Decoupling Scenarios from Behavior-Driven Tests

Seyed Mehran Kholdi
Mohammad Hossein Sekhavat

Supervisor: Seyed Hassan Mirian Hosseinabadi

Sharif University Of Technology
Computer Engineering Department

Summer 1395

Outline

- Background
- Proposed Framework
- Implementation
- Case study
- Future Works

Background

- **Testing is hard!**
- **TDD: Test Driven Development**
- **But...**

Background (cntd.)

- **BDD**

Scenario 1: Enrollments in an offering with no capacity must fail

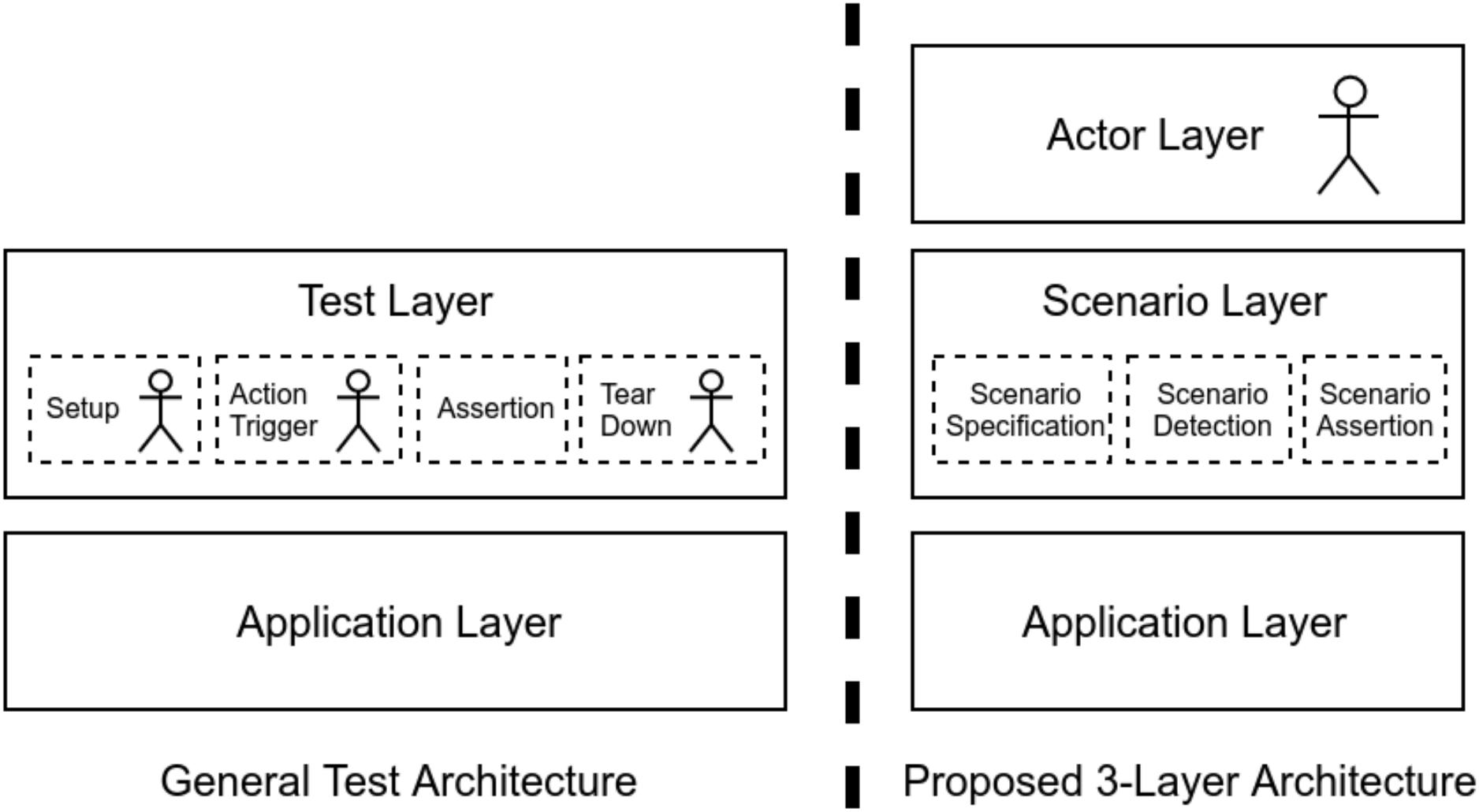
Given an offering `o1` with no capacity, and a student `s1`

When `s1` tries to enroll in `o1`

Then enrollment must fail

Proposed Framework

- **The problem with current BDD impl.**
- **Our proposed solution**



Proposed Framework (cntd.)

- **Pros:**
  - Reduced LOC
  - Reduced maintenance cost
  - Increased effectiveness
- **Actor**
  - Model-based user behavior simulation
  - An operational system

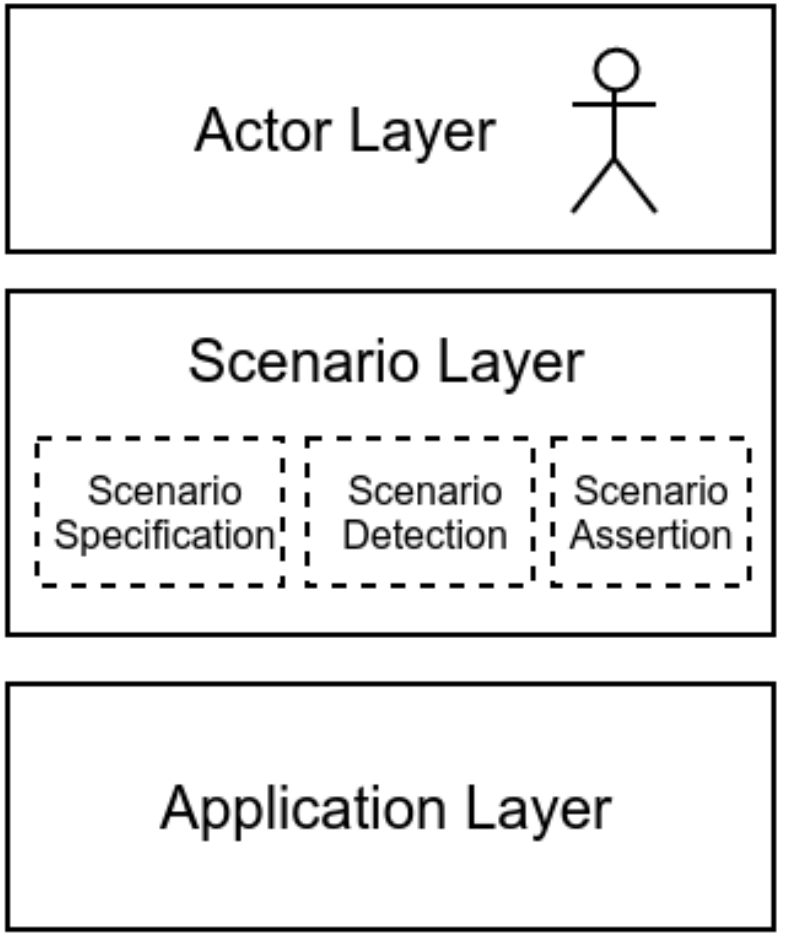
From Criteria to Test

Scenario 1: Enrollments in an offering with no capacity must fail

Given an offering `o1` with no capacity, and a student `s1`

When `s1` tries to enroll in `o1`

Then enrollment must fail



Proposed 3-Layer Architecture

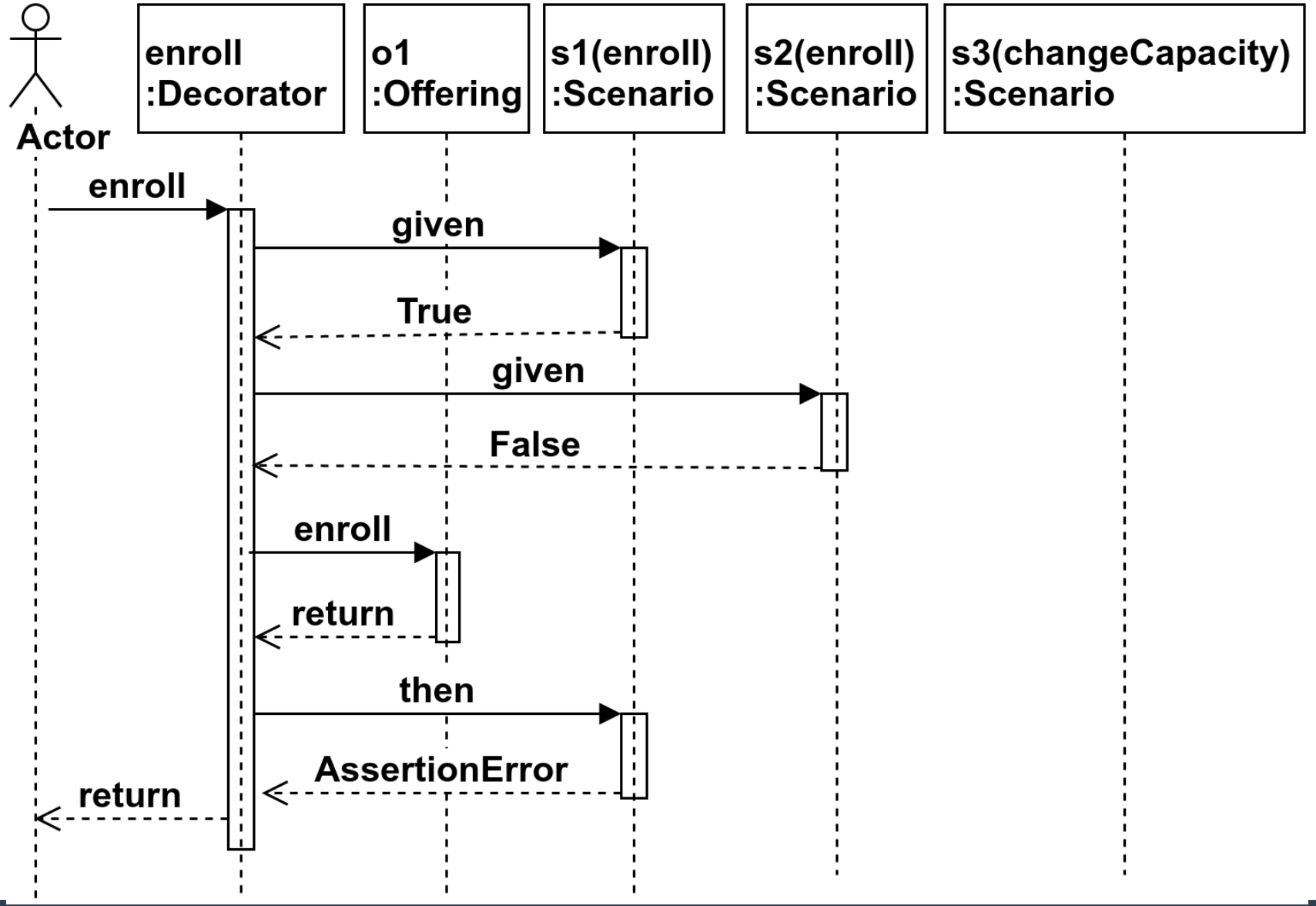
Final Solution: Scenario Specification

```
class EnrollmentShouldFailForOfferingWithZeroCapacity(Scenario):
    """
    Scenario: Enrollment should fail for offering with zero capacity
    Given offering o1 with zero capacity
    When someone enrolls in it
    Then it should fail with error
    """

    def given(scenario, self, **payload):
        return self.available_capacity == 0

    when = 'edu.models.Offering.enroll'

    def then(scenario, exc_type, **kwargs):
        assert issubclass(exc_type, EnrollmentError)
```

Final Solution: Scenario Detection

```
class Offering(models.Model):
    course = models.ForeignKey('edu.Course')
    semester = models.ForeignKey('edu.Semester')
    professor = models.ForeignKey('edu.Professor')
    available_capacity = models.IntegerField()


    @action()
    def enroll(self, student, commit=True):
        Enrollment = apps.get_model('edu.Enrollment')
        enrollment = Enrollment(offering=self, student=student)
        enrollment.save()
        self.available_capacity = self.available_capacity - 1
        self.save()
```

Final Solution: Scenario Detection (ctd.)
Given => When => Then



Future Works

- **Different actor implementations**
  - Integration with existing tools
- **Scenario pruning**
- **New metrics (e.g. coverage)**
- **Automatic action detection**
- **Real-world case study**

Q & A

Thanks