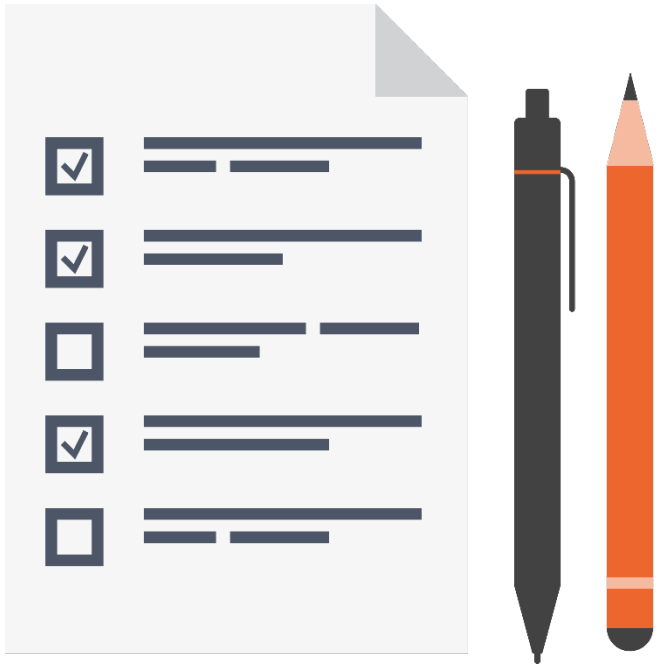
Extreme Programming (XP)

Stephen Haunts

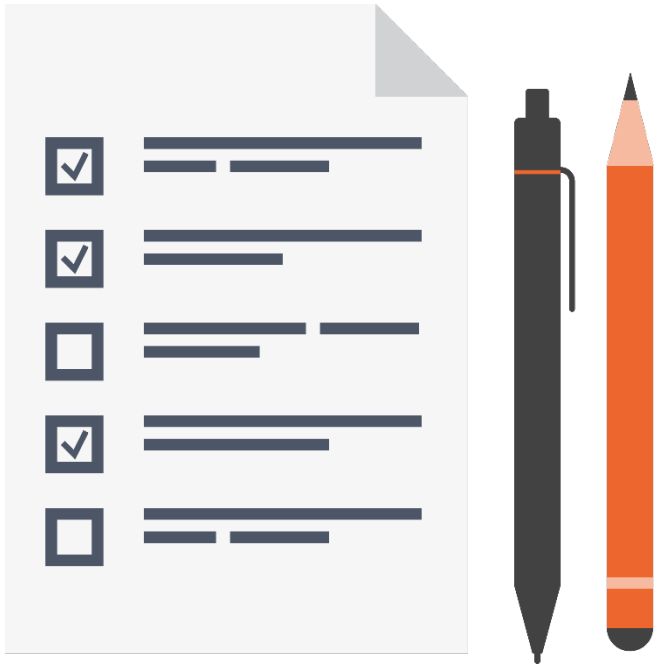
@stephenhaunts | www.stephenhaunts.com

Overview



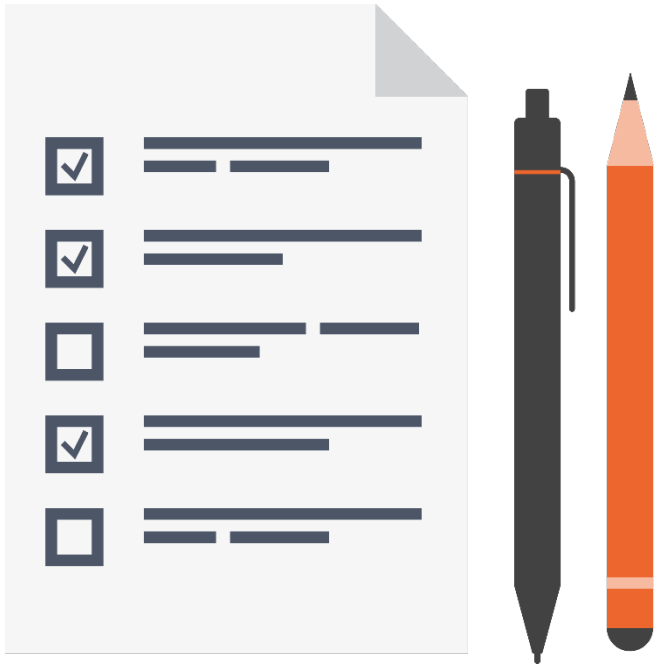
- History

Overview



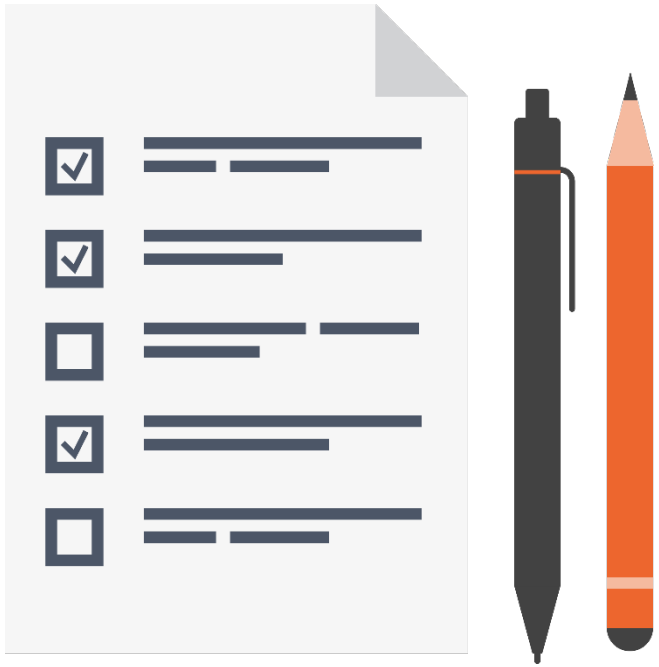
- History
- Overview

Overview



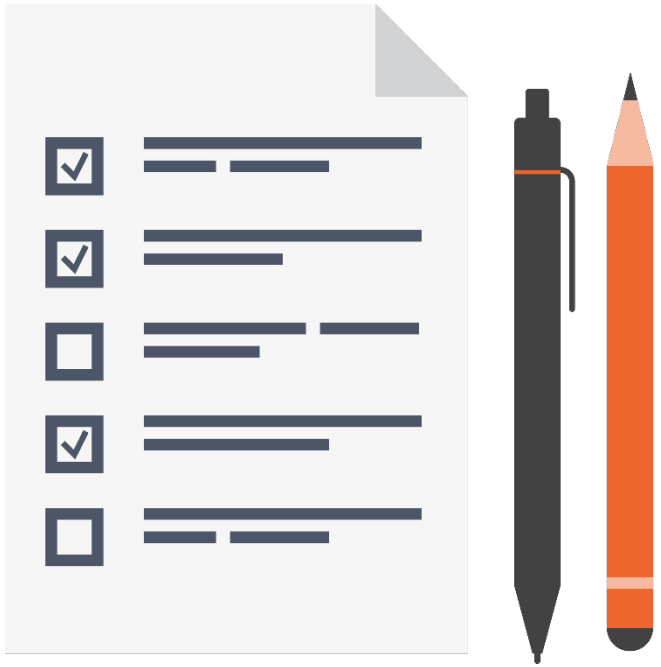
- History
- Overview
- Activities

Overview



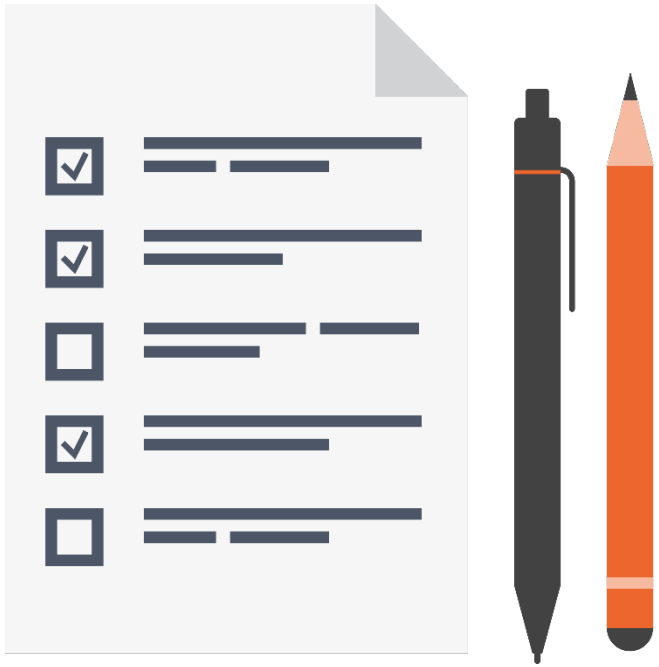
- History
- Overview
- Activities
- Values

Overview



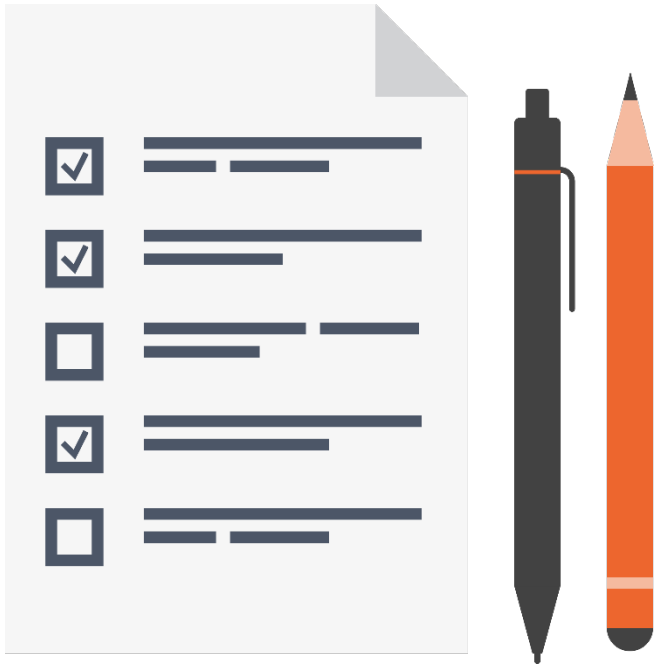
- History
- Overview
- Activities
- Values
- Principles

Overview



- History
- Overview
- Activities
- Values
- Principles
- Practices

Overview



- Rules
  - Planning
  - Managing
  - Design
  - Coding
  - Testing

Extreme Programming (XP)

Extreme programming (XP) is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements.

History of Extreme Programming

- Created by Kent Beck

History of Extreme Programming

- Created by Kent Beck
- C3 team focused on business value

History of Extreme Programming

- Created by Kent Beck
- C3 team focused on business value
- Working system delivered and refined with multiple smaller releases

Overview of Extreme Programming

- 4 x Activities

Overview of Extreme Programming

- 4 x Activities
- 5 x Values

Overview of Extreme Programming

- 4 x Activities
- 5 x Values
- 3 x Principles

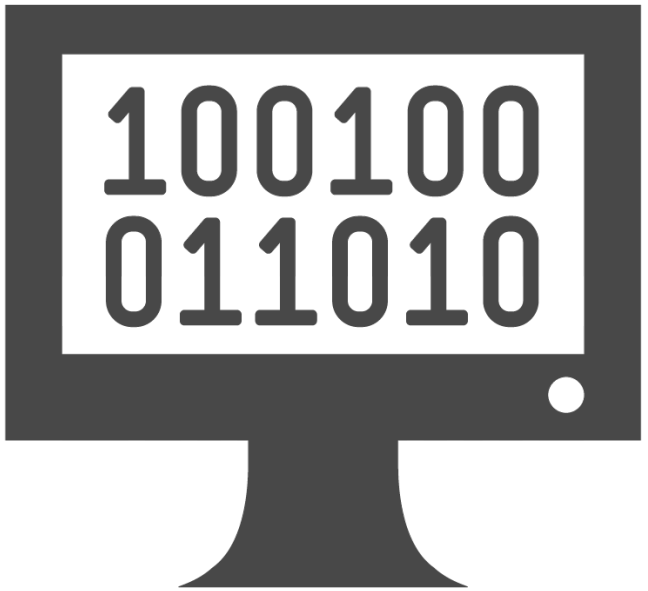
Overview of Extreme Programming

- 4 x Activities
- 5 x Values
- 3 x Principles
- 12 x Practices

Overview of Extreme Programming

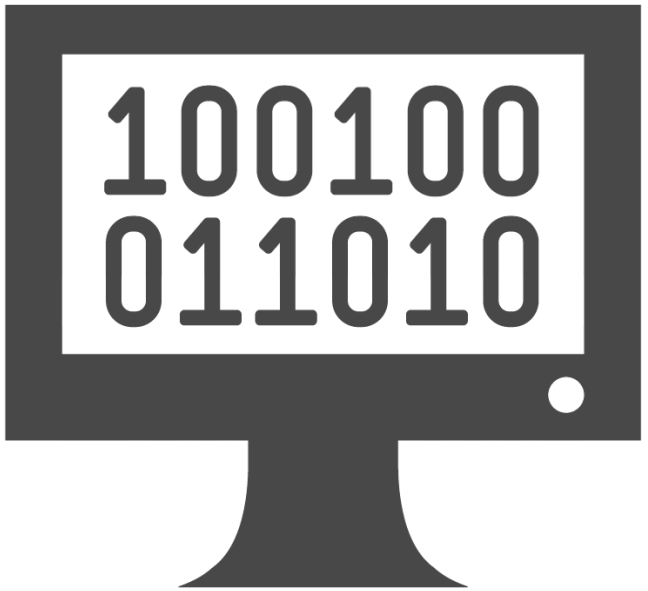
- 4 x Activities
- 5 x Values
- 3 x Principles
- 12 x Practices
- 29 x Rules

Extreme Programming Activities



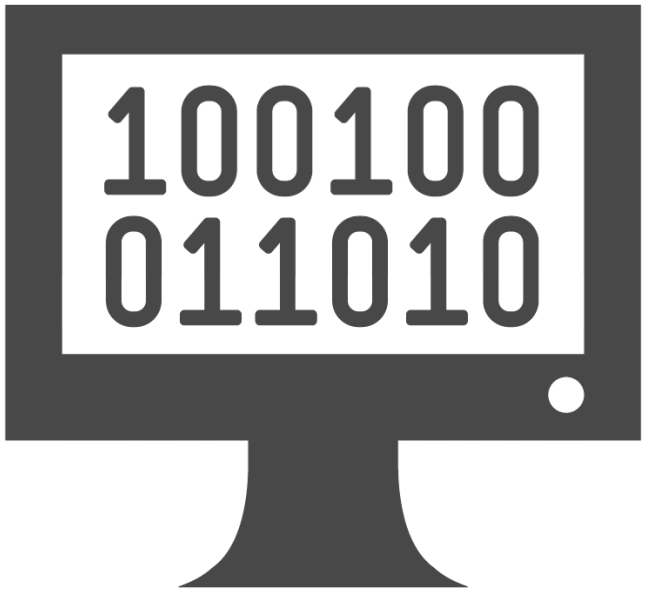
- Writing the application code

Extreme Programming Activities



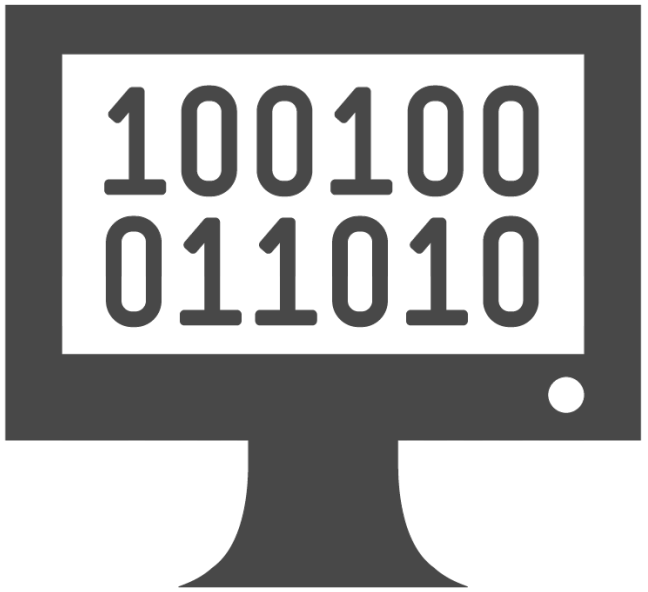
- Writing the application code
- Testing the system

Extreme Programming Activities



- Writing the application code
- Testing the system
- Listening to your customers and users

Extreme Programming Activities



- Writing the application code
- Testing the system
- Listening to your customers and users
- Designing your system to reduce coupling

Extreme Programming Values

- Communication is essential to any project

Extreme Programming Values

- Communication is essential to any project
- Build for simplicity

Extreme Programming Values

- Communication is essential to any project
- Build for simplicity
- Learning from feedback

Extreme Programming Values

- Communication is essential to any project
- Build for simplicity
- Learning from feedback
- Having courage

Extreme Programming Values

- Communication is essential to any project
- Build for simplicity
- Learning from feedback
- Having courage
- Having respect for the team and project

Extreme Programming Principles

Feedback

Extreme Programming Principles

Feedback

Assuming Simplicity

Extreme Programming Principles

Feedback | Assuming Simplicity | Embracing Change

Extreme Programming Practices

- Fine-scale feedback

Extreme Programming Practices

- Fine-scale feedback
- Continuous process

Extreme Programming Practices

- Fine-scale feedback
- Continuous process
- Shared understanding

Extreme Programming Practices

- Fine-scale feedback
- Continuous process
- Shared understanding
- Programmer welfare

Extreme Programming Practices

- Fine-scale feedback
  - Pair programming

Extreme Programming Practices

- Fine-scale feedback
  - Pair programming
  - Planning game

Extreme Programming Practices

- Fine-scale feedback
  - Pair programming
  - Planning game
  - Test-driven development

Extreme Programming Practices

- Fine-scale feedback
  - Pair programming
  - Planning game
  - Test-driven development
  - Whole team

Extreme Programming Practices

- Continuous process
  - Continuous integration

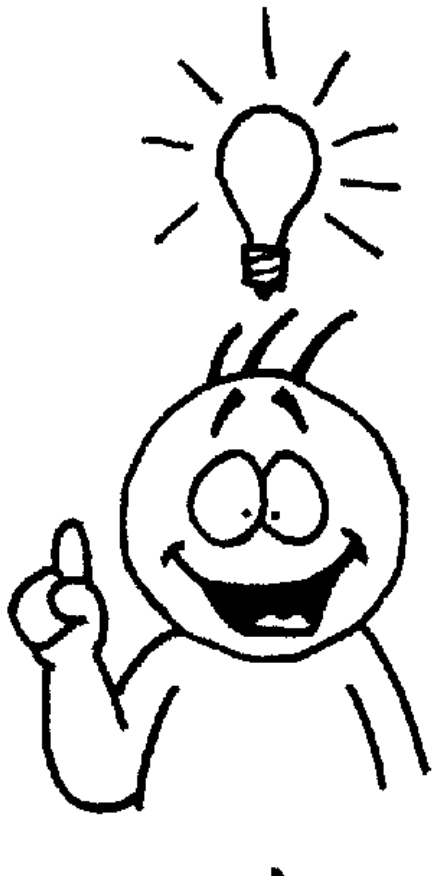
Extreme Programming Practices

- Continuous process
  - Continuous integration
  - Refactoring or design improvement

Extreme Programming Practices

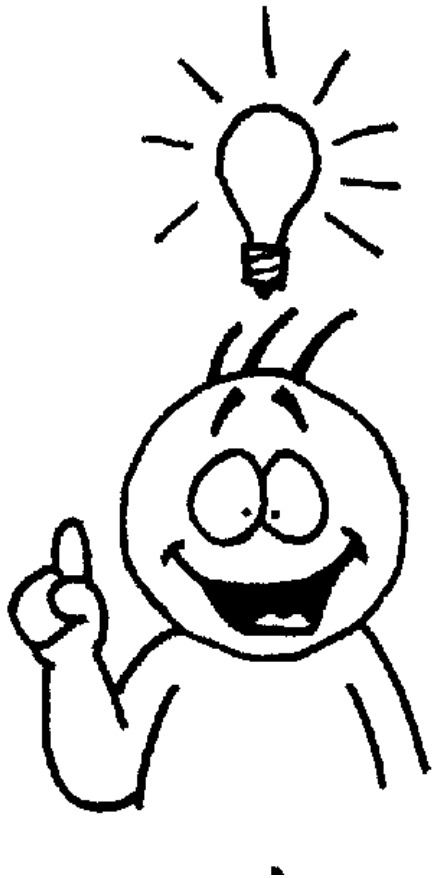
- Continuous process
  - Continuous integration
  - Refactoring or design improvement
  - Small releases

Extreme Programming Practices



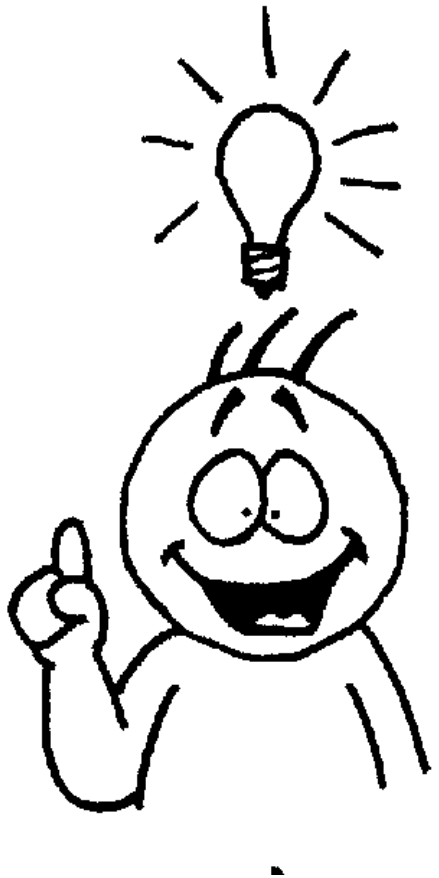
- Shared understanding
  - Coding standards

Extreme Programming Practices



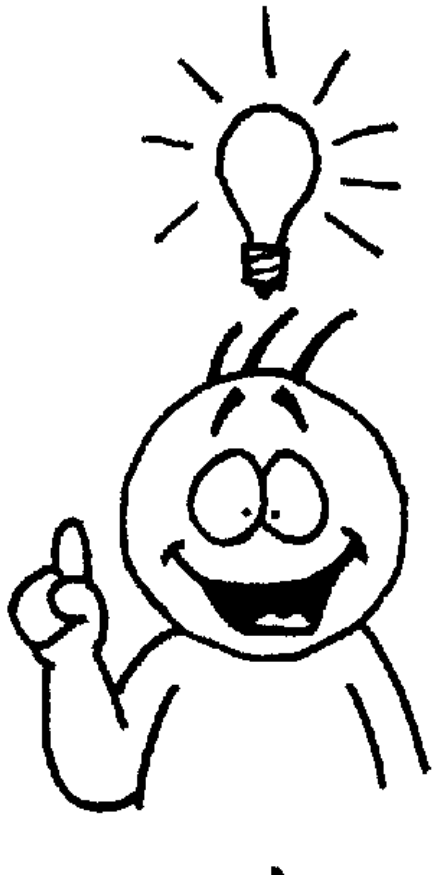
- Shared understanding
  - Coding standards
  - Collective code ownership

Extreme Programming Practices



- Shared understanding
  - Coding standards
  - Collective code ownership
  - Simple design

Extreme Programming Practices

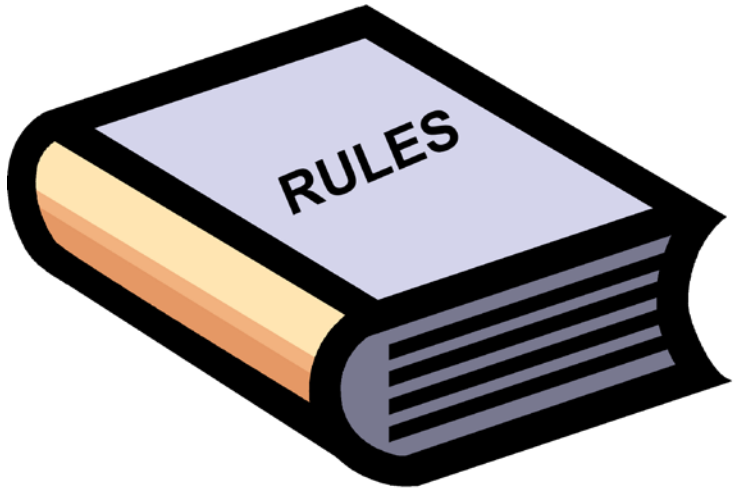


- Shared understanding
  - Coding standards
  - Collective code ownership
  - Simple design
  - System metaphor

Extreme Programming Practices

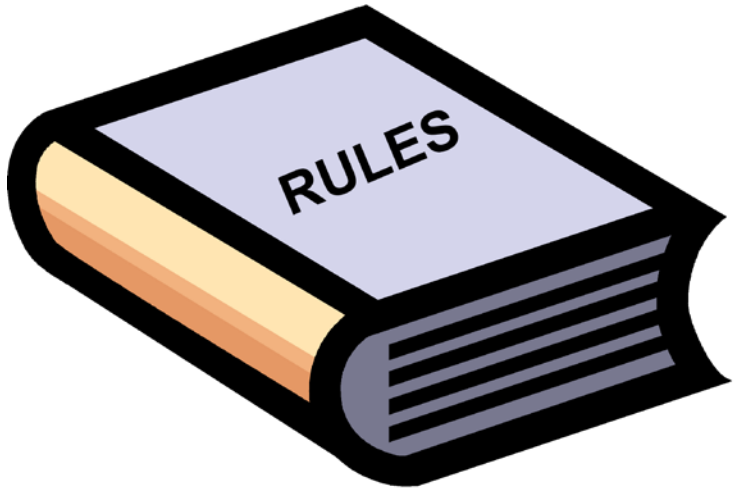
- Programmer welfare
  - Sustainable pace

Extreme Programming Rules



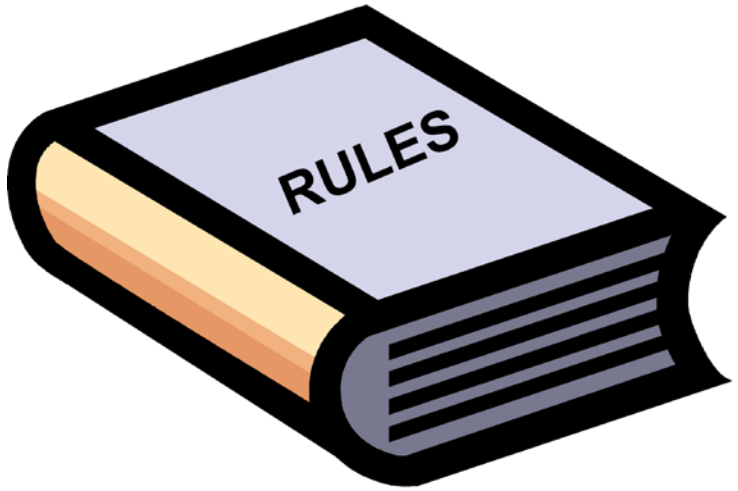
- Extreme programming has 29 rules split into 5 sections.
  - Planning

Extreme Programming Rules



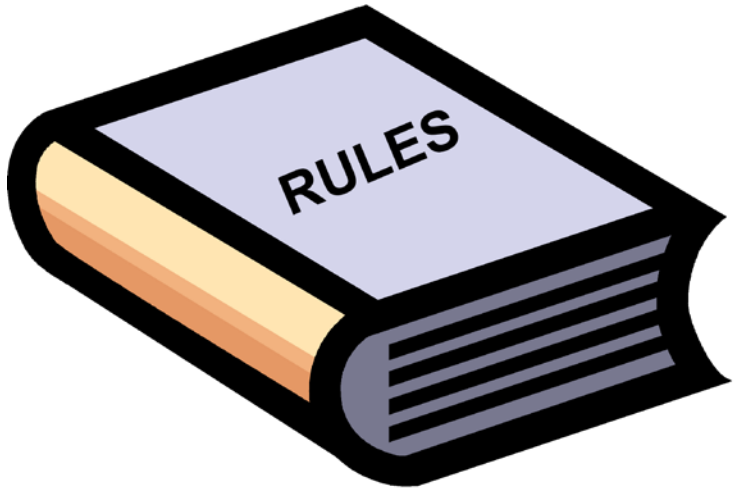
- Extreme programming has 29 rules split into 5 sections.
  - Planning
  - Managing

Extreme Programming Rules



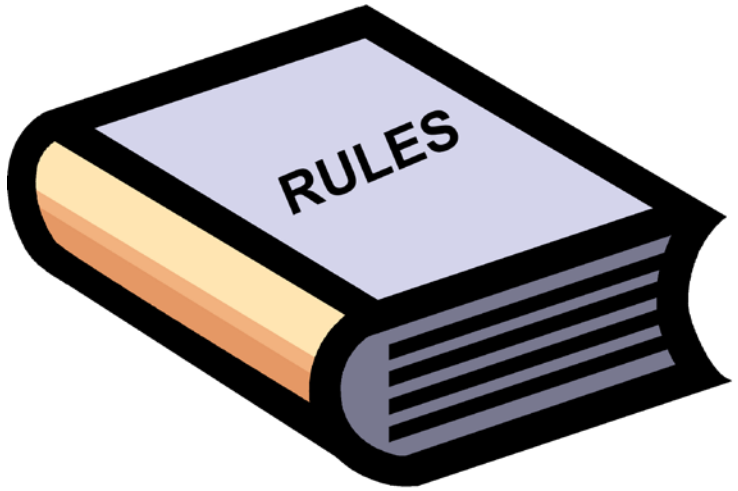
- Extreme programming has 29 rules split into 5 sections.
  - Planning
  - Managing
  - Designing

Extreme Programming Rules



- Extreme programming has 29 rules split into 5 sections.
  - Planning
  - Managing
  - Designing
  - Coding

Extreme Programming Rules



- Extreme programming has 29 rules split into 5 sections.
  - Planning
  - Managing
  - Designing
  - Coding
  - Testing

Planning

- User stories are written

Planning

- User stories are written
- Release planning

Planning

- User stories are written
- Release planning
- Make frequent small releases

Planning

- User stories are written
- Release planning
- Make frequent small releases
- Project divided into iterations

Planning

- User stories are written
- Release planning
- Make frequent small releases
- Project divided into iterations
- Iteration planning starts each iteration

Managing

- Give the team an open work space

Managing

- Give the team an open work space
- Set a sustainable pace

Managing

- Give the team an open work space
- Set a sustainable pace
- A stand-up meeting starts each day

Managing

- Give the team an open work space
- Set a sustainable pace
- A stand-up meeting starts each day
- The project velocity is measured

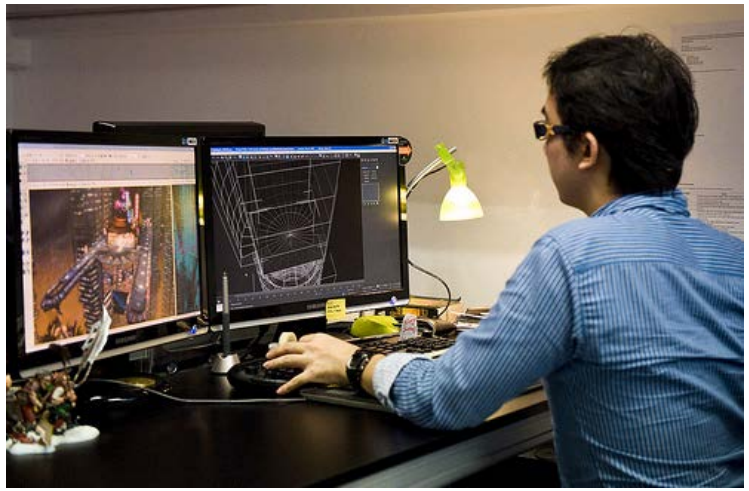
Managing

- Give the team an open work space
- Set a sustainable pace
- A stand-up meeting starts each day
- The project velocity is measured
- Move people around

Managing

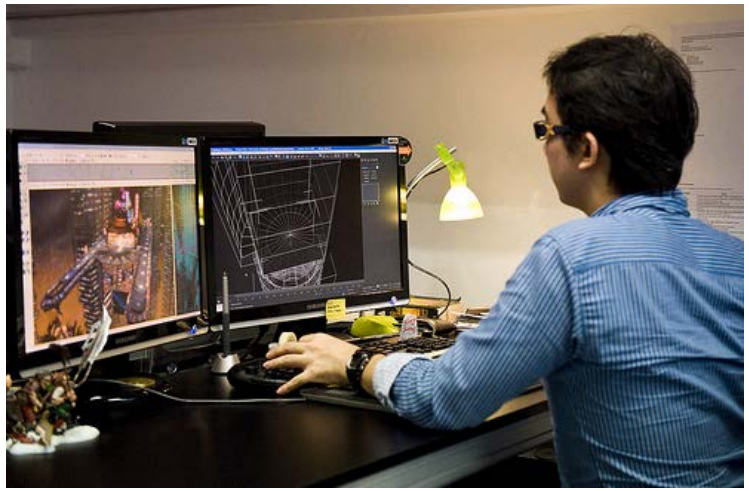
- Give the team an open work space
- Set a sustainable pace
- A stand-up meeting starts each day
- The project velocity is measured
- Move people around
- Fix XP when it breaks

Designing



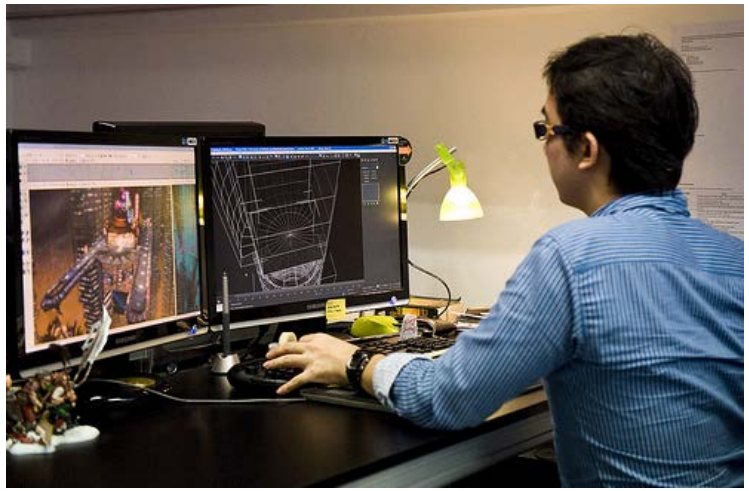
- Simplicity

Designing



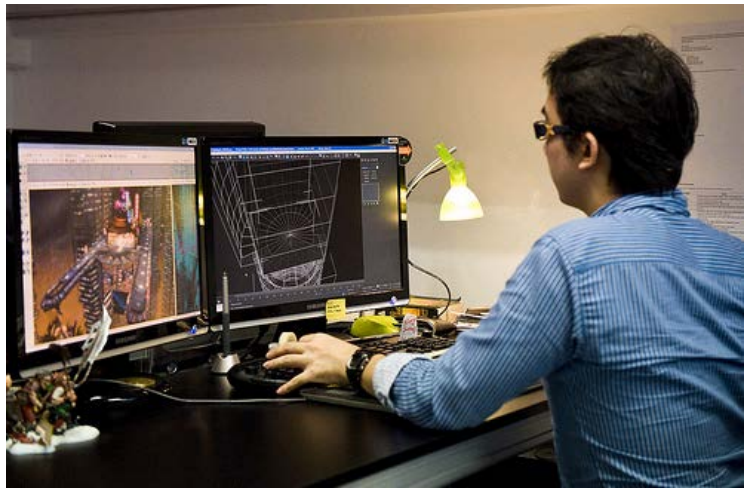
- Simplicity
- Choose a system metaphor

Designing



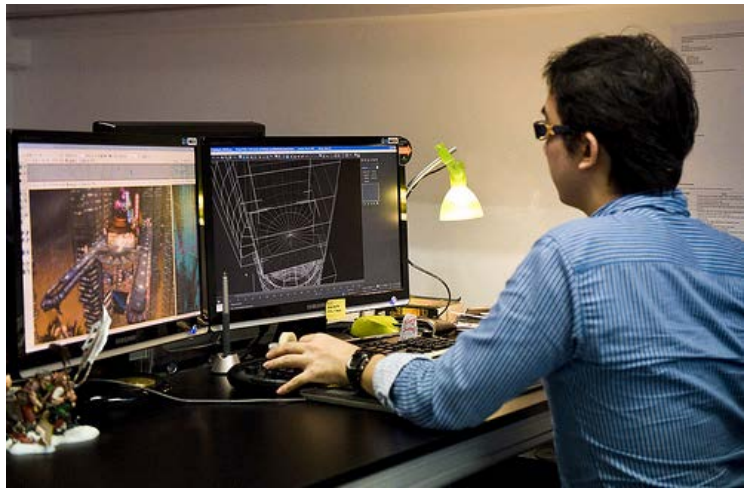
- Simplicity
- Choose a system metaphor
- Use CRC cards for design sessions

Designing



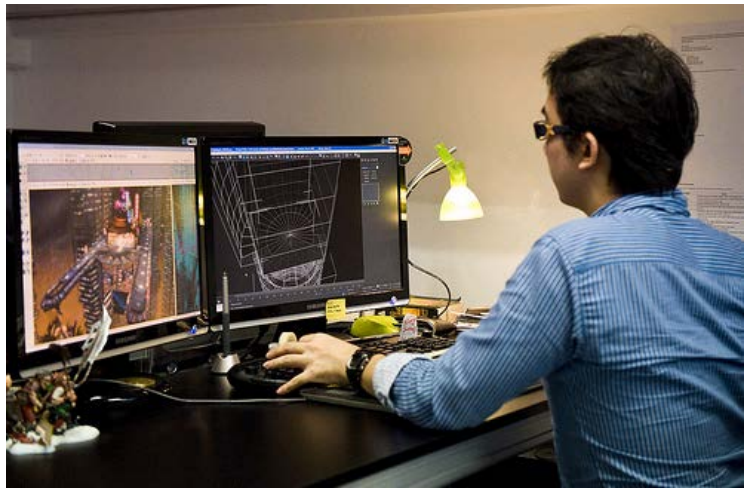
- Simplicity
- Choose a system metaphor
- Use CRC cards for design sessions
- Create spike solutions to reduce risk

Designing



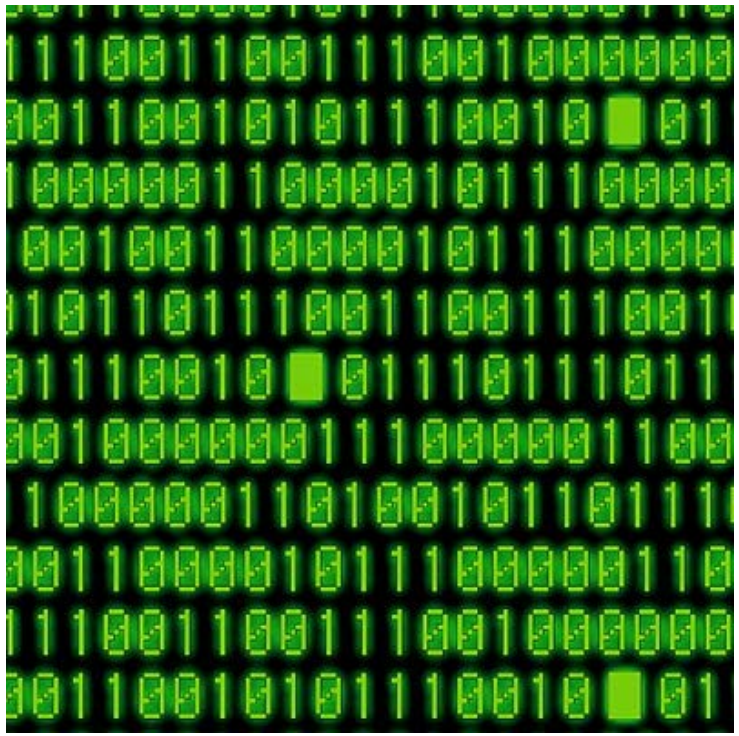
- Simplicity
- Choose a system metaphor
- Use CRC cards for design sessions
- Create spike solutions to reduce risk
- No functionality is added early

Designing



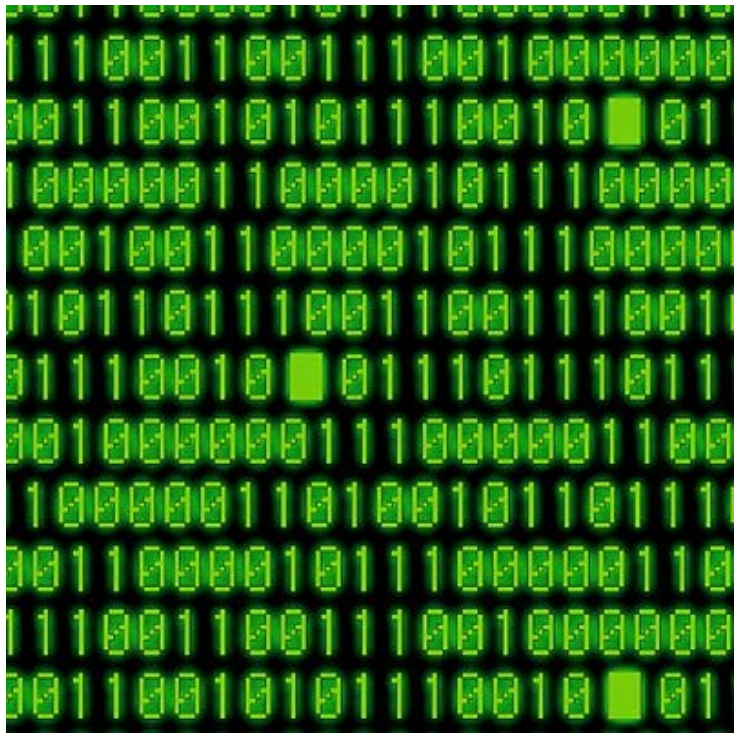
- Simplicity
- Choose a system metaphor
- Use CRC cards for design sessions
- Create spike solutions to reduce risk
- No functionality is added early
- Refactor whenever possible

Coding



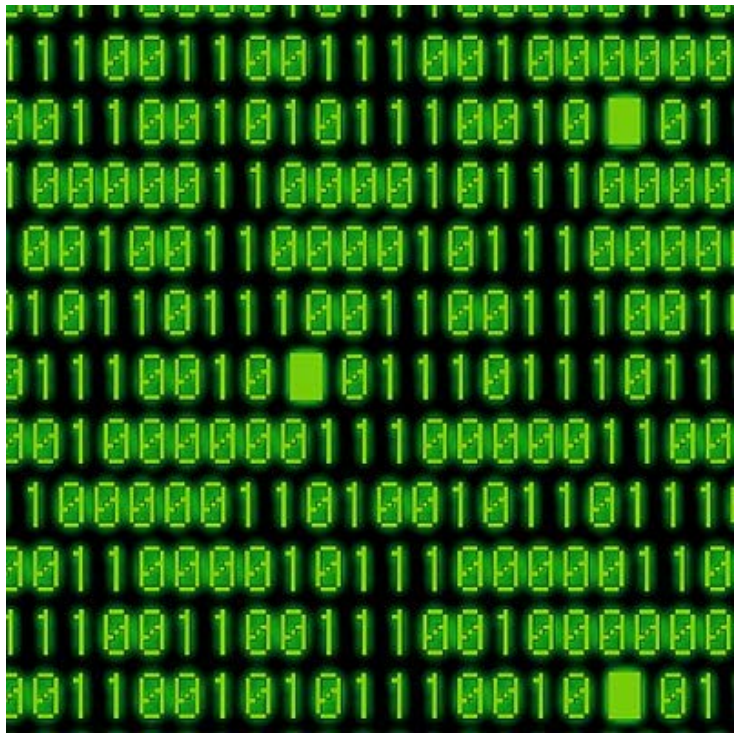
- Customer is always available

Coding



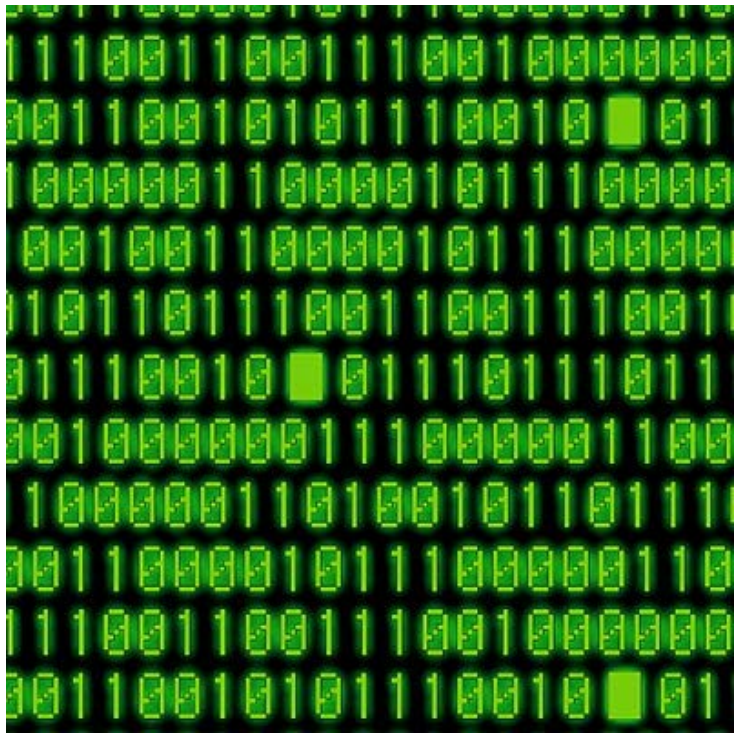
- Customer is always available
- Code written to agreed standards

Coding



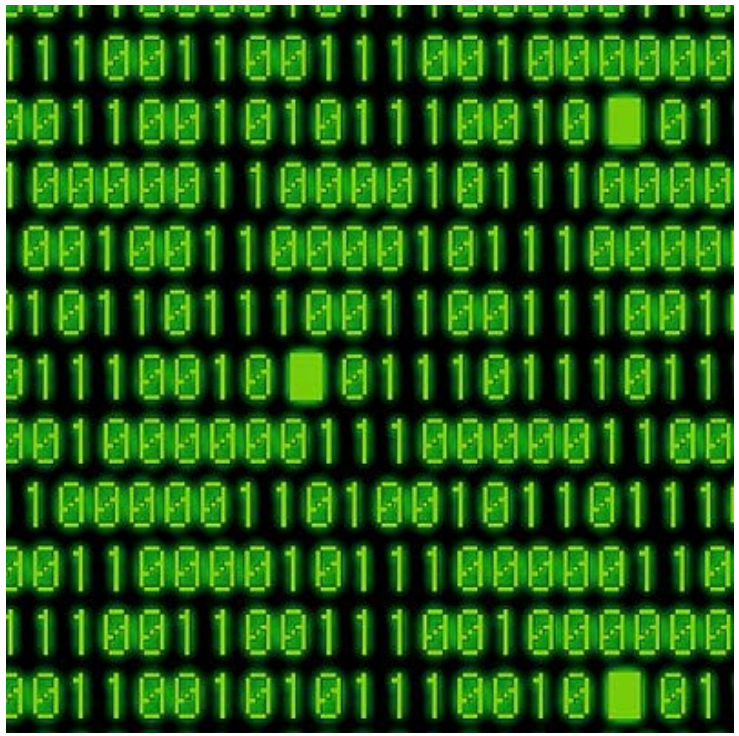
- Customer is always available
- Code written to agreed standards
- Code the unit test first

Coding



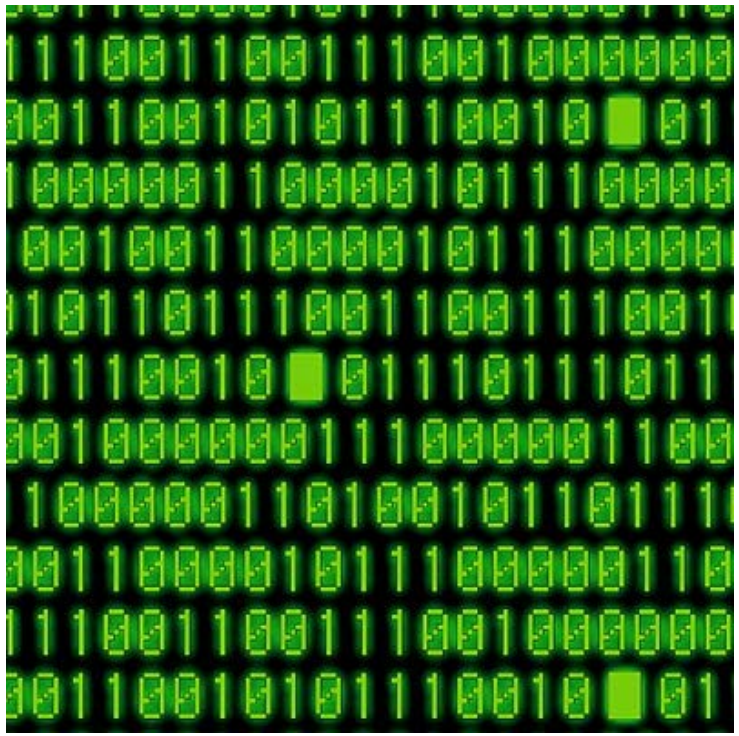
- Customer is always available
- Code written to agreed standards
- Code the unit test first
- Production code is pair programmed

Coding



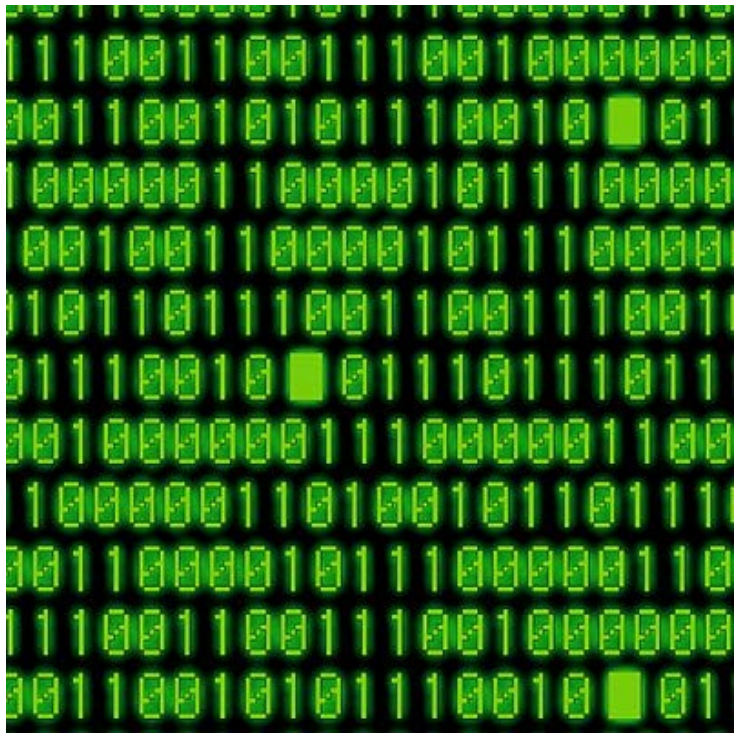
- Customer is always available
- Code written to agreed standards
- Code the unit test first
- Production code is pair programmed
- Only one pair integrates code at a time

Coding



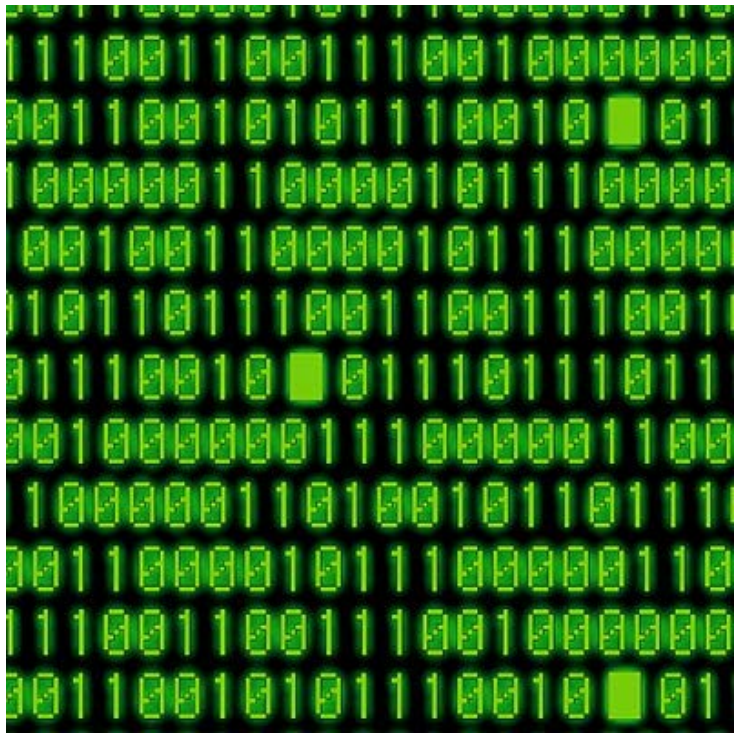
- Customer is always available
- Code written to agreed standards
- Code the unit test first
- Production code is pair programmed
- Only one pair integrates code at a time
- Integrate often

Coding



- Use a dedicated integration machine

Coding



- Use a dedicated integration machine
- Use collective ownership

Testing

- All code must have unit tests

Testing

- All code must have unit tests
- All code must pass all unit tests

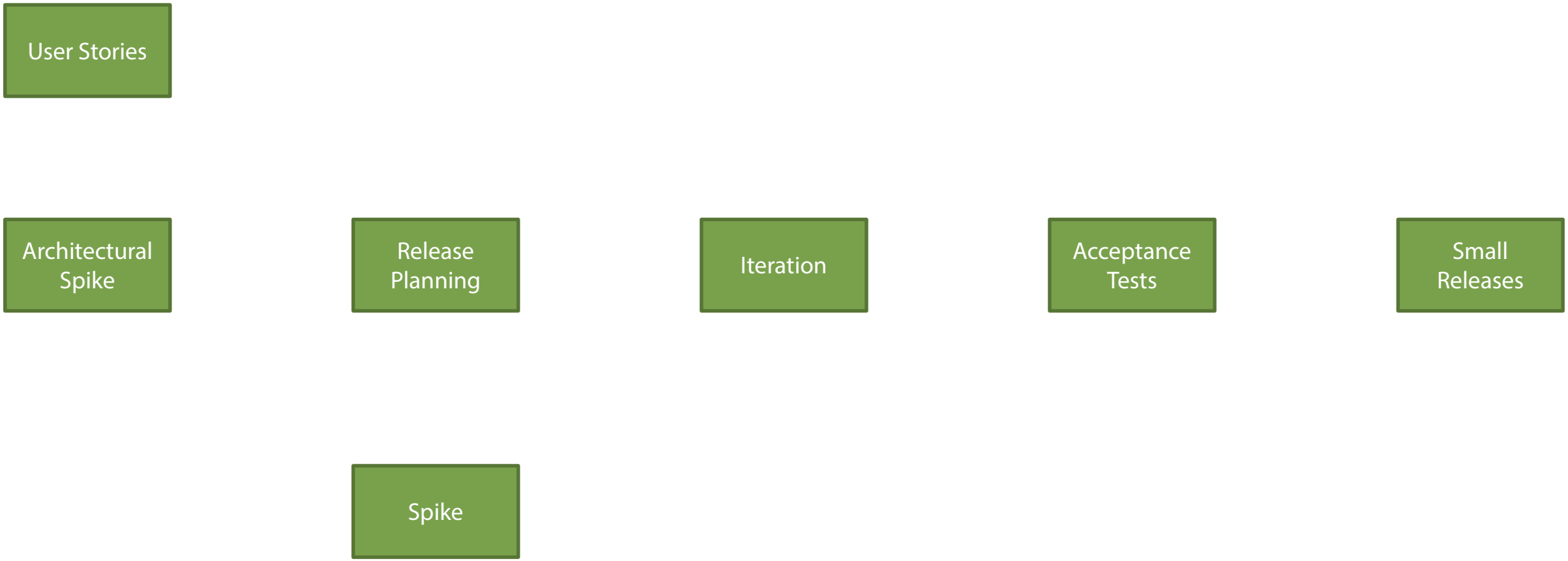
Testing

- All code must have unit tests
- All code must pass all unit tests
- When a bug is found tests are created

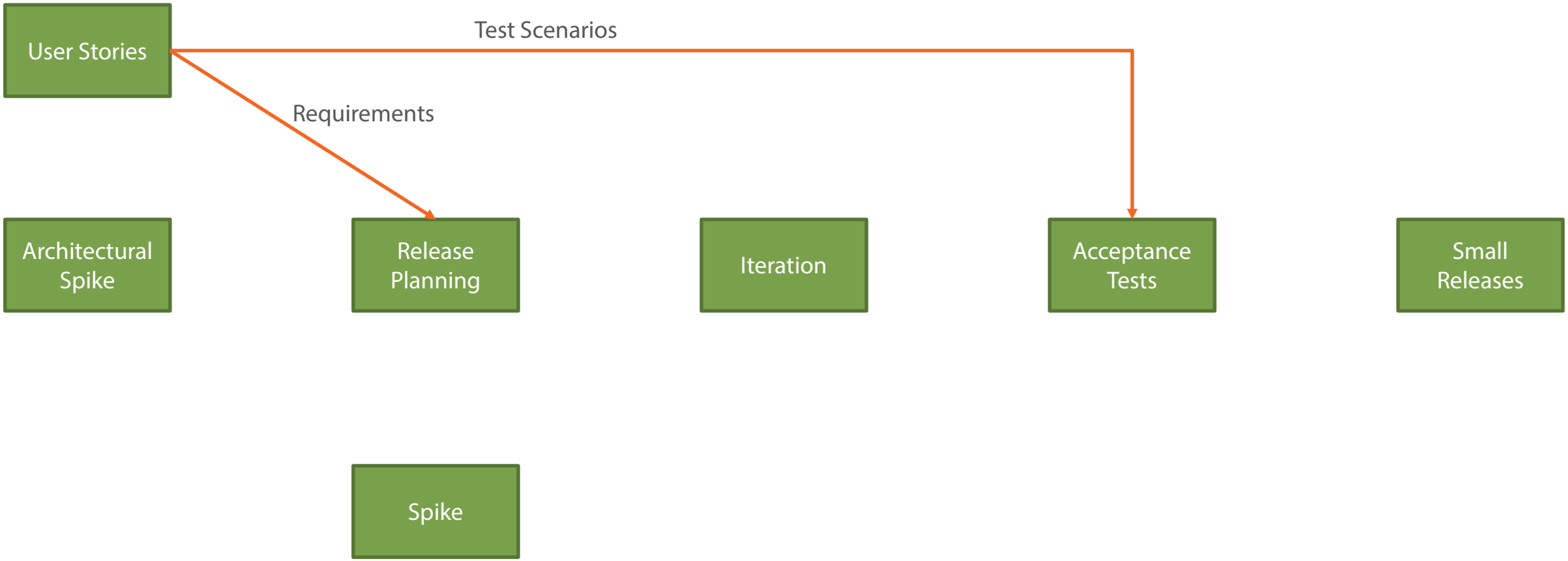
Testing

- All code must have unit tests
- All code must pass all unit tests
- When a bug is found tests are created
- Acceptance tests are run often and the score is published

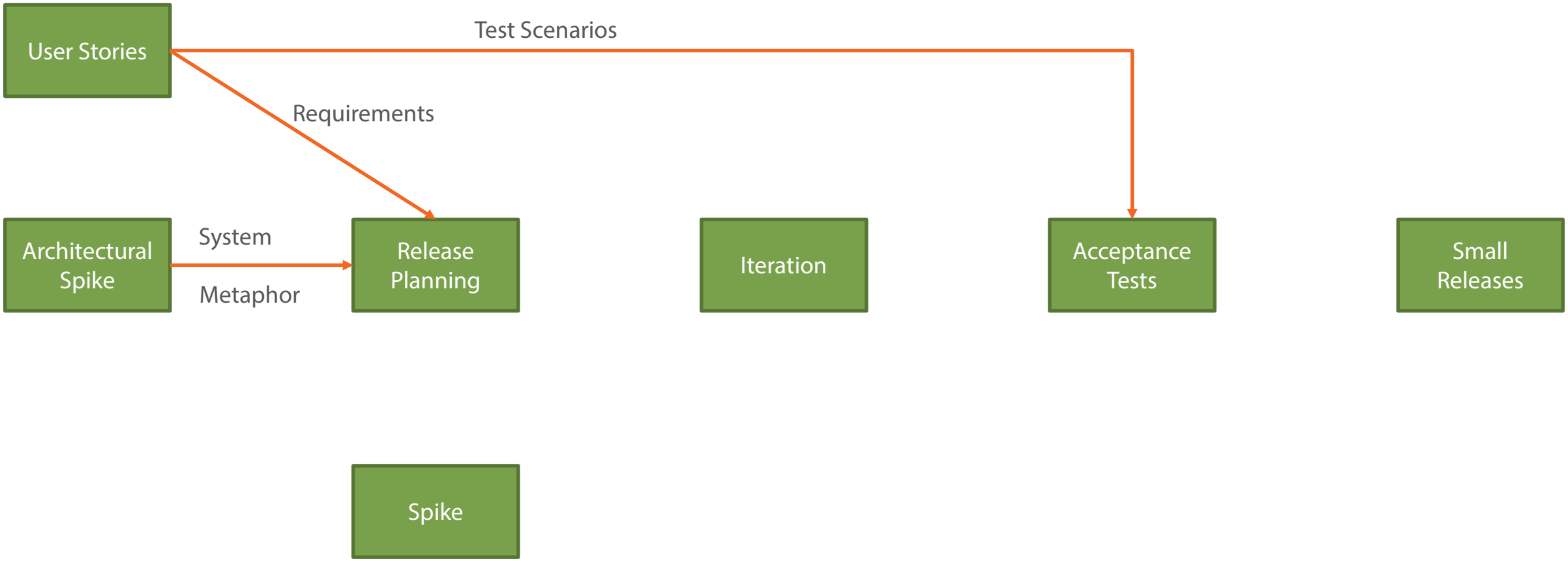
Extreme Programming Diagram



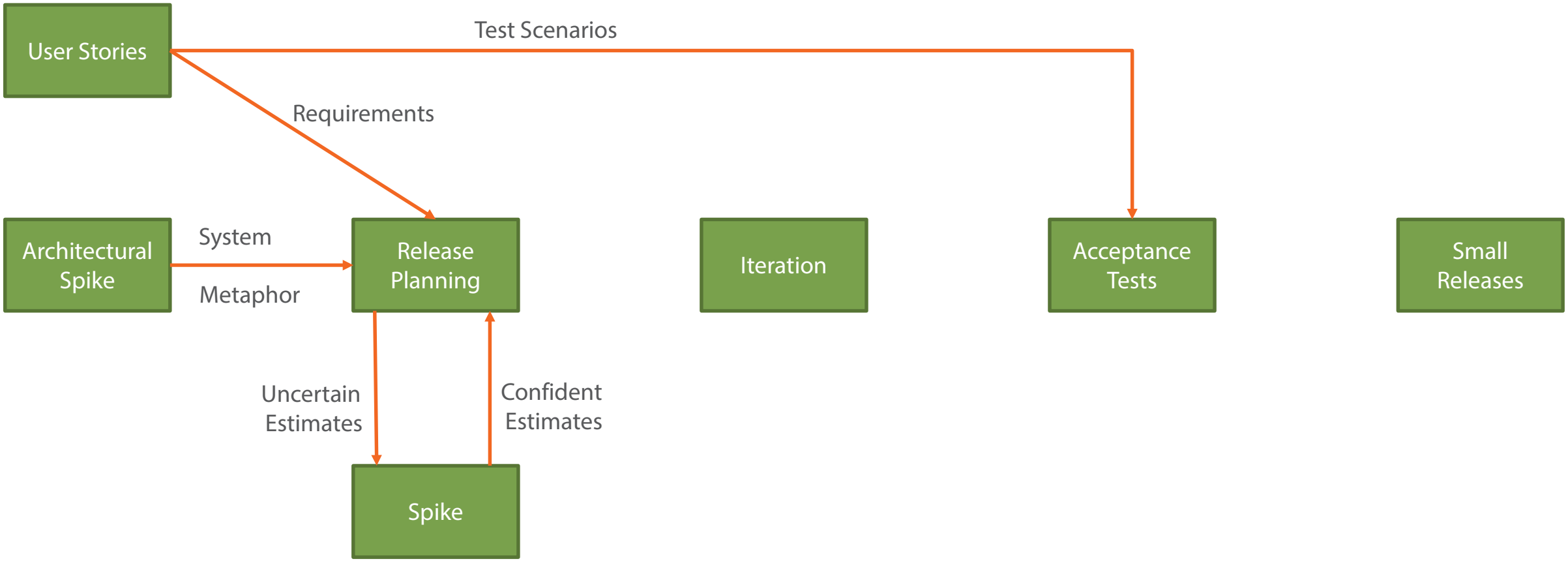
Extreme Programming Diagram



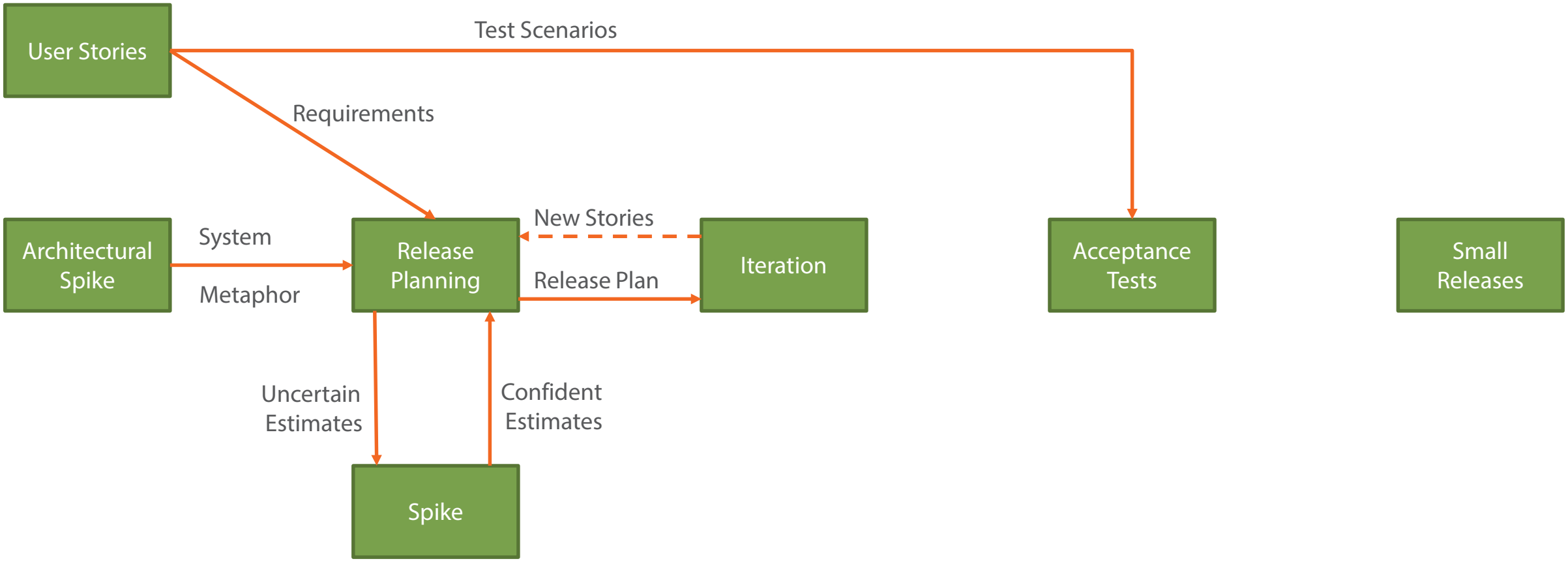
Extreme Programming Diagram



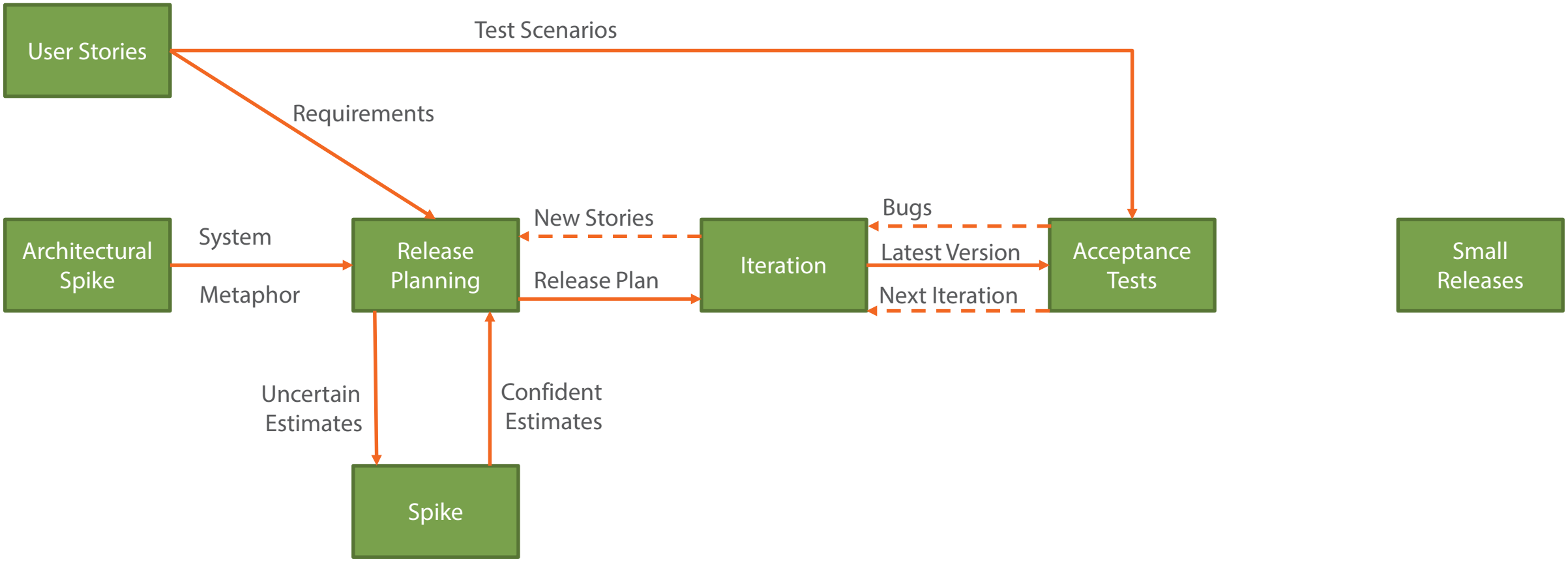
Extreme Programming Diagram



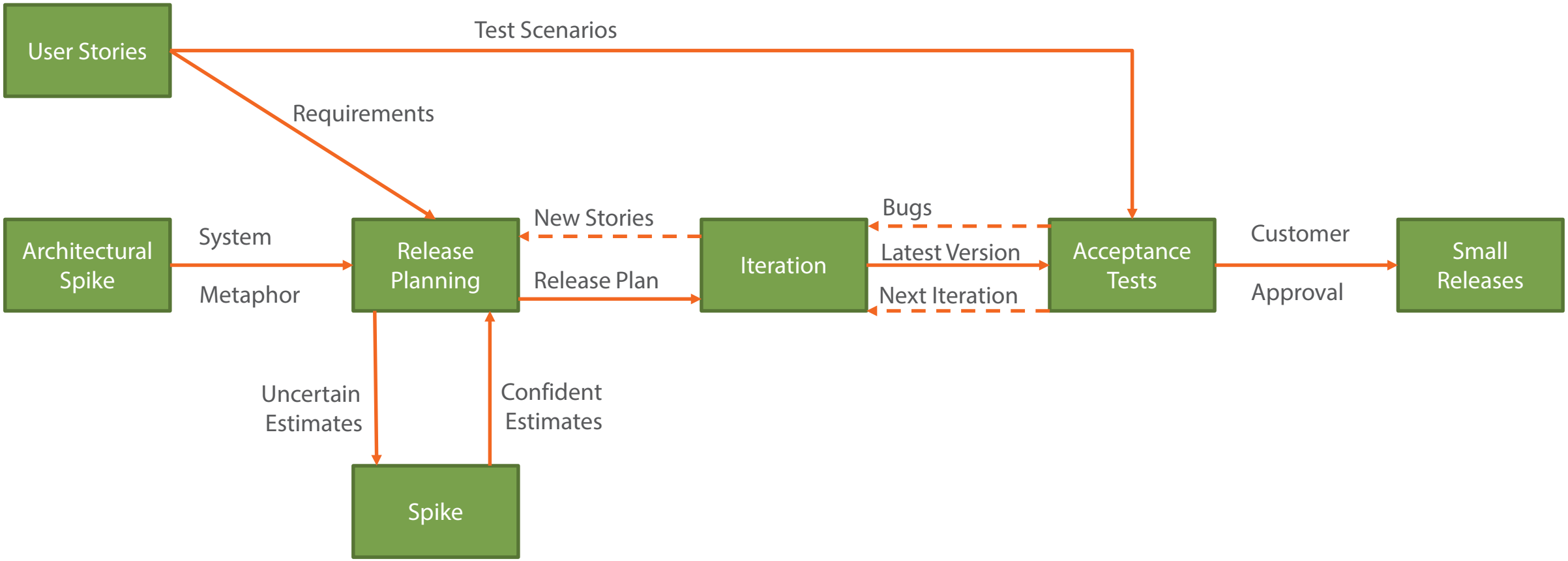
Extreme Programming Diagram



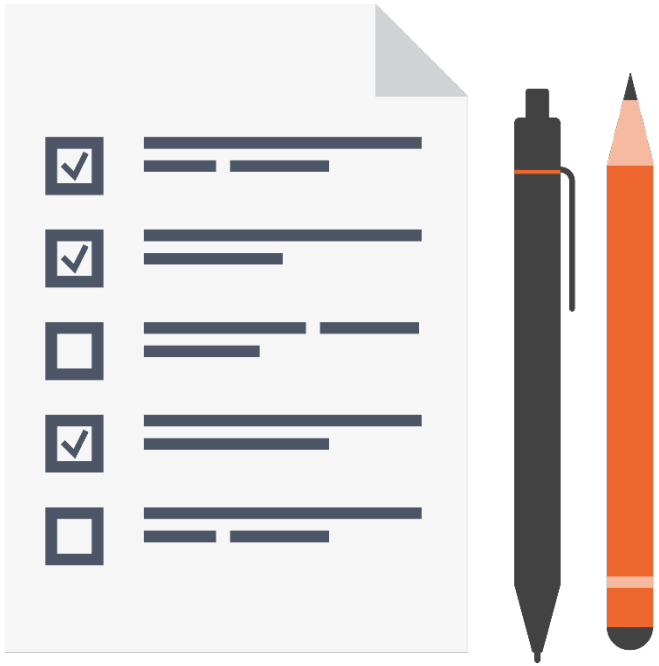
Extreme Programming Diagram



Extreme Programming Diagram

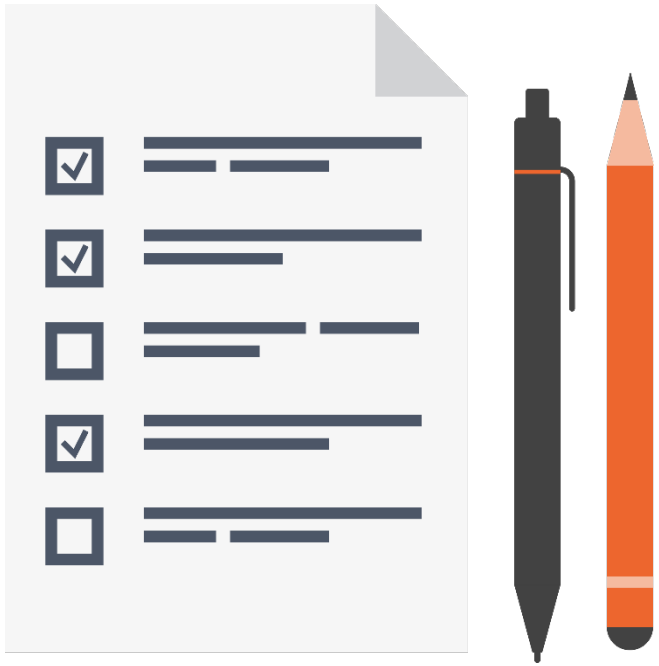


Summary



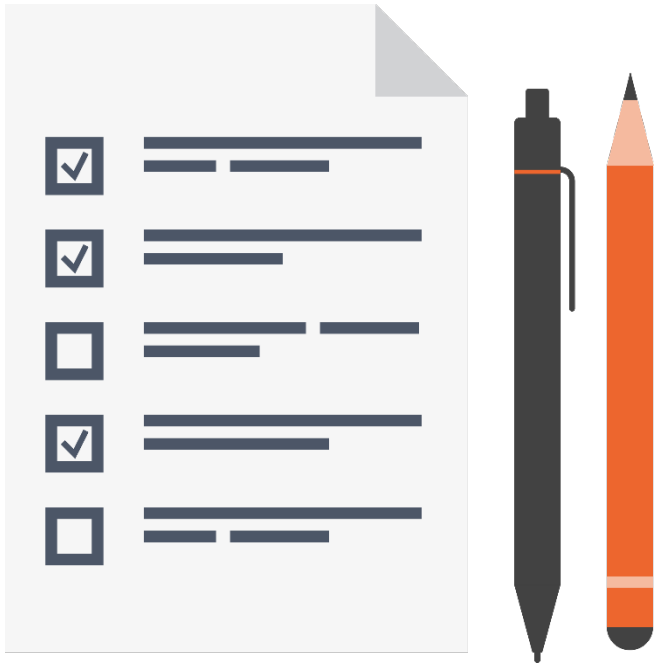
- History

Summary



- History
- Overview

Summary

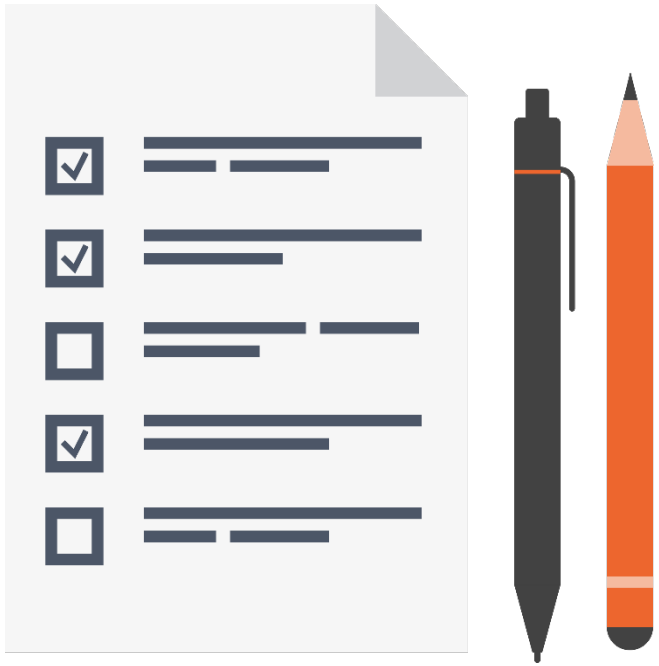


- History
- Overview
- Activities

Summary

- History
- Overview
- Activities
- Values

Summary

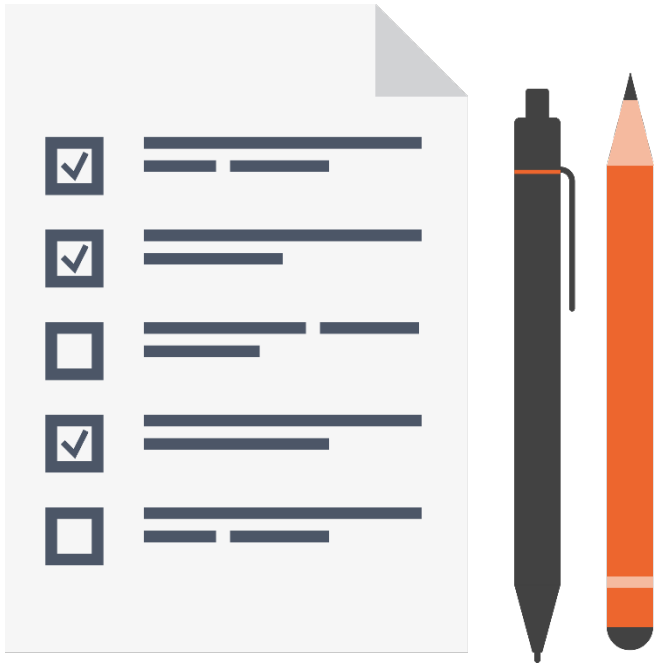


- History
- Overview
- Activities
- Values
- Principles

Summary

- History
- Overview
- Activities
- Values
- Principles
- Practices

Summary



- Rules