INF 502 – SOFTWARE DEVELOPMENT METHODOLOGIES

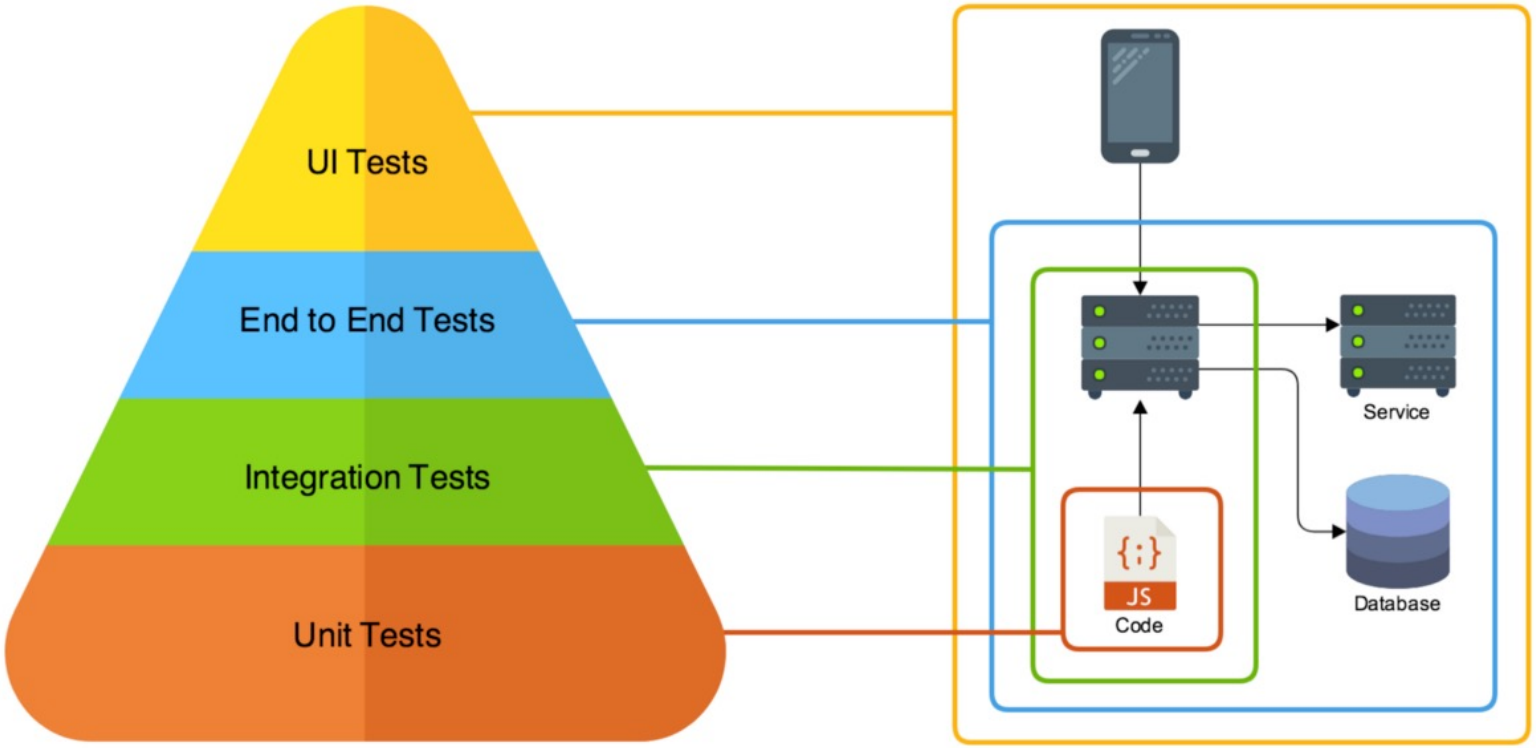
Testing

Tests

The software matches the expected requirements

The software has minimal errors and potential breaking points

This course will focus on unit tests



Unit Tests

First level of software testing

The smallest testable parts of a software are tested

Validate each unit of the software

Each test unit must be fully independent

Unit Tests: Why?

Check if pieces are working after you change the system

- Unit testing increases confidence in changing/ maintaining code

Point out defects while developing

Enforces more reusable code → you need to go modular

Debugging made easier

Helps you find where the issue is

unittest module for Python

- Import the module:
  - import unittest
- Import the file/function(s) you are testing:
  - import <function from file>
- Create a class that extends Testcase
  - class myTest(unittest.TestCase):
    - ...
- Convert the tests into methods using assertions

Assertions

Methods that evaluate the outcome of a function

assertEqual()- Tests that the two arguments are equal in value.

assertNotEqual()- Tests that the two arguments are unequal in value.

assertTrue()- Tests that the argument has a Boolean value of True.

assertFalse()- Tests that the argument has a Boolean value of False.

assertIs()- Tests that the arguments evaluate to the same object.

assertIsNot()- Tests that the arguments do not evaluate to the same object.

assertIsNone()- Tests that the argument evaluates to none.

assertIsNotNone()- Tests that the argument does not evaluate to none.

assertIn()- Tests that the first argument is in the second.

assertNotIn()- Tests that the first argument is not in the second.

assertIsInstance()- Tests that the first argument (object) is an instance of the second (class).

assertRaises()- Tests that Python raises an exception when we call the callable with positional/ keyword arguments we also passed to this method.

More here: https://kapeli.com/cheat_sheets/Python_unittest_Assertions.docset/Contents/Resources/Documents/index

Unit Tests: Basic!

```python
import unittest
class TestSum(unittest.TestCase):

    def test_sum(self):
        self.assertEqual(sum([1, 2, 3]), 6, "Should be 6")

    def test_sum_tuple(self):
        self.assertEqual(sum((1, 2, 2)), 6, "Should be 6")

if __name__ == '__main__':
    unittest.main()
```

```
.F
======================================================================
FAIL: test_sum_tuple (__main__.TestSum)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "basicTest.py", line 8, in test_sum_tuple
    self.assertEqual(sum((1, 2, 2)), 6, "Should be 6")
AssertionError: 5 != 6 : Should be 6

----------------------------------------------------------------------
Ran 2 tests in 0.000s

FAILED (failures=1)
```

Unit Tests: Basic!

```
def is_prime(number):
        for element in range(number):
                if (number % element == 0):
                        return False
        return True


def print_next_prime(number):
        index = number
        while True:
                index += 1
                if is_prime(index):
                        print(index)
```

The smallest unit is the **is_prime** function

(**print_next_time** uses it)

Test: is 5 prime?

Source: https://jeffknupp.com/blog/2013/12/09/improve-your-python-understanding-unit-testing/

Unit Tests: Basic!

```python
def is_prime(number):
    for element in range(number):
        if (number % element == 0):
            return False
    return True

def print_next_prime(number):
    index = number
    while True:
        index += 1
        if is_prime(index):
            print(index)
```

```python
import unittest
from prime import is_prime

class PrimesTestCase(unittest.TestCase):
    def test_is_five_prime(self):
        self.assertTrue(is_prime(5))

if __name__ == '__main__':
        unittest.main()
```

```
======================================================================
ERROR: test_is_five_prime (__main__.PrimesTestCase)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "PrimesTestCase.py", line 6, in test_is_five_prime
    self.assertTrue(is_prime(5))
  File "/Users/igorsteinmacher/Downloads/prime.py", line 3, in is_prime
    if (number % element == 0):
ZeroDivisionError: integer division or modulo by zero

----------------------------------------------------------------------
Ran 1 test in 0.001s
```

Unit Tests: Fixing…

```python
def is_prime(number):
    for element in range(1, number):
        if (number % element == 0):
            return False
    return True

def print_next_prime(number):
    index = number
    while True:
        index += 1
        if is_prime(index):
            print(index)
```

```python
import unittest
from prime import is_prime

class PrimesTestCase(unittest.TestCase):
    def test_is_five_prime(self):
        self.assertTrue(is_prime(5))

if __name__ == '__main__':
    unittest.main()
```

```
F
======================================================================
FAIL: test_is_five_prime (__main__.PrimesTestCase)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "PrimesTestCase.py", line 6, in test_is_five_prime
    self.assertTrue(is_prime(5))
AssertionError: False is not true

----------------------------------------------------------------------
Ran 1 test in 0.001s
```

Unit Tests: Fixing…

```python
def is_prime(number):
    for element in range(2, number):
        if (number % element == 0):
            return False
    return True

def print_next_prime(number):
    index = number
    while True:
        index += 1
        if is_prime(index):
            print(index)
```

```python
import unittest
from prime import is_prime

class PrimesTestCase(unittest.TestCase):
    def test_is_five_prime(self):
        self.assertTrue(is_prime(5))

if __name__ == '__main__':
    unittest.main()
```

```
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
```

What to test?

- Known cases (like we've done)

```python
def is_prime(number):
    for element in range(2, number):
        if (number % element == 0):
            return False
    return True

def print_next_prime(number):
    index = number
    while True:
        index += 1
        if is_prime(index):
            print(index)
```

```python
import unittest
from prime import is_prime

class PrimesTestCase(unittest.TestCase):
    def test_is_five_prime(self):
        self.assertTrue(is_prime(5))

    def test_is_four_prime(self):
        self.assertFalse(is_prime(4))

if __name__ == '__main__':
    unittest.main()
```

```
..
----------------------------------------------------------------------
Ran 2 tests in 0.000s

OK
```

What to test?

- Edge cases (ZERO!!!)

```
..F
======================================================================
FAIL: test_is_zero_not_prime (__main__.PrimesTestCase)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "PrimesTestCase_5.py", line 10, in test_is_zero_not_prime
    self.assertFalse(is_prime(0))
AssertionError: True is not false

----------------------------------------------------------------------
Ran 3 tests in 0.001s

FAILED (failures=1)
```

```python
import unittest
from prime import is_prime

class PrimesTestCase(unittest.TestCase):
    def test_is_five_prime(self):
        self.assertTrue(is_prime(5))
    def test_is_four_prime(self):
        self.assertFalse(is_prime(4))

    def test_is_zero_not_prime(self):
        self.assertFalse(is_prime(0))
```

What to test?

```python
def is_prime(number):
    if number in (0, 1):
        return False
    for element in range(2, number):
        if (number % element == 0):
            return False
    return True

def print_next_prime(number):
        index = number
        while True:
                index += 1
                if is_prime(index):
                        print(index)
```

```python
import unittest
from prime import is_prime

class PrimesTestCase(unittest.TestCase):
    def test_is_five_prime(self):
        self.assertTrue(is_prime(5))
    def test_is_four_prime(self):
        self.assertFalse(is_prime(4))

    def test_is_zero_not_prime(self):
        self.assertFalse(is_prime(0))
```

```
...
----------------------------------------------
Ran 3 tests in 0.000s

OK
```

What to test?

```python
def is_prime(number):
    if number in (0, 1):
        return False
    for element in range(2, number):
        if (number % element == 0):
            return False
    return True

def print_next_prime(number):
        index = number
        while True:
                index += 1
                if is_prime(index):
                        print(index)
```

```python
import unittest
from prime import is_prime

class PrimesTestCase(unittest.TestCase):
    ...
    def test_negative_number(self):
        self.assertFalse(is_prime(-1))
        self.assertFalse(is_prime(-2))
        self.assertFalse(is_prime(-3))
        ...
```

```
...F
======================================================================
FAIL: test_negative_number (__main__.PrimesTestCase)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "PrimesTestCase_6.py", line 12, in test_negative_number
    self.assertFalse(is_prime(-1))
AssertionError: True is not false

----------------------------------------------------------------------
Ran 4 tests in 0.000s
```

What to test?

```python
def is_prime(number):
    if number <= 1:
        return False

    for element in range(2, number):
        if (number % element == 0):
            return False
    return True


def print_next_prime(number):
        index = number
        while True:
                index += 1
                if is_prime(index):
                        print(index)
```

```python
import unittest
from prime import is_prime

class PrimesTestCase(unittest.TestCase):
    ...
    def test_negative_number(self):
        self.assertFalse(is_prime(-1))
        self.assertFalse(is_prime(-2))
        self.assertFalse(is_prime(-3))
        ...
```

```
....
----------------------------------------------------------------------
Ran 4 tests in 0.000s

OK
```

What to test?

```python
def is_prime(number):
    if number <= 1:
        return False

    for element in range(2, number):
        if (number % element == 0):
            return False
    return True


def print_next_prime(number):
        index = number
        while True:
                index += 1
                if is_prime(index):
                        print(index)
```

```python
import unittest
from prime import is_prime

class PrimesTestCase(unittest.TestCase):
    ...
    def test_TypeError(self):
        self.assertRaises(TypeError, is_prime, "a")
```

```
.....
----------------------------------------------------------------------
Ran 5 tests in 0.000s

OK
```

Agile Manifesto

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

Principles

- Customer satisfaction by rapid delivery of useful software
- Welcome changing requirements, even late in development
- Working software is delivered frequently
- Working software is the primary measure of progress
- Sustainable development, able to maintain a constant pace
- Close, daily co-operation between business people and developers
- Face-to-face conversation is the best form of communication
- Projects are built around motivated individuals who should be trusted
- Continuous attention to technical excellence and good design
- Simplicity – the art of maximizing the amount of work not done
- Self-organizing teams
- Team regularly reflects on how to become more effective, then adjusts

Methods

- eXtreme Programming (XP)
- Scrum
- Kanban
- Feature-Driven Development (FDD)
- Dynamic Systems Development Method (DSDM)
- Crystal
- Lean

The XP

Software is continuously delivered in intervals called ***sprints***

Principles

- Planning game
- Small releases
- Customer acceptance tests
- Simple design
- Pair programming
- Test-driven development
- Refactoring
- Continuous integration
- Collective code ownership
- Coding standards
- Metaphor
- Sustainable pace

Scrum

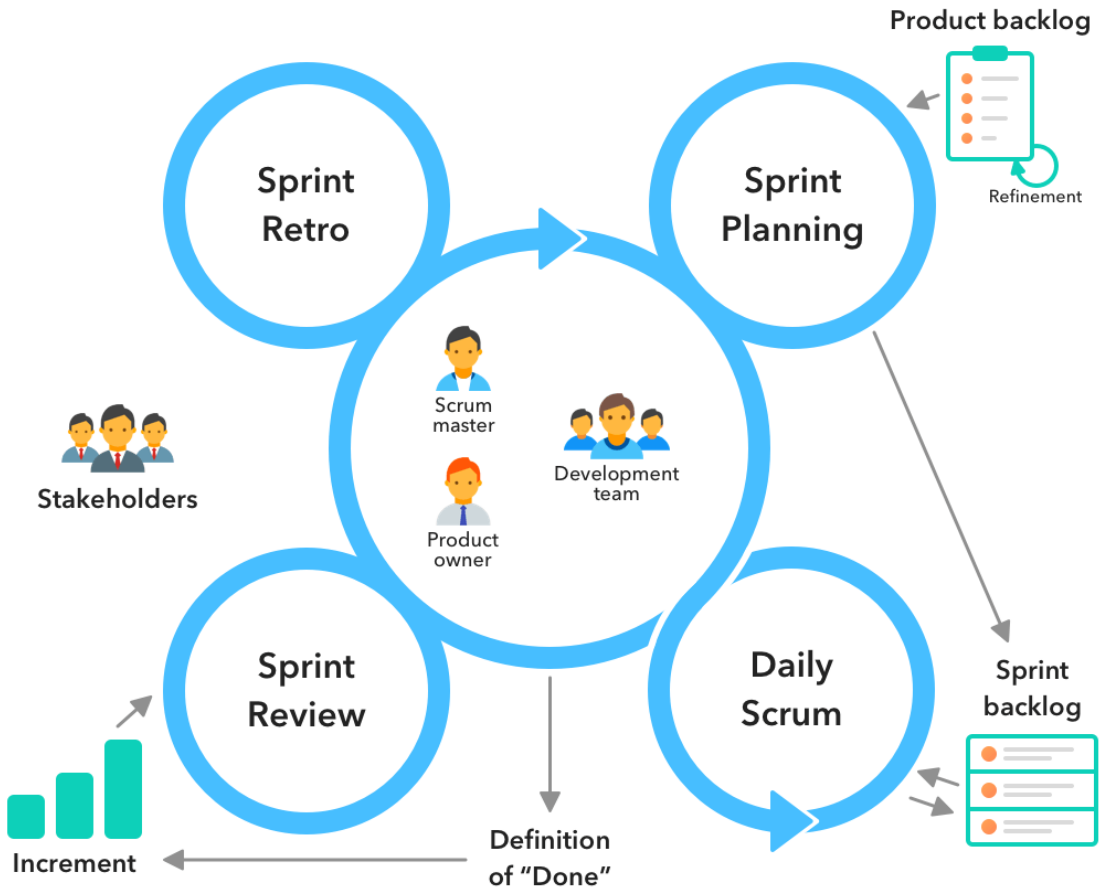
Framework to generate value through adaptative solutions

Product owner maintains the product backlog

The scrum team selects the sprint backlog that returns the best value

The team reviews and reflects on the sprint to improve the following iterations

The scrum master is a leader that serves the team (not bossing or pressuring the team!)

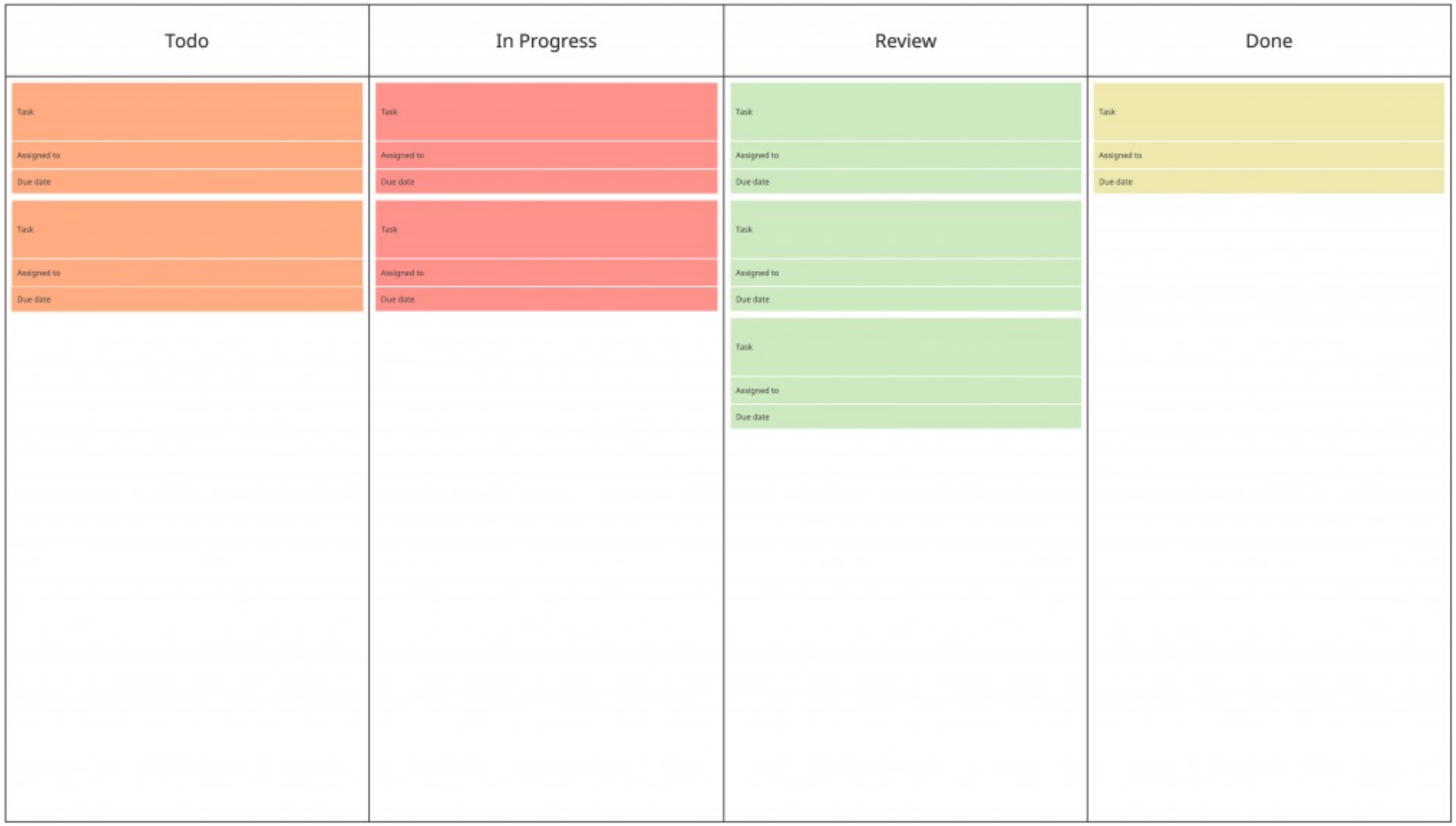


Source: https://startinfinity.s3.us-east-2.amazonaws.com/t/xxFYvXnlAeya4B2AkWuYLJhxeNQ9DyVarRxv7CaJ.png

Kanban

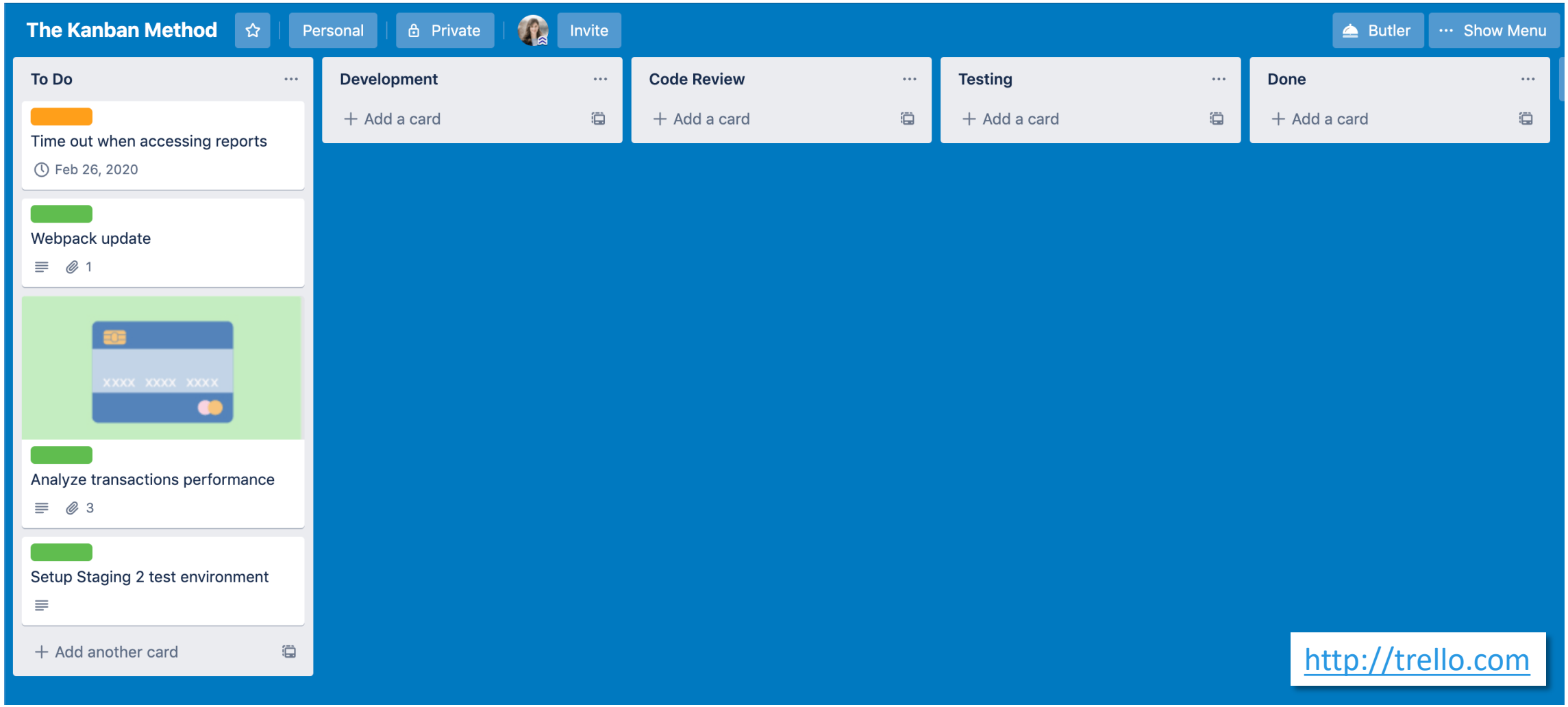
Workflow management based on a board where the development process is organized

More flexible than XP and Scrum



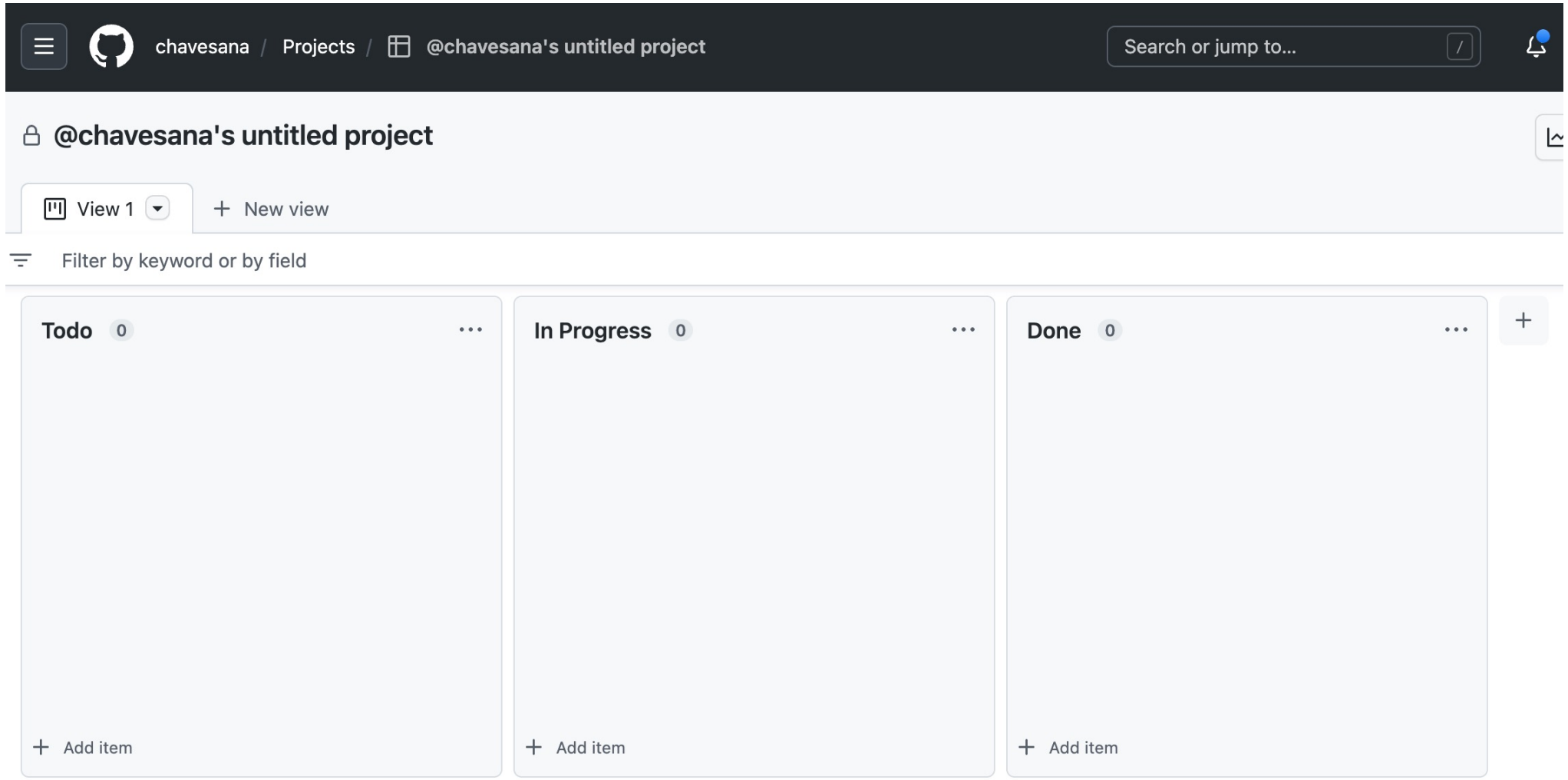
Source: https://d3n817fwly711g.cloudfront.net/uploads/2021/03/Kanban-Board-1024x586.png

Kanban



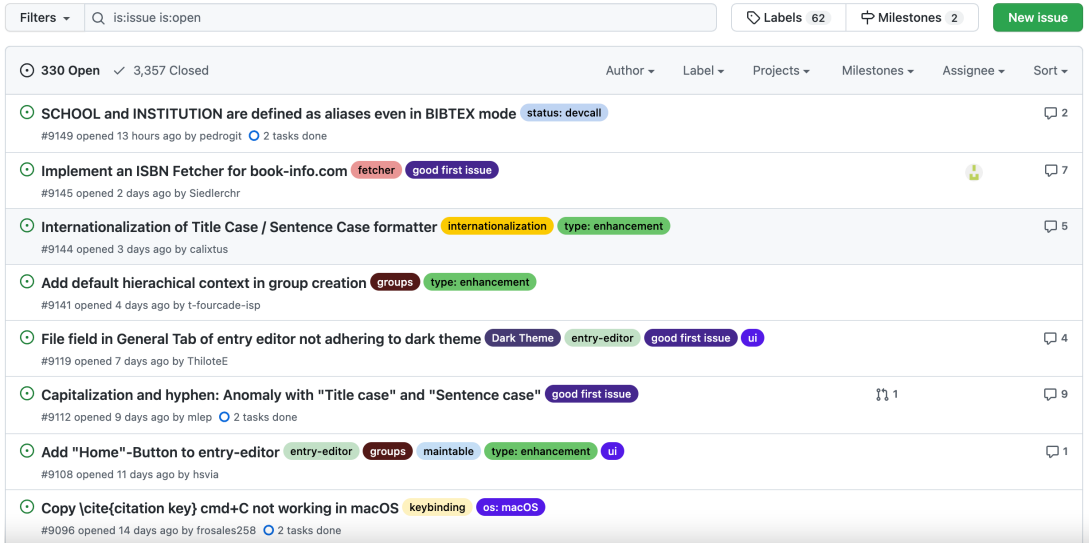
http://trello.com

Kanban


chavesana / Projects / @chavesana's untitled project
Search or jump to...
@chavesana's untitled project
View 1
New view
Filter by keyword or by field
Todo 0
In Progress 0
Done 0
Add item
Add item
Add item

GitHub issues

- On GitHub projects, development tasks are called **issues**
  - Originally called bug tracker
- Keeps record of changes that need to be done
  - Or nice new features
- Allows efficient communication with the community
- Can be added to the GitHub Kanban board



GitHub issues

List with Details | Size of Project | Status

Filter by keyword or by field

Free to take 43

jabref #8893
Support Citation Style Language (CSL) Styles in LibreOffice/OpenOffice - University Project
entry-preview | openoffice/libreoffice | Project: SE HIT 2022 | type: feature | Medium

jabref #8739
Improve the usability of keywords for tagging
keywords | Project: SE HIT 2022 | ui | #1

jabref #8645
An empty entry can still be added without warning message (and lead to an error when compiling)
bib(la)tex | Project: SE HIT 2022 | type: enhancement | #8941 | #8981 | Small

jabref #6601
Add right click menu for main table header
maintable | type: enhancement | ui | #7729 | #7964 | #8762 | Small

jabref #8055

Reserved 0

In Progress 2

jabref #7906
Please record search history
Project: SE HIT 2022 | search | type: enhancement | ui | #8974

jabref #6190
Improve entry merge dialog (3-way merge)
project: GSoC | type: enhancement | ui | Big

Under Review 0

Done 3

jabref #521
drag'n'drop to another library should be possible
good first issue | Project: SE HIT 2022 | #8982 | Small

jabref #12
Merge entries should be able to merge keywords and groups
Project: SE HIT 2022 | #8983 | Medi

jabref #9068
title case may be no right when the en-d exist in title
bib(la)tex | good first issue | type: enhancement | #9099 | #91 | #9102

The end