

Overview/ Summary



This is about Agile Methodologies

- How we work together
- Not one line of code

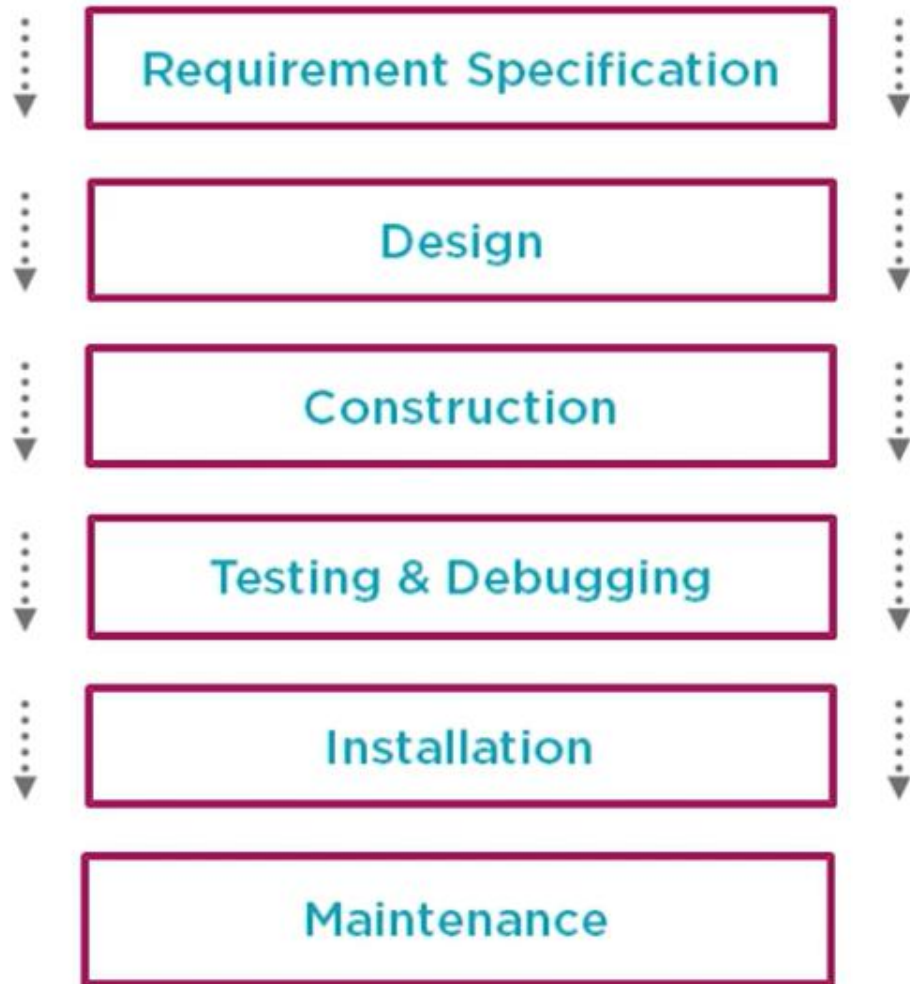


Comparing Development Processes

Agile and Plan Driven



Winston Royce's Waterfall Model



From the 1970 IEEE paper “Managing the Development of Large Software Systems”

Each phase should pass iteratively to the next

The entire process should be exercised twice before release

Royce knew that a single pass will fail

Unfortunately, for the process illustrated, the design iterations are never confined to the successive steps.



The Agile Manifesto

Original
Signatories

Kent Beck
Mike Beedle
Arie Van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Robert C. Martin
Steve Mellor
Ken Sutherland
Dave Thomas



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:

Individuals and Interactions

Working Software

Customer Collaboration

Responding to Change



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

Processes and Tools

Comprehensive Documentation

Contract Negotiation

Following a Plan

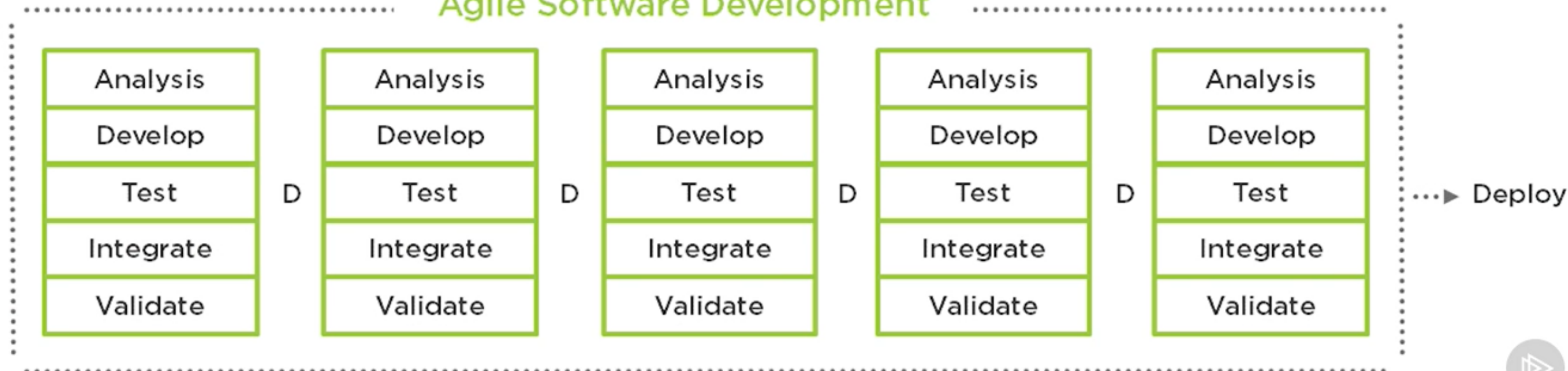


The Big Difference

Plan Driven Software Development



Agile Software Development



Comparing Methodologies

Plan Driven Methodologies

Change is bad, therefore discouraged
and actively controlled

Adherence to the plan determines
success or failure

I am done when my part of the plan
is signed off

Agile Methodologies

Change is inevitable and valuable,
therefore encouraged and embraced

Incentives are often based on customer
satisfaction and ROI

I am done when the customer is happy



Comparing Methodologies

Plan Driven Methodologies

Lots of gates to control quality

Inspect product when it is complete

Start by predicting what will
be delivered

Agile Methodologies

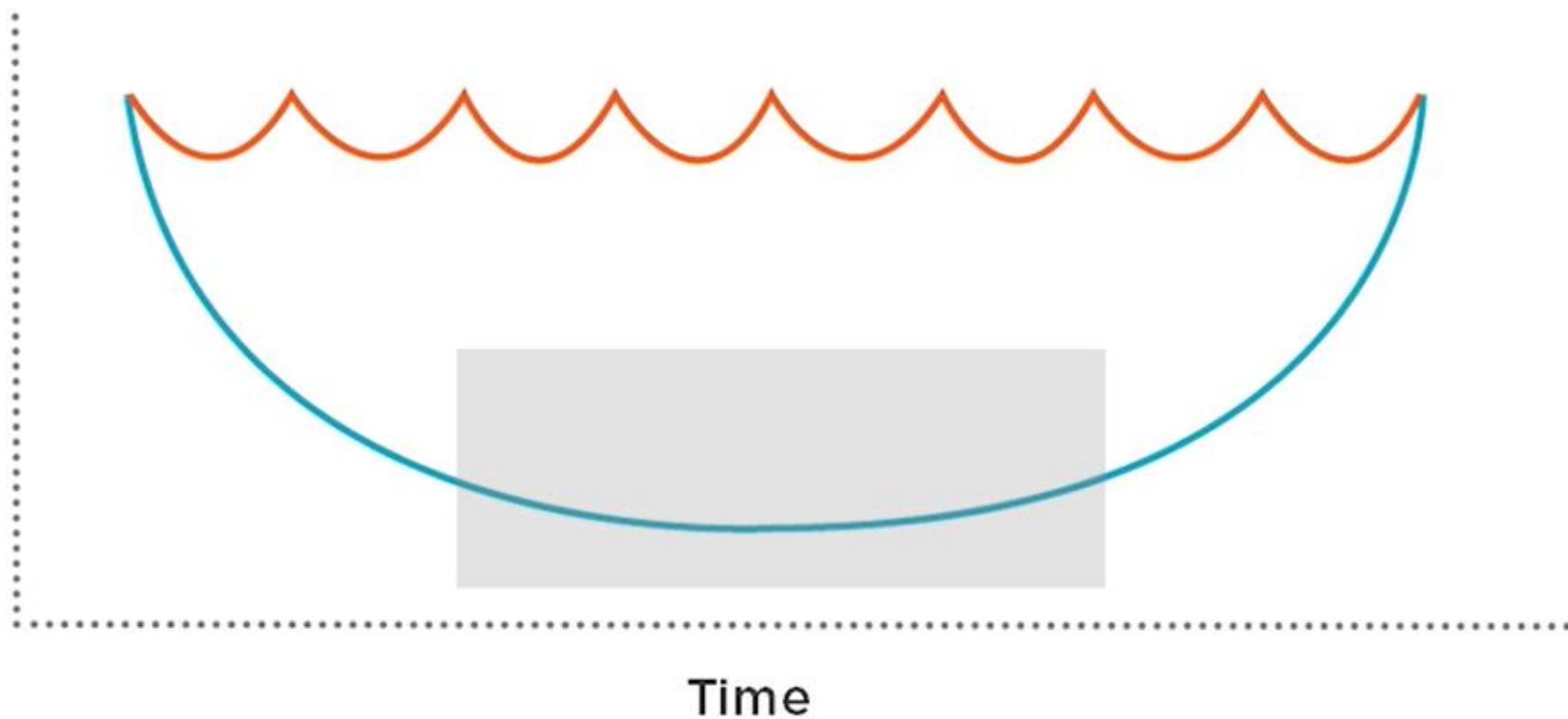
Highly iterative to achieve quality

Inspect work as it is being done

Start with a goal of filling a need



Visibility

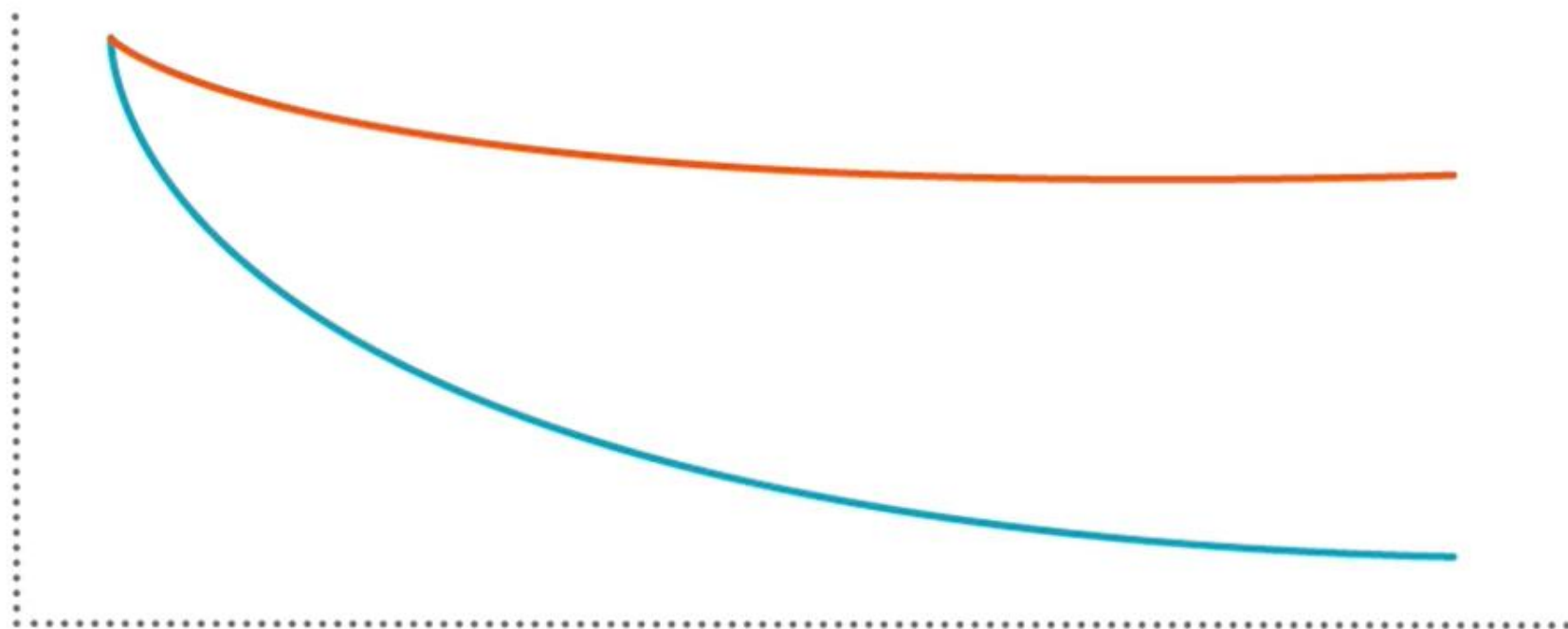


Agile Software Development
Plan Driven Development



Ability to Change

This is true of software produced and business decisions made



Time



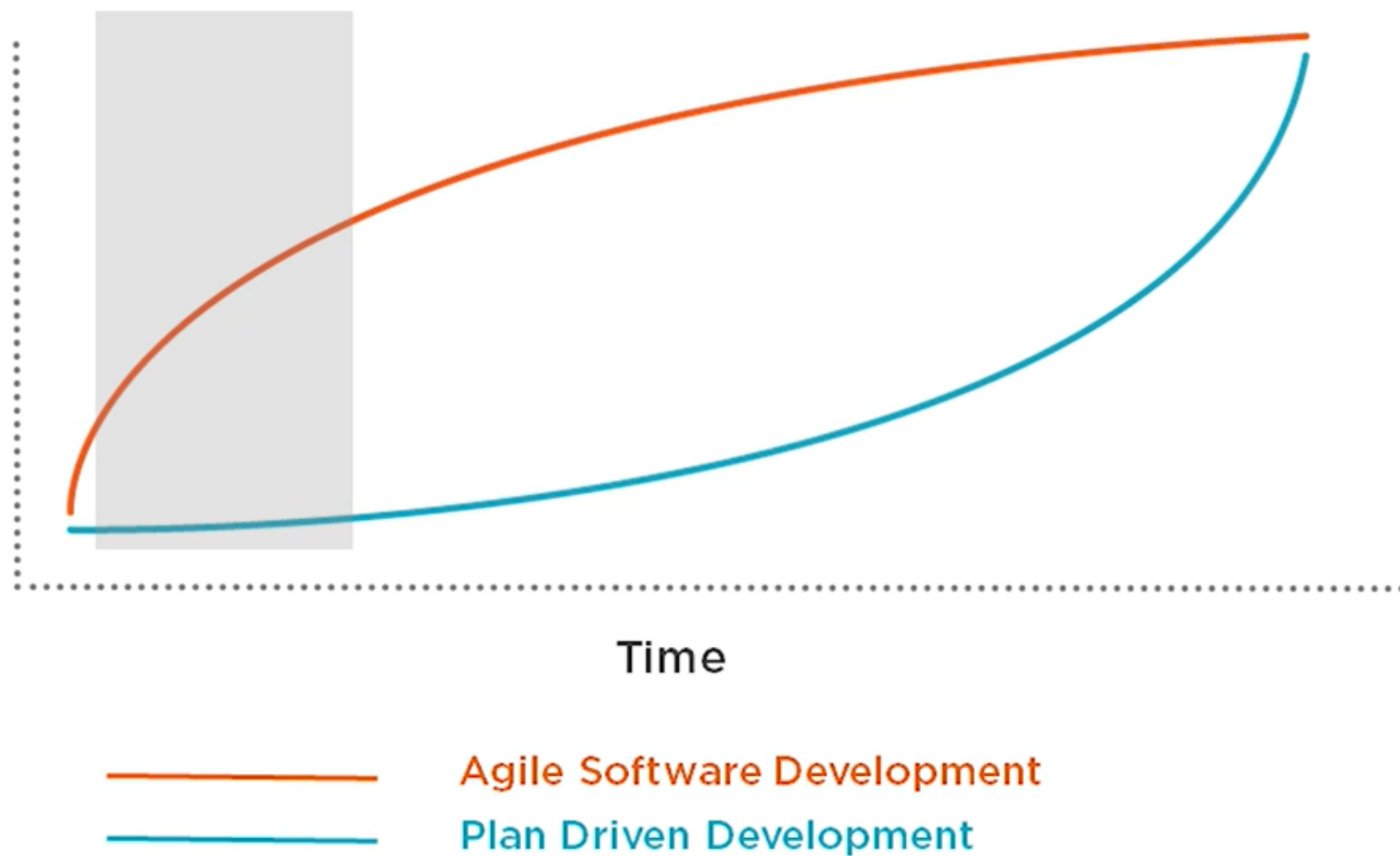
Agile Software Development



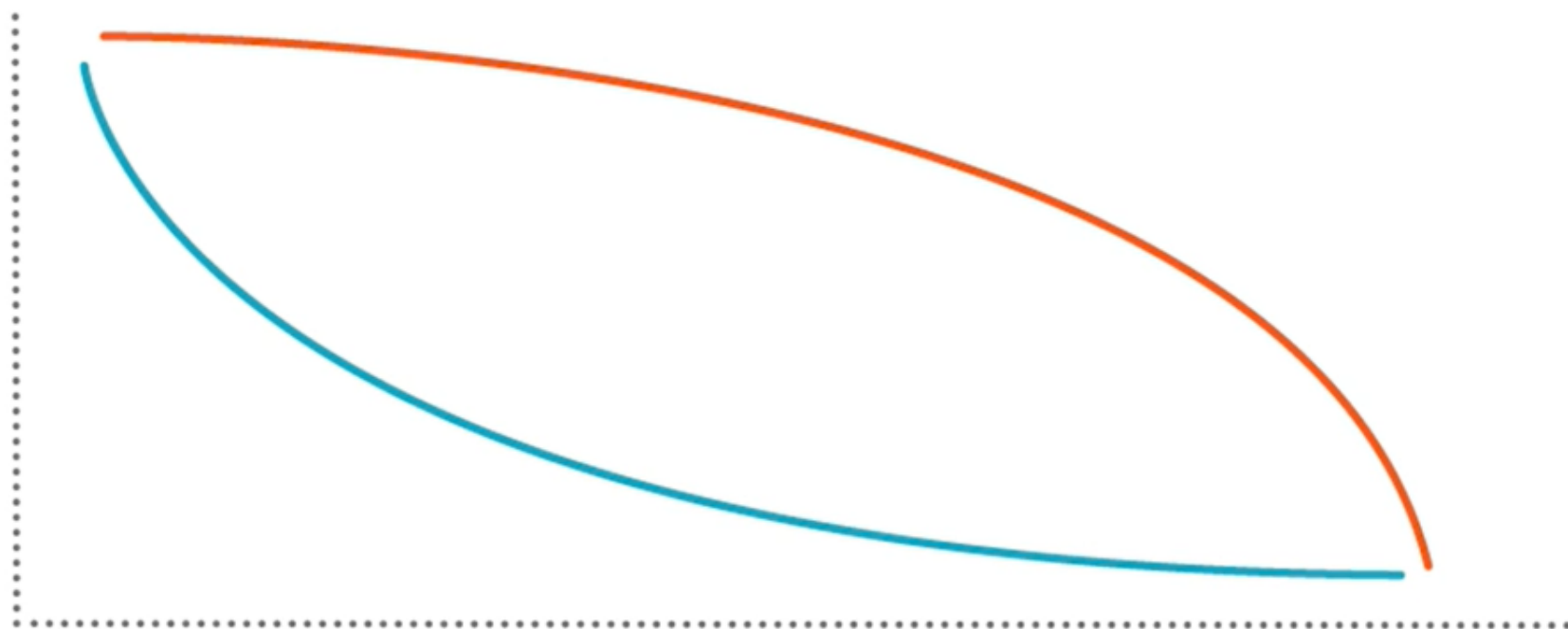
Plan Driven Development



Business Value



Risk



Time



Agile Software Development



Plan Driven Development



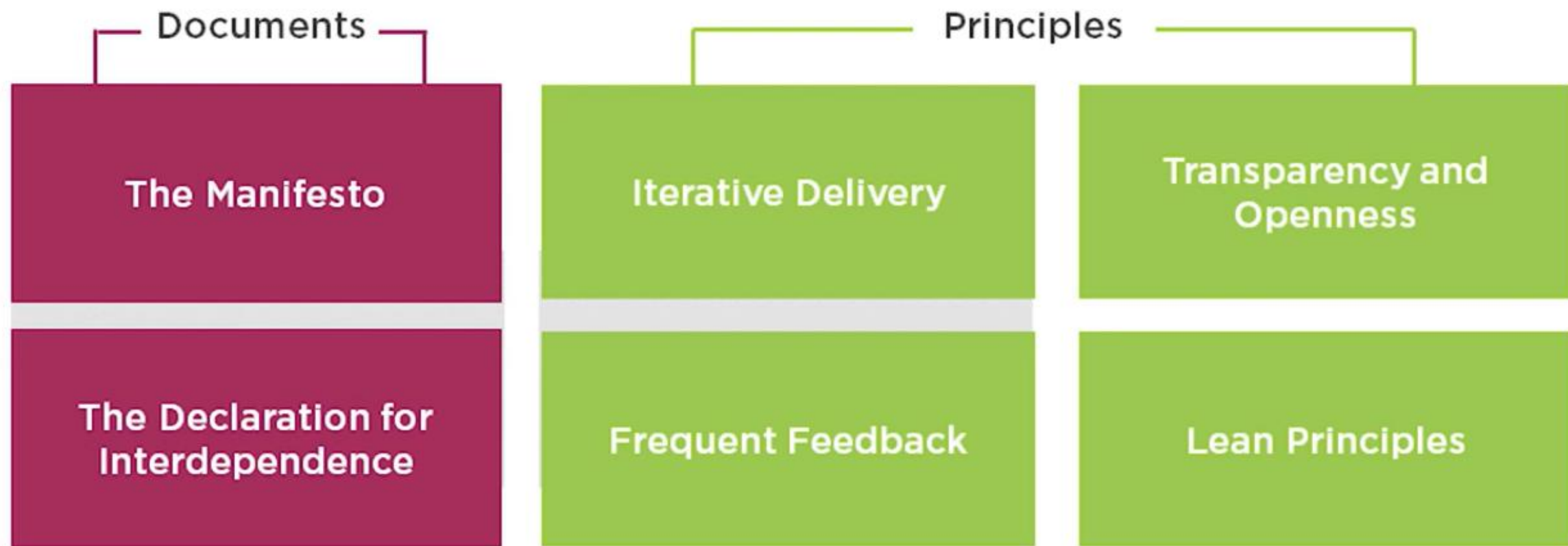
What Agile is and is not

Dispelling Common Myths



Touchstones of Agility

Come back to these things to recall what we are trying to accomplish.



The Two Parts of the Agile Discussion

Processes and Methodologies

Working with People

- Planning
- Teamwork
- Engaging customers
- Providing leadership
- Collaboration
- Learning

Techniques and Practices

Working with Software

- Design
- Coding
- Testing
- Deploying
- User experience



Agile Is

Iterative

Adaptive

Value Based

Easy to
Understand

Hard to
Implement



Agile Is Not

Just about
writing code

Undisciplined

Unstructured

Whatever you
want it to be

A placebo for
pesky developers



Contemporary Agile Methodologies

Proven Practices



Extreme Programming (XP)

Ancestor of most Agile methodologies

Originated with Kent Beck in 1999

Blends processes and practices

Found success in smaller teams

XP became controversial early on due to advocates teaching it as dogma



Extreme Programming (XP)

1. Take observed effective team practices 2. Push them to extreme levels

Good Practice

Code Reviews
Testing
Software Design
Simplicity
Integration Testing
Short Iterations

Pushed to the Extreme

Pair Programming
TDD and constant regression
Relentless Refactoring
The simplest thing that could work
Continuous Integration
The Planning Game



XP's 12 Practices

1. The Planning Game

5. Testing

9. Collective Ownership

2. Small Releases

6. Refactoring

10. On-site Customer

3. Metaphor

7. Pair Programming

11. The 40-hour Week

4. Simple Design

8. Continuous Integration

12. Coding Standards



Scrum



An iterative project management process



Used beyond software development



Originated with Ken Schwaber and Jeff Sutherland in 1990s



Blends well with XP development



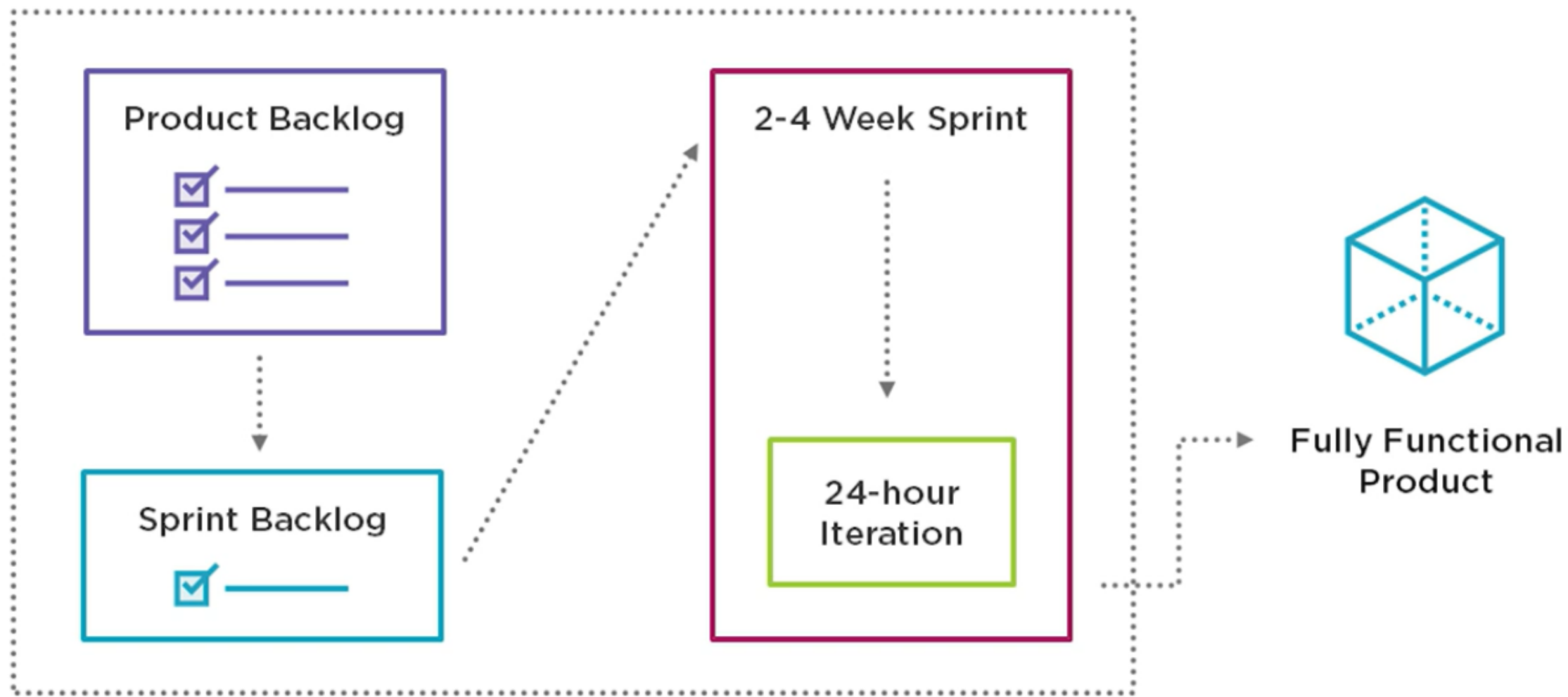
Does not advocate any specific engineering techniques



Simple



Scrum



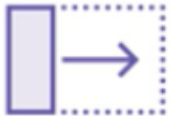
Lean Software Development



More a set of guidelines than a formal methodology



Originally applied in manufacturing, now used in software development



Can be applied to improve any process



Focuses on Continuous Improvement (Kaizen) and value flow



Principles of Lean Software Development

Eliminate waste

Amplify learning

Respect people

Build Quality

Defer commitment

Deliver Fast

Recognize and optimize the whole

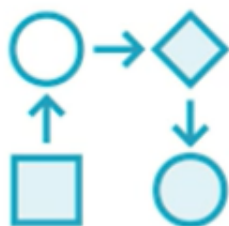


Feature Driven Development

Originated with Jeff De Luca while working on a 50-person project at Singapore bank in 1997 and adheres to strict processes.



Based on time-honored engineering practices



Advocates modeling as the base currency of process



Some reports claim FDD scales more effectively than Scrum



Feature Driven Development

Activities

1. Develop Overall Model
2. Build Feature List
3. Plan by Feature
4. Design by Feature
5. Build by Feature

Milestones

1. Domain Walkthrough
2. Design
3. Design Inspection
4. Code
5. Code Inspection
6. Promote to Build



<action> <result> <object>

Sum the total for monthly sales

Validate the password of the user



Feature Driven Development

Activities

1. Develop Overall Model
2. Build Feature List
3. Plan by Feature
4. Design by Feature
5. Build by Feature

Milestones

1. Domain Walkthrough
2. Design
3. Design Inspection
4. Code
5. Code Inspection
6. Promote to Build



FDD Practices

Domain Object Modeling

Developing by Feature

Individual Code Ownership

Feature Teams



FDD Practices

Inspections

Configuration Management

Regular Builds

**High Visibility Progress
and Results**



References

The Manifesto for Agile Software Development
AgileManifesto.org

Agile Alliance
AgileAlliance.org

