•••

Unit 2: Design Principles

Session 5

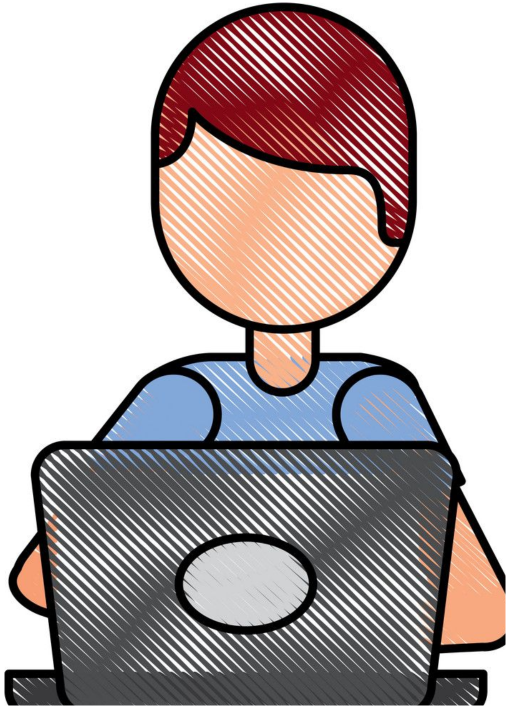
Lesson Outcome:

Understanding Test Driven Development (TDD), Behaviour Driven Development (BDD), and Domain Driven Development (DDD)

Test Driven Development (TDD)

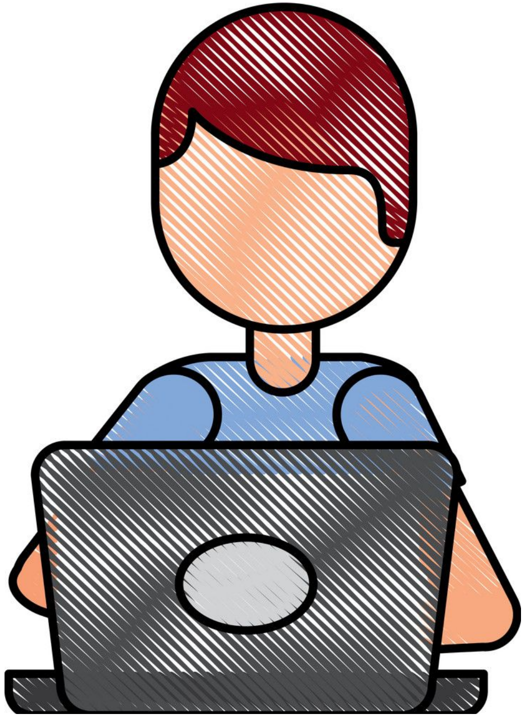
Test Driven Development

- "Test-driven development" refers to a style of programming in which three activities are tightly interwoven:
  - coding
  - testing (in the form of writing unit tests)
  - design (in the form of refactoring).



Check Understanding: 15 minutes

- Read up on unit testing and refactoring
    - refactoring
    - unit tests

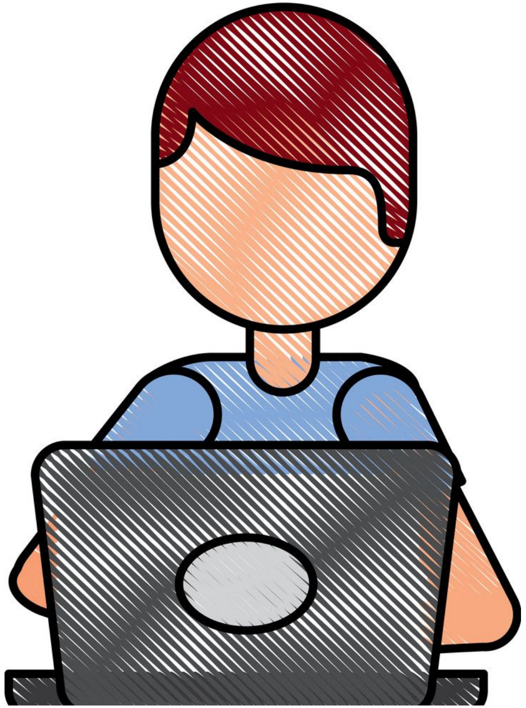


How does TDD work?

- TDD can be described by the following set of rules:
  - “write a “single” unit test describing an aspect of the program
  - run the test, which should fail because the program lacks that feature
  - write “just enough” code, the simplest possible, to make the test pass
  - “refactor” the code until it conforms to the simplicity criteria
  - repeat, “accumulating” unit tests over time

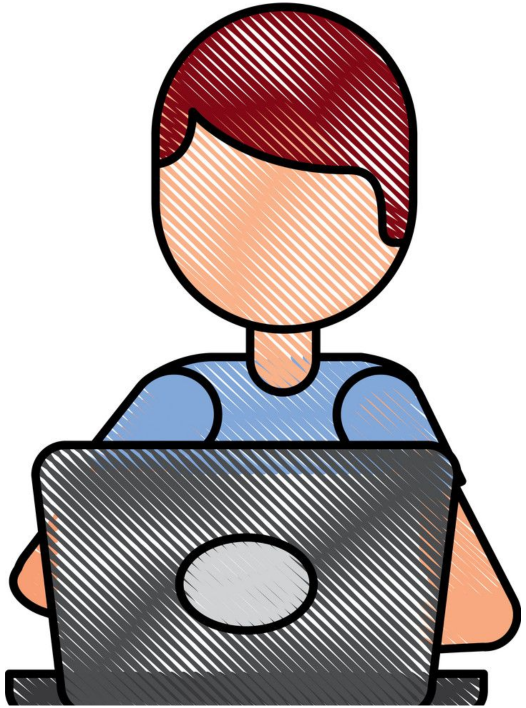
Benefits of TDD

- Fewer bugs in code
- Reduction in effort in projects' final phases
- Less debugging
- Improved design qualities in the code



Disadvantages of TDD

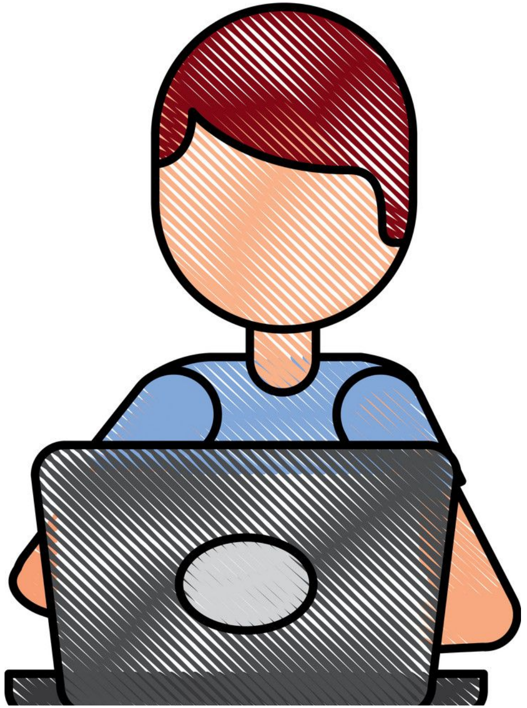
- Typical mistakes include:
  - Forgetting to run tests frequently
  - writing too many tests at once
  - writing tests that are too large or coarse-grained
  - writing overly trivial tests, for instance omitting assertions
  - writing tests for trivial code



Signs of Use of TDD

- “code coverage” is a common approach to evidencing the use of TDD
    - coverage below 80% is likely to indicate deficiencies in a team’s mastery of TDD

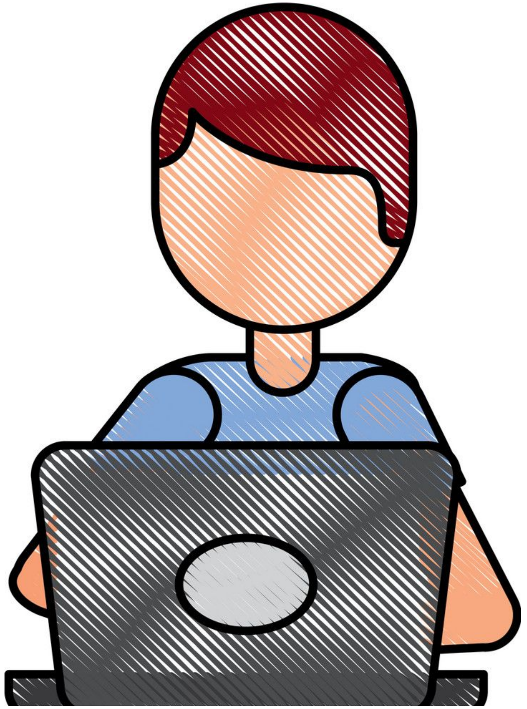
- version control logs should show that test code is checked in each time product code is checked in, in roughly comparable amounts



Skill Levels

Beginner

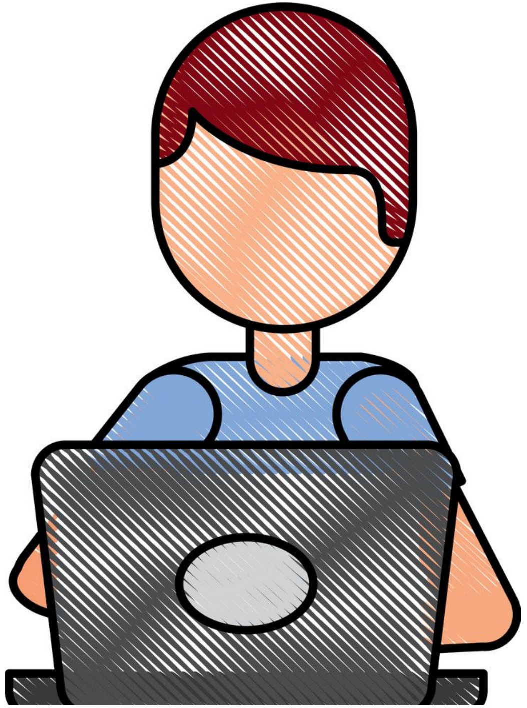
- able to write a unit test prior to writing the corresponding code
- able to write code sufficient to make a failing test pass



Skill Levels

Intermediate

- practices “test driven bug fixing”: when a defect is found, writes a test exposing the defect before correction
- able to decompose a compound program feature into a sequence of several unit tests to be written



How are you feeling?

RED
I have no idea what you're talking about

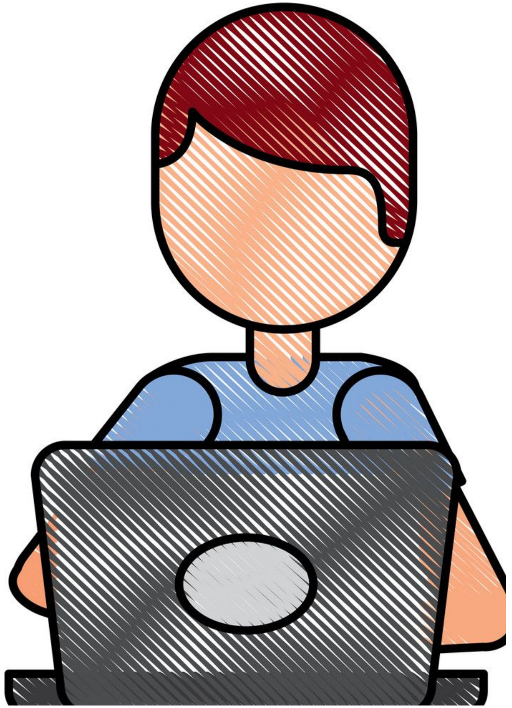
YELLOW
I have some questions but feel like I understand some things

GREEN
I feel comfortable with everything you've said

Behaviour Driven Development (BDD)

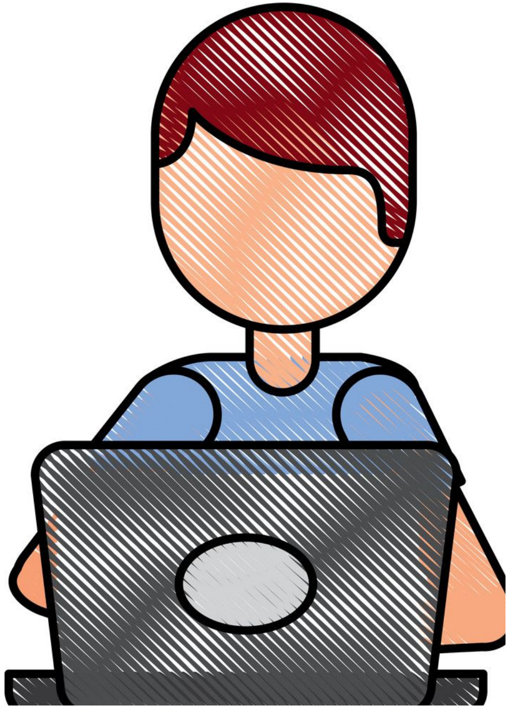
Behaviour Driven Development

- Behaviour-Driven Development (BDD) is the software development process that closes the gap between business people and technical people by:
  - Encouraging collaboration across roles to build shared understanding of the problem to be solved
  - Working in rapid, small iterations to increase feedback and the flow of value
  - Producing system documentation that is automatically checked against the system's behaviour



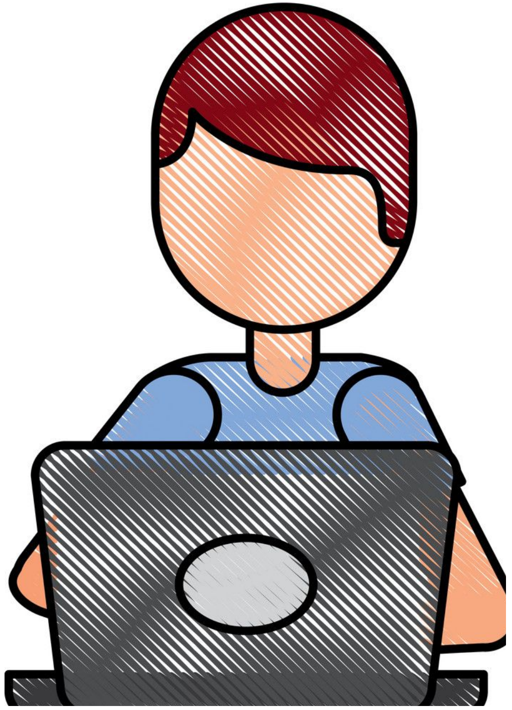
Behaviour Driven Development

- Behaviour-Driven Development (BDD) works by focusing collaborative work around concrete, real-world examples that illustrate how we want a system to behave.
- We use those examples to guide us from concept through to implementation, in a process of continuous collaboration



Behaviour Driven Development

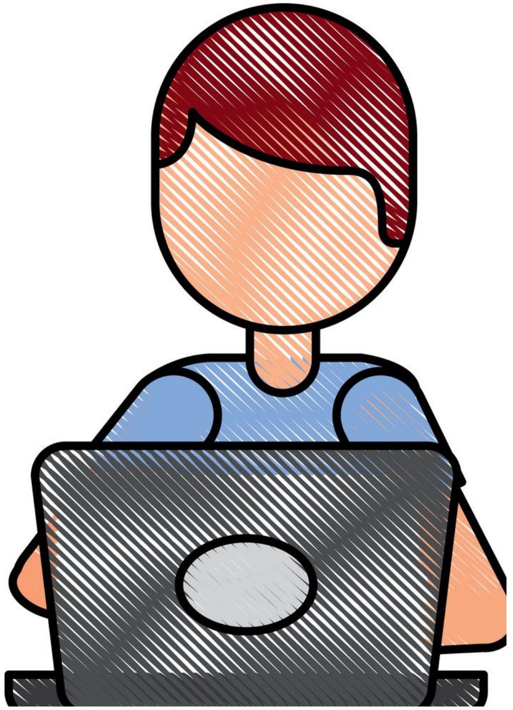
- Behaviour-Driven Development (BDD) is a way of improving an AGILE process
- What is AGILE? We will come back to this!



Behaviour Driven Development: Steps

BDD activity is a three-step, iterative process:

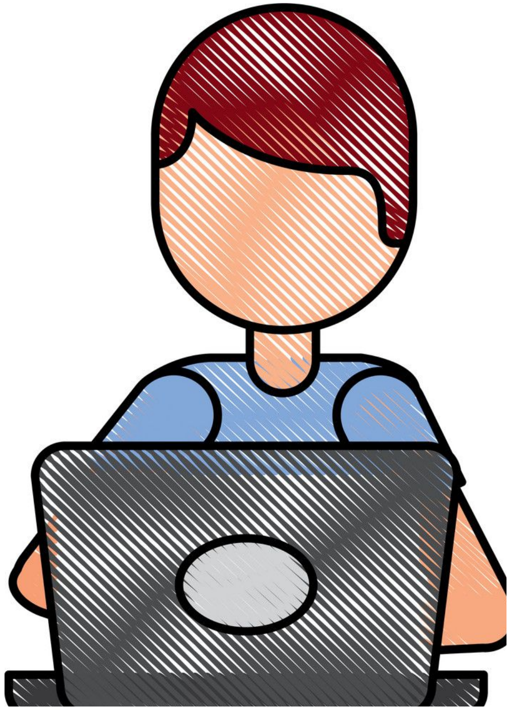
1. Step 1: Discovery
2. Step 2: Formulation
3. Step 3: Automation



Behaviour Driven Development: Discovery

BDD activity is a three-step, iterative process:

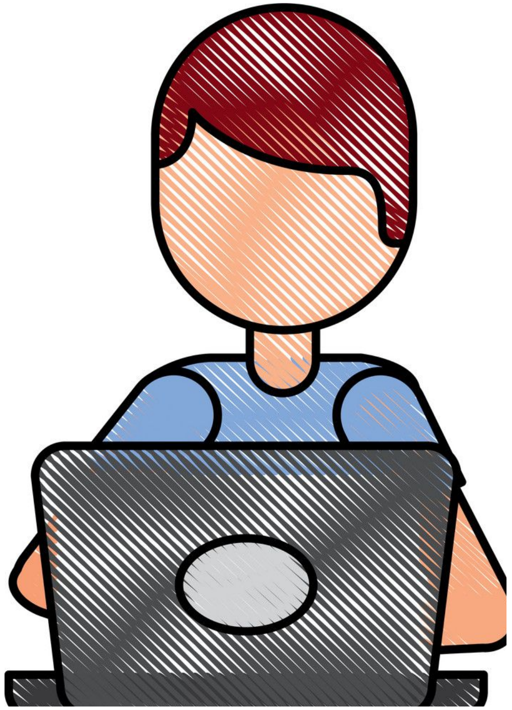
First, take a small upcoming change to the system – a User Story – and talk about concrete examples of the new functionality to explore, discover and agree on the details of what's expected to be done.



Behaviour Driven Development: Formulation

BDD activity is a three-step, iterative process:

Next, document those examples in a way that can be automated, and check for agreement.

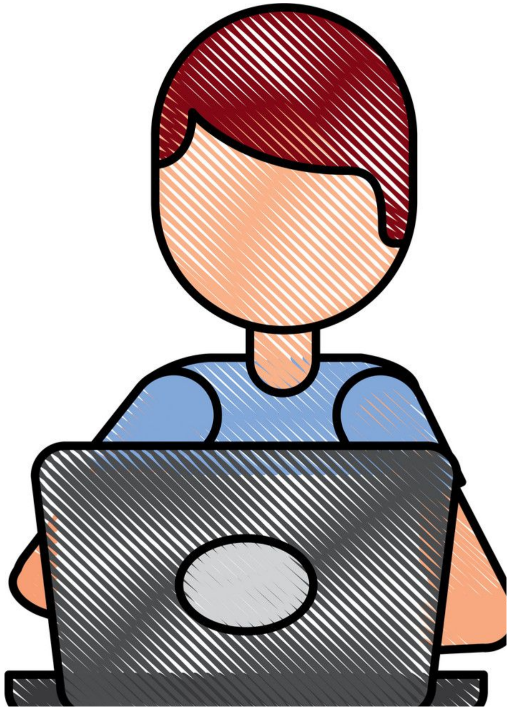


Behaviour Driven Development: Automation

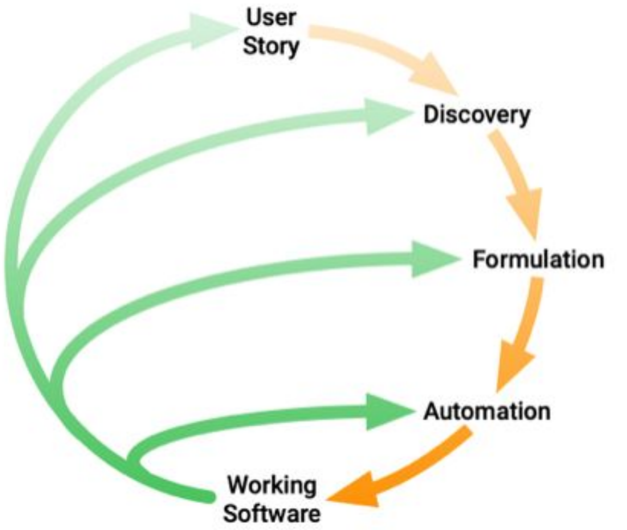
BDD activity is a three-step, iterative process:

Finally, implement the behaviour described by each documented example, starting with an automated test to guide the development of the code.

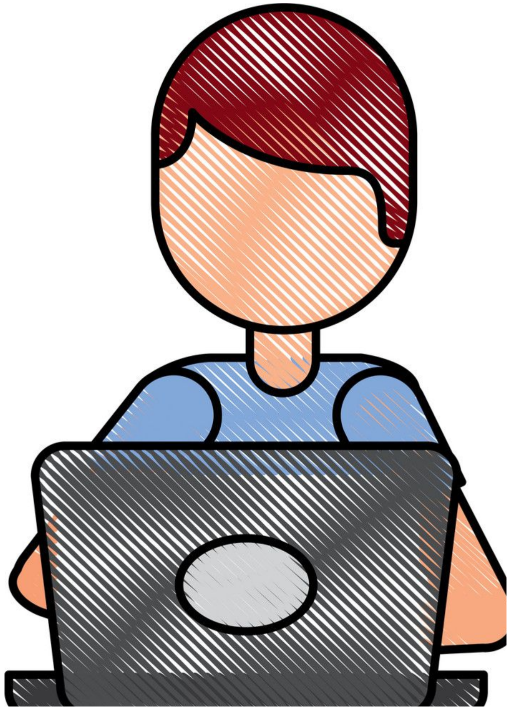
The idea is to make each change small and iterate rapidly, moving back up a level each time you need more information. Each time you automate and implement a new example, you've added something valuable to your system, and you're ready to respond to feedback.



Behaviour Driven Development



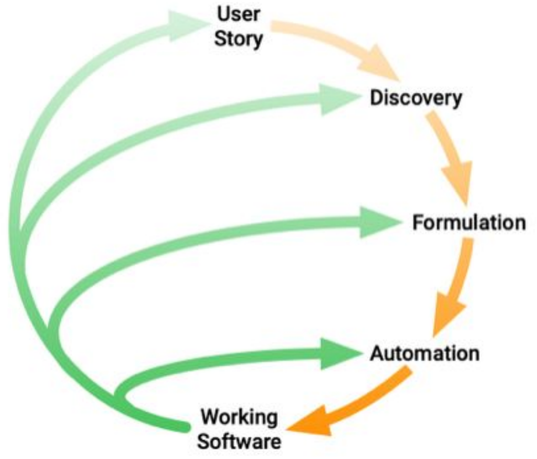
Discovery, Formulation and Automation



Behaviour Driven Development:

Over time, the documented examples become an asset that enables your team to continue confidently and rapidly making changes to the system.

The code reflects the documentation, and the documentation reflects the team's shared understanding of the problem domain. This shared understanding is constantly evolving.

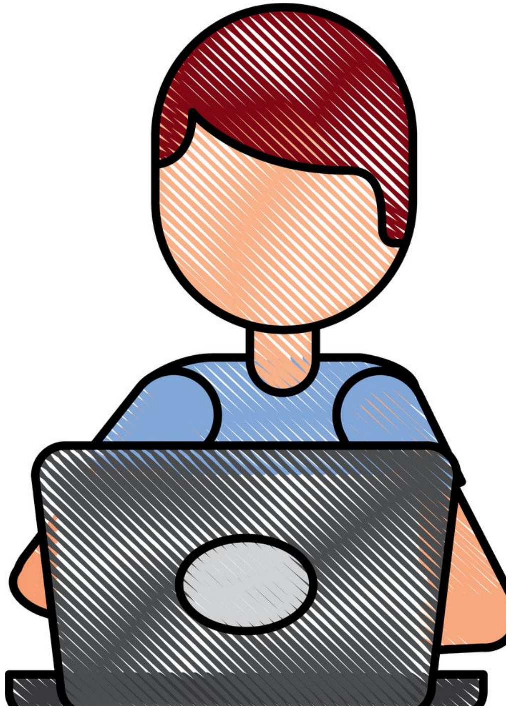


Discovery, Formulation and Automation

Behaviour Driven Development: Benefits

Key benefits of BDD

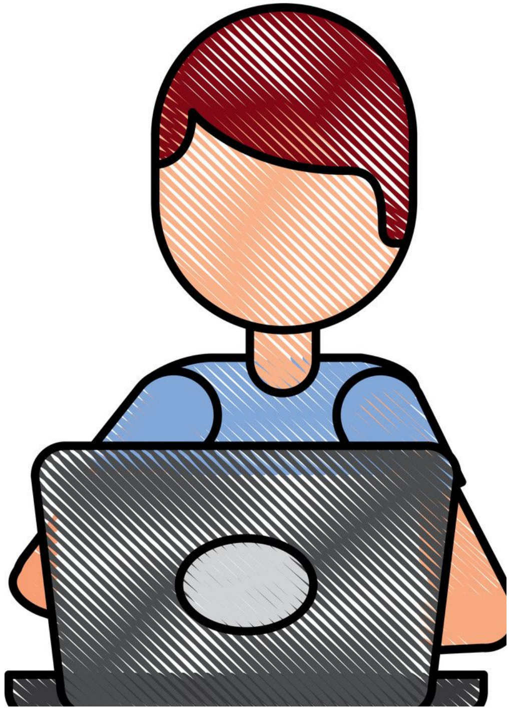
- All development work can be traced back directly to business objectives.
- Software development meets user need. Satisfied users = good business.
- Efficient prioritisation – business-critical features are delivered first.
- All parties have a shared understanding of the project and can be involved in the communication.



Behaviour Driven Development: Benefits

Key benefits of BDD

- A shared language ensures everyone (technical or not) has thorough visibility into the project's progression.
- Resulting software design that matches existing and supports upcoming business needs.
- Improved quality code reducing costs of maintenance and minimising project risk.



When is a good scenario to use BDD?

TDD vs. BDD What are the differences?

How are you feeling?

RED

I have no idea what you're talking about

YELLOW

I have some questions but feel like I understand some things

GREEN

I feel comfortable with everything you've said

Session 6

TDD and BDD Recap

What are the main advantages of TDD?

What are the main advantages of BDD?

How are you feeling?

RED
I have no idea what you're talking about

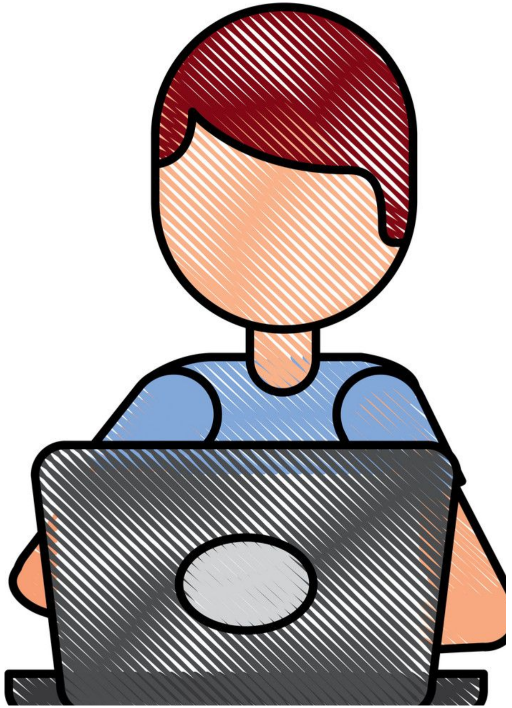
YELLOW
I have some questions but feel like I understand some things

GREEN
I feel comfortable with everything you've said

Final topic: AGILE Software Development

What is AGILE?

- You may have heard about AGILE before as a Project Management term or a Software Development term.
- You now have 25 minutes to read into **AGILE Software Development**



How are you feeling?

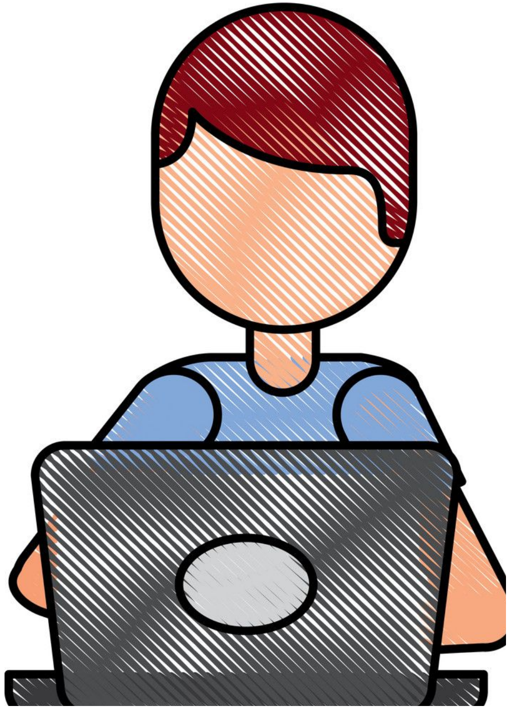
RED
I have no idea what you're talking about

YELLOW
I have some questions but feel like I understand some things

GREEN
I feel comfortable with everything you've said

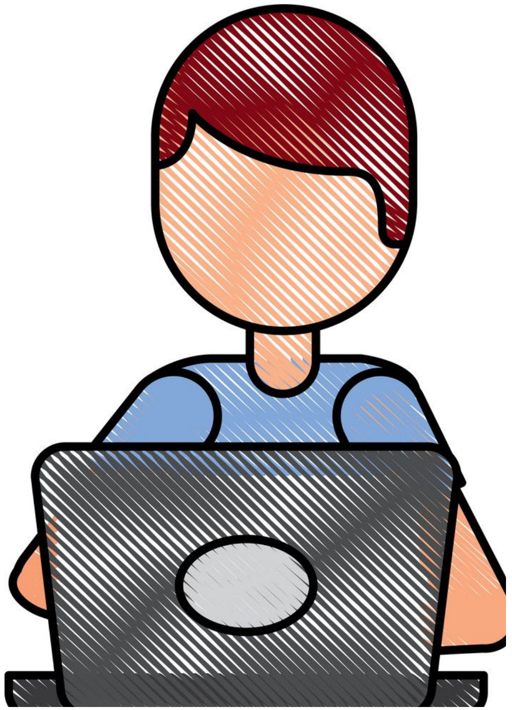
AGILE Recap

- AGILE is a Software Development Methodology that emphasises:
  - Collaboration between the development team and business stakeholders
  - Frequent delivery of business value
  - Small, self-organizing teams
  - Innovative ways to create, test and deploy code

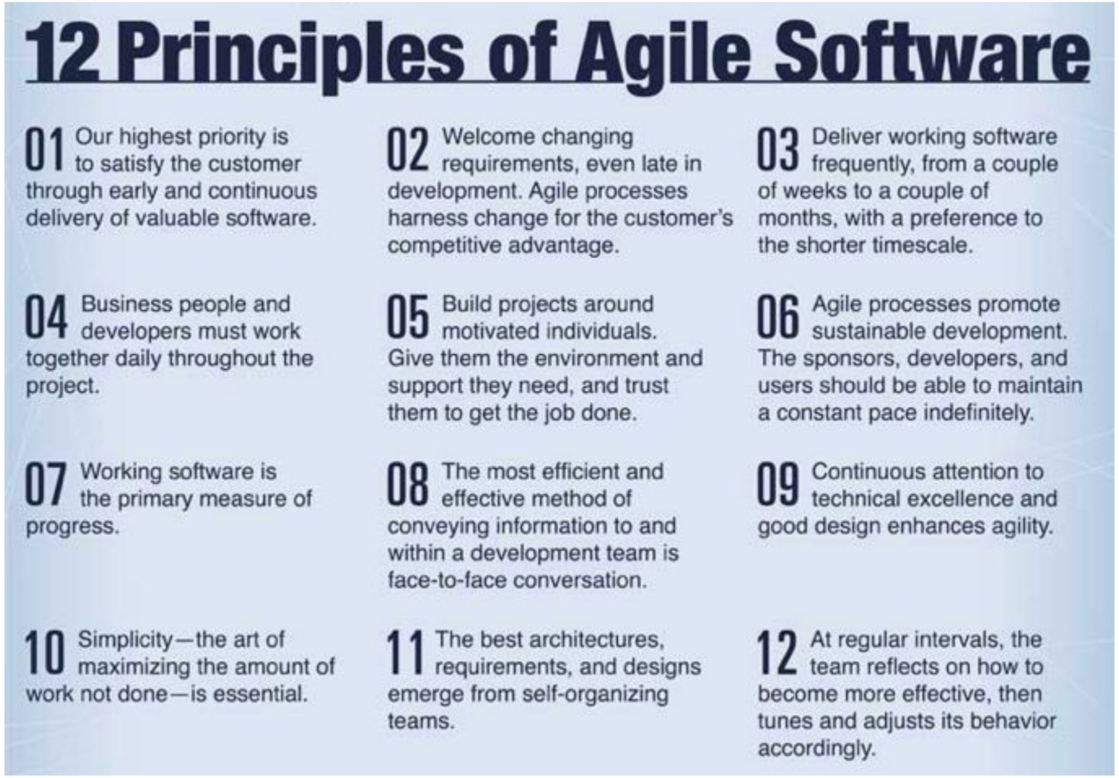


AGILE

The term “Agile” was applied to this collection of methodologies 21 years ago in 2001 when 17 software development practitioners co-located in Utah to debate and share their various approaches to software development to create the Agile Manifesto



AGILE



12 Principles of Agile Software

01 Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

02 Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

03 Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

04 Business people and developers must work together daily throughout the project.

05 Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

06 Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

07 Working software is the primary measure of progress.

08 The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

09 Continuous attention to technical excellence and good design enhances agility.

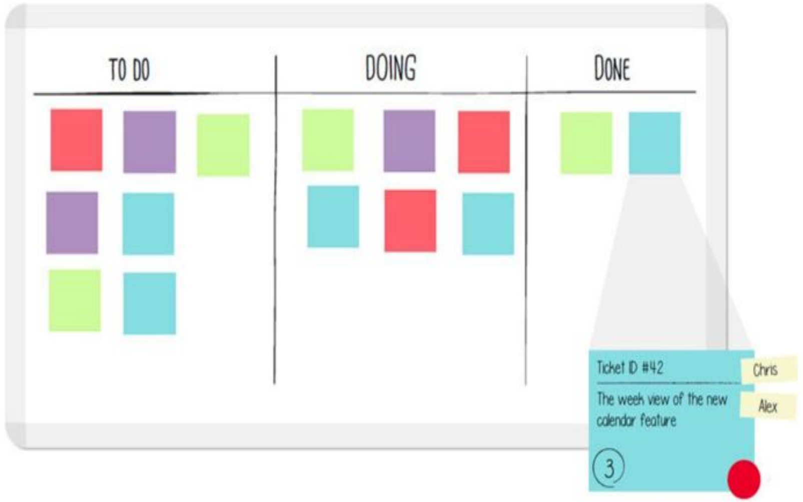
10 Simplicity—the art of maximizing the amount of work not done—is essential.

11 The best architectures, requirements, and designs emerge from self-organizing teams.

12 At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

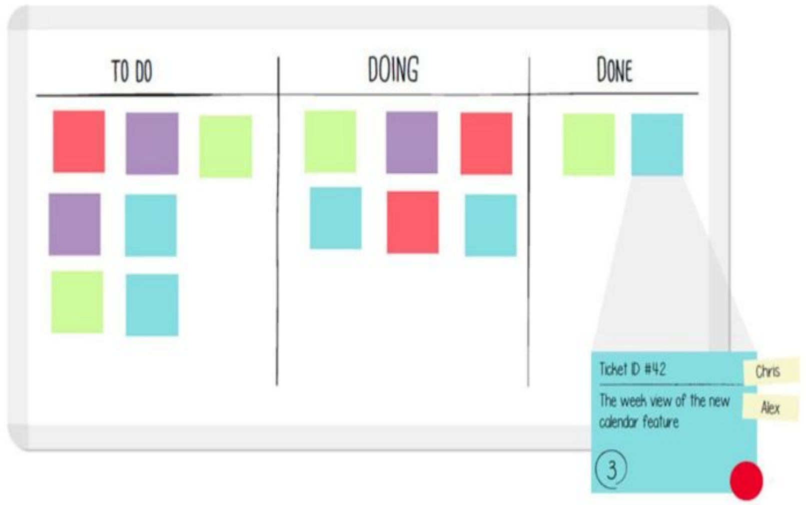
Key AGILE concepts: Kanban

Kanban is Japanese for “visual signal” or “card.” Kanban helps you harness the power of visual information by using sticky notes on a whiteboard to create a “picture” of your work.

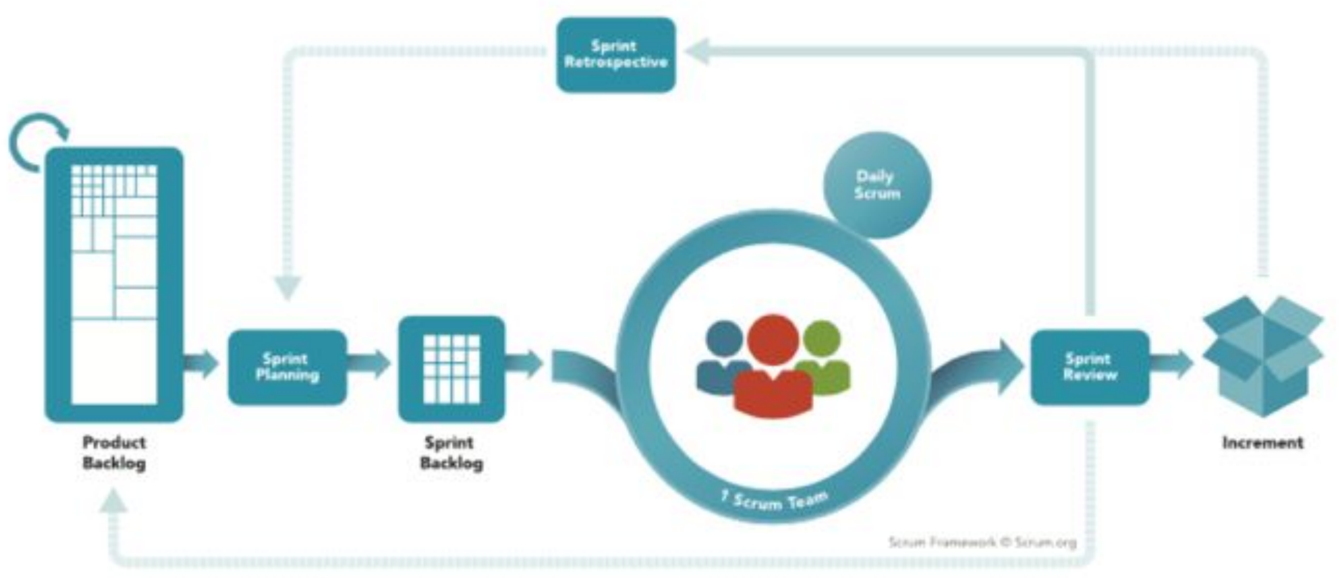


Key AGILE concepts: Kanban

Seeing how your work flows within your team's process lets you not only communicate status but also give and receive context for the work.



Key AGILE concepts: Scrum



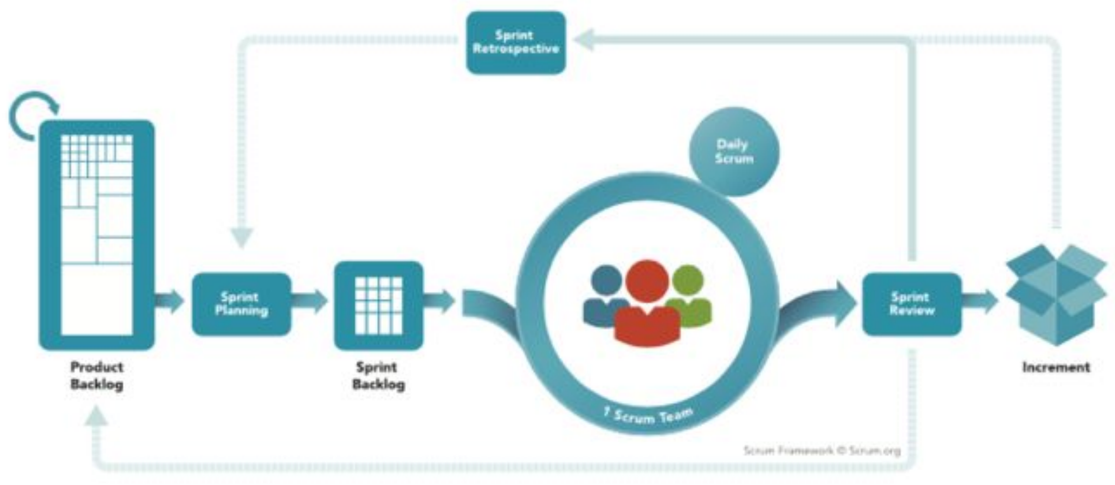
Key AGILE concepts: Scrum

Scrum is a simple framework for effective team collaboration on complex software projects.

The Framework is based off The Scrum Guide which Scrum co-creators Ken Schwaber and Jeff Sutherland have written to explain Scrum clearly and succinctly.

Key AGILE concepts: Scrum

- 25 minutes to read into [The Scrum Guide]



Key AGILE concepts: Sprint

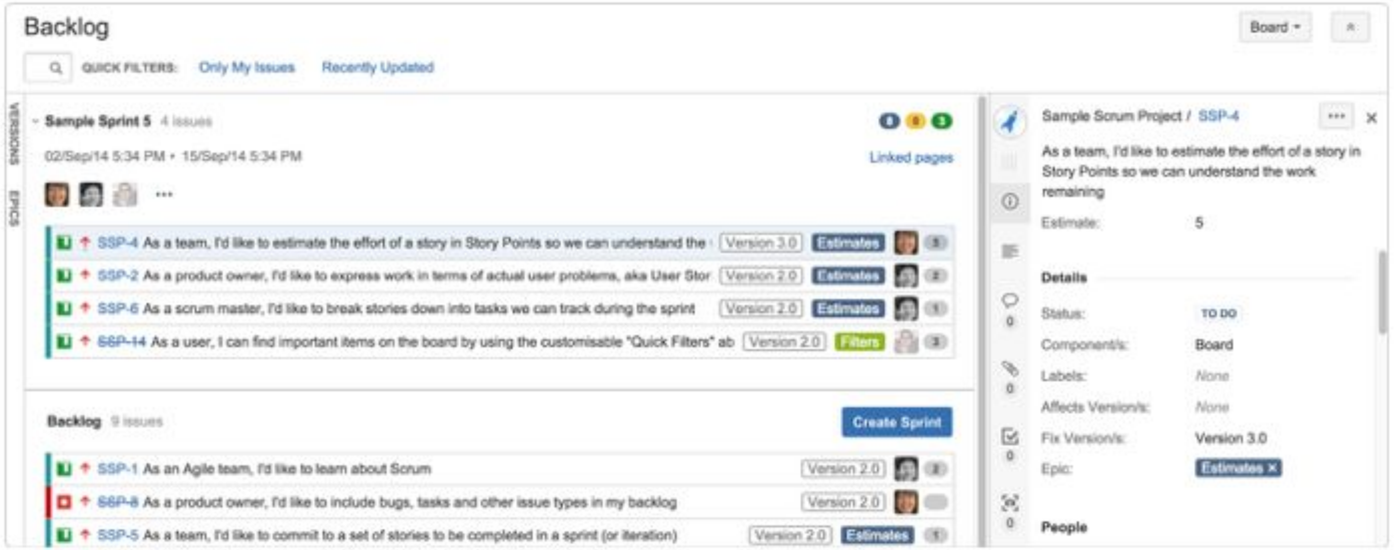
A sprint, also known as an iteration, is a short (ideally two to four week) period in which the development team implements and delivers a “done” product increment.

A new Sprint starts immediately after the conclusion of the previous Sprint.

Key AGILE concepts: Backlog

A backlog is a list of features or technical tasks which the team maintains. The backlog is the primary point of entry for knowledge about requirements, and the single authoritative source defining the work to be done. The backlog is expected to change throughout the project’s duration as the team gains knowledge.

Key AGILE concepts: Backlog



Backlog

QUICK FILTERS: Only My Issues Recently Updated

Board

VERSIONS EPICS

Sample Sprint 5 4 issues

02/Sep/14 5:34 PM • 15/Sep/14 5:34 PM

Linked pages

SSP-4 As a team, I'd like to estimate the effort of a story in Story Points so we can understand the	Version 3.0	Estimates	5
SSP-2 As a product owner, I'd like to express work in terms of actual user problems, aka User Stor	Version 2.0	Estimates	2
SSP-6 As a scrum master, I'd like to break stories down into tasks we can track during the sprint	Version 2.0	Estimates	1
~~SSP-14~~ As a user, I can find important items on the board by using the customisable "Quick Filters" ab	Version 2.0	Filters	3

Backlog 9 issues

Create Sprint

SSP-1 As an Agile team, I'd like to learn about Scrum	Version 2.0		2
~~SSP-8~~ As a product owner, I'd like to include bugs, tasks and other issue types in my backlog	Version 2.0		
SSP-5 As a team, I'd like to commit to a set of stories to be completed in a sprint (or iteration)	Version 2.0	Estimates	1

Sample Scrum Project / SSP-4

As a team, I'd like to estimate the effort of a story in Story Points so we can understand the work remaining

Estimate: 5

Details

Status:	TO DO
Component/s:	Board
Labels:	None
Affects Version/s:	None
Fix Version/s:	Version 3.0
Epic:	Estimates ×

People

How are you feeling?

RED
I have no idea what you're talking about

YELLOW
I have some questions but feel like I understand some things

GREEN
I feel comfortable with everything you've said

Thank You