Программная инженерия. Лекция 5
Реализация + тестирование

Михаил Пожидаев

13 марта 2024 г.

Модель реализации

Всё в виде файлов

Модель реализации должна включать:

- исходный код;
- вспомогательные файлы;
- базы данных;
- скомпилированные файлы.

Цель и задачи

Движемся к бета-версии

Главная цель — начальная работоспособность, что подразумевает создание продукта, готового к бета-тестированию. Для этого нужно:

- расширить определение и описание вариантов использования, полностью их реализовать;
- завершить анализ, проектирование, реализацию и тестирование;
- Поддерживать целостность архитектуры, производить её модификацию только в случае крайней необходимости.
- отслеживать риски, в случае их реализации принимать меры по устранению их последствий.

Порядок построения

Дополнительные соглашения для фазы реализации

1. Необходимо поддерживать актуальный вариант модели развёртывания, чтобы иметь возможность контролировать её потенциальную выполнимость.
2. Между файлами в реализации и вариантами использования должна поддерживаться согласованная ассоциация.
3. Процесс сборки проекта должен быть формализован и поддерживаться в актуальном состоянии.

План реализации

Построение системы малыми приращениями

Основной подход

Один билд соответствует реализации одного варианта использования каждым из исполнителей.

Что для этого делаем?

Для этого составляем план итераций в соответствии с рейтингом вариантов использования. Необходимо учитывать приоритеты и риски. По возможности организуем параллельную работу тестеров.

Модель тестирования

Составные части каждого компонента тестирования

1. *Тестовый пример:* исходные данные и ожидаемые результаты.
2. *Процедура тестирования:* описание, как запускать тестовые примеры.
3. *Тестовый компонент:* артефакт для реализации процедуры тестирования.

Классическая основа

Как работает тестер?

От вариантов использования

Тестер в классическом виде ориентируется на модель вариантов использования и обязан убедиться, что продукт в точности соблюдает их.

Типы тестирования

1. Функциональное тестирование.
2. Регрессионное тестирование.
3. Стресс-тестирование.
4. Тестирование безопасности.

Повсеместность

Как сделать идеальный тест?

Критерии всеобщности тестов:

- охват вариантов пользовательской активности;
- доля покрытия тестами исходного кода;
- охват вариантов условий эксплуатации ПО;
- степень погружения тестеров в компоненты проекта;
- частотность проведения тестирования.

Машинное обучение

Проанализируем действия пользователя

Идея подвергнуть действия фактических пользователей анализу при помощи интеллектуальных алгоритмов напрашивается сама собой, но:

1. Требуется построение модели взаимодействия человека с продуктом .
2. Требуется построение модели обучающих данных на основе модели взаимодействия.
3. Разработанная модель обучения оказывается глубоко привязанной к конкретному типу приложения.

QAOps

Поменяем организационную структуру

Почему разные команды?

Исторически сложилось так, что функция тестеров доверяются отдельным членам команды, которые могут плохо ориентироваться

А что будет, если QA скрестить с DevOps?

1. Процесс тестирования не разделяется с процессом развёртывания.
2. Тестеры в точности знают, в каких условиях эксплуатируется ПО.

Test as text

Описание тестов на естественном языке

Ассистивный подход

IntelliJ Idea предлагает инструменты быстрого создания шаблонов unit-тестов на основе описания, но реализацию тестов разработчик всё равно готовит вручную.

Автоматический подход

Идеально было бы получать готовые тесты, просто словами описывая, требуемое поведение системы. На эту задачу можно смотреть в свете генерации тестов по функциональному определению, создаваемому с использованием, например, PlantUML.

BDD

Behaviour-Driven Development

Agile для междисциплинарных продуктов

Накладные расходы

Во всех методологиях заказчик и исполнитель говорят на разных языках, и требуется предусматривать «переводчика» в лице системного аналитика.

Унифицируем язык

Предлагается научиться общаться на языках, как можно ближе друг к другу, параллельно развивая инструментарий, подходящий для обработки материалов на таком языке.

Технологии

Опишем задачу и тесты на естественном языке

Постановка задачи

Фиксацию моделей анализа и проектирования можно выполнять на обычном естественном языке, но можно и в специальной нотации, которая позволяет автоматически получить диаграммы UML.

Тестирование

Фиксацию тестов также можно проводить с использованием описаний на естественном языке.

Behave

BDD для Python

Каждая функциональная часть тестируется путём формулировки трёх утверждений на естественном языке:

- *Given:* условие для выполнения теста;
- *When:* момент выполнения теста;
- *Then:* результат выполнения теста.

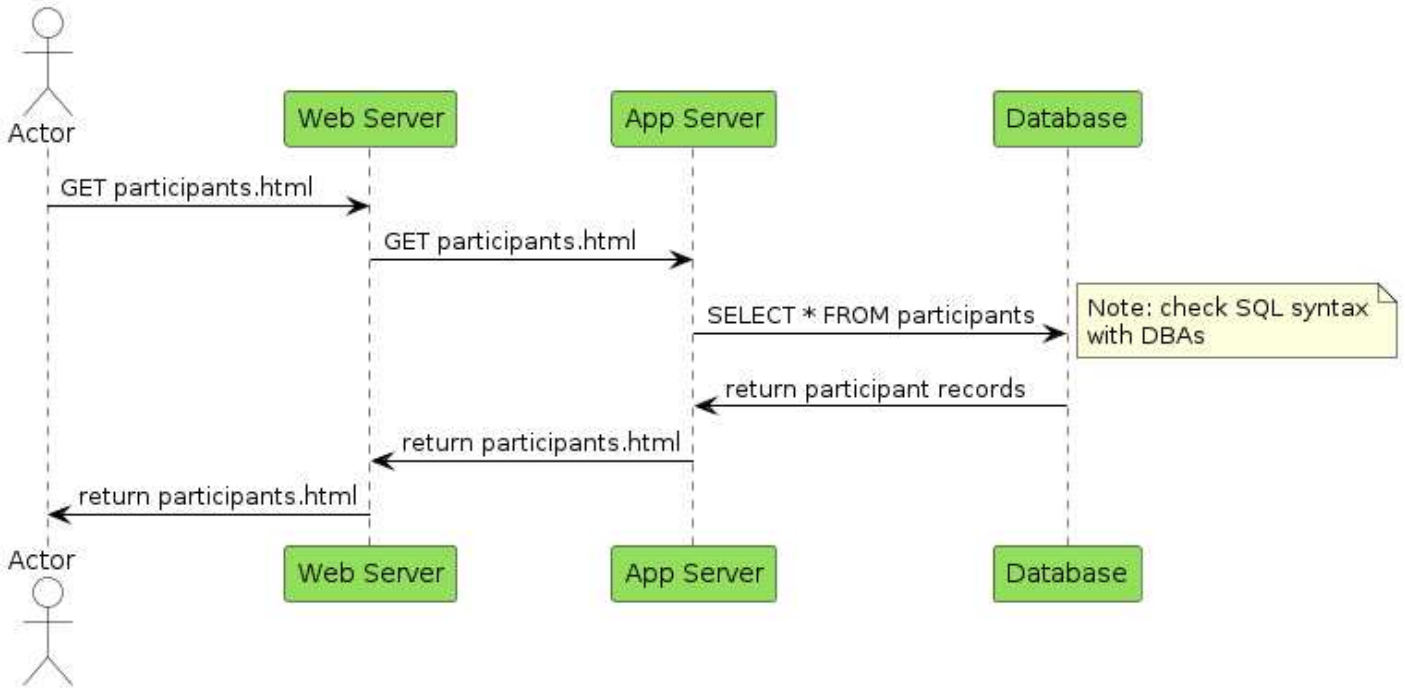
Фиксация теста

```
Feature: Aircraft operations
Scenario: Takeoff
  Given a airport
  When obtained a command to takeoff
  Then the aircraft flies to its destination
```

Реализация теста

```
@given(u'a airport')
def step_impl(context):
    return
@when(u'obtained a command to takeoff')
def step_impl(context):
    return
@then(u'the aircraft flies to its destination')
def step_impl(context):
    return
```

Диаграмма PlantUML



Текст PlantUML

```
...
actor Actor
participant "Web Server" as web
participant "App Server" as app
participant "Database" as db
Actor -> web: GET participants.html
web -> app: GET participants.html
...
```

Спасибо за внимание!

Всё о курсе: https://marigostra.ru/materials/engineering.html
E-mail: msp@luwrain.org
Канал в Телеграм: https://t.me/MarigostraRu