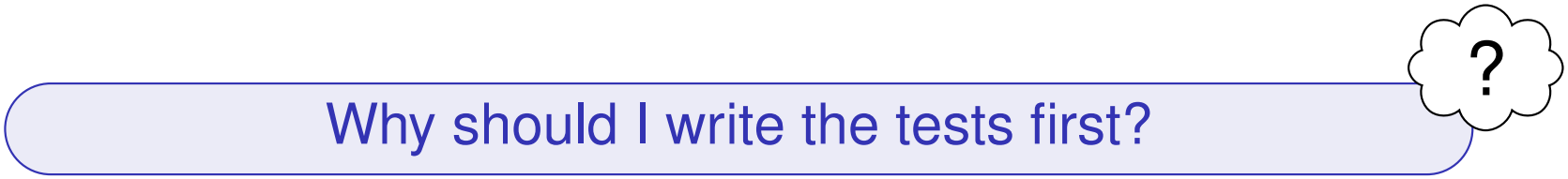
Cours de C++

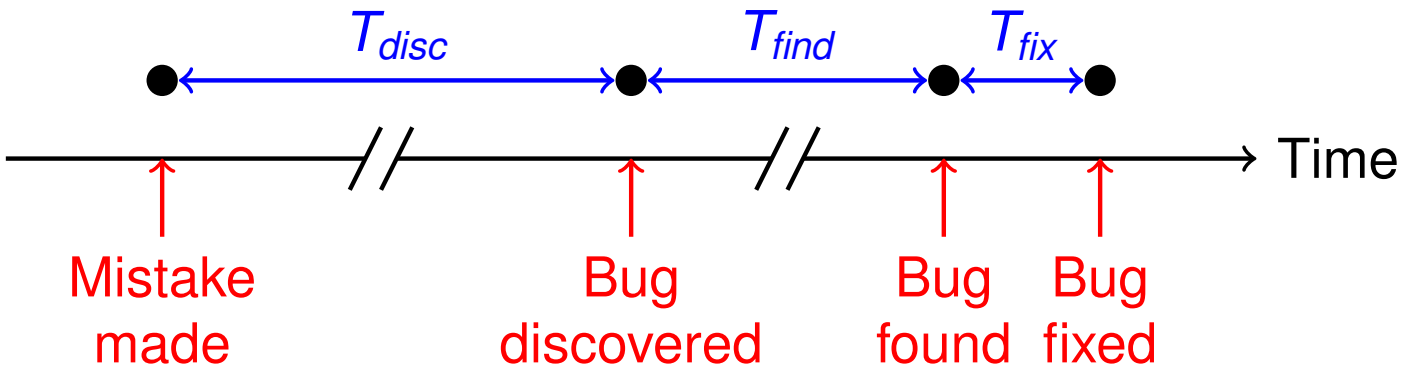
TDD : Test Driven Development

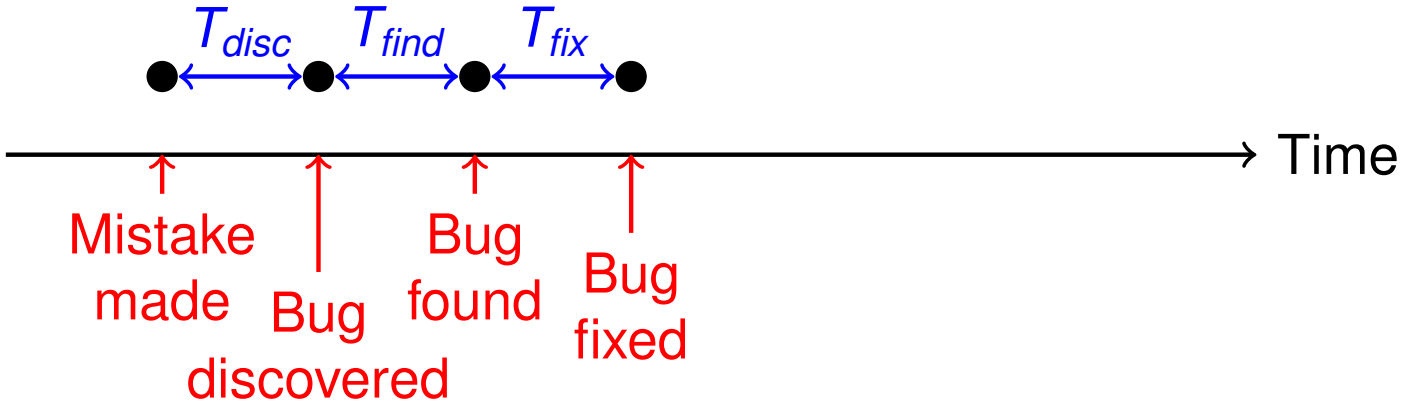
Cécile Braunstein

cecile.braunstein@lip6.fr

Motivation


Why should I write the tests first?
?

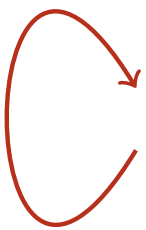

T_{disc}
T_{find}
T_{fix}
Time
Mistake made
Bug discovered
Bug found
Bug fixed


T_{disc}
T_{find}
T_{fix}
Time
Mistake made
Bug discovered
Bug found
Bug fixed

Advantages of TDD

- Improved software quality (fewer bugs)
- Reduced debugging time
- Leads to clean and modular design
- Allows for safe code changes
- Executable documentation
- Keeps track of progress
- Predictable development time

TDD Cycle



1. Write a test
2. Run the tests and watch it fail
3. Write the code to make it pass
4. Run the tests and see it works
5. Refactor code

Getting Started

Which tests should I write?

Writing a test plan

- What behavior should be implemented?
- What output do we expect?
- Input constraints
- Failure conditions

Incremental Design

- Unit by unit
- Non regression
- Test the behaviors not the methods !

Fondations

Unit Test

Verify the behavior of a code unit.

1. (Optional) Set up the execution context (Given)
2. Some statements to invoke the behavior you want to check (When)
3. Some statements to check the expected outcome (Then)

They can be grouped in a test suite.

Using tests to describe behavior

Given a meaningful name + test statement test statement

- Documentation purpose
- Provide simple example

Frameworks

- CppUnit
- Google Test
- customize assert
- Boost Test library
- Microsoft Unit Testing
- Catch2
- doctest
- ...


```
// Let DocTest provide main():
#define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN

#include "doctest.h"
#include "passgrid.hh"

TEST_SUITE("grid") {
  TEST_CASE("1: Size of the grid are initialized by construction"){
    PassGrid p0(0,0);
    CHECK(p0.getW() == 0);
    CHECK(p0.getH() == 0);
    PassGrid p1(10,8);
    CHECK(p1.getW() == 10);
    CHECK(p1.getH() == 8);
  }

  TEST_CASE("2: copy same grid give two identical grids"){
    PassGrid p0(0,0);
    PassGrid p1(p0);
    CHECK(!notEqualGrids(p0,p1));
  }
}
```