# Software Engineering

dr. Asta Slotkienė
asta.slotkiene@vgtu.lt

# Why iterative and incremental process?

- Software is part of almost all business operations, so new software has to be **developed quickly** to take advantage of new opportunities and to respond to competitive

- Rapid software development and delivery is the **most critical requirement for most business systems**

# Iterative and incremental process

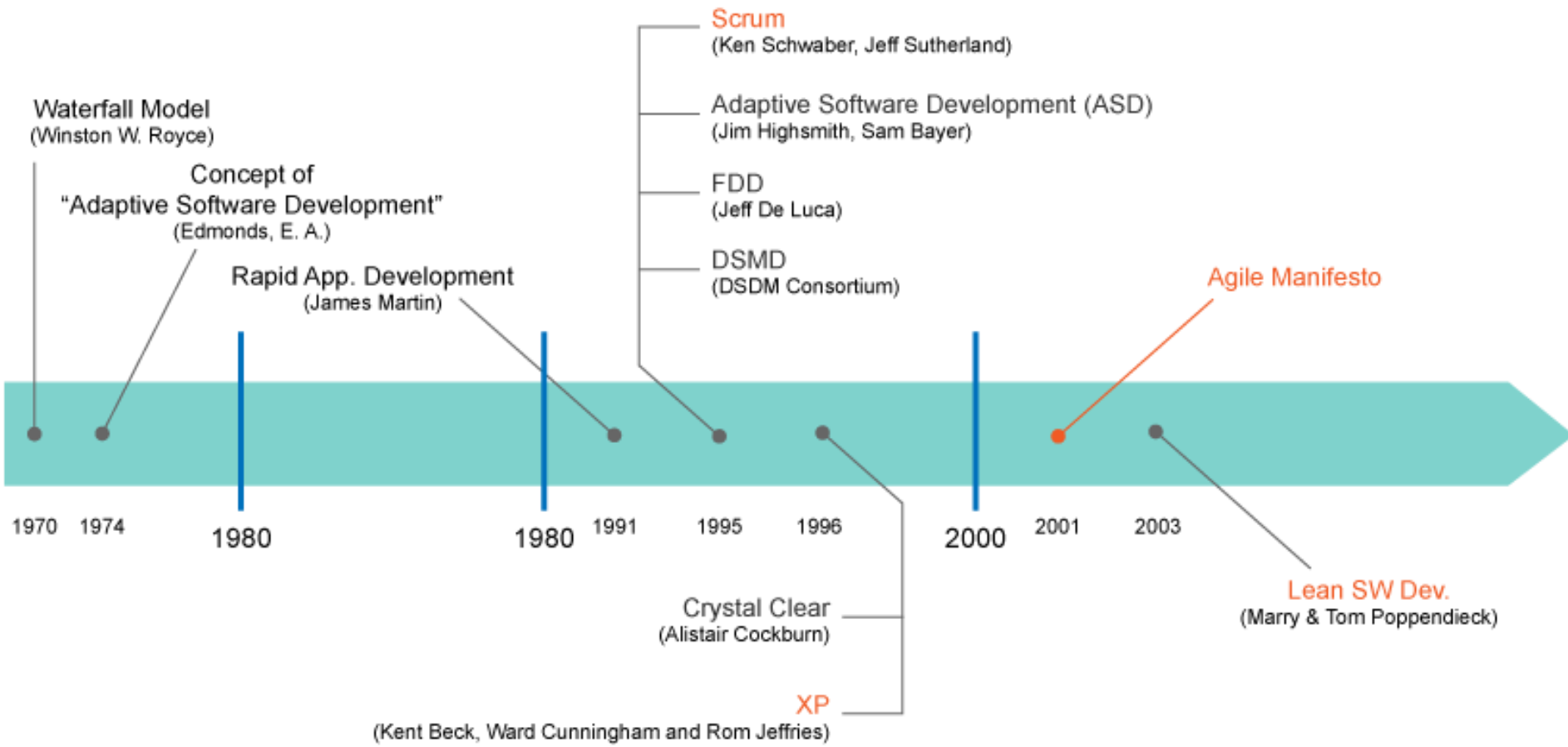
- Iterative: we develop software through repeated iterations (cycles).

- Incremental: we develop software in small portions at a time.

# Software development vs. businesses

- Businesses are operating in a changing environment
- Requirements change because customers find it impossible to predict:
  1. how a system will affect working practices?
  2. how it will interact with other systems?
  3. what user operations should be automated?

# History of Agile




https://www.agilealliance.org/agile101/practices-timeline/

# History of rapid software development

- 1990s with the development of the idea of "agile methods"
  - Extreme Programming (Beck 1999),
  - Scrum (Schwaber and Beedle 2001),
  - DSDM (Stapleton 2003).

Rapid software development became known as **agile development** or **agile methods.**

https://www.agilealliance.org/agile101/practices-timeline/

# Agile manifesto

*We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*

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

*That is, while there is value in the items on the right, we value the items on the left more.*

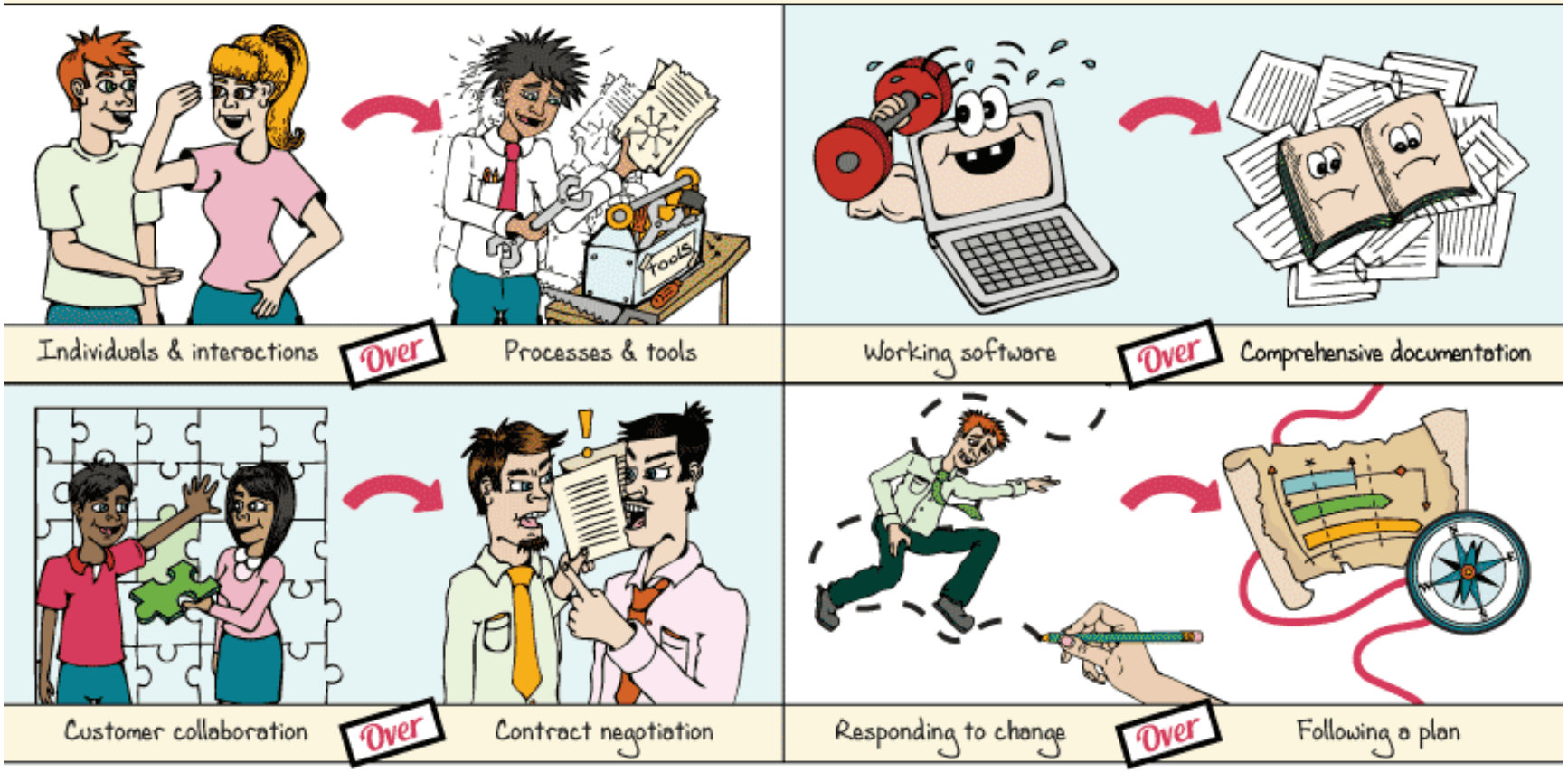
http://agilemanifesto.org/

# Agile manifesto


http://agilemanifesto.org/

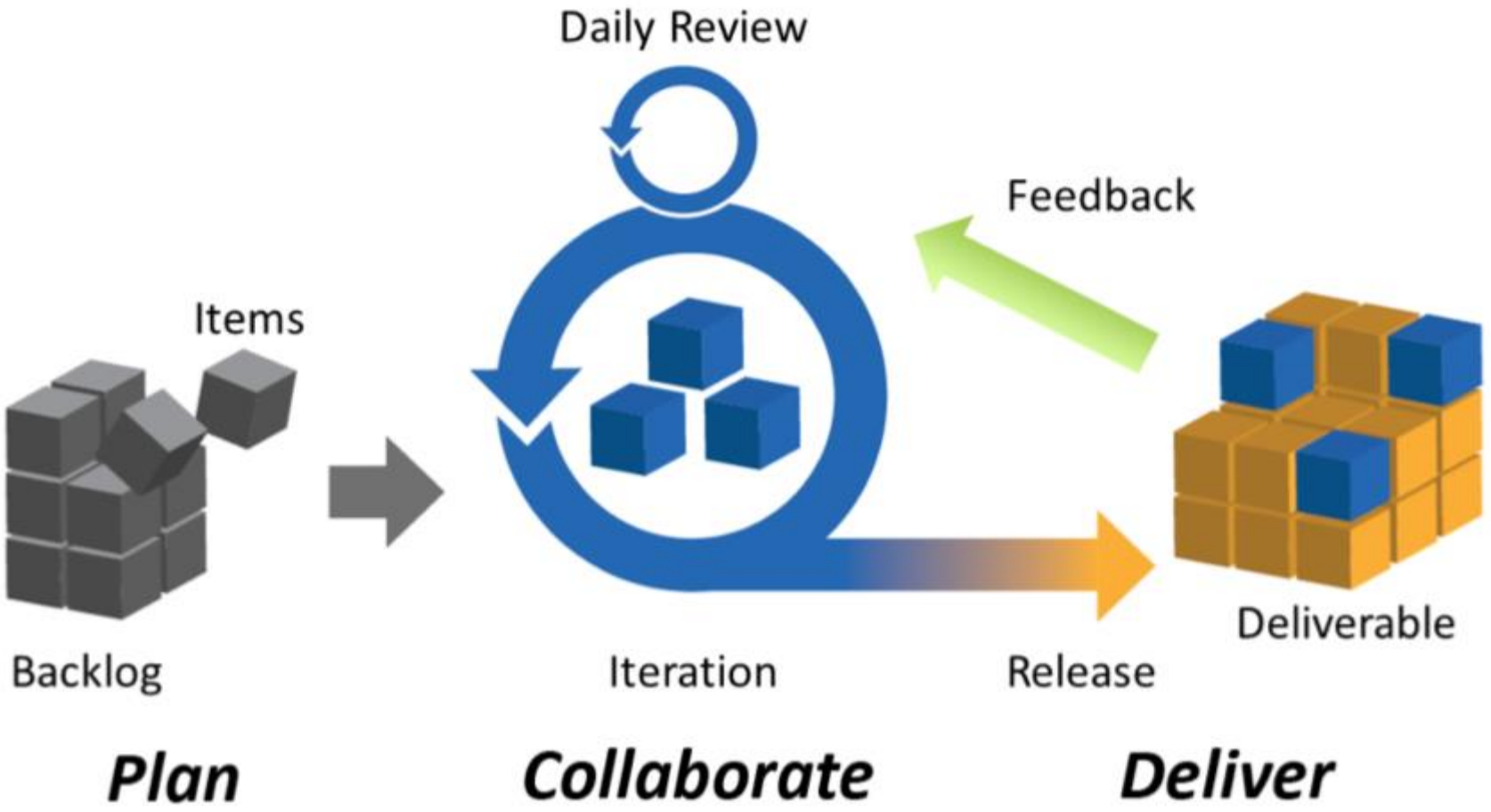
# Manifesto for Agile Software Development*

"We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:




That is, while there is value in the items on the right, we value the items on the left more."

# Agile methodology

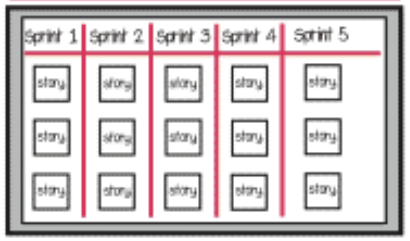


# The 12 agile principles*

1 Satisfy the **customer**


Welcome **change**

Deliver **frequently**



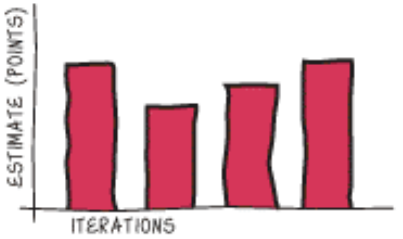

4 Work **together**

5 Trust and **support**

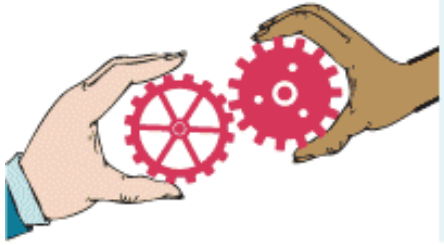
Face-to-face **conversation**

Working **software**

8 Sustainable **development**




9 Continuous **attention**



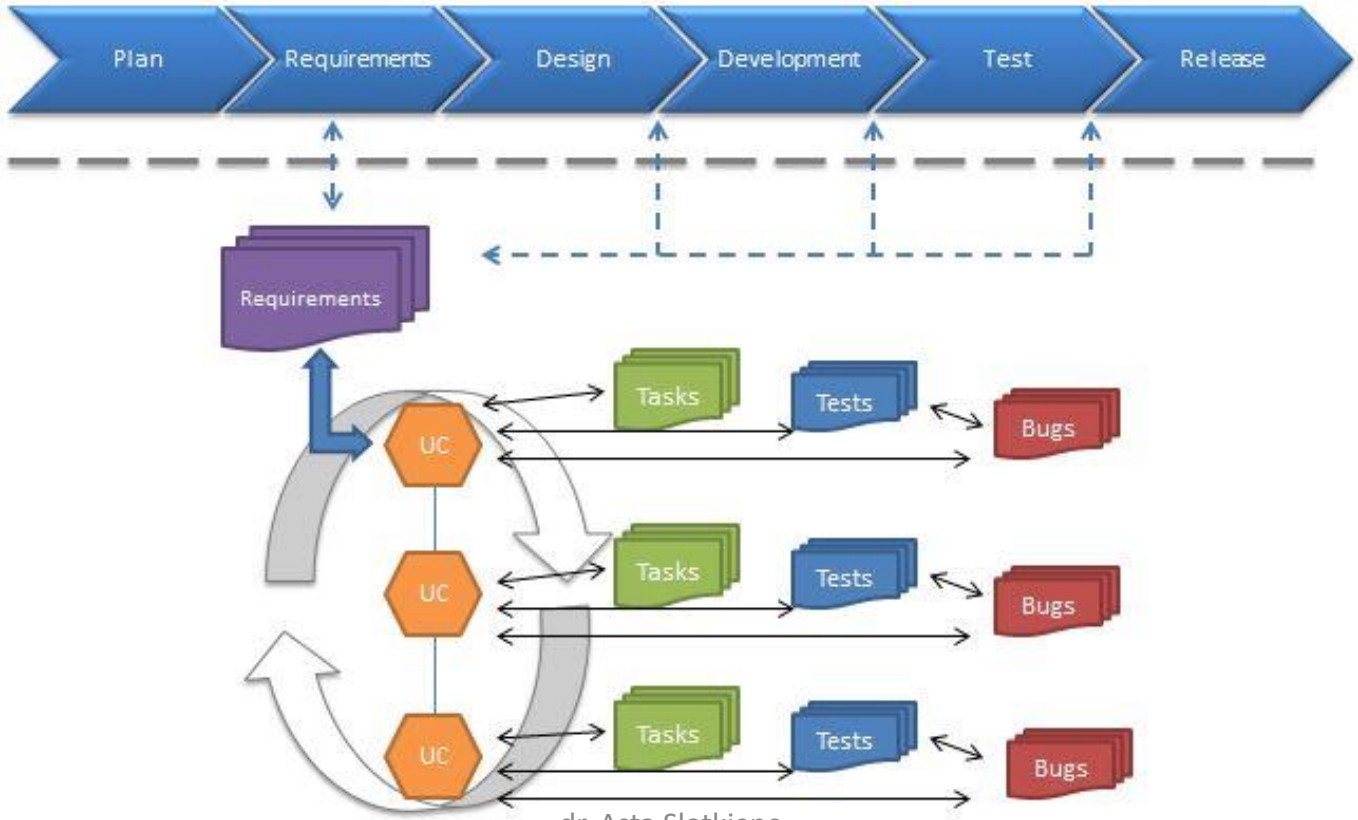
10 Maintain **simplicity**


11 Self-organizing **teams**

12 Reflect and **adjust**

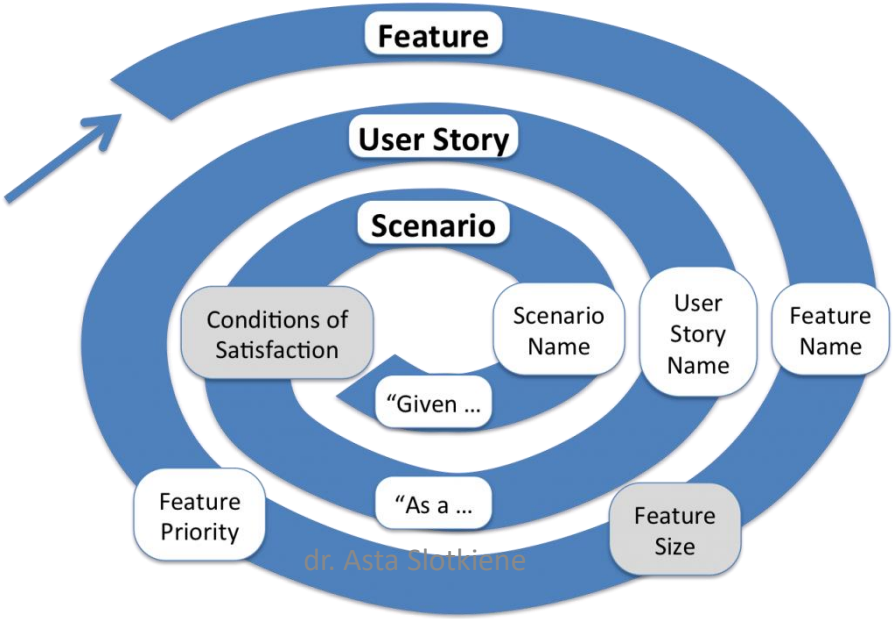
# Characteristics of Agile 1

- The processes of specification, design and implementation are interleaved.



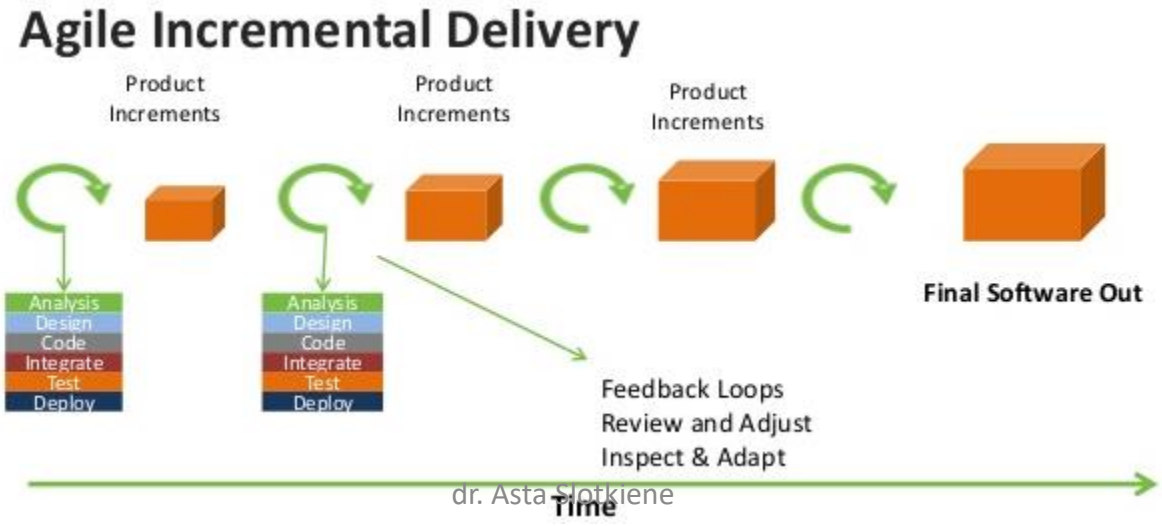
# Characteristics of Agile 2

- There is **no detailed system specification**, and **design documentation** is minimized or generated automatically by the programming environment used to implement the system.
- The user requirements document is an outline definition of the most important characteristics of the system.



# Characteristics of Agile 3

- The system is developed in a series of increments.
- End-users and other system stakeholders are involved in specifying and evaluating each increment.

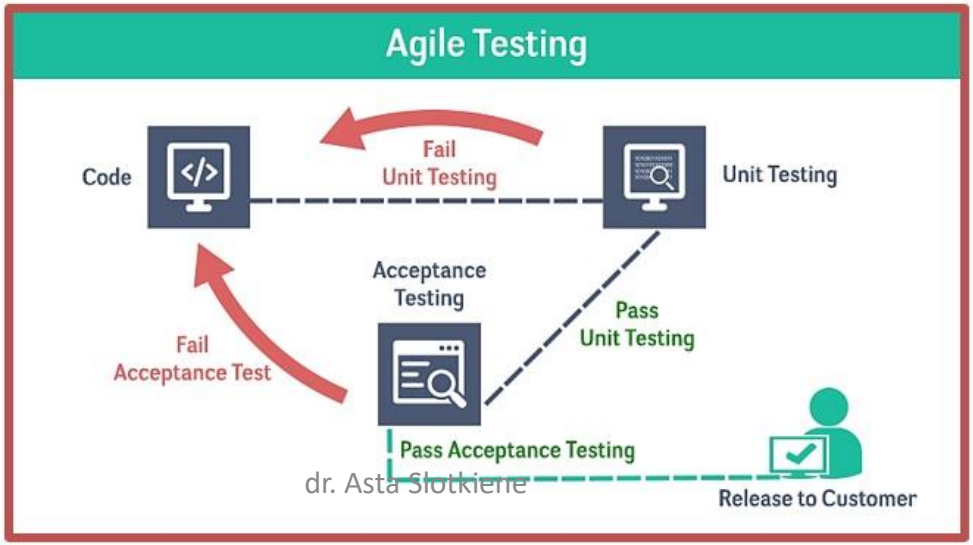


# Characteristics of Agile 4

- They may propose changes to the software and new requirements that should be implemented in a later version of the system.

# Characteristics of Agile 5

- Extensive tool support is used to support the development process.
- Tools that may be used include automated testing tools, tools to support configuration management, and system integration and tools to automate user interface production



# Characteristics of Agile 6

- They involve customers in the development process to get rapid feedback on changing requirements.
- They minimize documentation by using informal communications rather than formal meetings with written documents

# Characteristics of Agile 6

- Based on iterative and incremental development
- The increments are small, and, typically, new releases of the system are created and made available to  customers every two or three weeks.

# Agile project management statistics

- Almost 71% of organizations report using Agile approaches sometimes, often, or always. (source)
- Agile projects are 28% more successful than traditional projects. (source)
- By 2030, artificial intelligence will automate 80% of routine Agile work (source)
- 76% of users choosing an enterprise Agile planning tool do so to increase their projects' visibility. (source)

# WaterFall vs Agile

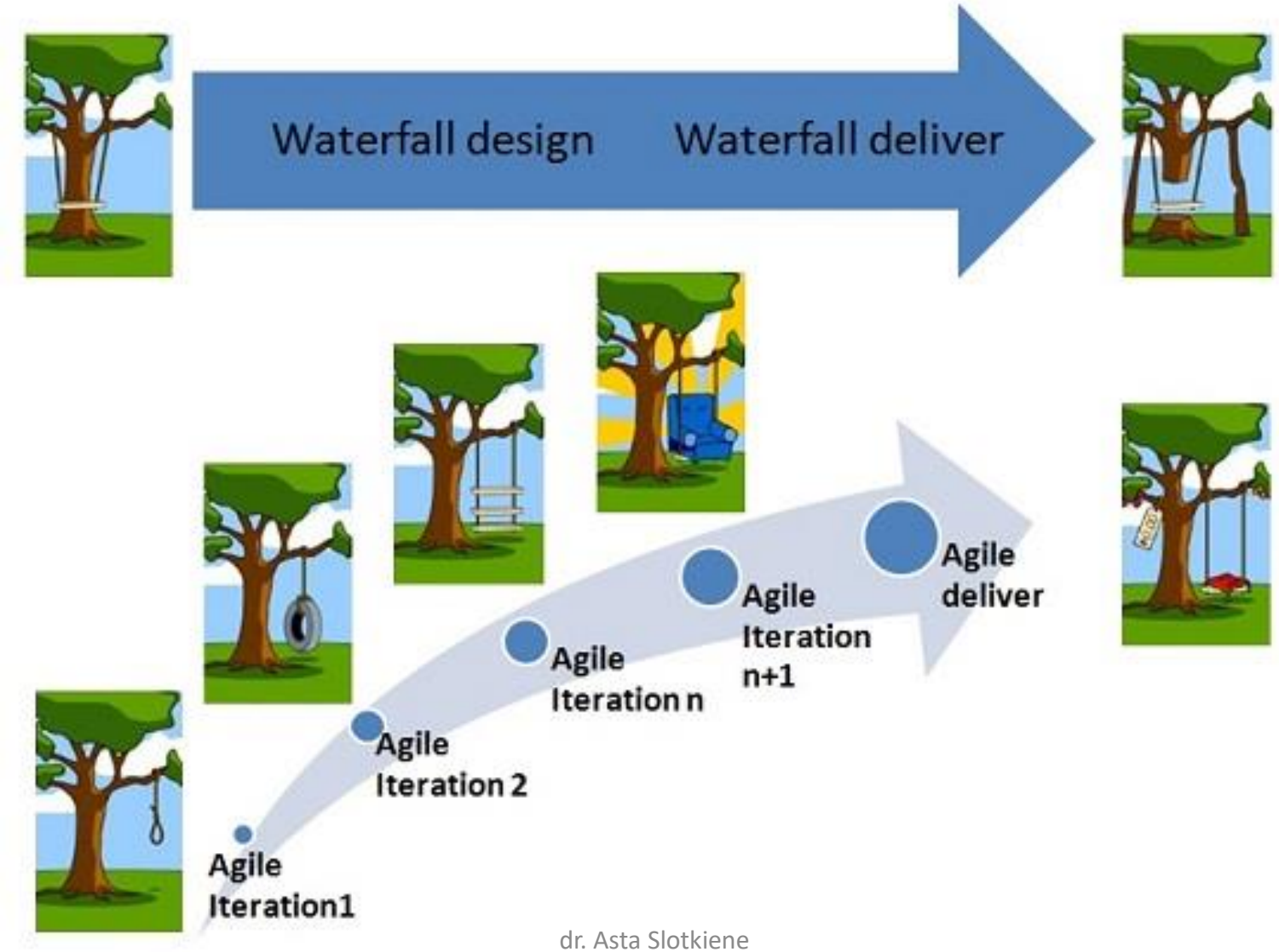
## Traditional

- Client **knows what he wants**
- Creators **know how to create**
- **Nothing** will change

## Agile

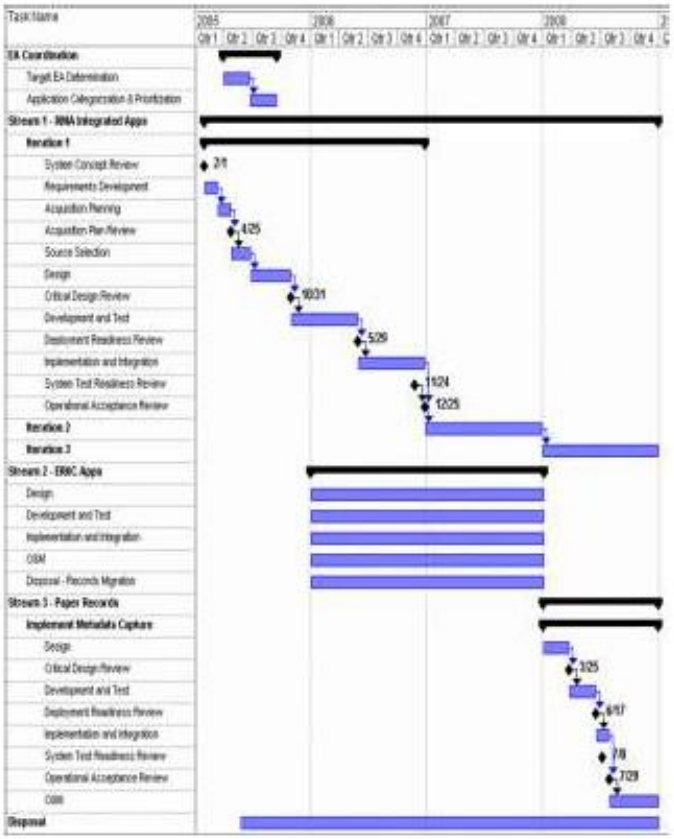
- Client will **figure out** what he wants
- Creators **will find out** how to create
- There **will be changes**

# WaterFall vs Agile



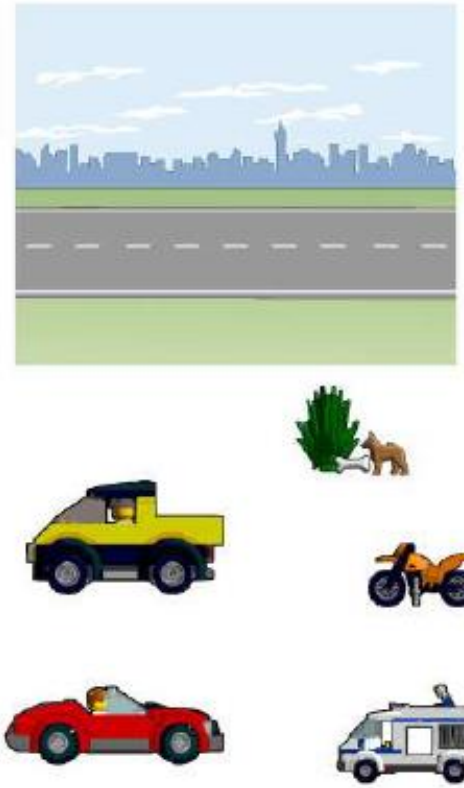
# Traditional (waterfall) project

## Plan




## Requirements

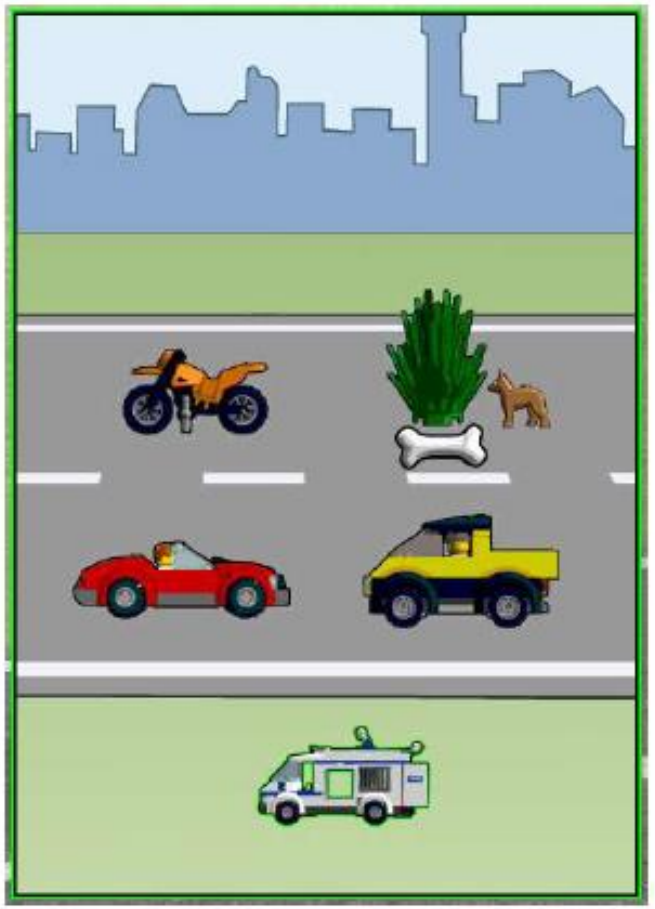
## Development



https://www.slideshare.net/AgileLietuva/iteracinio-inkrementinioagilemetodonaudosirrizikosuzsakovuivaidasadomauskas

# Traditional (waterfall) project

## Integration and testing



dr. Asta Slotkiene

https://www.slideshare.net/AgileLietuva/iteracinio-inkrementinioagilemetodonaudosirrizikosuzsakovuivaidasadomauskas

# Agile is process to create products or services

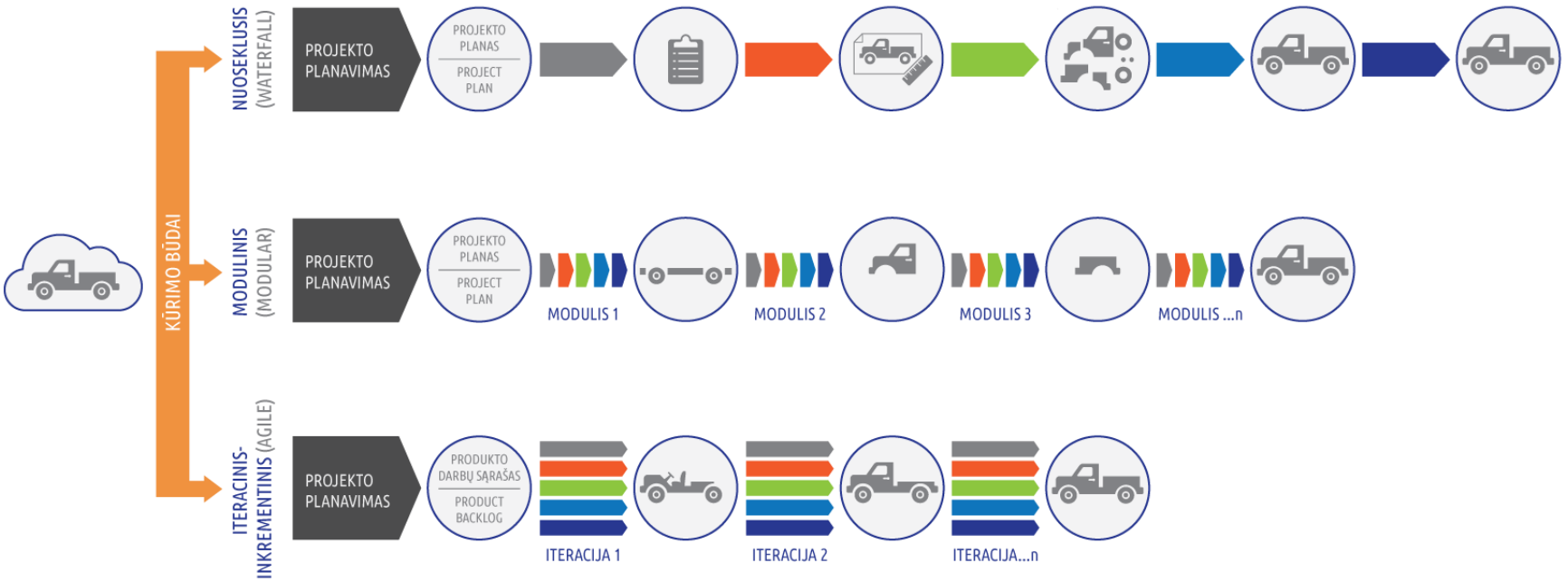
# in increments!



https://www.slideshare.net/AgileLietuva/iteracinio-inkrementinioagilemetodonaudosirrizikosuzsakovuivaidasadomauskas

# WaterFall vs Agile

## Palyginimas pagal kūrimo etapus




**Žymėjimai:**

- Analizė
- Projektavimas
- Konstravimas
- Testavimas
- Diegimas
- Rezultatas
- Pradinis tikslas

Agile LIETUVA



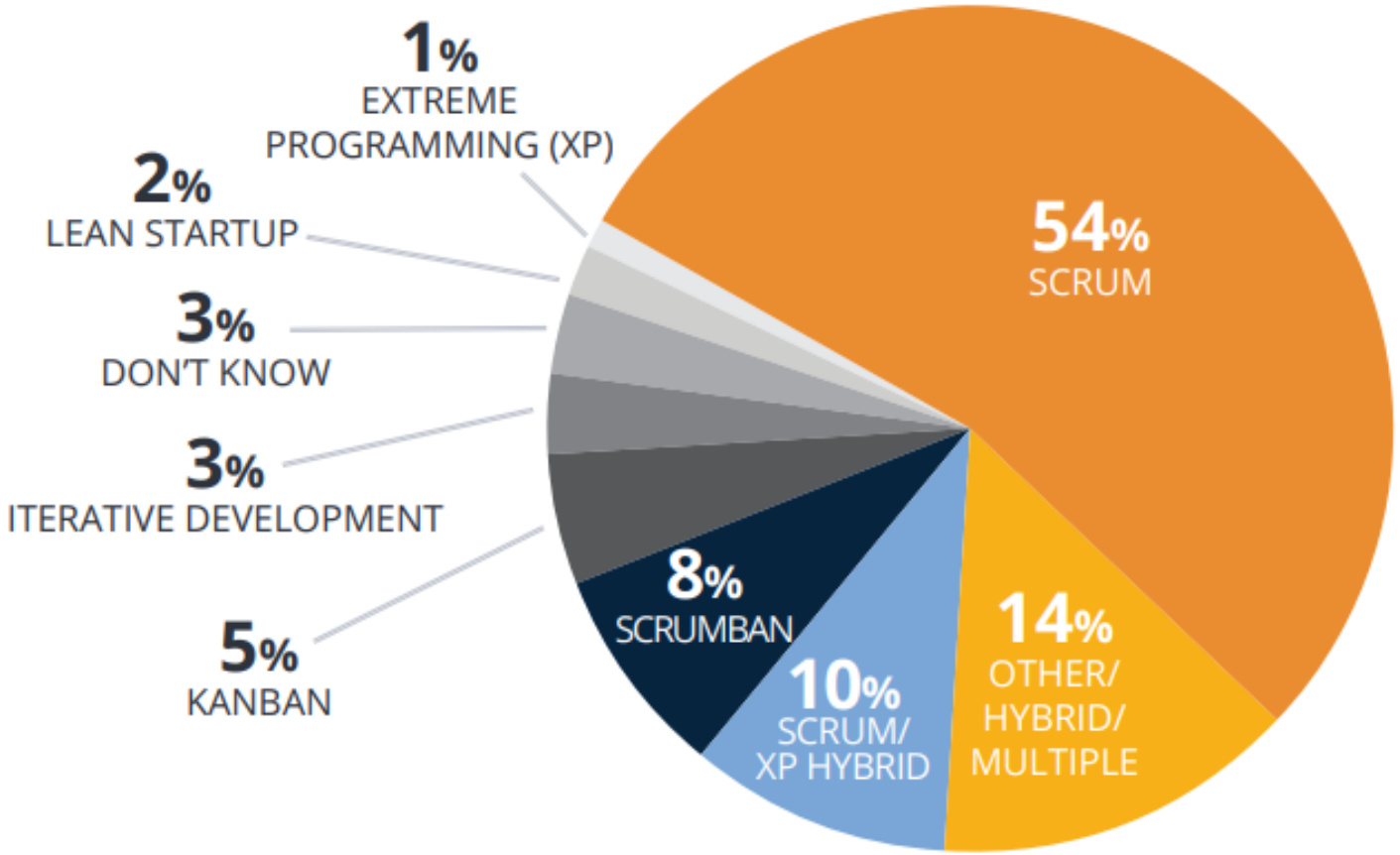
https://www.slideshare.net/AgileLietuva/iteracinio-inkrementinioagilemetodonaudosirrizikosuzsakovuivaidasadomauskas

# Iterative software development

- Iterative development is an approach to building software (or anything) in which the overall lifecycle is composed of several iterations in sequence.

- Each iteration is a self-contained mini-project composed of activities such as requirements analysis, design, programming, and test.

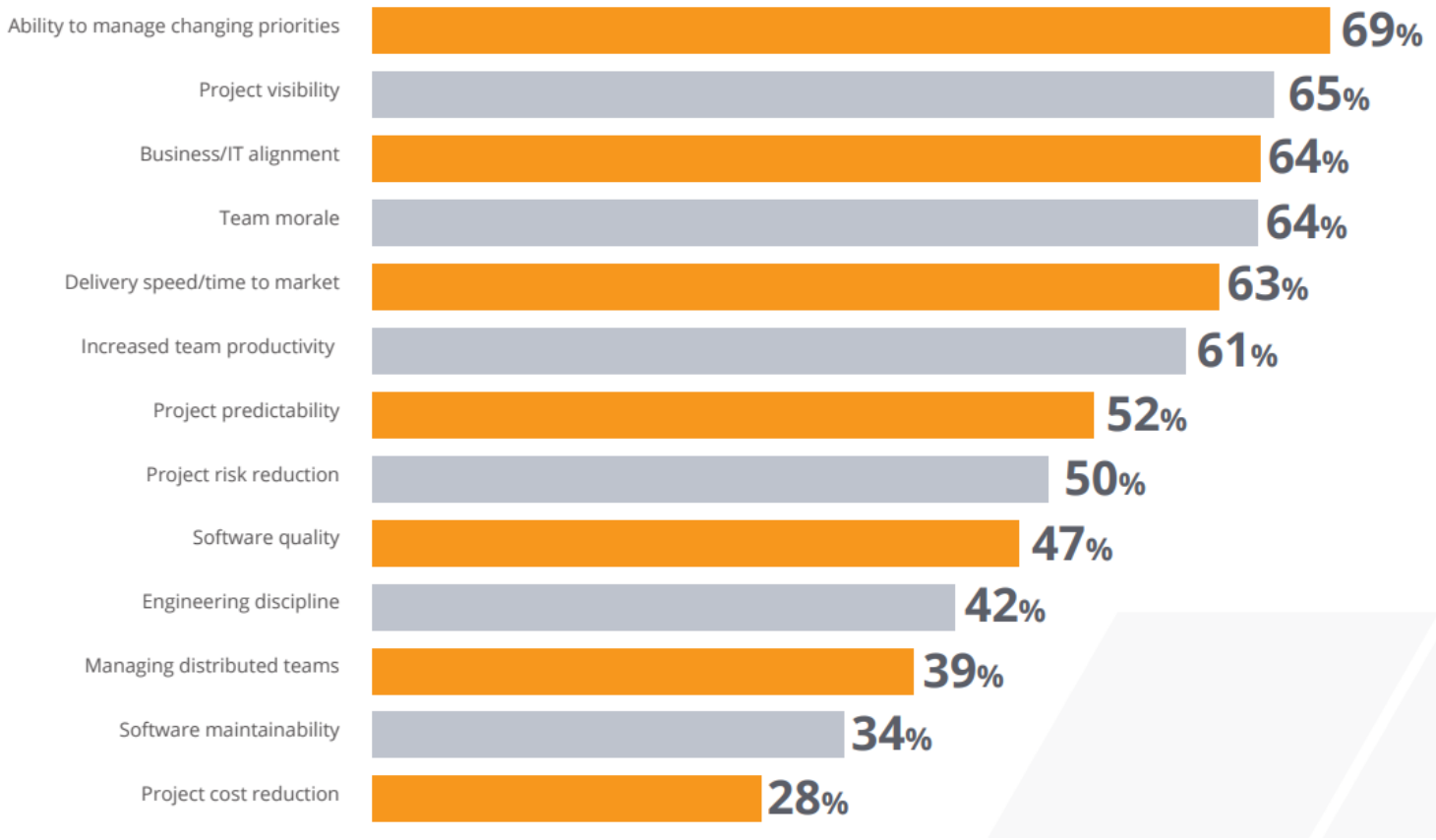
# Agile Methodologies Used






13th-annual-state-of-agile-report.pdf

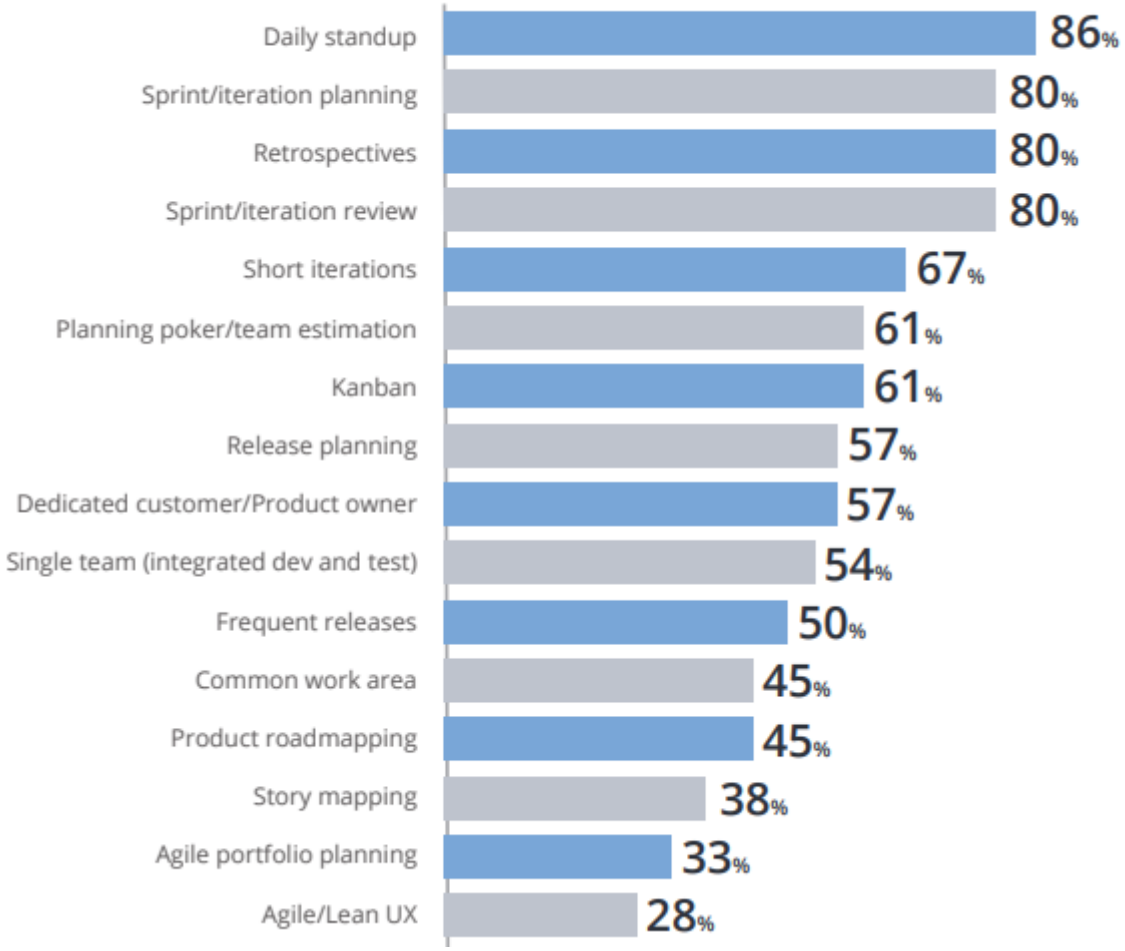
# Benefits of Adopting Agile






13th-annual-state-of-agile-report.pdf

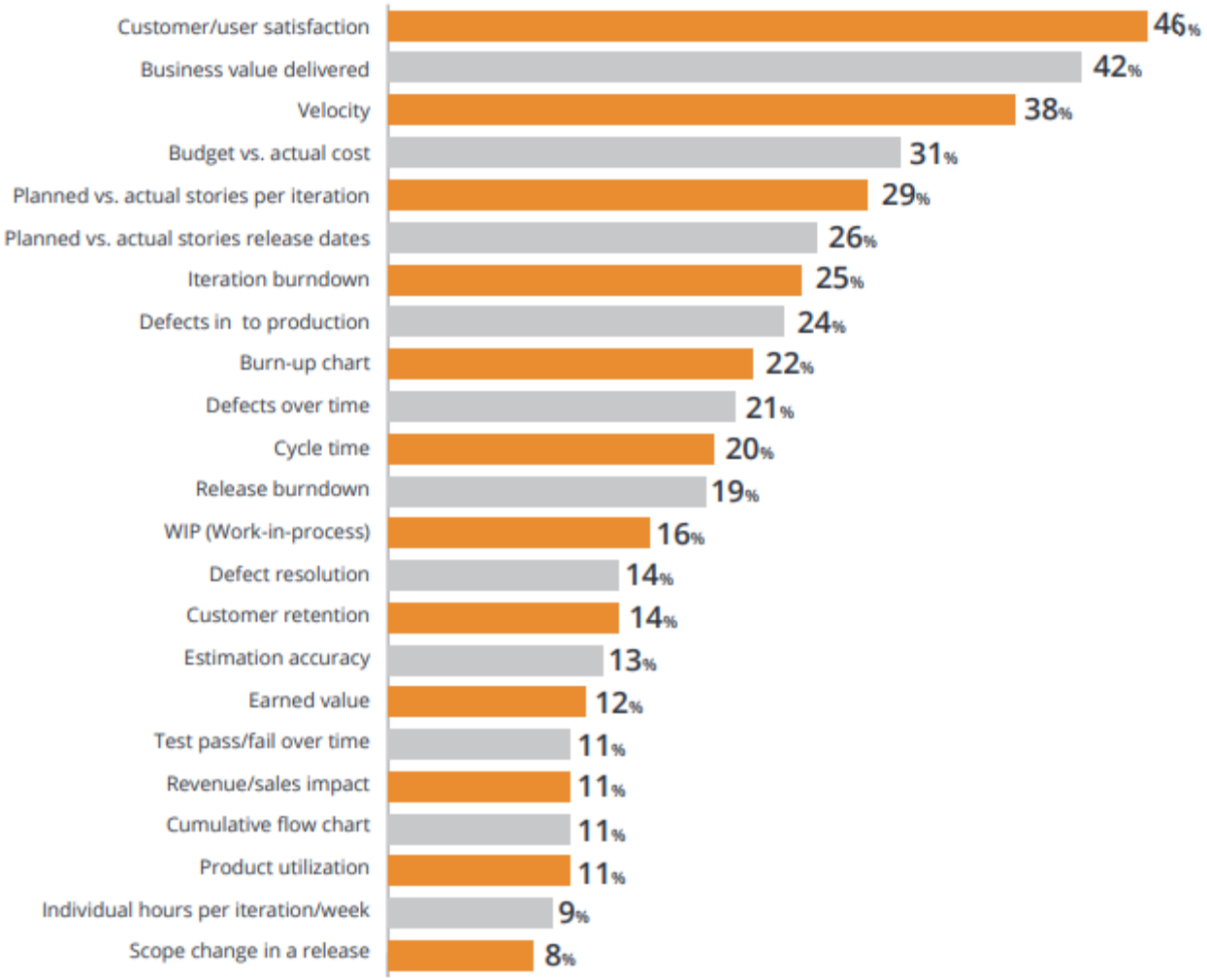
# Agile Techniques Employed






13th-annual-state-of-agile-report.pdf

# How succes is measured with agile






13th-annual-state-of-agile-report.pdf

# When to apply Agile methods

1. Product development where a software company is developing a small or medium-sized product for sale.
   - Virtually all software products and apps are now developed using an agile approach.

# When to apply Agile methods

2. Custom system development within an organization, where there is a clear commitment from the customer to become involved in the development process and where there are few external stakeholders and regulations that affect the software.

# When recomented to apply Agile methods

- Agile methods may not be suitable for other types of software development:
  - embedded systems engineering
  - the development of large and complex systems

# Problems with agile methods

1. The informality of agile development is incompatible with the legal approach to contract definition that is commonly used in large companies.

2. Agile methods are most appropriate for new software development rather than for software maintenance. Yet the majority of software costs in large companies come from maintaining their existing software systems.

3. Agile methods are designed for small co-located teams, yet much software development now involves worldwide distributed teams

# What is Agile Software Development?

- Agile software development is more than frameworks such as Scrum, Extreme Programming or Feature-Driven Development (FDD).
- Agile software development is more than practices such as pair programming, test-driven development, stand-ups, planning sessions and sprints.


Scrum
Lean software development
Kanban (process + method)
Extreme Programming (**XP**)
Continuous Integration (**CI**)
Continuous Delivery (**CD**)
Feature Driven development (**FDD**)
Test Driven Development (**TDD**)
Crystal Clear
...

Lightweight approaches

Scrum-of-Scrums
Scrum at Scale (**Scrum@Scale**)
Large-scale Scrum (**LeSS**)
Scaled Agile Framework (**SAFe**)
Disciplined Agile Delivery (**DAD**)
Dynamic Systems Development Method (**DSDM**)
Agile Project Management (**AgilePM**)
Agile Unified Process (**AUP**)
Open Unified Process (**OpenUP**)
...

Fuller approaches (beyond 1 team)

# Scrum

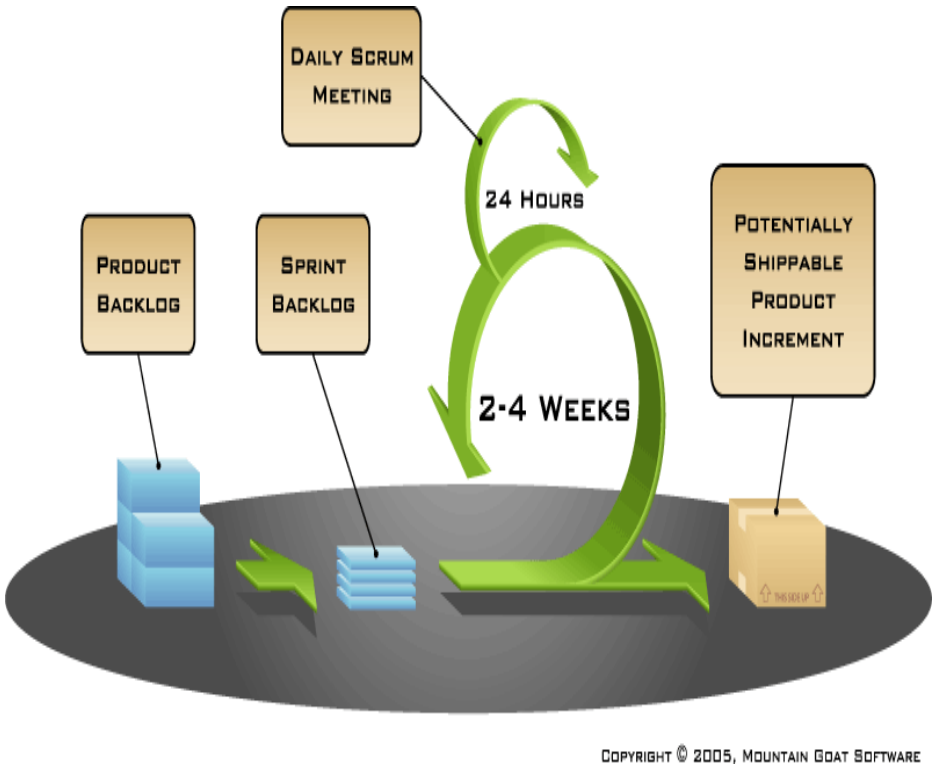
Scrum is a framework for developing and sustaining complex products.

- Moreover, Scrum is:
  - An empirical process
  - Lightweight
  - Simple to understand
  - Difficult to master

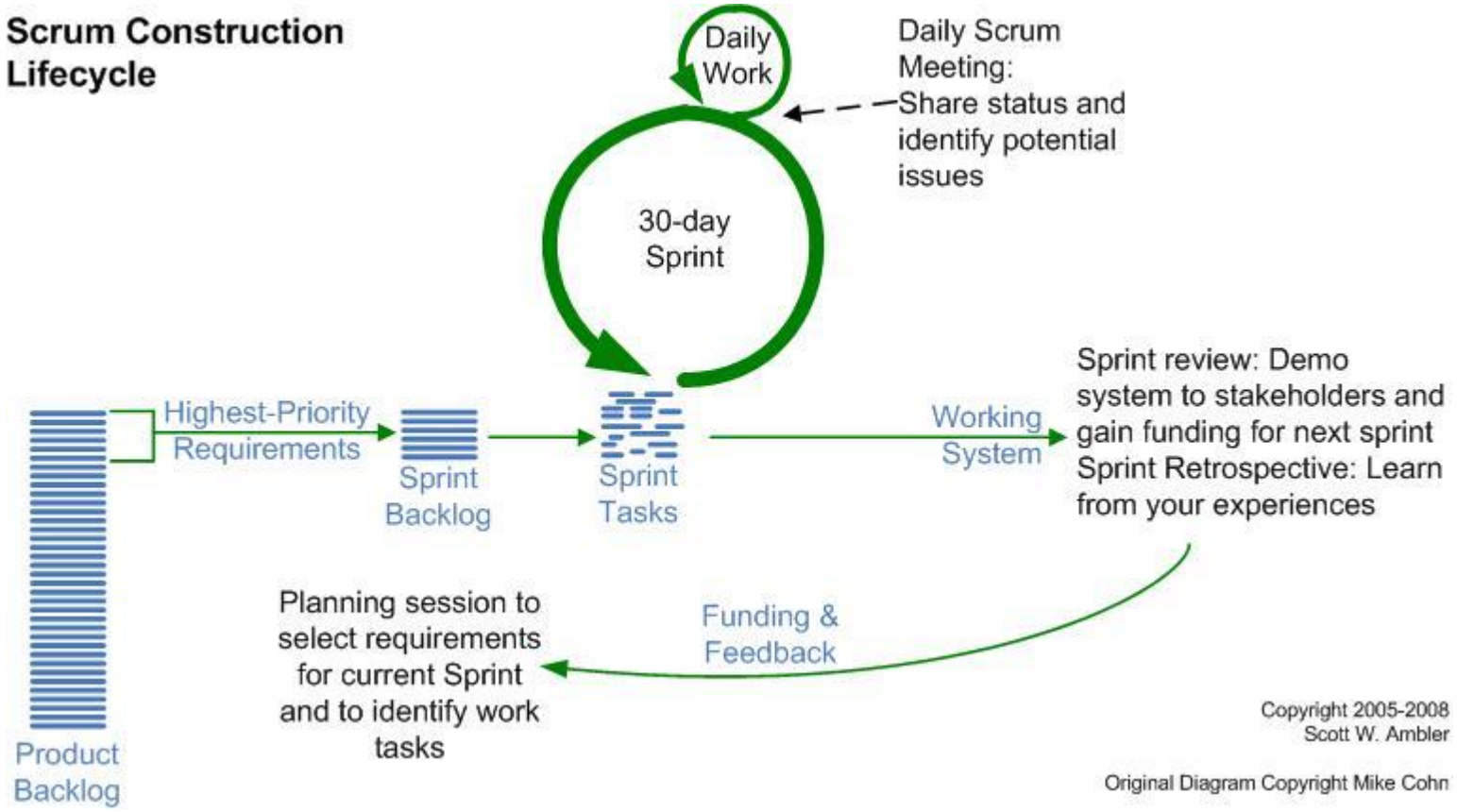
https://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum- Guide-US.pdf

# Scrum

- Scrum is an iterative and incremental agile software development framework
- A flexible, holistic product development strategy
- Development team works as an atomic unit and is committed to a Sprint Goal
- Fixed Iterations



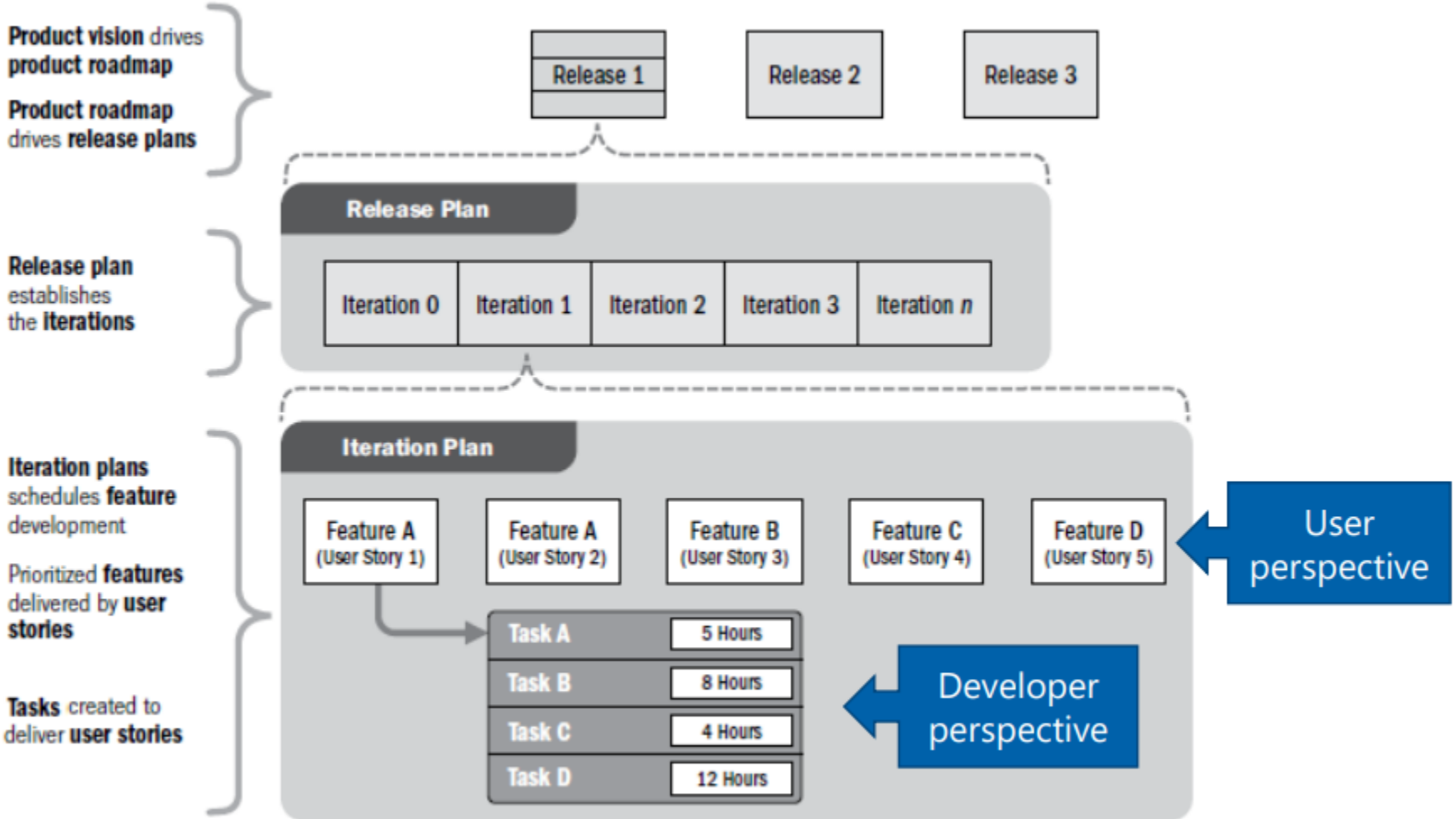
# Scrum Life Cycle



# Release planning

- A release is made up **of one or more iterations**
- Release planning refers to determining a balance between a **projected timeline and a desired set of functionality**

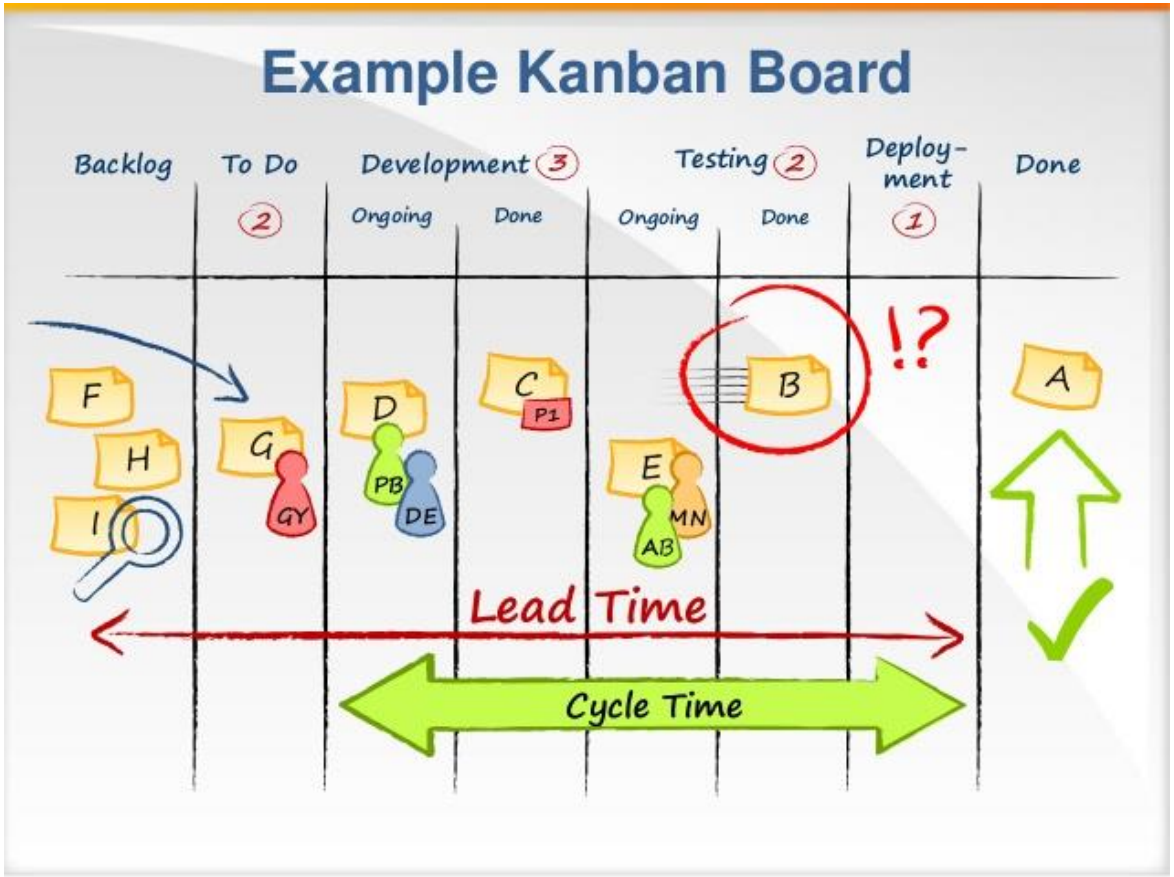
# Release planning



# Kanban

- **Kanban** is an approach for *changes management* and *process improvements* of a supply chain.

# Kanban




- Limit Work-in-Progress
- Visualize and Manage Workflow
- Measure Cycle Time

# Kanban principles

- Kanban is based on a continuous workflow structure that keeps teams nimble and ready to adapt to changing priorities

- Common workflow stages are **To Do, In Progress, In Review, Blocked, and Done.**

- In Kanban, updates are released whenever they are ready, without a regular schedule or predetermined due dates:
  - if the task gets completed earlier (or later), it can be released as needed without having to wait for a release milestone like sprint review.

# Scrum vs. Kanban

- Iteration: regular fixed length sprints (2 weeks)
- Release: At the end of each sprint
- Metric: Velocity

- No iteration: Continuous flow
- Release: Continuous delivery
- Metric: Lead time, cycle time, WIP

https://www.slideshare.net/pawelbrodzinski/kanban-basics-5834758

# Scrum vs. Kanban

- Change management
  - Teams should not make changes during the sprint.
  - During the sprint retrospective, **scrum teams should discuss how to limit change in future**, as changes put the potentially shippable increment at risk.

- Change management:
  - Change can happen at any time

https://www.slideshare.net/pawelbrodzinski/kanban-basics-5834758

# Lean Software Development

- Originates from Toyota Production System (TPS)
  - Also called Just-In-Time system
- Inspiration for TPS found in the 1950's from U.S. supermarkets
  - **Customers could get what they wanted, when they wanted it and shelves were refilled when items were about to run out.**
- The concepts transferred to the domain of software engineering by Mary and Tom Poppendieck (2003, 2007).

# Lean Principles

1. *Eliminate waste*
2. *Amplify learning*
3. *Decide as late as possible*
4. *Deliver as fast as possible*
5. *Empower the team*
6. *Build integrity in*
7. *See the whole*

1. Pašalinkite atliekas
2. Skatinkite/stiprinkite mokymąsi
3. Nuspręskite kaip galima vėliau
4. Pristatykite produktą kaip galima greičiau
5. Įgalinkite/motyvuokite komandą
6. Integruokite
7. Įvertinkite visumą

# Lean Software Development

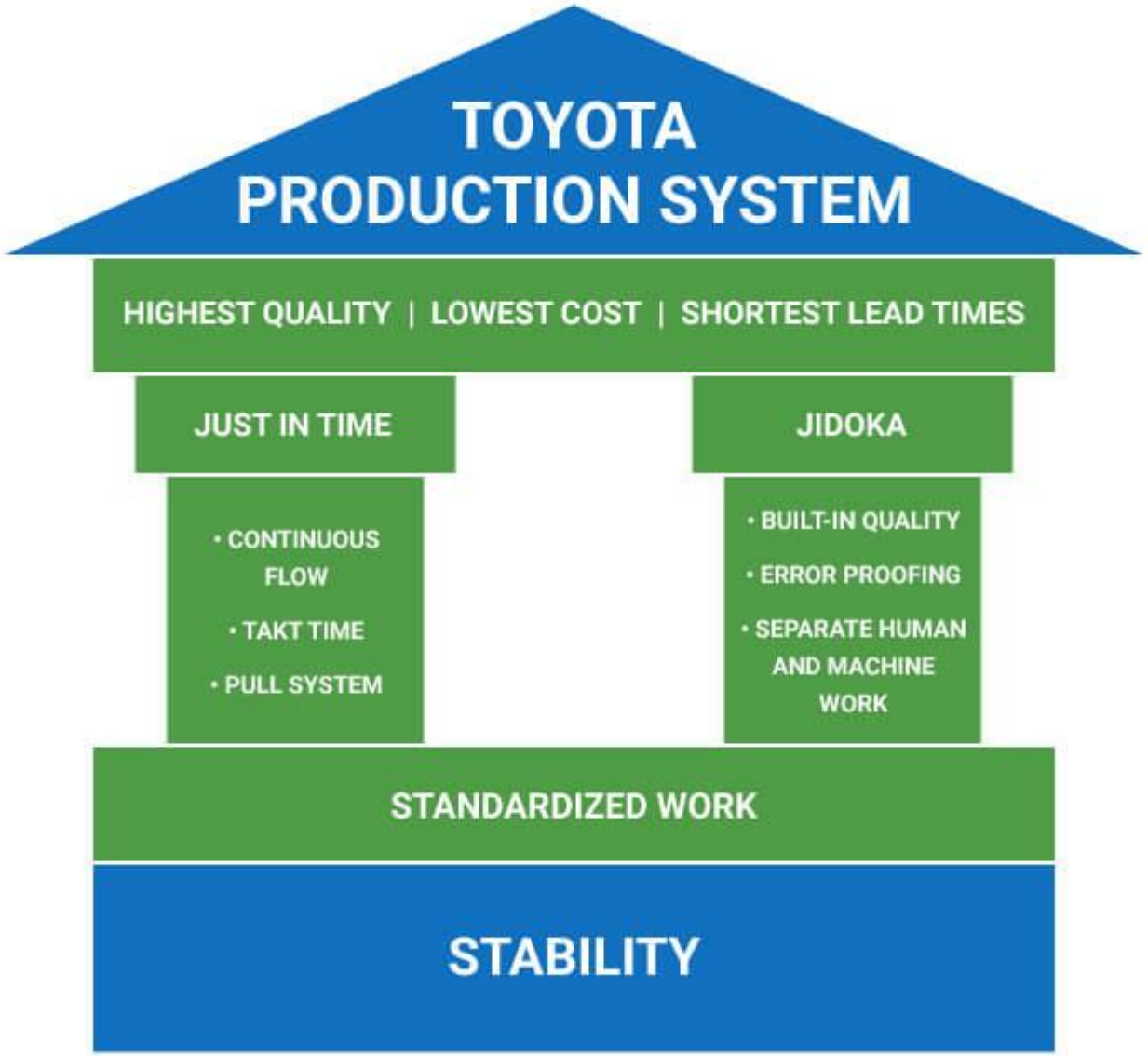
- All processes shall give **value**
- Ensure good flow in the processes to avoid **bottlenecks and queues**
- All activity shall be based on need
- Become a learning organization with focus on continuous stepwise improvement

# Lean Software Development

What do we need to learn?

1. Push and pull systems
2. Kanban
3. Systems Thinking
4. Flow
5. Work cell

# Lean Software Development



# Agile vs. Lean

- The goal of Agile is ultimately **to make the developing process flexible**, which is done by delivering in small, frequent iterations.
- The goal of Lean is **to make the developing process sustainable**, which is done by continuously **improving processes.**

# Agile vs. Lean

- Agile teams work in **feature-focused iterations**, with a pre-defined definition of "done" as the **measure** of progress for each iteration.

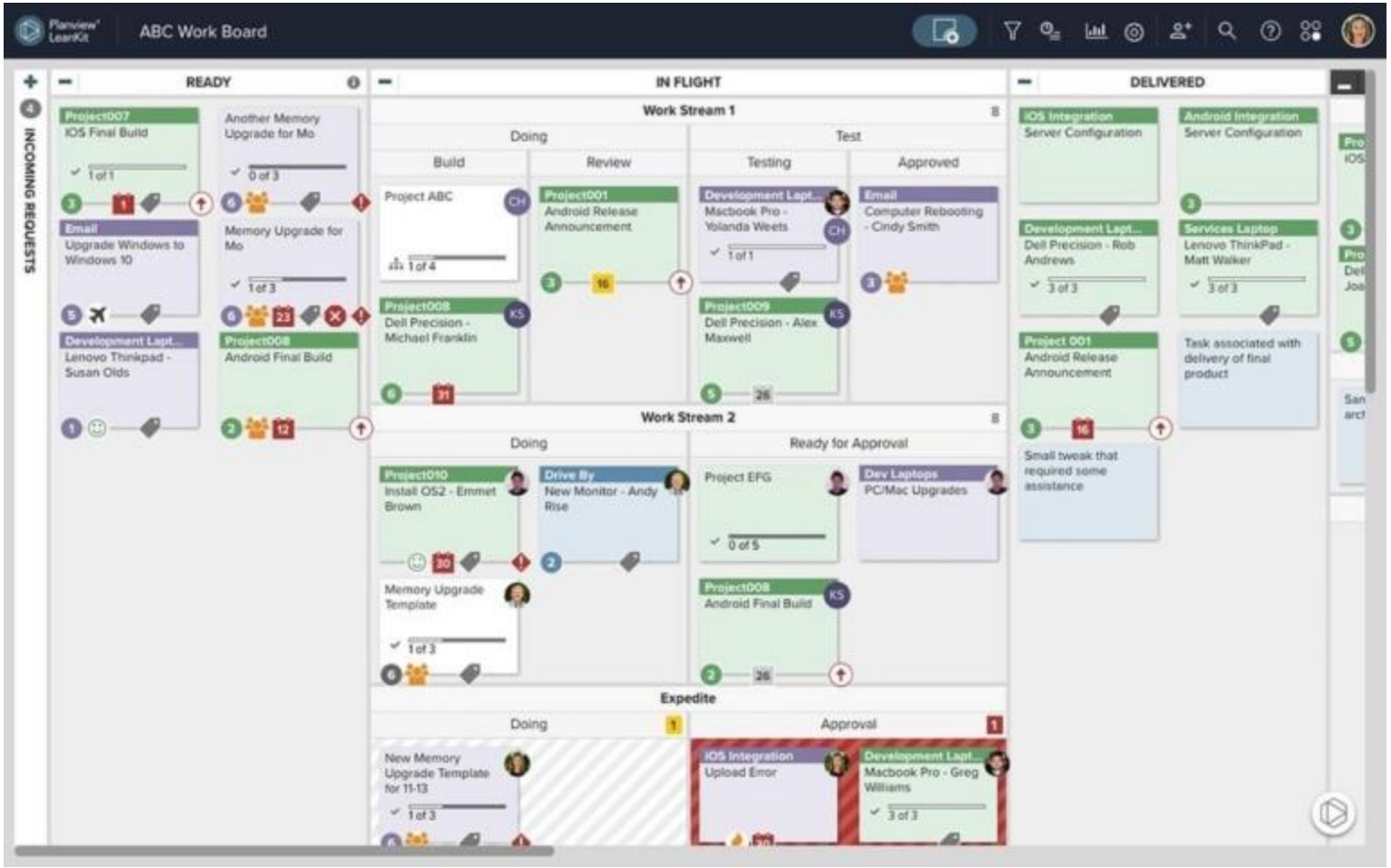
- Lean teams operate in a cycle of **"Build-Measure-Learn,"** defining progress as validated learning.
- Lean development involves testing, measuring, and validating hypotheses **based on trends in the market and past work**
- When planning and prioritizing work, Lean teams **focus on efforts** that would provide greatest value to the customer.
- They continuously identify ways to reduce waste while maximizing customer value.

# Agile vs. Lean

- Scrum boards
- Kanban boards

- Lean teams use Kanban boards
- They also use Kaizen, a continuous improvement method, to habitually identify and **remove waste**.

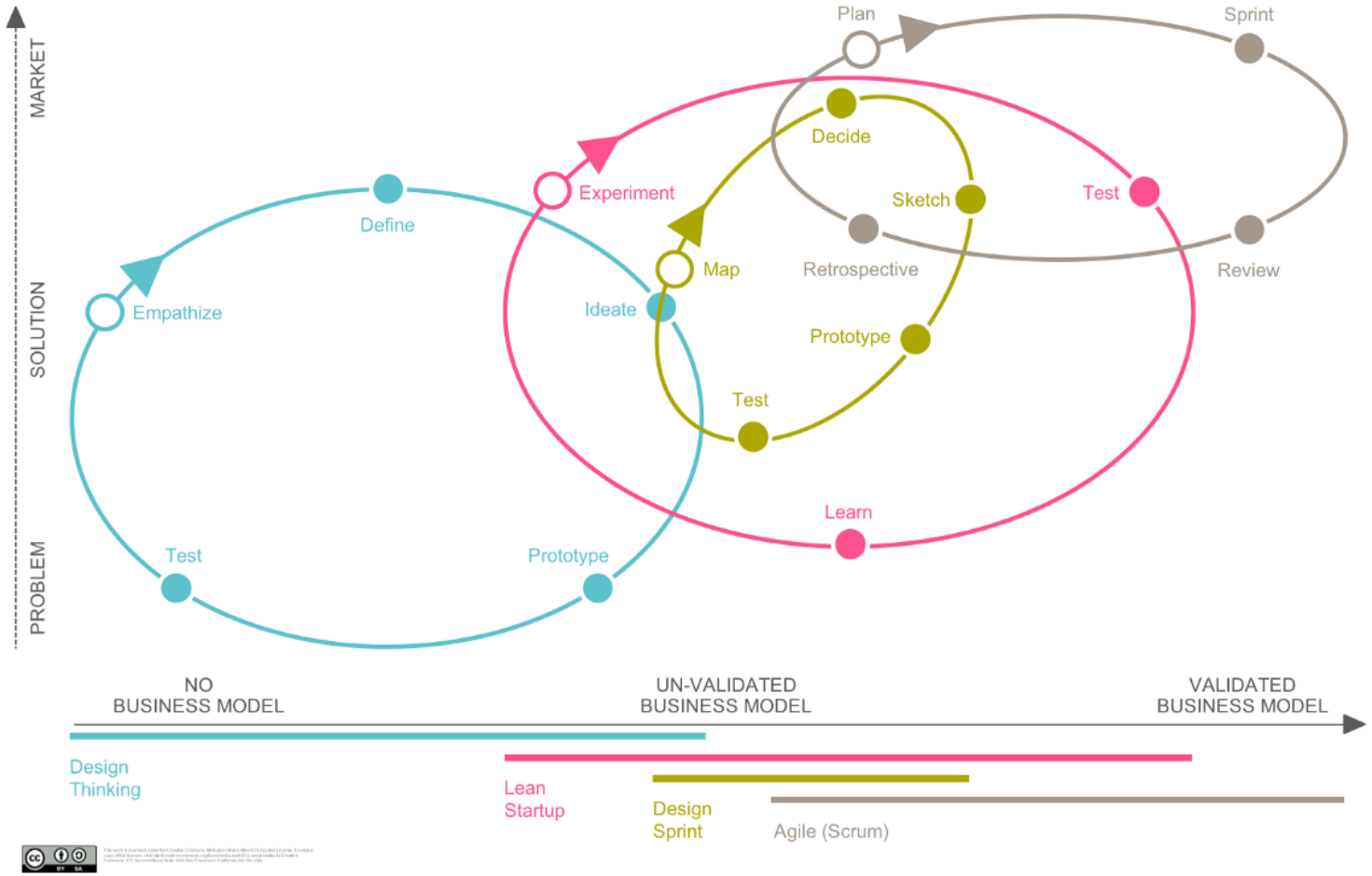
# Agile Lean board



# Agile vs. Lean

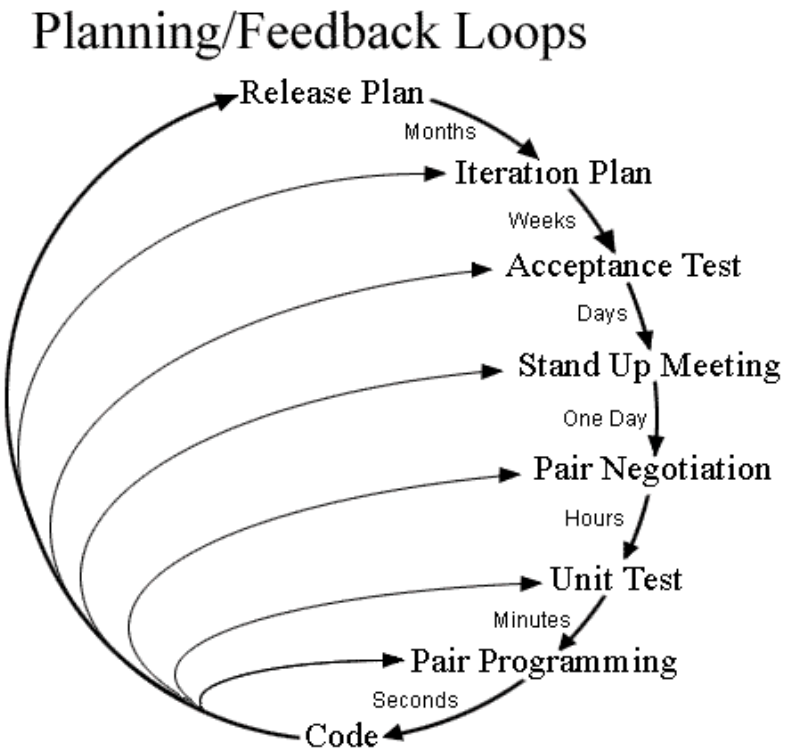
- Agile focuses on evolving products to better meet customer requirements, **but it does not address how to evolve processes to better support the evolution of the product.**
- Lean use continuous improvement for **how to evolve processes to better support the evolution of the product.**

# Design Thinking, Lean, Design Sprint, Agile

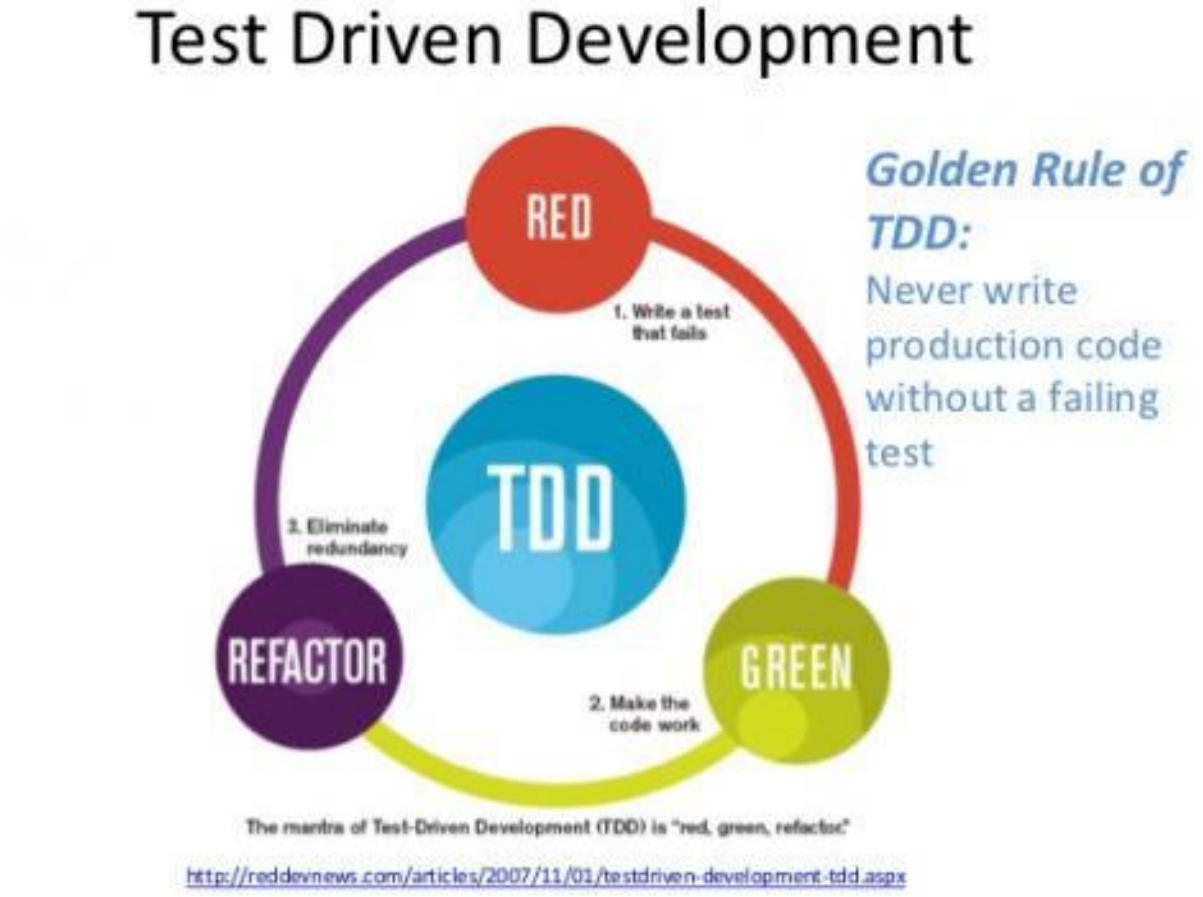


# Extreme Programming (XP)

- Improve software quality and responsiveness to changing customer requirements
- A type of agile software development
- Frequent "releases" in short development cycles
- Introduce checkpoints where new customer requirements can be adopted.



# Test-driven development (TDD)



# Test-driven development (TDD)

1. First the developer writes an (initially failing) automated test case that defines a desired improvement or new function,
2. Then produces the minimum amount of code to pass that test,
3. Finally refactors the new code to acceptable standards.

# Test-driven development (TDD)

1. First the developer writes an (initially failing) automated test case that defines a desired improvement or new function,
2. Then produces the minimum amount of code to pass that test,
3. Finally refactors the new code to acceptable standards.