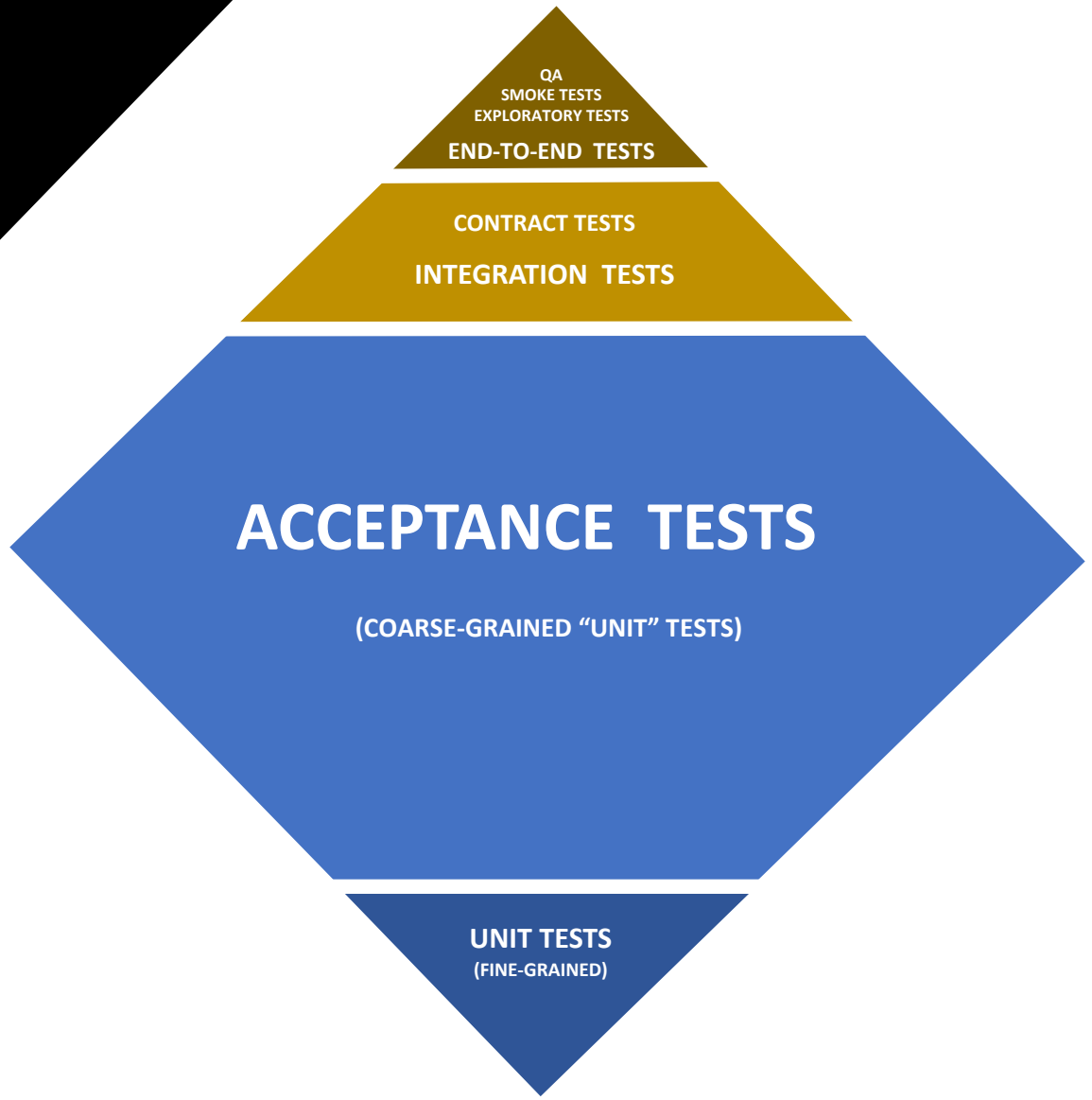
Write Antifragile & Domain-Driven Tests with

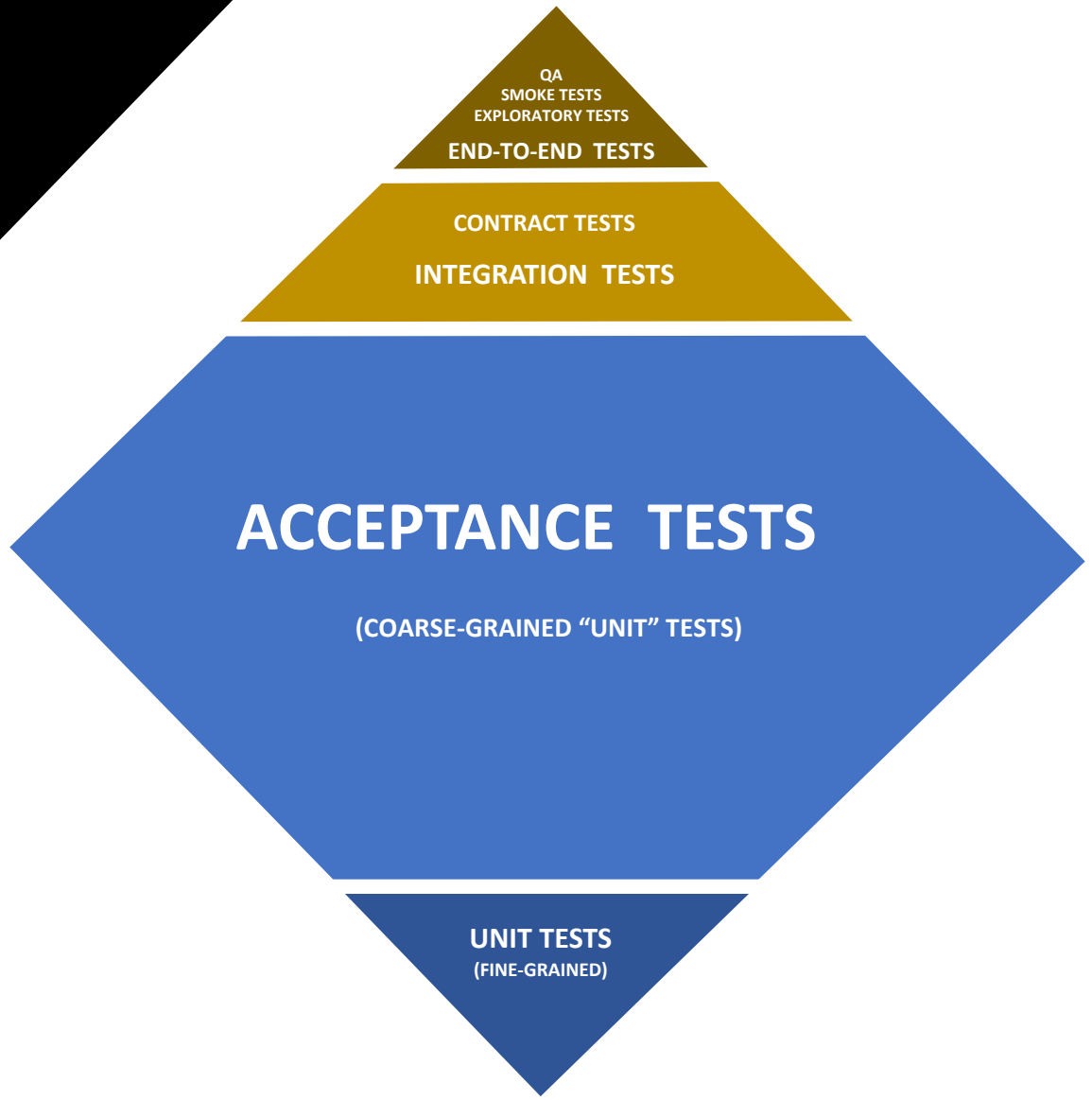
OUTSIDE-IN DIAMOND 🔷 TDD



AGICAP
Thomas PIERRAIN
pierrain (use case driven)

Write Antifragile & Domain-Driven Tests with

OUTSIDE-IN DIAMOND 🔷 TDD



AGICAP
Thomas PIERRAIN
pierrain (use case driven)

Disclaimers

There is no silver bullet Your testing strategy & techniques must always be chosen accordingly to your context (both human & technical)

As long as you **understand your trade-offs** there is no reason not to **explore new paths…**

Thanks to **Kent Beck**, **Martin Fowler**, **Michael Feathers**, **Nat Pryce** & **Steve Freeman** for their great source of inspiration over the years **#shouldersOfGiants**

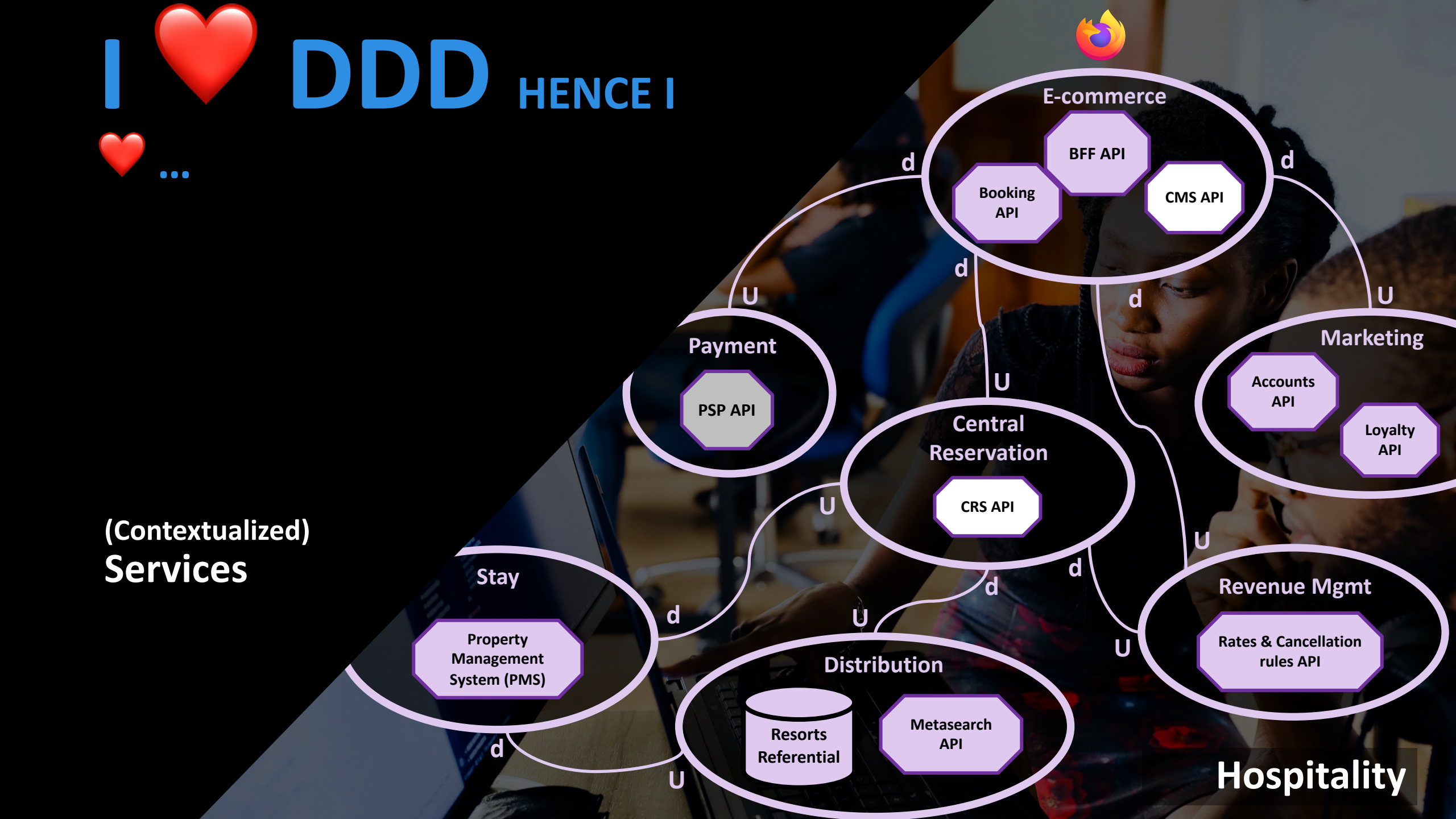
Preamble

I ❤ DDD HENCE I ❤ ...

(Contextualized)
Services

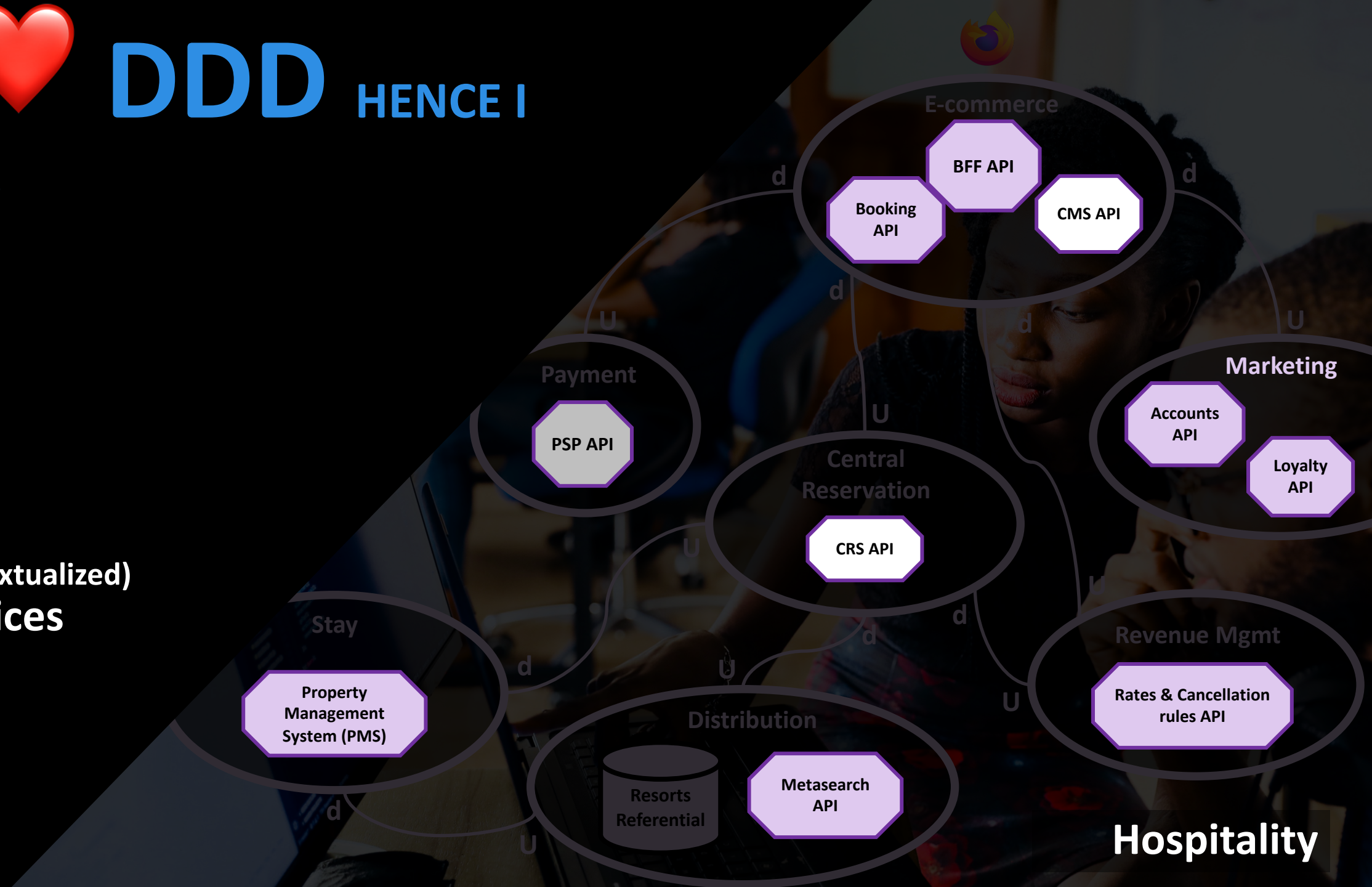
I ❤ DDD HENCE I ❤ ...

(Contextualized)
Services



I ❤ DDD HENCE I ❤ ...

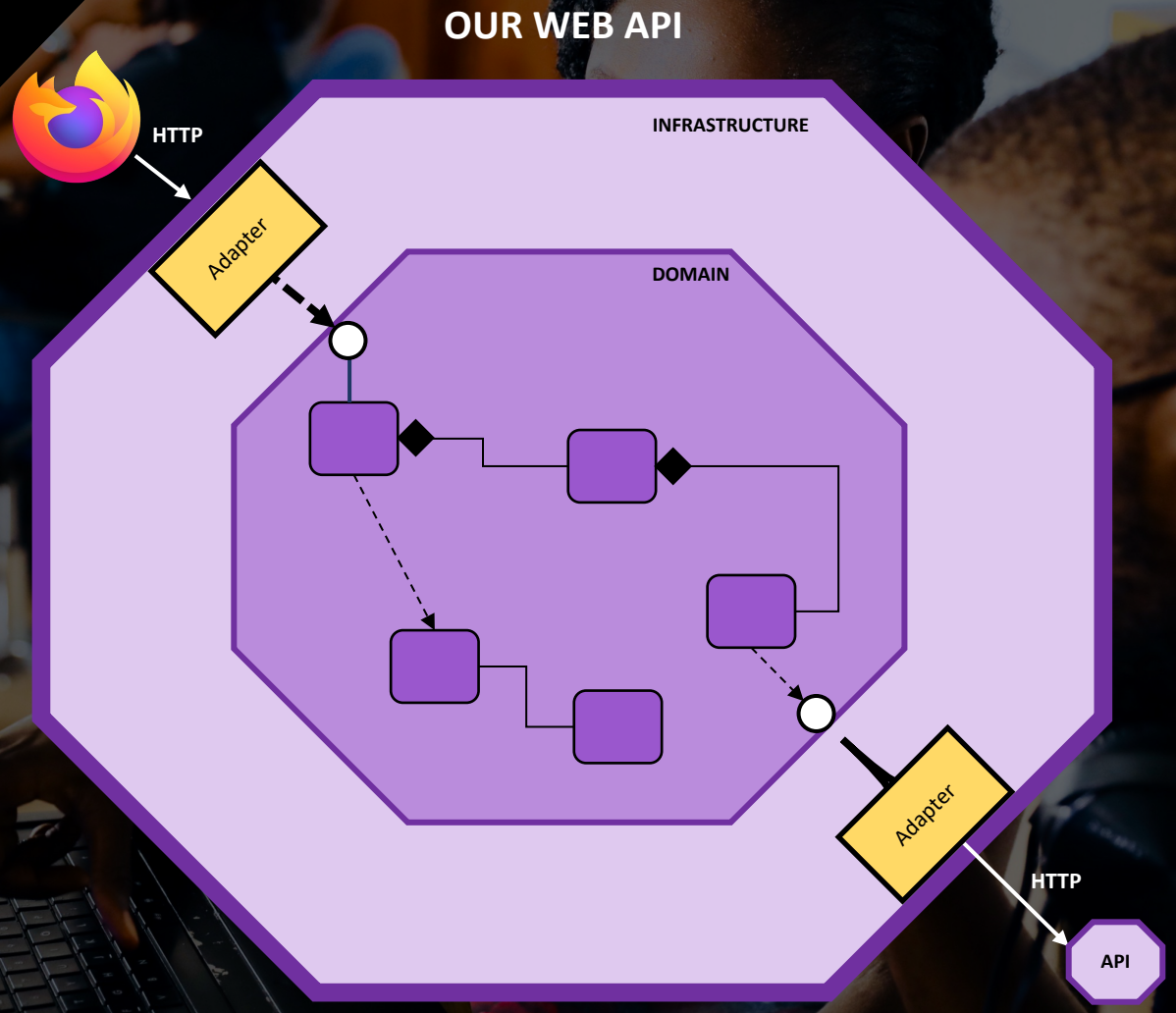
(Contextualized)
Services



Hospitality

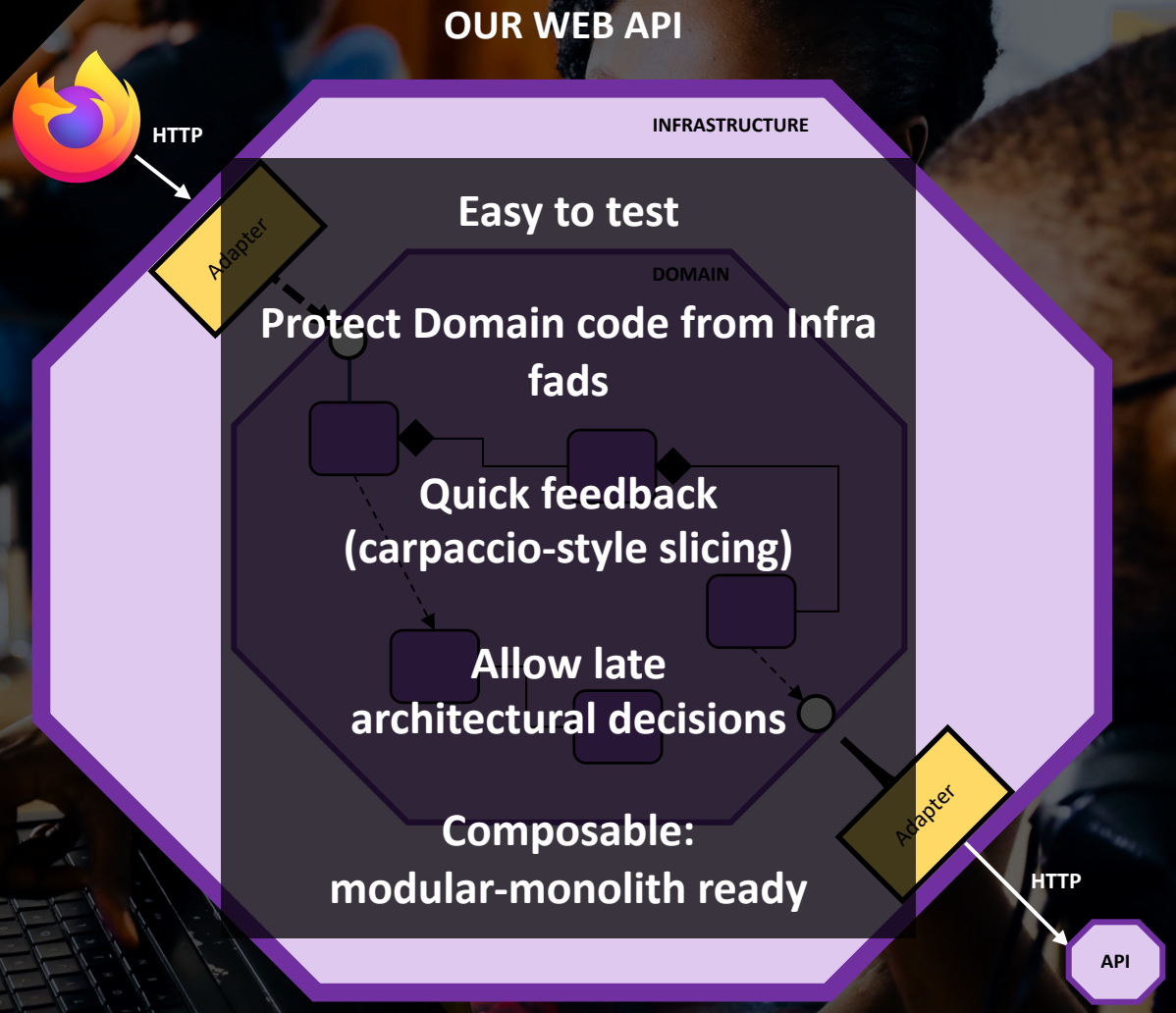
DDD & TDD ALSO LOVE...

Ports & Adapters

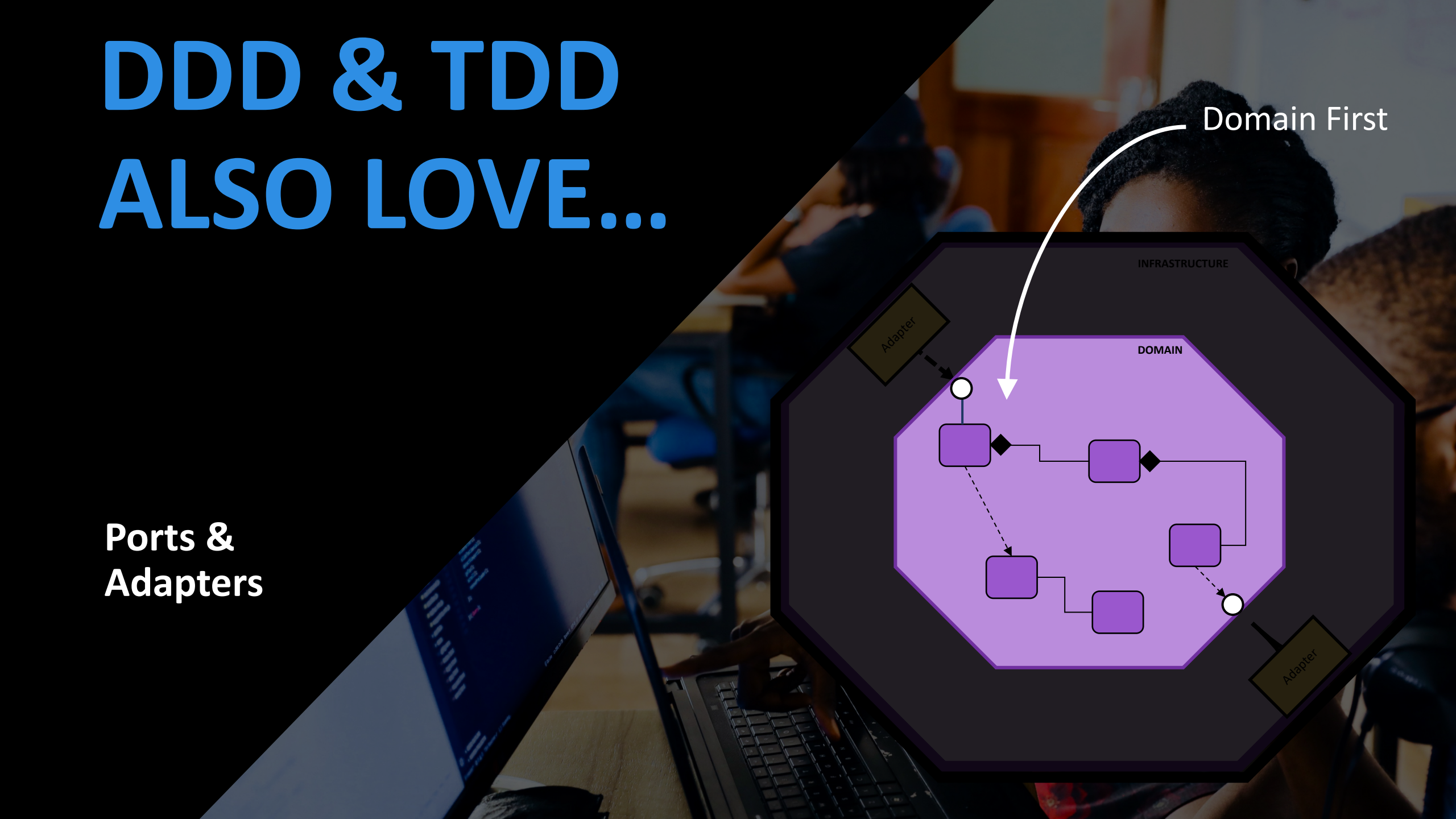


DDD & TDD ALSO LOVE...

Ports & Adapters



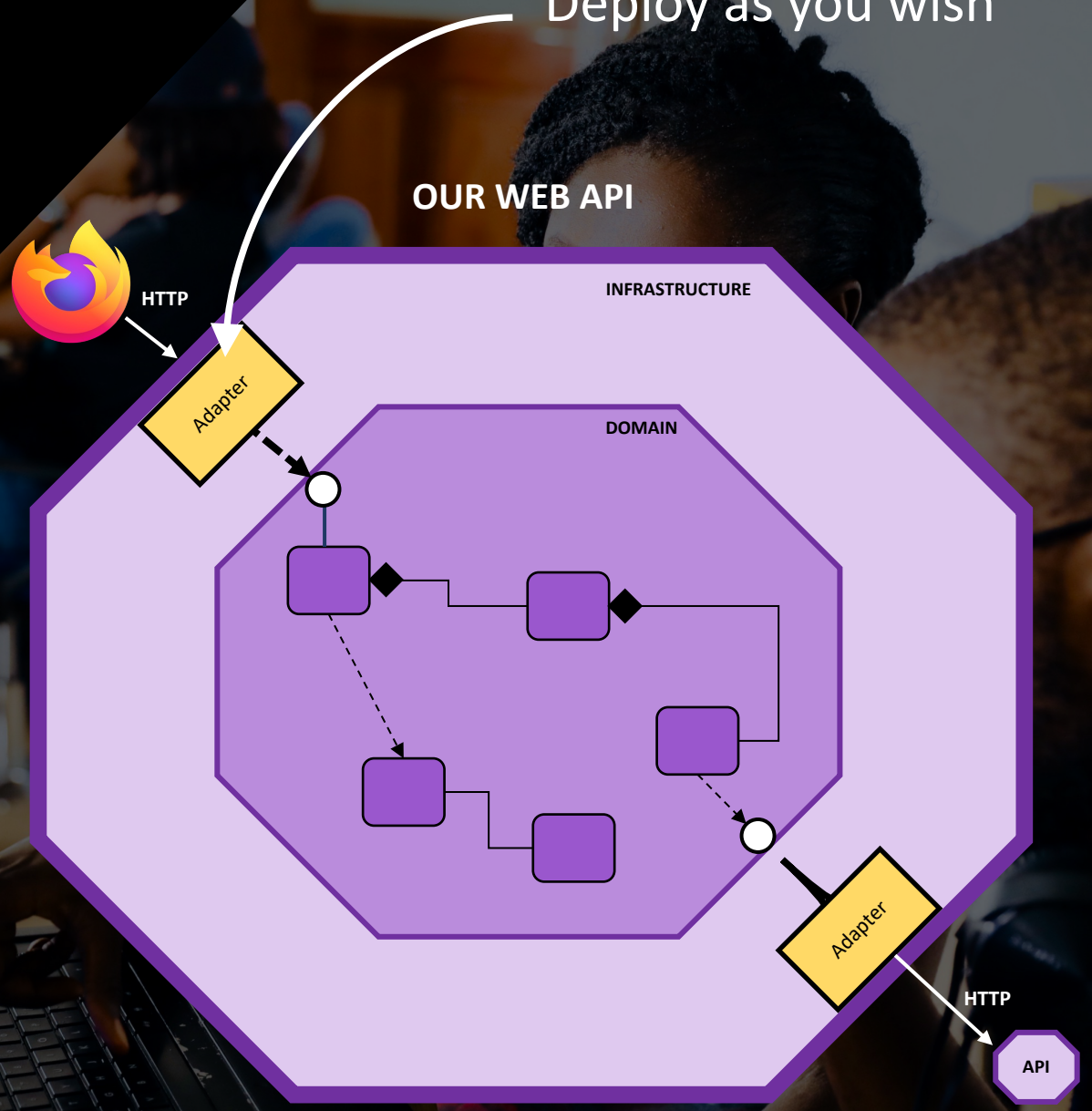
DDD & TDD ALSO LOVE...



Ports & Adapters

DDD & TDD ALSO LOVE...

Ports & Adapters



Let’s talk about tests!

A WORLD OF TESTS

Different Types

ET — Exploratory Tests (E2E)

IT — Integration Tests (Contract)

AT — (coarse-grained unit) Acceptance Tests

UT — (fine-grained) Unit Tests

Why do I do TDD?

TEST DRIVEN DEVELOPMENT

Made me…

TDD Workflows

TEST DRIVEN DEVELOPMENT

Outside-in

TEST DRIVEN DEVELOPMENT

AT

Outside-in

TEST DRIVEN DEVELOPMENT

Outside-in

TEST DRIVEN DEVELOPMENT

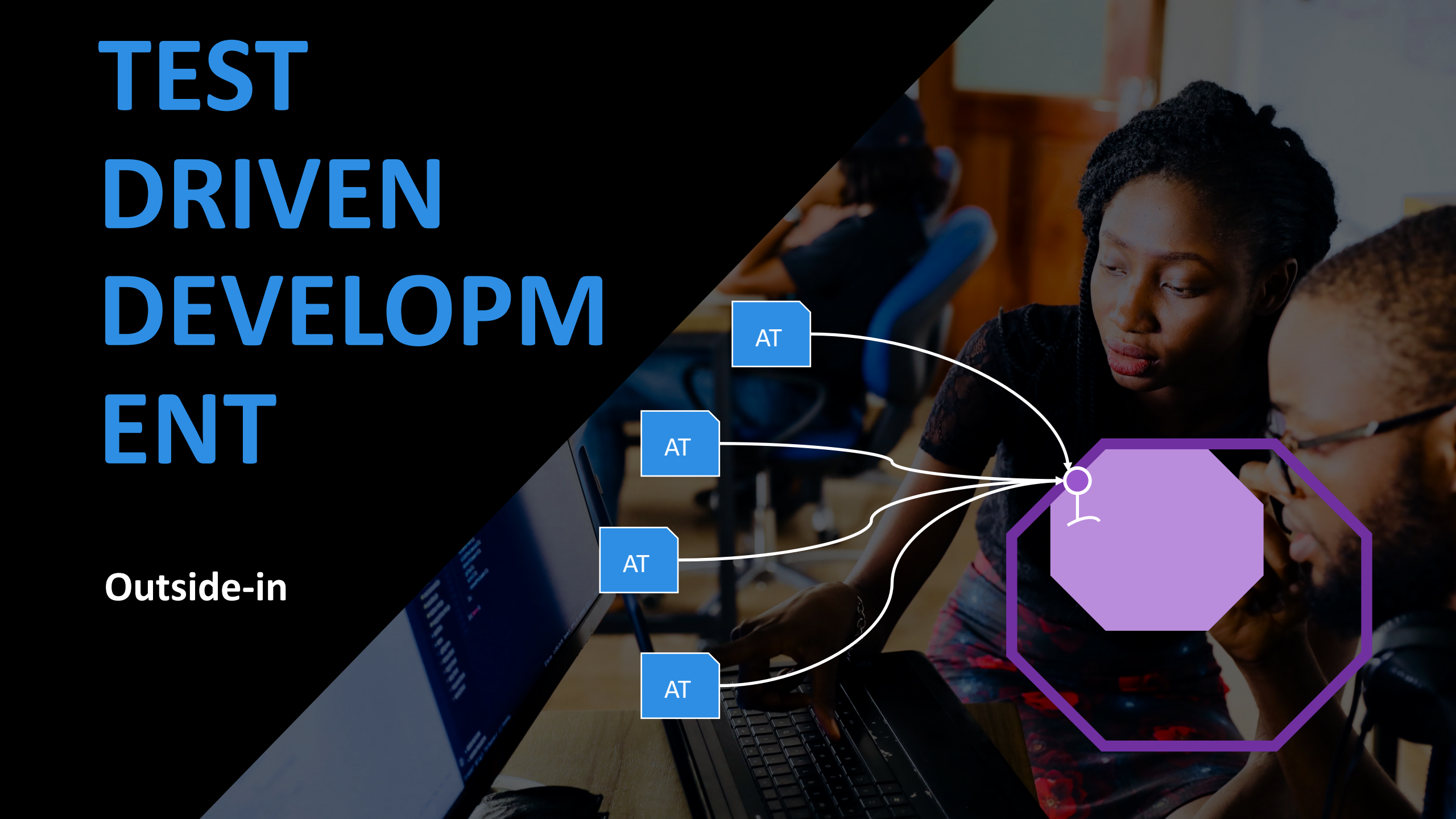
Outside-in

TEST DRIVEN DEVELOPMENT

Outside-in

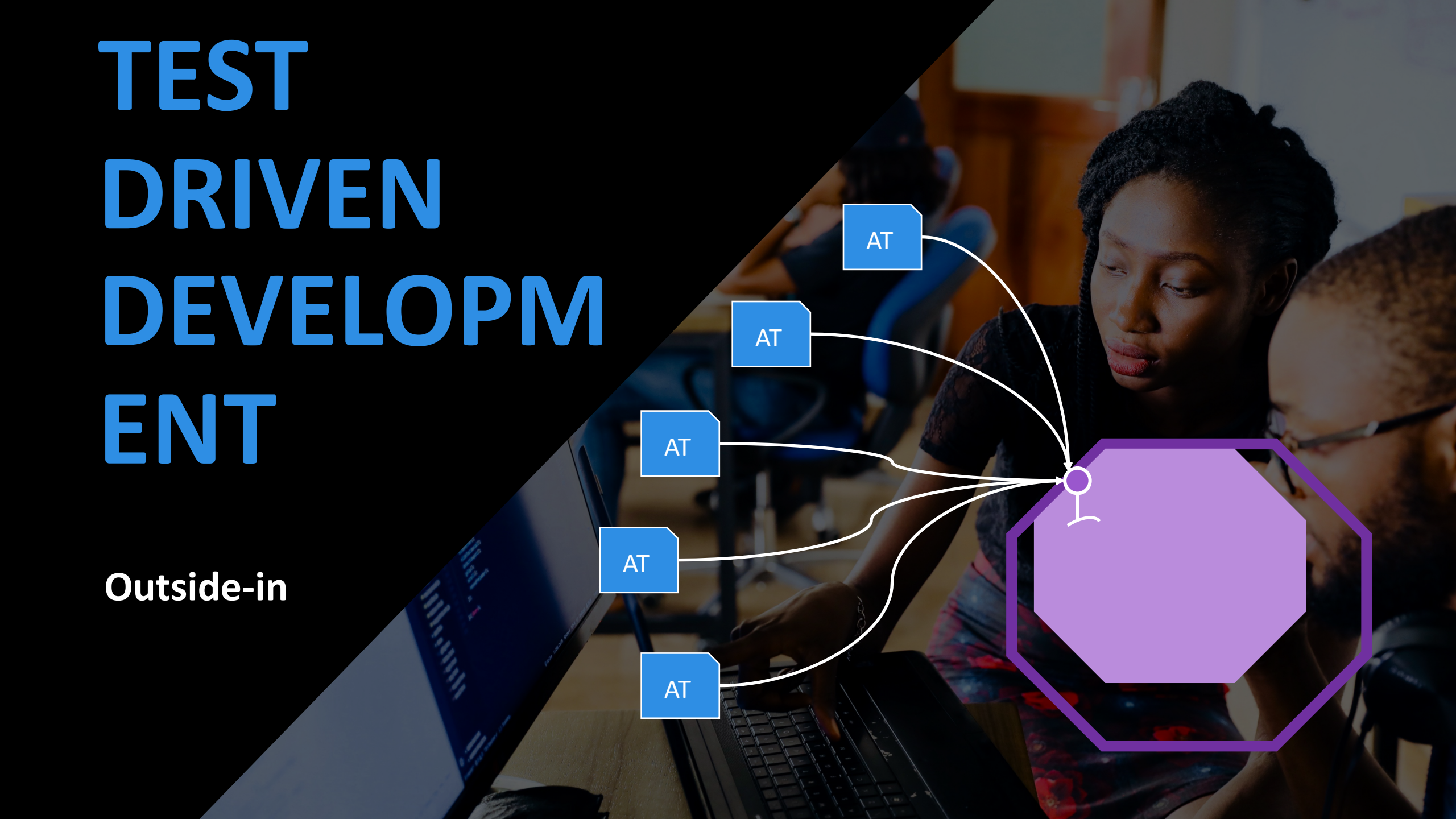
TEST DRIVEN DEVELOPMENT

Outside-in



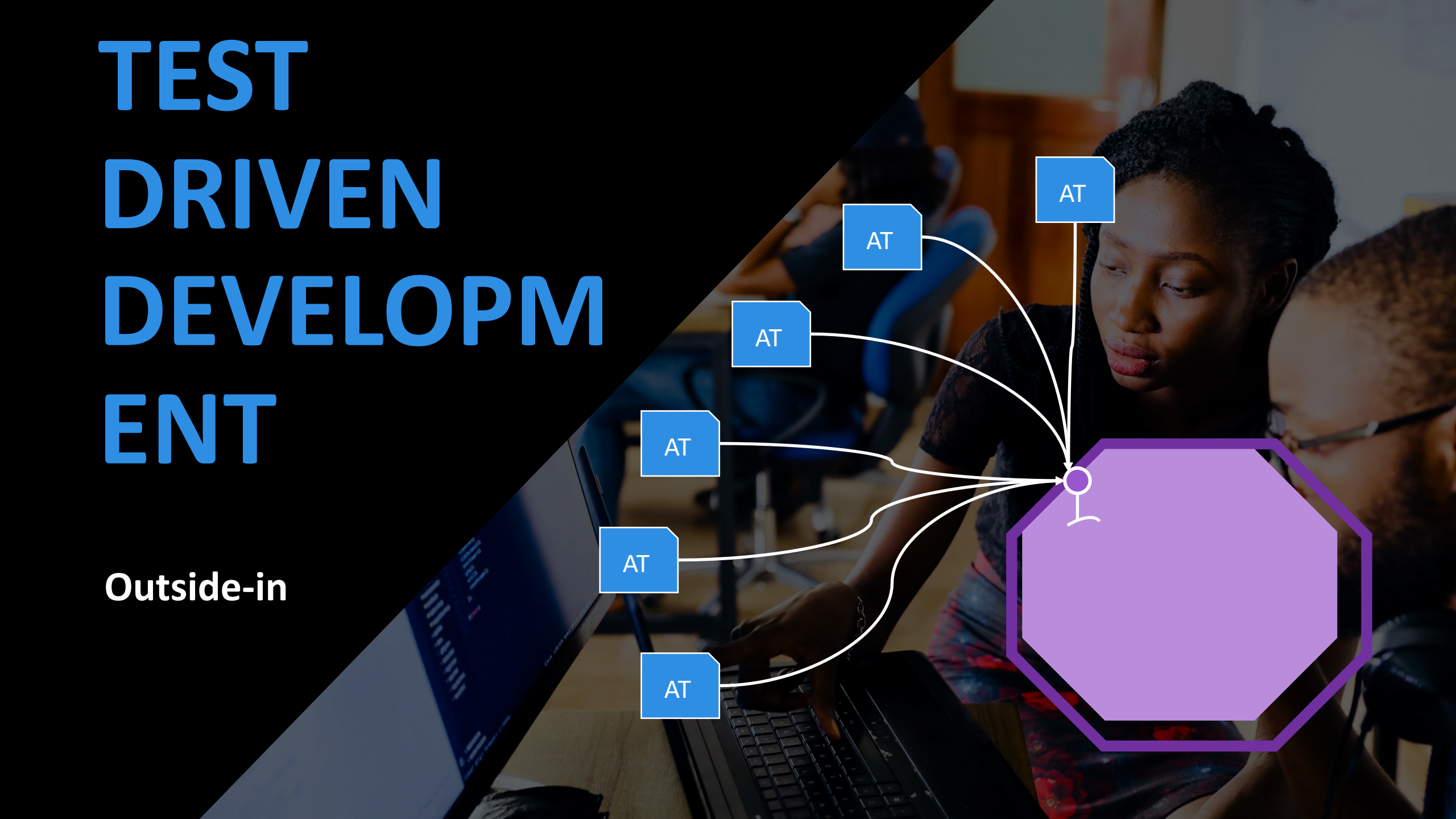
TEST DRIVEN DEVELOPMENT

Outside-in



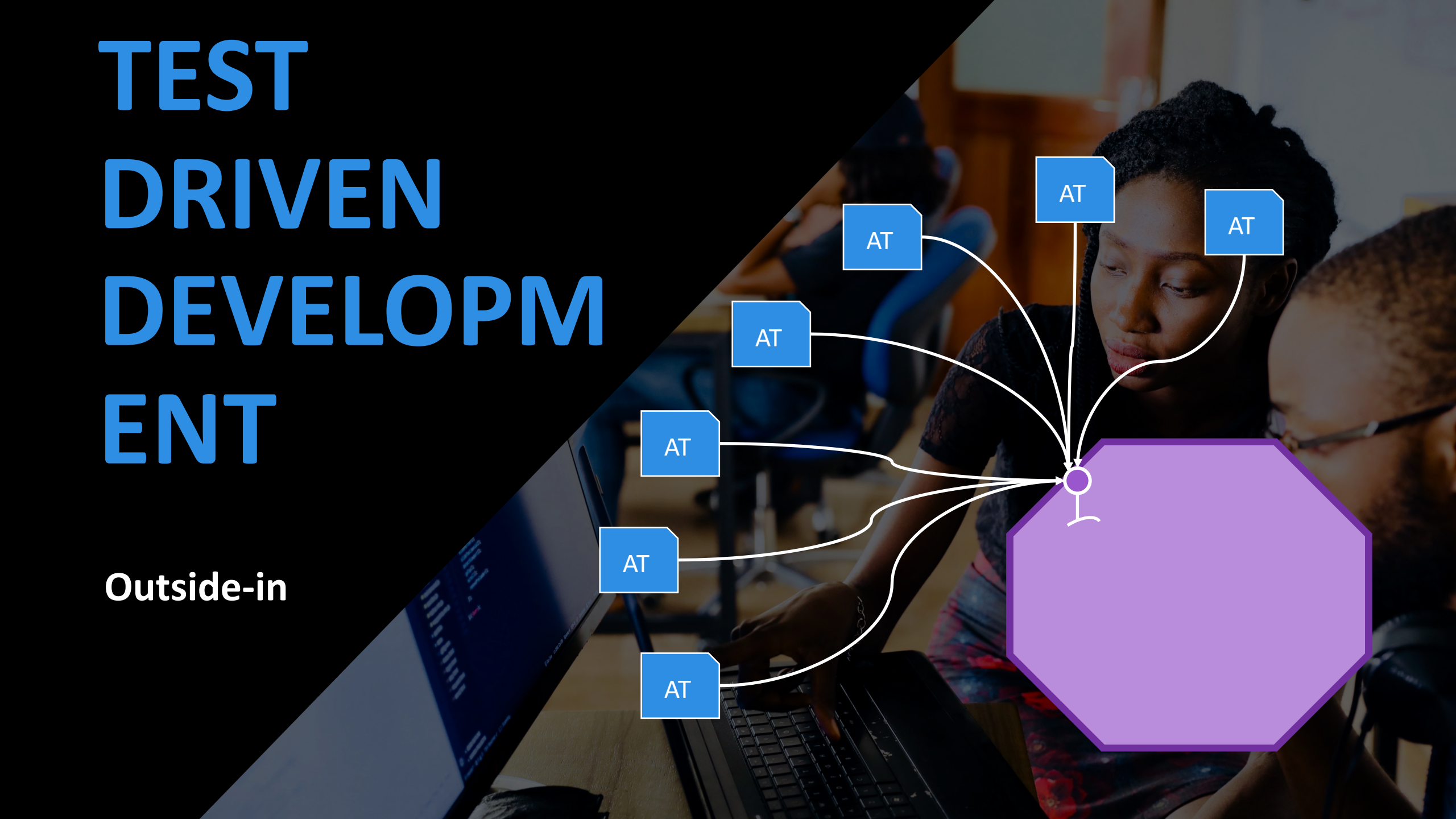
TEST DRIVEN DEVELOPMENT

Outside-in



TEST DRIVEN DEVELOPMENT

Outside-in



TEST DRIVEN DEVELOPMENT

Inside-out

TEST DRIVEN DEVELOPMENT

Inside-out

TEST DRIVEN DEVELOPMENT

Inside-out

TEST DRIVEN DEVELOPMENT

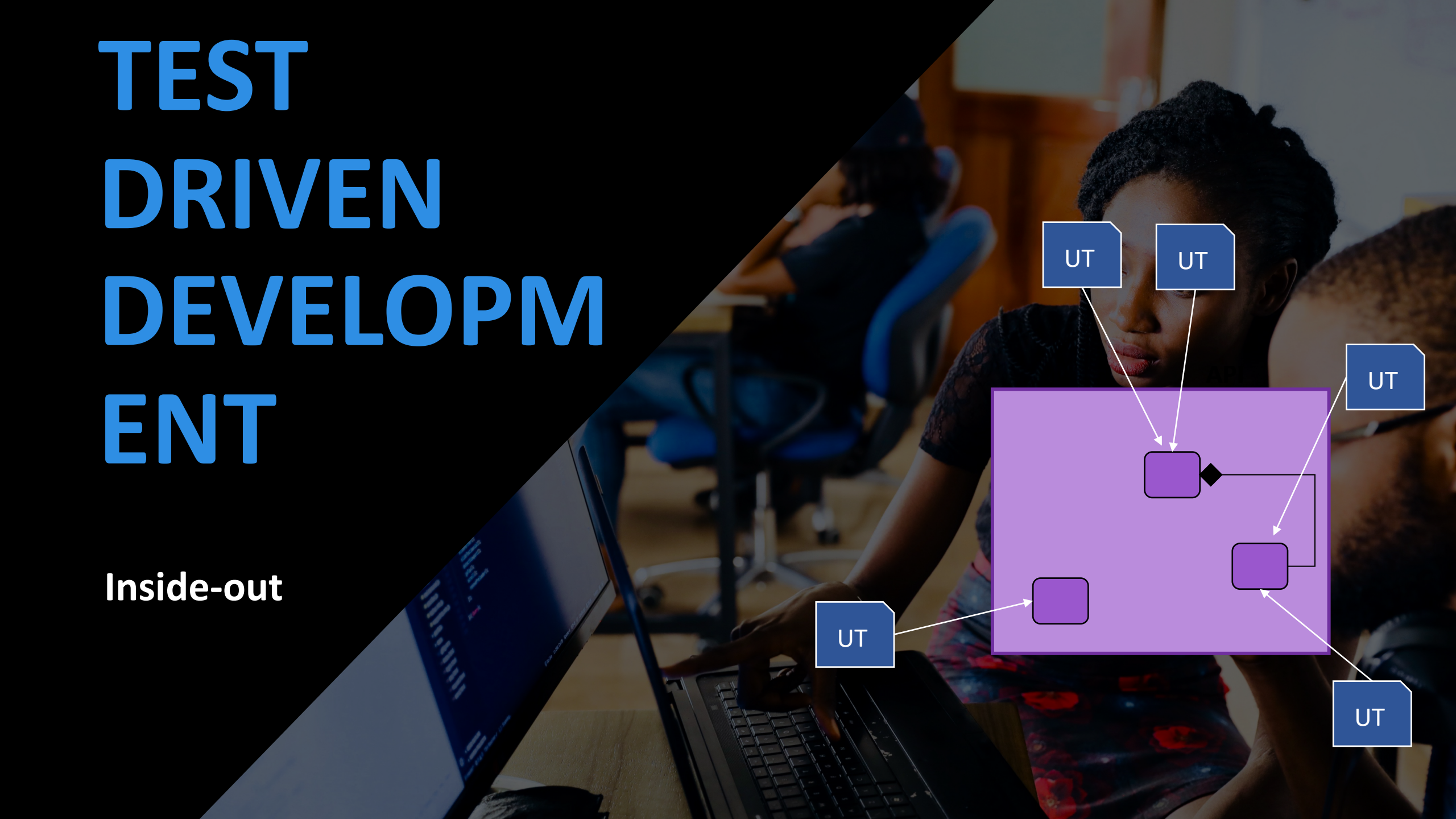
Inside-out

TEST DRIVEN DEVELOPMENT

Inside-out

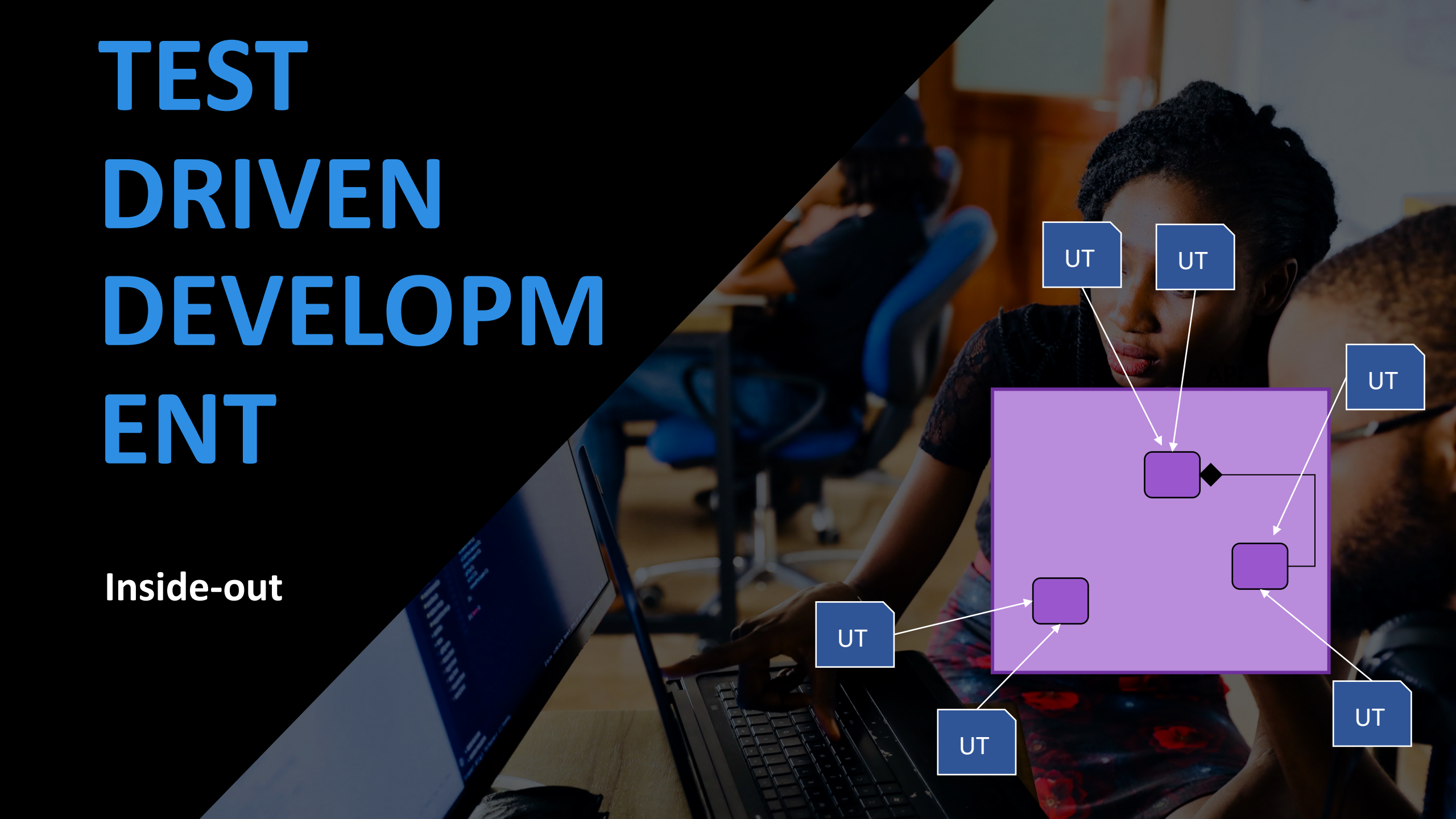
TEST DRIVEN DEVELOPMENT

Inside-out



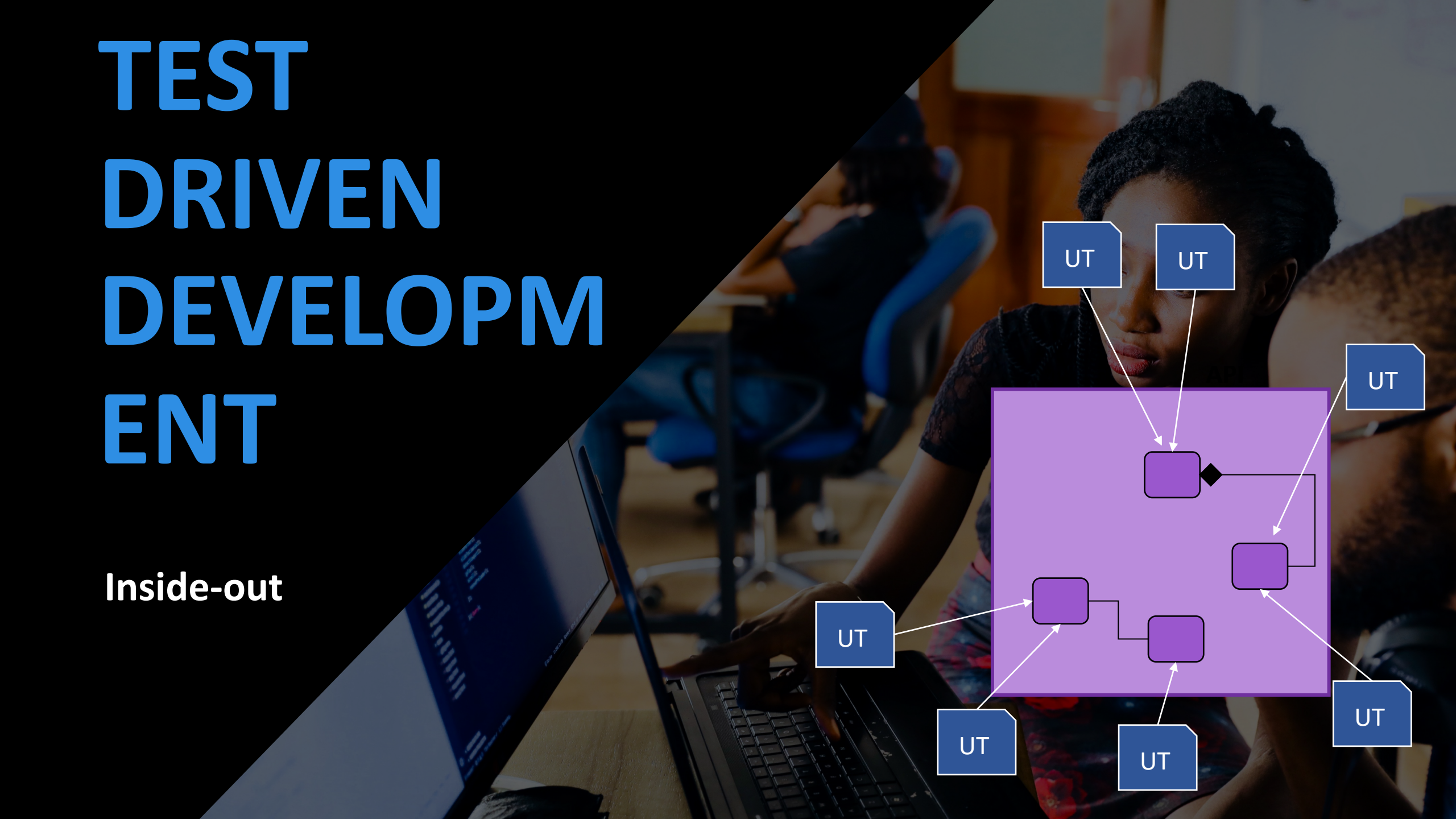
TEST DRIVEN DEVELOPMENT

Inside-out



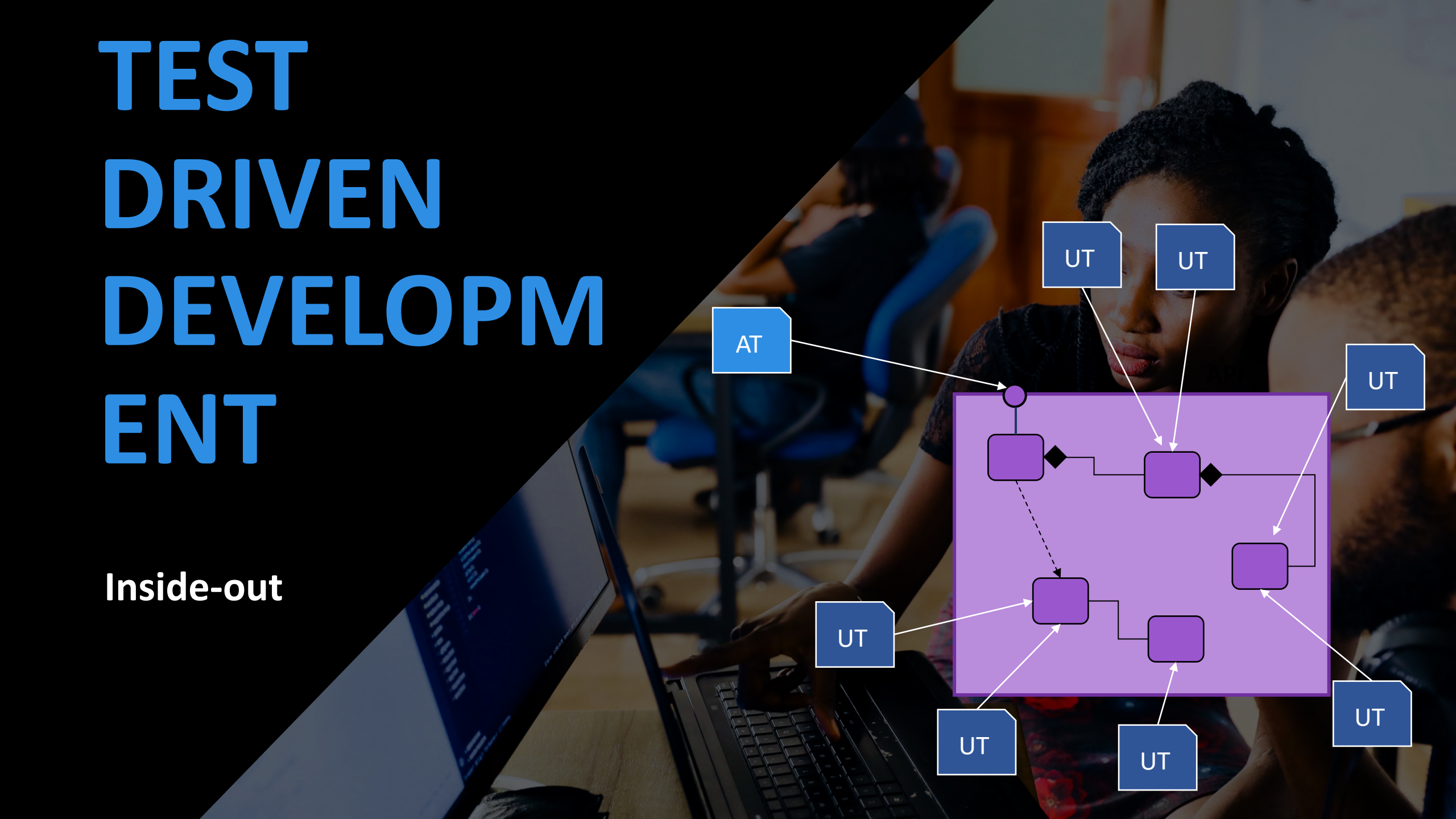
TEST DRIVEN DEVELOPMENT

Inside-out



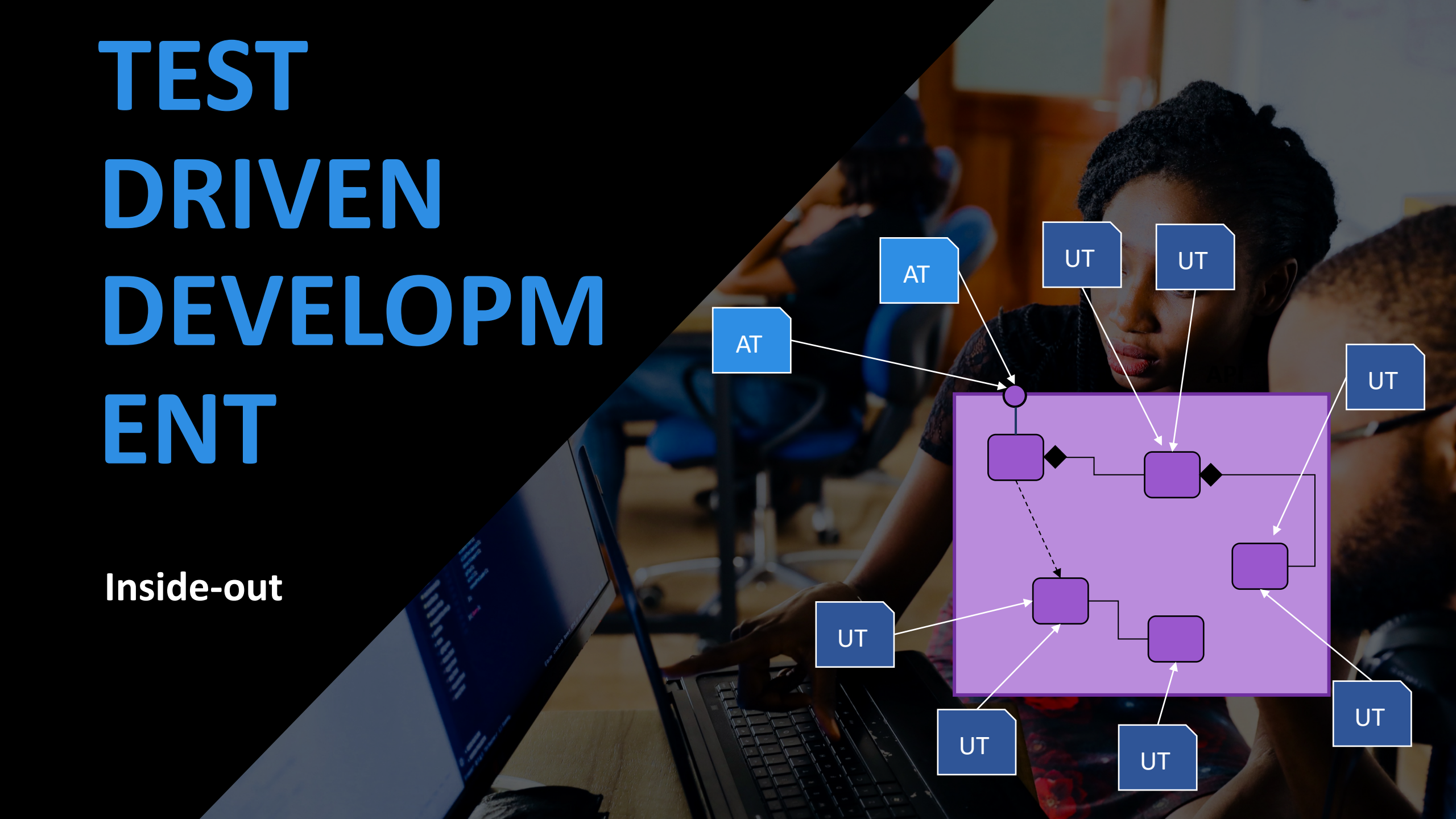
TEST DRIVEN DEVELOPMENT

Inside-out



TEST DRIVEN DEVELOPMENT

Inside-out



TEST DRIVEN DEVELOPMENT

Workflows

Common pitfalls & mitigations

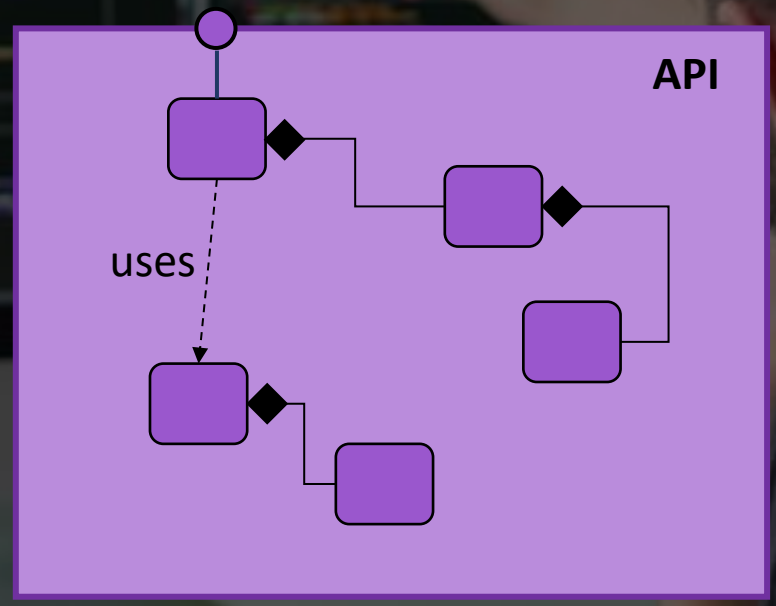
TEST DRIVEN DEVELOPMENT

Beware of…

1. **Fragile tests**
2. **Blind spots**
3. **Complex setups**

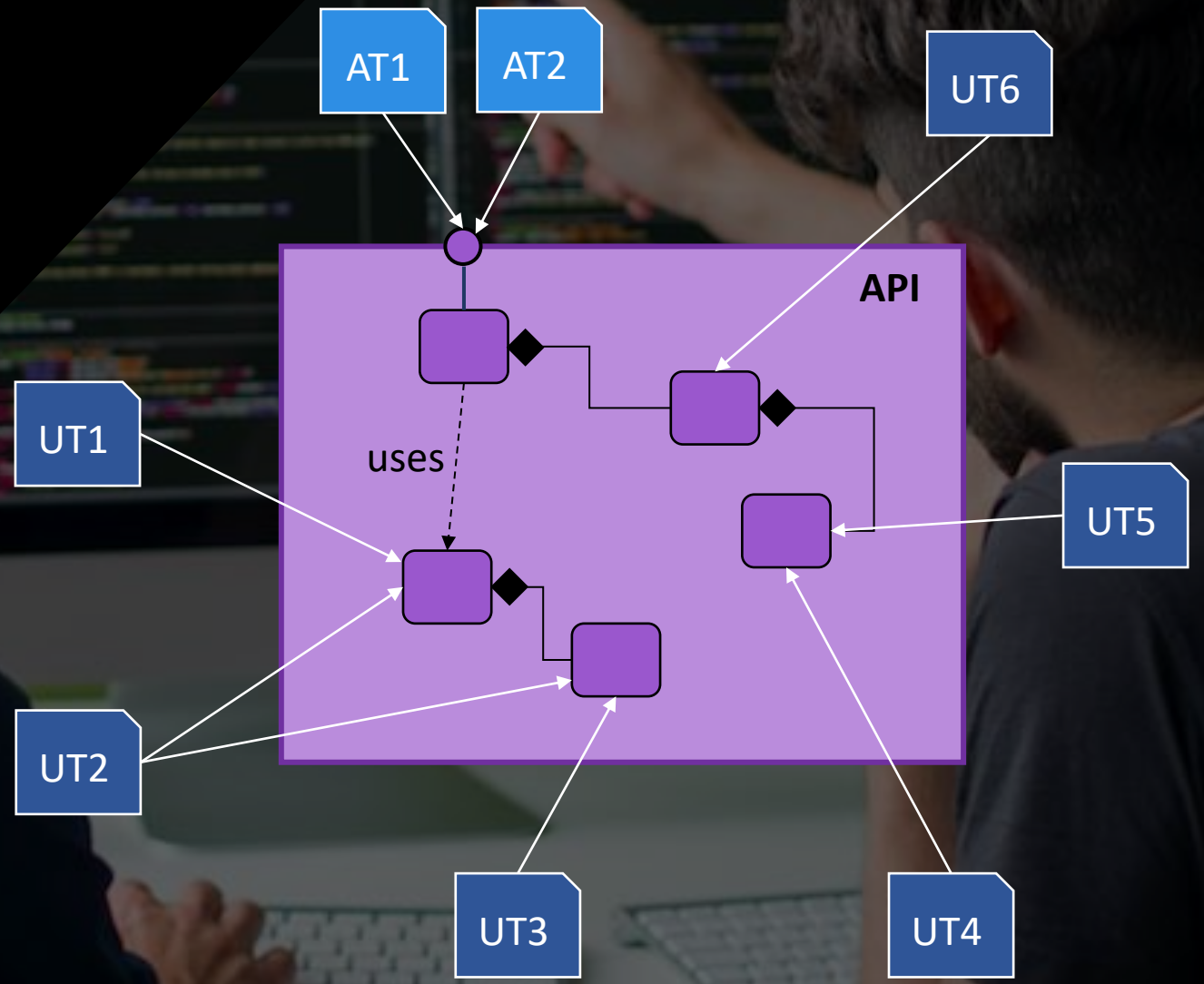
TEST DRIVEN DEVELOPMENT

1.
Beware of…
Fragile tests



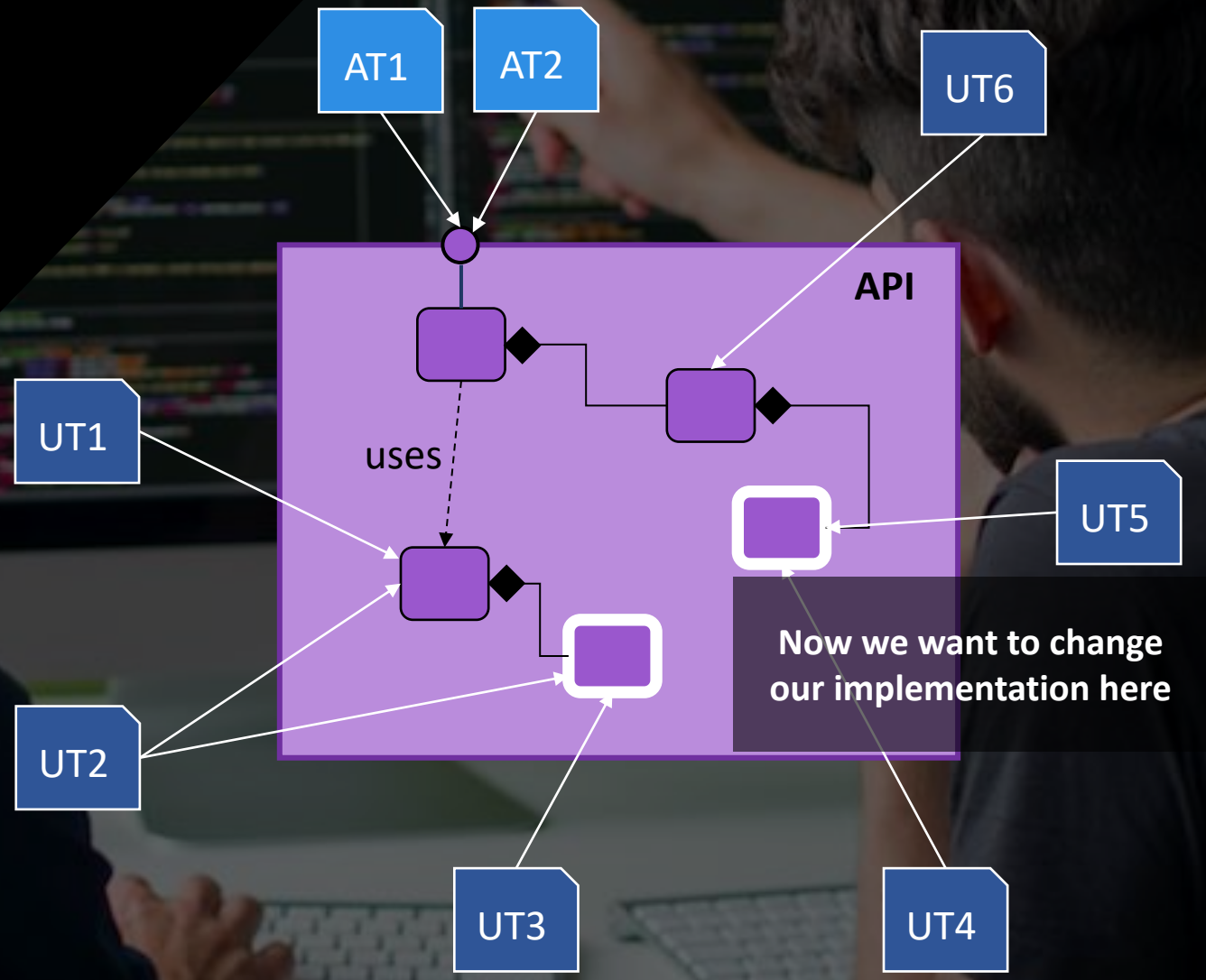
TEST DRIVEN DEVELOPMENT

1.
Beware of…
Fragile tests



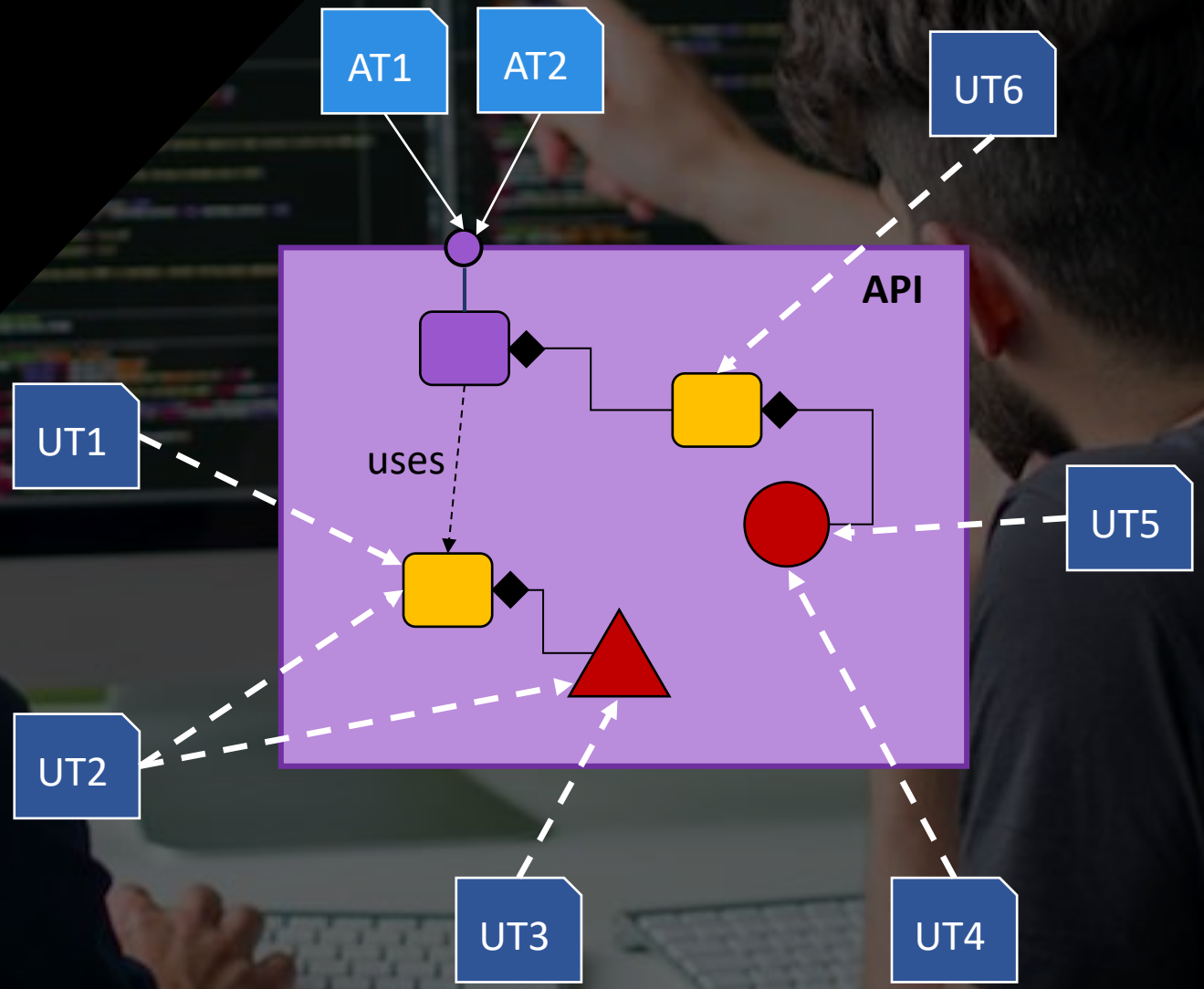
TEST DRIVEN DEVELOPMENT

1.
Beware of…
Fragile tests



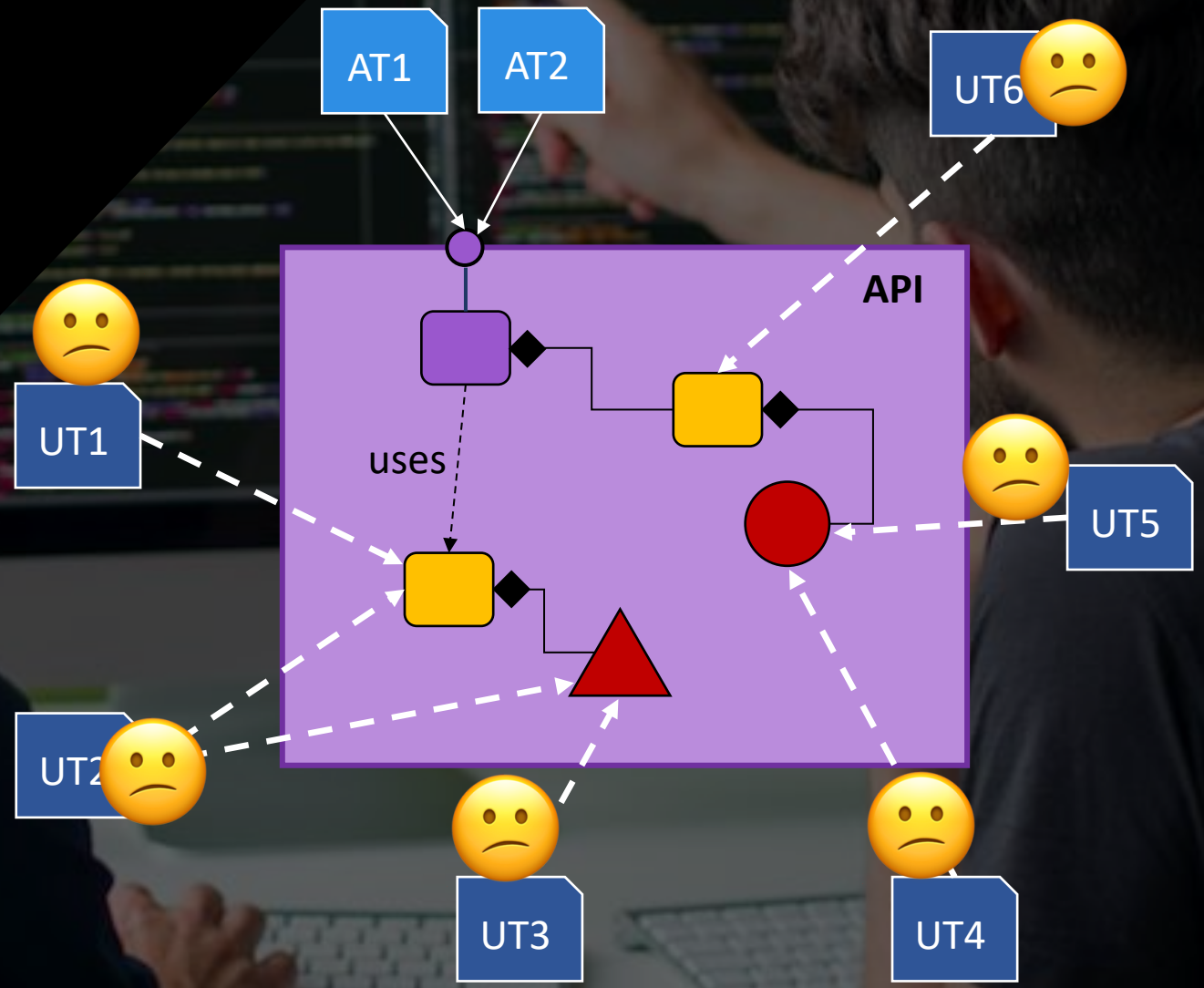
TEST DRIVEN DEVELOPMENT

1.
Beware of…
Fragile tests



TEST DRIVEN DEVELOPMENT

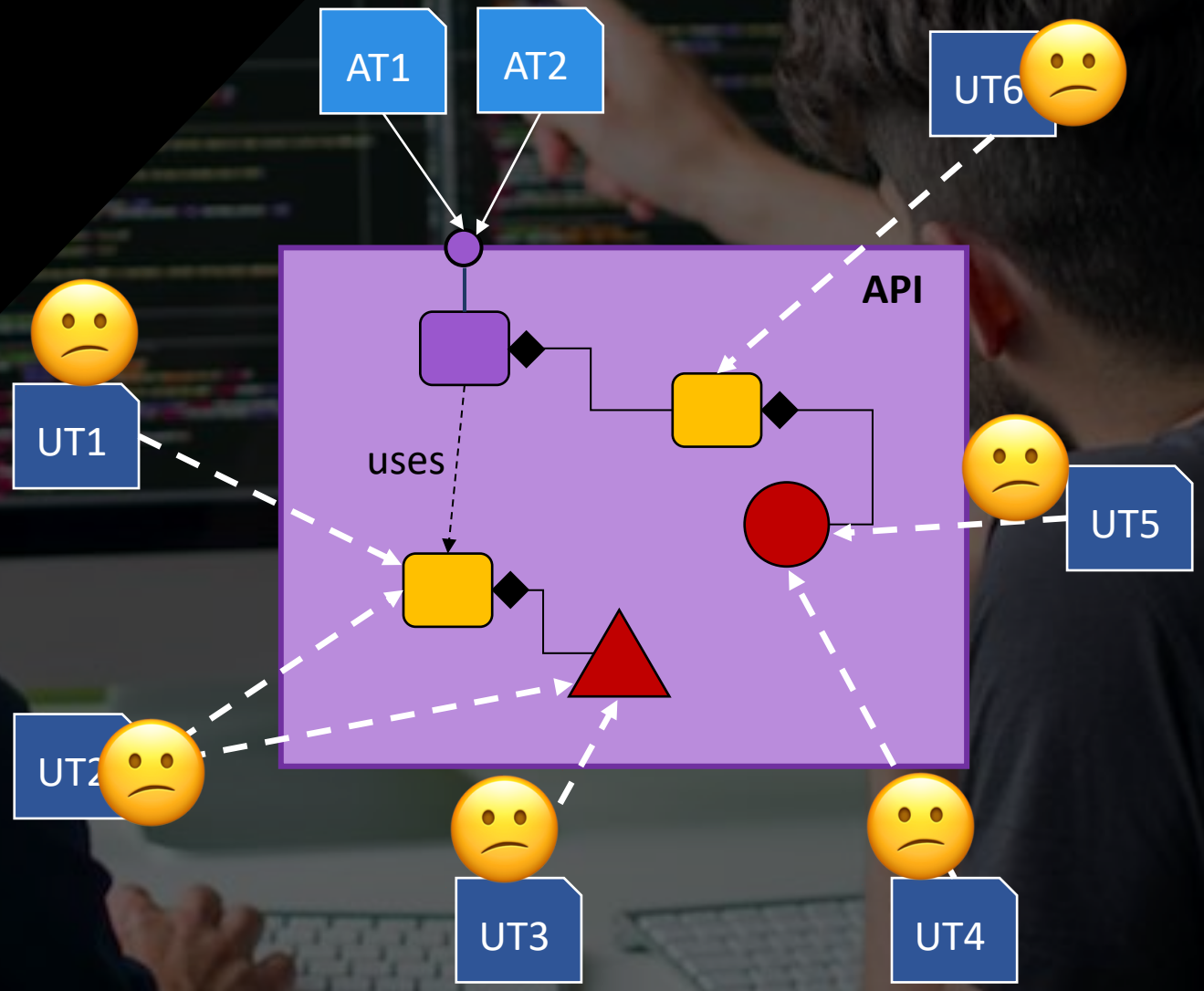
1.
Beware of…
Fragile tests 😕



TEST DRIVEN DEVELOPMENT

Fragile tests

➔ **Less refactoring**

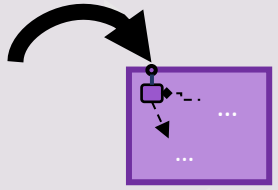


TEST DRIVEN DEVELOPMENT

Fragile tests

MITIGATIONS

- **Do not test “implementations”**
- **Focus on external behaviours instead (with Outside-in TDD)**



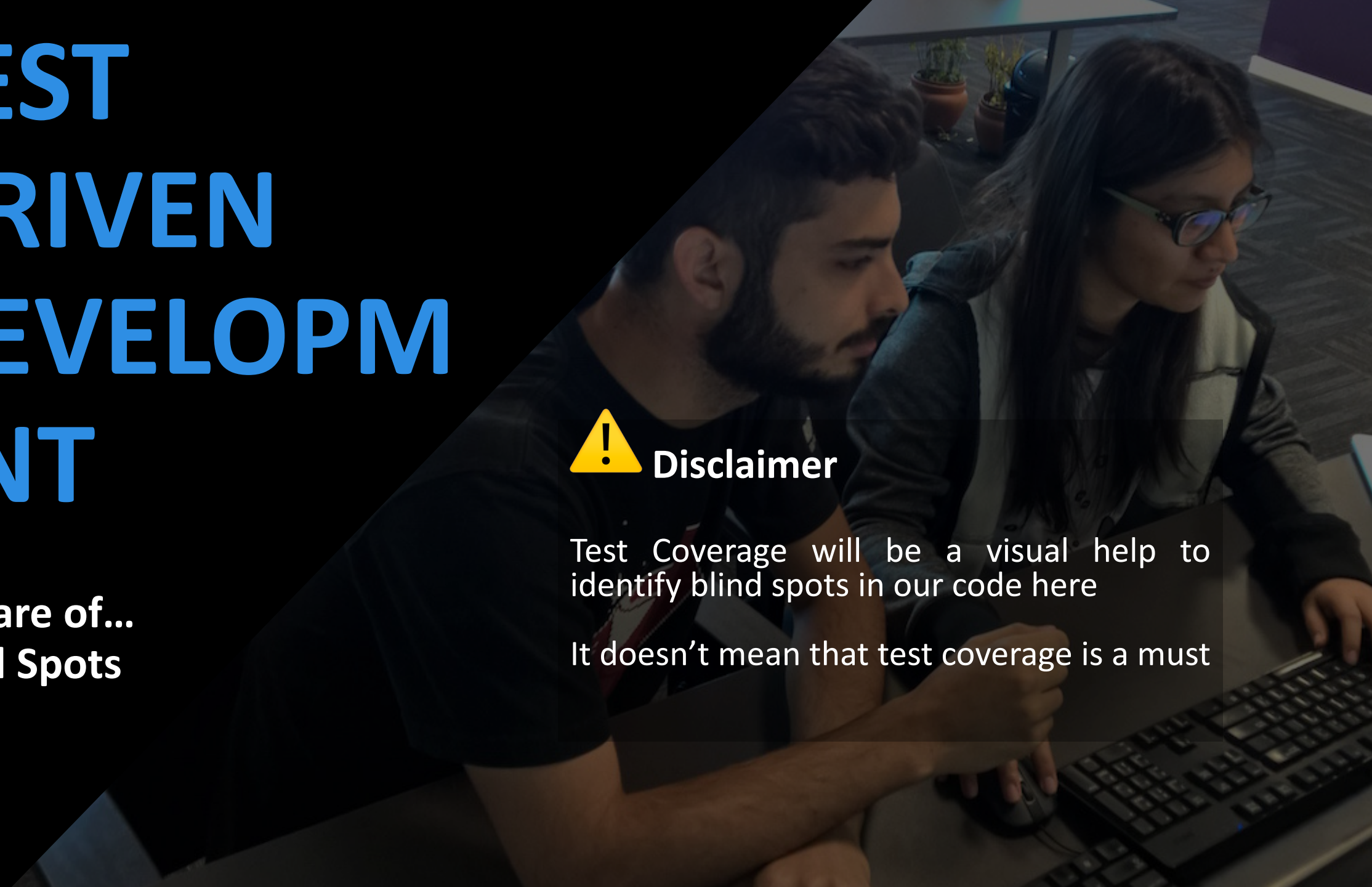
- **Favor (coarse-grained unit) Acceptance tests over fine-grained unit tests**

AT

TEST DRIVEN DEVELOPMENT

2.

Beware of…
Blind Spots



⚠ Disclaimer

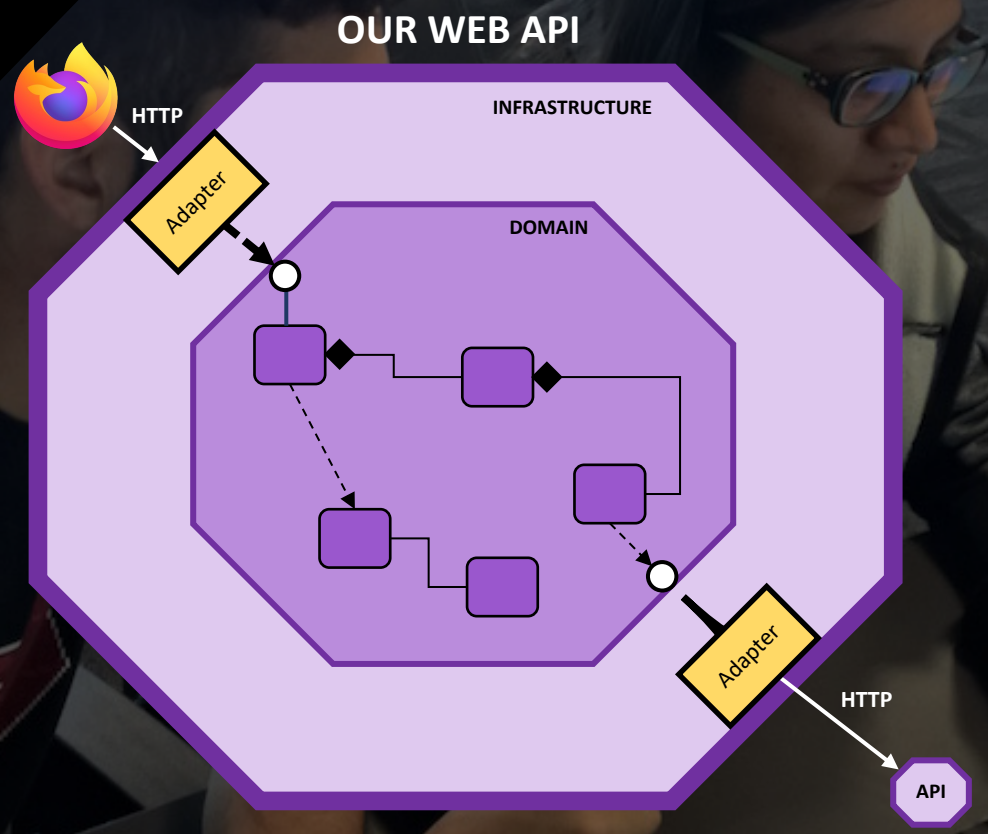
Test Coverage will be a visual help to identify blind spots in our code here

It doesn’t mean that test coverage is a must

TEST DRIVEN DEVELOPMENT

2.

Beware of…
Blind Spots

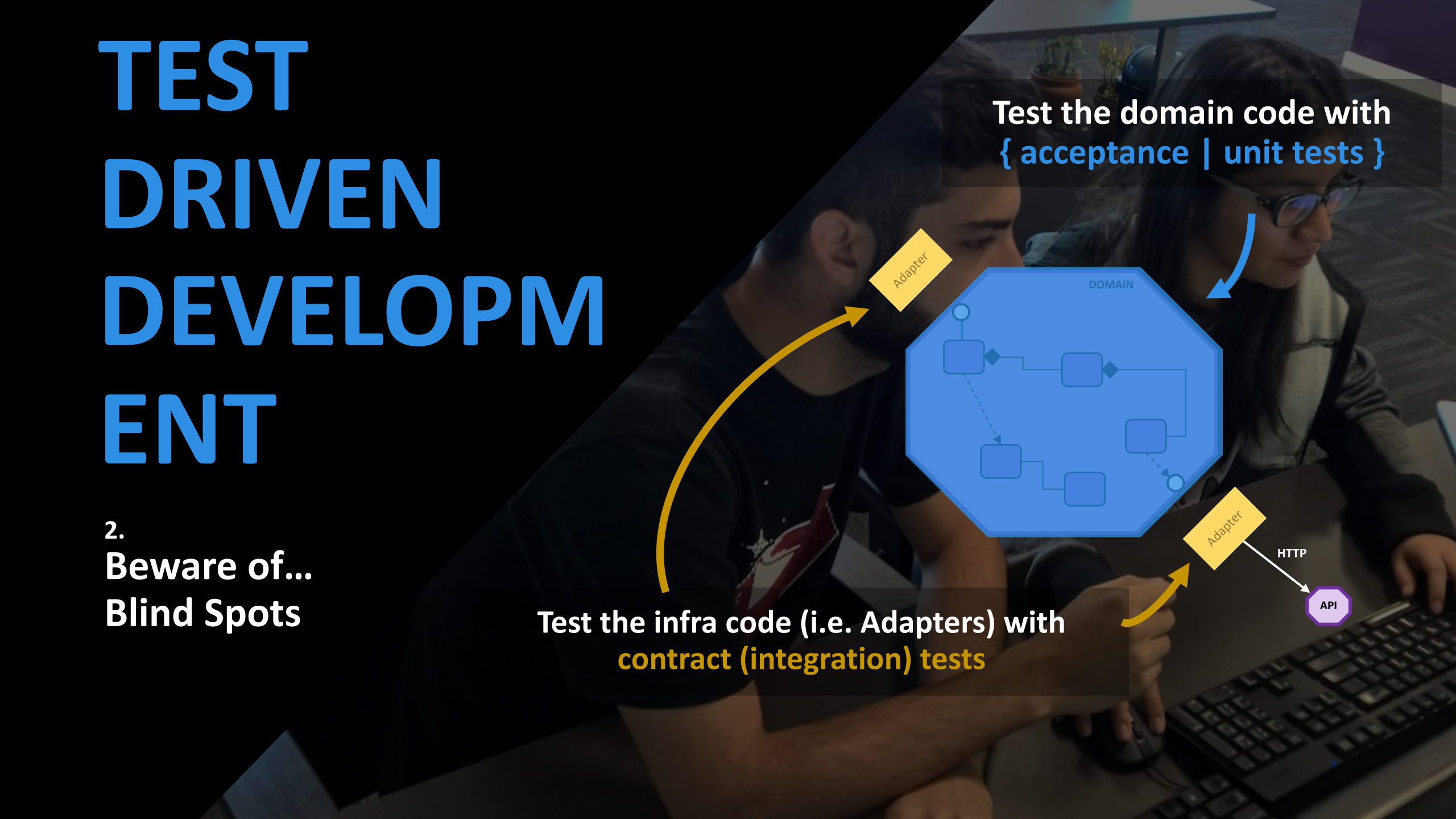


TEST DRIVEN DEVELOPMENT

2.
Beware of…
Blind Spots

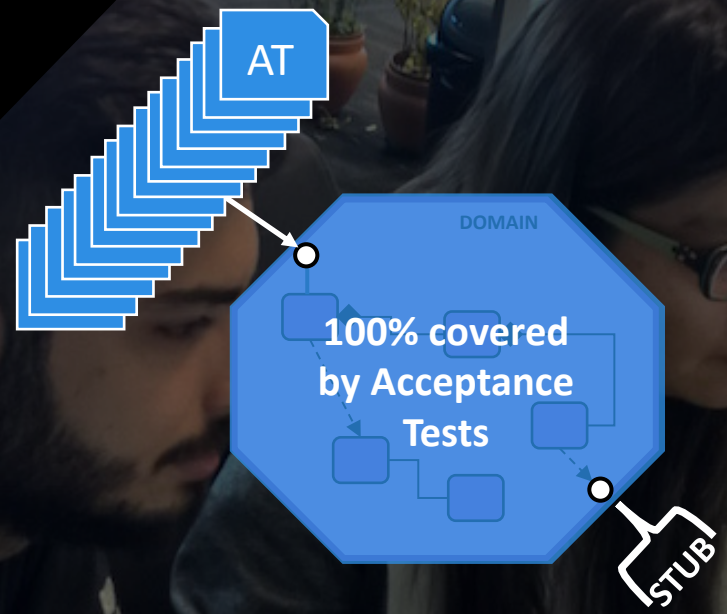
TEST DRIVEN DEVELOPMENT

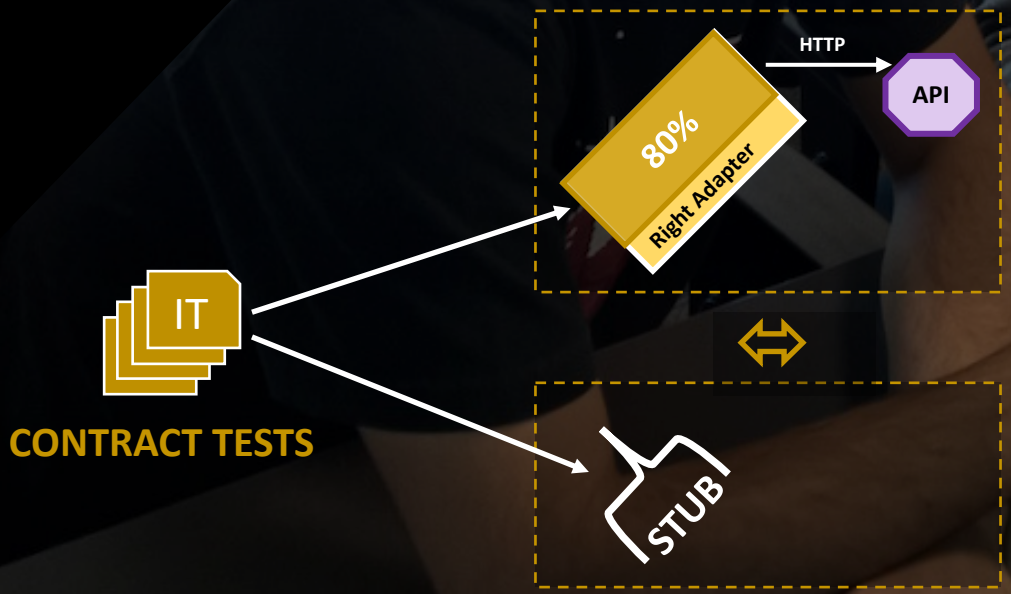
2.
Beware of…
Blind Spots



TEST DRIVEN DEVELOPMENT

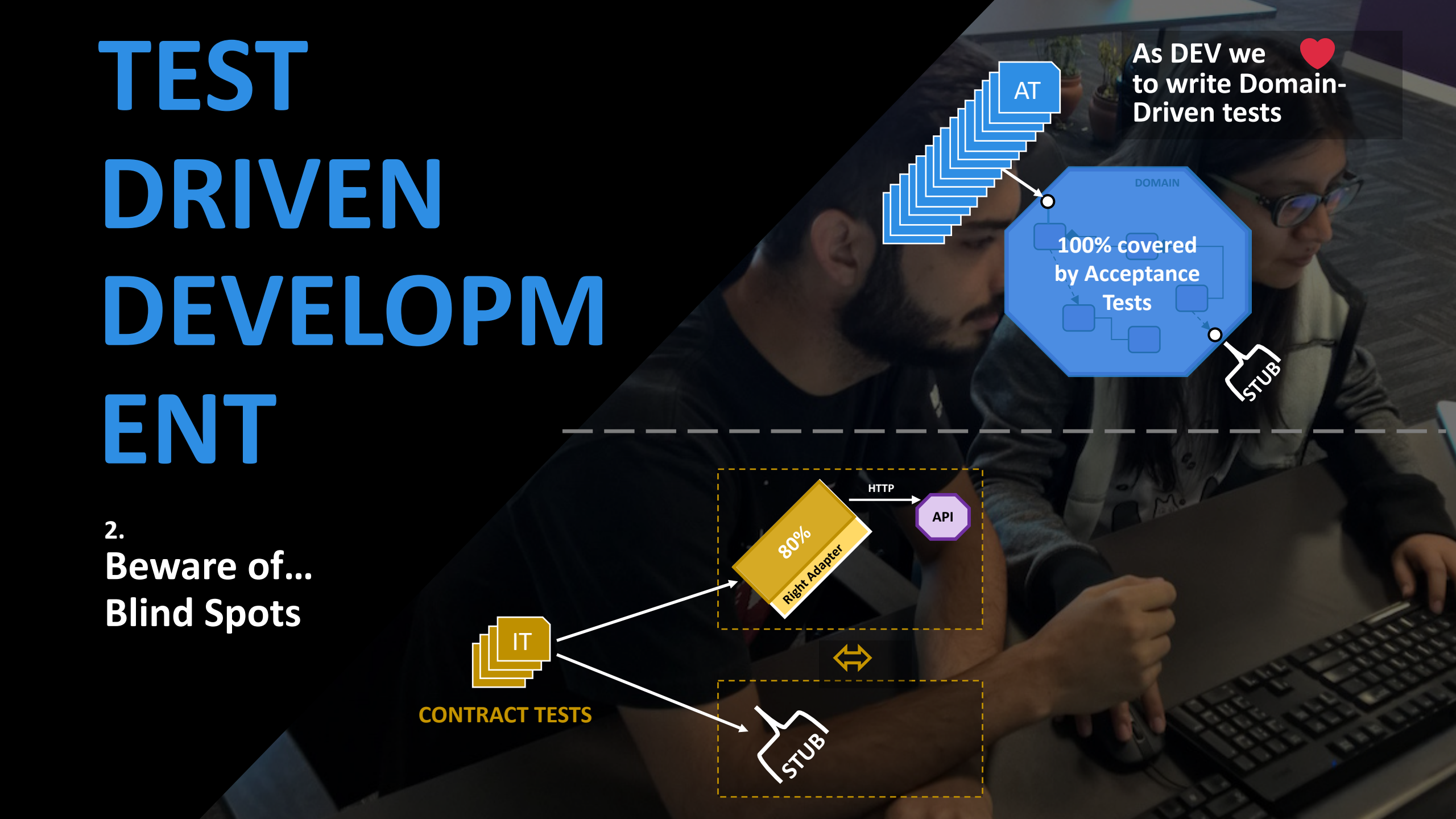
2.
Beware of…
Blind Spots





TEST DRIVEN DEVELOPMENT

2.
Beware of…
Blind Spots



TEST DRIVEN DEVELOPMENT

2.
Beware of... Blind Spots

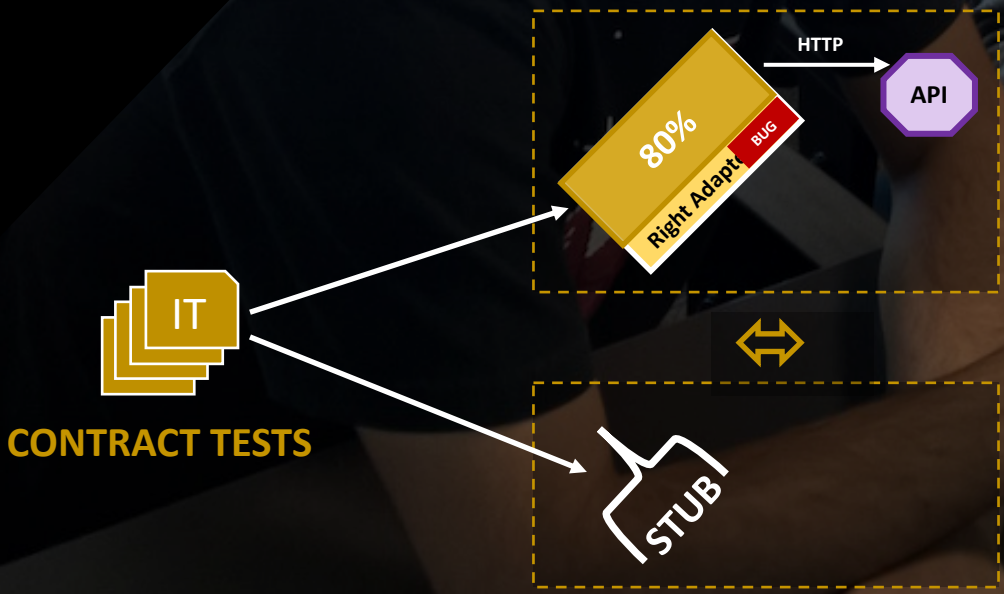
BUG

TEST DRIVEN DEVELOPMENT

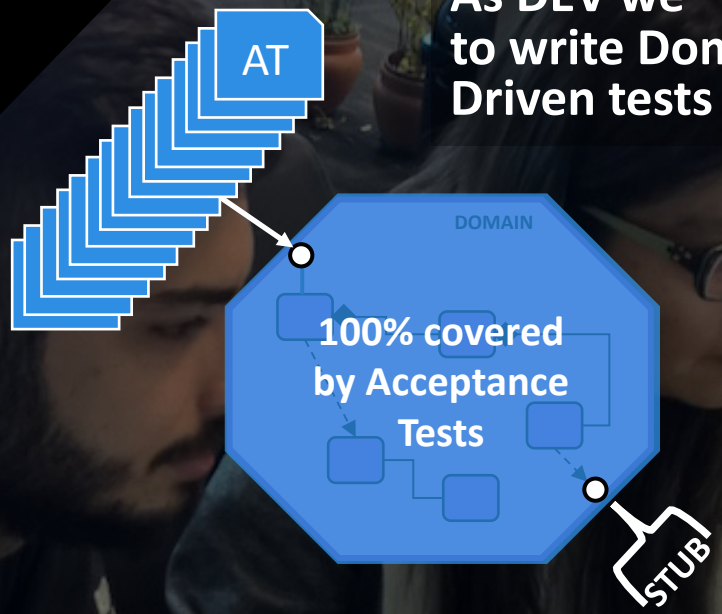
2.

Beware of… Blind Spots

BUG



As DEV we ♥ to write Domain-Driven tests



As DEV we don't write enough integration tests

Because they are slow to run & boring

➔ BUGS

TEST DRIVEN DEVELOPMENT

Blind Spots

MITIGATIONS

- Detect b BUGS Adapters via our beloved ❤ cceptance tests
- Include the Adapters code and stub only the I/Os

TEST DRIVEN DEVELOPMENT

3.

Beware of…

Complex setups

TEST DRIVEN DEVELOPMENT

3.
Beware of…
Complex setups

Test Suite for Rooms Availability

740 lines of Init()

```
    [Fixture]
    public class AvailabilityResolverTest
    {
        private IAvailabilityService _availabilityServiceMock;
        private IAccountService _accountServiceMock;
        private IObservableService _observableServiceMock;
        private IResortContentService _resortContentServiceMock;
        private ITripAdvisorService _tripAdvisorServiceMock;
        private IPictoService _pictoServiceMock;
        private IPricingService _pricingServiceMock;
        private IPackageOptionService _packageOptionService;
        private ICompanyService _companyServiceMock;
        private IContentService _contentServiceMock;
        private IEReputationService _eReputationServiceMock;
        private IGraphQLCache _graphQLCacheMock;

        private AvailabilityResolver _availabilityResolver;

        private AvailabilityResortTypeDto _availabilityResortType;
        private AvailabilityPricesTypeDto _availabilityPricesType;
        private AvailabilityRoomTypeDto _availabilityRoomType;
        private MbRoomStayDto _mbRoomStayType;
        private GetPackagesRequest _mbPackagesRequest;

        private RoomsRateDetailedChargesResponse _roomsRateDetailedChargesType;

        private SearchAvailabilityResponse _searchAvailabilityResponse;
        private SearchAvailabilityResponse _searchAvailabilityResponseWithoutPrices;
```

TEST DRIVEN DEVELOPMENT

3.

Beware of…

Complex setups

Test Suite for Rooms Availability

740 lines of Init()

```
    ResortCode = "FRA21800"
};

_searchAvailabilityResponse = new SearchAvailabilityResponse
{
    AvailableResortPrices = new List<ResortPricesDto>
    {
        new ResortPricesDto
        {
            BestPrice = new PriceDto
            {
                Amount = 100,
                Currency = EurCurrency,
                InitialAmount = 100,
                InitialCurrency = EurCurrency
            },
            SecondPrice = new PriceDto
            {
                Amount = 120,
                Currency = EurCurrency,
                InitialAmount = 120,
                InitialCurrency = EurCurrency
            },
            ResortCode = "RESORT01",
            DiscountType = DiscountType.MemberRate
        },
        new ResortPricesDto
        {
            BestPrice = new PriceDto
```

TEST DRIVEN DEVELOPMENT

3.

Beware of…

Complex setups

Test Suite for Rooms Availability

740 lines of Init()

```
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
```

```
_availabilityServiceMock
    .SearchResortWithFiltersAsync(Arg.Any<SearchAvailabilityRequest>(), Arg.Any<string>(), true, false
                .Returns(_searchAvailabilityResponseWithoutPrices);

_availabilityServiceMock
    .RoomAvailabilityAsync(Arg.Any<AvailabilitySearchParameters>(),
                    Arg.Any<string>(), Arg.Any<string>())
                .Returns(_searchRoomsAvailabilityResponse);

_tripAdvisorServiceMock
    .GetTripAdvisorContentsAsync(Arg.Any<IEnumerable<(string ResortCode, string BrandCode)>>(),
                    Arg.Any<string>())
                .Returns(_tripAdvisorOverviews);

_pictoServiceMock
    .GetPictosAsync(Arg.Any<string>(), Arg.Is(PictoConst.MapIconPictoType))
    .Returns(_pictos.Where(p :PictoDto => p.Type == PictoConst.MapIconPictoType).ToList());

_pictoServiceMock
    .GetPictosAsync(Arg.Any<string>(), Arg.Is(PictoConst.BrandPictoType))
    .Returns(_pictos.Where(p :PictoDto => p.Type == PictoConst.BrandPictoType).ToList());

_pictoServiceMock
    .GetPictosAsync(Arg.Any<string>(), Arg.Is(PictoConst.BedTypePictoType))
    .Returns(_pictos.Where(p :PictoDto => p.Type == PictoConst.BedTypePictoType).ToList());

_resortContentServiceMock
    .GetResortOverviewInformation(Arg.Any<string>(), Arg.Any<string>())
                .Returns(_resortInfo);
```

TEST DRIVEN DEVELOPMENT

3.

Beware of…

Complex setups

Test Suite for Rooms Availability

```
[Test(Description = "To be deleted because resolver no longer used")]
public async Task
    0 references
    AvailabilityResolverUnitTest_Should_ReturnAvailabilityResponseWhenAvailabilityResolveMethodIsCalled_Wi
{
    // Act
    _availabilityResolver.Resolve(_objectGraphType);

    // Assert
    Check.That(_objectGraphType.HasField("availabilityResortsV2")).IsTrue();

    var availabilityField = _objectGraphType.Fields.FirstOrDefault(a :FieldType => a.Name == "availabilityRe
    var availabilityResort = (await ((Task<object>)availabilityField.Resolver.Resolve(
        new ResolveFieldContext
        {
            Arguments = new Dictionary<string, object>
            {
                ["availabilityInput"] = _availabilityResortType
            }
        }))) as MbAvailabilityResortDto;

    var expectedResortCode = _searchAvailabilityRes
    var expectedBrandCode = _searchAvailabilityResp
    var expectedBrandMapIconPicto = _pictos
        .FirstOrDefault(p :PictoDto => p.Key == expectedBrandCode && p.Type
    var expectedBrandPicto = _pictos
        .FirstOrDefault(p :PictoDto => p.Key == expectedBrandCode && p.Type

    var expectedOtherBrandResortCode = _searchAvailabilityResponse.AvailableOthersBrandsResortsList.Elemen
    var expectedOtherBrandBrandCode = _searchAvailabilityResponse.AvailableOthersBrandsResortsList.Element
```

Tests using initialized fields

➔ side effects nightmare

TEST DRIVEN DEVELOPMENT

3.
Beware of…
Complex setups

MITIGATIONS

- Avoid cognitive overload as much as possible
  The power of sameness
- Favor local variables over test suite members
  Everything should be created from the test
- Use Domain-Driven Builders to shorten and explicit the Arrange section of your tests
- **Treat your test code as production code!**
  ➔ merciless refactoring

TEST DRIVEN DEVELOPMENT

Beware of…

1. Fragile tests
2. Blind spots
3. Complex setups

TEST DRIVEN DEVELOPMENT

Mitigations

Do not test “implementation details”

When we test Implementations instead of behaviours :-(

Favor Acceptance tests
(coarse-grained unit tests)

1. Fragile tests
2. Blind spots
3. Complex setups

TEST DRIVEN DEVELOPMENT

Mitigations

Do not test “implementation details”

Favor Acceptance tests
(coarse-grained unit tests)

Include the Adapters to your Acceptance tests

- Test the adaptation in action
(All corner cases covered)

- Stub only the I/O
(super fast tests)

1. Fragile tests
2. Blind spots
3. Complex setups

TEST DRIVEN DEVELOPMENT

Mitigations

Do not test “implementation details”

Favor Acceptance tests
(coarse-grained unit tests)

Include the Adapters to your Acceptance tests

- Test the adaptation in action
(All corner cases covered)

- Stub only the I/O
(super fast tests)

Use Domain-Driven Builders to shorten and explicit the Arrange section of your tests

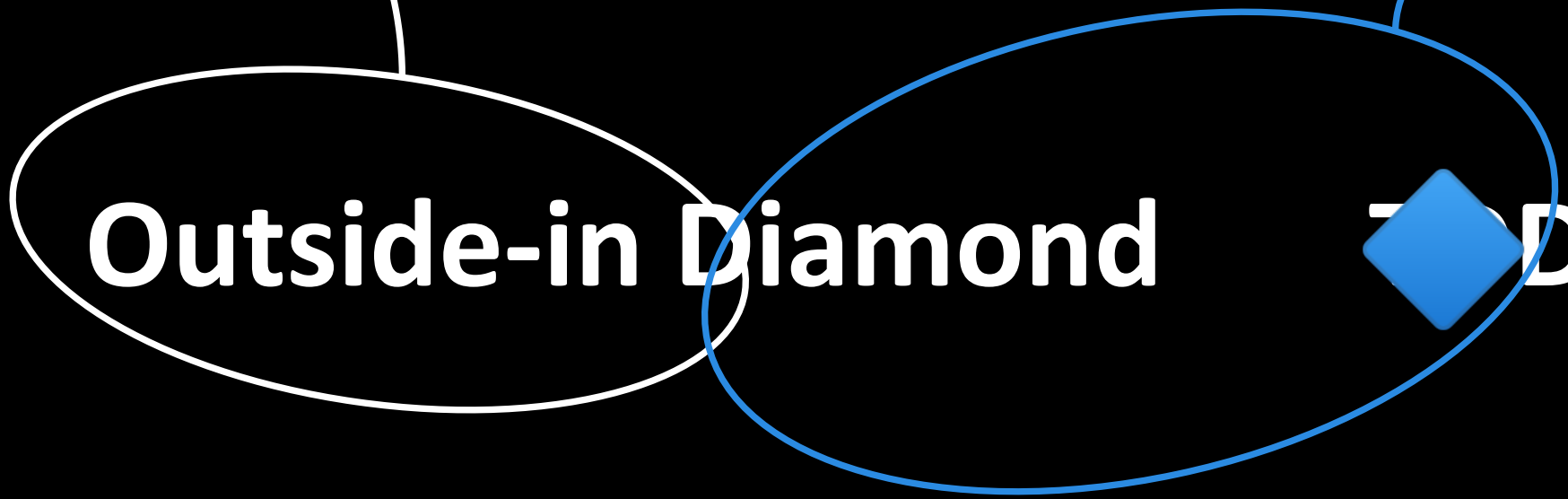
1. Fragile tests
2. Blind spots
3. Complex setups

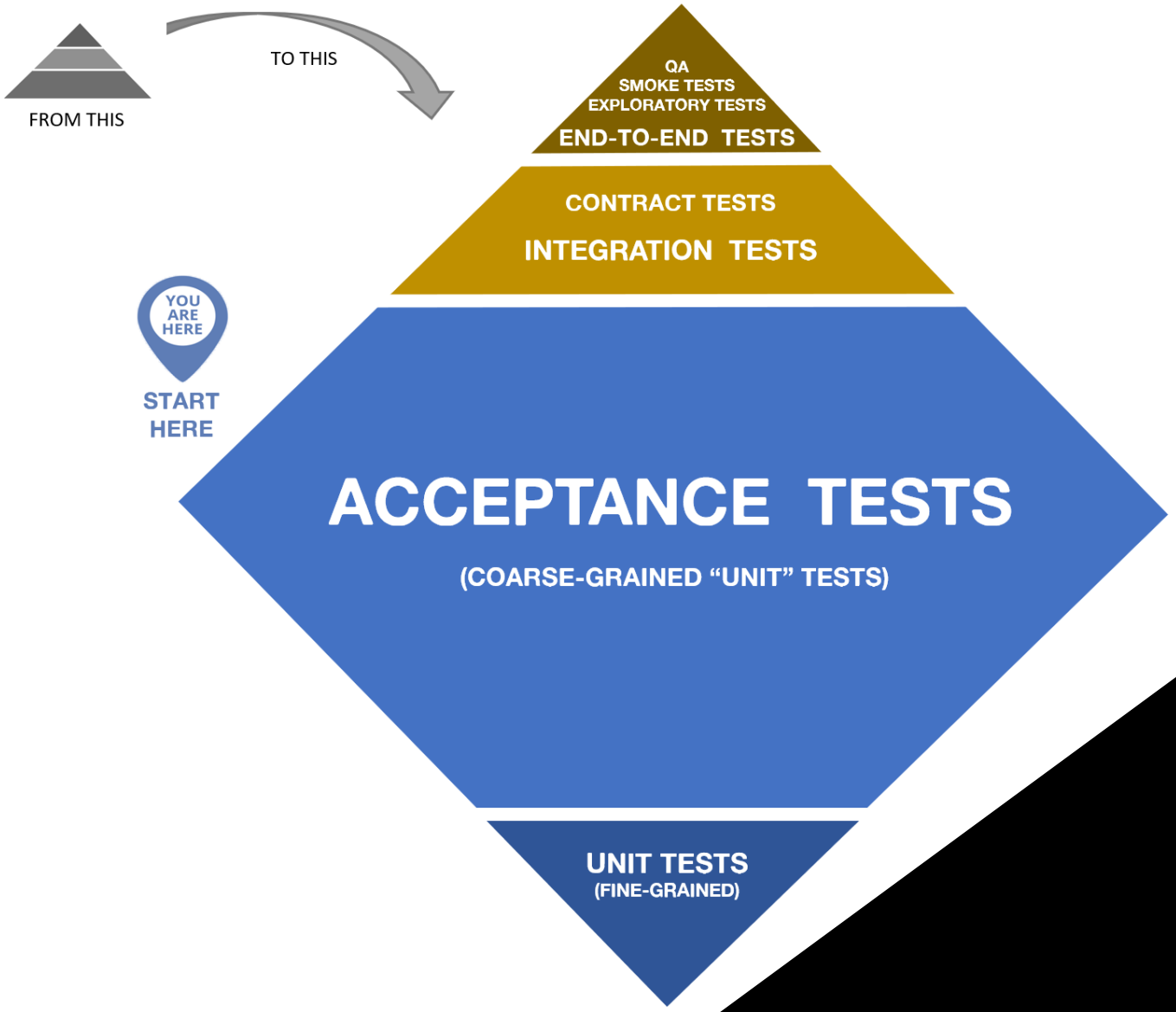
Trade offs made in reaction to people’s behaviors
(observed around me over and over and over again)

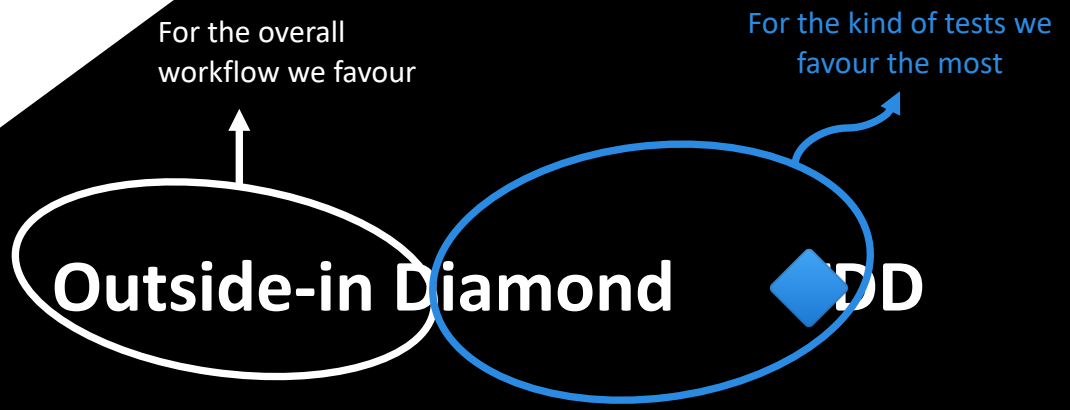
Outside-in Diamond 🔷D

For the overall workflow we favor

For the kind of tests we favor the most




FROM THIS
TO THIS
QA
SMOKE TESTS
EXPLORATORY TESTS
END-TO-END TESTS
CONTRACT TESTS
INTEGRATION TESTS
YOU ARE HERE
START HERE
ACCEPTANCE TESTS
(COARSE-GRAINED "UNIT" TESTS)
UNIT TESTS
(FINE-GRAINED)

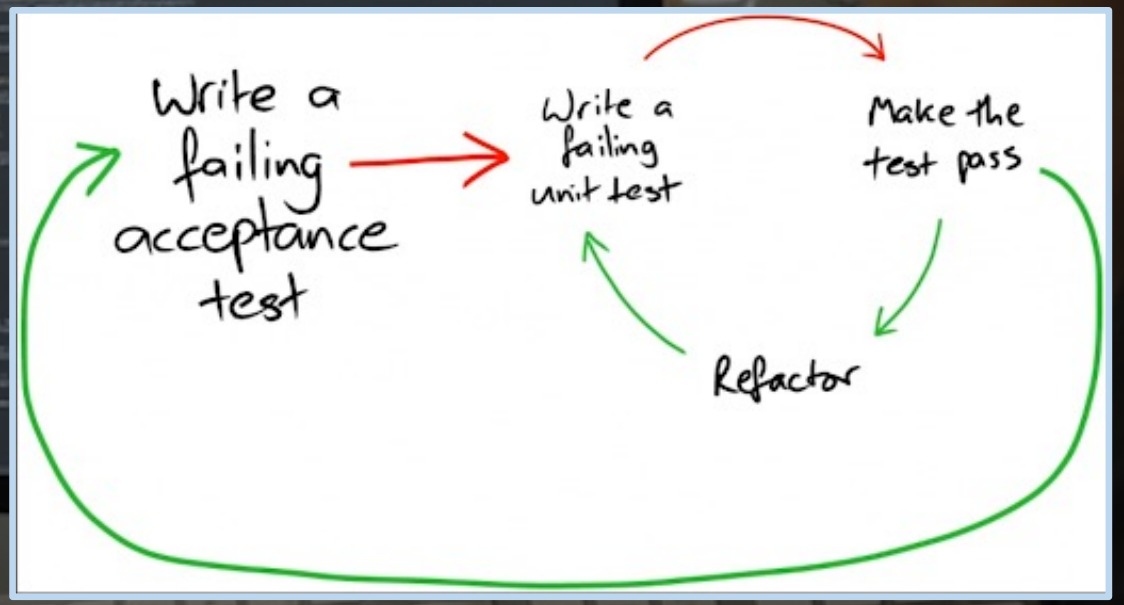

For the overall workflow we favour
For the kind of tests we favour the most
Outside-in Diamond
DD

TEST DRIVEN DEVELOPMENT

Outside-in
but…

Double loop?

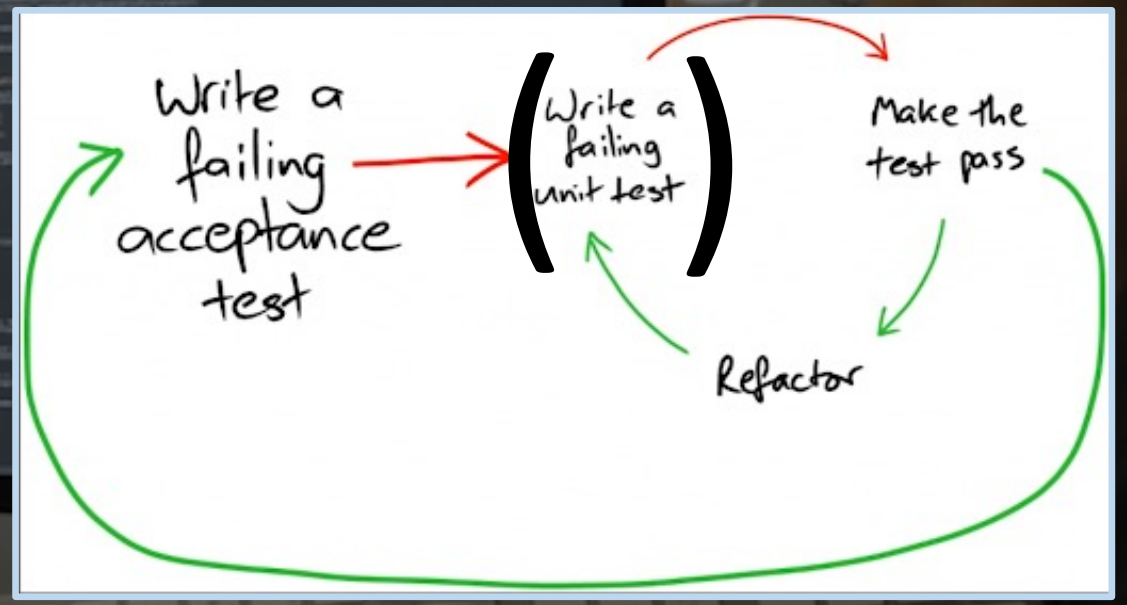
TEST DRIVEN DEVELOPMENT



Outside-in but…

Double loop

TEST DRIVEN DEVELOPMENT

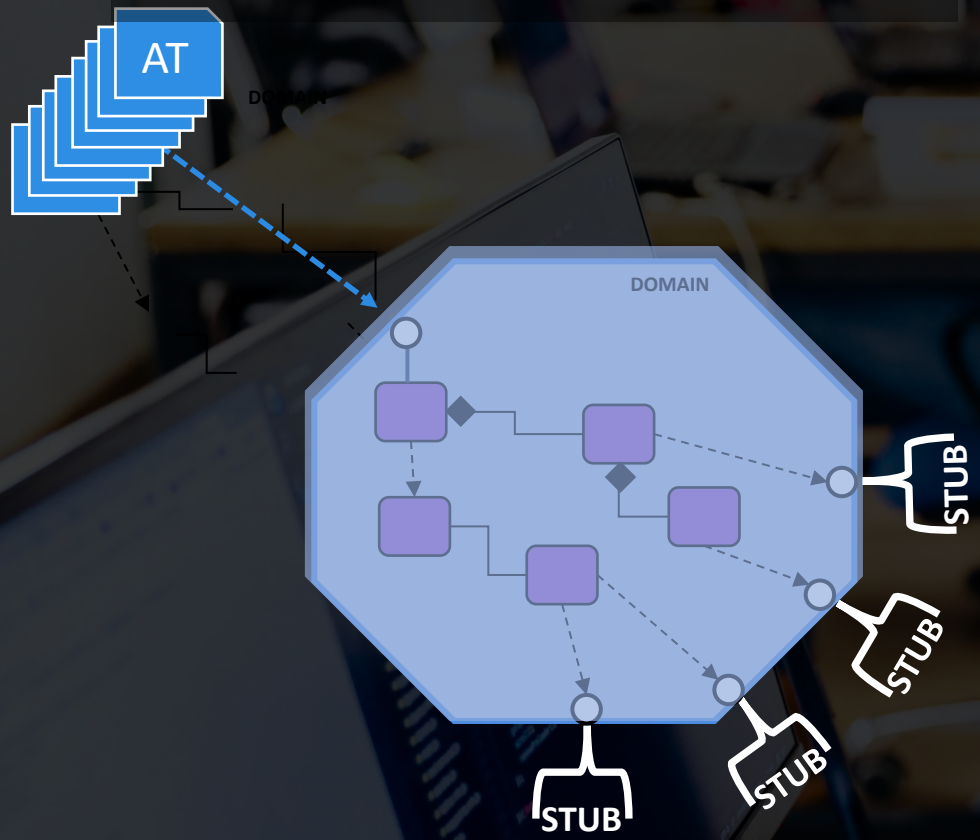


Outside-in
but…

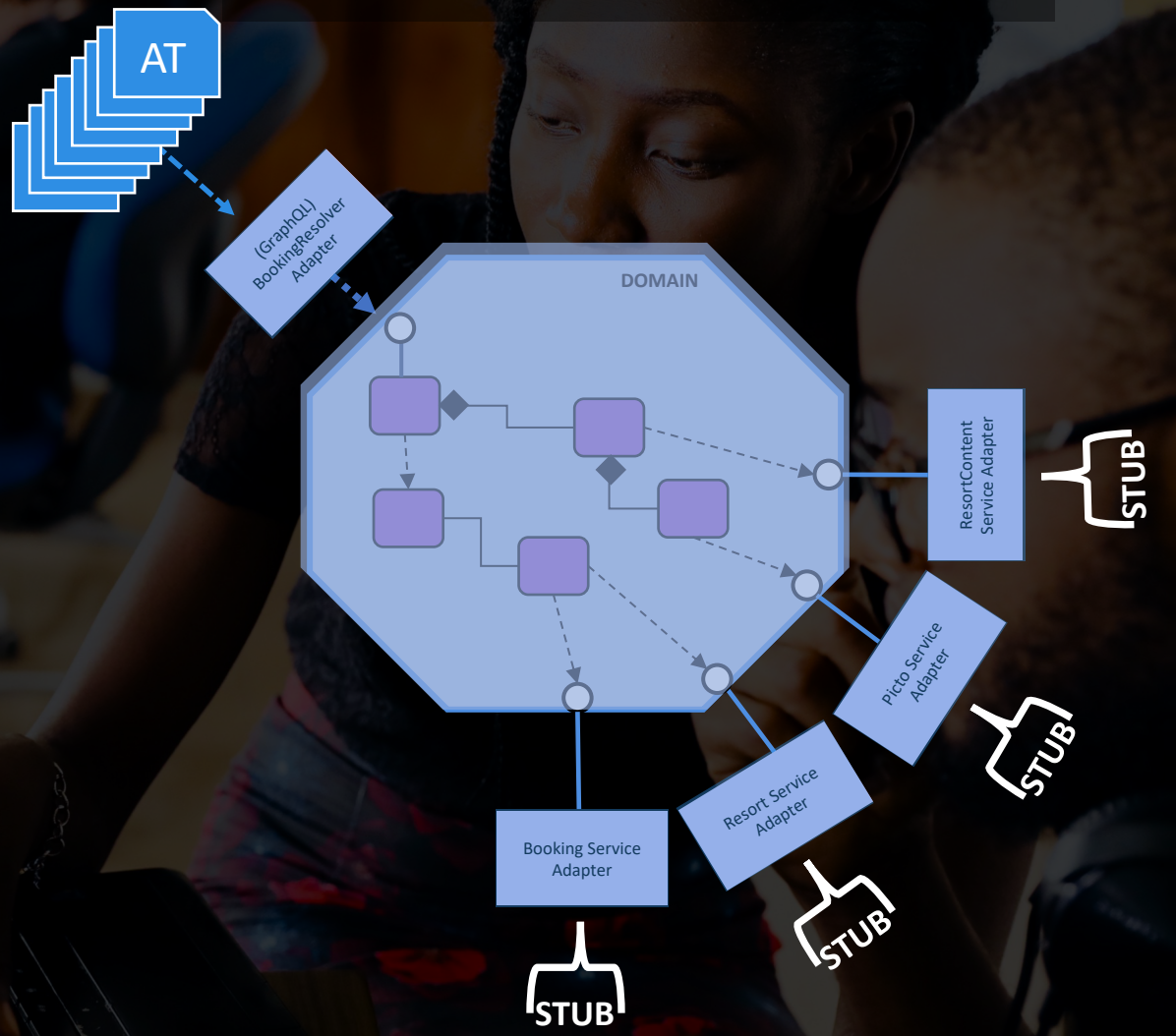
~~Double loop~~

One and a half loop

By-the-book tests of HEXAGONAL ARCHITECTURE



Outside-In Diamond ◆tests of HEXAGONAL ARCHITECTURE



~~Ports~~ Last-miles stubs

⚠ Not supported by Alistair

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -------------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT ----------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -------------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT ----------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -------------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT ----------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

Read it as a sentence divided into 2 parts

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -------------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(2).ToArray();

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT ----------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

Fuzzers help us

- to randomly detect hard-coded values or unsupported cases

OUTSIDE-IN DIAMOND ◆

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -----------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(2).ToArray();

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilities(expectedCity, roomTypesRequestedByTheClient));

        // --- ASSERT --------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

Fuzzers help us

- to randomly detect hard-coded values or unsupported case

- to shorten our test setup

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -----------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT --------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

Builders allow Behavioral & Domain-Driven intentions (no tech details here)

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -----------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT --------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

Builders allow Behavioral & Domain-Driven intentions (no tech details here)

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -----------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT --------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

Builders allow Behavioral & Domain-Driven intentions (no tech details here)

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -----------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT --------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

Builders allow Behavioral & Domain-Driven intentions (no tech details here)

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -----------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT --------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

Builders allow Behavioral & Domain-Driven intentions (no tech details here)

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -----------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT --------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);

    }
```

Builders allow Behavioral & Domain-Driven intentions (no tech details here)

Very important!

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzz
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -------------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await webControlle
            .GetRoomsAvailabilities(new (CityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT ----------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

Injecting the result of our builder

We test from the left-side Adapter
(here a web controller)

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -----------------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT --------------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity.Name, bellagioHotel.ResortId);
    }
```

We define the end-user request

No I/O here, but we test
the whole { black-box / component / API }

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -------------------------------------------------------------------
        var roomTypesRequestedByTheClient = fuzzer.GenerateRoomTypes(roomTypesCount: 2);

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT ----------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

Helper method to sum-up in “one line” our assert intention

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        private static void CheckThatRoomsAreMatchingTheRequestedCriteria(IActionResult response, string[] roomTypesRequestedByTheClient,
                                                                          City expectedCity, params string[] expectedResortIds)
        {
            response.CheckIsOk200();

            var roomsAvailabilities = response.ExtractValue<RoomsAvailabilities>();
            Check.That(roomsAvailabilities)
                .HasSize(roomTypesRequestedByTheClient.Count())
                .And.ContainsOnlyElementsThatMatch(roomAvailability => roomAvailability.CityName == expectedCity.Name)
                .And.ContainsOnlyElementsThatMatch(roomAvailability => roomAvailability.RoomType.IsOneOf(roomTypesRequestedByTheClient))
                .And.ContainsOnlyElementsThatMatch(roomAvailability => roomAvailability.ResortId == expectedResortIds.First());
        }

        // --- ACT -------------------------------------------------------------------

        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await _client
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT ----------------------------------------------------------------

        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

“One line” to hide our technical assertions towards DTOs etc.

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        private static void CheckThatRoomsAreMatchingTheRequestedCriteria(IActionResult response, string[] roomTypesRequestedByTheClient,
                                                                          City expectedCity, params string[] expectedResortIds)
        {
            response.CheckIsOk200();

            var roomsAvailabilities = response.ExtractValue<RoomsAvailabilities>();
            Check.That(roomsAvailabilities)
                .HasSize(roomTypesRequestedByTheClient.Count())
                .And.ContainsOnlyElementsThatMatch(roomAvailability => roomAvailability.CityName == expectedCity.Name)
                .And.ContainsOnlyElementsThatMatch(roomAvailability => roomAvailability.RoomType.IsOneOf(roomTypesRequestedByTheClient))
                .And.ContainsOnlyElementsThatMatch(roomAvailability => roomAvailability.ResortId == expectedResortIds.First());
        }

        // --- ACT -------------------------------------------------------------------

        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await _client
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT ----------------------------------------------------------------

        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

"One line" to hide our technical assertions towards DTOs etc.

"One line" to reuse the same check intention, but with various adapters

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -------------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT ----------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -------------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT ----------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -----------------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT --------------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

OUTSIDE-IN DIAMOND ◆

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -------------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT ----------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -------------------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT ----------------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -------------------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT ----------------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

ARRANGE

ACT

ASSERT

OUTSIDE-IN DIAMOND

Fuzzers to shorten & speed setup

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -------------------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT ----------------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

ARRANGE

ACT

ASSERT

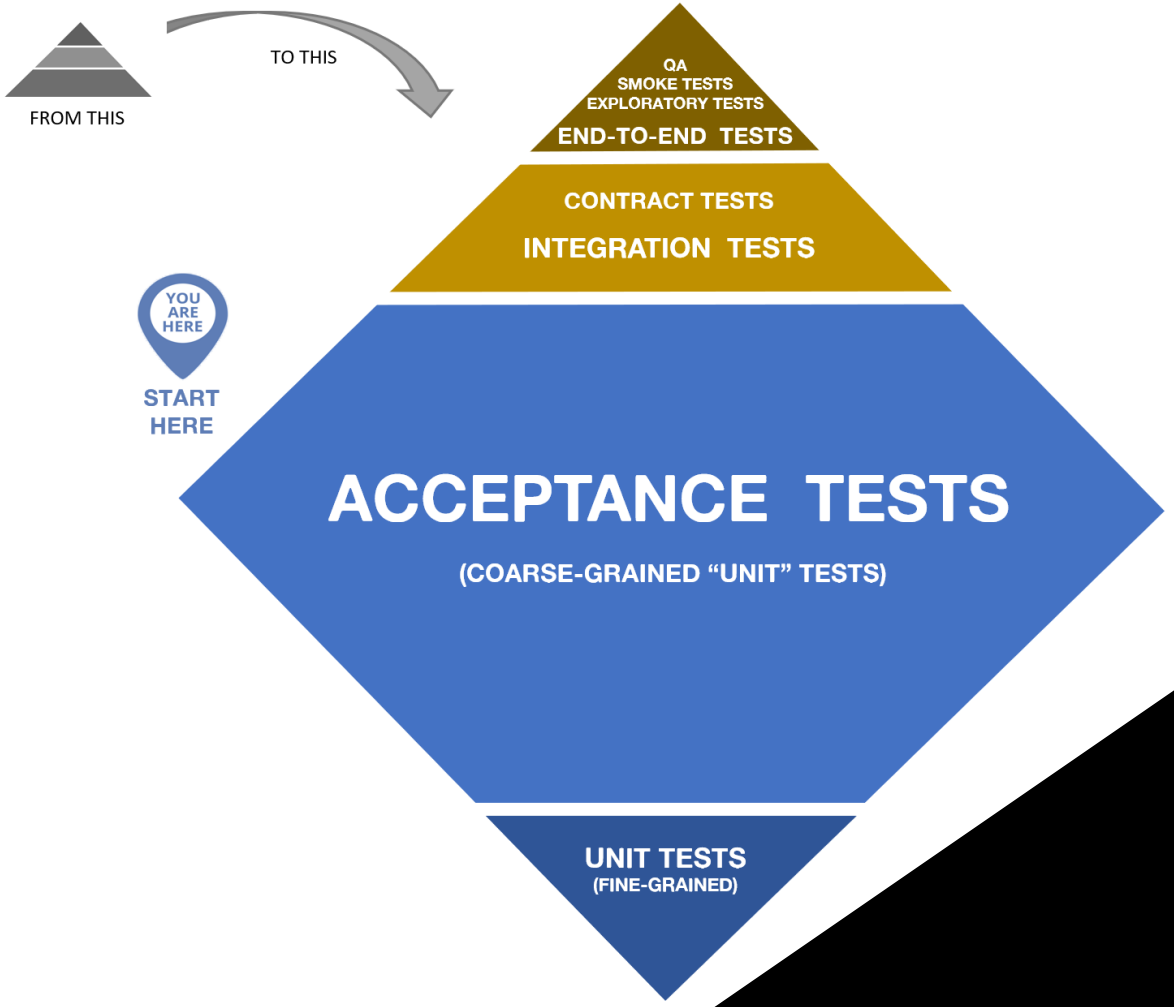
Builders to shorten and “domain-driven” our test setup

Domain-Driven!

OUTSIDE-IN DIAMOND

```
[TestFixture]
0 references
public class AvailabilityControllerShould
{
    [Test]
    0 references
    public async Task Return_RoomAvailabilities_giving_a_city_name_and_a_list_of_requested_RoomTypes()
    {
        var fuzzer = new Fuzzer();

        var expectedCity = fuzzer.GenerateCity();
        var bellagioHotel = fuzzer.GenerateHotel(expectedCity, supportedRoomTypes: fuzzer.GenerateRoomTypes(roomTypesCount: 3));
        var otherHotelInTownWithAllPossibleRoomTypes = fuzzer.GenerateHotel(expectedCity);

        var availabilityService = new AvailabilityServiceBuilder(fuzzer)
                                      .WithAffiliatedHotels(bellagioHotel, otherHotelInTownWithAllPossibleRoomTypes)
                                      .HavingSomeFullyBookedHotels(otherHotelInTownWithAllPossibleRoomTypes)
                                      .GeneratingOneAvailabilityPerSupportedRoomType()
                                      .Build();

        var webController = new AvailabilityController(availabilityService);

        // --- ACT -------------------------------------------------------------------
        var roomTypesRequestedByTheClient = bellagioHotel.SupportedRoomTypes.Skip(1).ToArray();

        var response = await webController
            .GetRoomsAvailabilities(new QueryRoomsAvailabilitiesDto(cityName: expectedCity.Name, roomTypesRequestedByTheClient));

        // --- ASSERT ----------------------------------------------------------------
        CheckThatRoomsAreMatchingTheRequestedCriteria(response, roomTypesRequestedByTheClient,
                                                      expectedCity, expectedResortIds: bellagioHotel.ResortId);
    }
```

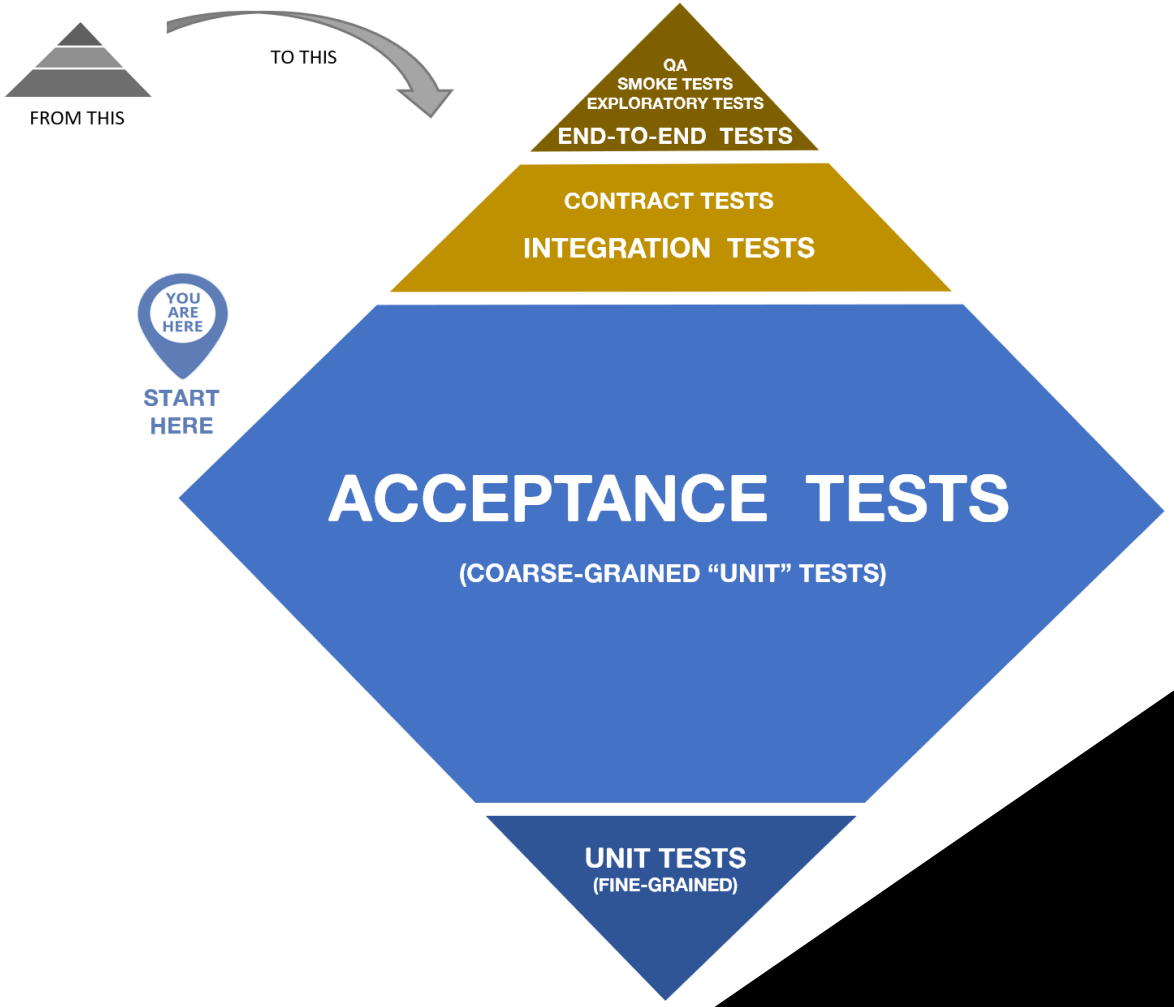
Wrap up



Thomas PIERRAIN
pierrain (use case driven)

Outside-in Diamond 🔷DD

- Elaborated from & for the people
- Write fast & Antifragile tests (❤️refactoring)
- DDD but covers more blind spots on the tech side



Thomas PIERRAIN
pierrain (use case driven)

Outside-in Diamond 🔷DD

- Elaborated from & for the people
- Write fast & Antifragile tests (❤️refactoring)
- DDD but covers more blind spots on the tech side

Any Questions?

BINARY IS FOR MACHINE
CODE IS FOR PEOPLE

WE CARE PEOPLE

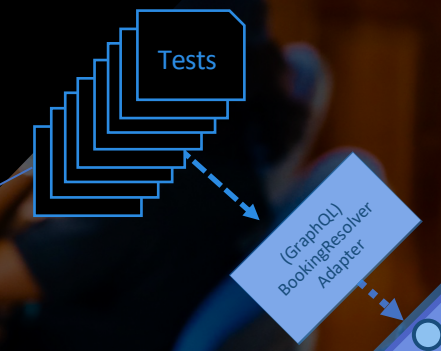
Special thanks to: **Bruno BOUCARD**, **Cyrille DUPUYDAUBY** & **Rui CARVALHO** for their kind reviews & feedbacks

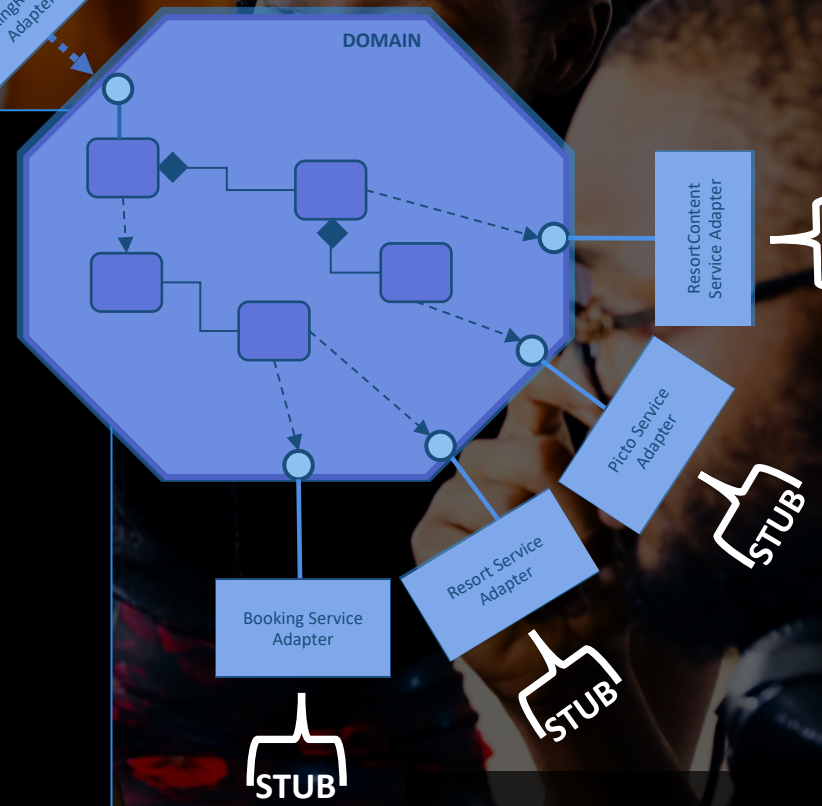
OUTSIDE-IN DIAMOND

TDD WITH HEXAGONAL ARCHITECTURE

#OUTSIDEINDIAMOND

The infra code (i.e. Adapters) is also covered by our { acceptance | unit tests }






```
[Test]
0 references
public async Task Return_the_MainPicture_of_the_HIM_resort_as_the_MainPicture_of_the_BookingInfos()
{
    var fuzzer = new Fuzzer();
    var account = new AccountSpecification(fuzzer);
    var resort = new ResortSpecification(fuzzer);
    var confirmationNumber = fuzzer.GenerateBookingConfirmationNumber();

    var bookingResolver = new BookingResolverBuilder(fuzzer)
        .WithAccount(account)
        .WithBooking(new BookingSpecification(account.AccountId, fuzzer, confirmationNumber, resort.ResortCode))
        .WithResorts(resort)
        .Build();

    var arguments = new Dictionary<string, object>
    {
        ["confirmationNumber"] = confirmationNumber,
        ["currencyCode"] = fuzzer.GenerateCurrencyCode(),
        ["locale"] = fuzzer.GenerateLocale(),
        ["personName"] = account.LastName
    };

    var bookingInfosDto = await bookingResolver.ExecuteGraphQLQuery<MbBookingDto>(BookingResolver.BookingInfosFieldName,
                                                                                  arguments, Substitute.For<IGraphQLCache>());

    Check.That(bookingInfosDto.MainPicture).IsEqualTo(resort.ResortMainPictureUrlForHIM);
}
```

Fuzzers & Specification objects to ease test setup & explore more

Builders to shorten and "domain-driven" our hexagonal setup for testing

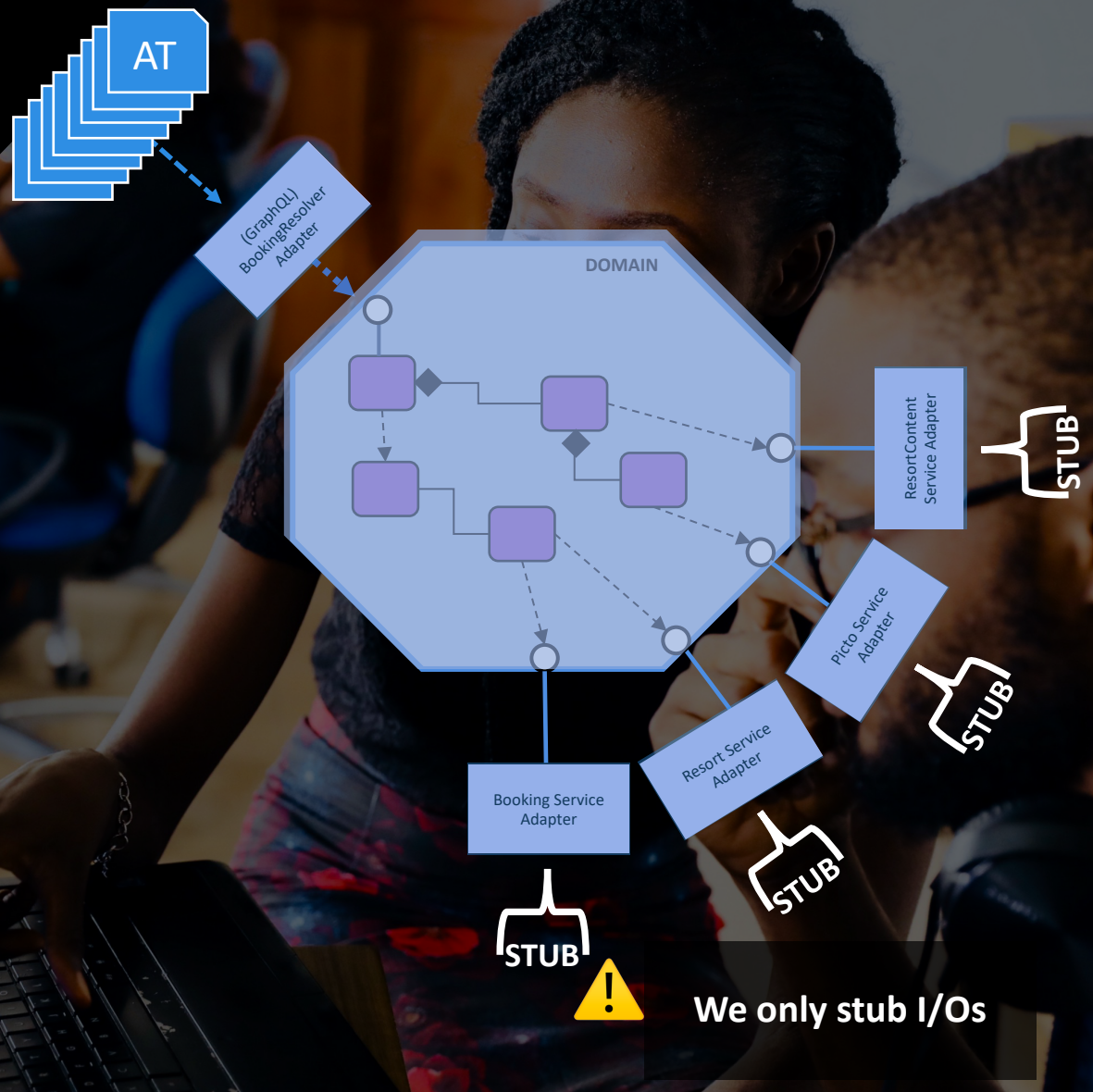
We only stub I/Os

HOW BY-THE-BOOK PEOPLE DO TEST

OUTSIDE-IN DIAMOND

the HEXAGONAL ARCHITECTURE case

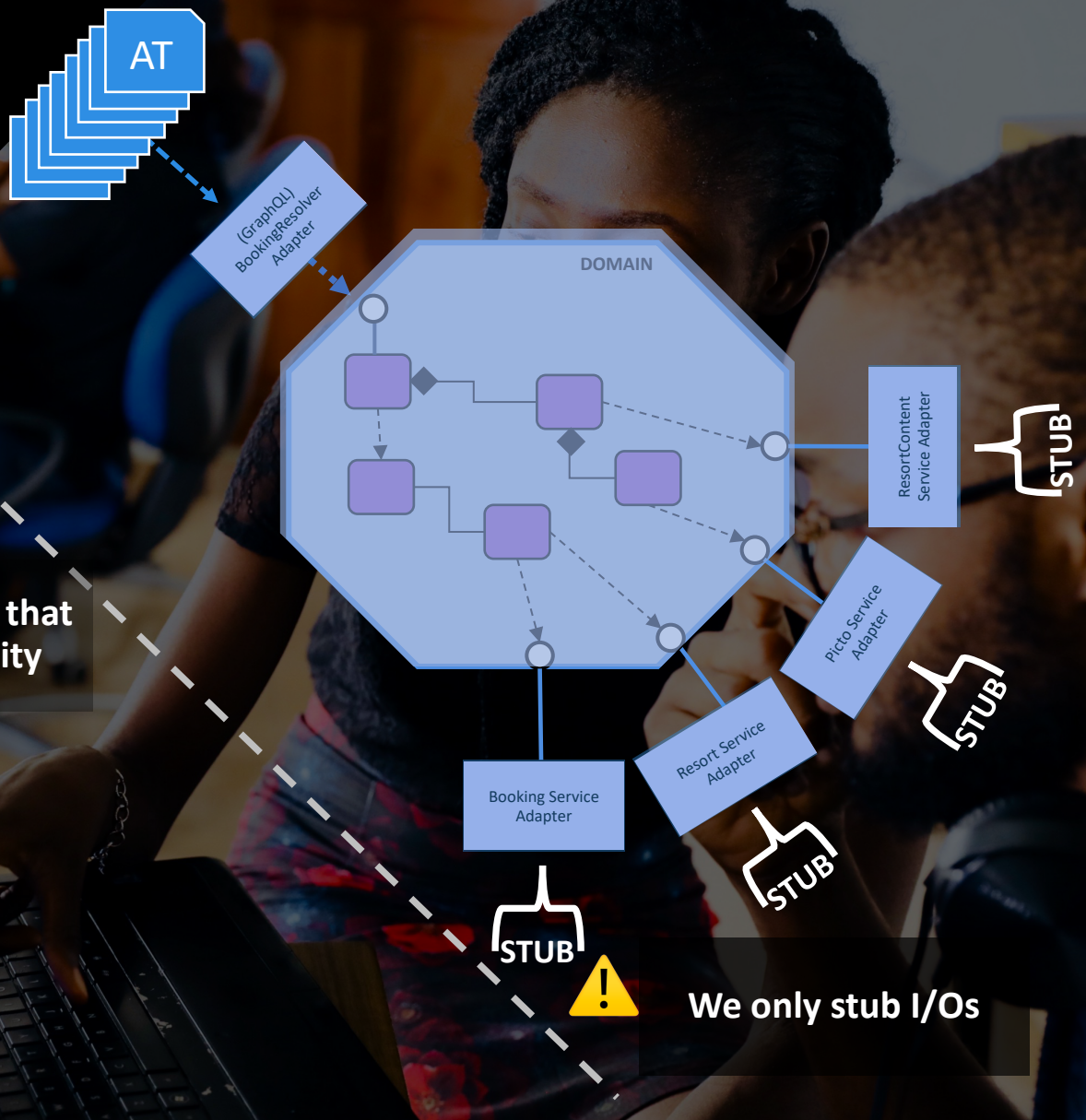
Most of the infra code (i.e. Adapters) is also covered by our
{ acceptance | unit tests }



We only stub I/Os

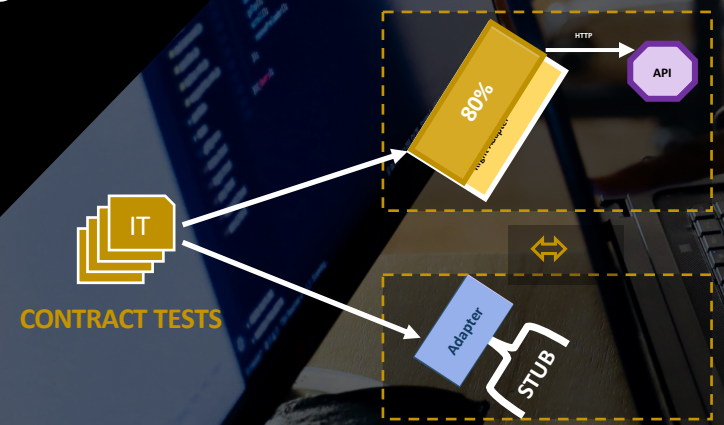
OUTSIDE-IN DIAMOND

Most of the infra code (i.e. Adapters) is also covered by our
{ acceptance | unit tests }



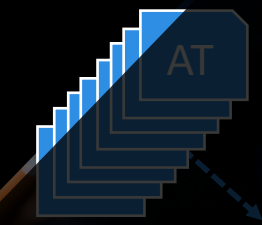
the HEXAGONAL ARCHITECTURE case

Contract tests aside to ensure that our stubs are faithful to reality



We only stub I/Os

OUTSIDE-IN DIAMOND



Most of the infra code (i.e. Adapters) is also covered by our
{ acceptance | unit tests }

(GraphQL) BookingResolver Adapter

85 %
(running all day long)

ResortContent Service Adapter

STUB

Picto Service Adapter

STUB

Resort Service Adapter

STUB

Booking Service Adapter

STUB

We only stub I/Os

Contract tests aside to ensure that our stubs are faithful to reality

15 %
(running nightly)

80%

API

IT

CONTRACT TESTS

Adapter

STUB

the HEXAGONAL ARCHITECTURE case