Testes automatizados em sistemas multi-agentes

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

DAS

CNPq

Testes

- Parte fundamental de qualquer software
- Permeia o desenvolvimento, em vez de ser uma parte separada do pipeline
- Complicado de se fazer:
  - Distribuído
  - Stack da aplicação

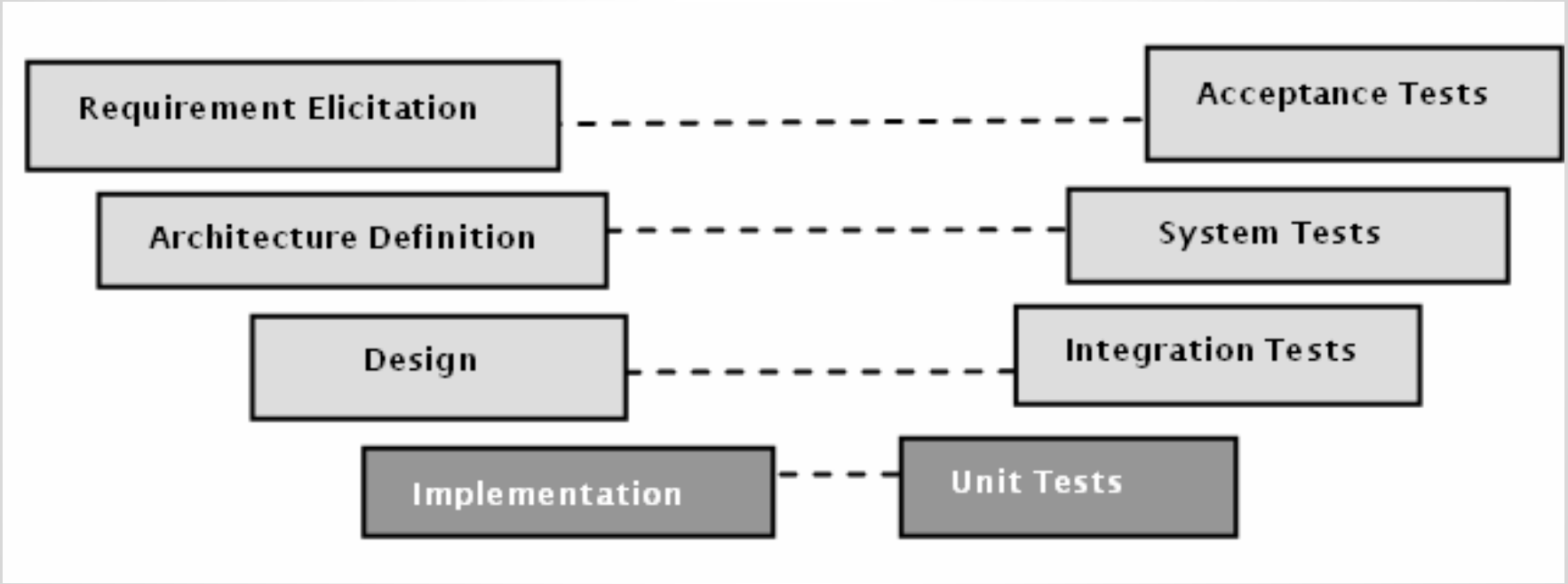
Históricamente

- Era a última fase do projeto
- Uma equipe especializada fazia parte do teste do software
- Ainda é assim para algumas áreas
  - Jogos, por exemplo, são muito difíceis de se testar a parte visual.

Testes Automatizados

Processo de testar um software com outro software.

- Muito usado em OOP
- Muitas metodologias usam como base
  - XP e TDD
- Útil para integração contínua
- Pouco explorado em SMA


Requirement Elicitation
Acceptance Tests
Architecture Definition
System Tests
Design
Integration Tests
Implementation
Unit Tests

Testes unitários

- Parte mais fundamental e base do teste
- Custoso
- Não garante uma integração, porém garante que cada parte está funcionando
- É rodada diversas vezes por dia/hora

Técnica - MOCK

- Mock, Stub ou dummy
  - Fazer uma classe "se passar" por outra

Mock

```
public class FinalInvoiceStep {

    private PrinterService printerService = null;
    private EmailService emailService = null;

    public FinalInvoiceStep(PrinterService printerService, EmailService emailService)
    {
        this.printerService = printerService;
        this.emailService = emailService;
    }

    public void handleInvoice(Invoice invoice, Customer customer)
    {
        if(customer.prefersEmails())
        {
            emailService.sendInvoice(invoice,customer.getEmail());
        }
        else
        {
            printerService.printInvoice(invoice);
        }
    }
}
```

```
public class FinalInvoiceStepTest {

    private FinalInvoiceStep finalInvoiceStep = null;
    private Customer customer = null;
    private Invoice invoice = null;

    @Before
    public void beforeEachTest() {
        customer = new Customer();
        finalInvoiceStep = new FinalInvoiceStep(Env.PrinterServiceLocator(),
                                                Env.EmailServiceLocator());
        invoice = new Invoice();

    }

    @Test
    public void normalCustomer() {
        customer.wantsEmail(true);
        finalInvoiceStep.handleInvoice(invoice, customer);

    }

    @Test
    public void customerWithPrintedInvoice() {
                        customer.wantsEmail(false);
        finalInvoiceStep.handleInvoice(invoice, customer);
    }
}
```

```java
public class DummyPrinterService  implements PrinterService{
    boolean anInvoiceWasPrinted = false;

        @Override
    public boolean isPrinterConfigured() {
        return true;
    }

        @Override
    public void printInvoice(Invoice invoice) {
        anInvoiceWasPrinted = true;
    }

    public boolean anInvoiceWasPrinted() {
        return anInvoiceWasPrinted;
    }
}
```

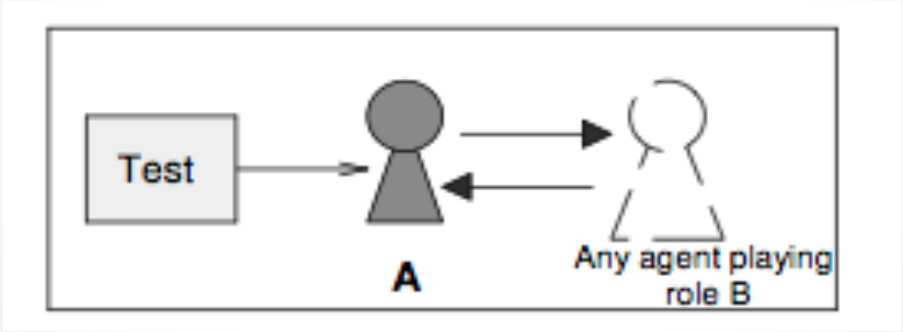
```java
public class FinalInvoiceStepTestImproved {

    private FinalInvoiceStep finalInvoiceStep = null;
    private Customer customer = null;
    private Invoice invoice = null;
    private DummyPrinterService dummyPrinterService = null;

    @Before
    public void beforeEachTest() {
        dummyPrinterService = new DummyPrinterService();
        customer = new Customer();
        finalInvoiceStep = new FinalInvoiceStep(dummyPrinterService, Env.EmailServiceLocator());
        invoice = new Invoice();

    }

    @Test
    public void normalCustomer() {
        customer.wantsEmail(true);
        finalInvoiceStep.handleInvoice(invoice, customer);
        assertFalse("Nothing should be printed",dummyPrinterService.anInvoiceWasPrinted());

    }

    @Test
    public void customerWithPrintedInvoice() {
        customer.wantsEmail(false);
        finalInvoiceStep.handleInvoice(invoice, customer);
        assertTrue("Invoice was printed",dummyPrinterService.anInvoiceWasPrinted());
    }
}
```

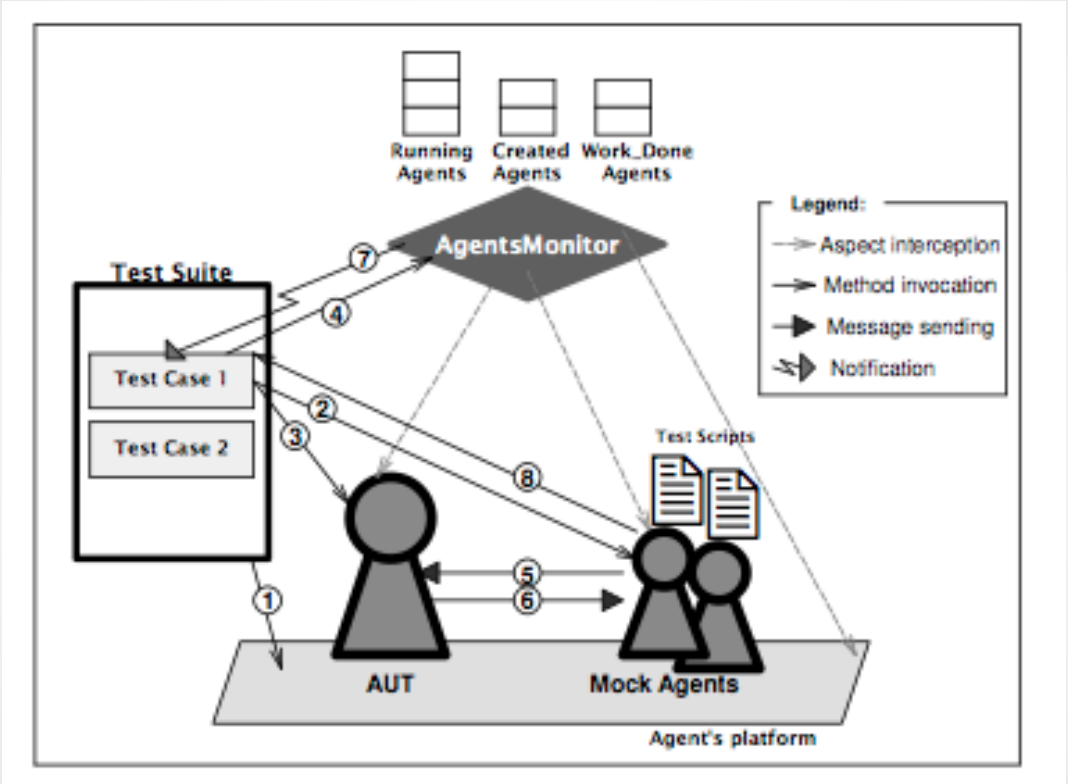
```java
@Test
public void generateShouldCheckTheCustomerDoesNotAlreadyExist() throws EntityNotFoundException {

    Customer customer = new Customer("c@gmail.com");
    EntityNotFoundException noSuchCustomer = new EntityNotFoundException();

    CustomerRepository rep = mock(CustomerRepository.class);
    when(rep.getByEmail(anyString())).thenReturn(customer).thenReturn(customer).thenThrow(noSuchCustomer);

    Customer guest = new GuestCustomerGenerator(rep).generate();

    assertThat(guest).isNotNull();
    verify(rep, times(3)).getByEmail(anyString());
}
```

Testes automatizados em SMA

- Unit Testing in Multi-agent Systems using Mock Agents and Aspects, Roberta de Souza Coelho
- Testing Techniques for Software Agents, Duy Nguyen

Unit testing with Mock




Running Agents
Created Agents
Work_Done Agents
AgentsMonitor
Legend:
Aspect interception
Method invocation
Message sending
Notification
Test Suite
Test Case 1
Test Case 2
Test Scripts
AUT
Mock Agents
Agent's platform
1
2
3
4
5
6
7
8

Workflow

- Para cada papel
  - Definir cada papel que ele interage
- Para cada um dos papéis que ele interage
  - Criar um agente Mock que responde as comunicações

Realização técnica

- Criação de um novo tipo de Behaviour e um novo tipo de agente
- Integração com o JUnit
- Criação do AgentMonitor que avalia os testes

```
1. public class MockBookBuyerAgent extends JADEMockAgent {
2.       ...
3.       protected void setup() {
4.     ...
5.     addBehaviour(new TestScenario());
6.}
7.}
8. private class TestScenario extends OneShotBehaviour {
9.    public void action(){
10.      try {
           ...
11.        sendMessage(msgType.CFP,sellerID, bookTitle);
12.        reply = receiveReply(6000, msgType.PROPOSE);
13.        sendMessage(msgType.Accept,sellerID,otherTitle);
14.        reply2 = receiveReply(6000, msgType.FAIL);
15.     } catch (ReplyReceptionFailed e) {
16.        setTestResult( prepareMessageResult(e));
17.     }
18.     setTestResult("OK");
19. }
20.}
```

Testing Techniques for Software Agents

Goal-oriented testing methodology (tropos)

- Levantar testes na fase de requerimentos
- Monitoring Agent
- Testes baseados em:
  - Objetivos
  - Criação "randomica" de mensagens

Poréms

- Todas as soluções foram feitos estendendo o JUnit para Jade.
- Testar é trabalhoso e necessita de tempo e preparo do programador

Referências

- http://zeroturnaround.com/rebellabs/how-to-mock-up-your-unit-test-environment-to-create-alternate-realities/
- Technologies, C. (2008). DIT - University of Trento Testing Techniques for Software Agents Cu Duy Nguyen, (December).
- H, J. F., & Sim, J. (n.d.). S - M oise + : A Middleware for developing Organised Multi-Agent Systems.
- Unit Testing in Multi-agent Systems using Mock Agents and Aspects. Coelho, Lucena. (2006)
- Bordini, R. H., Hübner, J. F., & Vieira, R. (n.d.). Chapter 1 JASON AND THE GOLDEN FLEECE OF AGENT-ORIENTED PROGRAMMING.
- Boissier, O., Bordini, R. H., Hübner, J. F., Ricci, A., & Santi, A. (2013). Multi-agent oriented programming with JaCaMo. *Science of Computer Programming*, *78*(6), 747–761.
- Wooldridge, M. (1999). *Multiagent Systems Intelligent Agents*.
- Ferber, J., Gutknecht, O., & Michel, F. (2003). From Agents to Organizations: an Organizational View of Multi-Agent Systems. *Agent-Oriented Software Engineering IV*, (July 2003), 443–459.
- Bordini, R. H., & Hübner, J. F. (2006). BDI agent programming in AgentSpeak using Jason. *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *3900 LNAI*, 143–164.