# COMP20050 - Software Engineering Project II

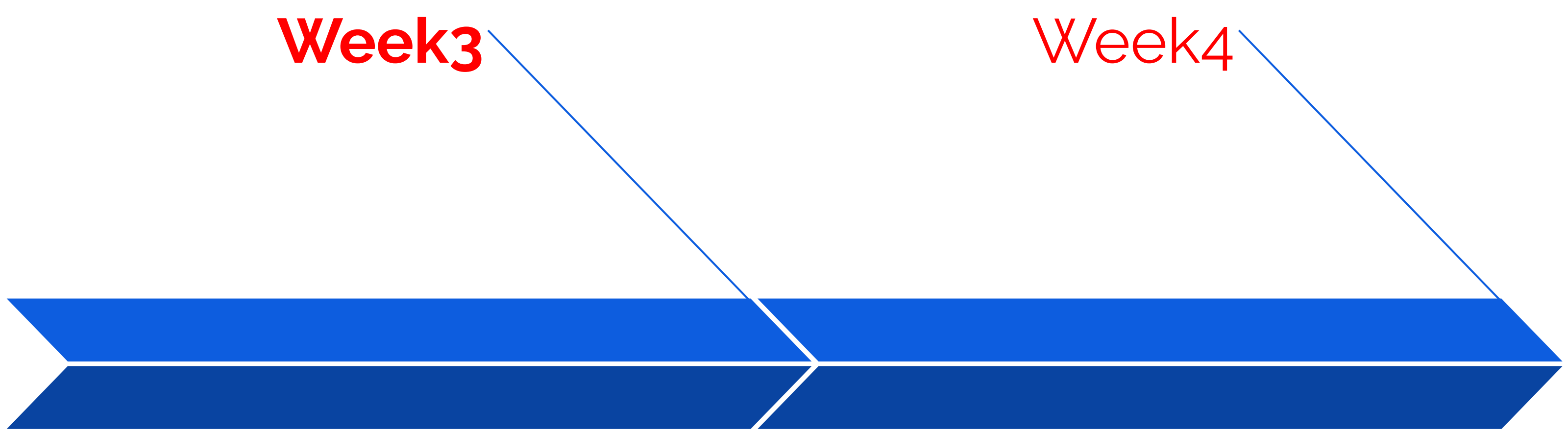
## Scrum Software Engineering Framework

**Ravi Reddy Manumachu**
ravi.manumachu@ucd.ie

UCD School of Computer Science.

**Scoil na Ríomheolaíochta UCD.**

# COMP20050 - Weeks 3 & 4



**Week3**

Week4

**Scrum**

Software Architecture

**Scrum Software Engineering Framework**

Software Architecture

JavaFX

JavaFX's SceneBuidler

Introduction to JavaFX

Introduction to JavaFX SceneBuilder

# Outline (Learning Objectives)

- Understand the core concepts of Scrum software engineering framework.
- Understand the Cynefin framework to learn where Scrum can be useful.
- Become familiar with the project tracking and communication tools.

# What is Scrum?

- **Scrum** is the most popular **agile** approach for developing innovative products and services.
- **Scrum** is not a standardized process but a framework for organizing and managing work.

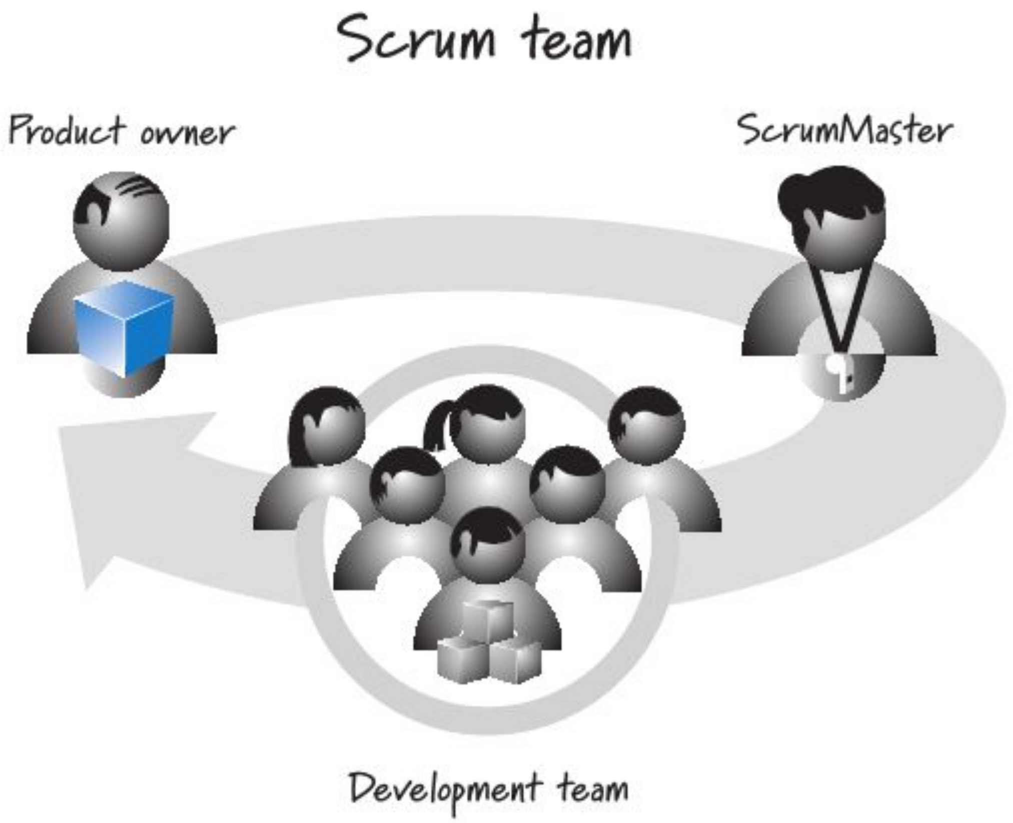
# Scrum Origins

- The term **Scrum** originated in a 1986 Harvard Business Review article, "**The New New Product Development Game**" (Takeuchi and Nonaka 1986).

*The traditional sequential or 'relay race' approach to product development - exemplified by the National Aeronautics and Space Administration's phased program planning (PPP) system - may conflict with the goals of maximum speed and flexibility. Instead, a holistic or 'rugby' approach - where a team tries to go the distance as a unit, passing the ball back and forth - may better serve today's competitive requirements.*

- Jeff Sutherland and Ken Schwaber codified and published **Scrum** in 1995.

# Scrum Roles



- A **Scrum team** has three roles:
  - 1 **Product owner**.
  - 1 **ScrumMaster**.
  - **Development team** (small team of developers).

# Product Owner

- The **product owner** is the single authority responsible for deciding which features and functionality to build and the order in which to build them.
- The **product owner** maintains and communicates to all other participants (stakeholders) a clear vision of what the Scrum team is trying to achieve.
- The **product owner** is responsible for the overall success of the solution being developed or maintained.
- The **product owner** actively collaborates with the ScrumMaster and development team.

# ScrumMaster

- The **ScrumMaster** champions the Scrum approach.
- A leader, not a manager (can't hire and fire the developers).
- As a facilitator, helps the team resolve issues.
- Responsible for removing impediments to team productivity.

# Development Team

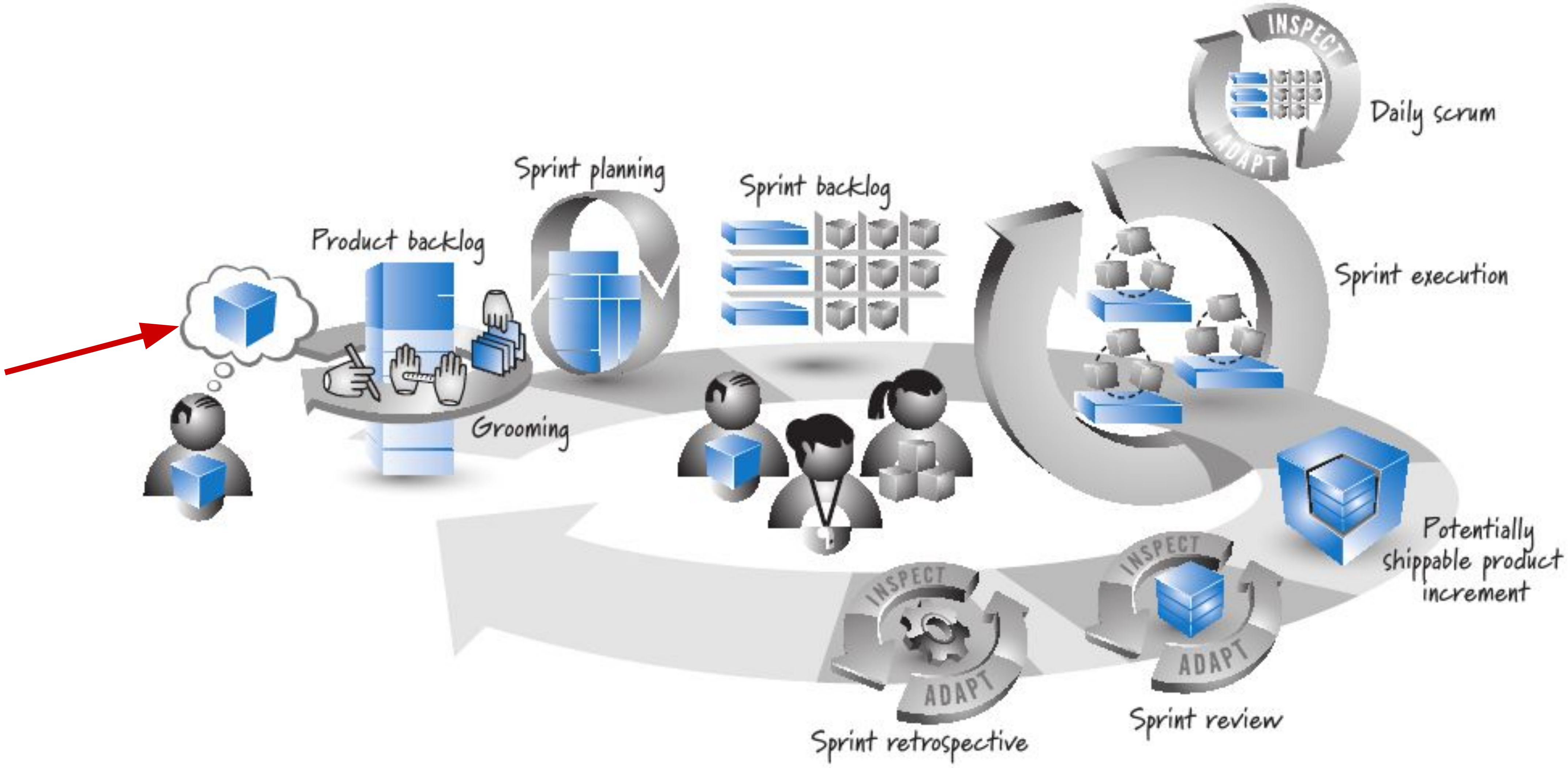
- Scrum defines the **development team** as a diverse, cross-functional collection (architect, programmer, database designer, user interface designer) who design, build and test the product.

- The **development team** self-organizes to determine the best way to accomplish the goal set out by the product owner.

# Scrum Roles in Your Group

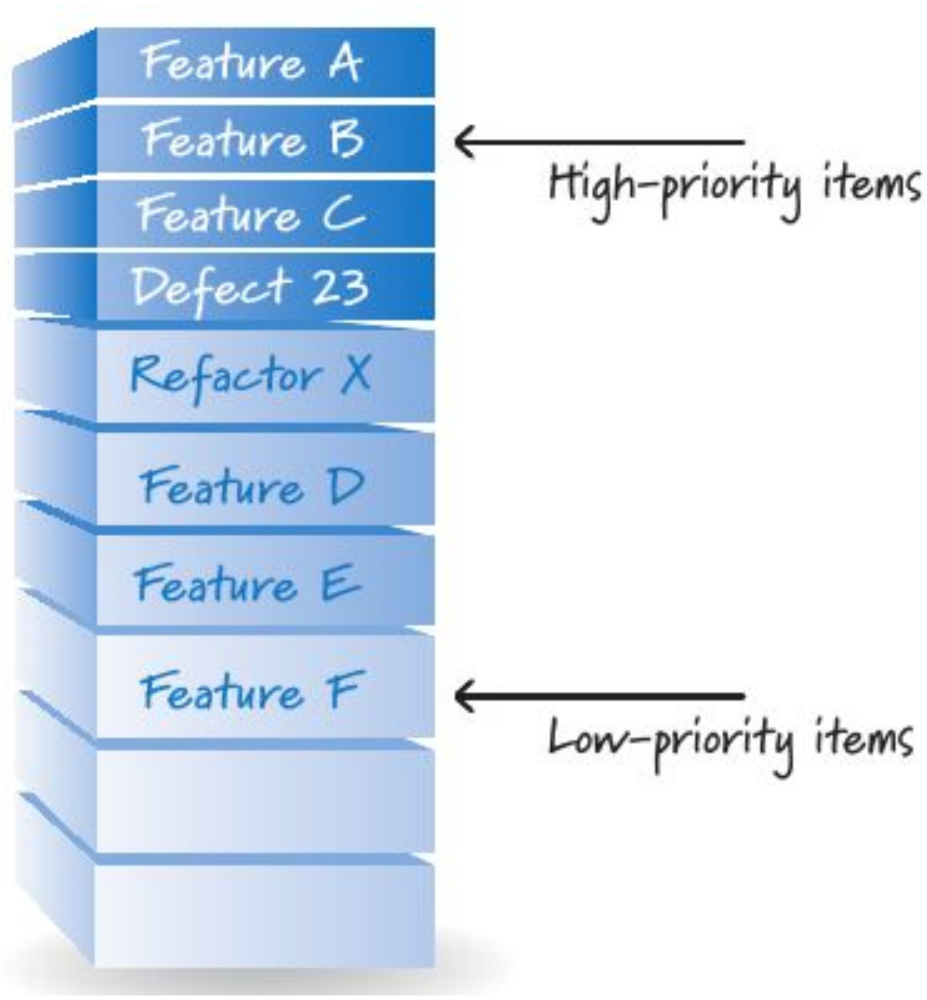
- Lecturer will be the **product owner**.
- The group members will form the **development team**.
- For each sprint, one group member can volunteer to be the **ScrumMaster**.

# Scrum Activities and Artifacts



- The **product owner** has a vision of what to create (**the big cube**).
- The **cube** is broken down into a set of features that are collected into a prioritized list called the **product backlog** through an activity called **grooming**.

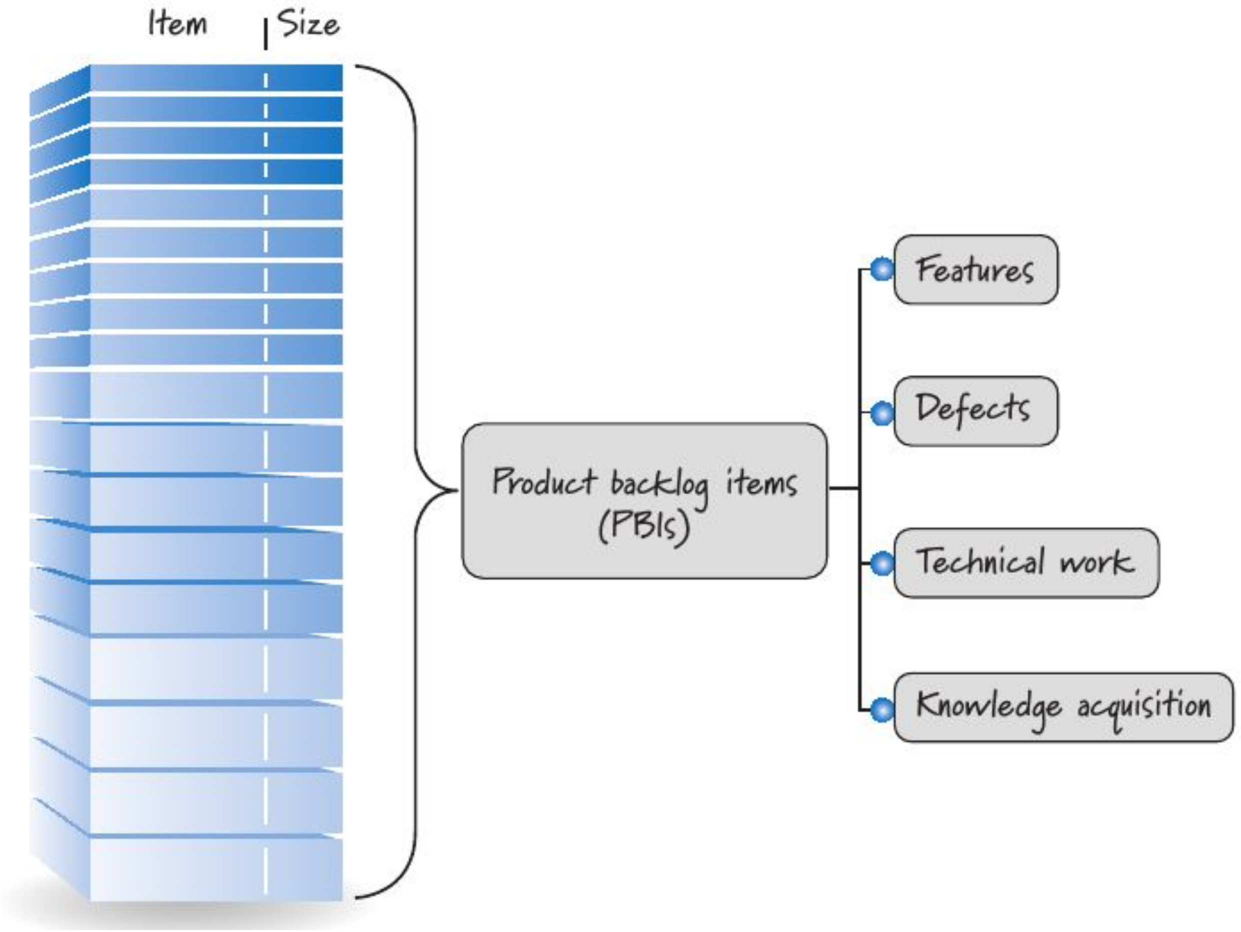
# Product Backlog



- The product owner is responsible for determining and managing the sequence of this work in the form of a prioritized (or ordered) list known as the **product backlog**.

- **High-value items** appear at the top of the **product backlog** and the **lower-value items** appear toward the bottom.

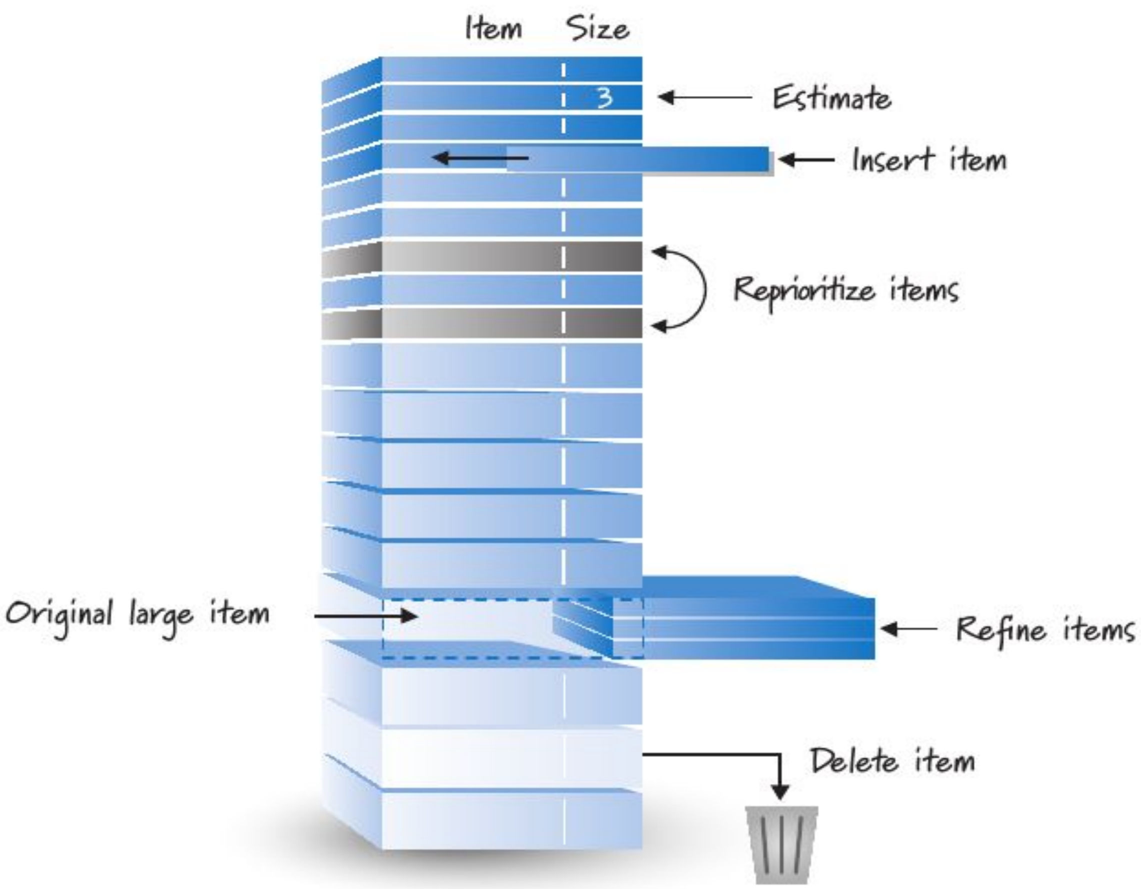
# Product Backlog Items (PBIs)



- Most **PBIs** are **features**, items of functionality that will have tangible value to the user or customer.

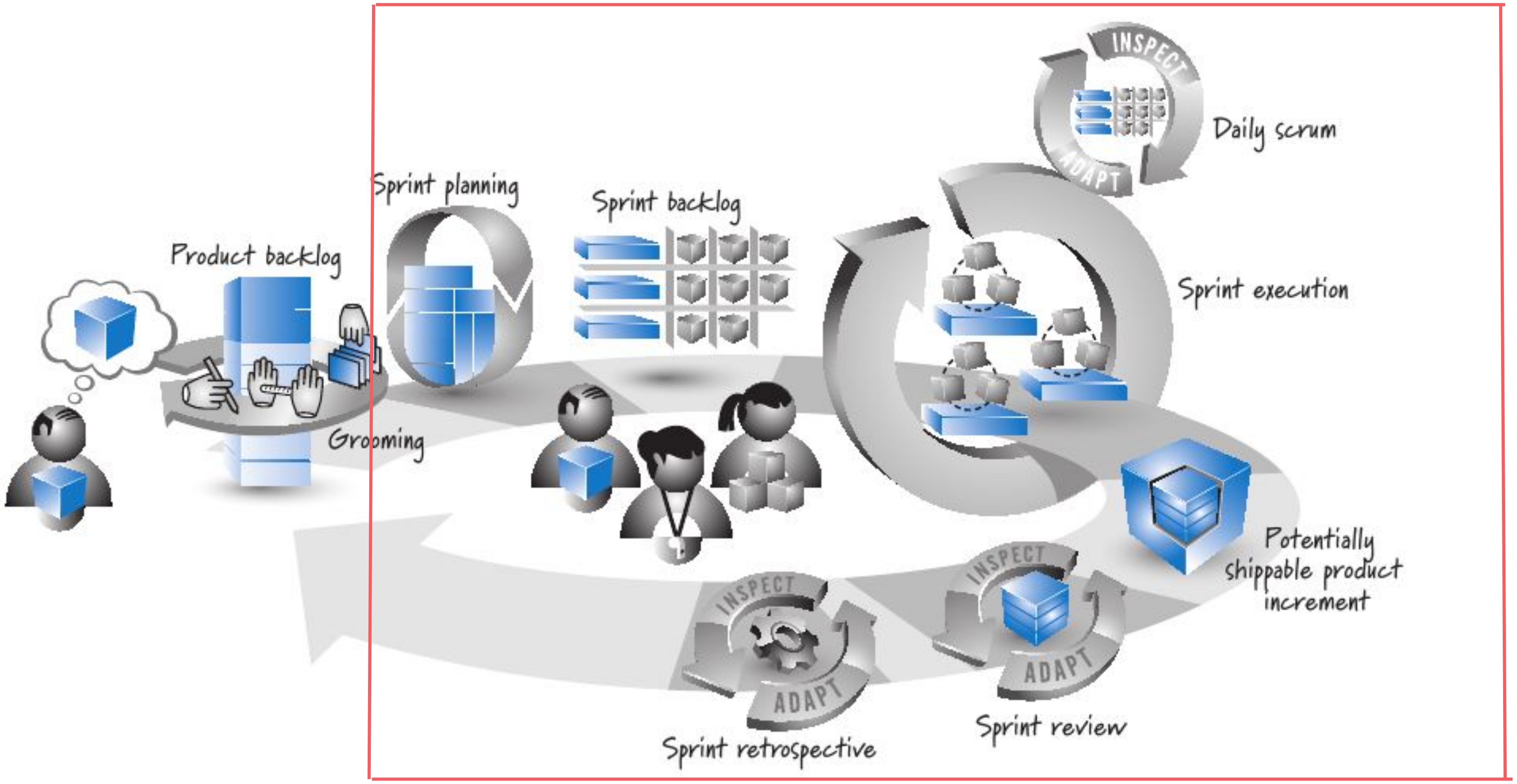
- Other **PBIs** include defects needing repair, technical improvements, and any other work the product owner deems valuable.

# Product Backlog Evolution



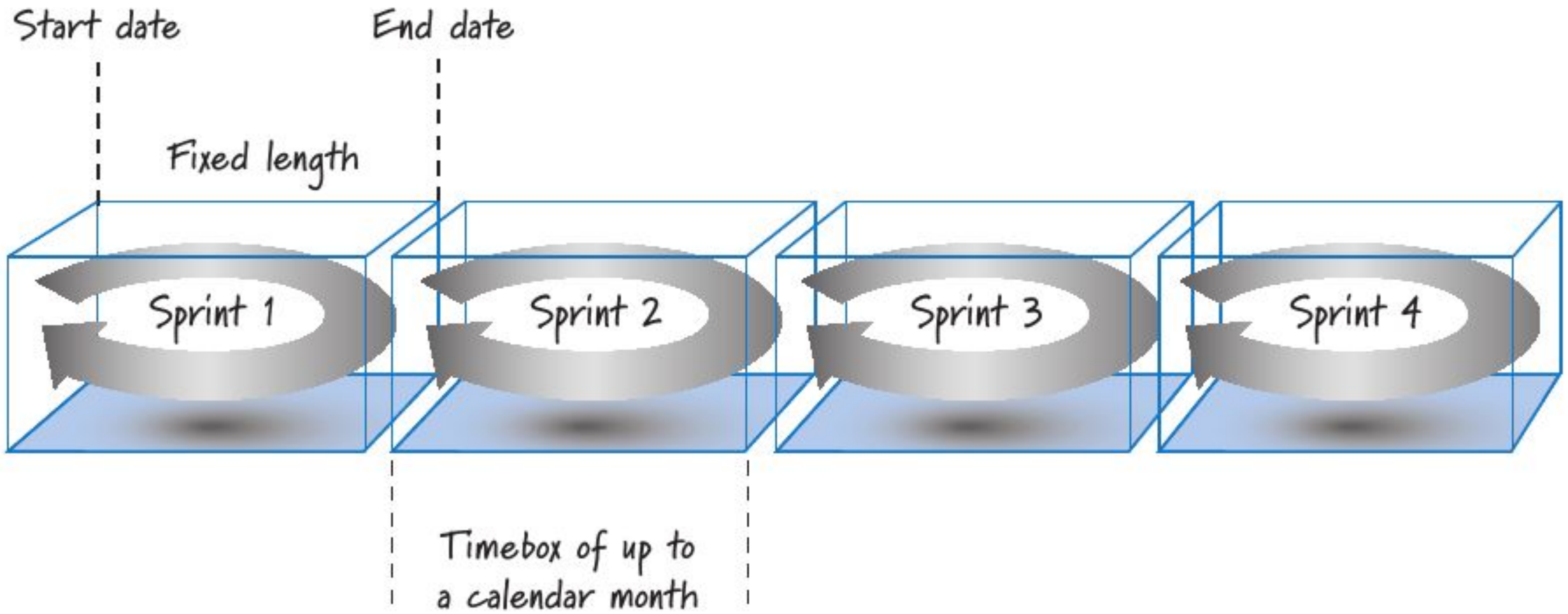
- The **Product Backlog** changes over the duration of the project as:
  - Items get **done** and removed from the backlog.
  - New items are identified and added.
  - The team spots defects that need fixes.
  - Vague long term items get clarified and split into many simpler, do-able, near term items.

# Sprints in Scrum



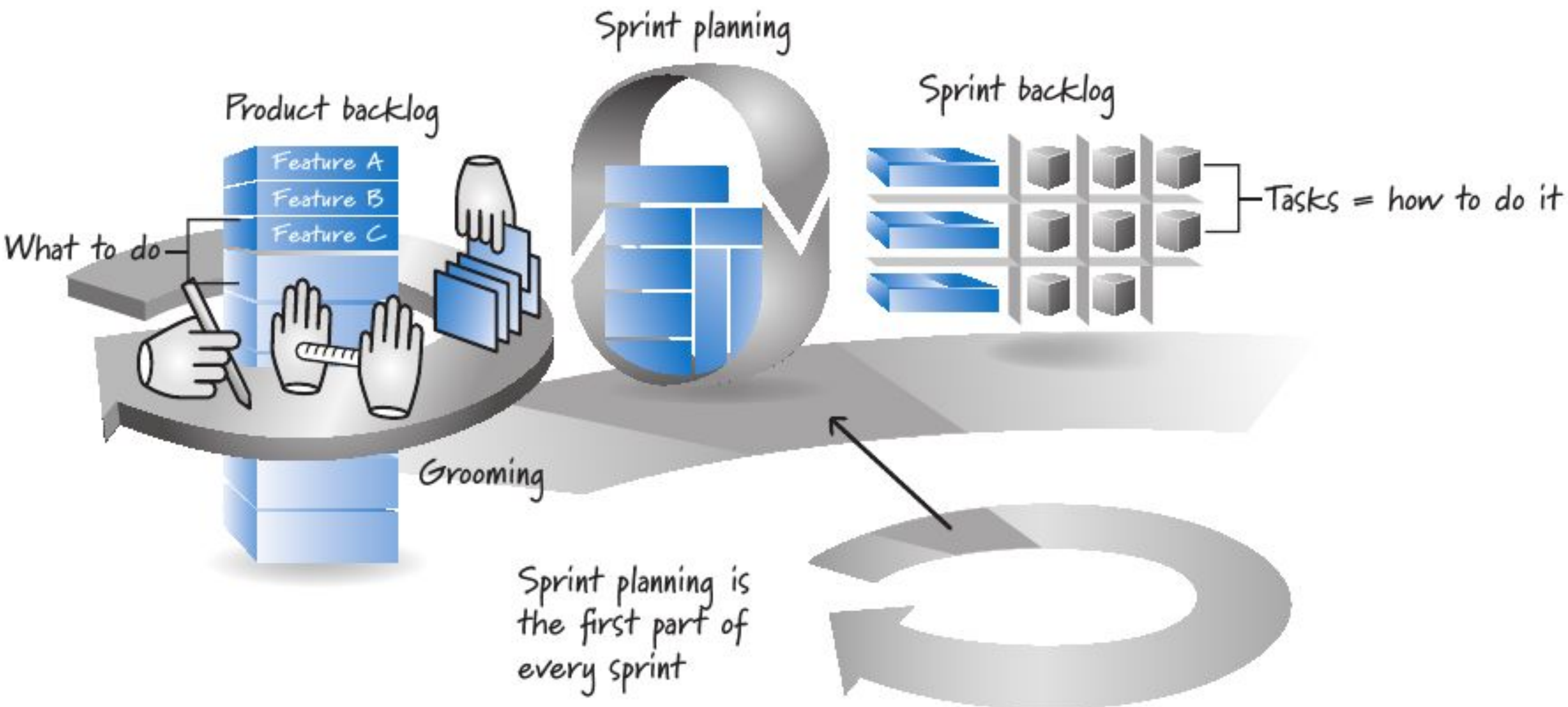
- In Scrum, work is performed in **iterations** or **cycles** of up to a calendar month called **sprints**.

# Sprints



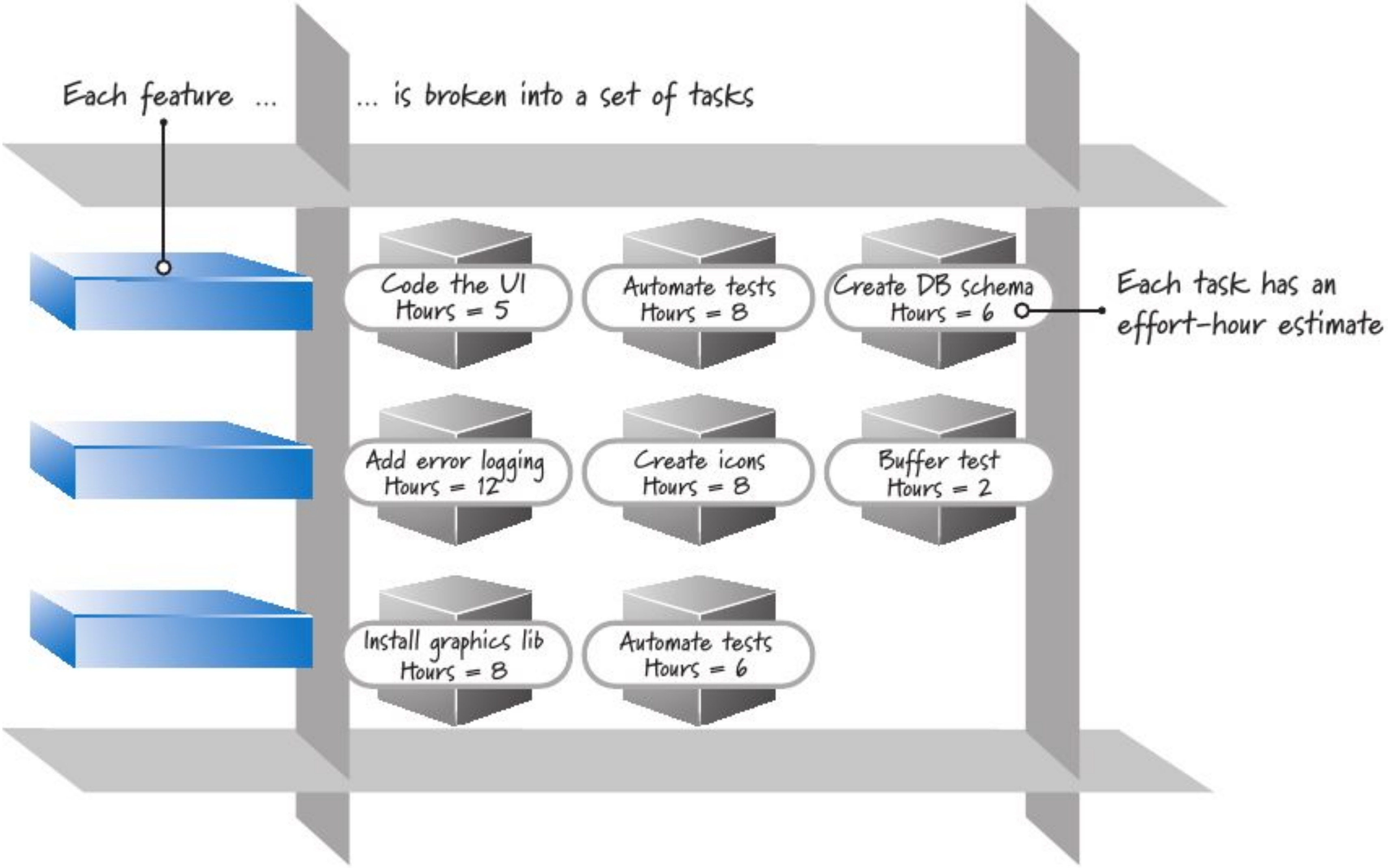
- The work completed in each sprint should create something of tangible value to the customer or user.
- Sprints are **timeboxed** so they always have a fixed start and end date, and generally they should all be of the same duration.

# Sprint Planning



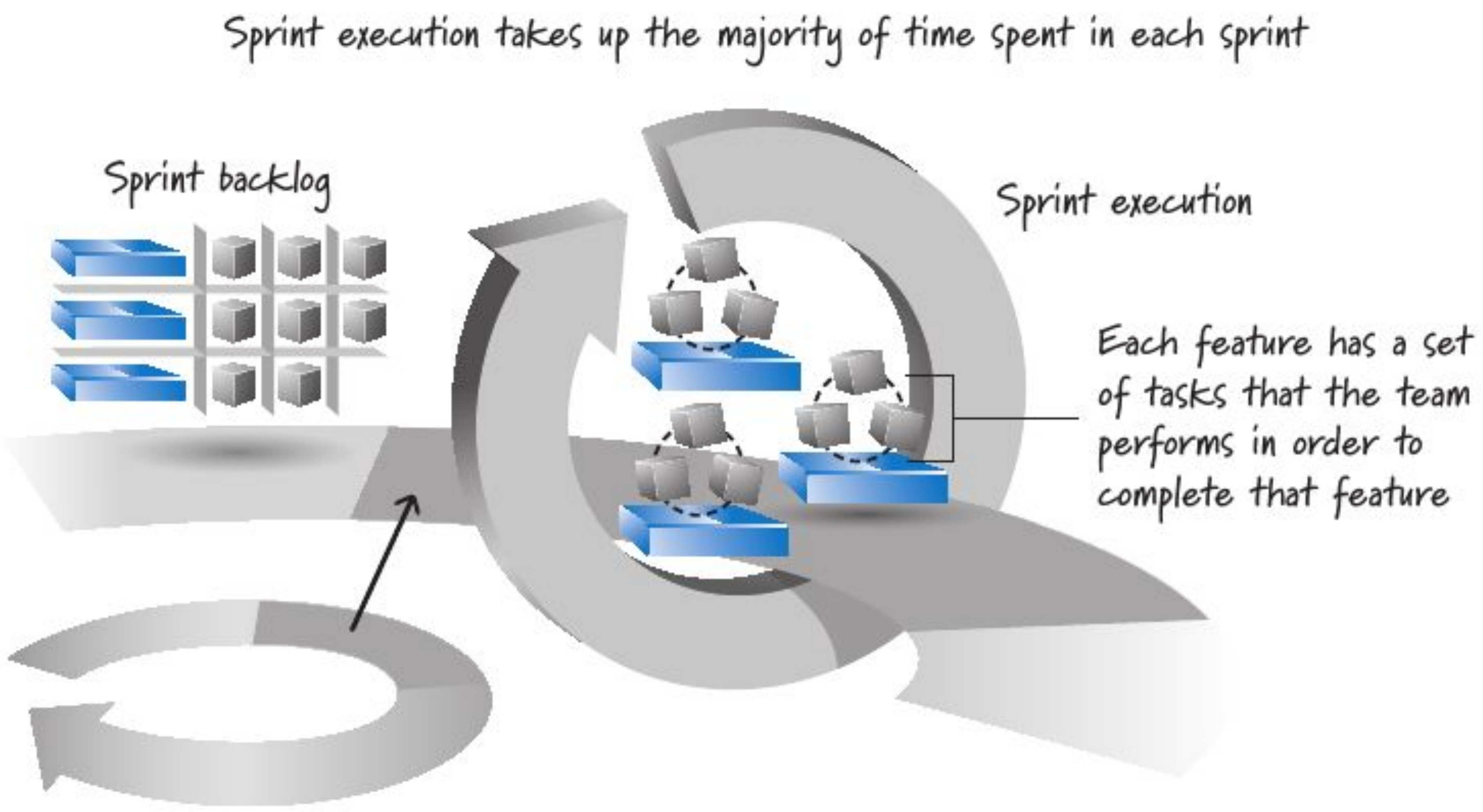
- During **sprint planning**, the product owner and development team agree on a sprint goal that defines what the upcoming sprint is supposed to achieve.

# Sprint Backlog


Each feature ...
... is broken into a set of tasks
Code the UI Hours = 5
Automate tests Hours = 8
Create DB schema Hours = 6
Each task has an effort-hour estimate
Add error logging Hours = 12
Create icons Hours = 8
Buffer test Hours = 2
Install graphics lib Hours = 8
Automate tests Hours = 6

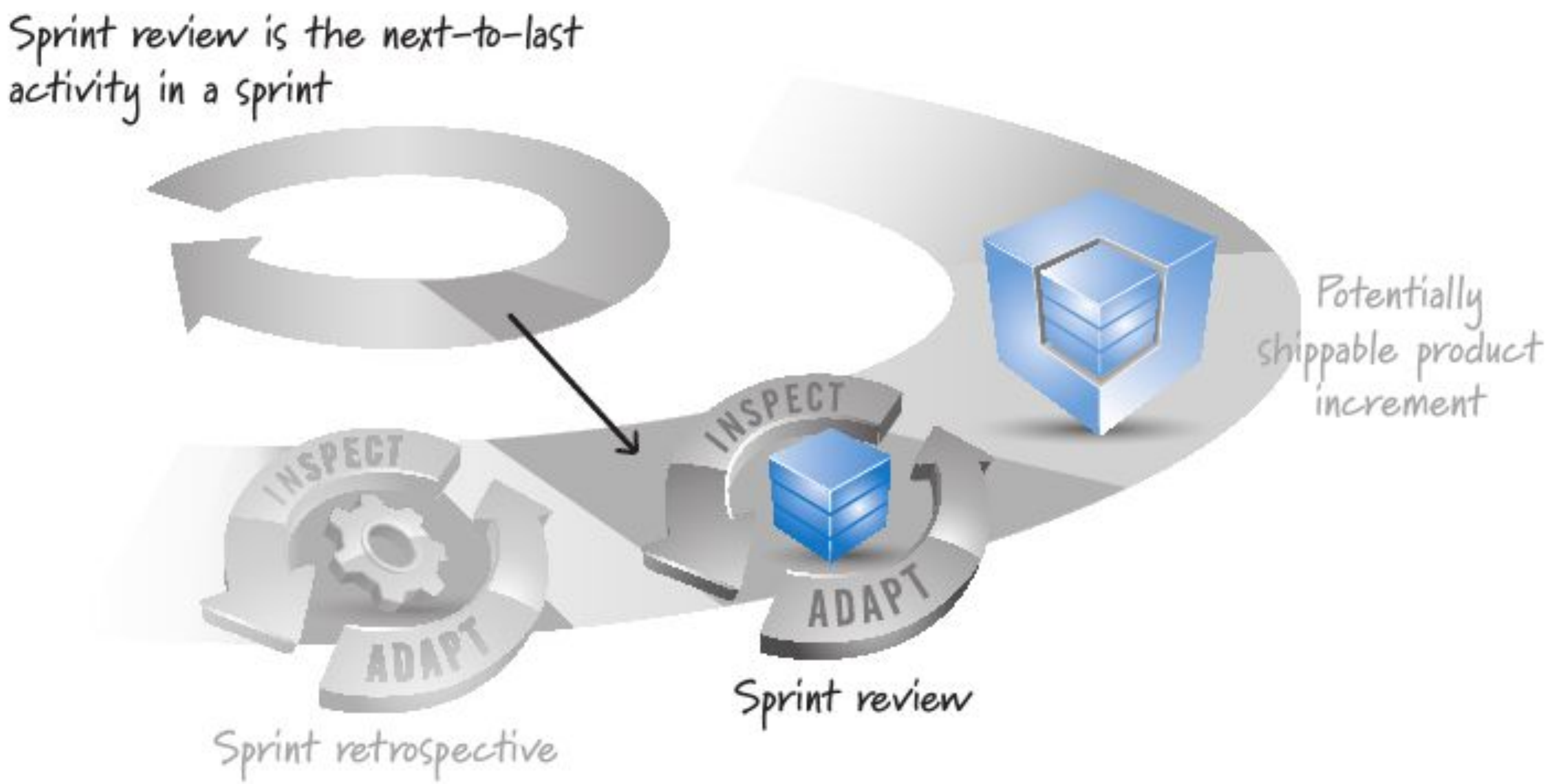
- Development teams break down each targeted feature into a set of tasks (**sprint backlog**).

# Sprint Execution


Sprint execution takes up the majority of time spent in each sprint
Sprint backlog
Sprint execution
Each feature has a set of tasks that the team performs in order to complete that feature

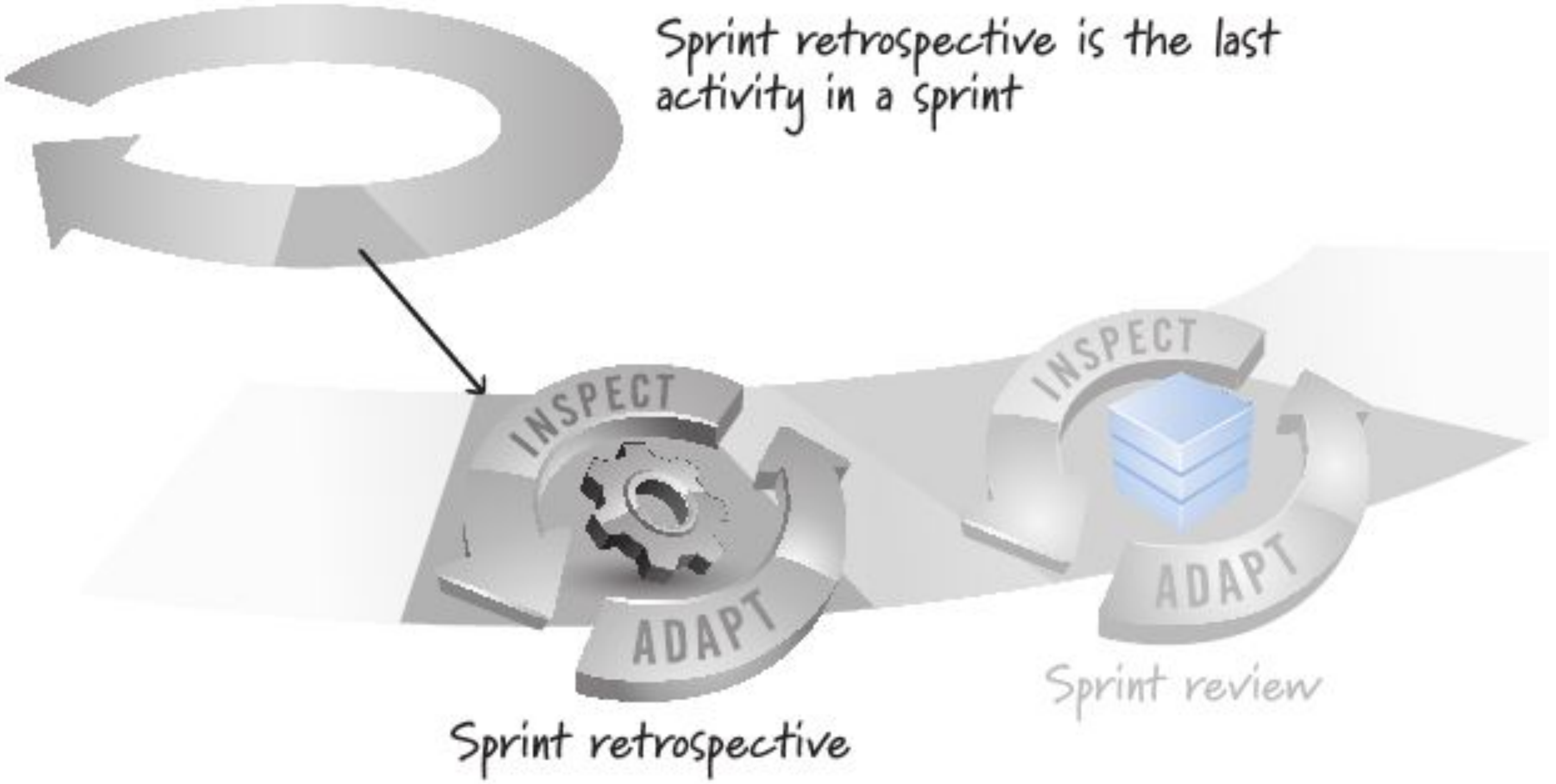
- The development team, guided by the ScrumMaster's coaching, performs all of the task-level work necessary to get the features **done**.

# Sprint Review


Sprint review is the next-to-last activity in a sprint
Potentially shippable product increment
INSPECT
ADAPT
INSPECT
ADAPT
Sprint review
Sprint retrospective

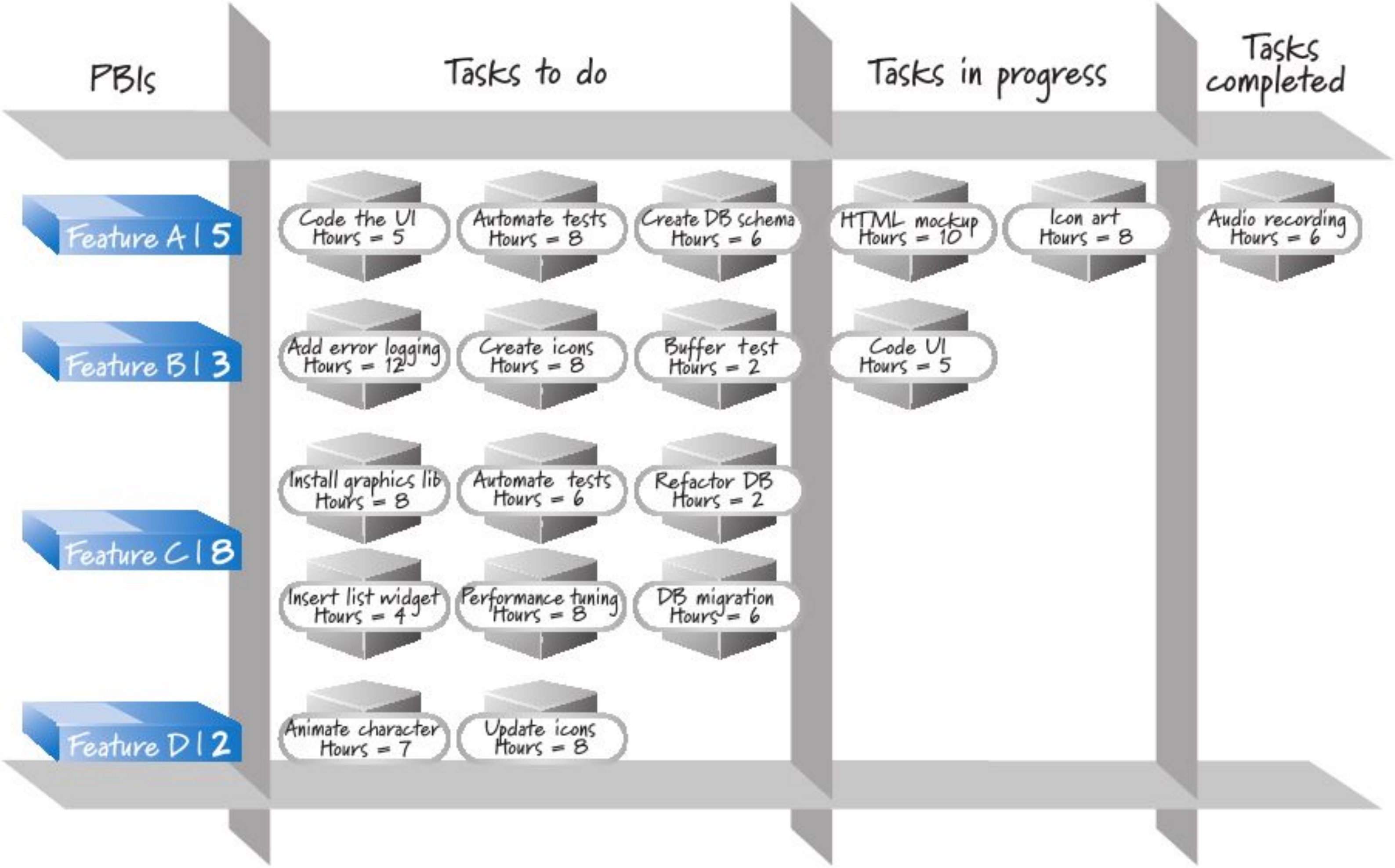
- The **sprint review** represents a scheduled opportunity to inspect and adapt the product.

# Sprint Retrospective



- **Sprint retrospective** occurs after the sprint review and before the next sprint planning.
- The development team, ScrumMaster, and product owner come together to discuss what is and is not working with Scrum.

# Communicating Progress



| PBIs | Tasks to do | Tasks in progress | Tasks completed |
|---|---|---|---|
| Feature A \| 5 | Code the UI<br>Hours = 5<br>Automate tests<br>Hours = 8<br>Create DB schema<br>Hours = 6 | HTML mockup<br>Hours = 10<br>Icon art<br>Hours = 8 | Audio recording<br>Hours = 6 |
| Feature B \| 3 | Add error logging<br>Hours = 12<br>Create icons<br>Hours = 8<br>Buffer test<br>Hours = 2 | Code UI<br>Hours = 5 | |
| Feature C \| 8 | Install graphics lib<br>Hours = 8<br>Automate tests<br>Hours = 6<br>Refactor DB<br>Hours = 2<br>Insert list widget<br>Hours = 4<br>Performance tuning<br>Hours = 8<br>DB migration<br>Hours = 6 | | |
| Feature D \| 2 | Animate character<br>Hours = 7<br>Update icons<br>Hours = 8 | | |

- The **task board** shows the evolving state of the sprint backlog over time.

# Where is Scrum Useful?

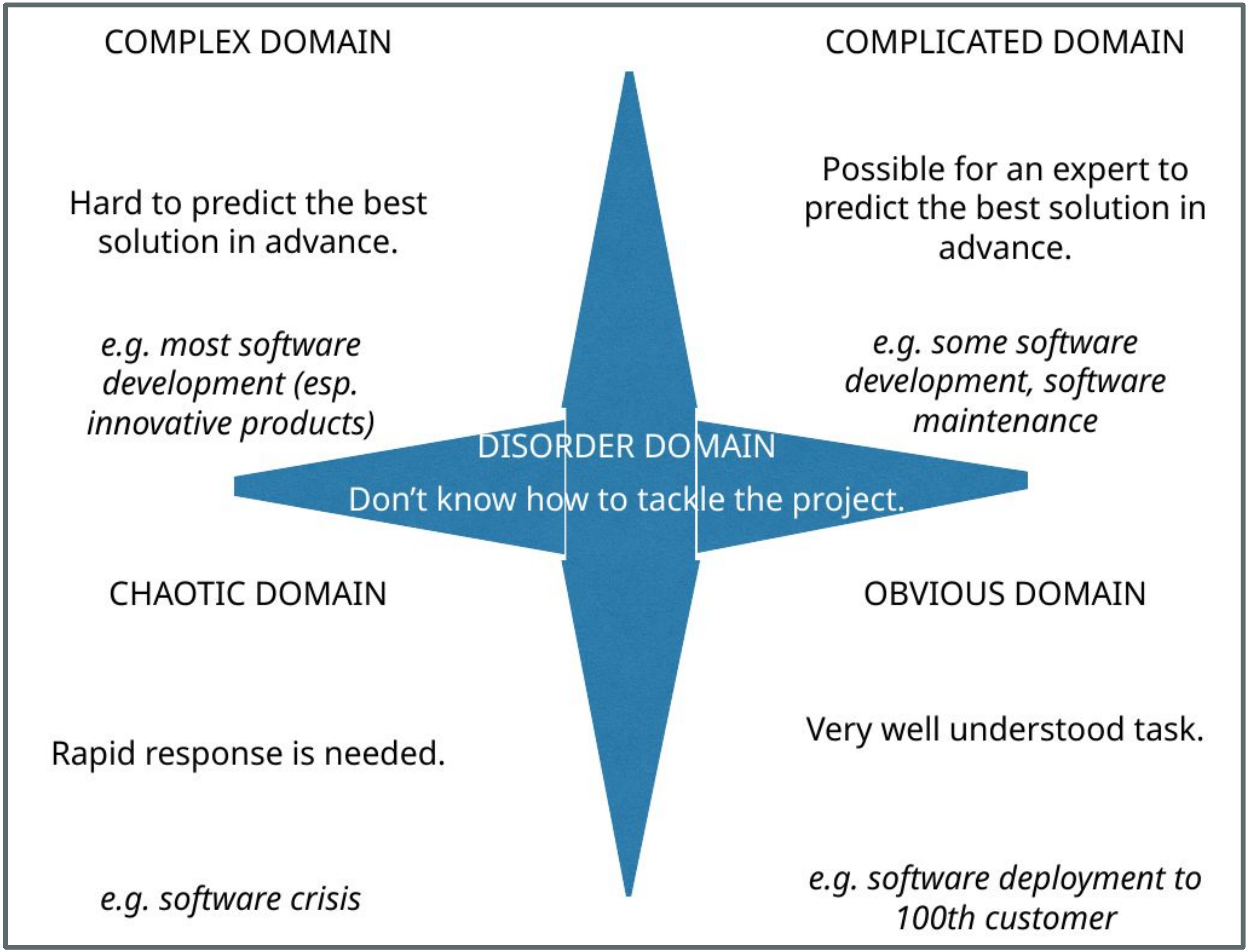
# Cynefin Framework

- The **Cynefin framework** (Snowden and Boone 2007) is a sense-making framework that helps you understand the situation and decide on a situation-appropriate approach.

- It defines and compares the characteristics of five different domains: **simple**, **complicated**, **chaotic**, **complex**, and **disorder**.

- **Disorder** domain is when you don't know which other domain you are in.

# Cynefin Framework for Software Development



COMPLEX DOMAIN

Hard to predict the best solution in advance.

*e.g. most software development (esp. innovative products)*

COMPLICATED DOMAIN

Possible for an expert to predict the best solution in advance.

*e.g. some software development, software maintenance*

DISORDER DOMAIN

Don’t know how to tackle the project.

CHAOTIC DOMAIN

Rapid response is needed.

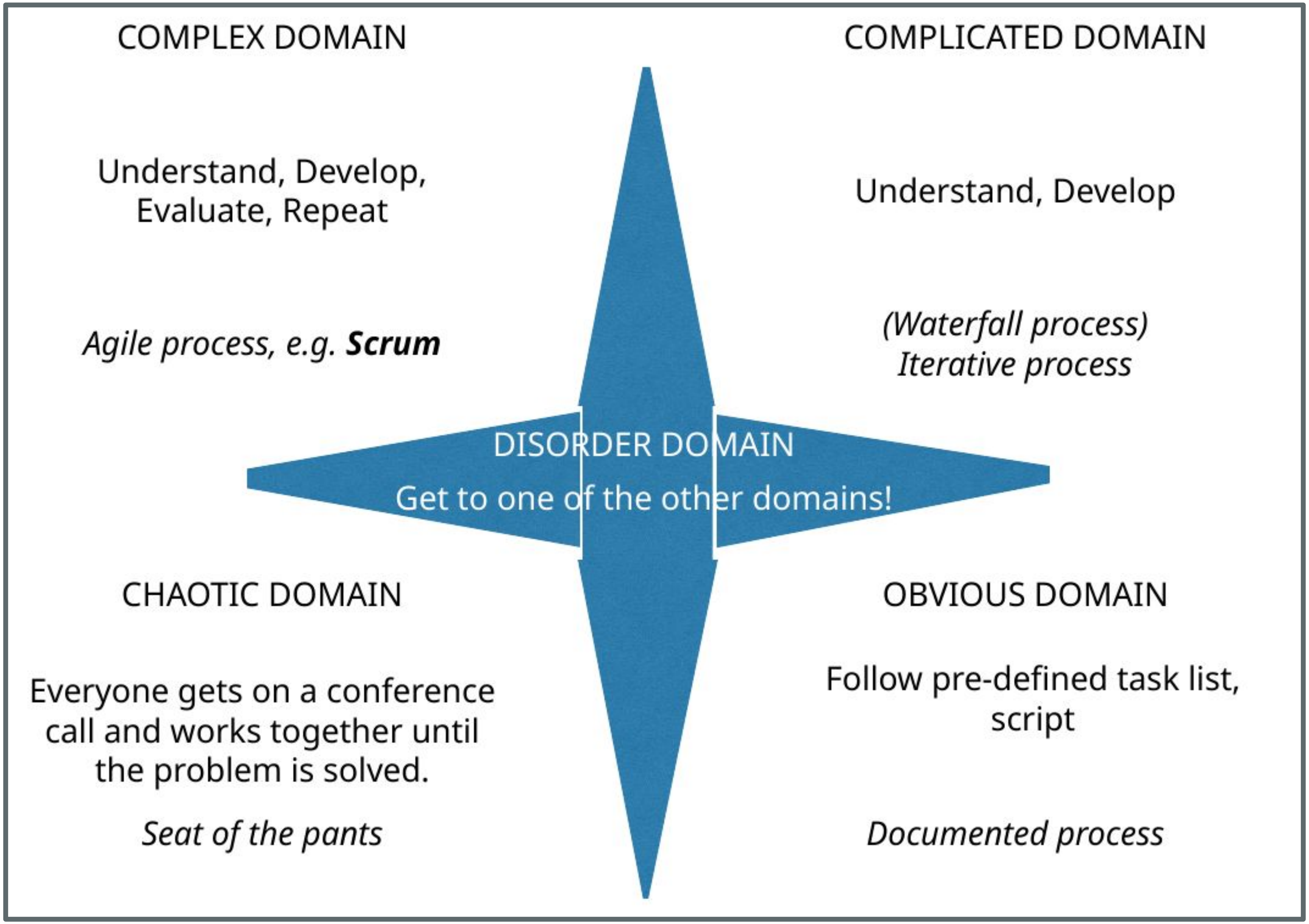
*e.g. software crisis*

OBVIOUS DOMAIN

Very well understood task.

*e.g. software deployment to 100th customer*

# Cynefin and Software Development Process



COMPLEX DOMAIN

Understand, Develop, Evaluate, Repeat

*Agile process, e.g.* ***Scrum***

COMPLICATED DOMAIN

Understand, Develop

*(Waterfall process)*
*Iterative process*

DISORDER DOMAIN

Get to one of the other domains!

CHAOTIC DOMAIN

Everyone gets on a conference call and works together until the problem is solved.

*Seat of the pants*

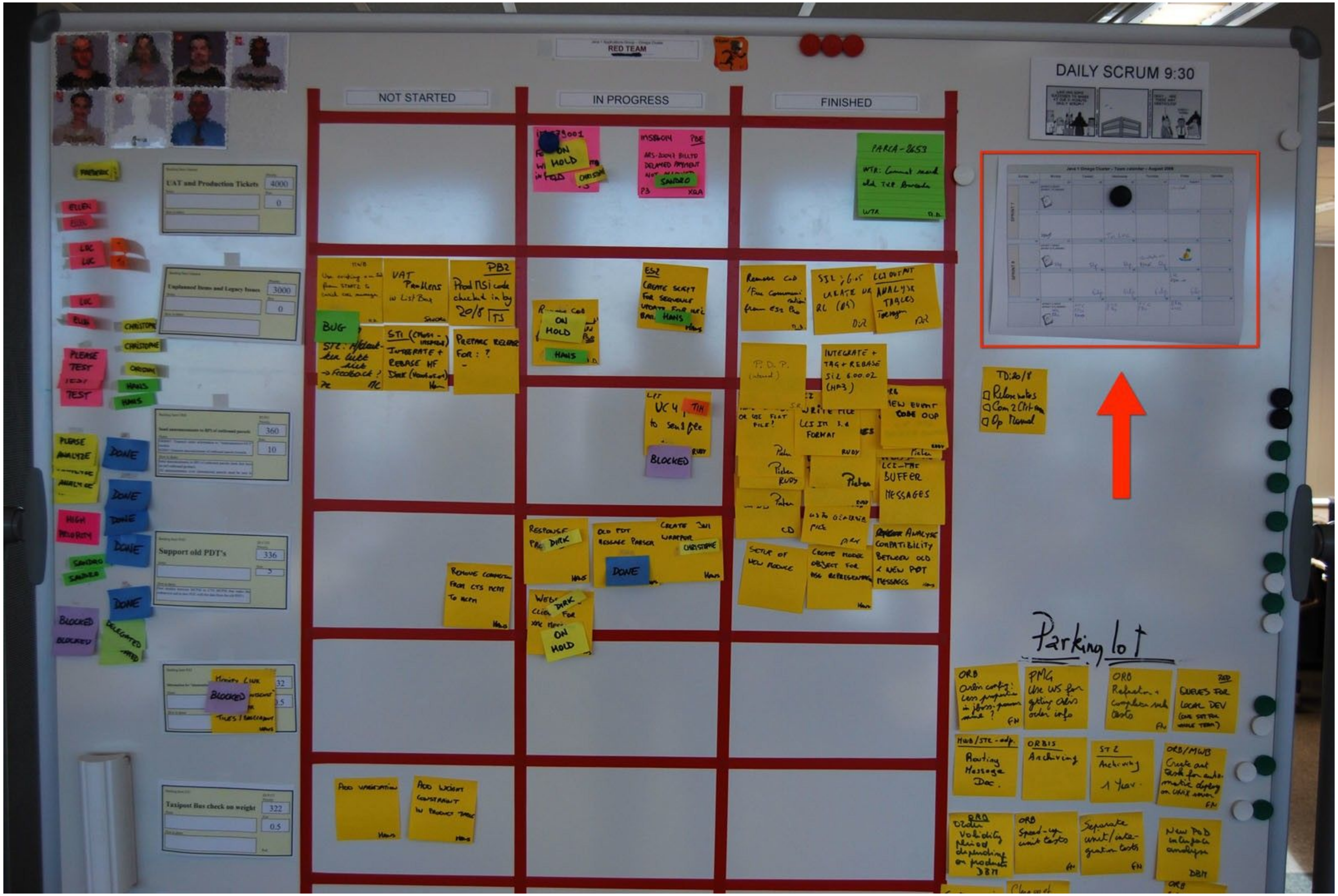
OBVIOUS DOMAIN

Follow pre-defined task list, script

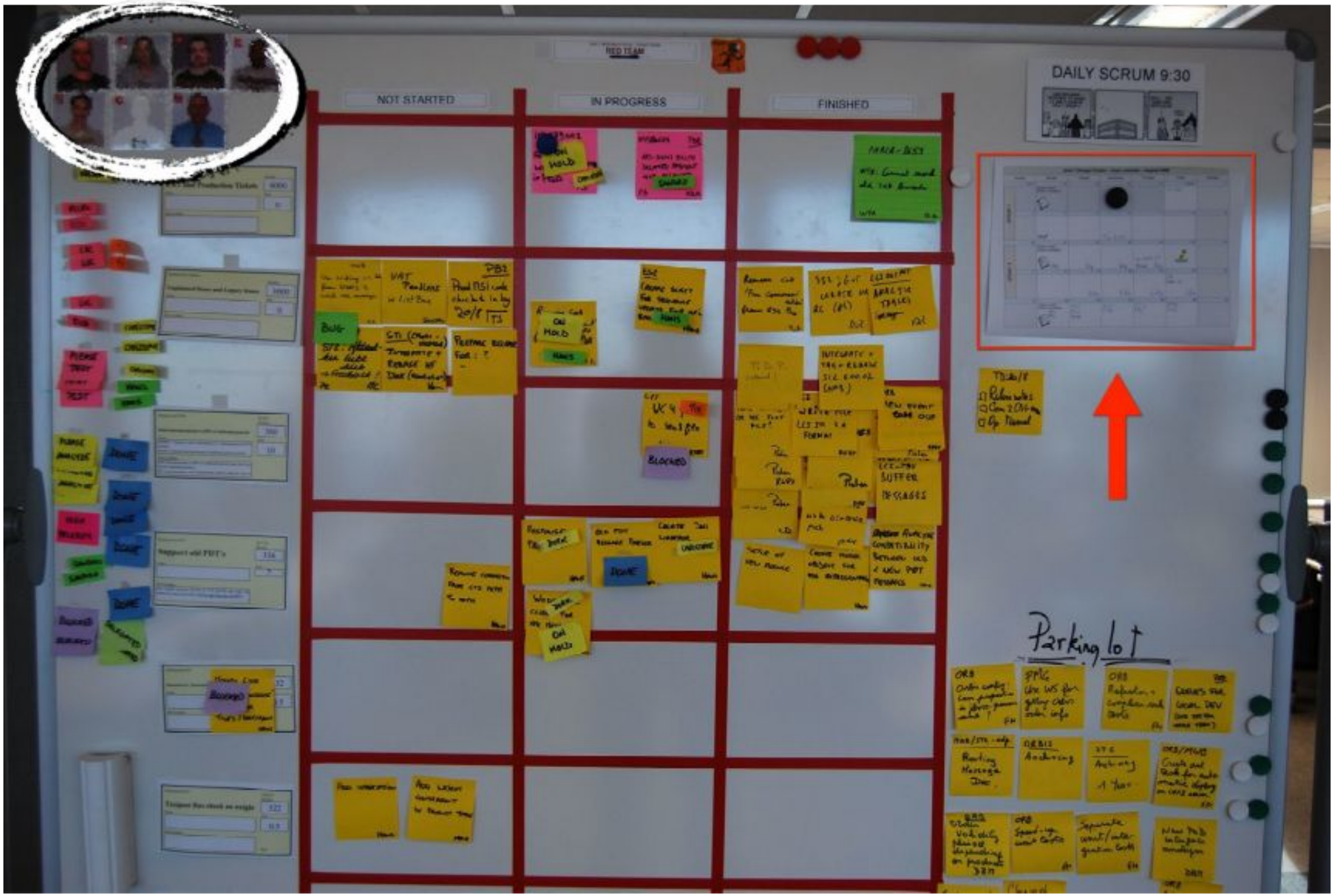
*Documented process*

# Scrum Practice

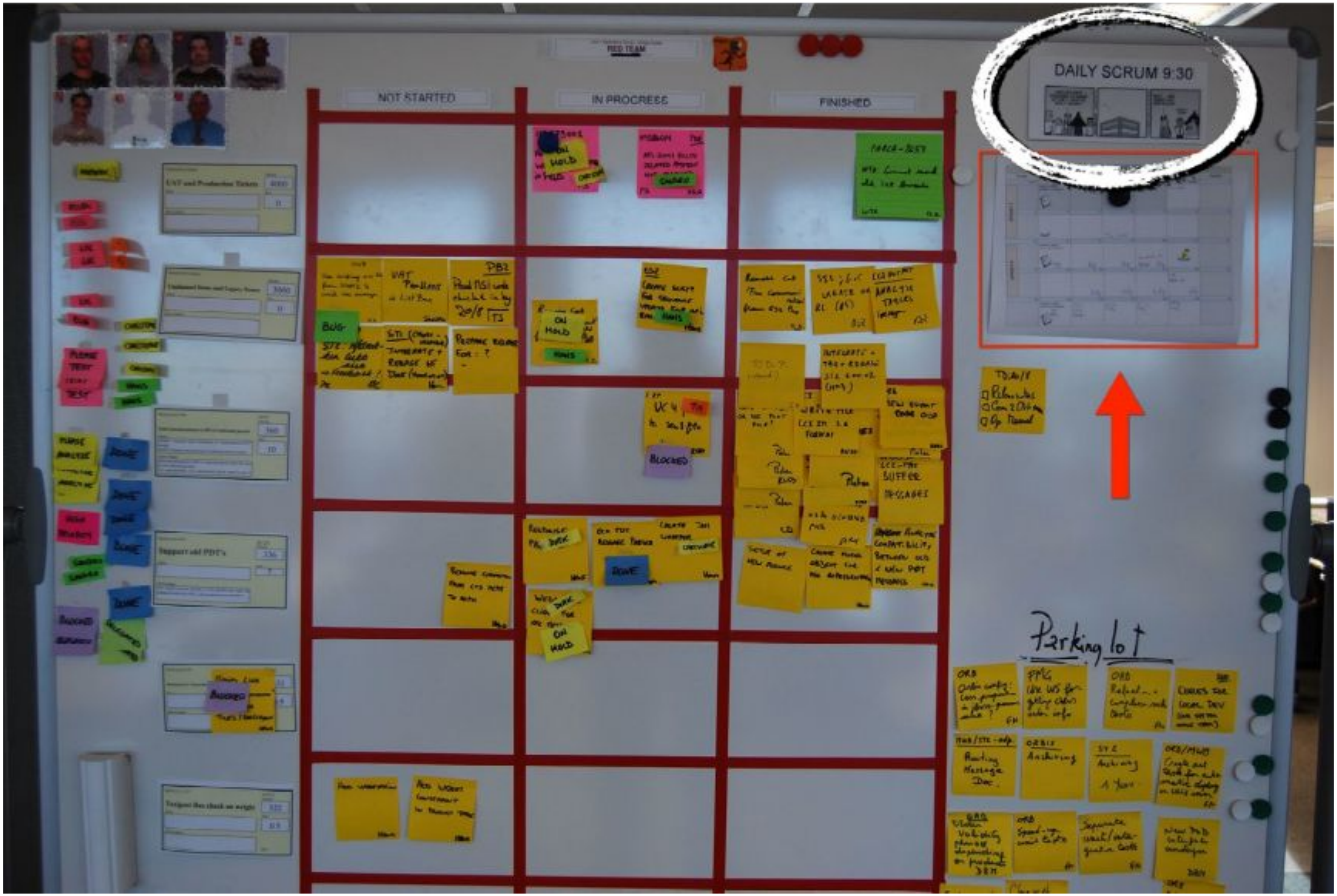
# Scrum Task Board



# Development Team

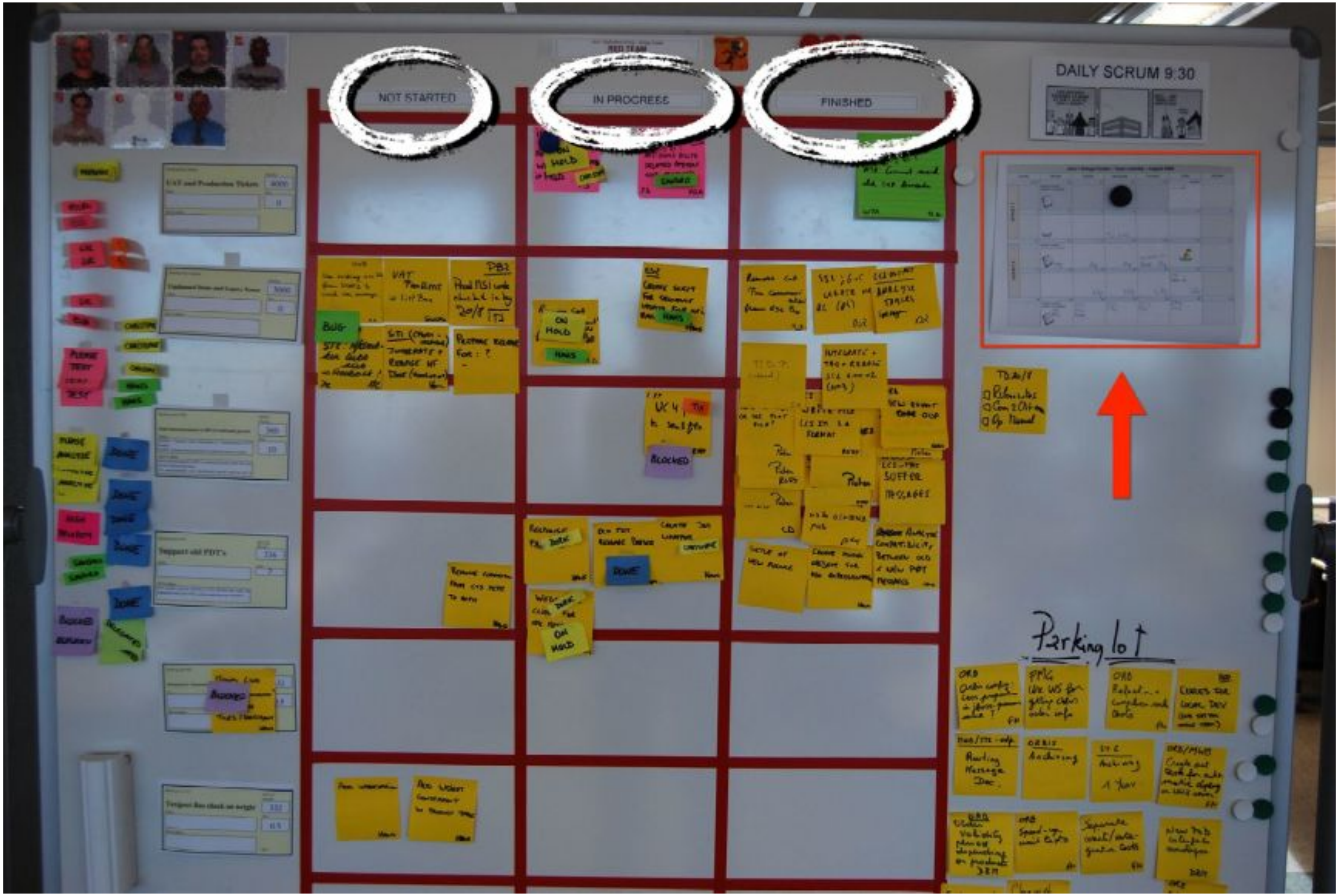


# Daily Scrum (Standup Meeting)



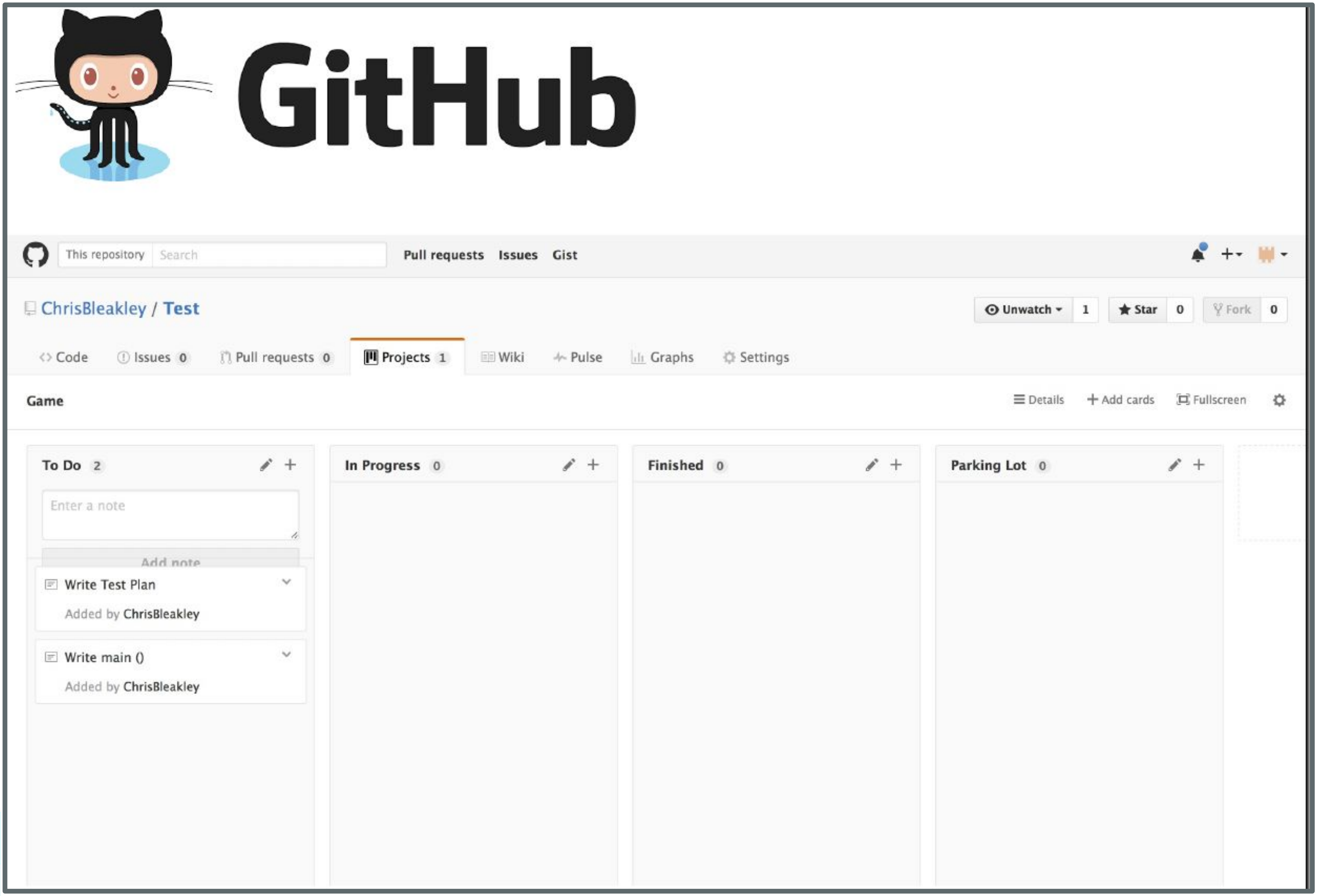
# Daily Scrum (Standup Meeting)

# Sprint Progress



# Project Tracking and Communication Tools

# GitHub (Feature Tracking)



ChrisBleakley / Test

Game

To Do 2

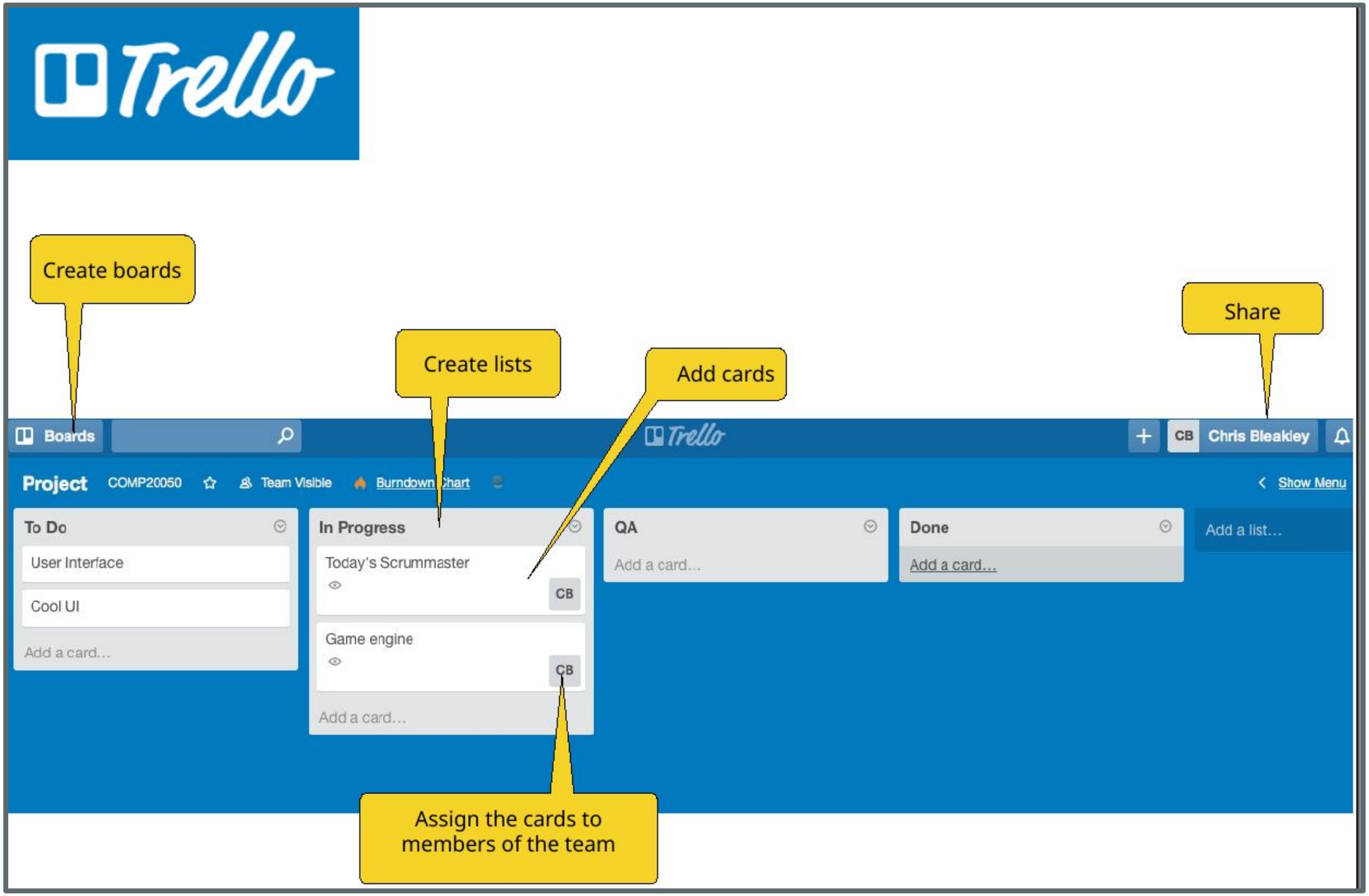
- Write Test Plan
  Added by ChrisBleakley
- Write main ()
  Added by ChrisBleakley

In Progress 0

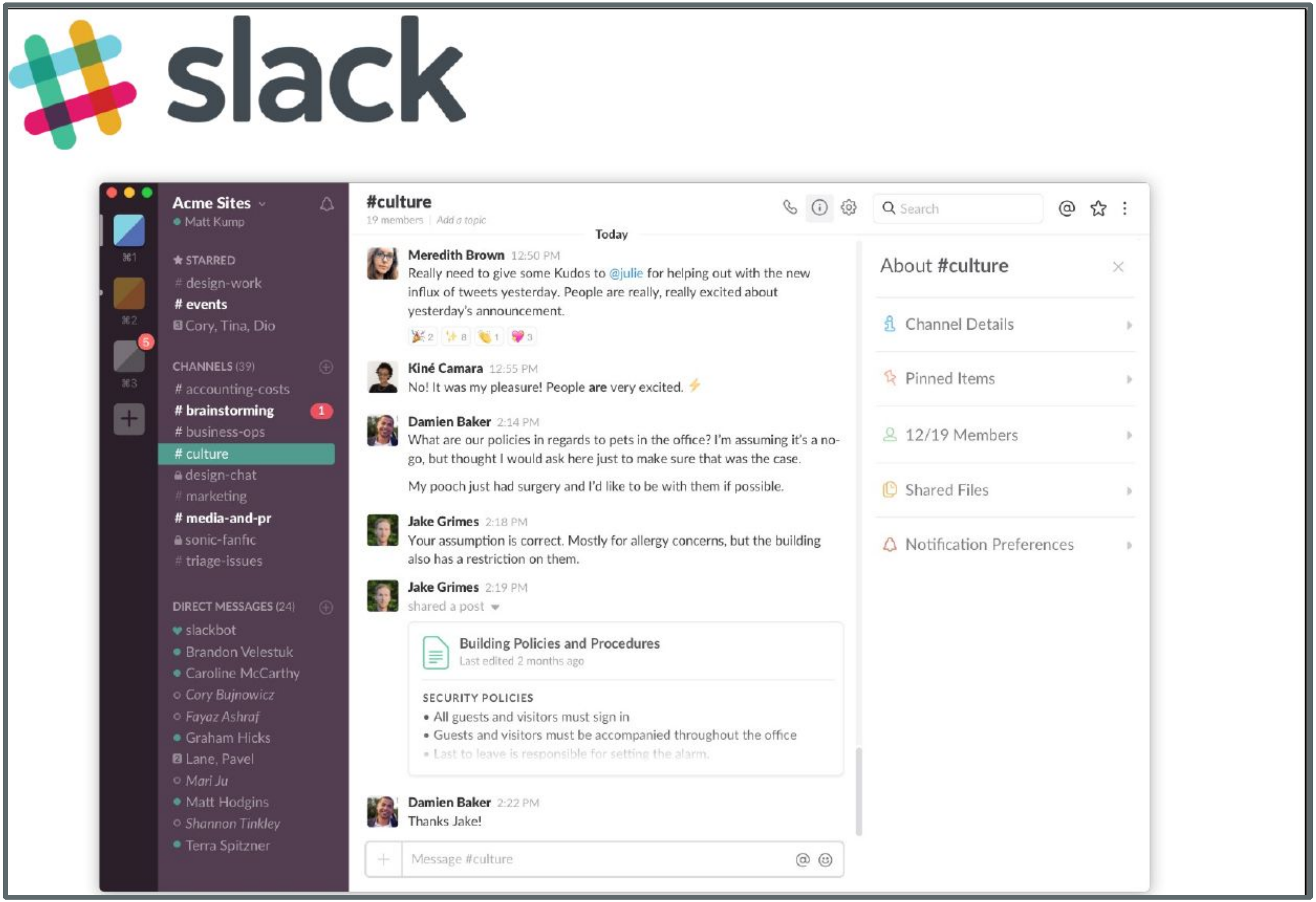
Finished 0

Parking Lot 0

# Trello Kanban Board (Feature Tracking)



# Slack for Team Communication



Today

**Meredith Brown** 12:50 PM
Really need to give some Kudos to @julie for helping out with the new influx of tweets yesterday. People are really, really excited about yesterday's announcement.

**Kiné Camara** 12:55 PM
No! It was my pleasure! People are very excited.

**Damien Baker** 2:14 PM
What are our policies in regards to pets in the office? I'm assuming it's a no-go, but thought I would ask here just to make sure that was the case.

My pooch just had surgery and I'd like to be with them if possible.

**Jake Grimes** 2:18 PM
Your assumption is correct. Mostly for allergy concerns, but the building also has a restriction on them.

**Jake Grimes** 2:19 PM
shared a post

Building Policies and Procedures
Last edited 2 months ago

SECURITY POLICIES
- All guests and visitors must sign in
- Guests and visitors must be accompanied throughout the office
- Last to leave is responsible for setting the alarm.

**Damien Baker** 2:22 PM
Thanks Jake!

# Scrum and Lab Sessions

# Lab Sessions

- Before the start of **SprintN** (**N** = 1, 2, 3, 4),
  - Hold **sprint planning** meeting.
  - Break down each targeted feature into a set of tasks (**sprint backlog**).

- At the end of **SprintN** (**N** = 1, 2, 3, 4),
  - Hold **sprint review** meeting.
  - Hold **sprint retrospective**.

# Sprint Execution

- Work on your assigned items in the Sprint backlog.
- Hold code reviews.
- Write tests.
- Update your task board periodically (needed for Brightspace submission).
- Build the potentially shippable increment.
- Submit to Brightspace.

# Q&A

# To follow…

Introduction to JavaFX