Software Engineering

Paulo Borba
Informatics Center
Federal University of Pernambuco

pauloborba.cin.ufpe.br

To do before class

- Watch videos
- Read chapter 7 and basic concepts of chapter 6 in the textbook
- Send questions and opinions through slack

Testing 1: implementation, maintenance and execution

Software Engineering

Paulo Borba
Informatics Center
Federal University of Pernambuco

pauloborba.cin.ufpe.br

What is software testing?

Main goal is…

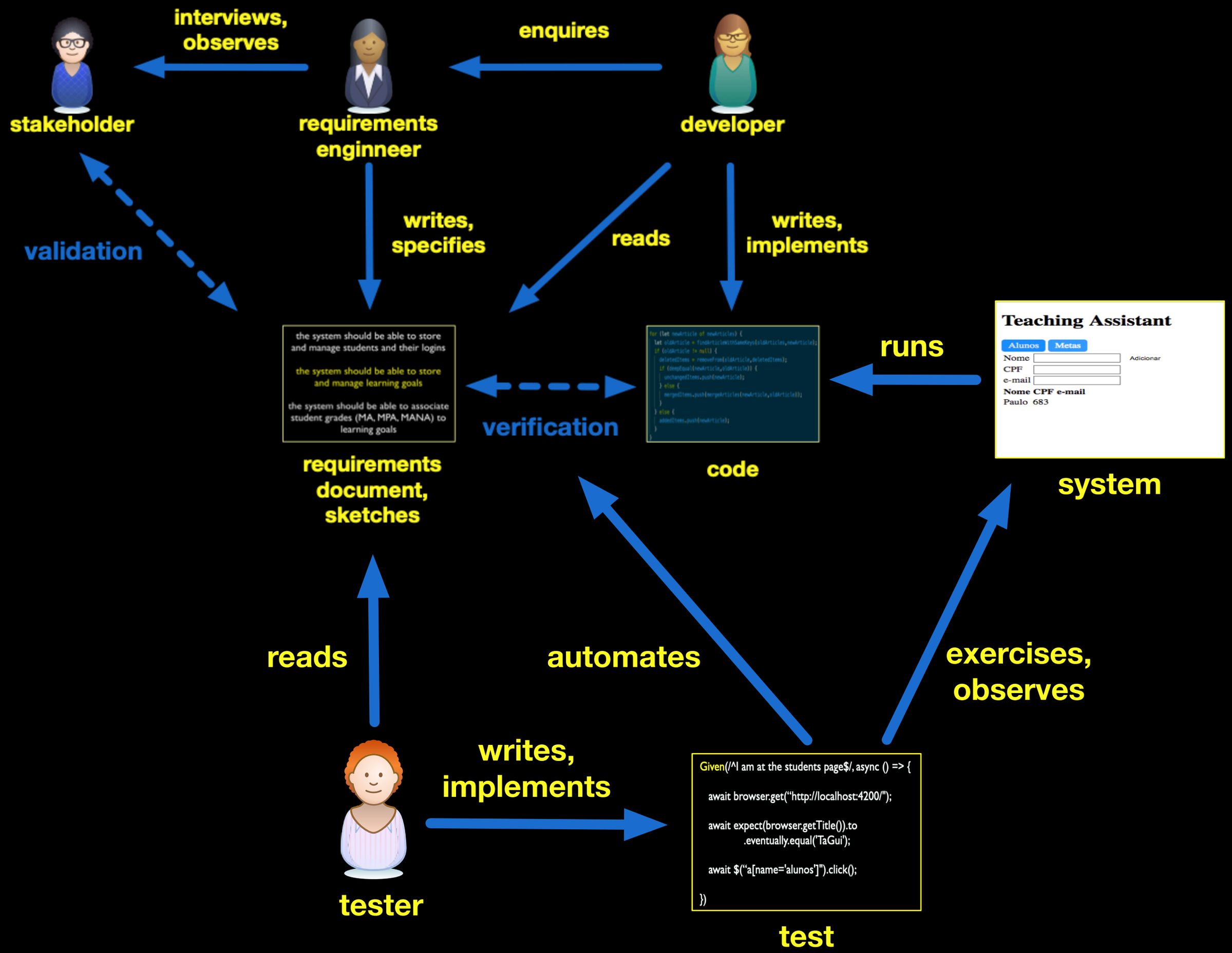
- quality "assurance", in theory
- evidence that system behaves as expected in specific situations, in practice

evidence is as strong as the test suite

supports not only bug detection, but also bug fixing, and refactoring

Behaves as expected…

- functionalities (correctness)
- robustness
- performance and scalability
- presentation and GUI (usability)
- security
- …


interviews, observes
enquires
stakeholder
requirements enginneer
developer
validation
writes, specifies
reads
writes, implements
the system should be able to store and manage students and their logins
the system should be able to store and manage learning goals
the system should be able to associate student grades (MA, MPA, MANA) to learning goals
verification
runs
Teaching Assistant
Alunos
Metas
Nome
CPF
e-mail
Adicionar
Nome CPF e-mail
Paulo 683
requirements document, sketches
code
system
reads
automates
exercises, observes
writes, implements
Given(/^I am at the students page$/, async () => {
await browser.get("http://localhost:4200/");
await expect(browser.getTitle()).to .eventually.equal('TaGui');
await $("a[name='alunos']").click();
})
tester
test

Software Engineering

Paulo Borba
Informatics Center
Federal University of Pernambuco

pauloborba.cin.ufpe.br

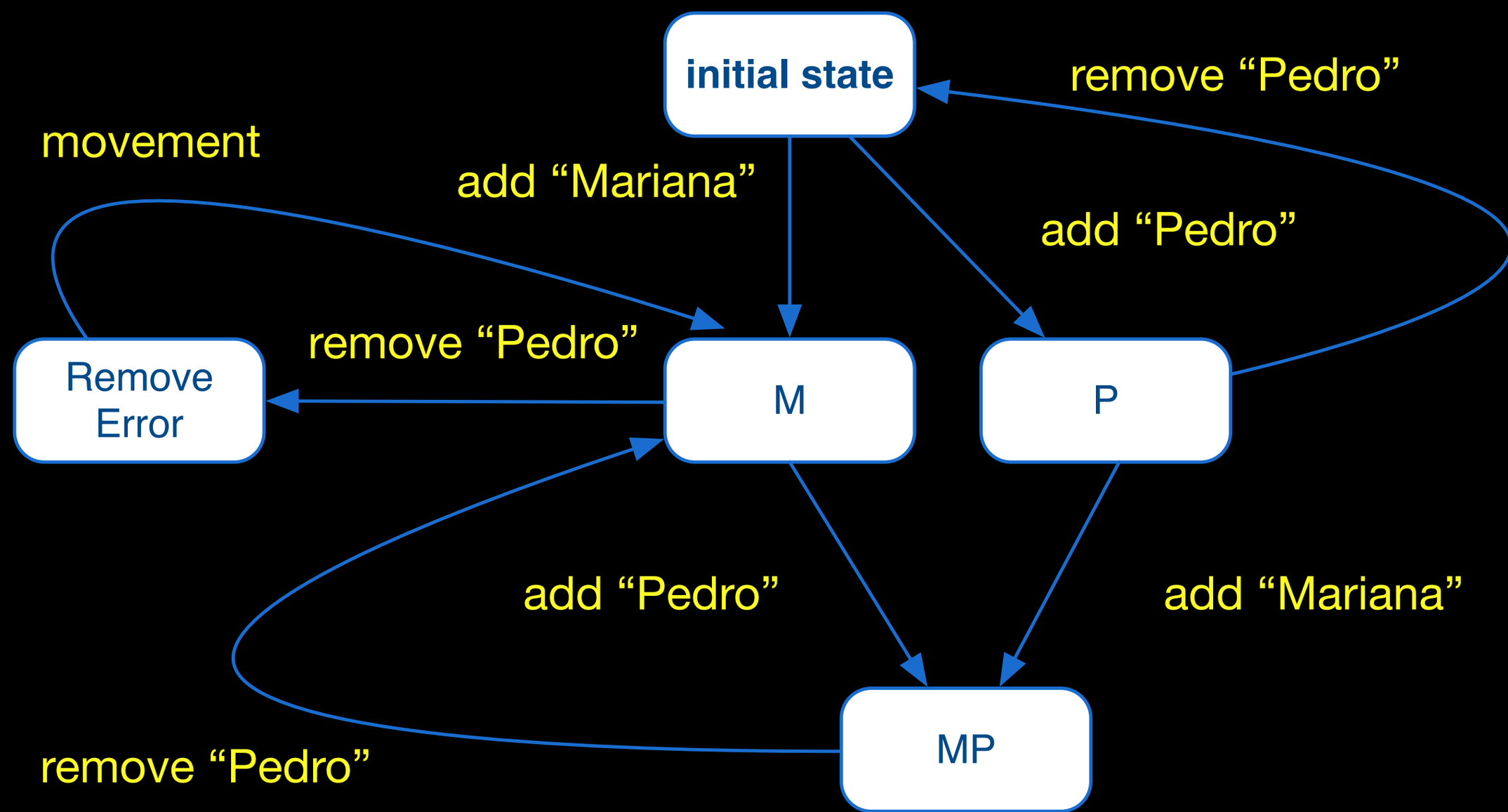
What are acceptance tests?

Acceptance tests

Test the system as a whole

Stakeholders accept system release if such tests pass

System, class, module, etc. as a state machine



Test as path (often of size one) in the corresponding graph

Directly connecting requirements (scenarios) and tests

GUI based test scenario

Scenario: adding new grade
Given I am at the "Grades" page
And I see a student "Maria Silva" with no grade for learning goal "Write quality tests"
When I add grade "MA" for "Maria Silva" goal "Write quality tests"
Then I'm still at the "Grades" page
And I can see student "Maria Silva" has grade "MA" for learning goal "Write quality tests"

Test structure

setup, input

Scenario: adding new grade
Given I am at the "Grades" page
And I see a student "Maria Silva" with no grade for learning goal "Write quality tests"
When I add grade "MA" for "Maria Silva" goal "Write quality tests"
Then I'm still at the "Grades" page
And I can see student "Maria Silva" has grade "MA" for learning goal "Write quality tests"

test actions

expected results

```
Feature: A manager updates account information for another user

  In order to correct mistakes a user has made
  As a Manager
  I want to update a users account information

  Background:
    Given The default settings and jnlp resources exist using factories
    And the database has been seeded

  Scenario Outline: Managers can change a users email address
    When I am logged in with the username mymanager
    And I am on the user preferences page for the user "<username>"
    Then I should see "User Preferences"
    When I fill in "user_email" with "<changed_email>"
    And I press "Save"
    When I am on the user preferences page for the user "<username>"
    Then I should see "User Preferences"
    And the "user_email" field should contain "<changed_email>"

    Examples:
      | username | changed_email          |
      | student  | test1@mailintator.com  |
      | teacher  | test2@mailintator.com  |

@javascript
Scenario Outline: Managers can change a users password
  When I am logged in with the username mymanager
  And I am on the user list page
  And I click "Reset Password" for user: "<userlogin>"
  Then I should see "Password for <username> (<userlogin>)"
  When I fill in "user_reset_password_password" with "<new_password>"
  And I fill in "user_reset_password_password_confirmation" with "<new_password>"
  And I press "Save"
  Then I should be on user list
  When I log out
  And I login with username: <userlogin> password: <new_password>
  Then I should see "Welcome"
  And I should see "SETTINGS"
```

One test suite

Two test scenarios

Three tests, or test cases

https://github.com/concord-consortium/rigse/blob/master/features/manager_updates_account_information.feature

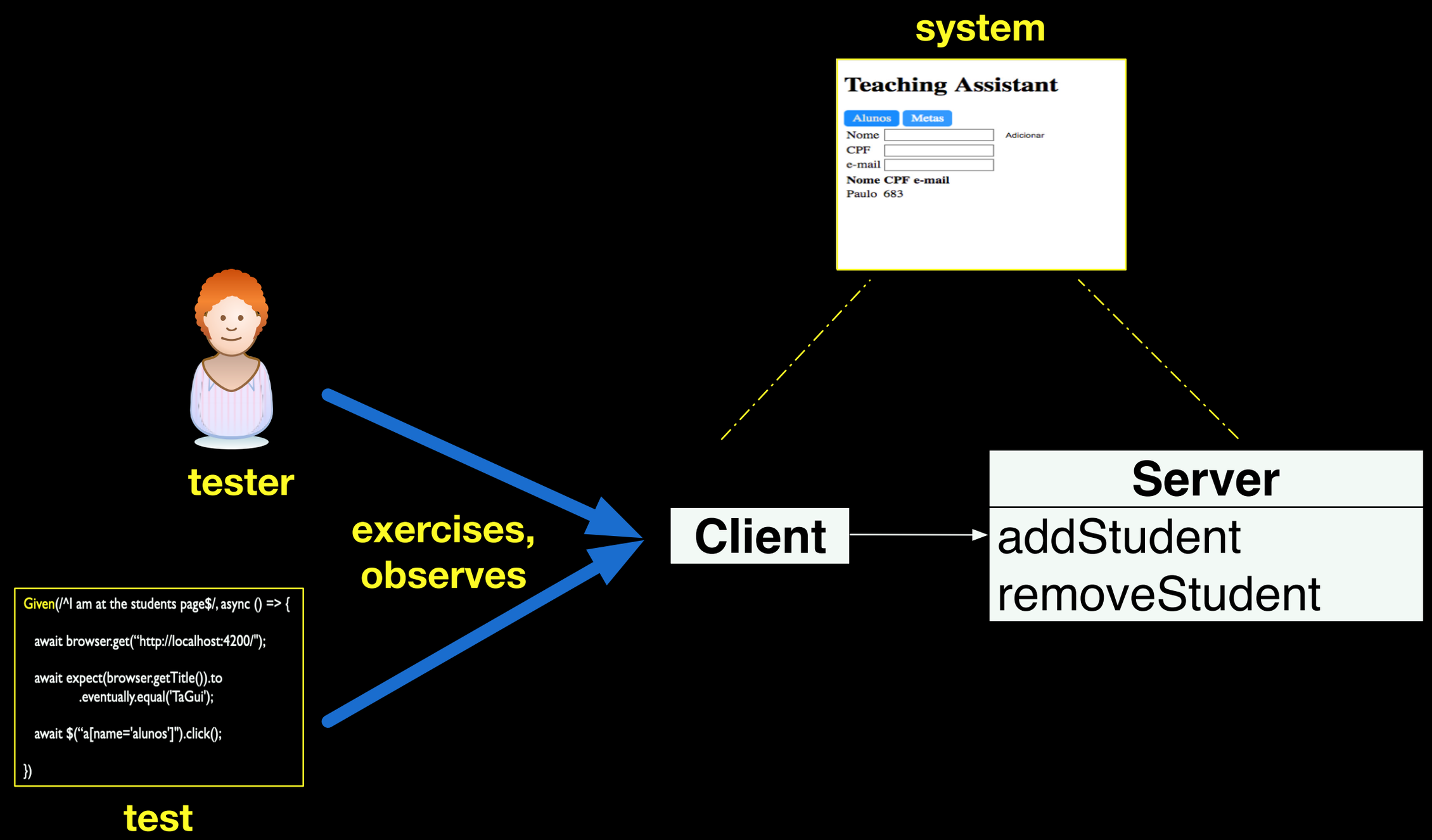
Software Engineering

Paulo Borba
Informatics Center
Federal University of Pernambuco

pauloborba.cin.ufpe.br

What are GUI acceptance tests?

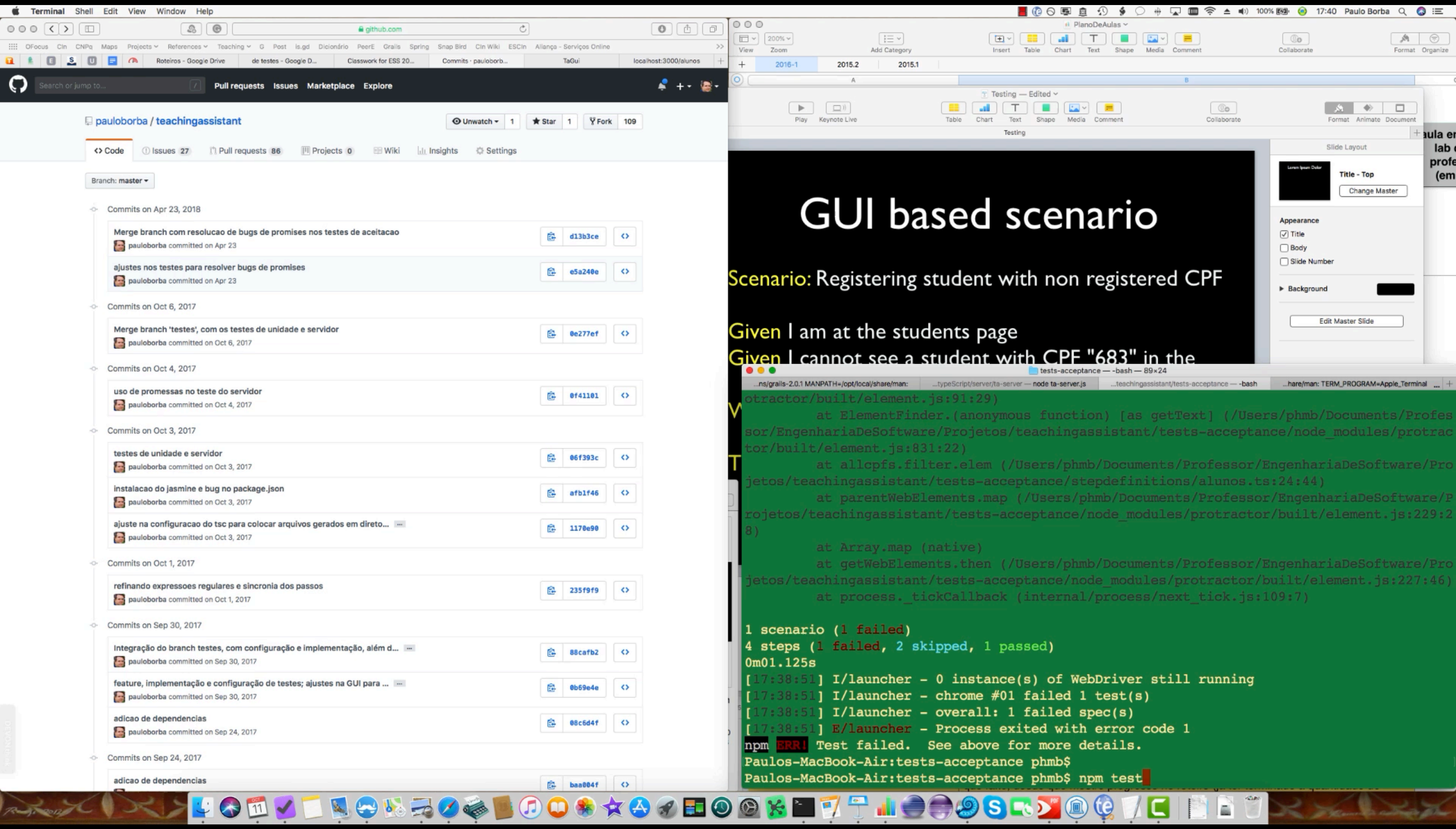
Exercising the system through its GUI


system
Teaching Assistant
Alunos
Metas
Nome
Adicionar
CPF
e-mail
Nome CPF e-mail
Paulo 683
tester
exercises,
observes
Client
Server
addStudent
removeStudent
Given(/^I am at the students page$/, async () => {
await browser.get("http://localhost:4200/");
await expect(browser.getTitle()).to
.eventually.equal('TaGui');
await $("a[name='alunos']").click();
})
test

Automated vs manual tests

- expensive to create
- inexpensive to execute
- enables regression testing, continuous testing
- some might be hard to create (complicated oracle)

- expensive to execute
- boring, failures might be missed
- avoids oracle problem



Executing tests

GUI based scenario

Scenario: Registering student with non registered CPF

Given I am at the students page
Given I cannot see a student with CPF "683" in the students list
When I try to register the student "Paulo" with CPF "683"
Then I can see "Paulo" with CPF "683" in the students list

GUI based test step

```
Given(/^I am at the students page$/, async () => {

  await browser.get(“http://localhost:4200/");

  await expect(browser.getTitle()).to
              .eventually.equal('TaGui');

  await $(“a[name='alunos']").click();

})
```

Steps exercise the system under test (SUT) by simulating user actions in the browser

Software Engineering

Paulo Borba
Informatics Center
Federal University of Pernambuco

pauloborba.cin.ufpe.br

How to implement GUI acceptance tests?

For each commit…

- Check commit changes
- Starting at commit “adicao de nomes nos elementos HTML a serem referenciados pelos testes”, going up to “criacao da feature e testes de alunos”

Hands on exercises

Testing 1: implementation, maintenance and execution

Testing 2: implementation, maintenance and execution

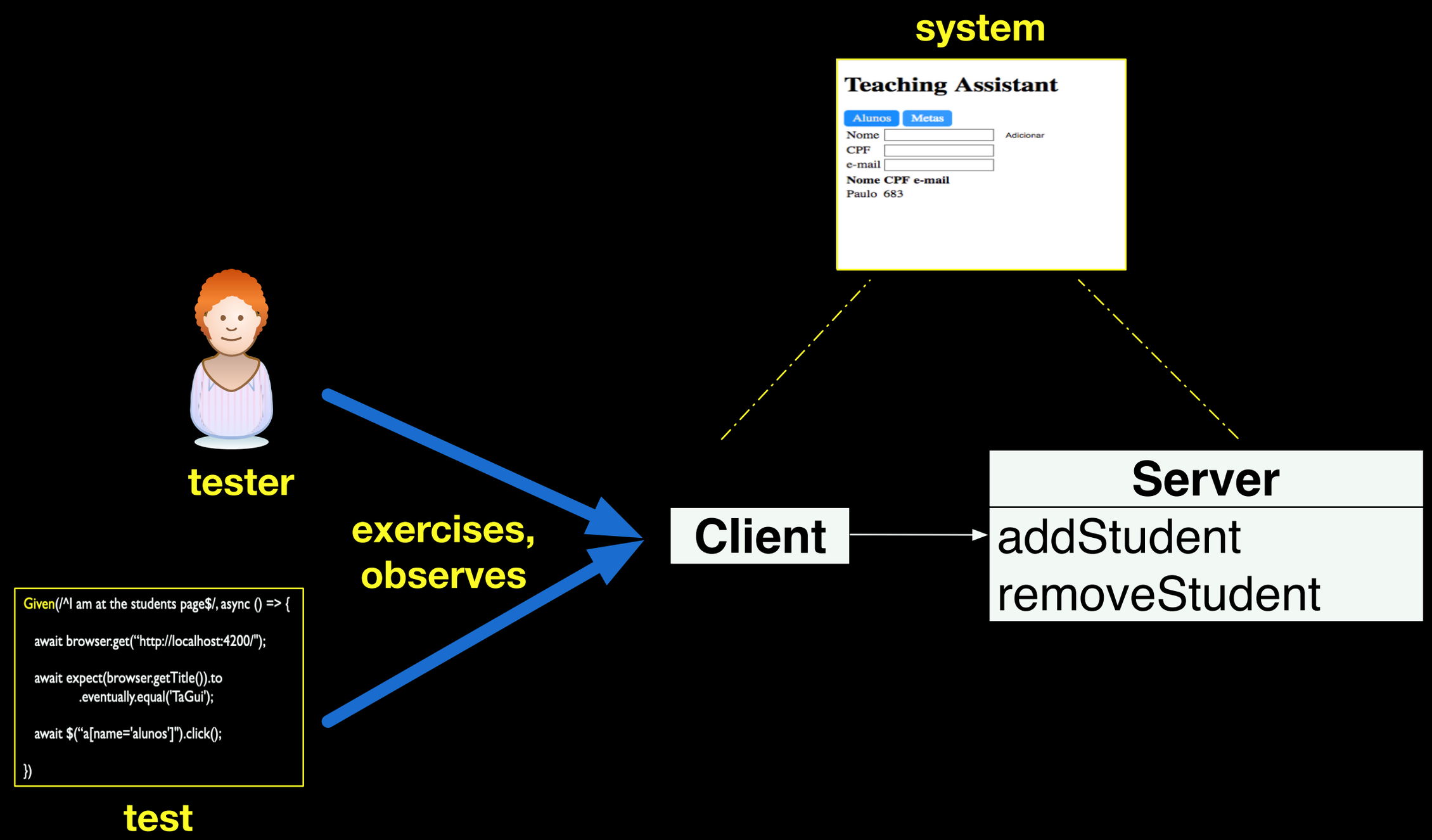
Software Engineering

Paulo Borba
Informatics Center
Federal University of Pernambuco

pauloborba.cin.ufpe.br

What are service acceptance tests?

Exercising the system through its server


system
Teaching Assistant
Alunos
Metas
Nome
Adicionar
CPF
e-mail
Nome CPF e-mail
Paulo 683
tester
exercises,
observes
Client
Server
addStudent
removeStudent
Given(/^I am at the students page$/, async () => {
await browser.get("http://localhost:4200/");
await expect(browser.getTitle()).to
.eventually.equal('TaGui');
await $("a[name='alunos']").click();
})
test

system

Client

Server

addStudent
removeStudent

exercises,
observes

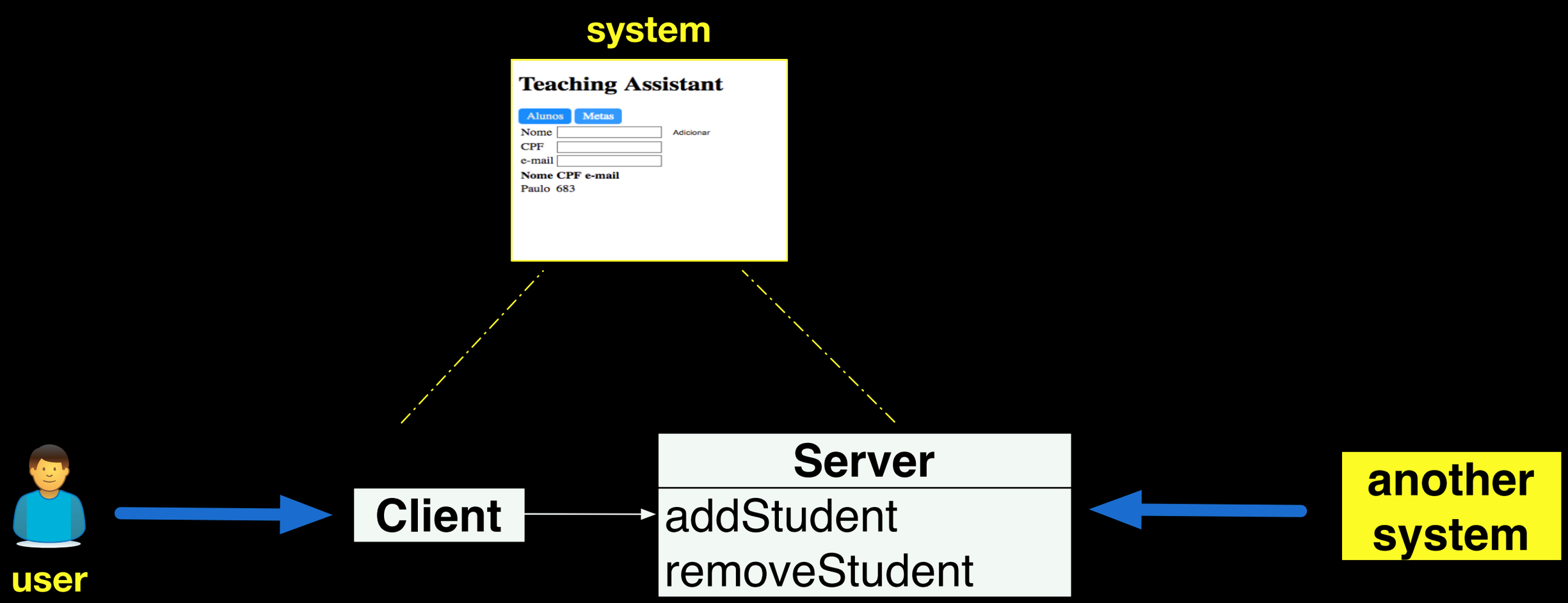
tester

```
Given(/^I am at the students page$/, async () => {

  await browser.get("http://localhost:4200/");

  await expect(browser.getTitle()).to
          .eventually.equal('TaGui');

  await $("a[name='alunos']").click();

})
```

test

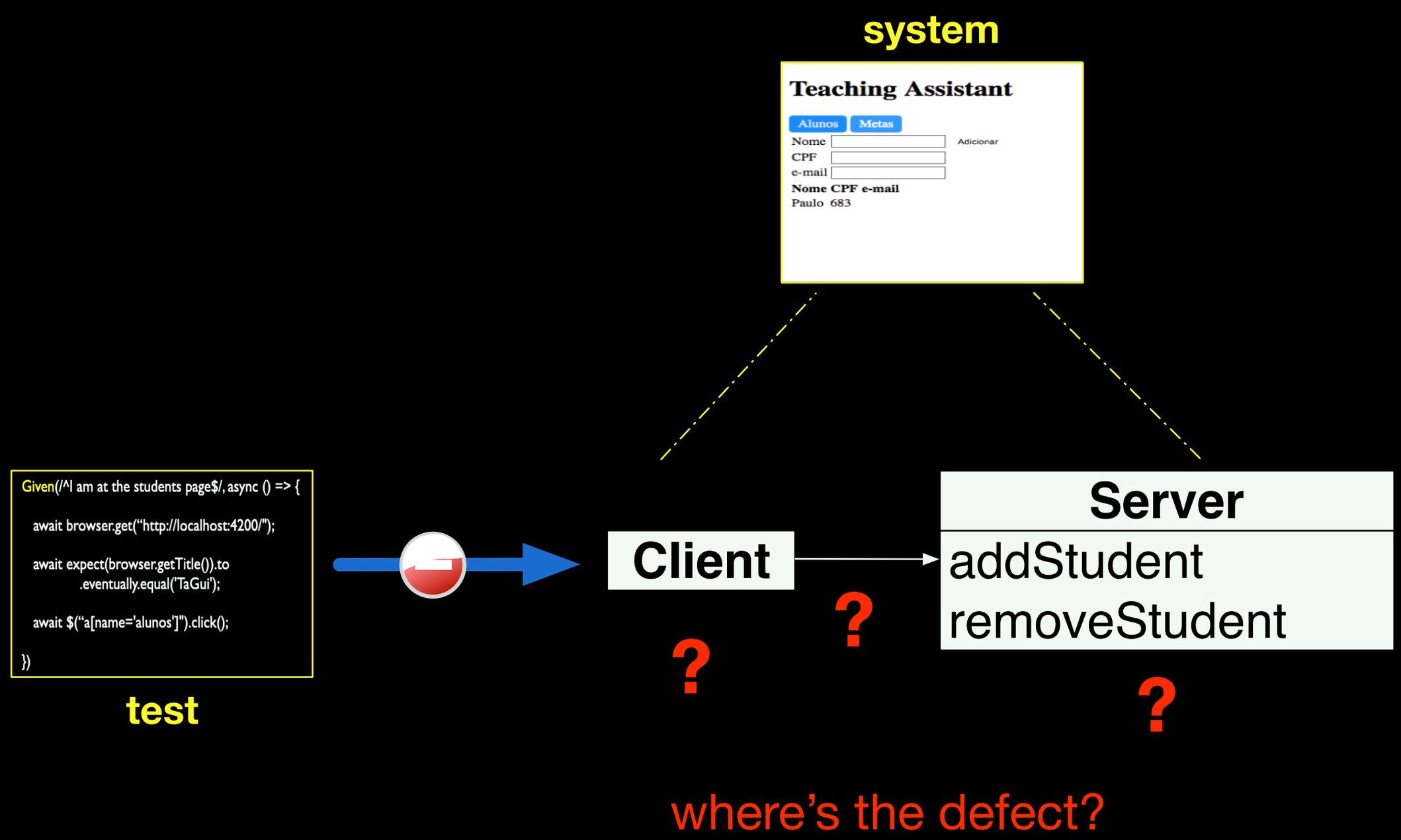
Why both GUI and service acceptance tests?

Make sure that system is OK for both users and other systems

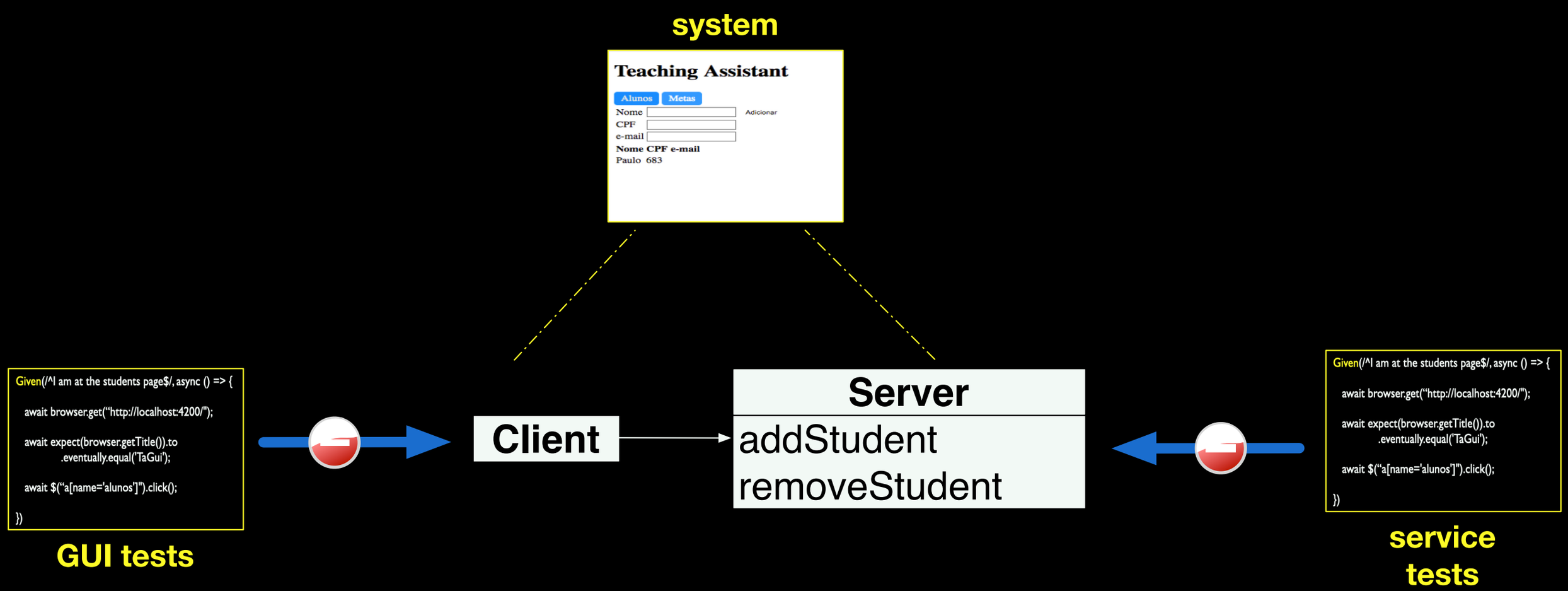


Not always needed, so consider your context

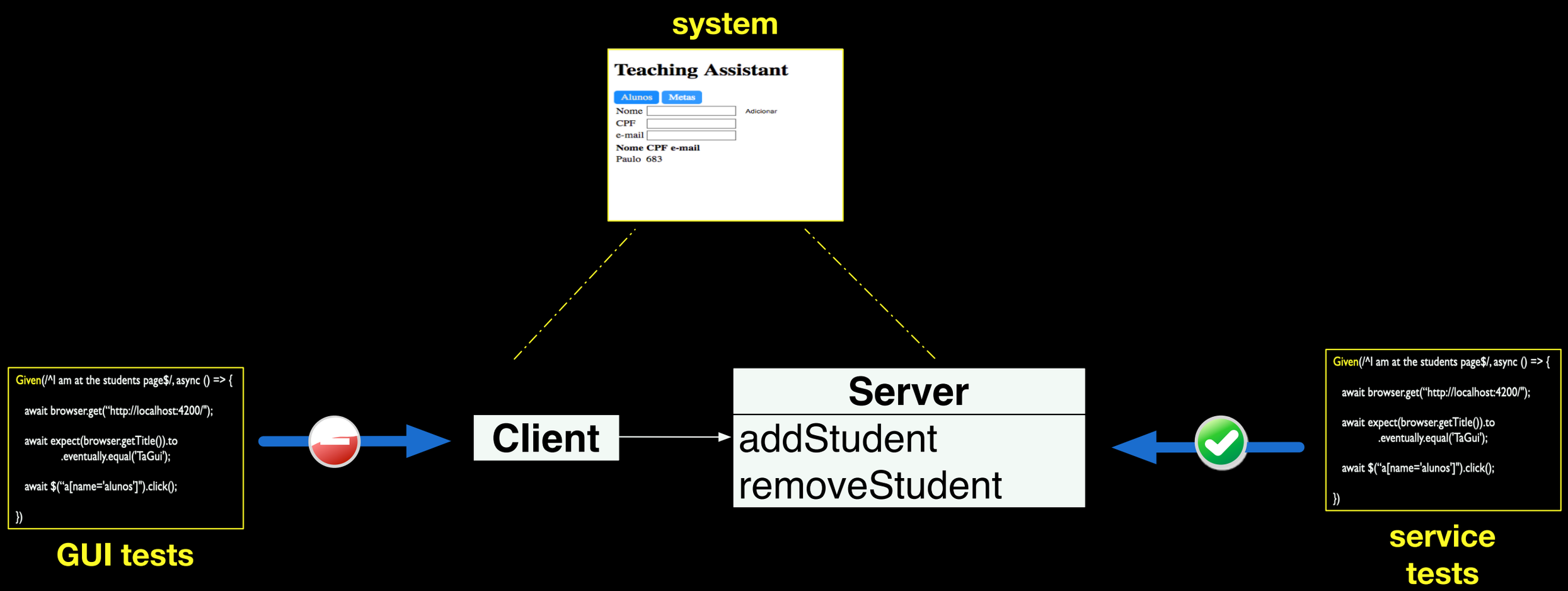
Help locating defects



Problem likely in the server



Problem likely in the client



Service based scenario

Scenario: Registering student with non registered CPF, service

Given the system has no student with CPF "683"
When I register the student "Paulo" with CPF "683"
Then the system now stores "Paulo" with CPF "683"

Service based test step

```
Given(/^the system has no student with CPF "(\d*)"$/,
        async (cpf) => {

      await request.get(base_url + "alunos")
         .then(body =>
            expect(body.includes(`"cpf":"${cpf}"`))
               .to.equal(false));


   });
})
```

Given can possibly establish the pre-condition

Steps exercise the system under test (SUT) by invoking services

Software Engineering

Paulo Borba
Informatics Center
Federal University of Pernambuco

pauloborba.cin.ufpe.br

How to implement service acceptance tests?

For each commit…

- Check commit changes
- Starting at commit “testes de aceitacao do servidor”, going up to “corrigindo teste de aceitacao de servico: usar parametro ao inves de …”

Software Engineering

Paulo Borba
Informatics Center
Federal University of Pernambuco

pauloborba.cin.ufpe.br

How to review cucumber tests?

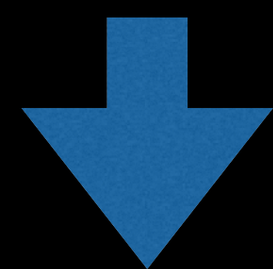
Checklist

Test behavior should strictly conform to scenario semantics

```
Given(/^I am at the students page$/, async () => {
  await browser.get(“http://localhost:4200/");
  await expect(browser.getTitle()).to
            .eventually.equal('TaGui');
  await $(“a[name='alunos']").click();
  await expect(browser.getTitle()).to
            .eventually.equal(‘Students');
})
```

Avoid ambiguities due to similar step sentences

```
Given I cannot see a student with CPF "683"
      in the students list
```



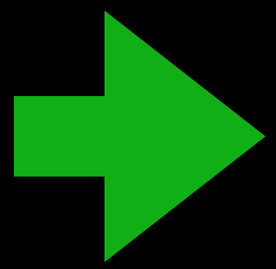
```
/^I cannot see a student with CPF "(\d*)"
  in the students list$/
```

a scenario
description step should match
only one expression in the step
implementations

Do not duplicate test code

```
def fillLoginDataOnly(...) {
        $("form").username = username
        $("form").password = password
}
```

```
def fillLoginDataAndSubmit(...) {
        $("form").username = username
        $("form").password = password
        $("form").click()
}
```



```
def fillLoginDataAndSubmit(...) {
        fillLoginDataOnly(...)
        $("form").click()
}
```

Tests should clean up environment at the end

...

```
def uploadsFolder = new File(...)
uploadsFolder.listFiles().each {
  innerFile =>
  innerFile.deleteOnExit()
}
```

Tests should be platform (including browser) and language independent

```
await $("input[name='cpfbox']").sendKeys(<string> cpf);
await element(by.buttonText('Adicionar')).click();
```

```
await $("input[name='cpfbox']").sendKeys(<string> cpf);
await element(by.buttonText(
                    prop.getButtonAddStudent()
            )).click();
```

Hands on exercises

Testing 2: implementation, maintenance and execution

Testing 3: implementation, maintenance and execution

Software Engineering

Paulo Borba
Informatics Center
Federal University of Pernambuco

pauloborba.cin.ufpe.br

Which are the basic types of test?

Many test classifications!

Some are orthogonal…

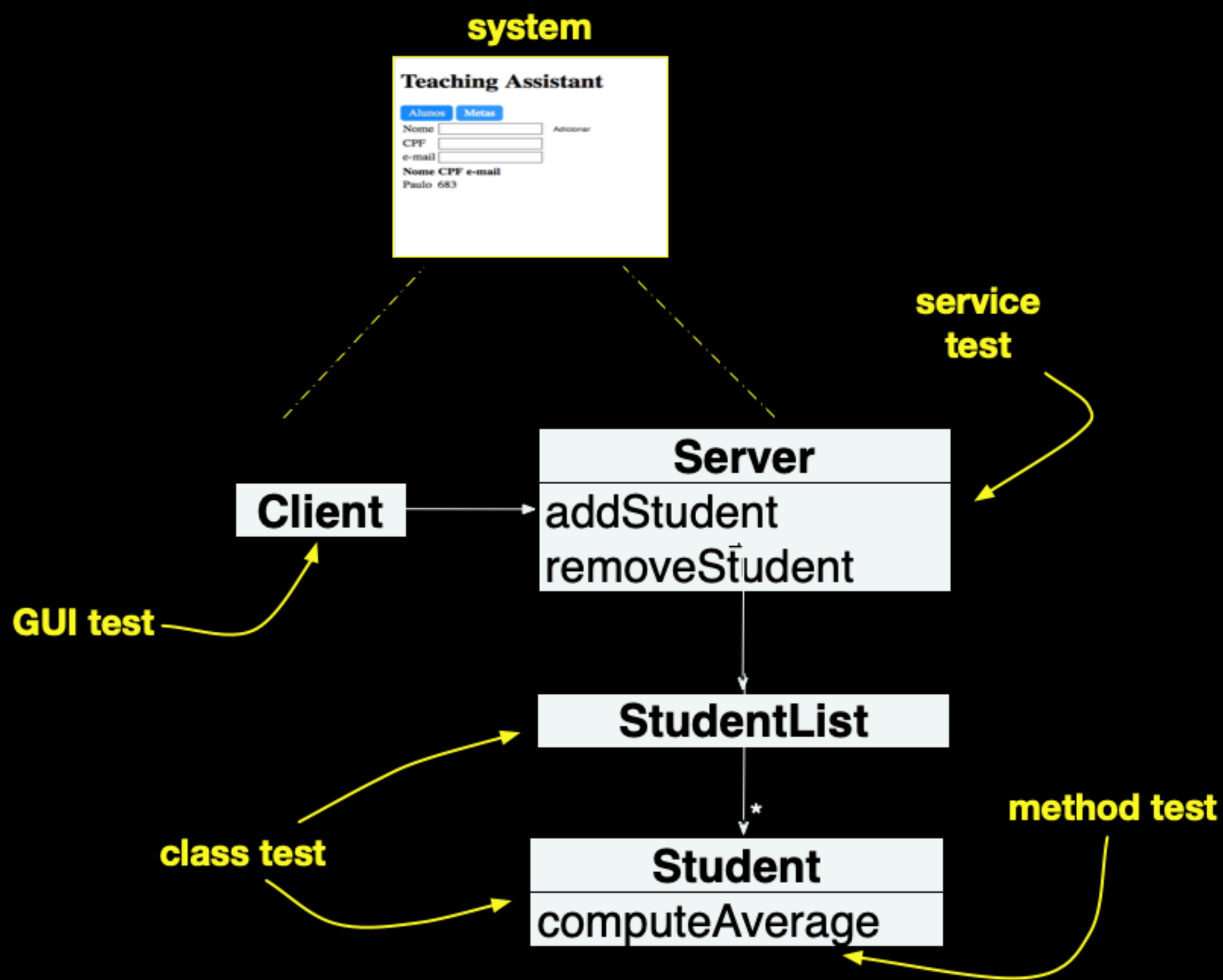
Not much consistency in the literature

Quality factor to be verified by the test

1. Correctness (functional)
2. Performance
3. Security
4. Usability (A/B)
5. Robustness
6. Compatibility
7. …

Programming entity directly exercised by the test

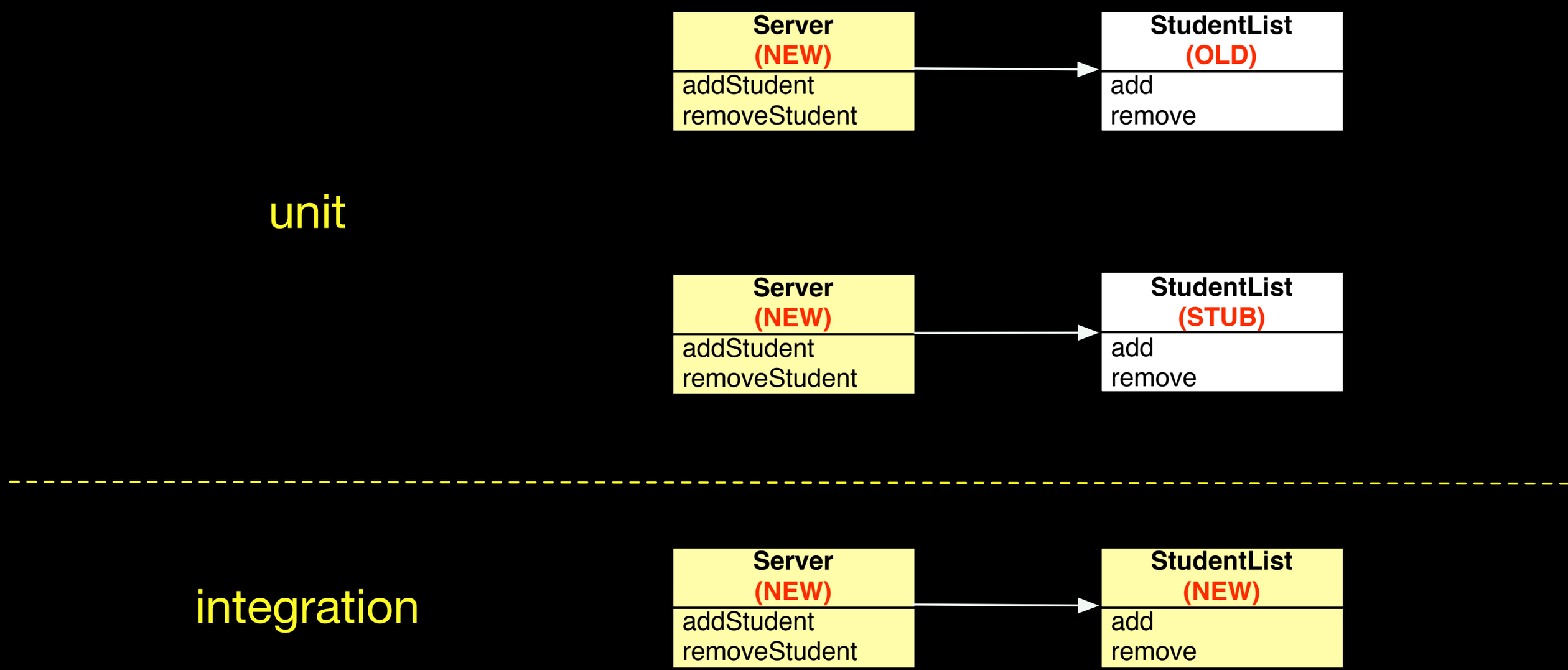
1. GUI
2. Class
3. Method
4. Service
5. …


system
Teaching Assistant
Alunos
Metas
Nome
CPF
e-mail
Adicionar
Nome CPF e-mail
Paulo 683
service test
Server
addStudent
removeStudent
Client
GUI test
StudentList
*
class test
method test
Student
computeAverage

Dependencies used to run the test

1. Unit: old version of dependencies, or even stubs and drivers
2. Integration: new version of dependencies evolved by other developers
3. End-to-end: as in integration, but all dependencies involved
4. System: as in end-to-end, but external dependencies from the production environment

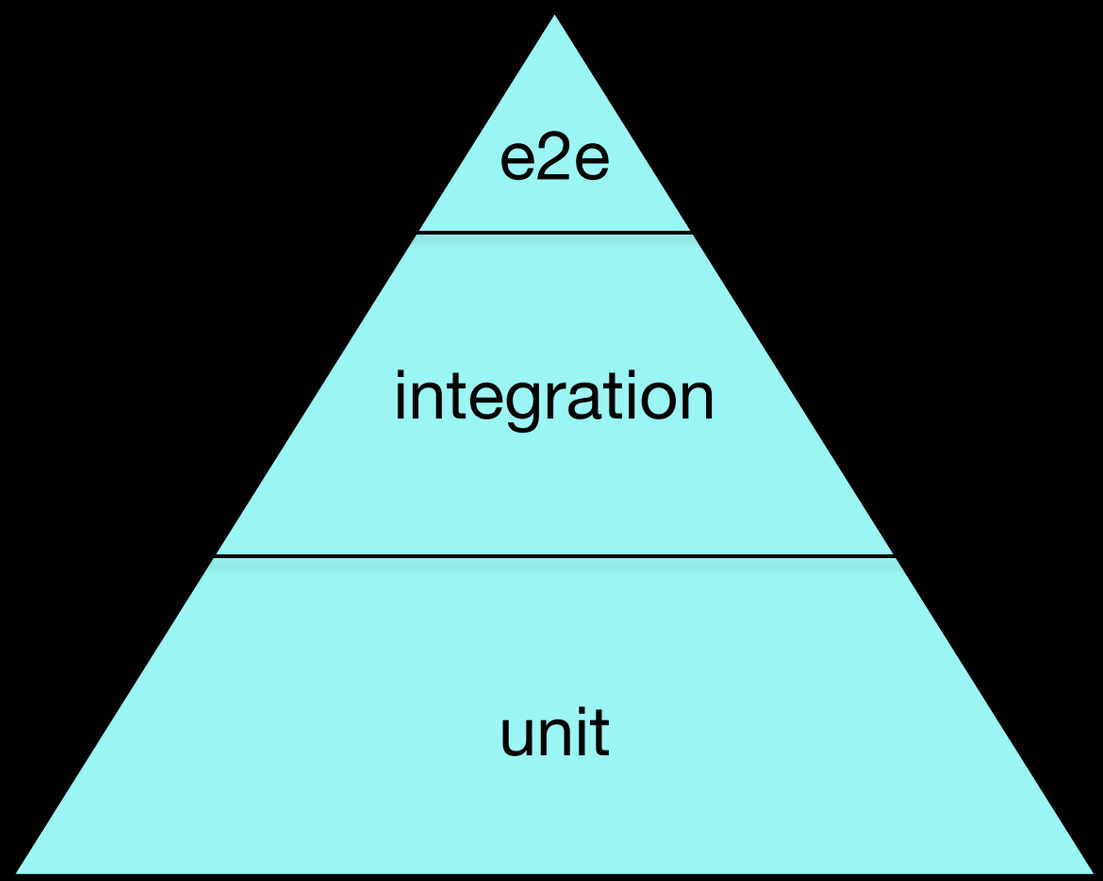
Assuming a class and its dependence are independently evolving…

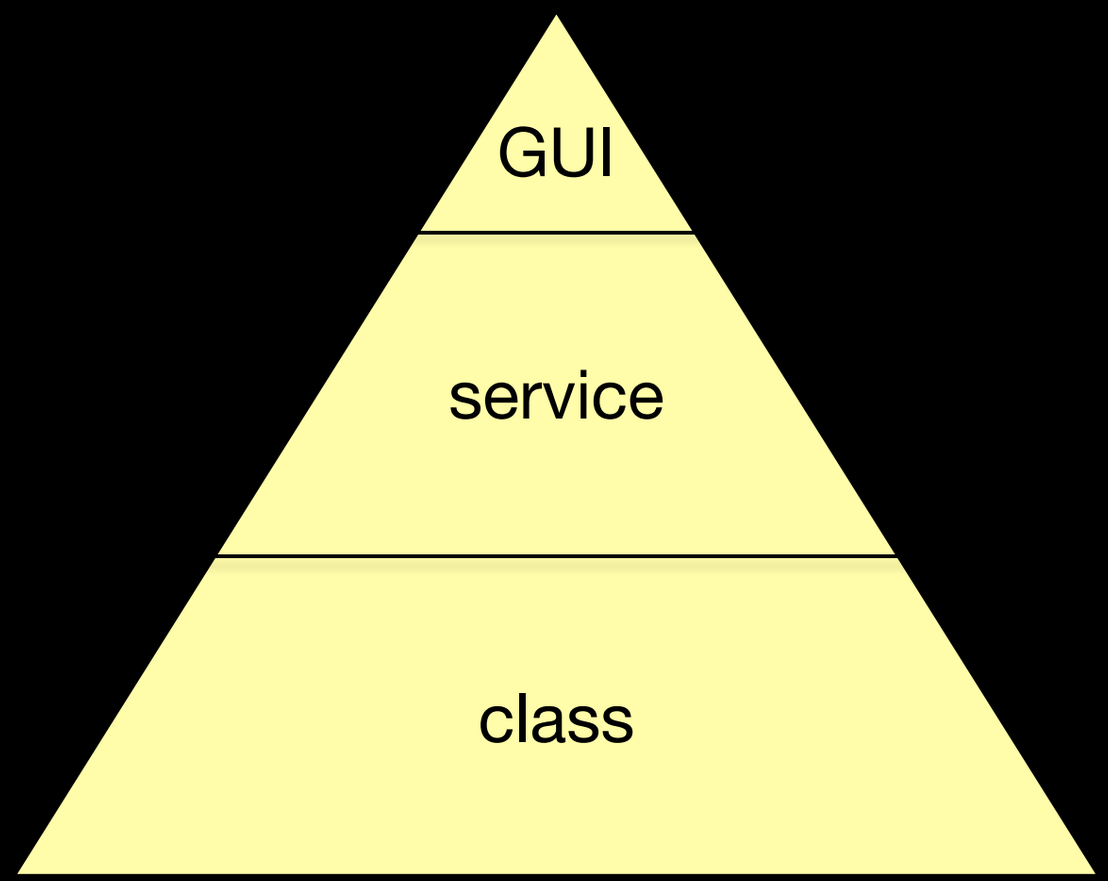


Unit test gives no guarantee that system works after code integration

Integration test gives such guarantee, but doesn't help to locate error or to give weaker guarantees early on

Test pyramids





One can also have test suites with different shapes, for different situations: before a commit, after merging, before a release, etc.

Guarantee provided by the test

1. Acceptance: the tests justify product acceptance, play the role of a contract
2. Regression: the tests give evidence that previously tested behavior was not affected
3. Smoke: the tests give evidence that basic system behavior is OK

Software Engineering

Paulo Borba
Informatics Center
Federal University of Pernambuco

pauloborba.cin.ufpe.br

What are (non acceptance) service tests?

Non acceptance service tests

- directly exercises services
- not directly derived from explicit requirements
- regression or smoke, depending on complexity and in which suite it is included
- functional
- unit, integration, or system, depending on selected dependencies

Service test, no link with scenarios

```
describe("O servidor", () => {

  it("inicialmente retorna uma lista de alunos vazia", () => {
    return request.get(base_url + "alunos").then(body =>
      expect(body).toBe("[]")).catch(e =>
      expect(e).toEqual(null));
  })
```

test action

expected result

Might check implicit requirements

```
it("só cadastra alunos", () => {
    var options = {method: 'POST', uri: (base_url + "aluno"),
                   body: {name: "Mari", cpf: "962"},
                   json: true};
    return request(options)
        .then(body =>
            expect(body).toEqual({failure: "O aluno não
                                  pode ser cadastrado”}))
        .catch(e => expect(e).toEqual(null))
});
```

Common actions and state for tests in a suite

```
var server:any;

beforeAll(() => {
    server = require('../ta-server')
});

afterAll(() => {
    server.closeServer()
});
```

test suite setup

test suite cleanup

Code exercises the system under test (SUT) by invoking services

Not driven by scenarios, but by properties developers want to check (component interfaces/ contracts, assumptions, etc.)

Software Engineering

Paulo Borba
Informatics Center
Federal University of Pernambuco

pauloborba.cin.ufpe.br

How to implement non acceptance service tests?

For each commit…

- Check commit changes
- Starting at commit ajustes na configuracao para testes do servidor”, going up to “testes de servidor”

Hands on exercises

Testing 3: implementation, maintenance and execution

Testing 4: implementation, maintenance and execution

Software Engineering

Paulo Borba
Informatics Center
Federal University of Pernambuco

pauloborba.cin.ufpe.br

How to implement class tests? (unit, integration)

Class test, no link with scenarios

```
describe("O cadastro de alunos", () => {
  var cadastro: CadastroDeAlunos;

  beforeEach(() => cadastro = new CadastroDeAlunos())

  it("é inicialmente vazio", () => {
    expect(cadastro.getAlunos().length).toBe(0);
  })
```

test setup

simple
test, no action

```
it("doesn't accept students with the same CPF ", () => {
    var student: Student = new Student();
    student.name = "Mariana";
    student.cpf = "683";
    cadastro.add(student);

    student = new Student();
    student.name = "Pedro";
    student.cpf = "683";
    cadastro.add(student);

    expect(cadastro.getStudents().length).toBe(1);
})
```

object creation and initialisation, test setup, input

method calls, test actions

expected result assertions

Code exercises the system under test (SUT) by creating objects and invoking methods

Not driven by scenarios, but by properties developers want to check (method specifications, invariants, etc.)

Software Engineering

Paulo Borba
Informatics Center
Federal University of Pernambuco

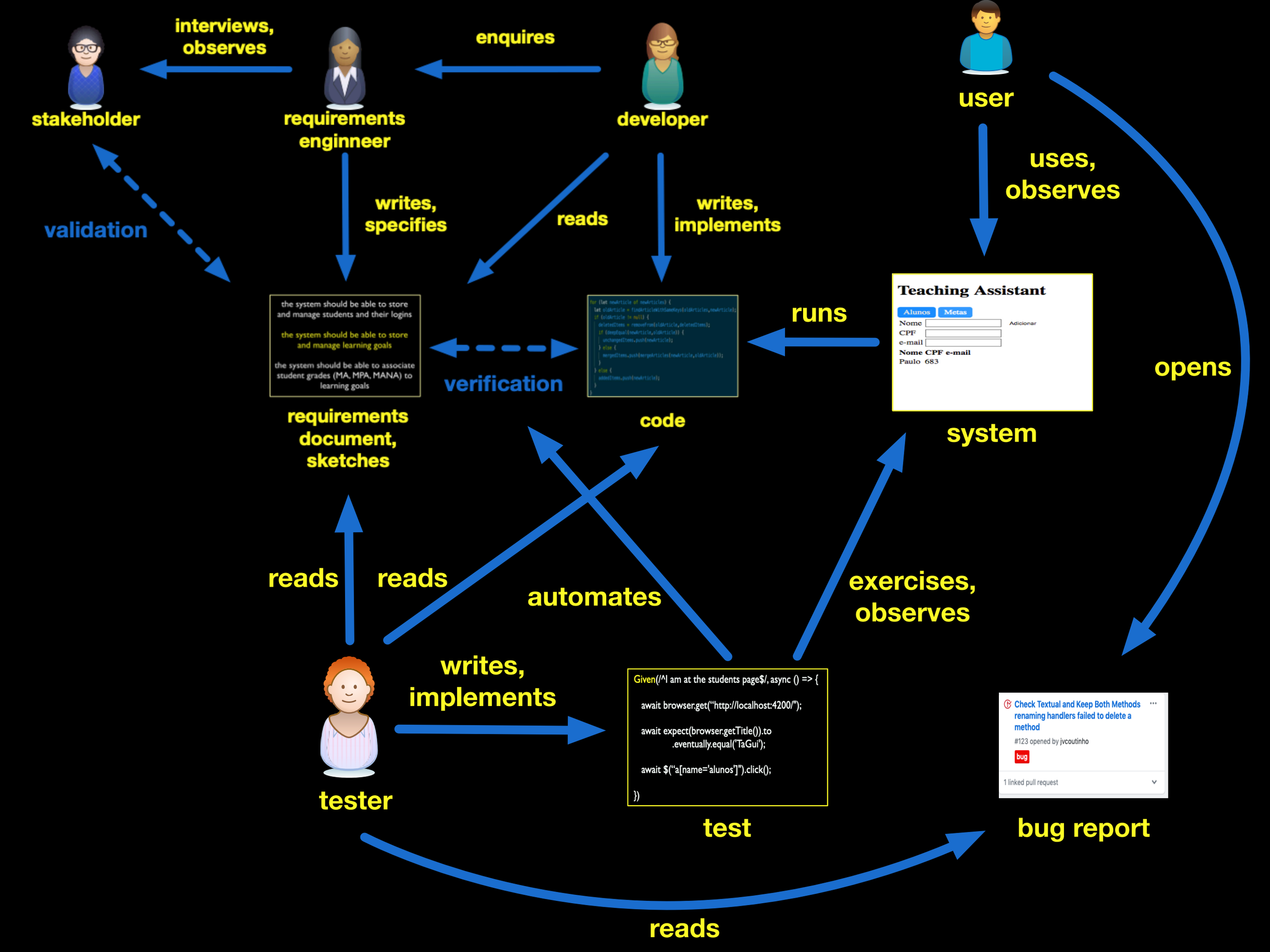
pauloborba.cin.ufpe.br

How to design tests?

Write tests right

versus

Write the right tests


stakeholder
interviews, observes
requirements enginneer
enquires
developer
user
validation
writes, specifies
reads
writes, implements
uses, observes
the system should be able to store and manage students and their logins
the system should be able to store and manage learning goals
the system should be able to associate student grades (MA, MPA, MANA) to learning goals
verification
runs
Teaching Assistant
Alunos
Metas
Nome
CPF
e-mail
Adicionar
Nome CPF e-mail
Paulo 683
requirements document, sketches
code
system
opens
reads
reads
automates
exercises, observes
writes, implements
Given(/^I am at the students page$/, async () => {
await browser.get("http://localhost:4200/");
await expect(browser.getTitle()).to
.eventually.equal('TaGui');
await $("a[name='alunos']").click();
})
Check Textual and Keep Both Methods renaming handlers failed to delete a method
#123 opened by jvcoutinho
bug
1 linked pull request
tester
test
bug report
reads

You won't have time to fully test the implemented functionality

So invest your time on tests that are relevant, exercise more critical behavior and states

not working properly will lead to significant problems (loss, complaints, etc.)

Focus on exercising more relevant (for both the stakeholders and the project) and complex behavior and states

- requirements, rules,
- constraints, event sequences,
- cases, etc.

not sure if it was completely understood or was implemented right

More principles

Weakening the pre condition

The more input states are exercised by your test suite, the stronger guarantees are provided by the suite

Exercise the same operations and functionality with different inputs (arguments and state content)

- focus on representative inputs
- boundary cases

  empty - full - some elements

  positive - negative - 0

  one MANA - no MANA - no MA
- aim is to increase coverage, avoiding redundancy

Input space is often multi-dimensional

- Hardware platforms (devices)
- Operating system, browser, library and auxiliary system versions, environment variables
- Product configurations and options, for configurable systems and product lines
- Processes, queues (notifications), network latency

Strengthening the post condition

The more constraints in the expected results of a test, the stronger guarantees are provided by the test

Checking if a report was generated…

- file report.pdf exists and is non empty?
- and is a valid PDF file?
- and does not contain page numbers?
- and contains a section on student evaluation?
- and this section contents is compatible with the data currently stored by the system?

positive verification

negative verification

Right balance between the different kinds of tests

- GUI, service, and class tests
- Unit, integration, and end-to-end tests
- and their combinations

Each test should run independently of the others

- assuming a fresh system instance
- without assuming or leaving side-effects
- with the aim of reusability and compositionality

if deletion test depend on the insertion test, we won't be able to execute the first alone

We should try to have the same for test steps, but one step might have to setup state for another step

Regression testing

Before pushing (sometimes committing), make sure all tests pass

Practices

Tests as partial specifications

Behavior driven design

acceptance (GUI or service) test implementation before feature implementation

Test driven design

class test
implementation before
class or method
implementation

Create interface when little functionality is available

```
static public void createArticle(String title, filename) {
    def cont = new PeriodicoController()
    cont.params << TestData.findArticleByTitle(title)
                << [file: filename]
    cont.request.setContent(new byte[1000])
    cont.create()
    cont.save()
    cont.response.reset()
}
```

functionality interface

```
class PeriodicoController {
    create() {}
    save() {}
}
```

Interface is not always a list of method signatures

```
Given(/^I am at the students page$/, async () => {
    await browser.get("http://localhost:4200/");
    await expect(browser.getTitle()).to.eventually.equal('TaGui');
    await $("a[name='alunos']").click();
})
```

HTML interface

```
<a name="alunos">
</a>
```

Requires established architecture and code structure, but not the actual implementation

Debugging

- use a proper tool
- step back
- find the wrong assumption
- change code

Software Engineering

Paulo Borba
Informatics Center
Federal University of Pernambuco

pauloborba.cin.ufpe.br

How to implement class tests? (unit, integration)

For each commit…

- Check commit changes
- Starting at commit “ajustes na configuracao para testes de unidade”, going up to “refinamento do teste de CPF duplicado, e refatoracao”

Take notes, now!

Hands on exercises

Testing 4: implementation, maintenance and execution

Testing research at CIn

- Test generation and static analysis tools: Marcelo e Paulo
- Model-based testing: Alexandre Mota, Juliano e Augusto
- Test selection and execution: Juliano

To do after class

- Answer questionnaire (check classroom assignment), study correct answers
- Finish exercise (check classroom assignment), study correct answers
- Read, again, chapter 7 and basic concepts of chapter 6 in the textbook
- Evaluate classes (check classroom assignment)
- Study questions from previous exams

Questions from previous exams

- Explique brevemente a diferença entre testes de unidade e testes de integração (a). Qual o impacto negativo de realizar apenas os testes de unidade? (b) Qual o impacto negativo de realizar apenas os testes de integração?
- Explique brevemente a diferença entre testes de aceitação e testes de integração, e porque você acha que algumas empresas realizam os dois tipos de teste.

Software development

Paulo Borba
Informatics Center
Federal University of Pernambuco

phmb@cin.ufpe.br ◈ twitter.com/pauloborba