Agile Practices

```
- 12 principles of agile manifesto
  - ways that you have seen them applied or not
  - what have the consequences been?
- conscious capitalism
- cynefin
- read the scrum guide
- scope creep the Pentagon wars (Youtube video)

extra:
- Kent Beck extreme programming
```

12 Principles of Agile Manifesto:

1.

Waterfall Development and Its Problems

Waterfall methodology

- What is Waterfall?
- Advantages
- Disadvantages

V-Model methodology

- What is the V-Model?
- Advantages
- Disadvantages

Extreme Programming

What is extreme programming?

- Explanation of extreme programming and its history

Key concepts of extreme programming

- Activities
- Principles
- Values
- Rules

Scrum

What is Scrum?

Explanation of Scrum and its history

Key concepts of Scrum

Scrum values, roles, artifacts, and activities

History of the Waterfall Model

First introduced by Winston Royce

Was not referred to as Waterfall until later

Requirements specification

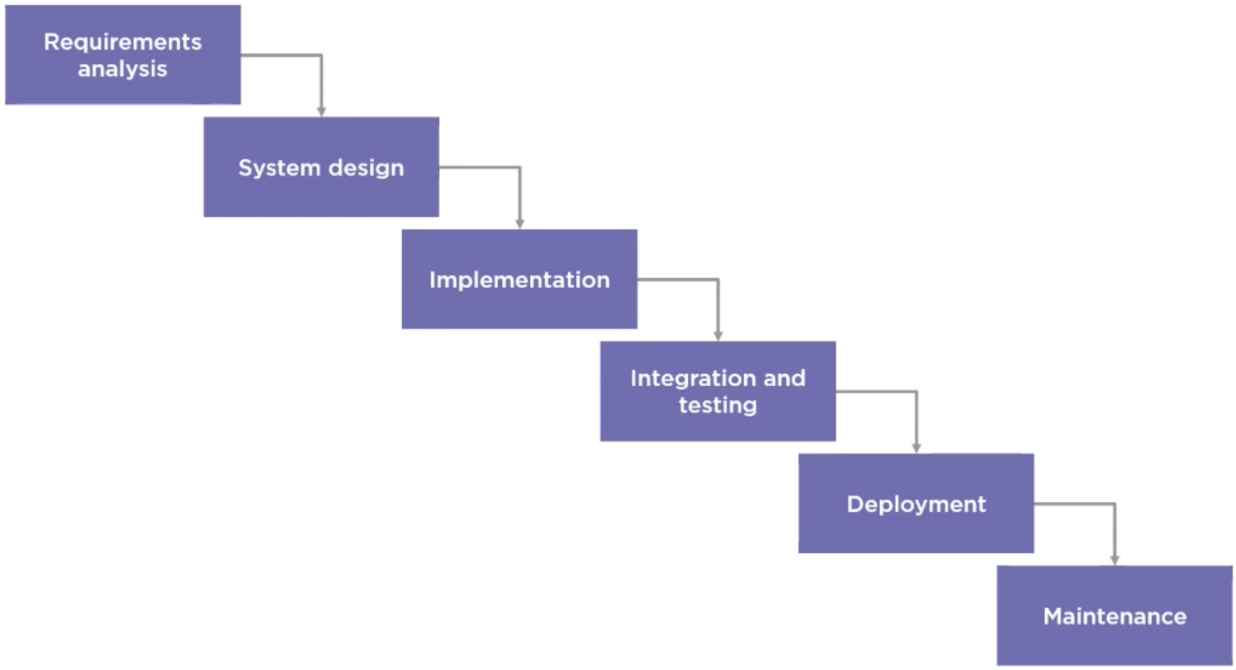
Design

Construction

Integration

Testing and debugging

Installation



When to use Waterfall process:

Requirements are well documented, clear and fixed

Product definition is stable

Technology is well understood

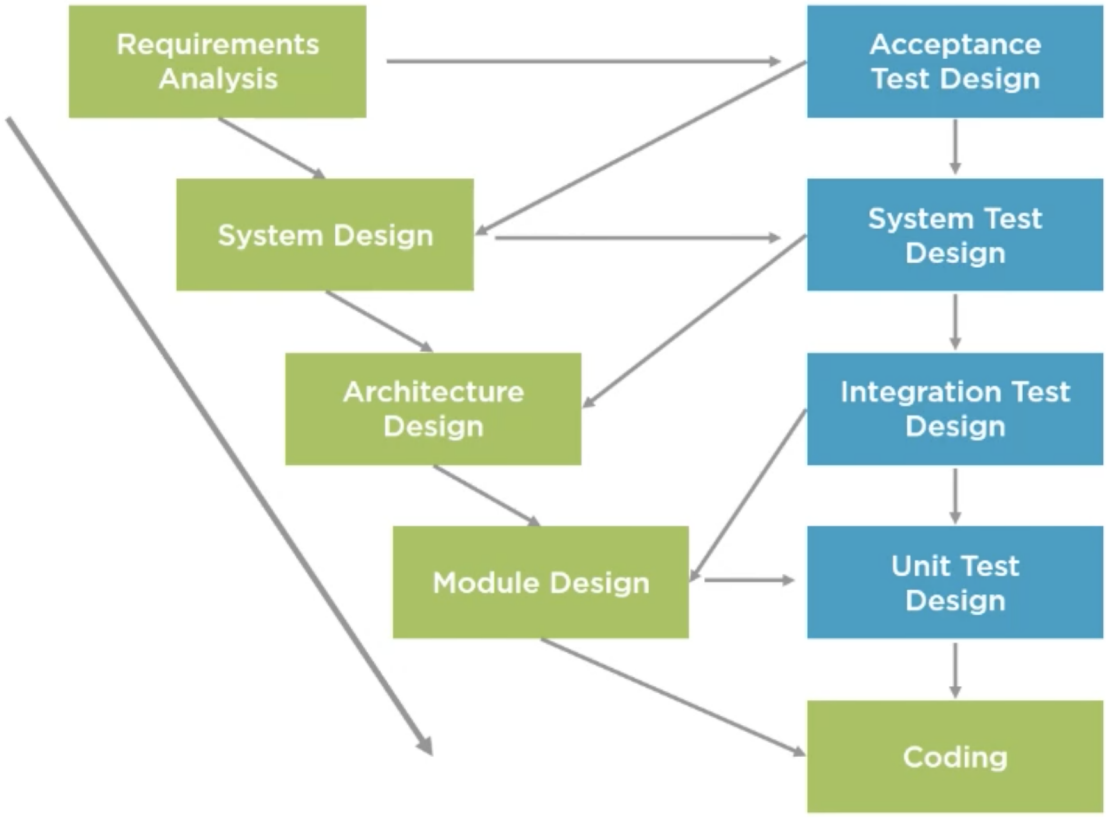
No ambiguous requirements

The project is short

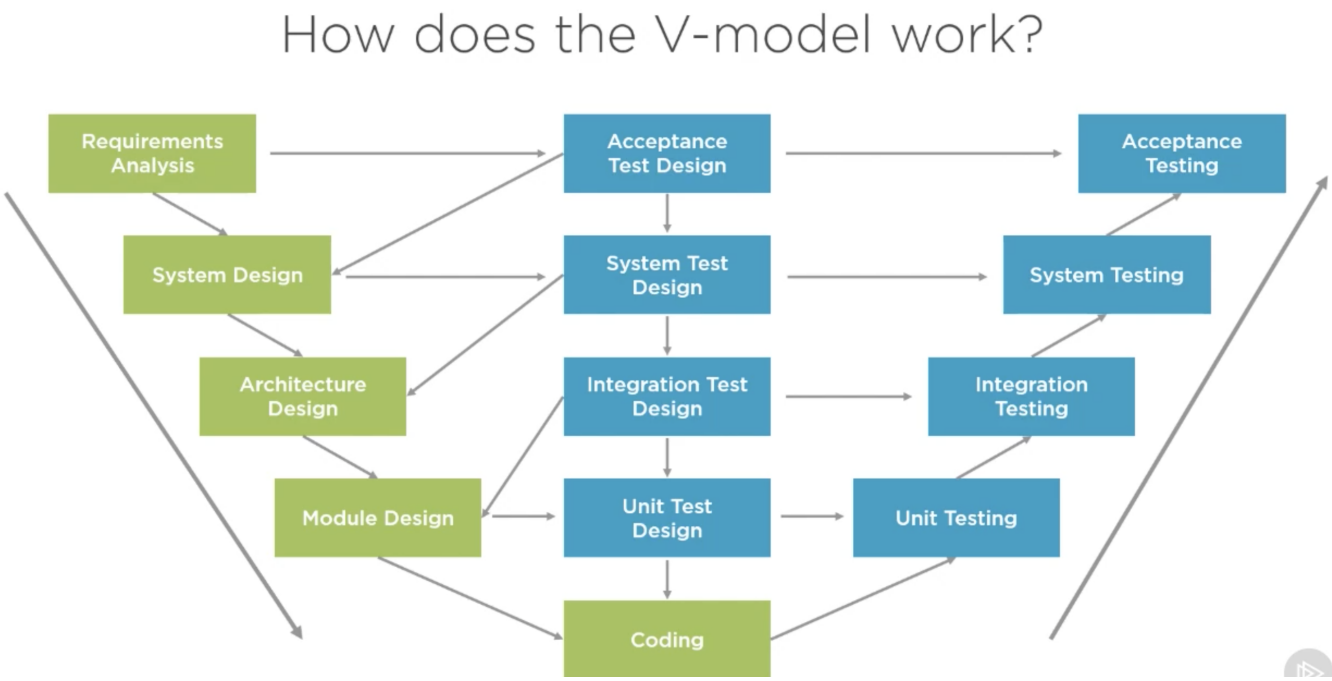
Suitable resources available

V Model:

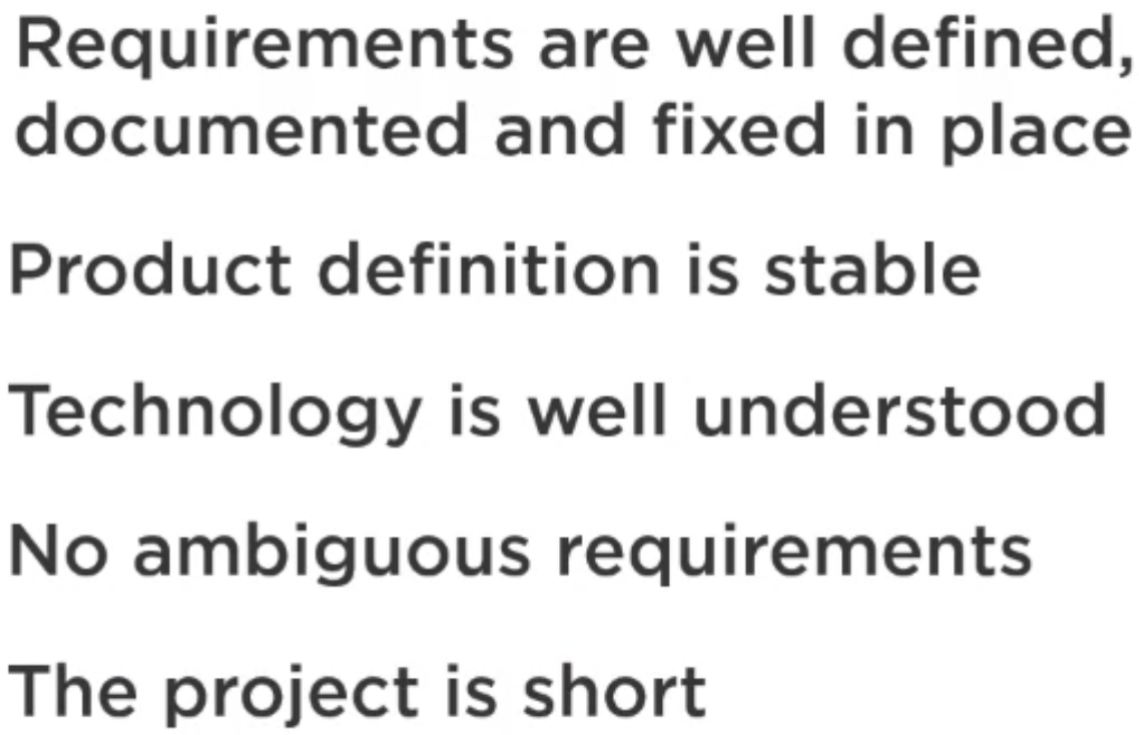
Left side:



Full V model:



When to use the V Model:

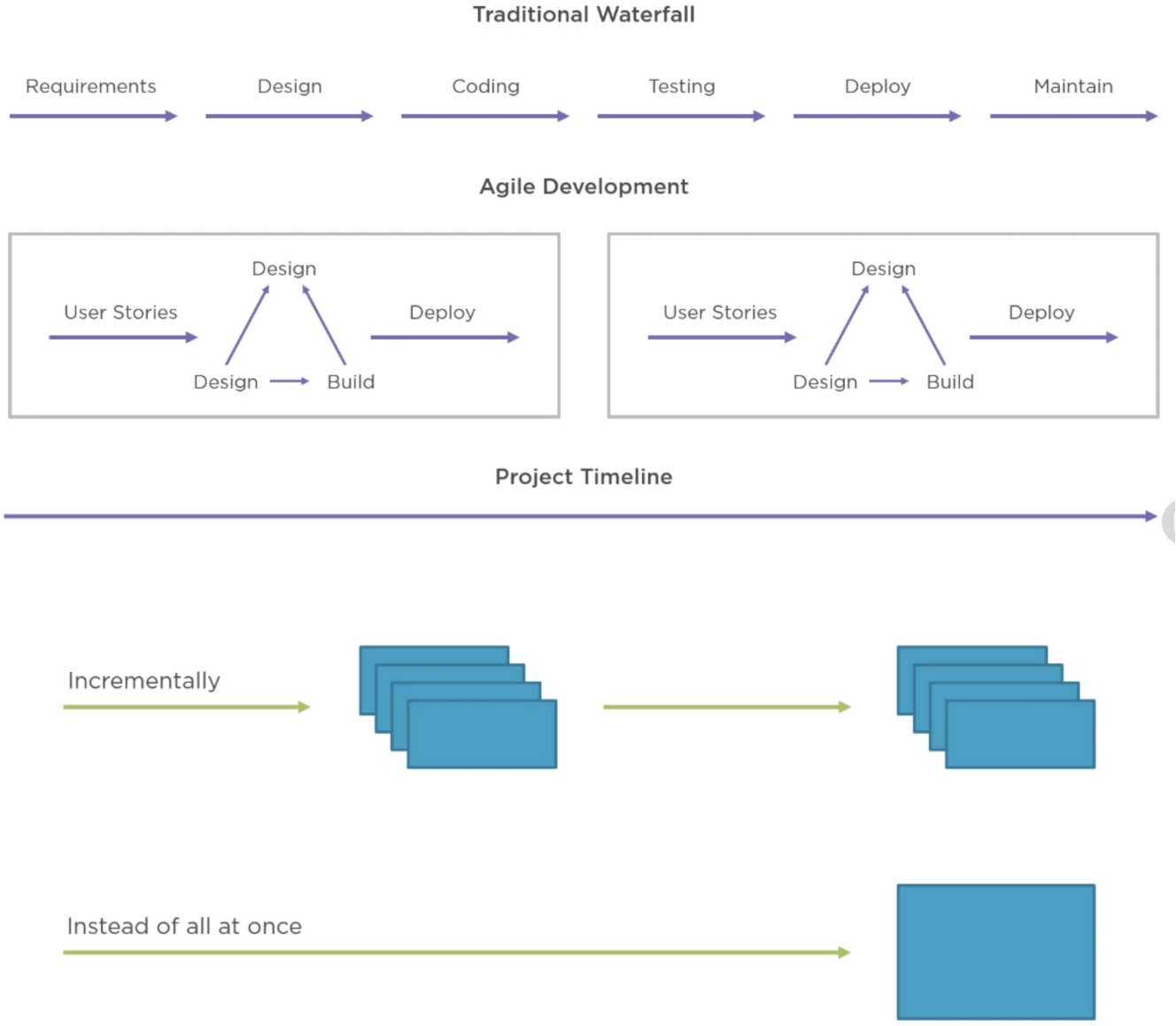


These are Pros and Cons of V Model:

Pros	Cons
Disciplined model where phases are complete one at a time	High risk and uncertainty
Works well for small projects	Not a good model for complex projects
East to understand and use	Not good for changes in requirements
Easy to manage due to the rigidity of the model	Once in testing, difficult to go back

There is also the issue of users not seeing the product until late in the development phase.

This is the Agile timeline:


Traditional Waterfall
Requirements
Design
Coding
Testing
Deploy
Maintain
Agile Development
User Stories
Design
Deploy
Design
Build
User Stories
Design
Deploy
Design
Build
Project Timeline
Incrementally
Instead of all at once

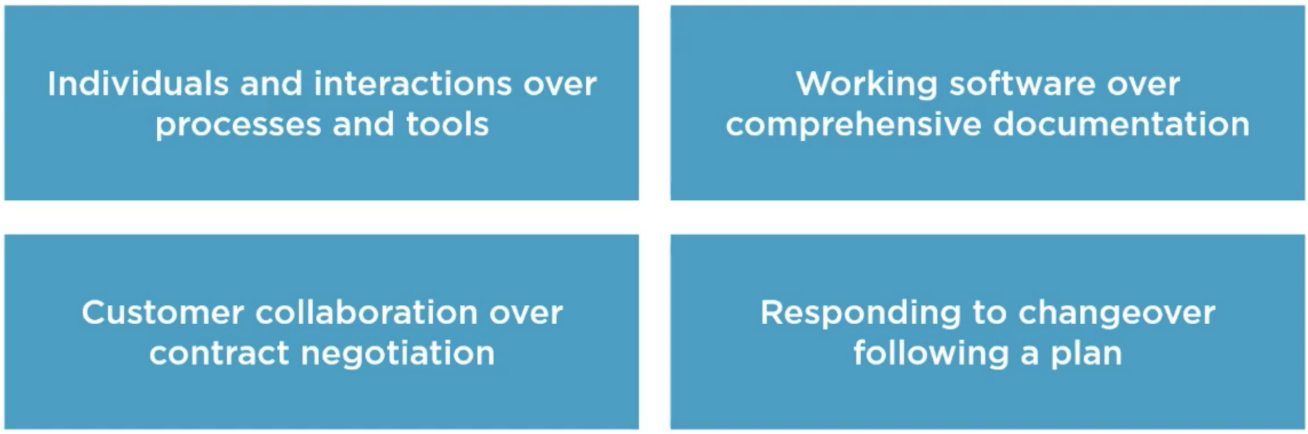
Agile Manifesto:

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

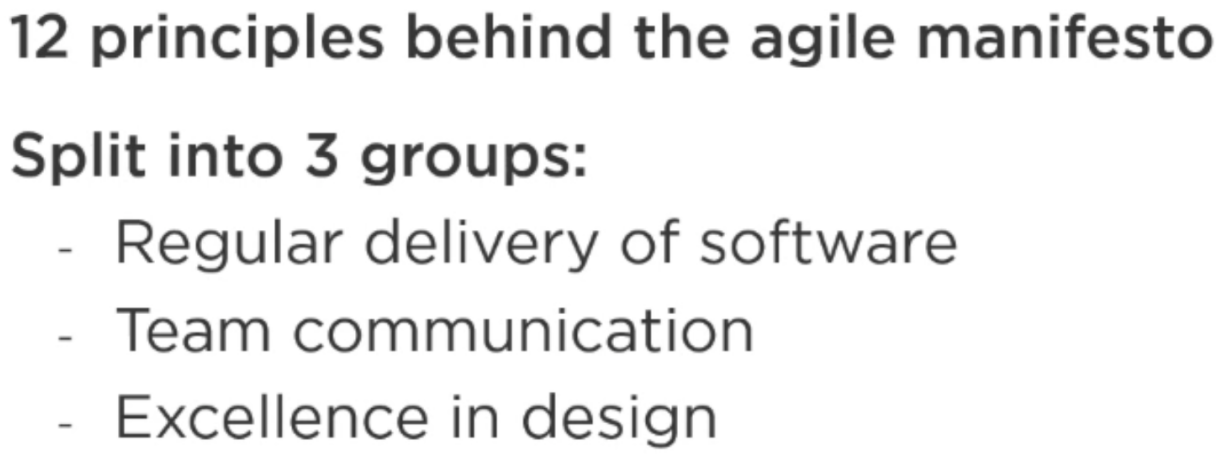
- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

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

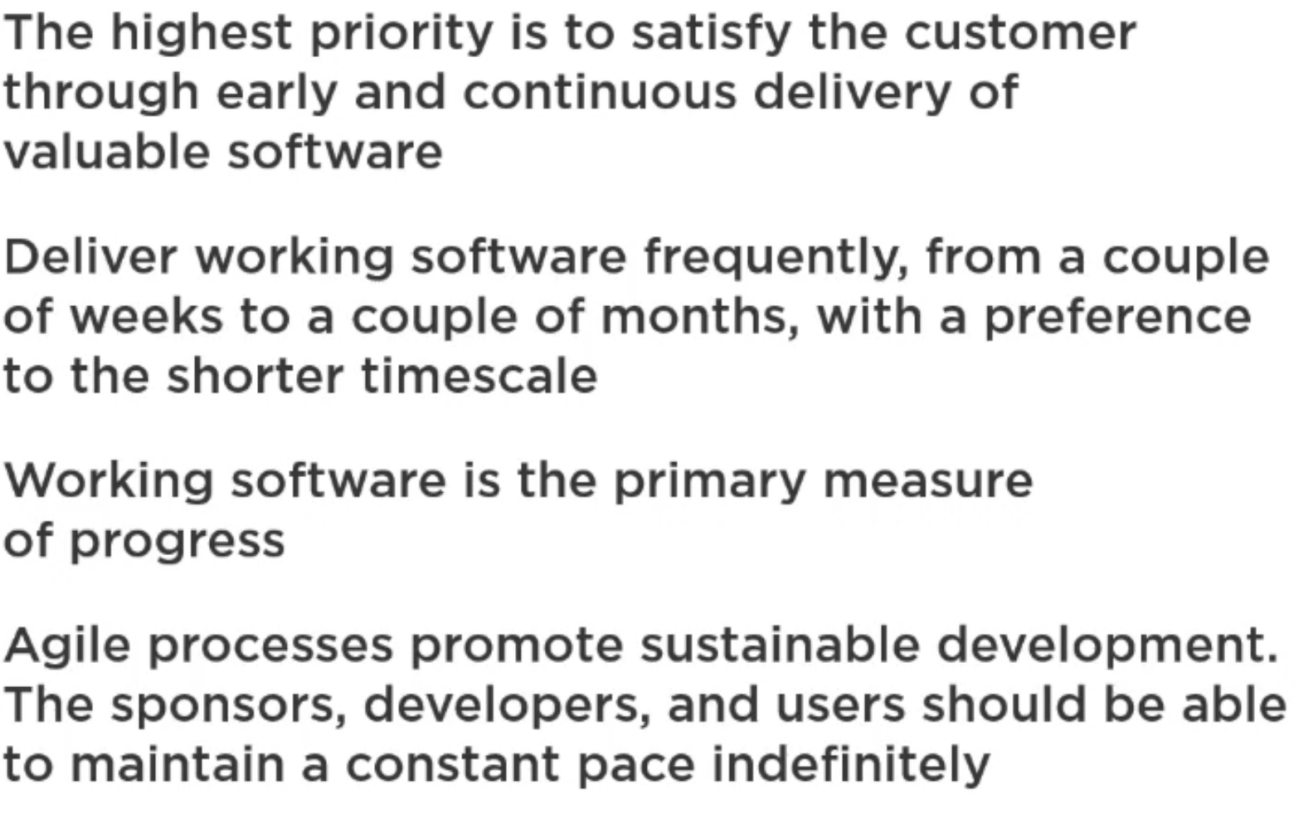
These are the values:



Agile Manifesto principles can be split into 3 main groups:



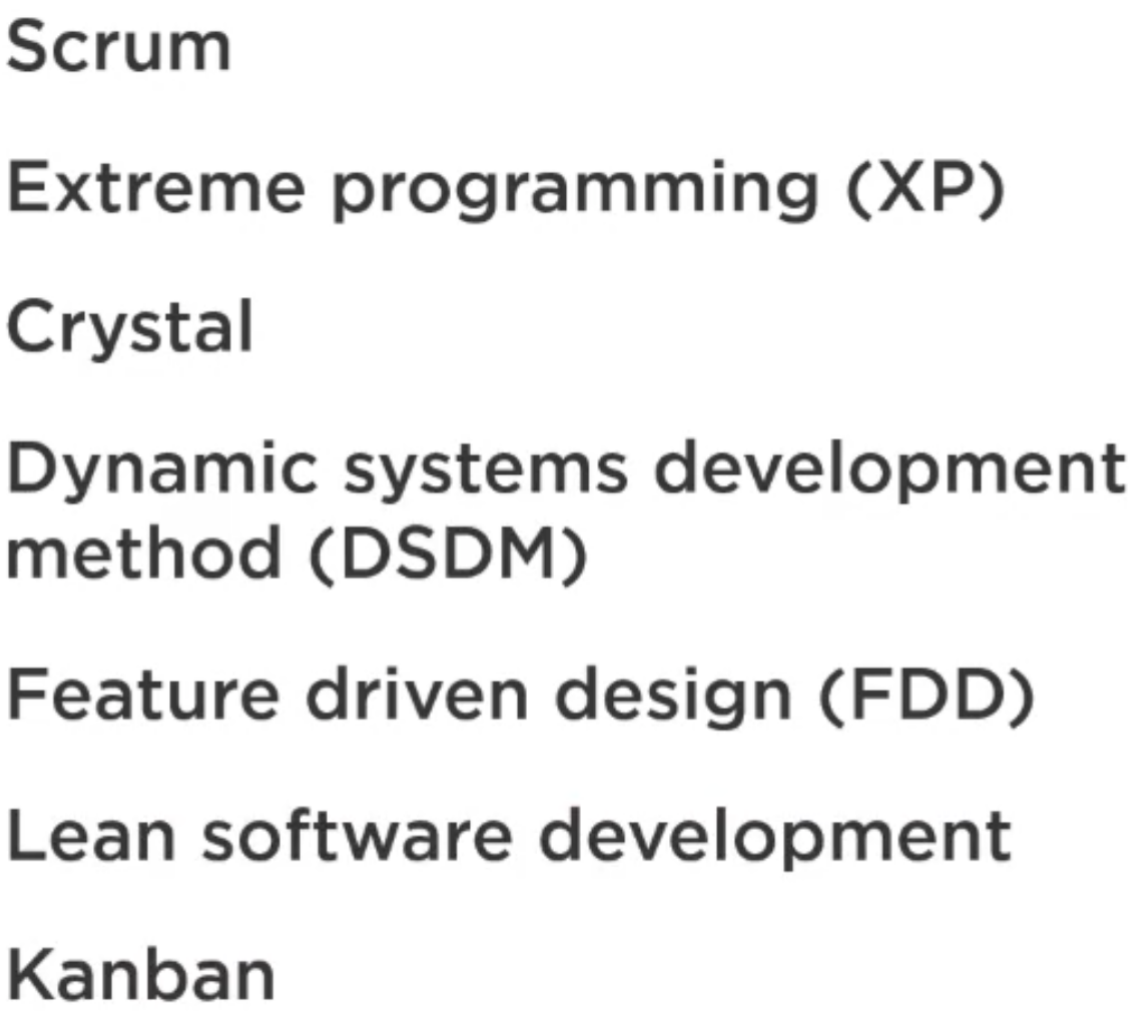
Regular delivery of Software:



These are the principles for team communication:

Excellence in Design

These are different models for agile:

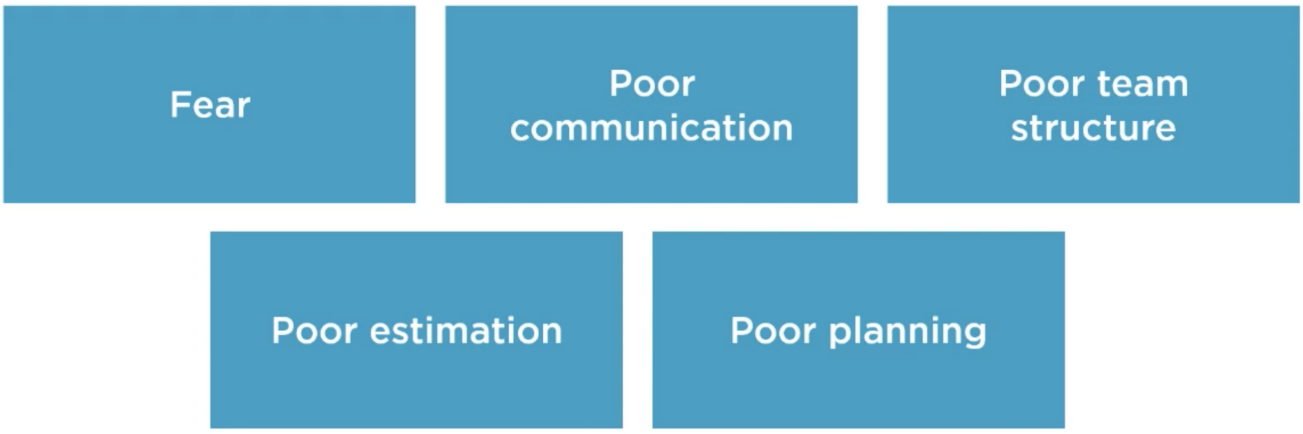


These are the Roles in an Agile Team:

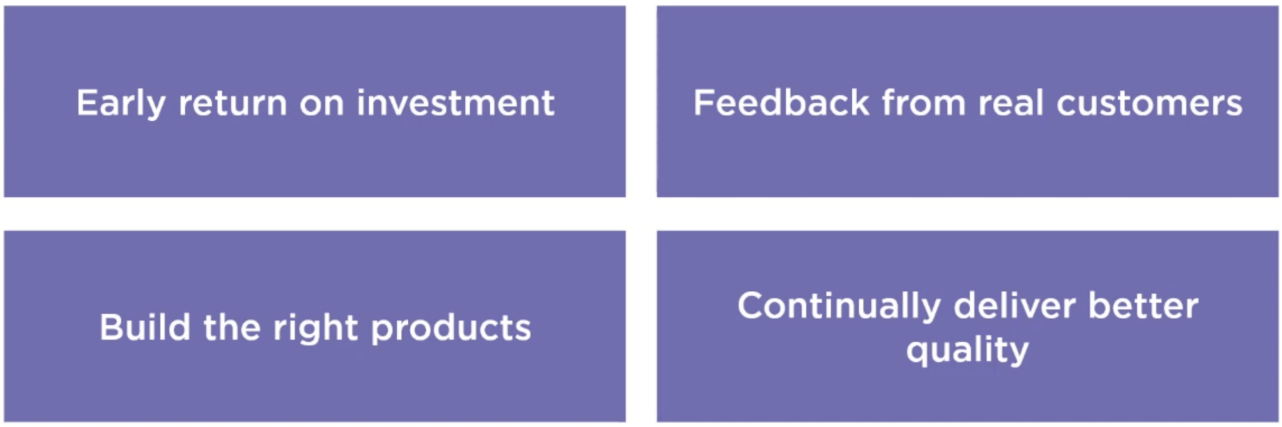
The main point regarding agile is that we have iterative development towards a shared goal

These are some common agile misconceptions:

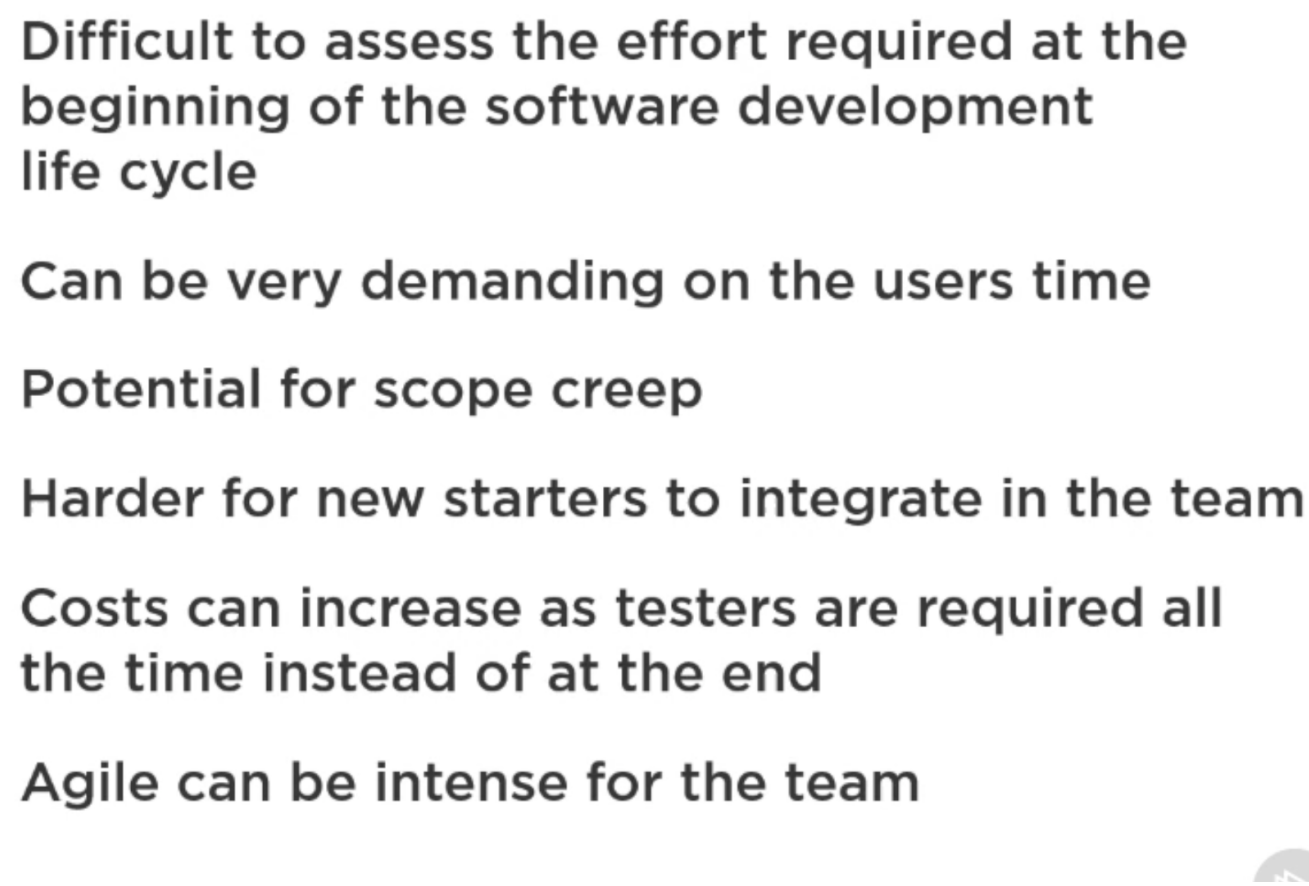
These are common mistakes made by agile teams:



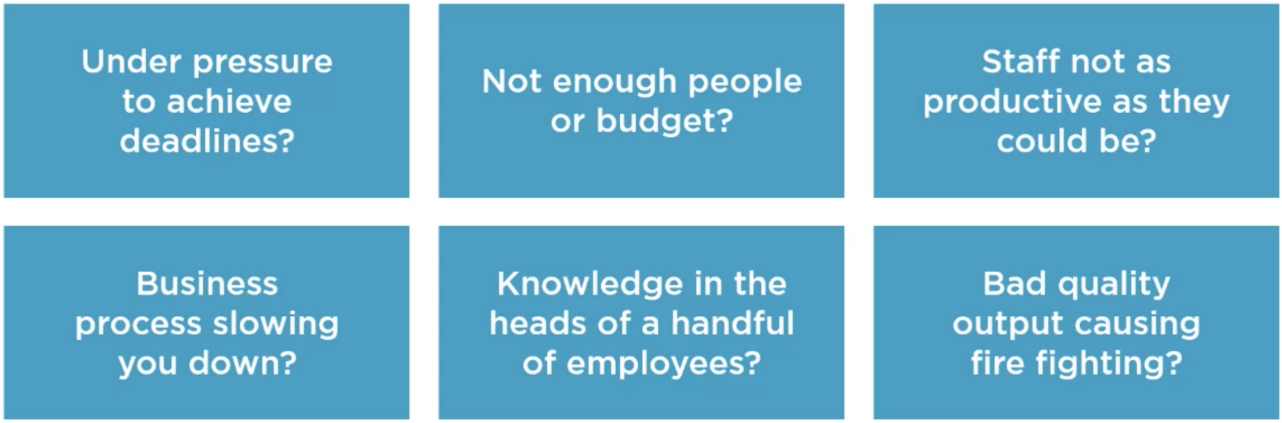
Agile Advantages and Disadvantages



Here are some disadvantages of Agile:



These are some examples of situations that may profit from an agile approach:



XP Programming

- Pairs
- Flat management structure
- Frequent communication with the customer
- Code reviews This is where XP programming came from (Kent Beck's work with Chrysler) https://www.martinfowler.com/bliki/C3.html

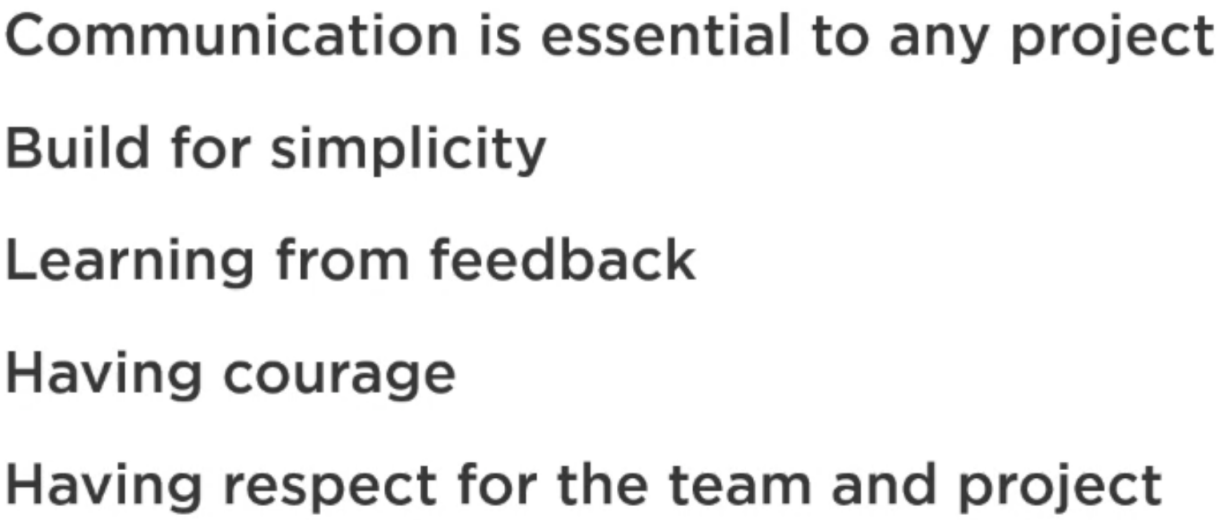
XP Programming

- 4 x Activities - coding, listening, testing, design

- 5 values - communication, simplicity, feedback, courage and respect
- 3 principles
- 12 practices
- 29 rules

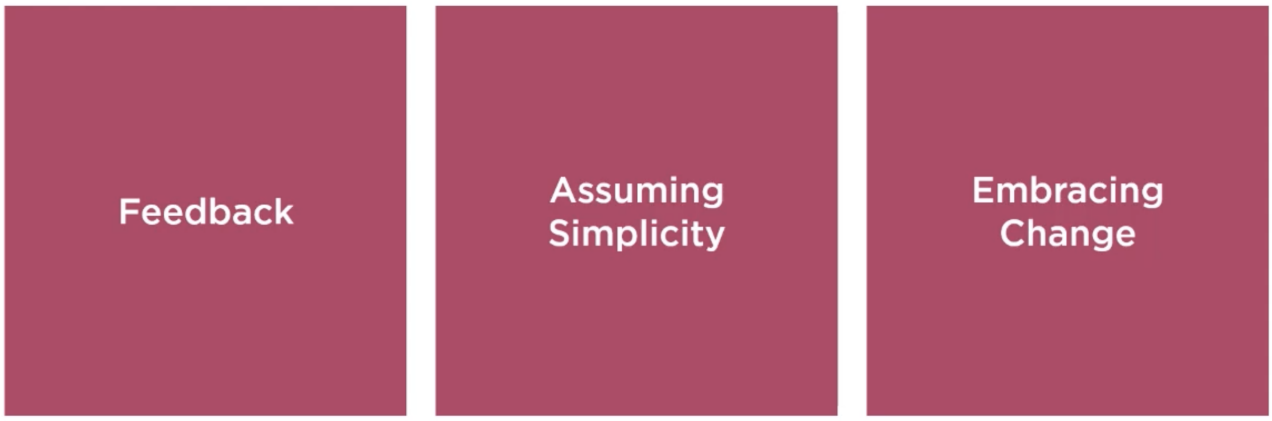
XP Programming 4 activities:

5 values of XP programming:

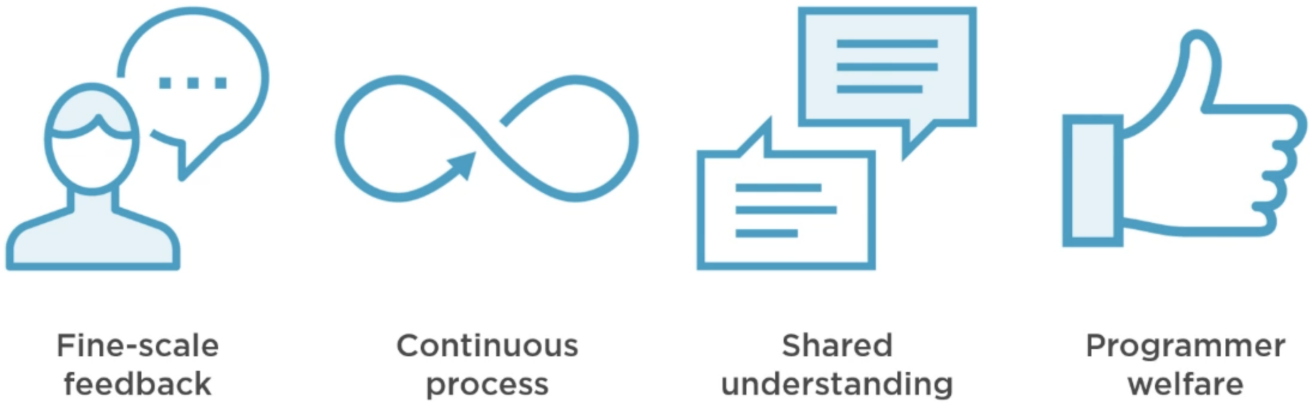


Extreme Programming Principles

There are 3 main XP principles -



These are the XP Practices:



Fine Scale Feedback

- Pair Programming
- Planning game
- Test driven development
- Whole team

Continuous Process

- Continuous integration
- Refactoring or design improvement
- Small releases

Shared understanding

- coding standards (linter etc.)
- collective code ownership
- simple design

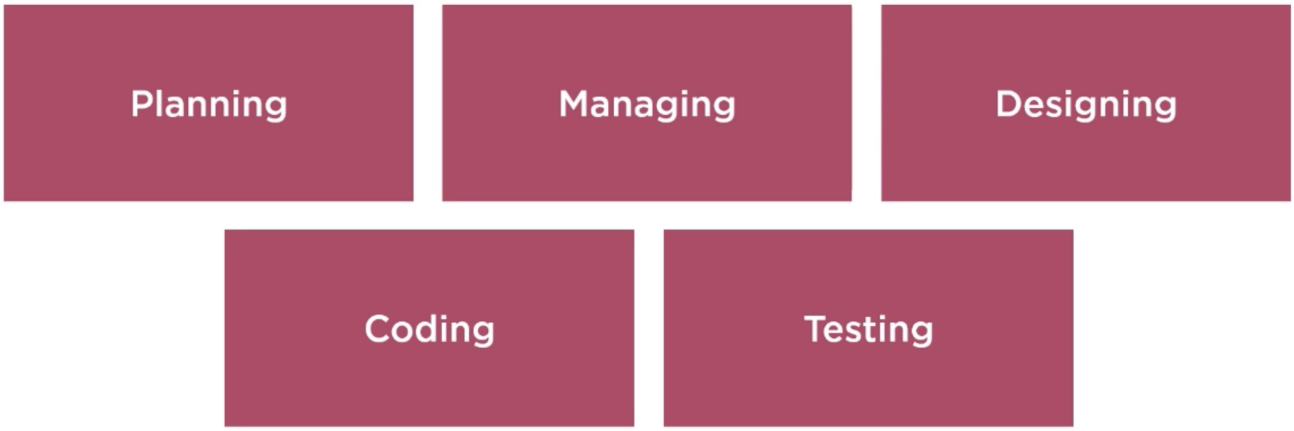
- System metaphor

Programmer welfare

- sustainable pace

Extreme Programming Rules

Extreme programming has 29 rules split into 5 sections:



Planning

- user stories are written
- release planning
- make frequent small releases
- divide project to iterations
- plan each iteration

Managing

- give the team an open work space
- set a sustainable pace
- stand up meeting starts each day
- velocity is measured
- change roles
- fix XP when it isn't working

Design

- simplicity
- choose a system metaphor
- use CRC cards for design sessions (which objects send messages to other objects)
- use spike solutions to reduce risk
- no functionality added early
- refactor whenever possible

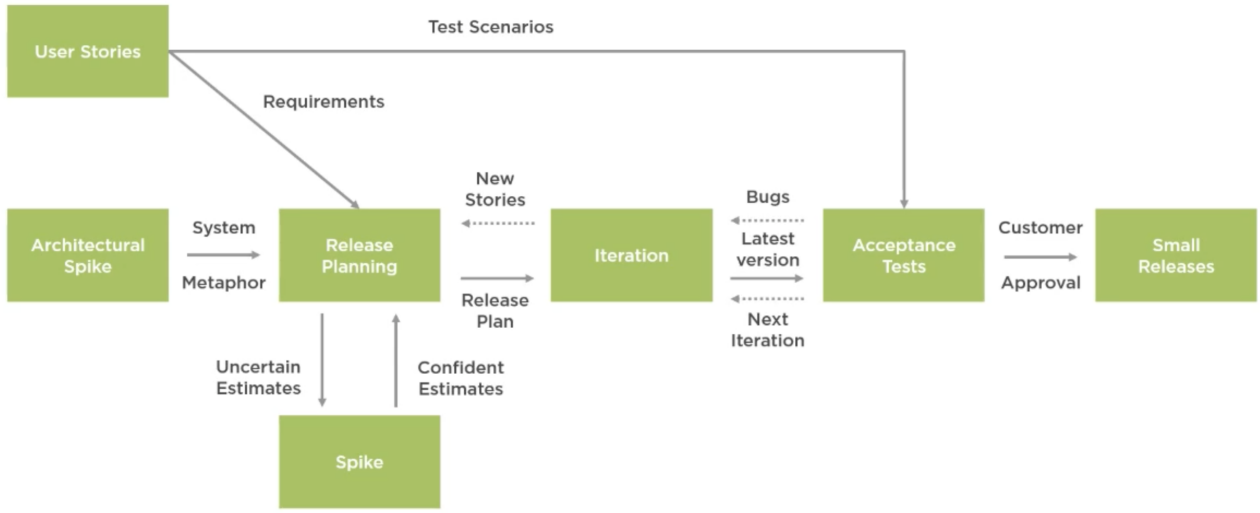
Coding

- Customer is always available
- Code written to agreed standards
- More common to use a tool to ensure code standards
- code the unit test first
- production code is pair programmed
- only one pair integrates code at a time

Testing

- All code has unit tests
- All code must pass all unit tests
- when a bug is found tests are created

XP diagram:



This is the Conscious Capitalism Credo:

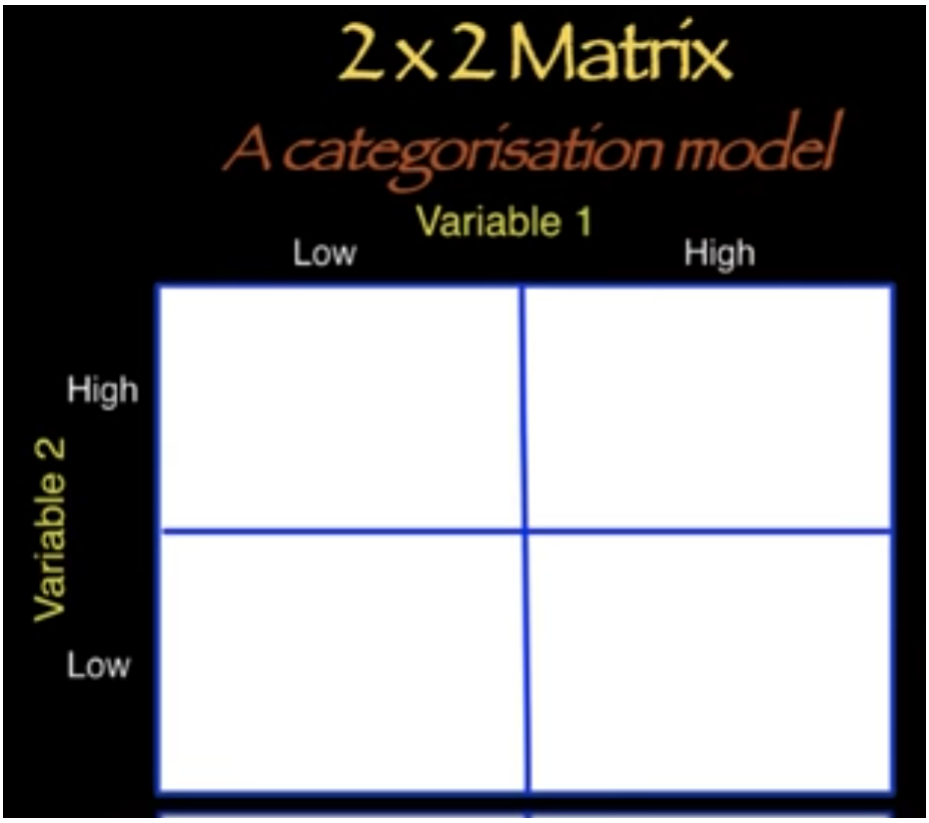
> “We believe that business is good because it creates value, it is ethical because it is based on voluntary exchange, it is noble because it can elevate our existence, and it is heroic because it lifts people out of poverty and creates prosperity. Free enterprise capitalism is the most powerful system for social cooperation and human progress ever conceived. It is one of the most compelling ideas we humans have ever had. But we can aspire to even more.
>
> Conscious Capitalism is a way of thinking about capitalism and business that better reflects where we are in the human journey, the state of our world today, and the innate potential of business to make a positive impact on the world. Conscious businesses are galvanized by higher purposes that serve, align, and integrate the interests of all their major stakeholders. Their higher state of consciousness makes visible to them the interdependencies that exist across all stakeholders, allowing them to

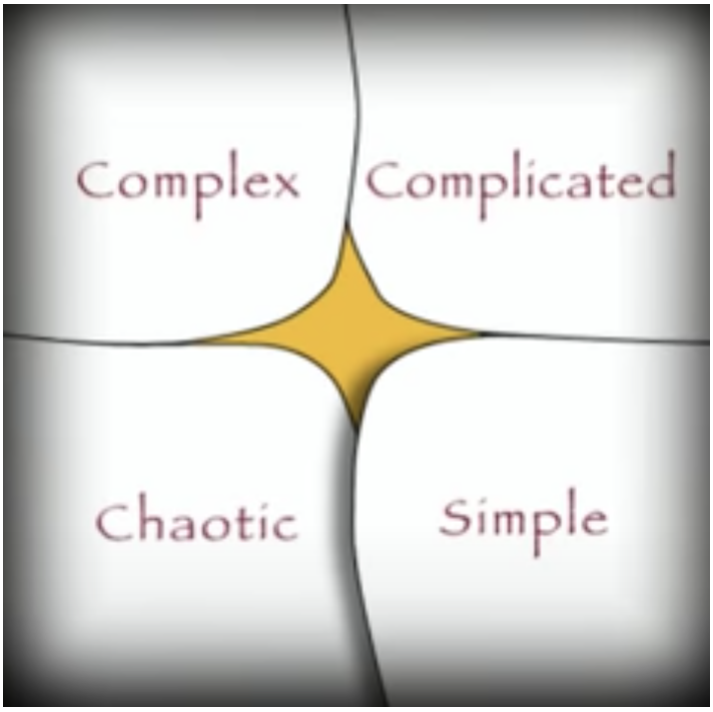
> discover and harvest synergies from situations that otherwise seem replete with trade-offs. They have conscious leaders who are driven by service to the company's purpose, all the people the business touches, and the planet we all share together. Conscious businesses have trusting, authentic, innovative and caring cultures that make working there a source of both personal growth and professional fulfillment. They endeavor to create financial, intellectual, social, cultural, emotional, spiritual, physical and ecological wealth for all their stakeholders.
>
> Conscious businesses will help evolve our world so that billions of people can flourish, leading lives infused with passion, purpose, love and creativity; a world of freedom, harmony, prosperity, and compassion."

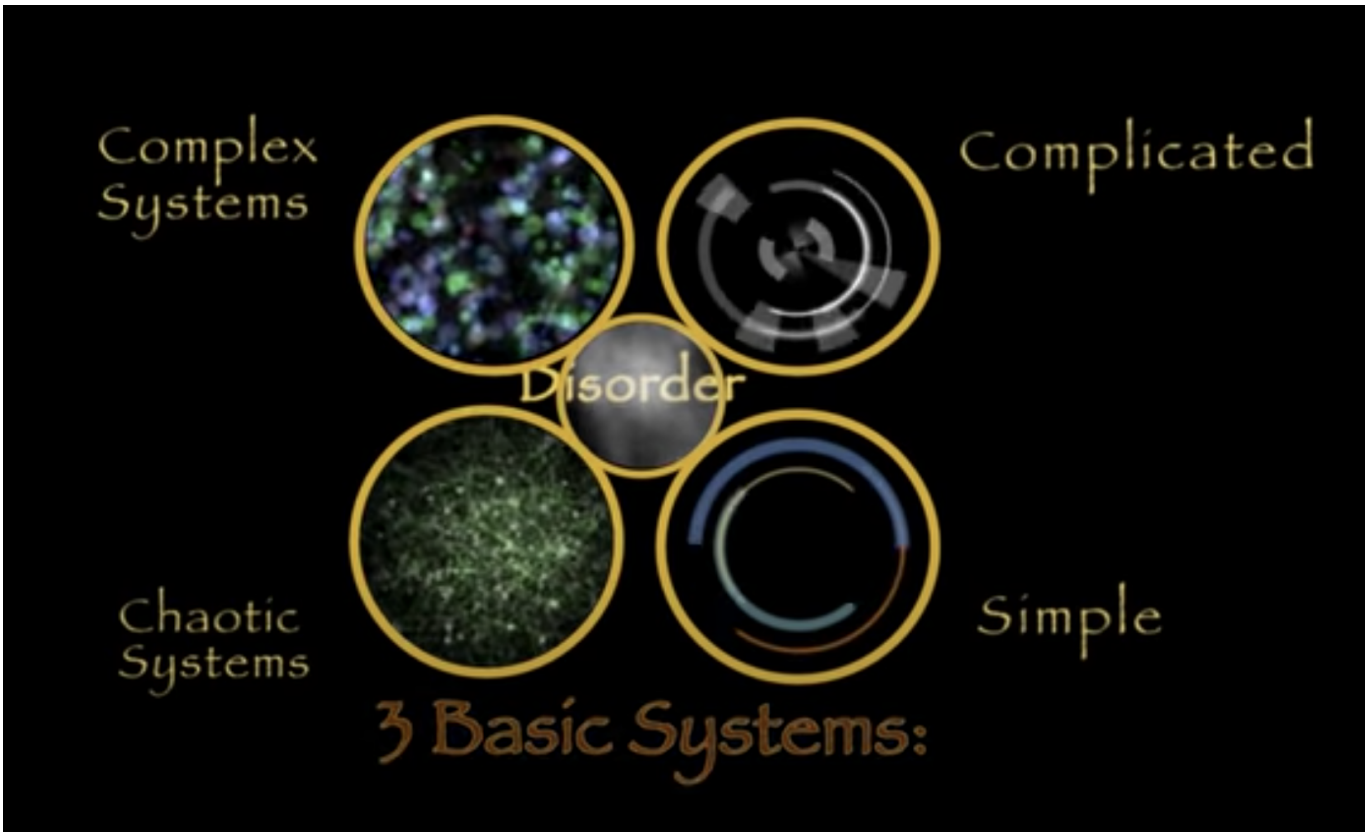
Cynefin Framework

Sense making model - data precedes the framework

- standard capitalist consultancy model:


2 x 2 Matrix
A categorisation model
Variable 1
Low
High
High
Variable 2
Low


Complex
Complicated
Chaotic
Simple



Simple domain:

- cause and effect relationships exist are predictable and repeatable
- sense - categorise - respond
- best practice

complicated domain

- relationship between cause and effect is not self evident
- good practice rather than best practice - there are several ways of doing things
- sense - analyse - respond
- good practice

Complex

- probe - sense - respond
- experiment when we have identified amplify and dampen strategies - not before
- emergent practice

Chaotic

- Act - sense - respond
- Novel Practice

The Simple zone overlaps as a cliff onto the chaotic zone.

Summary:

A decision-making framework that recognises the causal differences that exist between system types and proposes new approaches to decision making in complex social environments.

Scrum Guide:

https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

Developed and sustained by Scrum creators: Ken Schwaber and Jeff Sutherland

```
Scrum Theory
Scrum is founded on empirical process control theory, or empiricism.
Empiricism asserts that
knowledge comes from experience and making decisions based on what is
known. Scrum
employs an iterative, incremental approach to optimize predictability and
control risk.
Three pillars uphold every implementation of empirical process control:
transparency,
inspection, and adaptation.
```

```
Transparency
Significant aspects of the process must be visible to those responsible
for the outcome.
Transparency requires those aspects be defined by a common standard so
observers share a
common understanding of what is being seen.
For example
• A common language referring to the process must be shared by all
participants; and,
• Those performing the work and those inspecting the resulting increment
must share a
common definition of “Done”.
Inspection
Scrum users must frequently inspect Scrum artifacts and progress toward a
Sprint Goal to detect
undesirable variances. Their inspection should not be so frequent that
inspection gets in the way
of the work. Inspections are most beneficial when diligently performed by
skilled inspectors at
the point of work.
Adaptation
If an inspector determines that one or more aspects of a process deviate
outside acceptable
limits, and that the resulting product will be unacceptable, the process
or the material being
processed must be adjusted. An adjustment must be made as soon as possible
to minimize
further deviation.
Scrum prescribes four formal events for inspection and adaptation, as
described in the Scrum
```

```
Events section of this document:
• Sprint Planning
• Daily Scrum
• Sprint Review
• Sprint Retrospective
```

The Product Owner is one person, not a committee. The Product Owner may represent the desires of a committee in the Product Backlog, but those wanting to change a Product Backlog item's priority must address the Product Owner.

This was quite useful on Standup meetings:

https://www.planday.com/blog/daily-standup-meetings/

https://www.atlassian.com/agile/scrum/standups