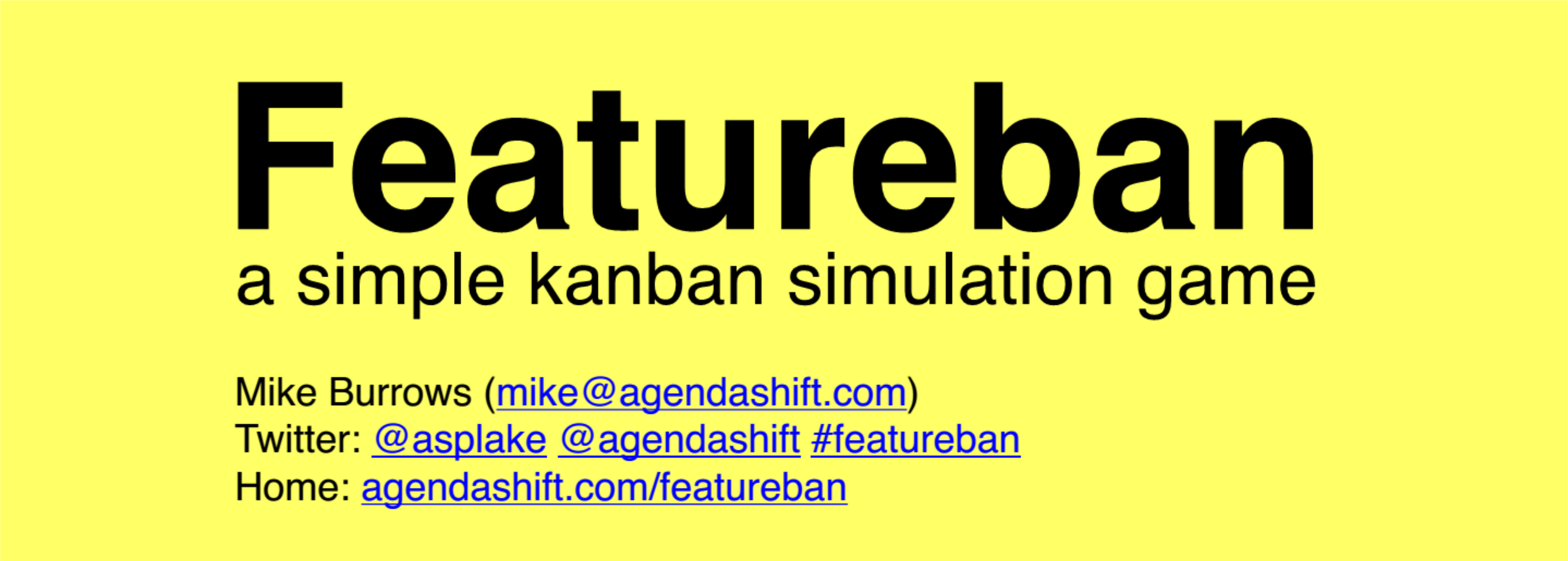
# Quiz (cont.)

| Scrum Master 的責任 | |
|---|---|
| 向產品負責人承諾工作會做完 | |
| 負責 Sprint 進度 | |
| 輔導建立高效能的團隊 | |
| 決定開發團隊成員 | |
| 導入與維護 Scrum 流程 | |
| 幫開發團隊解決問題 | |
| 導入工程實踐 | |
| 管理專案 | |
| 引導各項 Scrum 會議 | |
| 指導 (Coaching) 產品負責人與開發團隊 | |

Agile Development

# KANBAN

# Featureban Game



Featureban

a simple kanban simulation game

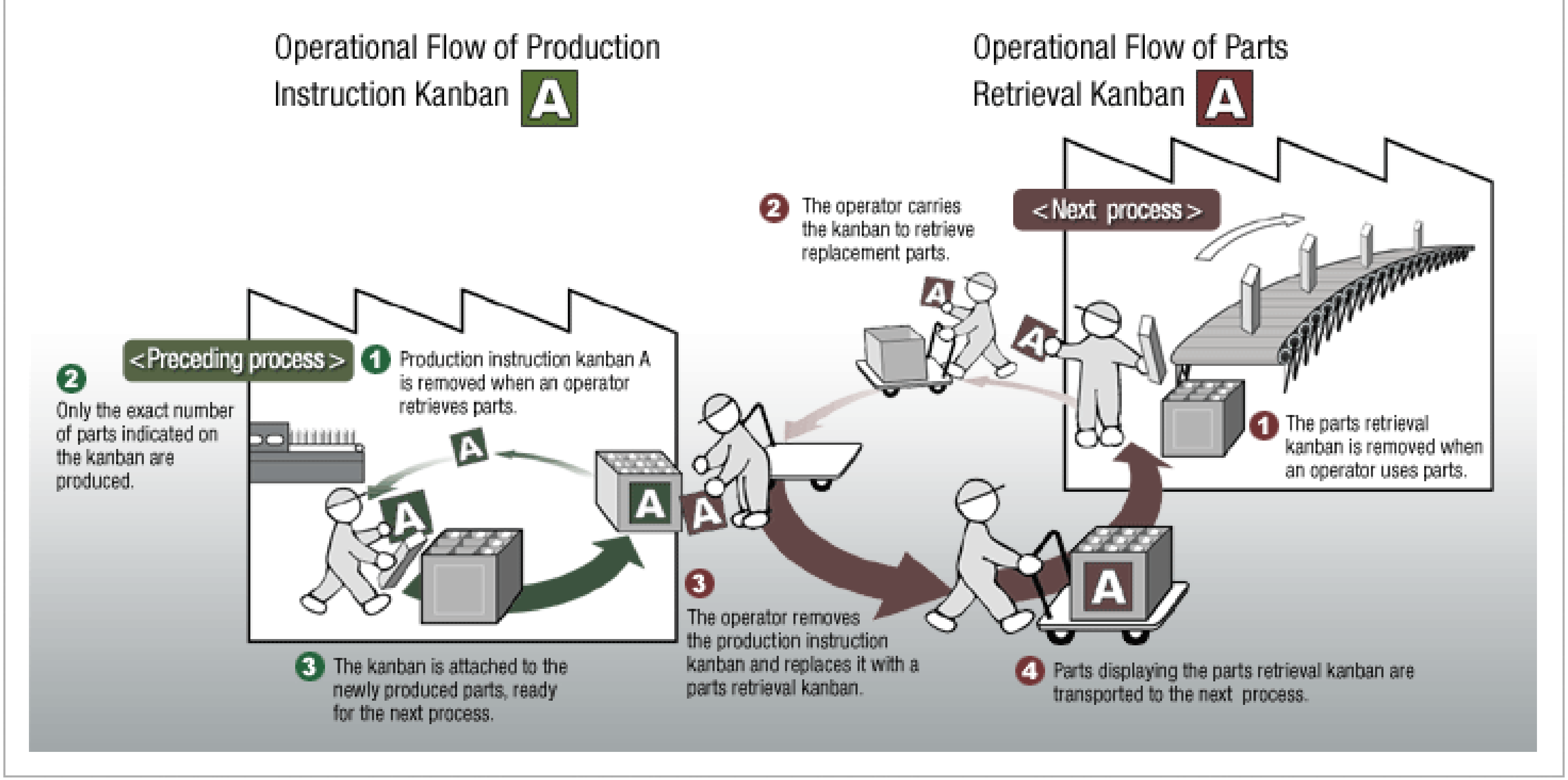
Mike Burrows (mike@agendashift.com)
Twitter: @asplake @agendashift #featureban
Home: agendashift.com/featureban

# What is Kanban

- https://youtu.be/5izyN66PTxs

# Toyota Production System

Conceptual diagram of the Kanban System



Operational Flow of Production Instruction Kanban A

Operational Flow of Parts Retrieval Kanban A

<Preceding process>

1 Production instruction kanban A is removed when an operator retrieves parts.

2 Only the exact number of parts indicated on the kanban are produced.

3 The kanban is attached to the newly produced parts, ready for the next process.

<Next process>

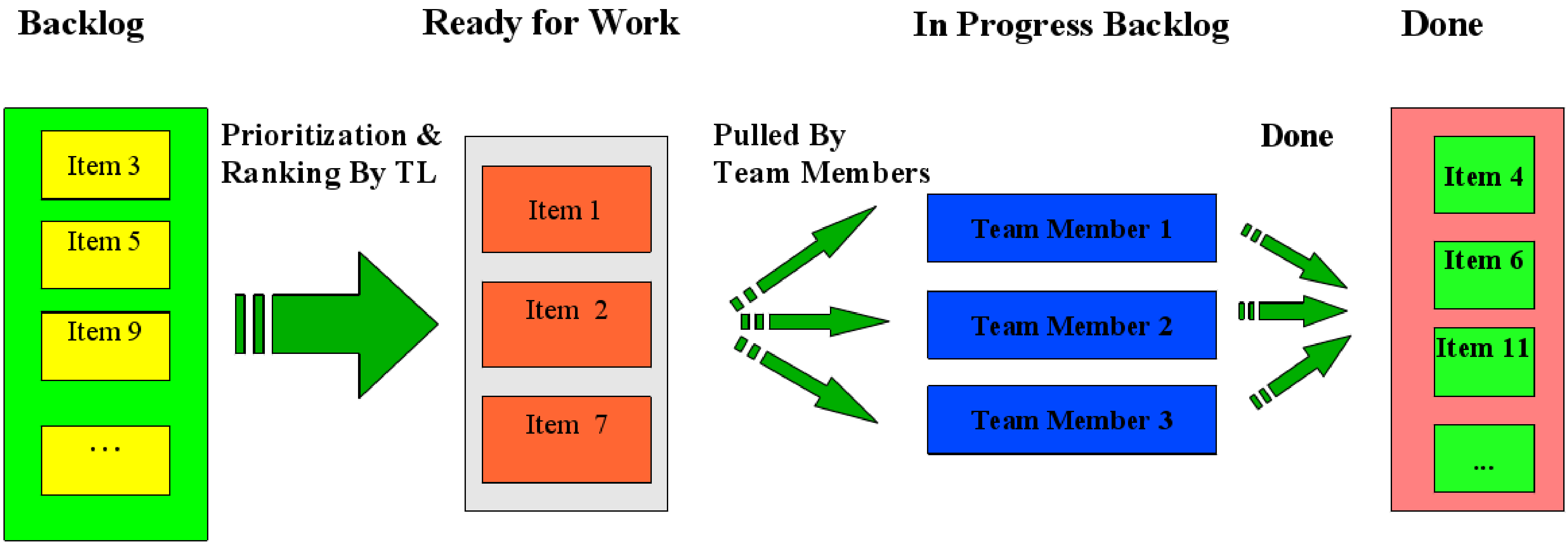
1 The parts retrieval kanban is removed when an operator uses parts.

2 The operator carries the kanban to retrieve replacement parts.

3 The operator removes the production instruction kanban and replaces it with a parts retrieval kanban.

4 Parts displaying the parts retrieval kanban are transported to the next process.

# Pull Model



Backlog

Ready for Work

In Progress Backlog

Done

Item 3

Item 5

Item 9

. . .

Prioritization &
Ranking By TL

Item 1

Item 2

Item 7

Pulled By
Team Members

Team Member 1

Team Member 2

Team Member 3

Done

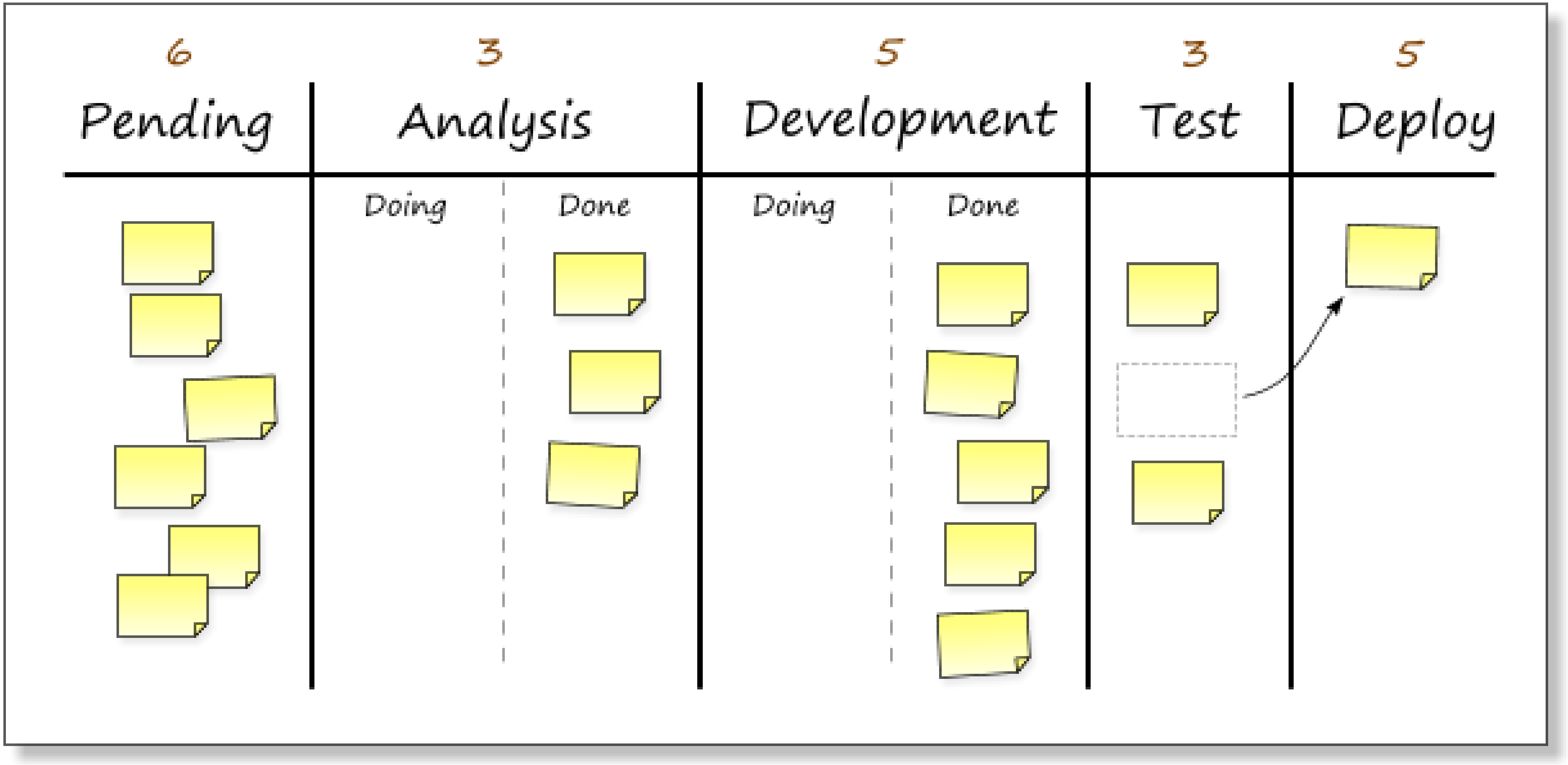
Item 4

Item 6

Item 11

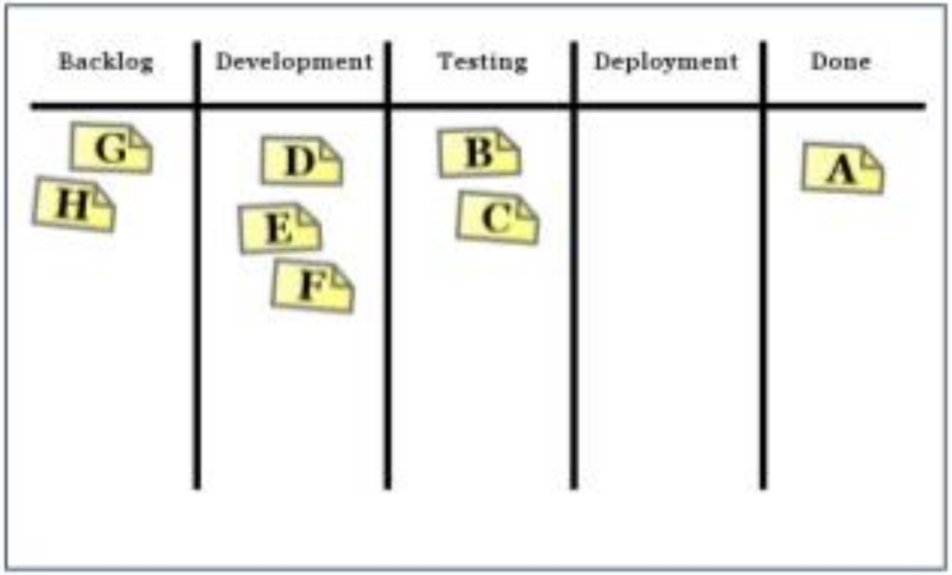
...

# Example Kanban



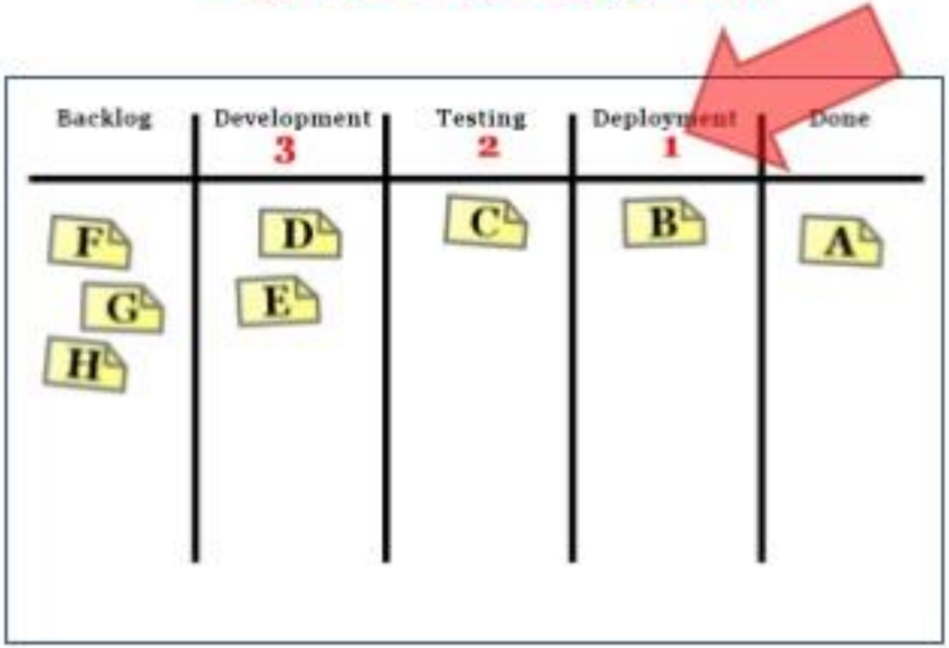
# Scrum versus Kanban Board

## SCRUM



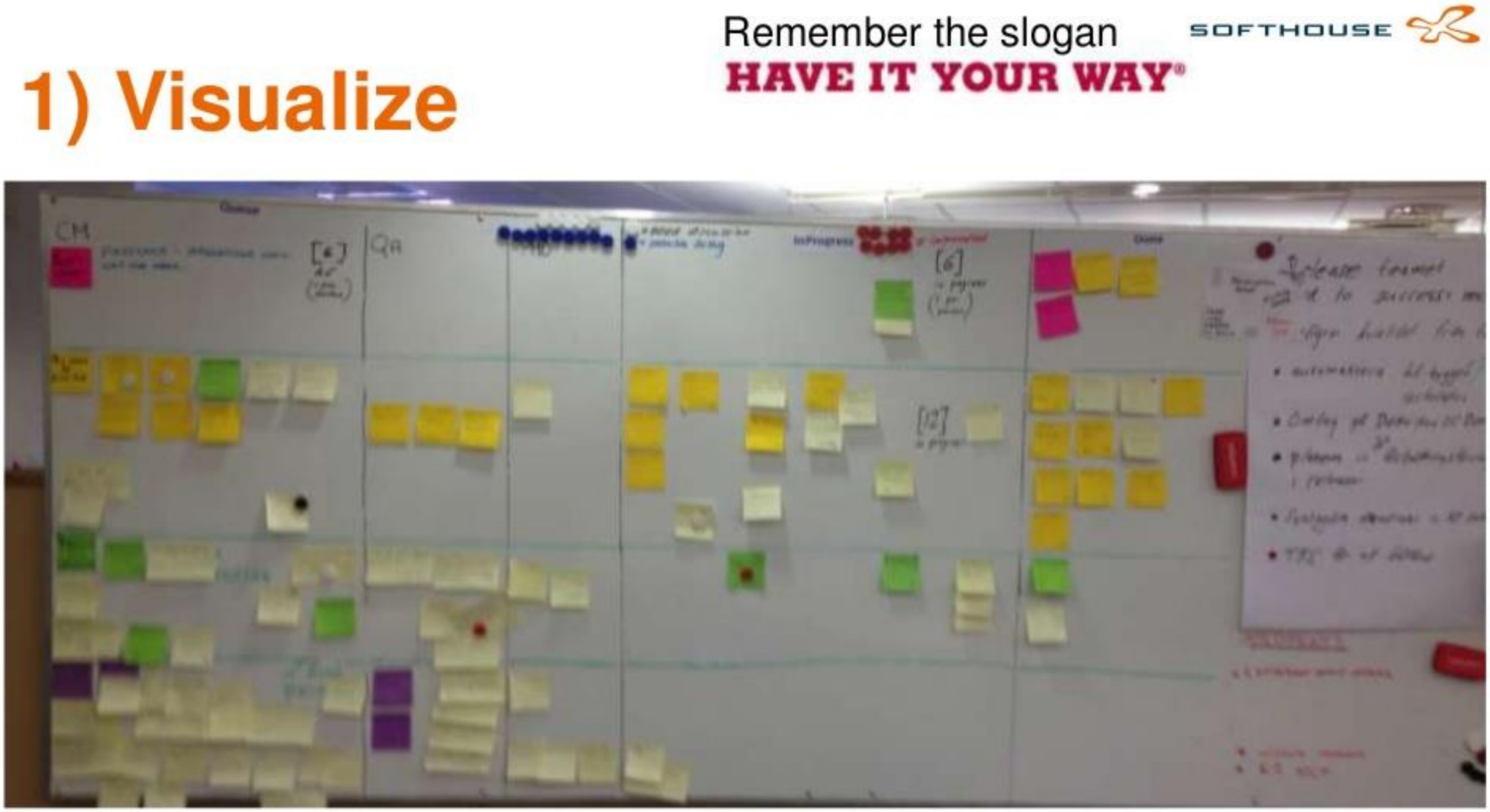
| Backlog | Development | Testing | Deployment | Done |
|---|---|---|---|---|
| G<br>H | D<br>E<br>F | B<br>C | | A |

## KANBAN



| Backlog | Development<br>3 | Testing<br>2 | Deployment<br>1 | Done |
|---|---|---|---|---|
| F<br>G<br>H | D<br>E | C | B | A |

# Core Kanban Practices

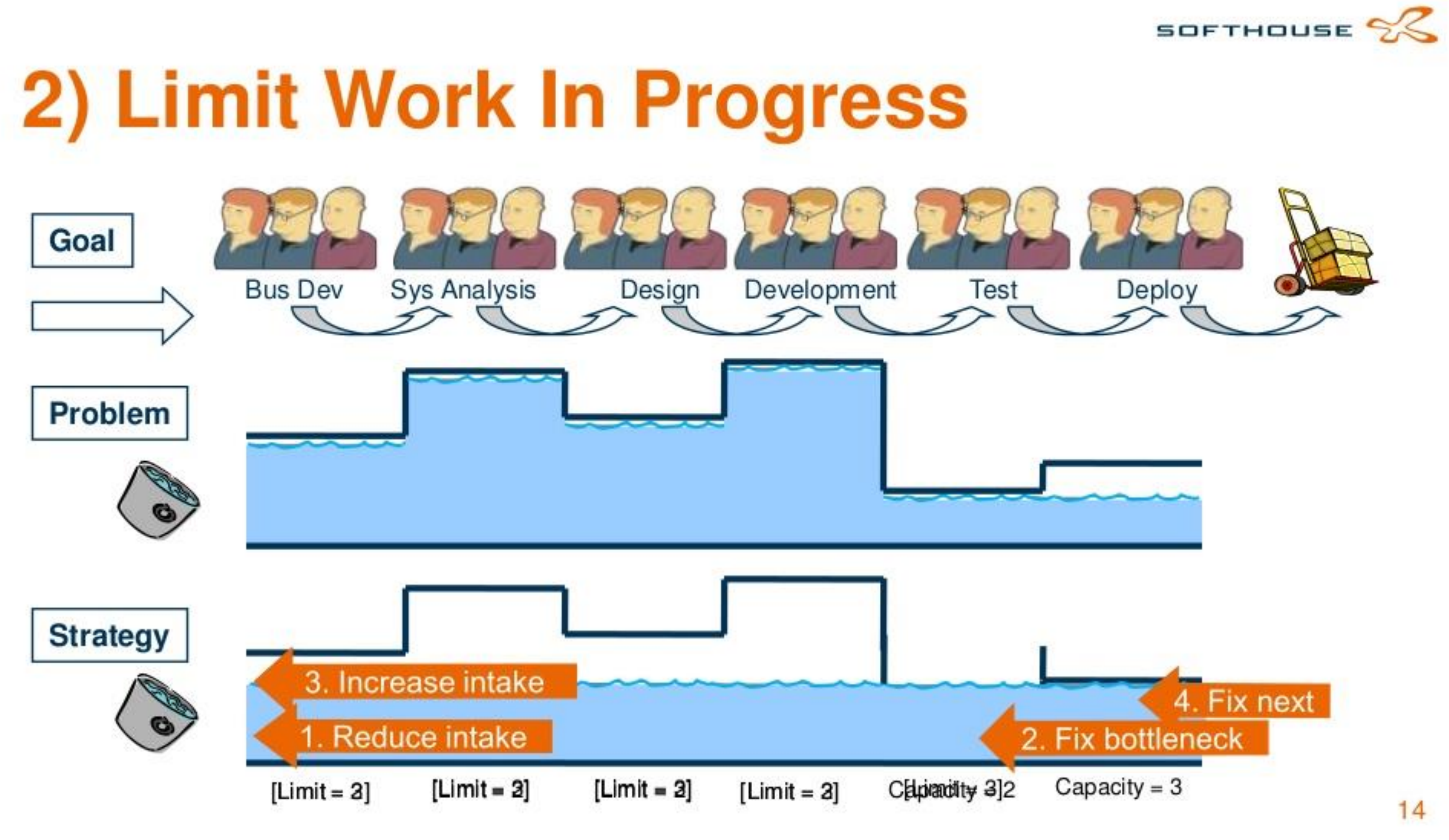


1) Visualize

Remember the slogan
**HAVE IT YOUR WAY®**

SOFTHOUSE

# Core Kanban Practices (cont.)



SOFTHOUSE

## 2) Limit Work In Progress

Goal

Bus Dev

Sys Analysis

Design

Development

Test

Deploy

Problem

Strategy

3. Increase intake

1. Reduce intake

4. Fix next

2. Fix bottleneck

[Limit = 2]

[Limit = 3]

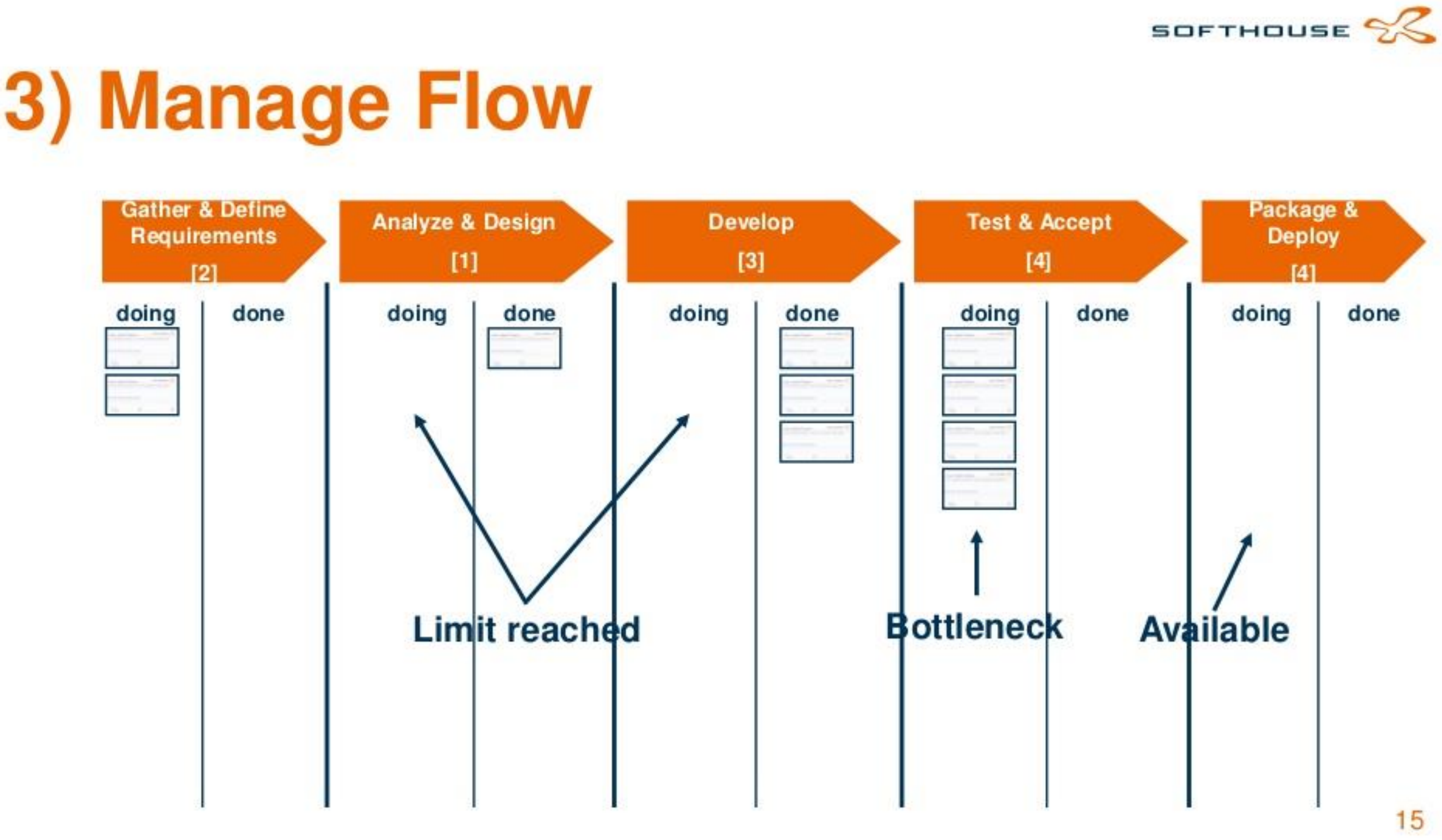
[Limit = 3]

[Limit = 2]

Capacity = 2 [Limit = 3]

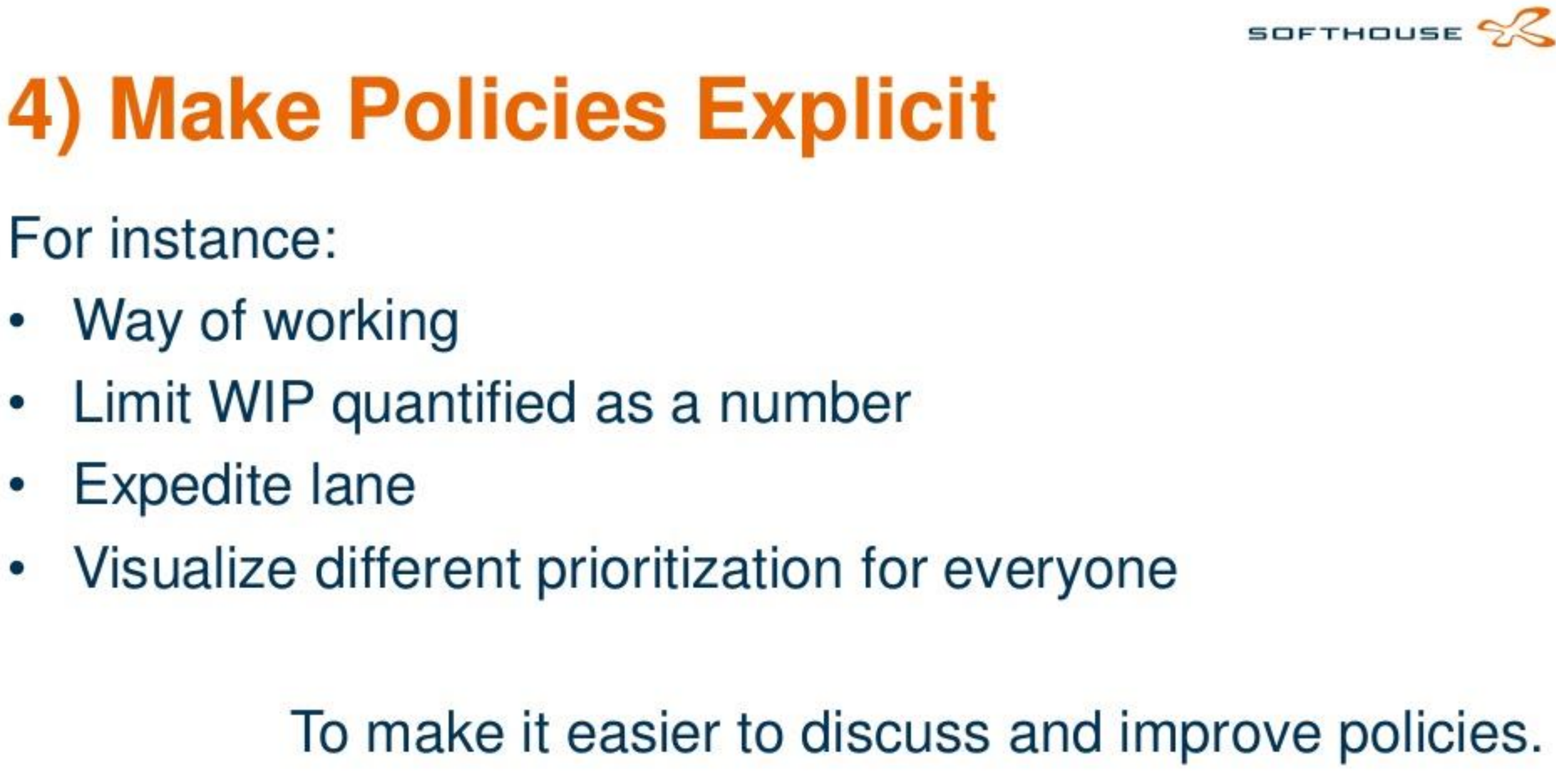
Capacity = 3

# Core Kanban Practices (cont.)



## 3) Manage Flow

# Core Kanban Practices (cont.)



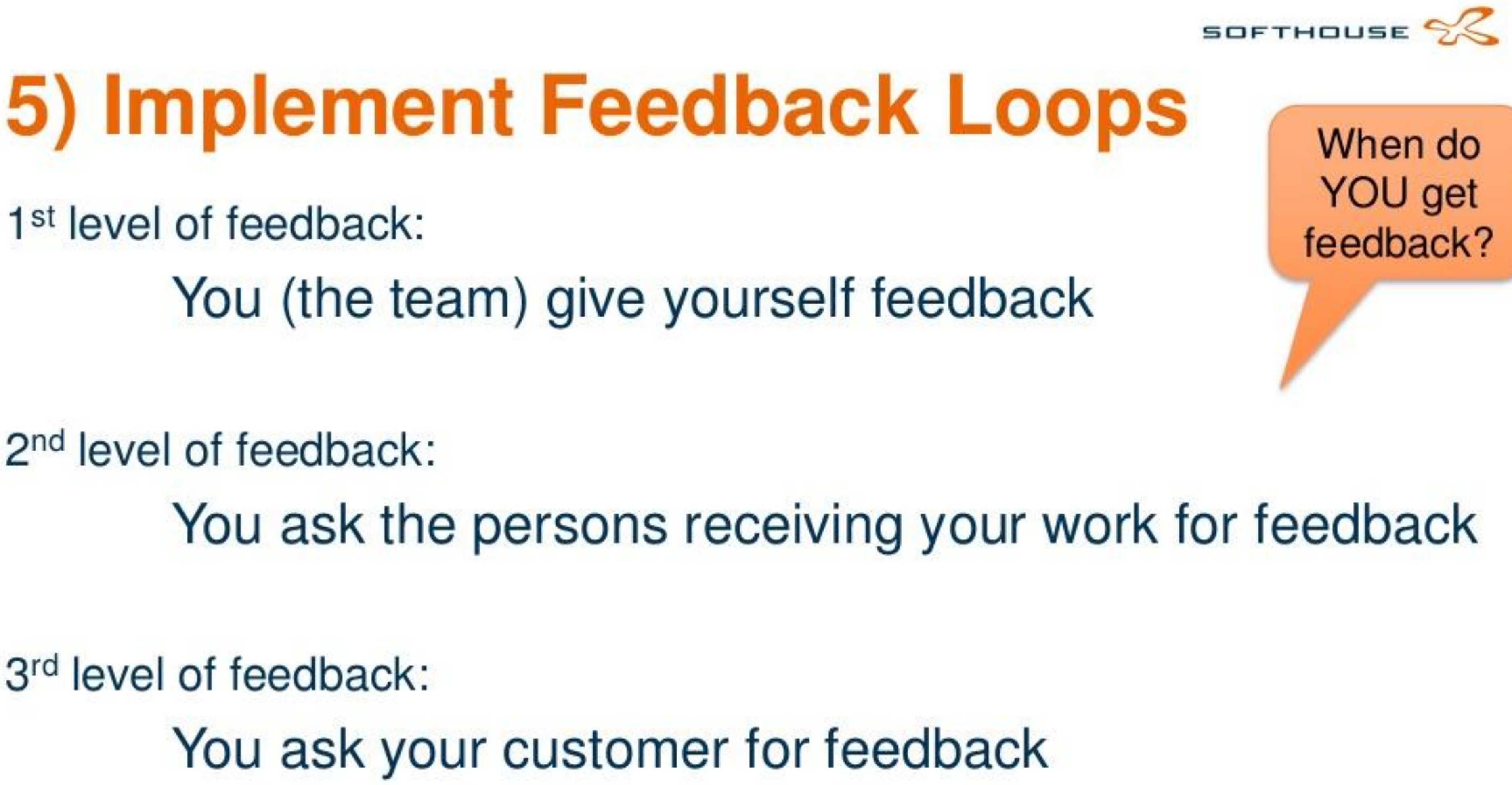
## 4) Make Policies Explicit

For instance:

- Way of working
- Limit WIP quantified as a number
- Expedite lane
- Visualize different prioritization for everyone

To make it easier to discuss and improve policies.

# Core Kanban Practices (cont.)



SOFTHOUSE

## 5) Implement Feedback Loops

When do YOU get feedback?

1st level of feedback:

You (the team) give yourself feedback

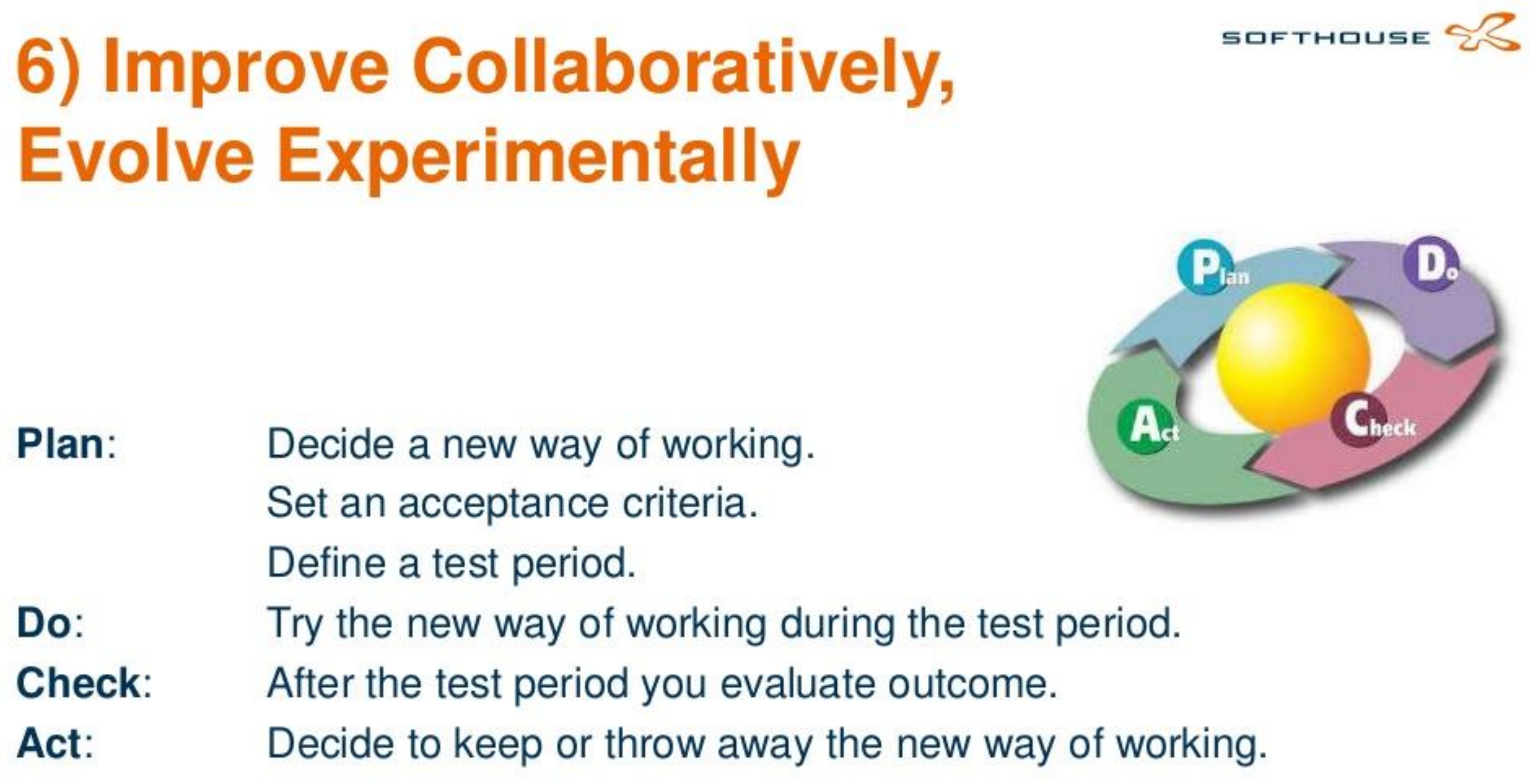
2nd level of feedback:

You ask the persons receiving your work for feedback

3rd level of feedback:

You ask your customer for feedback

# Core Kanban Practices (cont.)



## 6) Improve Collaboratively, Evolve Experimentally

Plan
Do
Check
Act

**Plan**: Decide a new way of working.
Set an acceptance criteria.
Define a test period.

**Do**: Try the new way of working during the test period.

**Check**: After the test period you evaluate outcome.

**Act**: Decide to keep or throw away the new way of working.

# Scrum vs. Kanban

- https://youtu.be/9Jgu1BlTlSc
- https://youtu.be/n2ZrUQNwrUk
- https://youtu.be/sOlFPi5xJqQ
- https://youtu.be/rIaz-l1Kf8w