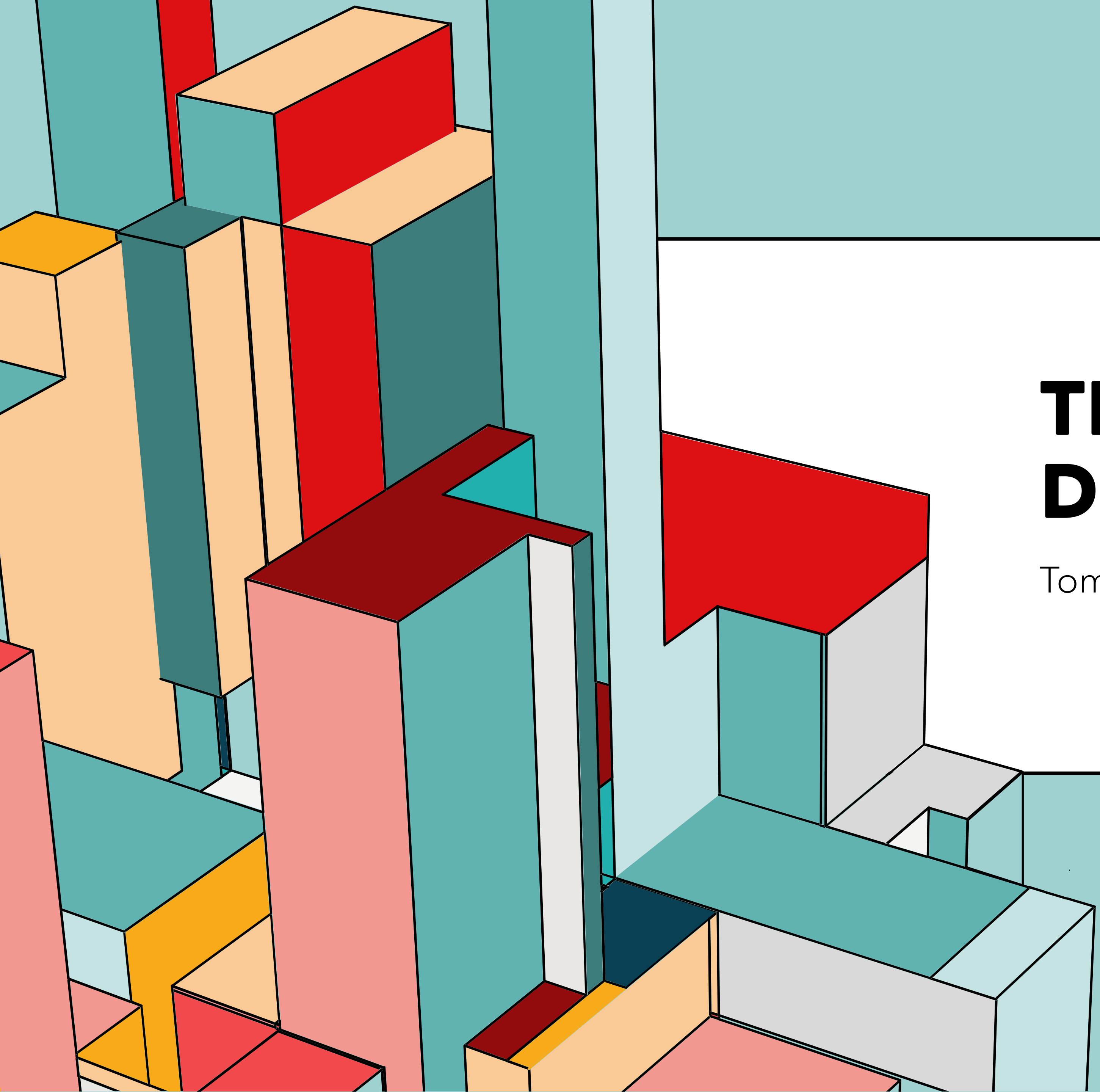
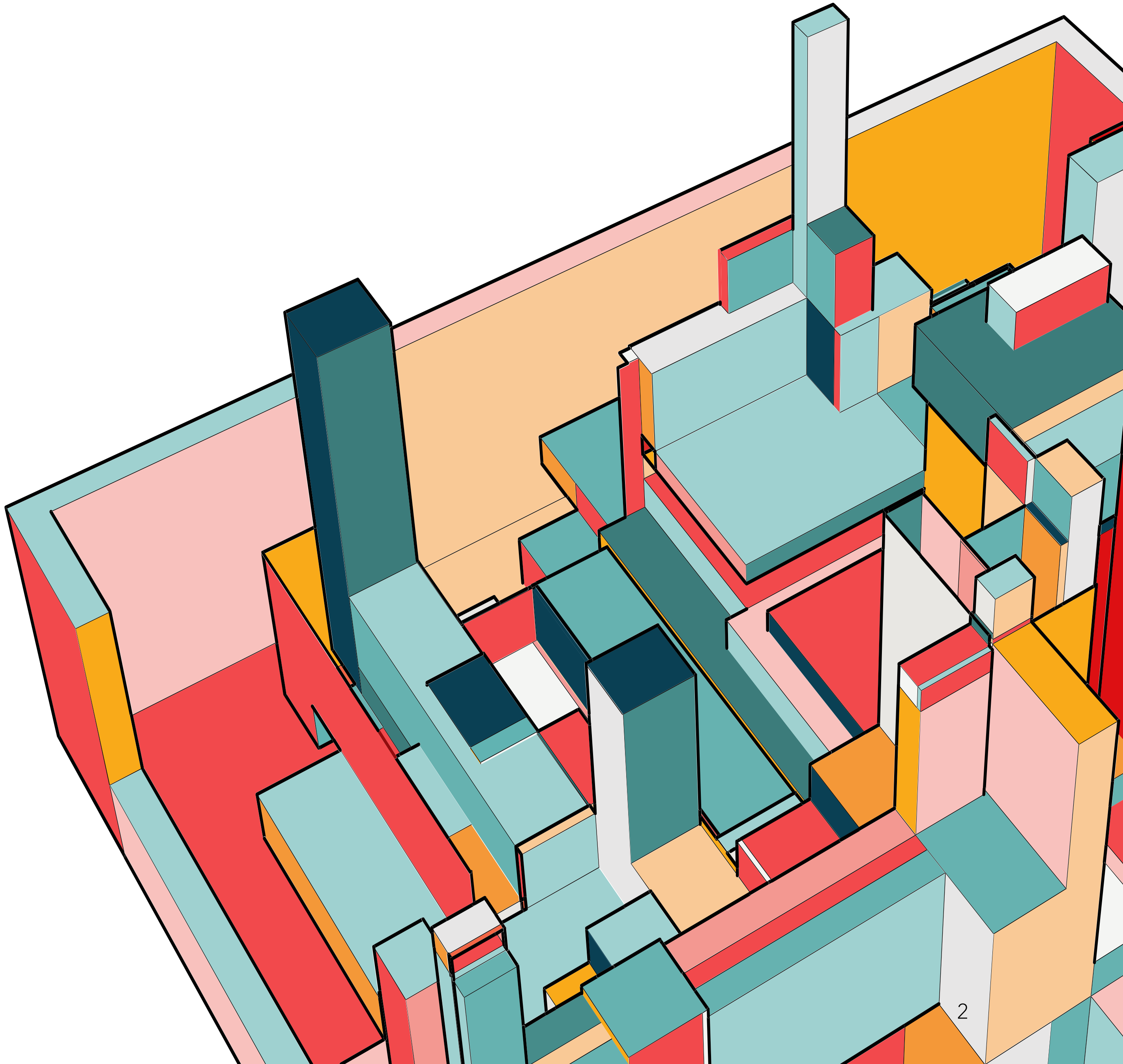


TEST DRIVEN DEVELOPMENT

Tom Legel

CONTENT



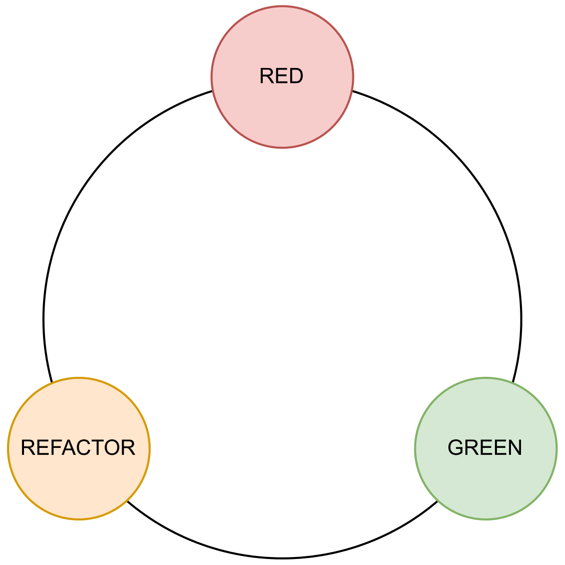
OVERVIEW

IDEA

Tests are written (beforehand) to drive development

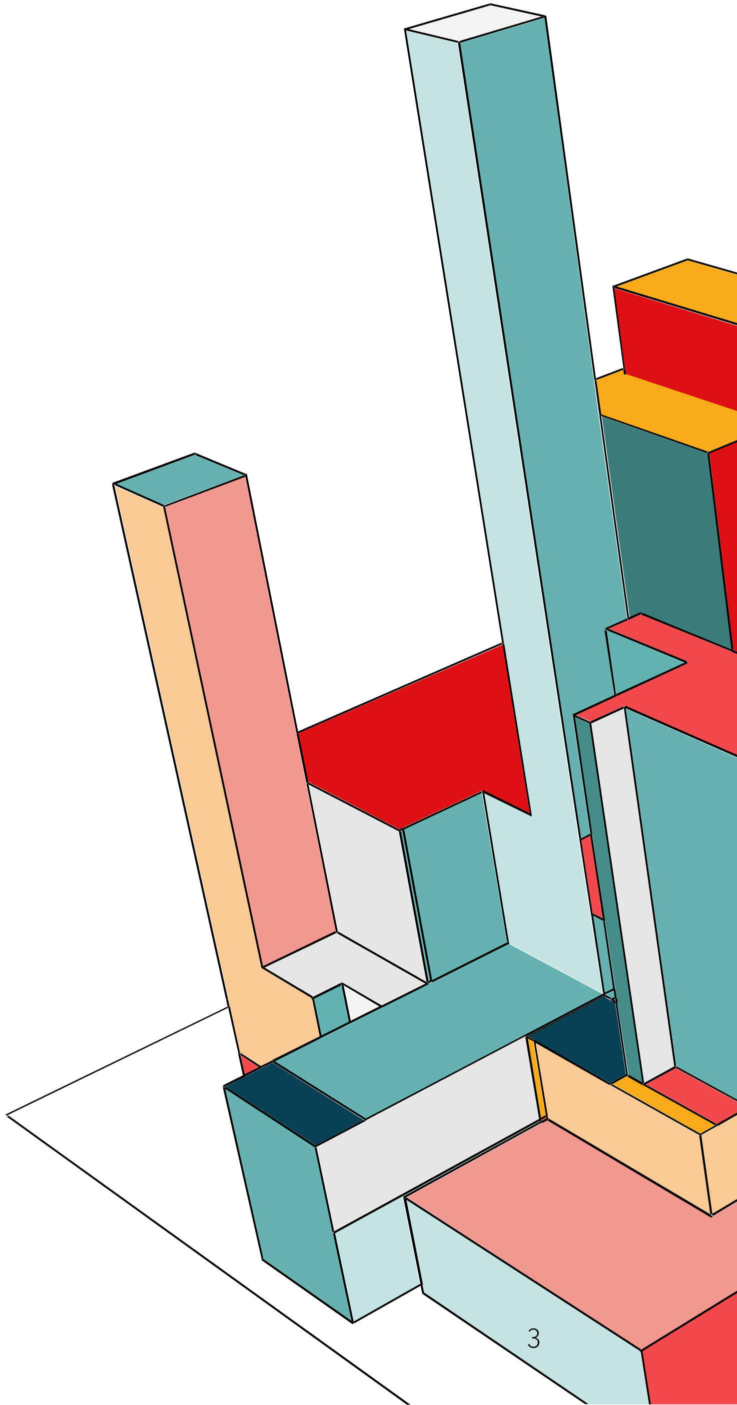
HISTORY

Developed by Kent Beck as a part of extreme Programming

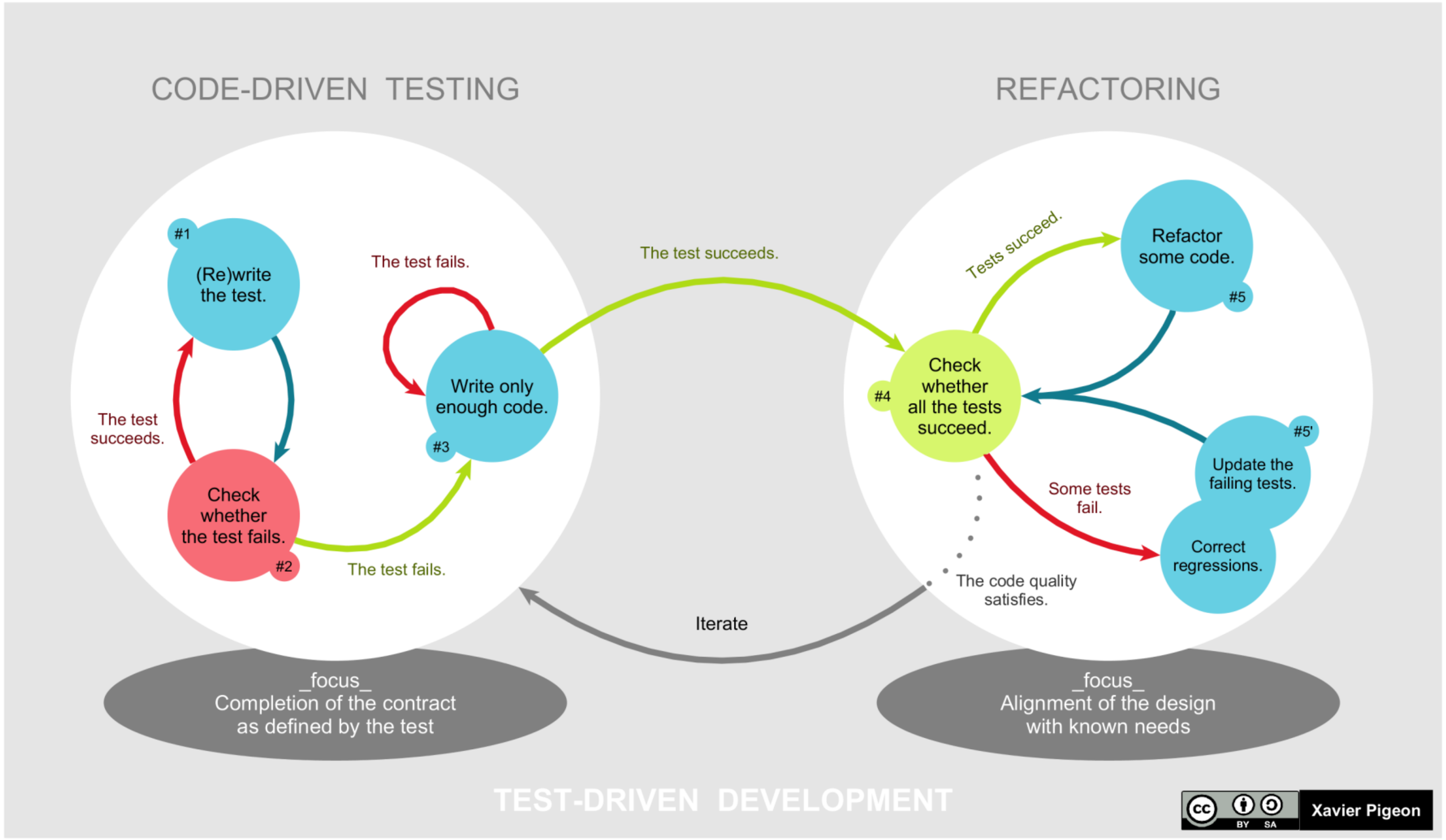


CORE PRINCIPLES

1. Write a test and see it fail:
   RED phase
2. Write minimal solution to make it pass:
   GREEN phase.
3. Refactor your code.
4. Go back to RED and (re-)write/extend
   your tests.



THE PROCESS IN MORE DETAIL



BUT… Y?

YAGNI (you aren‘t gonna need it)

TDD forces you to be minimalistic. Stick to your requirements / only implement what is worth the effort to test it.

You‘ll need to test anyways

At some point you will need a certain amount of test coverage. Why not begin with it?

Your test is actually working

Writing a failing test ensures that the following implementation is actually necessary to let it pass.

Yet it yields deeper understanding

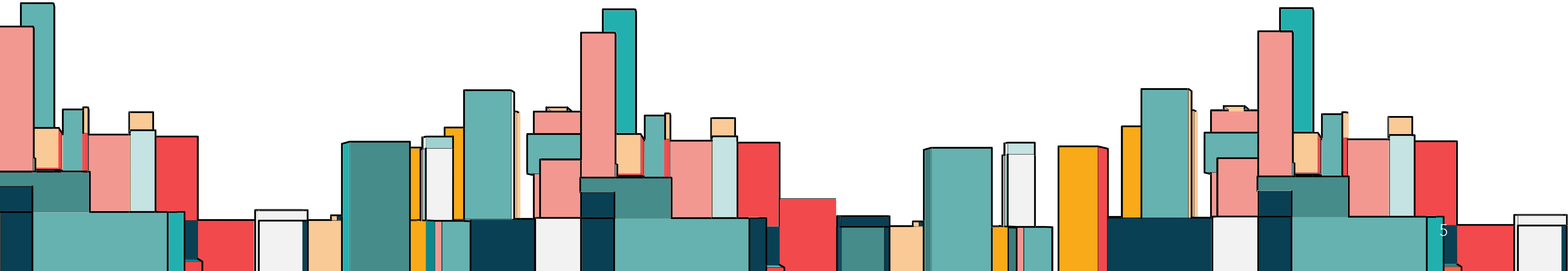
Writing the tests beforehand leads to a deeper understanding of the requirements instead of jumping into design patterns

You‘ll have fast feedback

You‘ll immediately have a feedback on the results of your code.

You can refactor more safely

If refactoring goes wrong, you will see it.



SOME VARIANTS OF TDD

UTTD

Unit-Test-Driven-Development

- Traditional understandment of TDD
- Focus lies on Unit-Tests / Components

ATTD

Acceptance-Test-Driven-Development

- Extends the UTTD workflow by acceptance tests
- Focus on user stories / functional requirements / black-box-testing

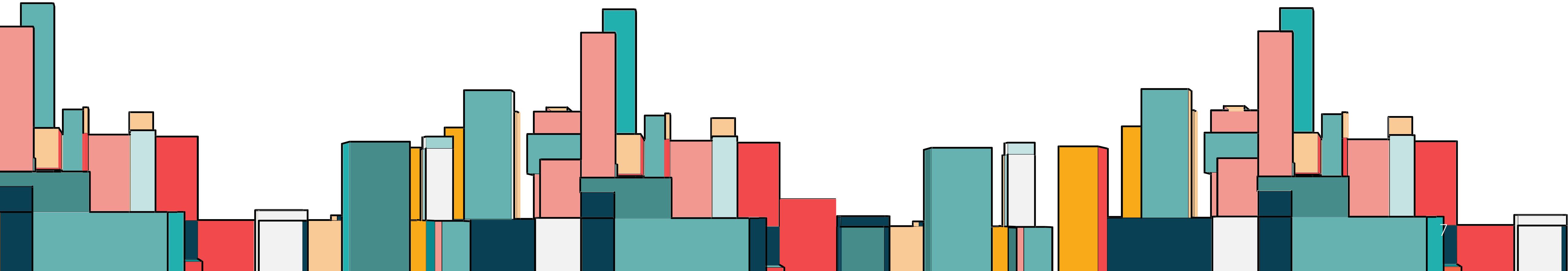
BDD

Behaviour-Driven-Development

- Focuses on the behaviour of a product
- Tries to be inclusive towards non-technical contributors

TOOLS

- Tools depend on programming language
- Tools do not differ from tools for ‚conventional' testing and test case management
- Eg.:
    - Junit and TestNG for Java
    - PyUnit and DocTest for Python
    - Cypress (for JS) and Selenium (JS, Java, ...) for Webapps



INSTALL CYPRESS

Basic Installation with NPM

```
cd /your/project/path
npm install cypress --save-dev
```

or YARN

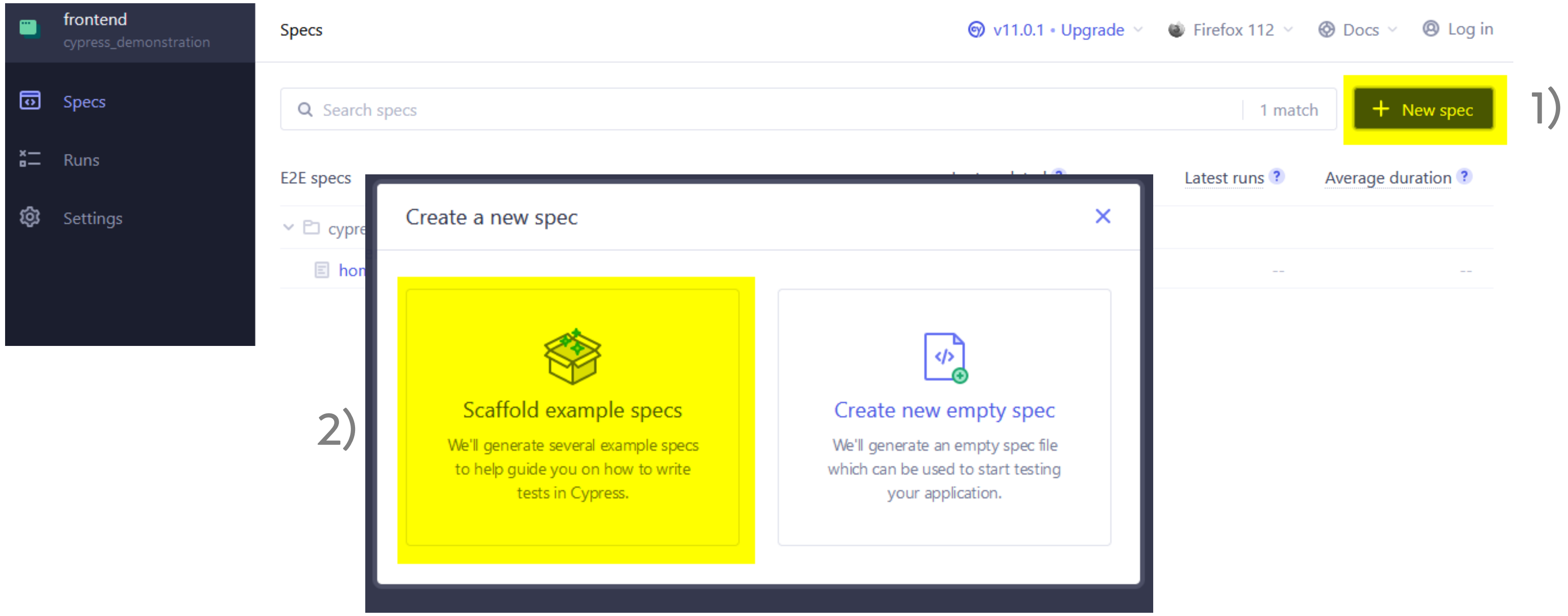
```
cd /your/project/path
yarn add cypress --dev
```

Start cypress UI:

```
cd /your/project/path
npx cypress open
```

GETTING TO KNOW CYPRESS

- Cypress website with tutorials, best practices, examples and API, e.g. offering command line guide
- Included example specs with lots of standard test cases for E2E-testing with cypress



We'll generate several example specs to help guide you on how to write tests in Cypress.

We'll generate an empty spec file which can be used to start testing your application.

SOURCES

- Test Driven Development with React
  https://link.springer.com/book/10.1007/978-1-4842-6972-5
- Dalton, J. (2019). Test-Driven Development. In: Great Big Agile. Apress, Berkeley, CA.
  https://doi.org/10.1007/978-1-4842-4206-3_67