

ConFoo.ca 20th

AssertTrue(isDecoupled("MyTests"))



Dave Liddament

February 23-25, 2022

@daveliddament

**DECOPLED TESTS REDUCE THE
DEVELOPMENT AND MAINTENANCE
COSTS OF THE TEST SUITE.**

**VALUE OF TESTS =
COST OF BUGS FOUND BY TESTS
- COST OF TEST SUITE**

AGENDA

- ▶ Why
- ▶ Terminology



```
..... 63 / 444 ( 14%)
..... 126 / 444 ( 28%)
..... 189 / 444 ( 42%)
..... 252 / 444 ( 56%)
..... 315 / 444 ( 70%)
..... 378 / 444 ( 85%)
..... 441 / 444 ( 99%)
...
```

Time: 1.99 seconds, Memory: 24.75MB

OK (444 tests, 1201 assertions)

..... **FFFFFFFFFFFFFFFFFFF**..... 63 / 444 (14%)
..... **FFFFFFFFFFF**..... 126 / 444 (28%)
..... **FFFFFFFFFFF**..... 189 / 444 (42%)
FFFFFFFFFFF..... 252 / 444 (56%)
..... **FF** .. **FFF** .. **FFFF** .. **FFFF** .. **FFFF** .. 315 / 444 (70%)
..... **FFFFFFFFFFF**..... 378 / 444 (85%)
FFFFFFFFFFF..... **FFFFFFFFFFF**..... 441 / 444 (99%)
...

Time: 1.55 seconds, Memory: 24.75MB

There were lots of failures:

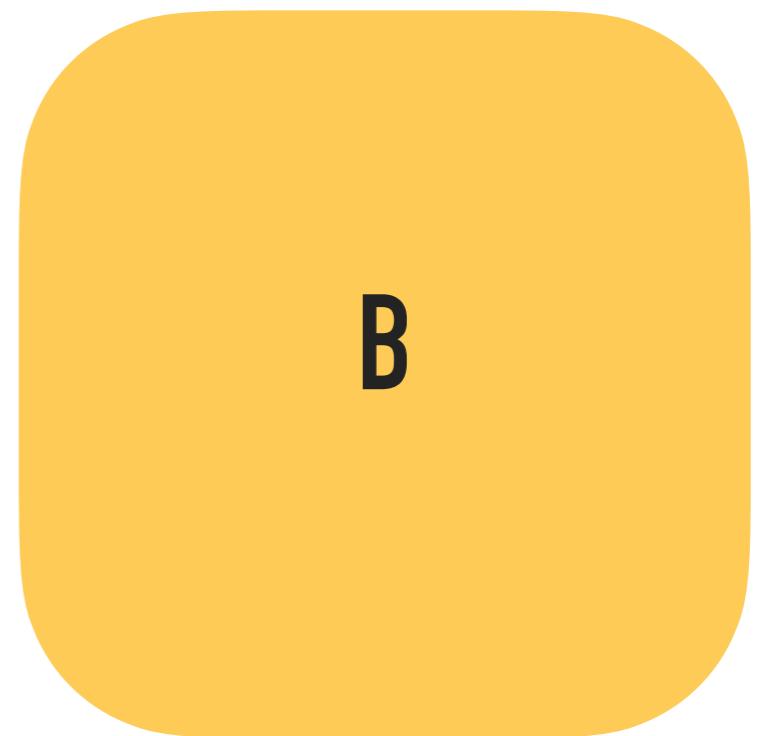
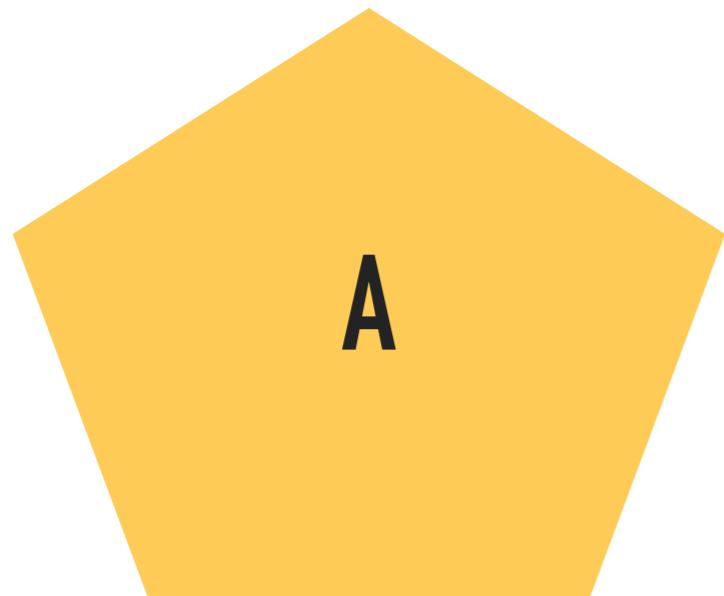




@daveliddament

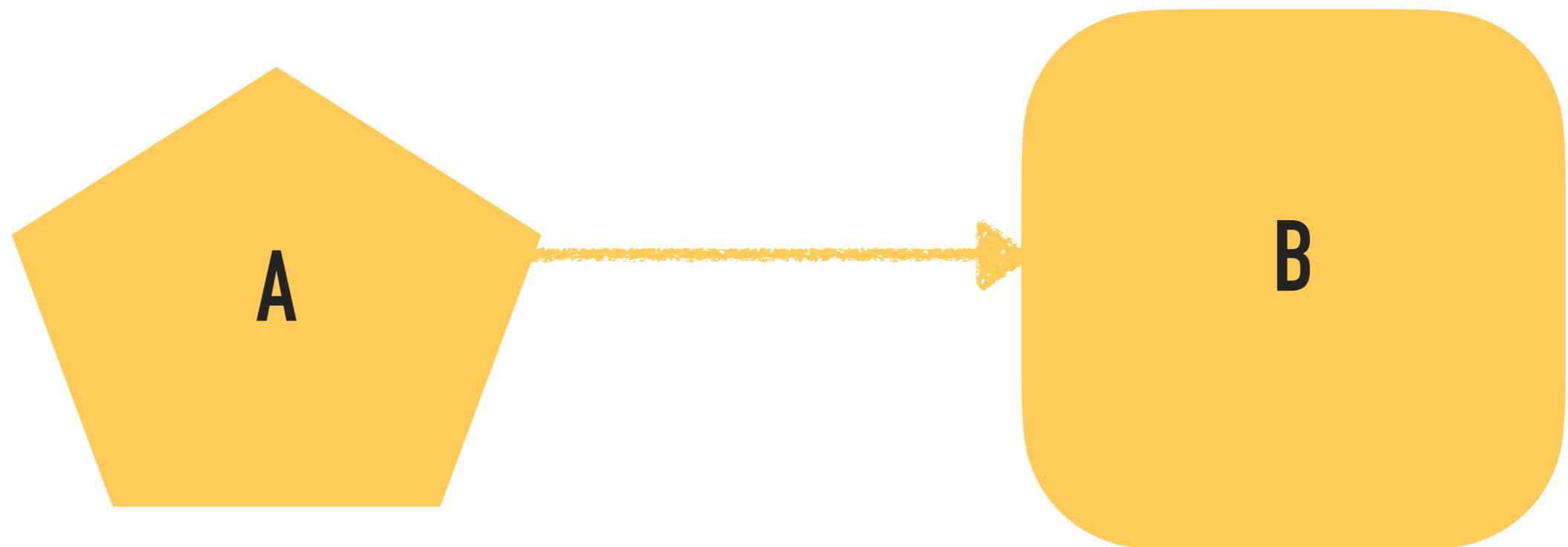
TERMINOLOGY

COUPLING



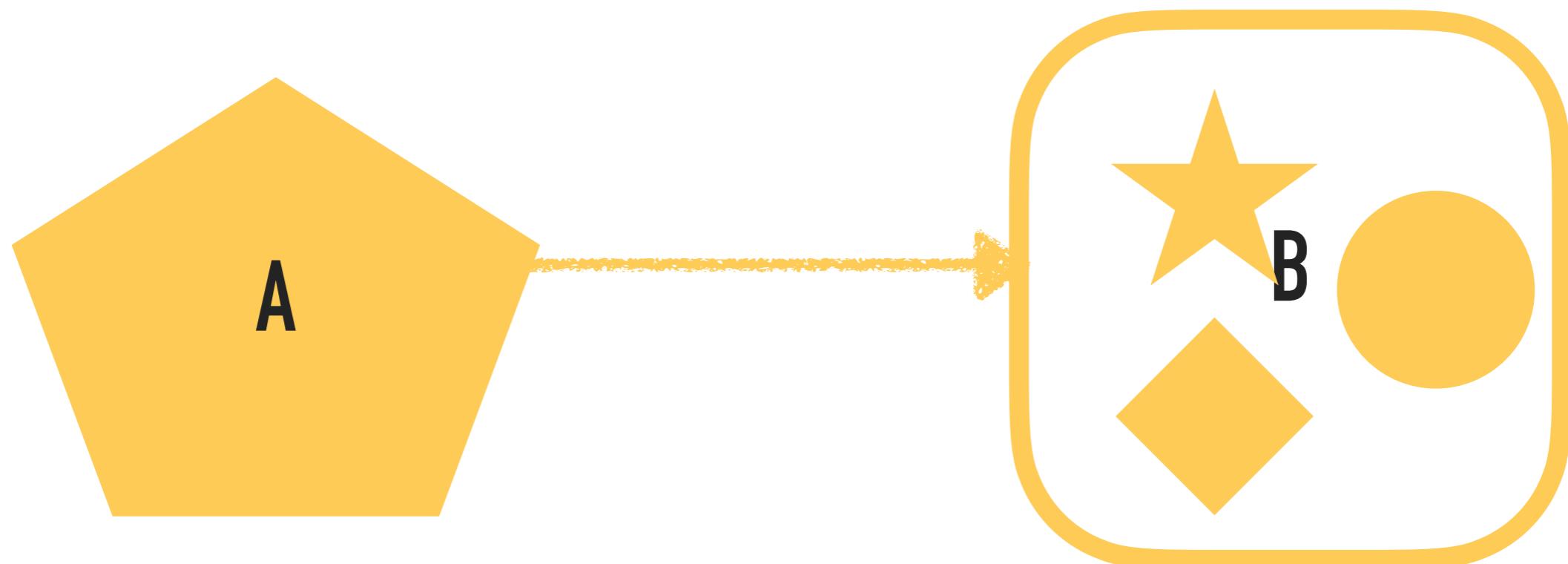
TERMINOLOGY

COUPLING

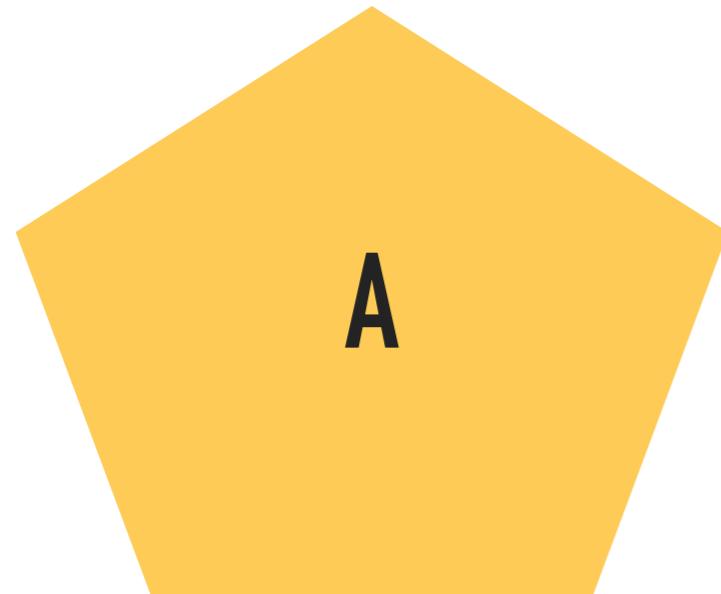


TERMINOLOGY

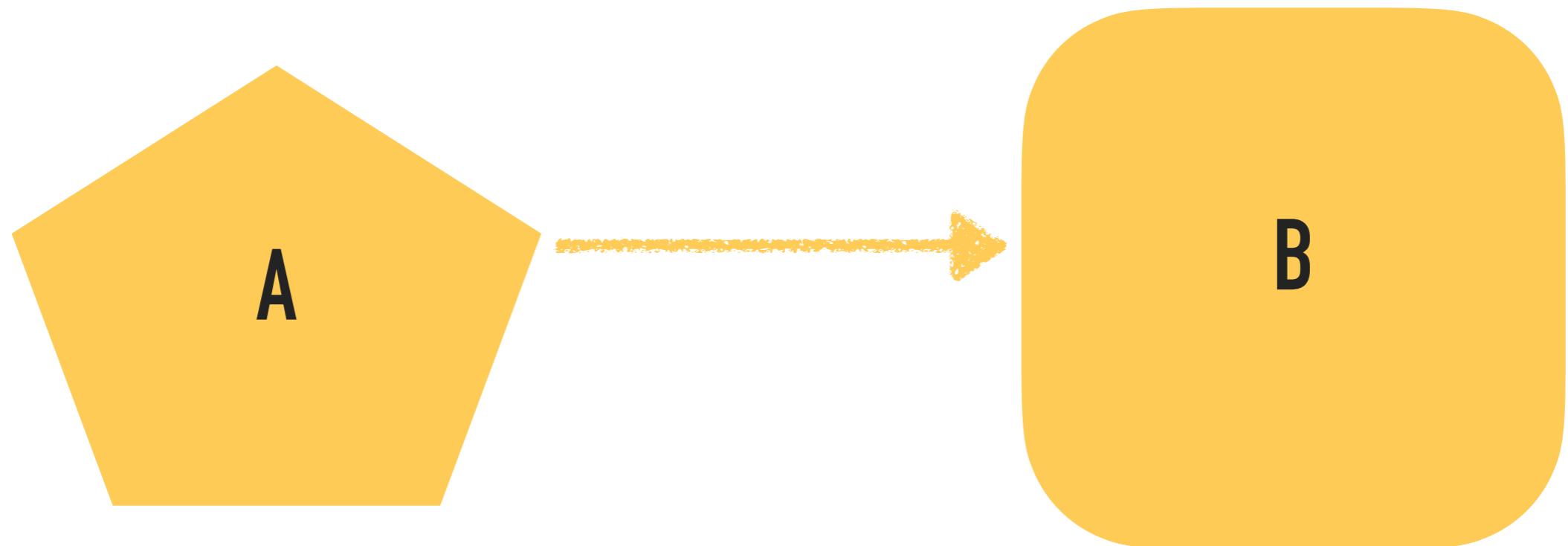
COUPLING



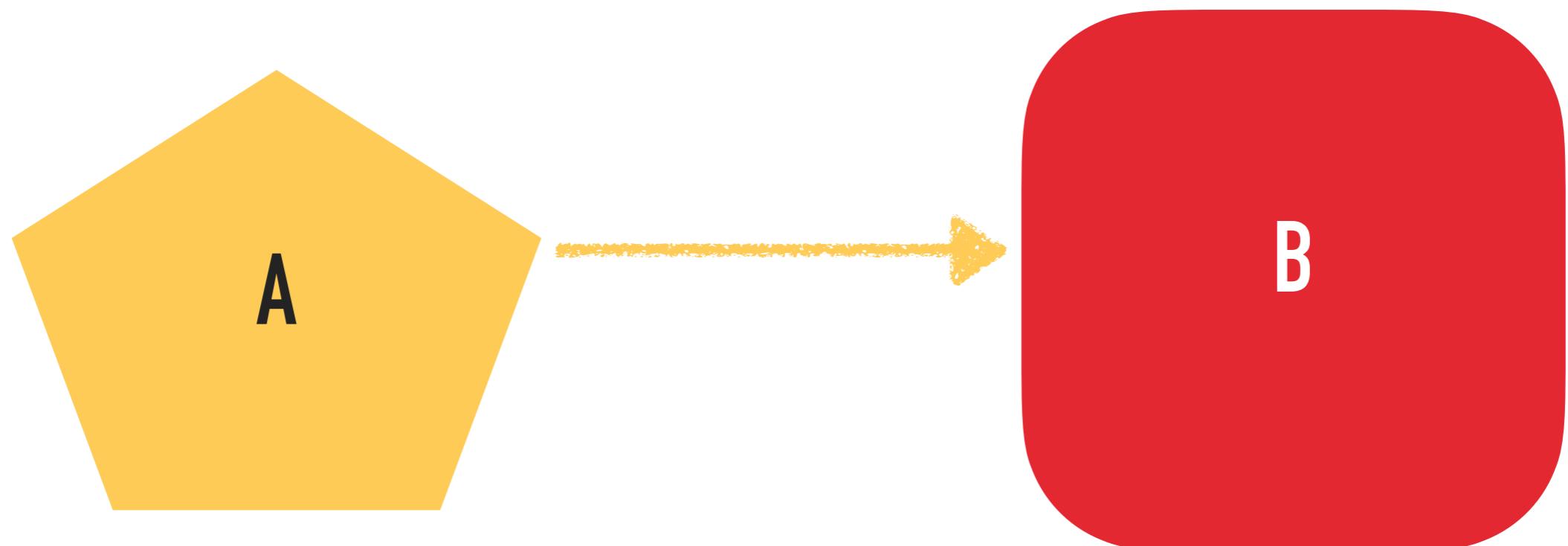
TEST DOUBLES



TEST DOUBLES



TEST DOUBLES



TEST PYRAMID



**DECOPLED TESTS REDUCE THE
DEVELOPMENT AND MAINTENANCE
COSTS OF THE TEST SUITE.**



#1

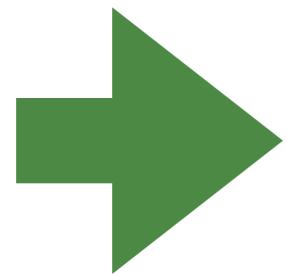
TYPICAL USER JOURNEY

- ▶ Bob would log in
- ▶ Bob see a list of quizzes
- ▶ Pick one he hadn't done
- ▶ Complete the quiz
- ▶ See his score
- ▶ His team's score would be updated

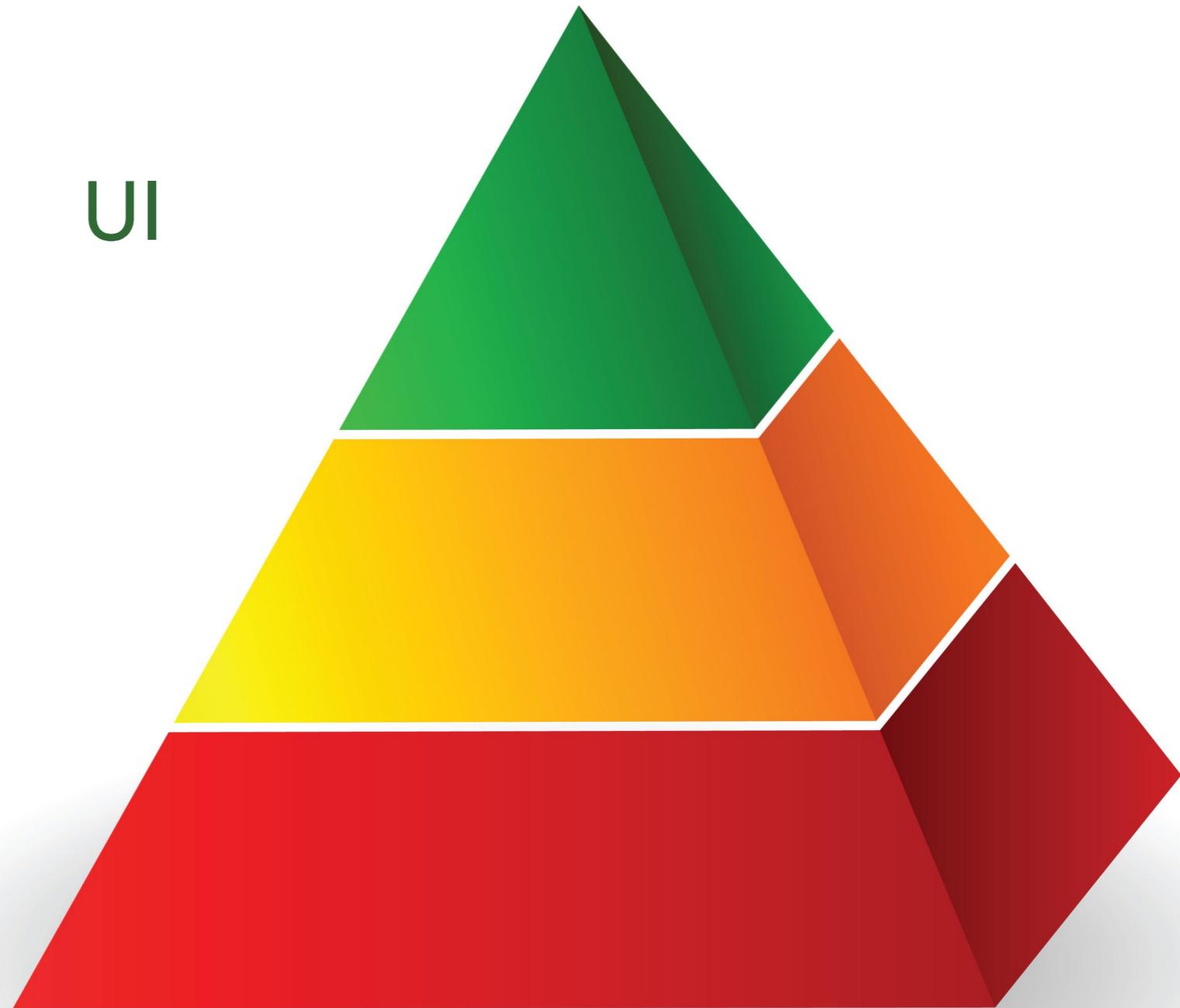
I WANT TO TEST...

Does an individual's score get
correctly allocated to their team?

STORY 1



UI



STORY 1

INITIALLY TESTS WOULD DO THIS KIND OF THING...

- ▶ Visit home page
- ▶ Find login link.
- ▶ Click login link
- ▶ Find form element with name “username”
- ▶ Enter username
- ▶ Find form element with name “password”
- ▶ Enter password
- ▶ Find button with type “submit”
- ▶ Click button
- ▶ ... etc ...

A TINY CHANGE REQUEST....

Can we change the layout of the page
showing the lists of quizzes?

..... **FFFFFFFFFFFFFFFFFFF**..... 63 / 444 (14%)
..... **FFFFFFFFFFF**..... 126 / 444 (28%)
..... **FFFFFFFFFFF**..... 189 / 444 (42%)
FFFFFFFFFFF..... 252 / 444 (56%)
..... **FF** .. **FFF** .. **FFFF** .. **FFFF** .. **FFFF** .. 315 / 444 (70%)
..... **FFFFFFFFFFF**..... 378 / 444 (85%)
FFFFFFFFFFF..... **FFFFFFFFFFF**..... 441 / 444 (99%)
...

Time: 20 minutes 54 seconds, Memory: 24.75MB

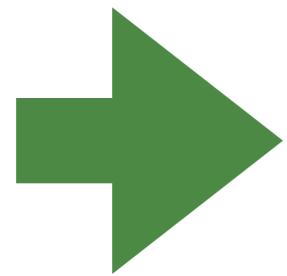
There were lots of failures:





@daveliddament

STORY 1

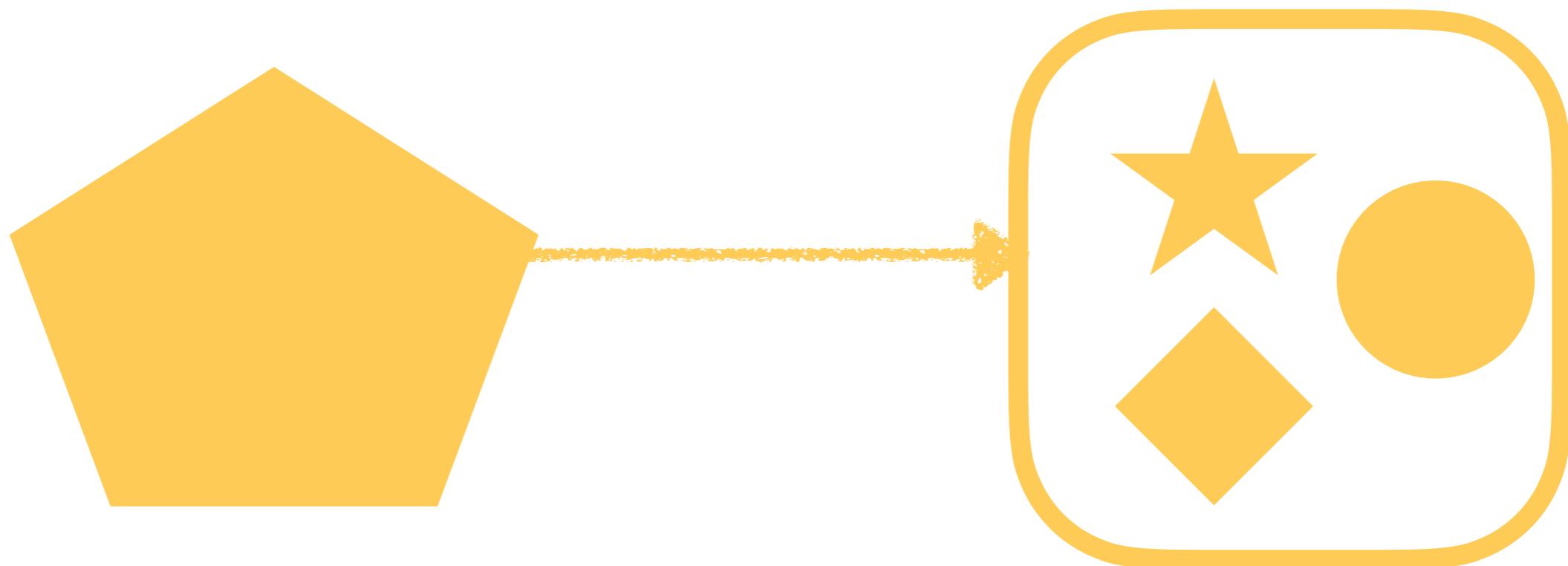


UI



STORY 1

PROBLEM: TIGHT COUPLING

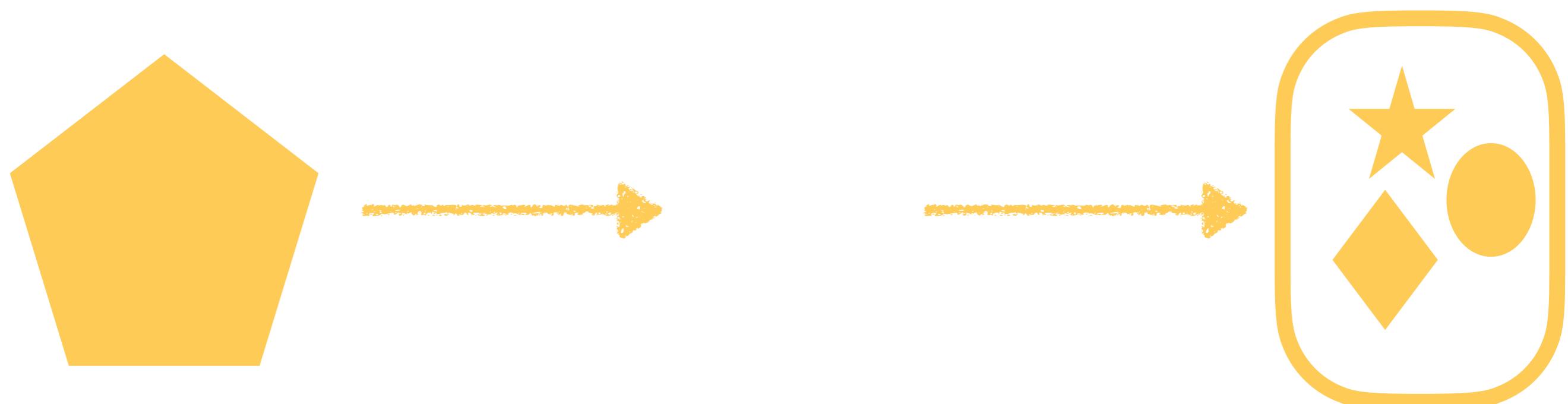


TEST

SOFTWARE
UNDER TEST

STORY 1

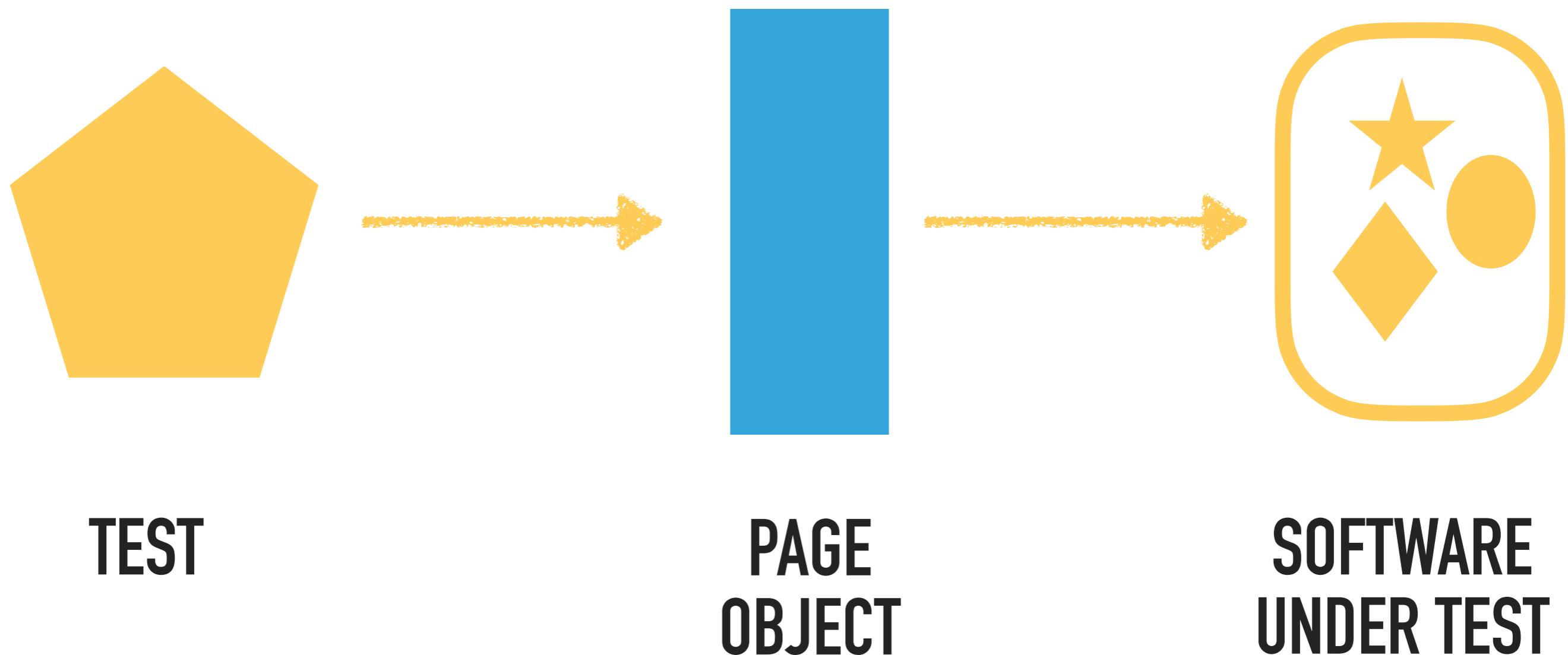
REDUCE COUPLING WITH PAGE OBJECT



TEST

SOFTWARE
UNDER TEST

REDUCE COUPLING WITH PAGE OBJECT

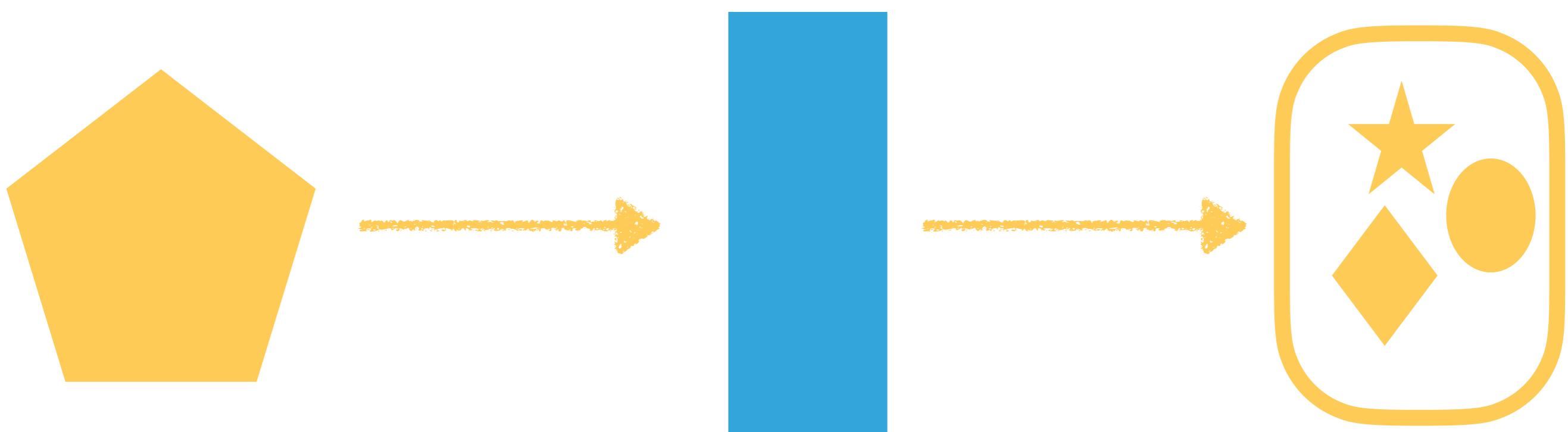


STORY 1

REDUCE COUPLING WITH PAGE OBJECT

`login ($username, $password)`

`answerQuestion ($answer)`



TEST

**PAGE
OBJECT**

**SOFTWARE
UNDER TEST**

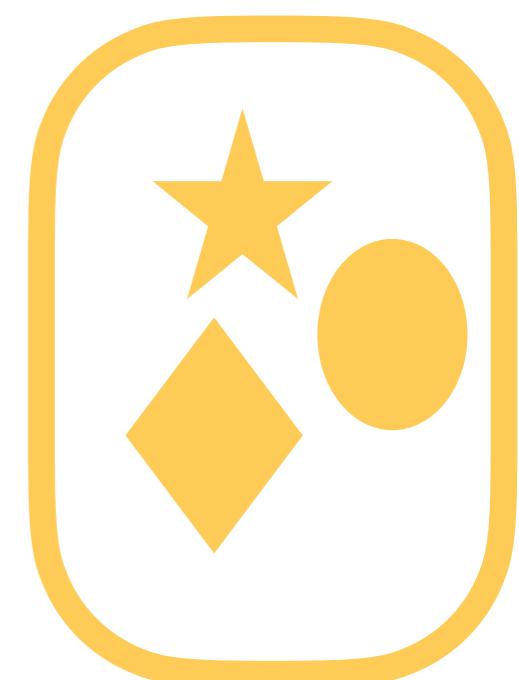
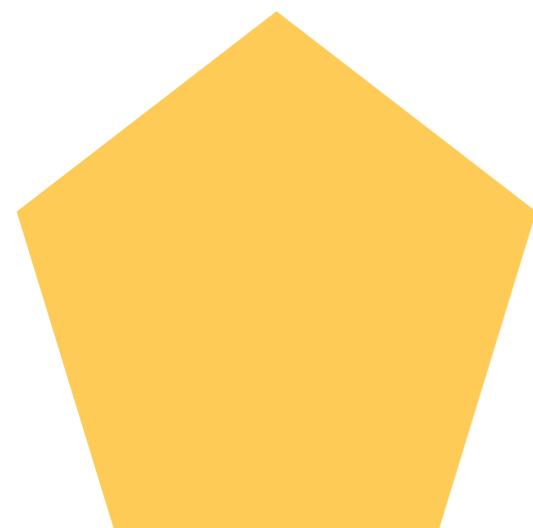
REDUCE COUPLING WITH PAGE OBJECT

login (\$username, \$password)

answerQuestion (\$answer)

findElementByName (\$name)

click ()



TEST

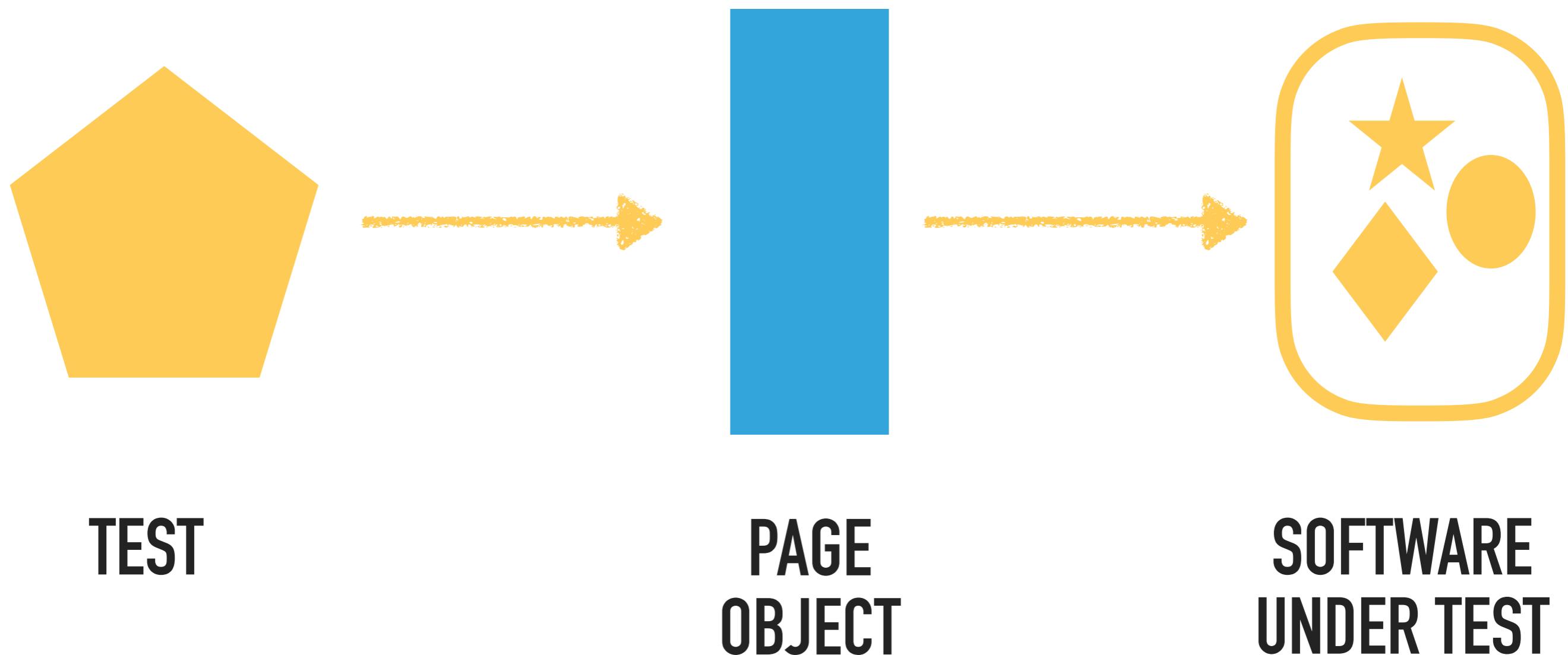
PAGE
OBJECT

SOFTWARE
UNDER TEST

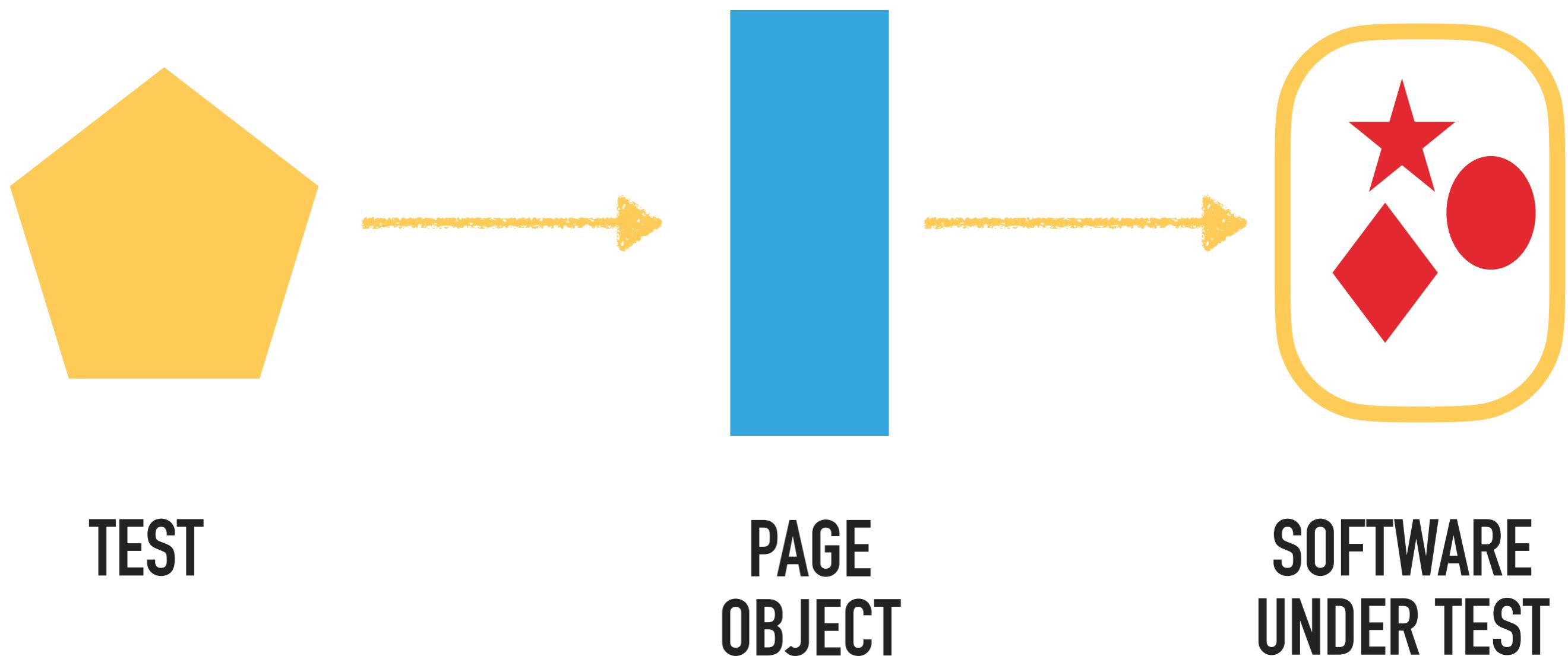
A PAGE OBJECT CAN...

- ▶ Simulate an action a human would do.
- ▶ Grab data from the page.
- ▶ Navigate to another page.

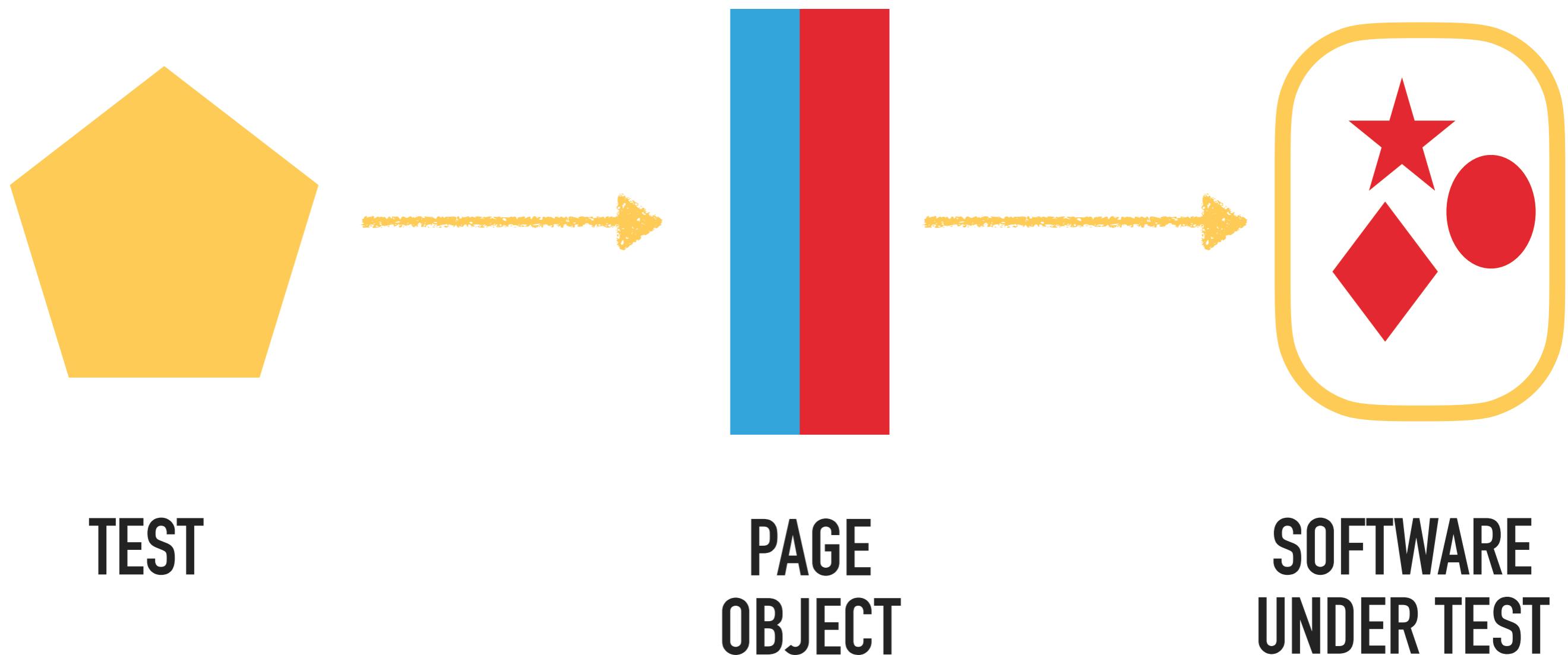
REDUCE COUPLING WITH PAGE OBJECT



REDUCE COUPLING WITH PAGE OBJECT



REDUCE COUPLING WITH PAGE OBJECT

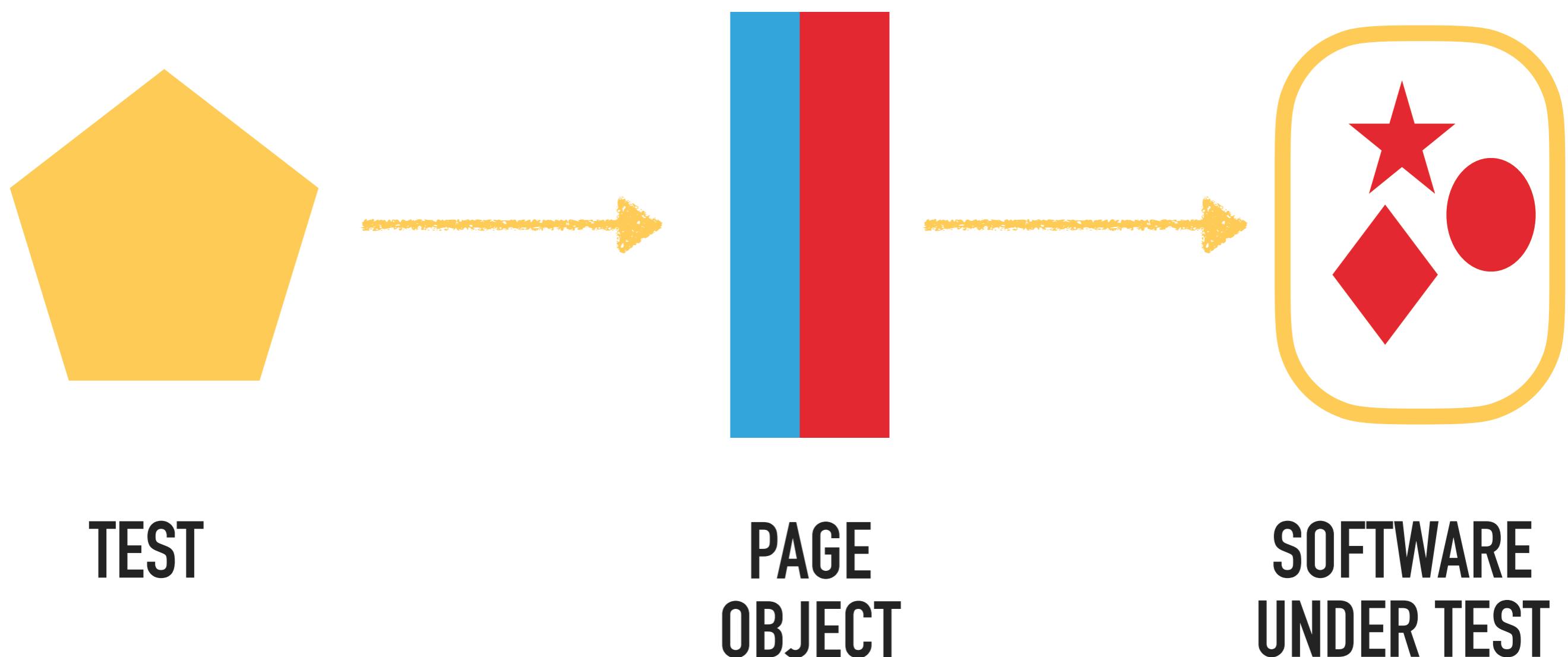


STORY 1

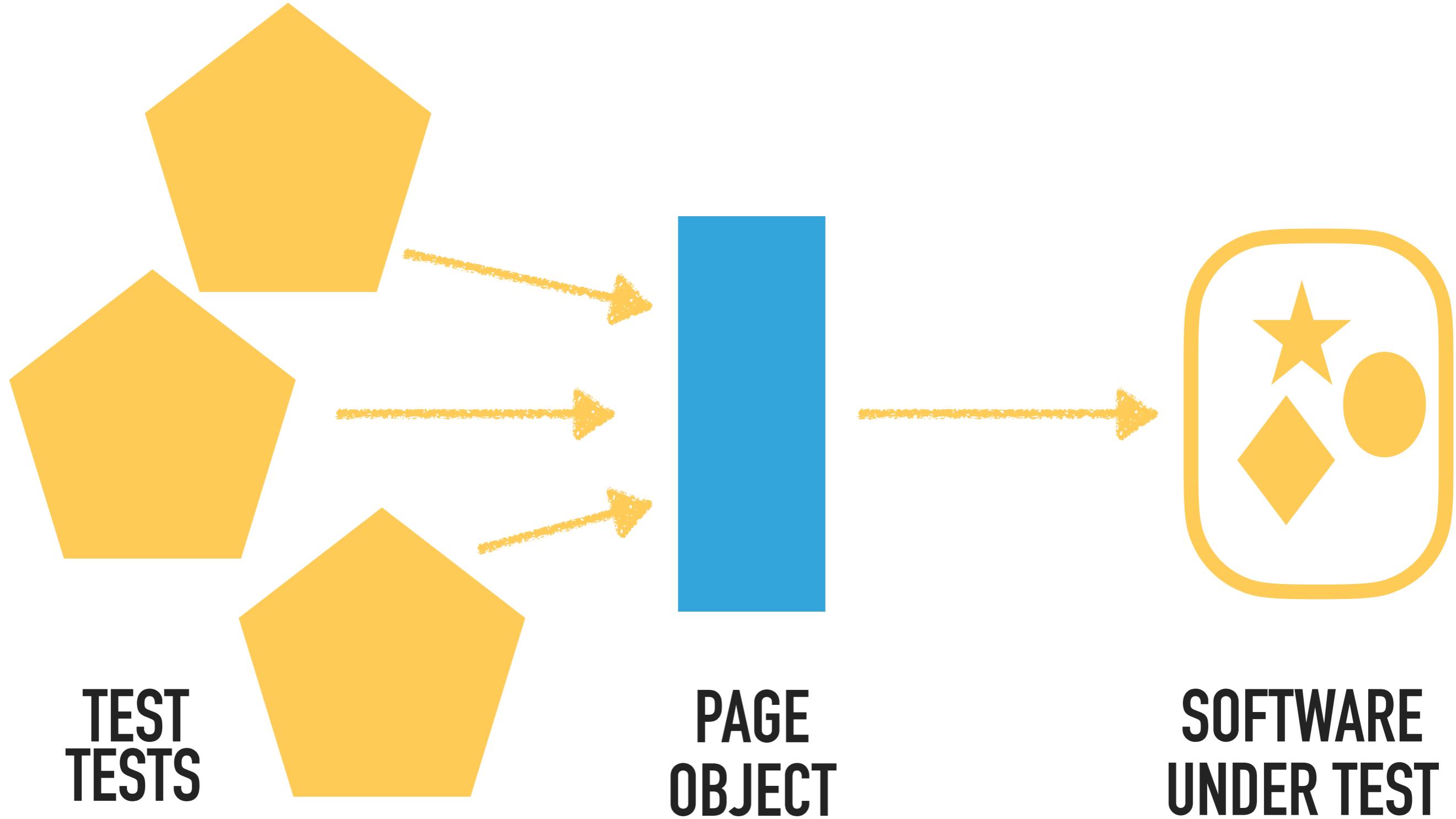
REDUCE COUPLING WITH PAGE OBJECT

`login ($username, $password)`

`answerQuestion ($answer)`



REDUCE COUPLING WITH PAGE OBJECT



TEST LOOK A BIT MORE LIKE THIS

```
$loginPage = $homePage->getLoginPage();
```

```
$myQuizzesPage = $loginPage->login("bob", "password");
```

```
$quiz1Page = $myQuizesPage->findQuiz(1);
```

```
$quiz1Page->setAnswer1('a');
```

```
$quiz1Page->setAnswer2('b');
```

```
$resultsPage = $quiz1Page->submitAnswers();
```

```
assertEquals(3, $resultsPage->getScore());
```

... etc ...

A TINY CHANGE REQUEST....

Could we change the page a user goes to after logging in?

THE TESTS WILL BREAK

```
$LoginPage = $HomePageObject->getLoginPageObject();  
$myQuizzesPage = $LoginPage->login("bob", "password");  
$quiz1Page = $myQuizesPage->findQuiz(1);  
$quiz1Page->setAnswer1('a');  
$quiz1Page->setAnswer2('b');  
$resultsPage = $quiz1Page->submitAnswers();  
assertEquals(3, $resultsPage->getScore());  
... etc ...
```

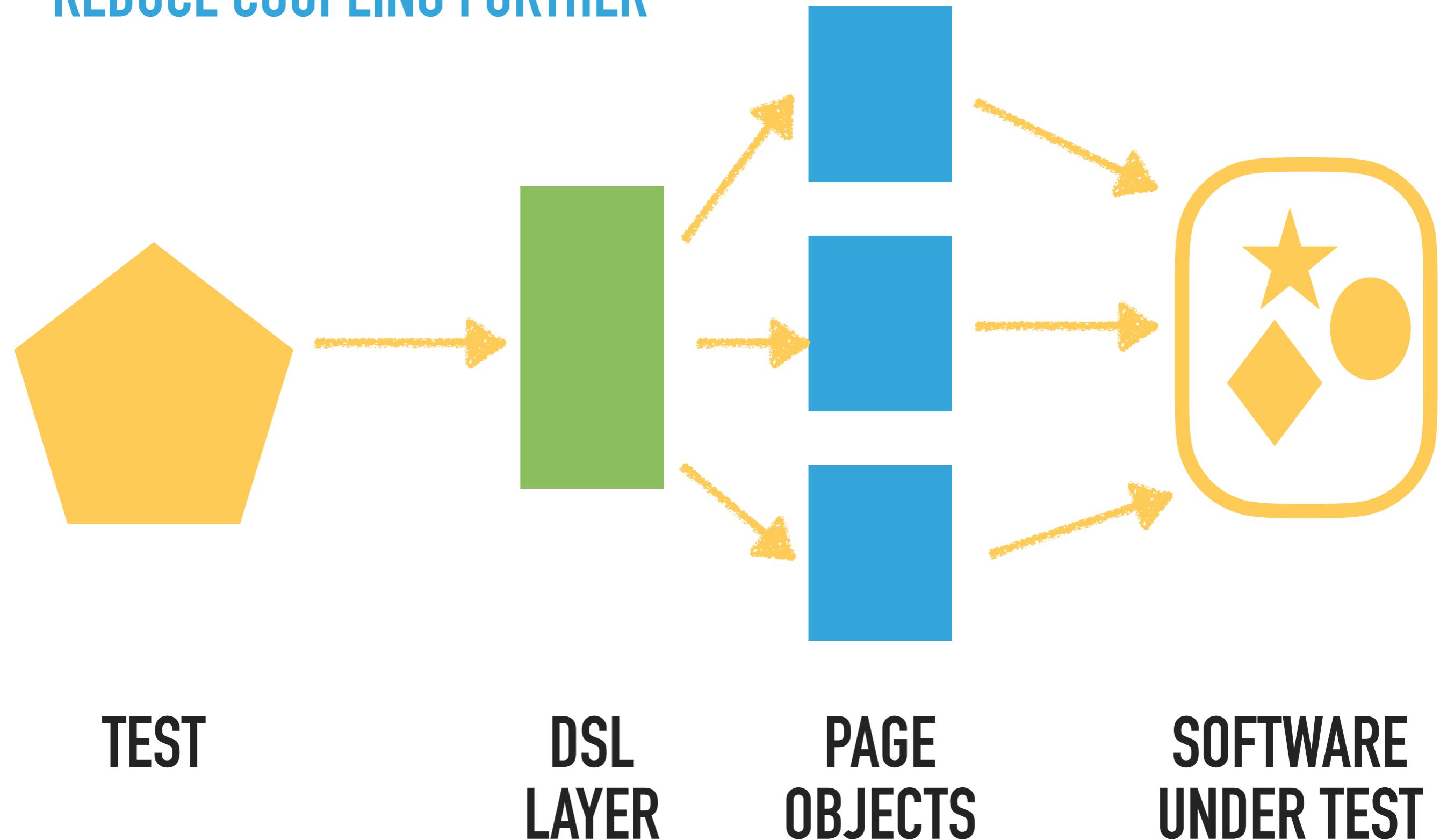
..... **FFFFFFFFFFFFFFFFFFF**..... 63 / 444 (14%)
..... **FFFFFFFFFFF**..... 126 / 444 (28%)
..... **FFFFFFFFFFF**..... 189 / 444 (42%)
FFFFFFFFFFF..... 252 / 444 (56%)
..... **FF** .. **FFF** .. **FFFF** .. **FFFF** .. **FFFF** .. 315 / 444 (70%)
..... **FFFFFFFFFFF**..... 378 / 444 (85%)
FFFFFFFFFFF..... **FFFFFFFFFFF**..... 441 / 444 (99%)
...

Time: 20 minutes 54 seconds, Memory: 24.75MB

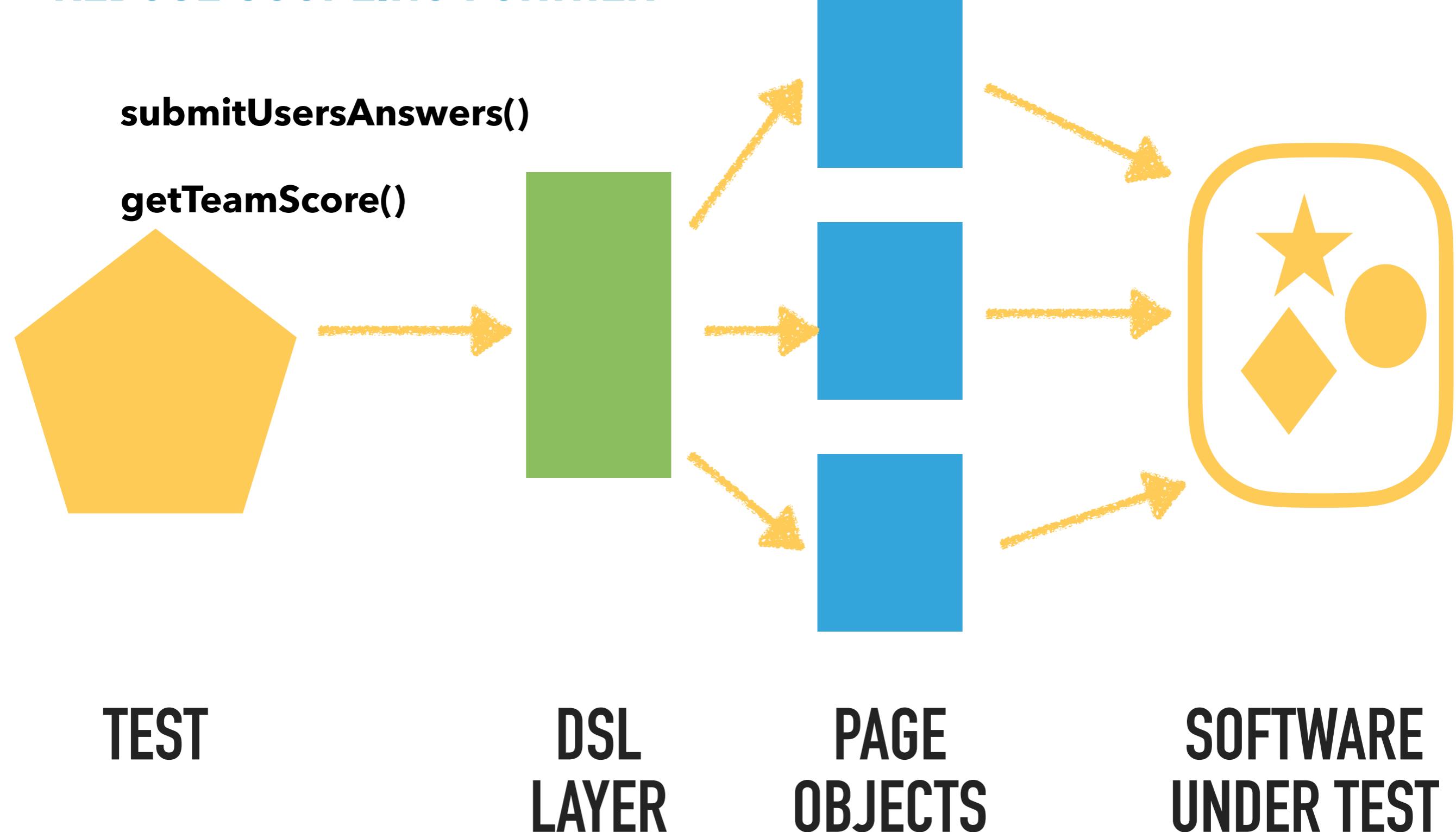
There were lots of failures:



REDUCE COUPLING FURTHER



REDUCE COUPLING FURTHER



STORY 1

TEST LOOK A BIT MORE LIKE THIS

```
assignUserToTeam($bob, $teamApple);

submitUsersAnswers($bob, self::QUIZ_1,
['engagement' => 'a', 'enjoyment' => 'b', ... etc ... ]);

$score = getTeamScore($apple);

assertEquals(7, $score);
```

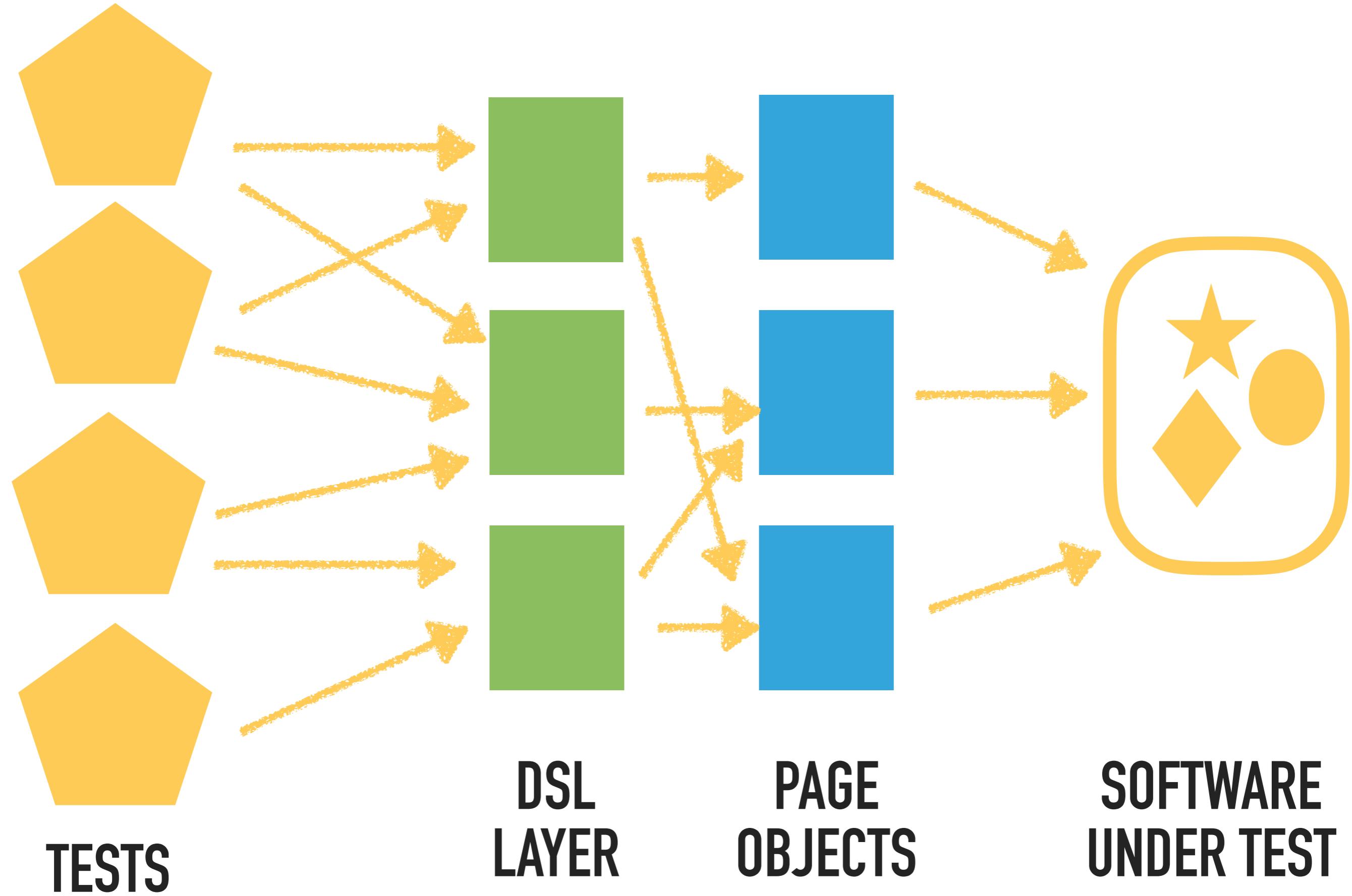
TEST LOOK A BIT MORE LIKE THIS

```
assignUserToTeam($bob, $teamApple);  
  
submitUsersAnswers($bob, self::QUIZ_1,  
['engagement' => 'a', 'enjoyment' => 'b', ... etc ...]);
```

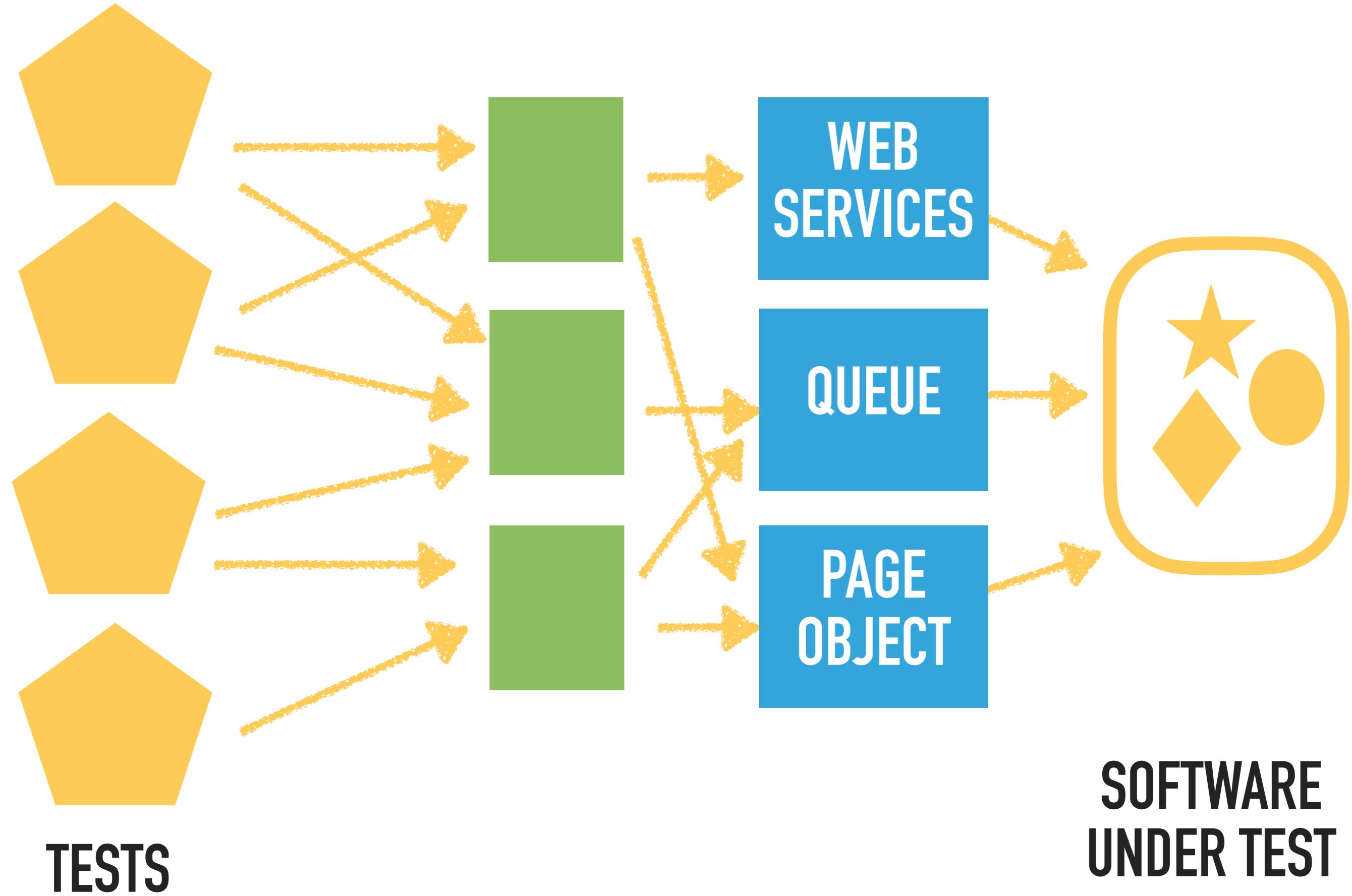
```
$score = getTeamScore($apple);  
  
assertEquals(7, $score);
```

Does an individual's score get correctly allocated to their team?

STORY 1



STORY 1



THE MORAL OF STORY 1...

- ▶ Testing an application's business logic via the UI layer is difficult, time consuming and requires a lot of effort.
- ▶ Introduce layers between the tests and the SUT to:
 - ▶ Reduce coupling
 - ▶ Isolate changes to updates in these layers
 - ▶ Tests don't change unless the functionality of the SUT changes.
- ▶ I don't like doing this kind of testing!

**DECOPLED TESTS REDUCE THE
DEVELOPMENT AND MAINTENANCE
COSTS OF THE TEST SUITE.**

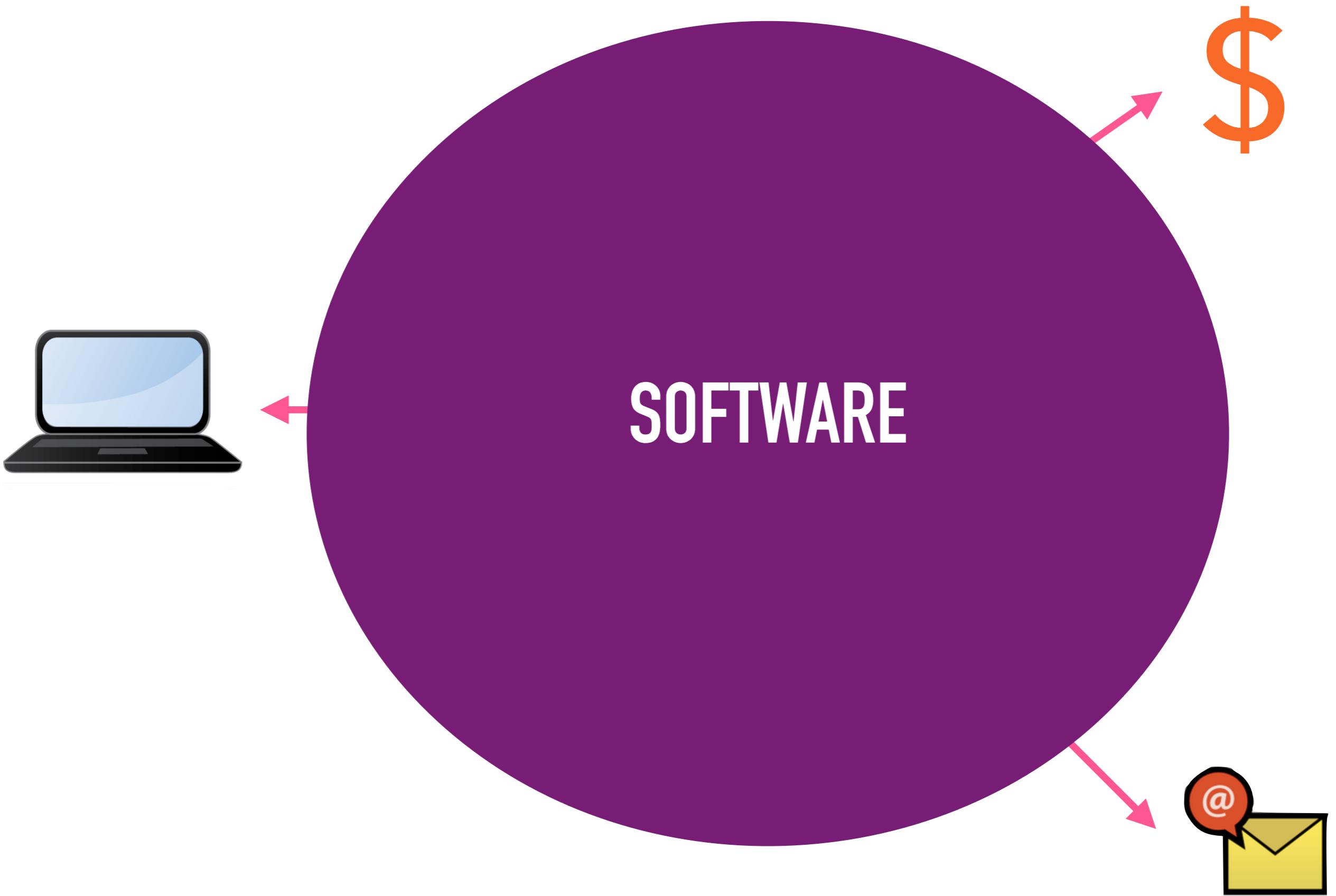
BUT WHAT IF ...

We replace the entire website with
an app?



#2

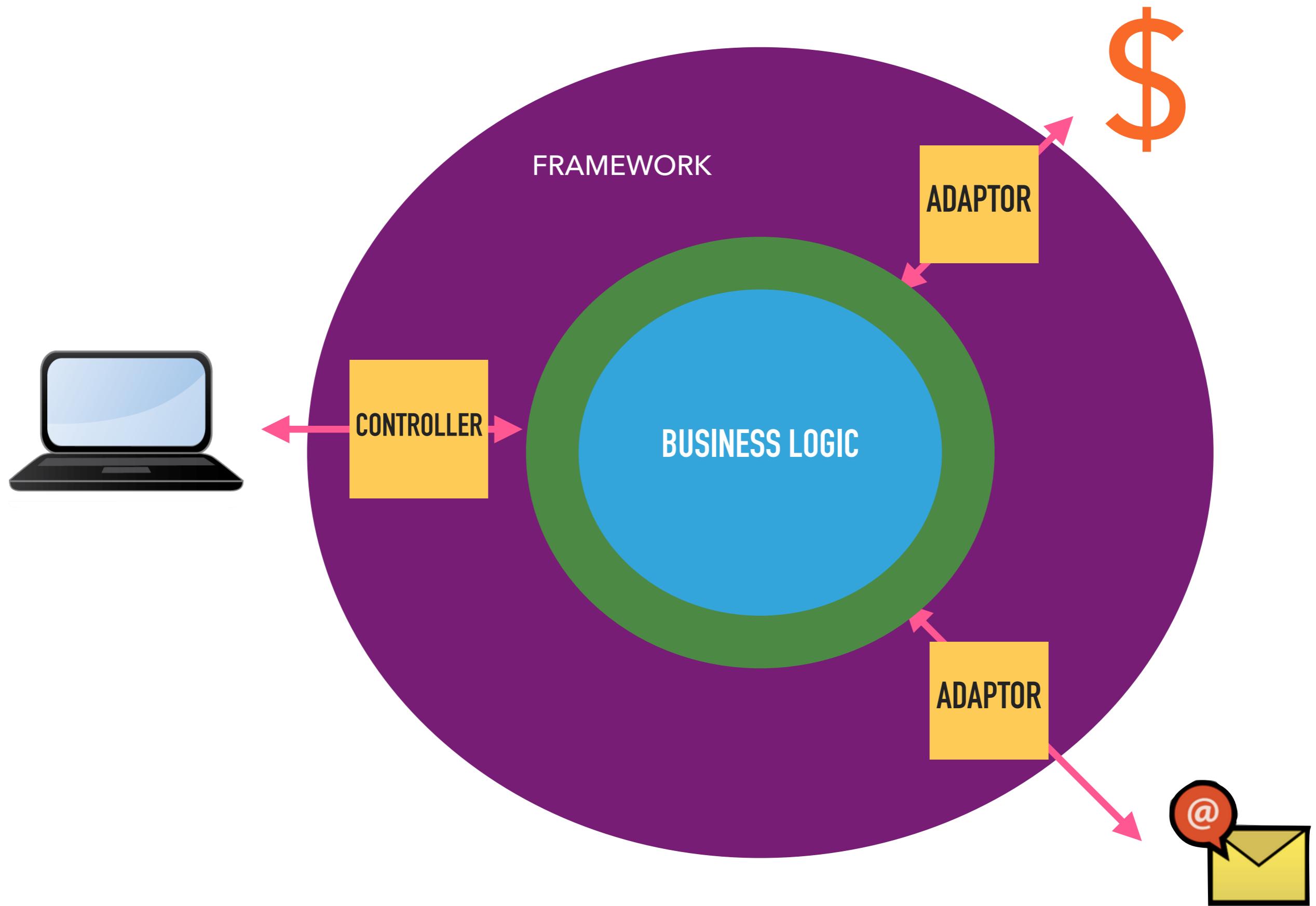
STORY 2



THERE MUST BE A BETTER WAY...

- ▶ Layered architecture
- ▶ Hexagonal architecture

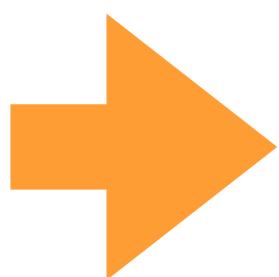
STORY 2



SERVICE LAYER

```
interface AnswerSubmissionService
{
    public function submitUsersAnswers (
        User $user,
        int $quizId,
        array $answers
    ) : void;
}
```

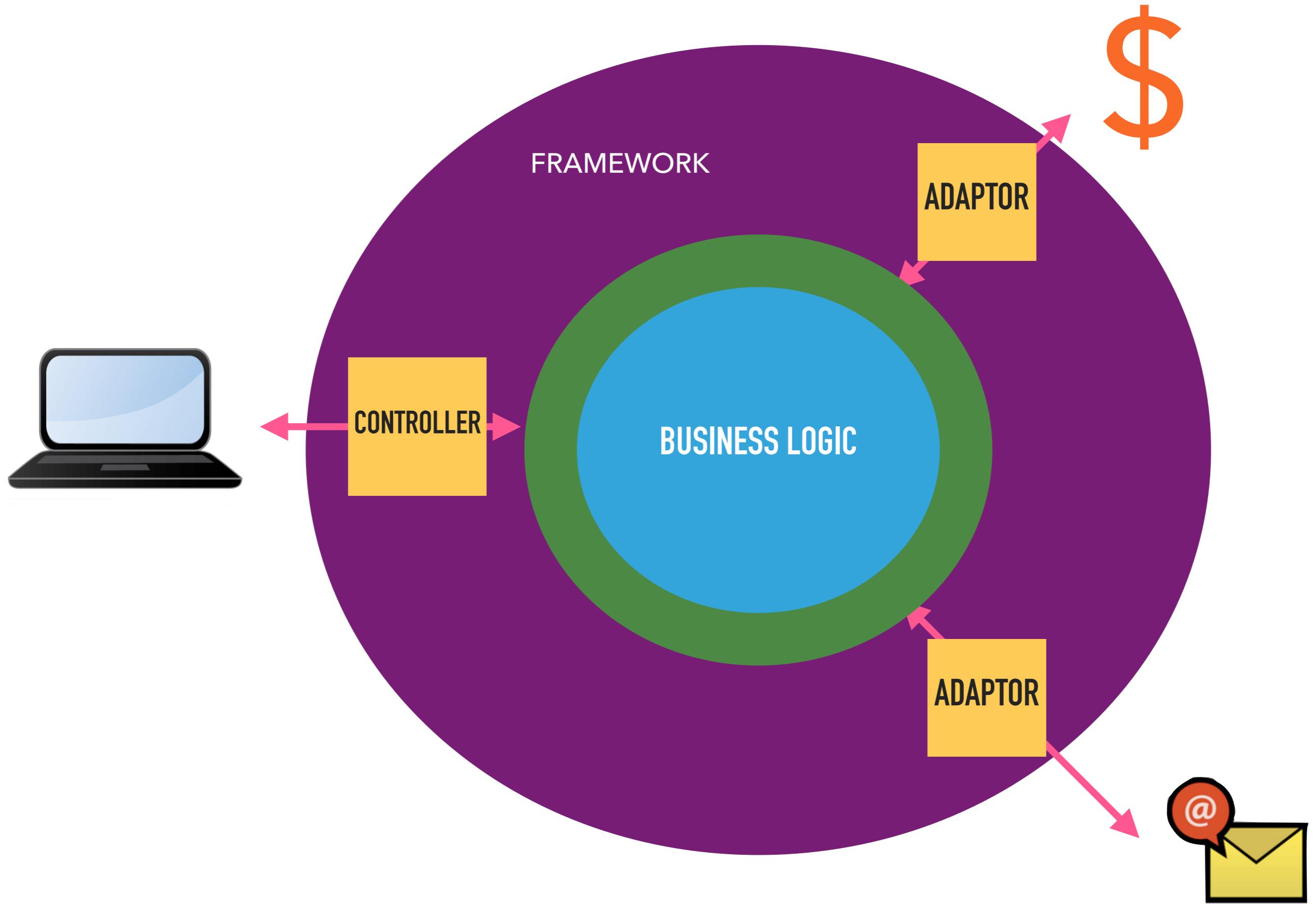
STORY 2



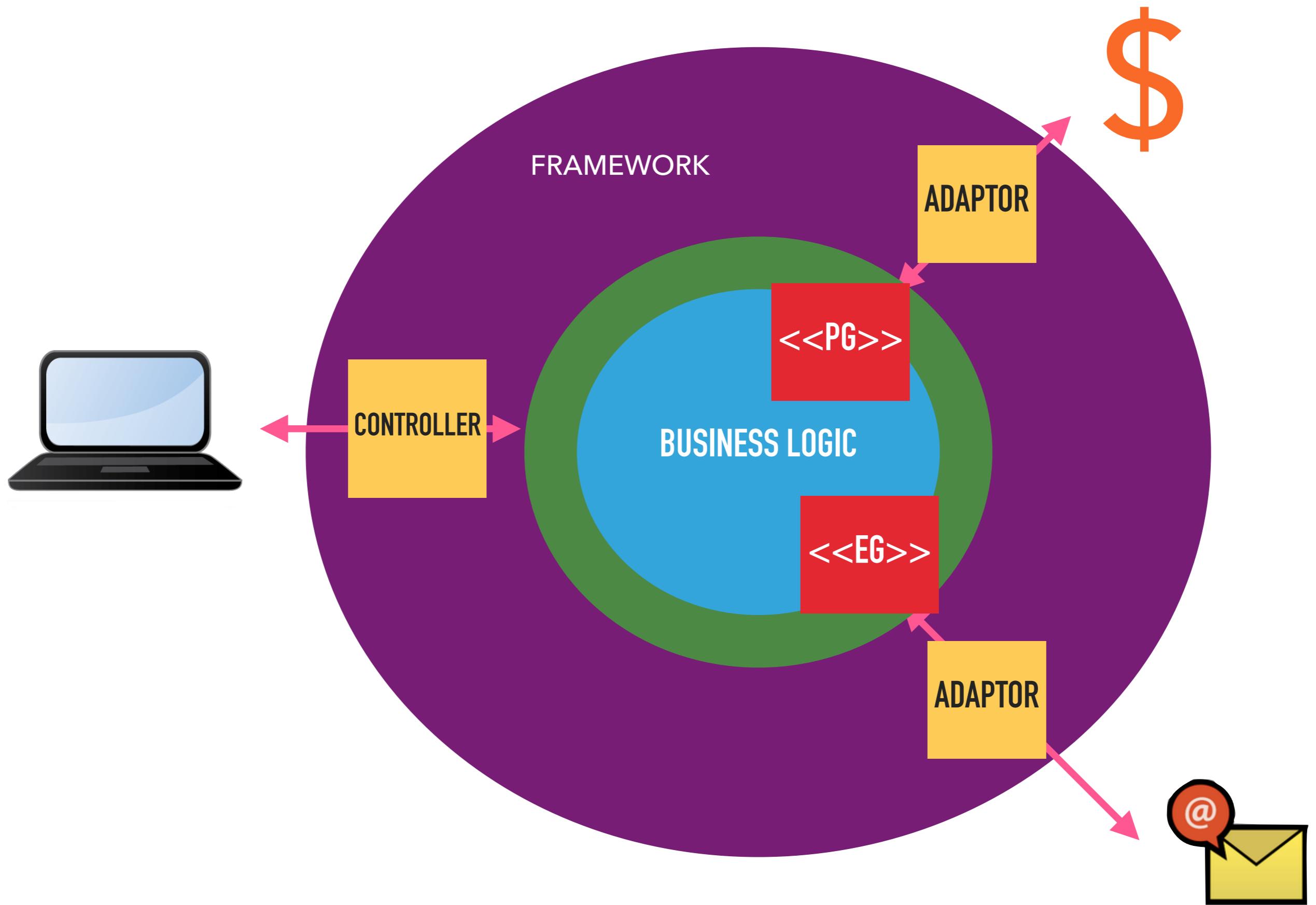
Integration



STORY 2



STORY 2



STORY 2

EMAIL GATEWAY

EMAIL GATEWAY

```
interface EmailGateway
```

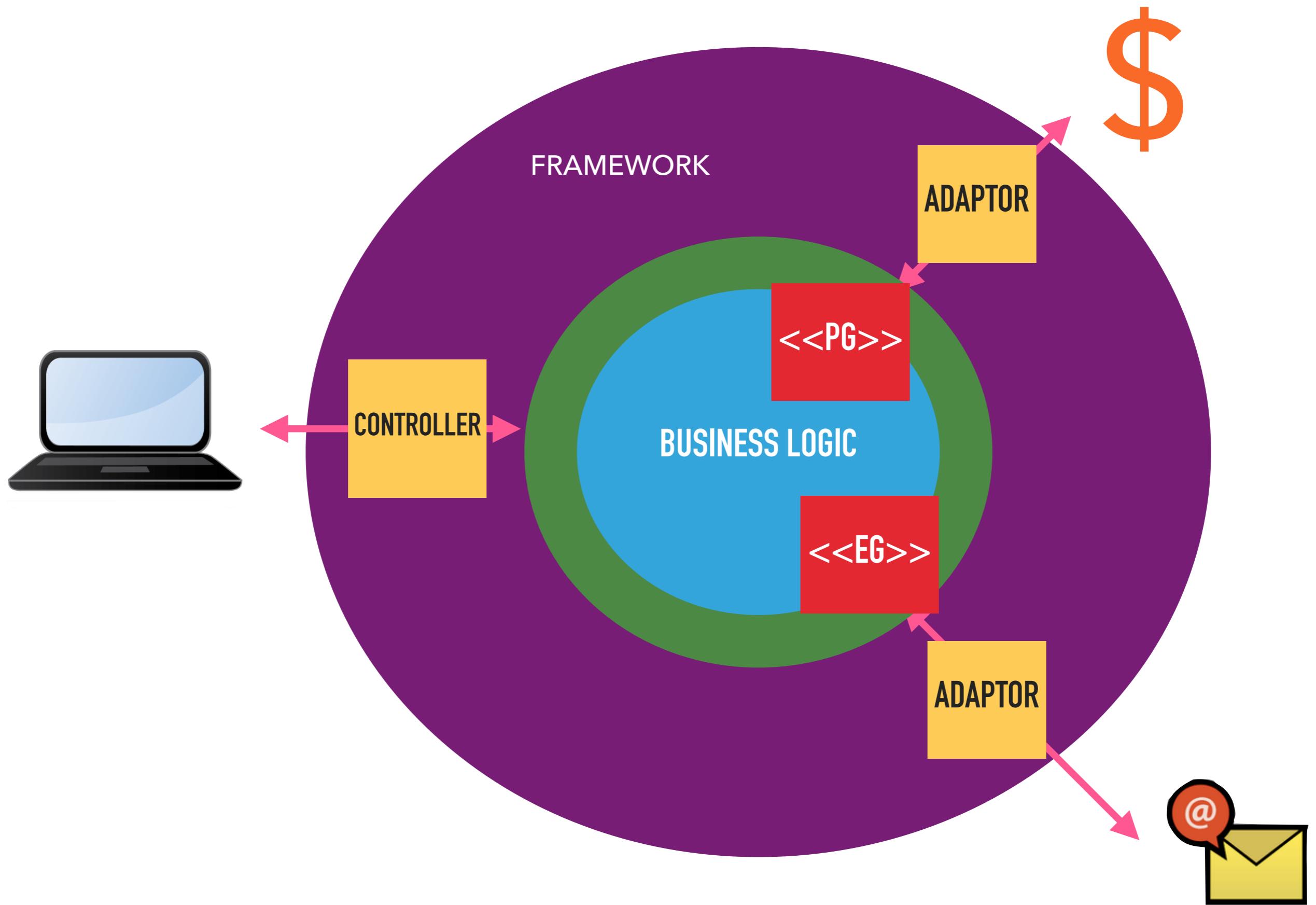
```
{
```

```
    /**
     * Sends an email
     */
```

```
    public function sendEmail (
        $to,
        $from,
        $subject,
        $message
    ) : void;
```

```
}
```

STORY 2



EMAIL GATEWAY TEST IMPLEMENTATION

EmailGatewaySpy implements EmailGateway

{

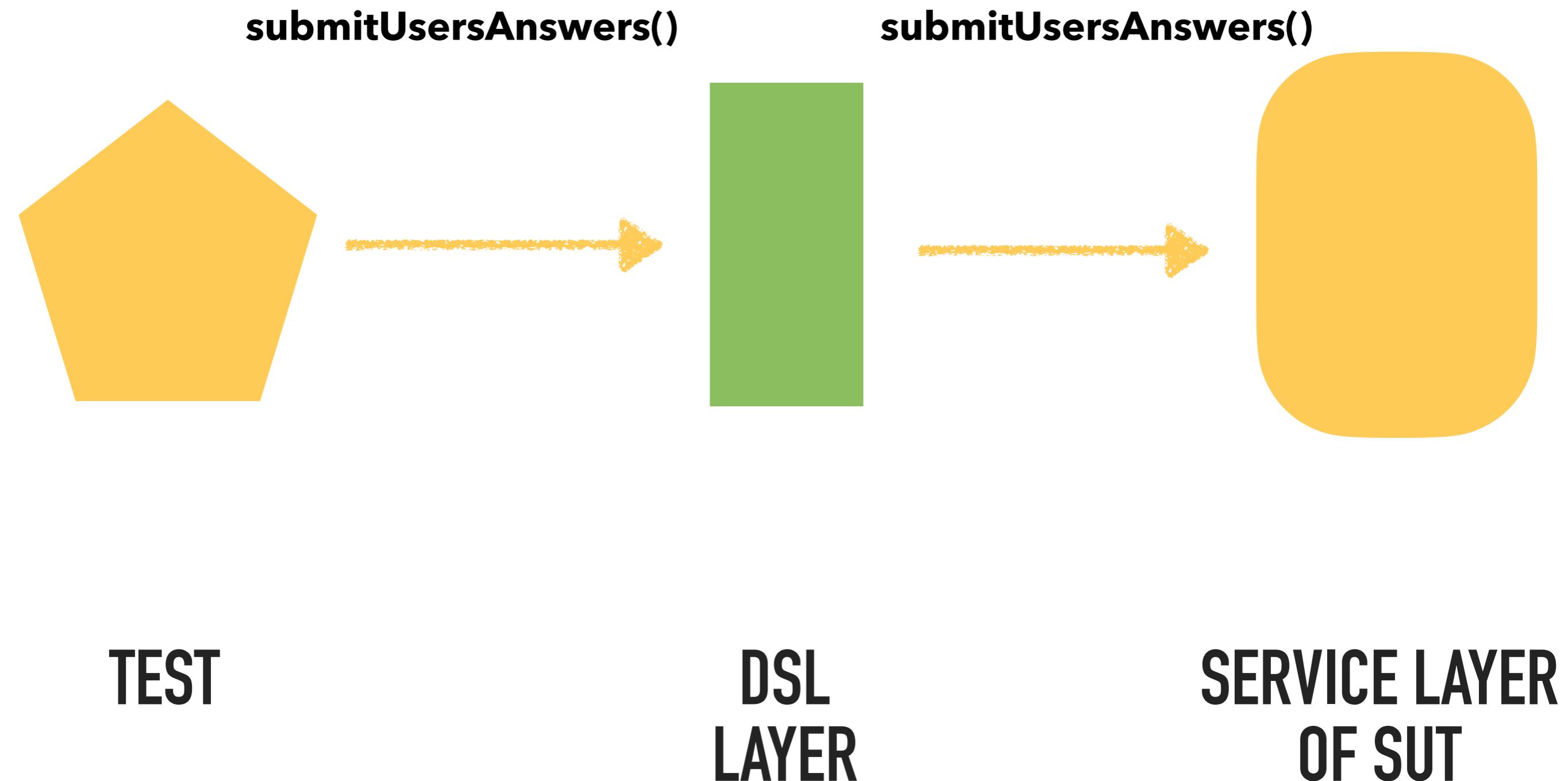
```
public function sendEmail(... parameters ...) {  
    // Store email in array;  
}
```

```
public function getEmails(): array {  
    return array of emails  
}
```

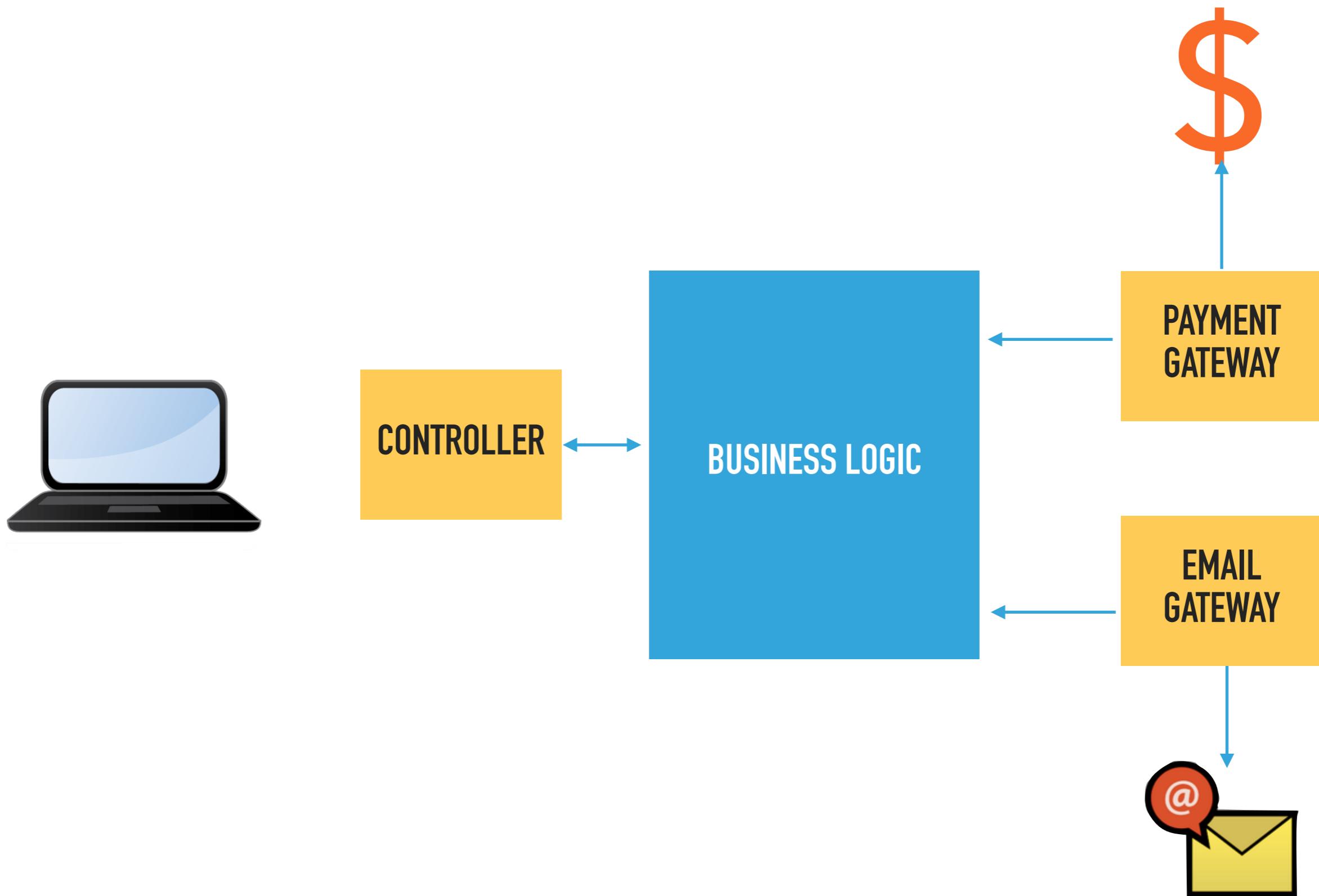
}

STORY 2

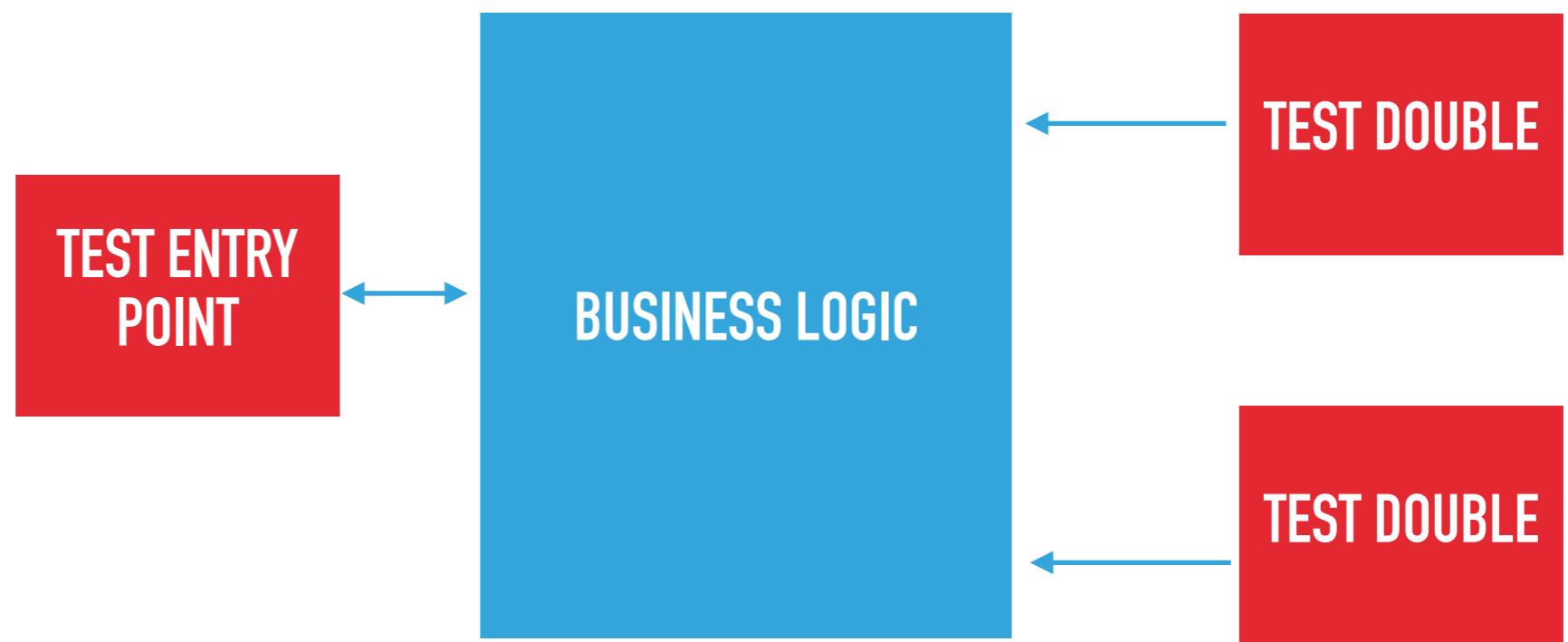
TESTING IS EASIER



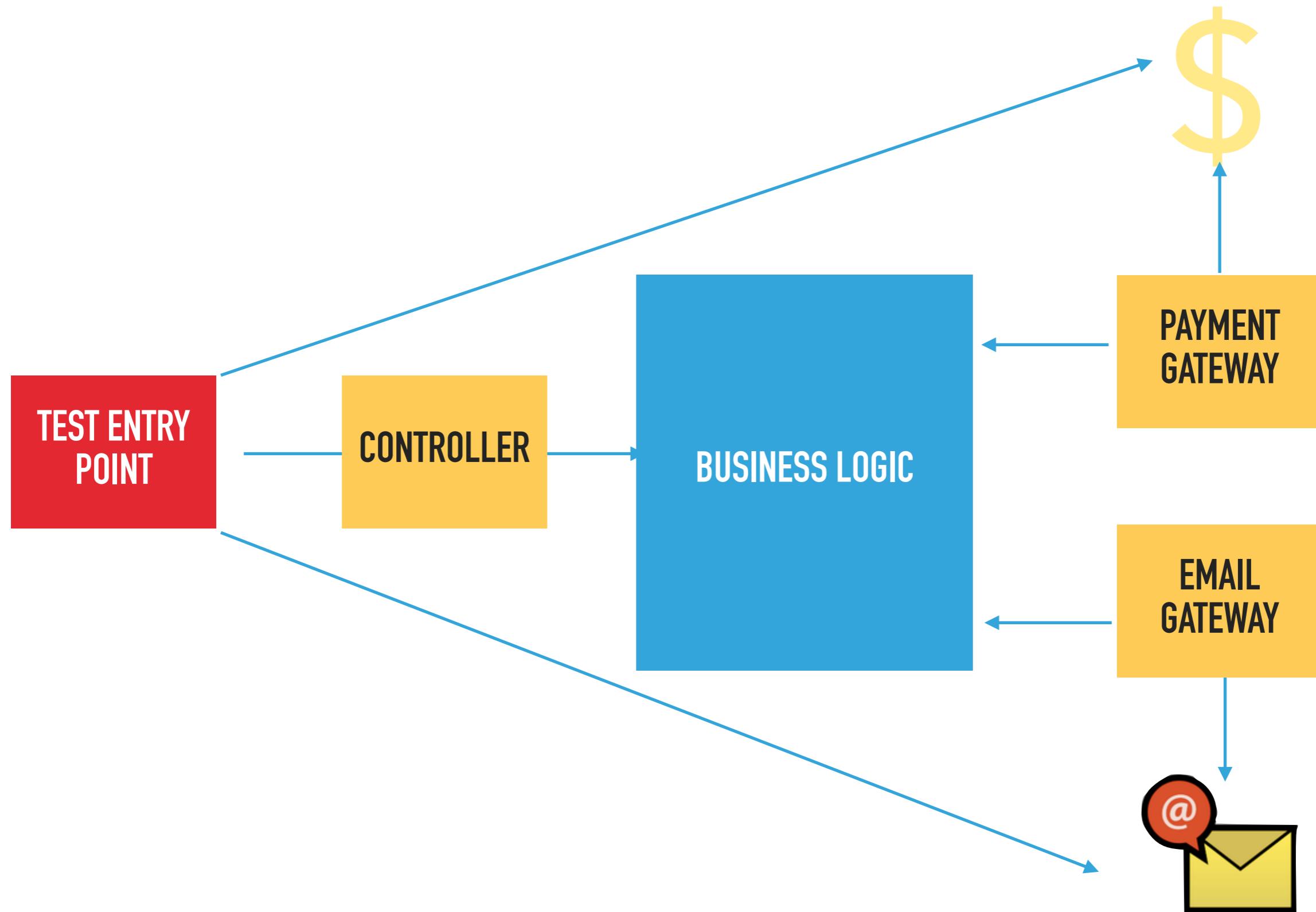
STORY 2



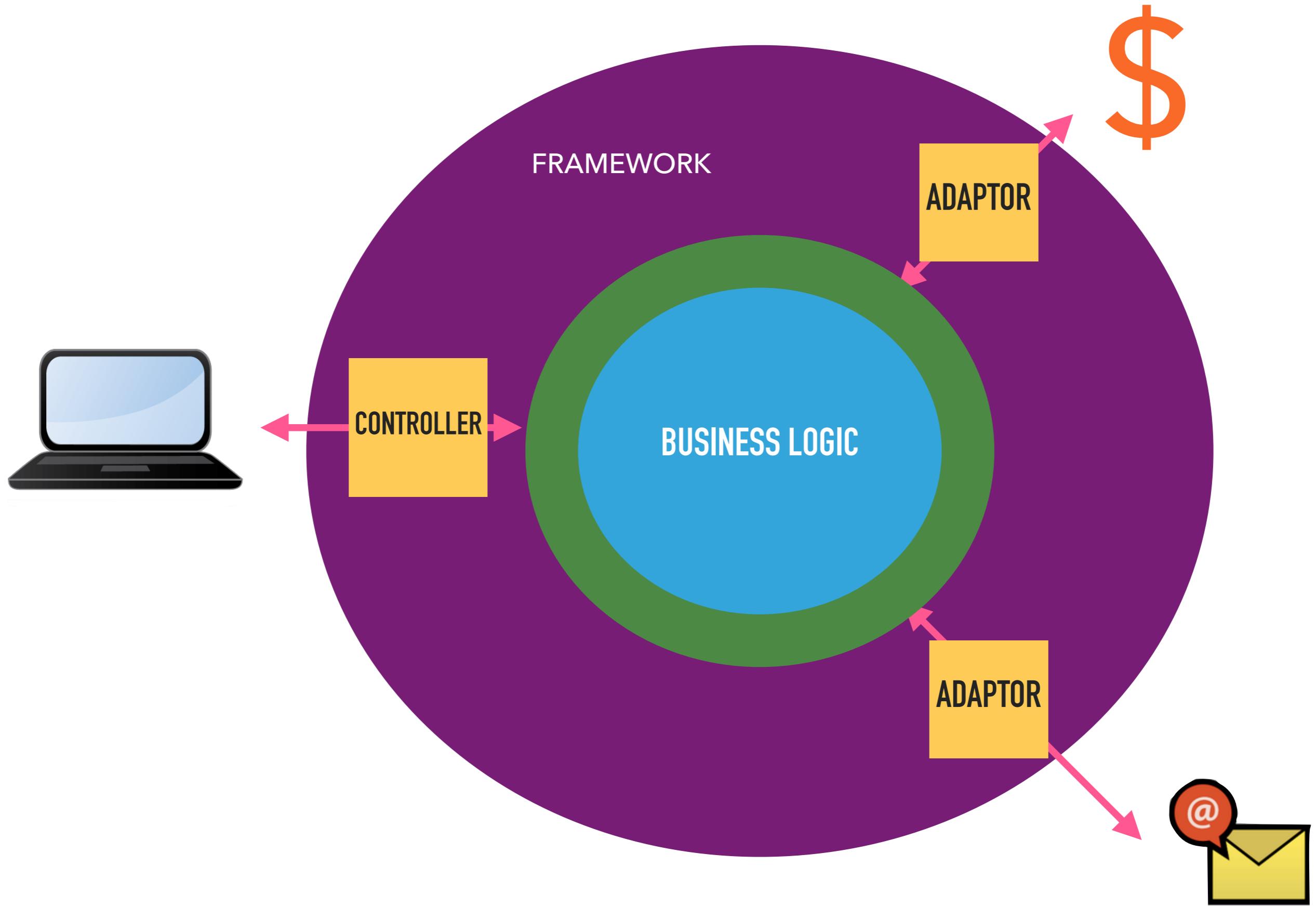
STORY 2



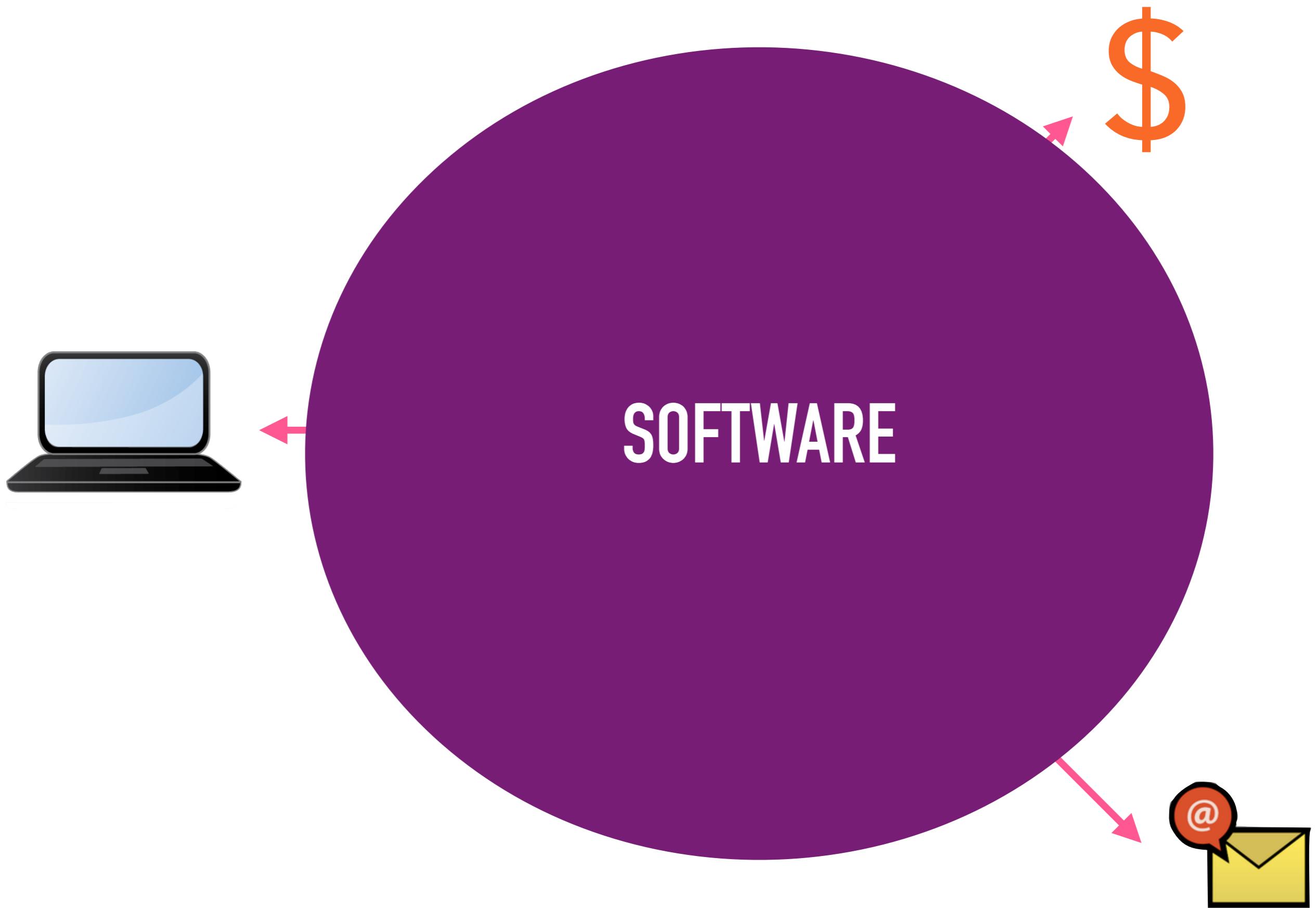
STORY 2



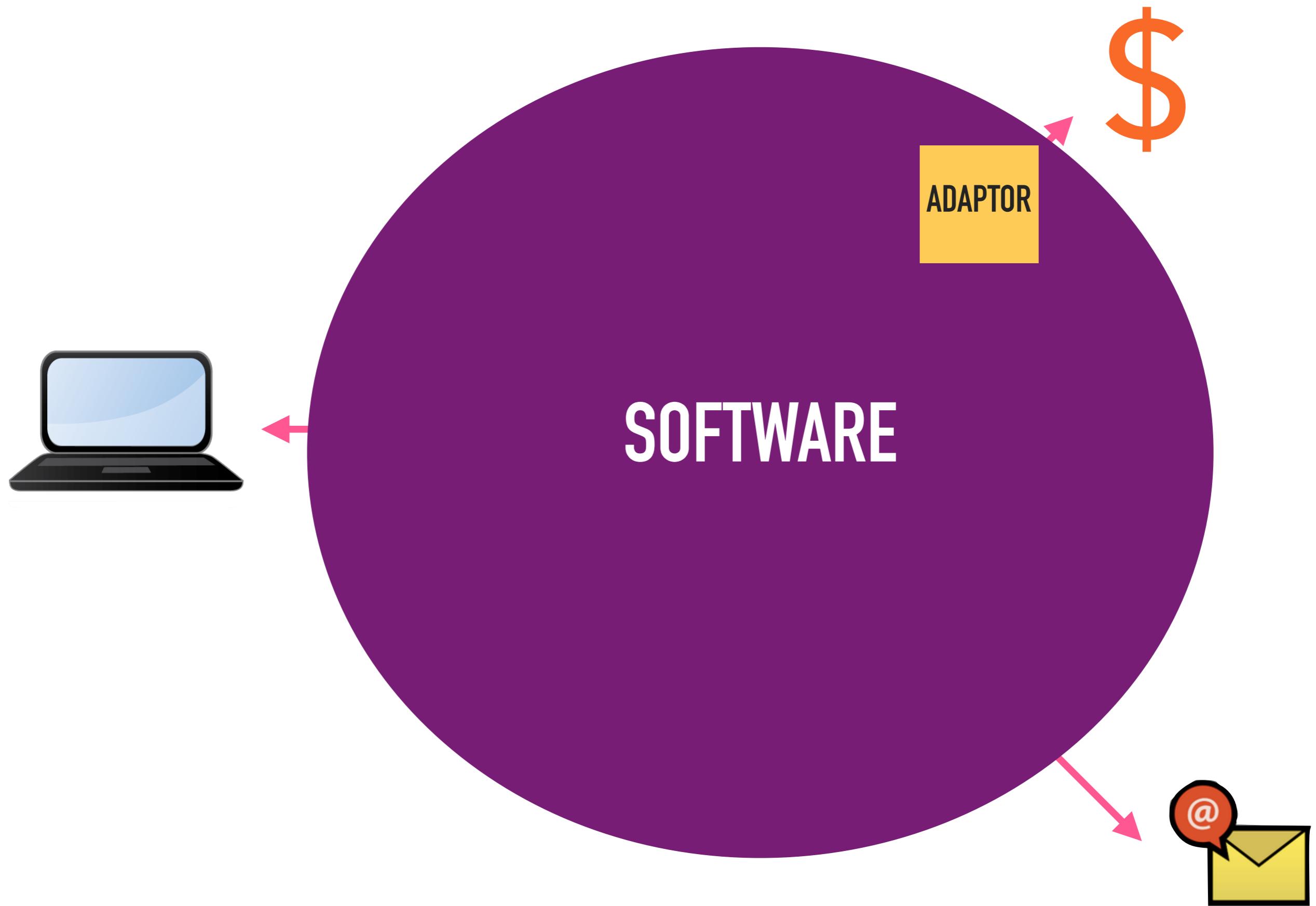
STORY 2



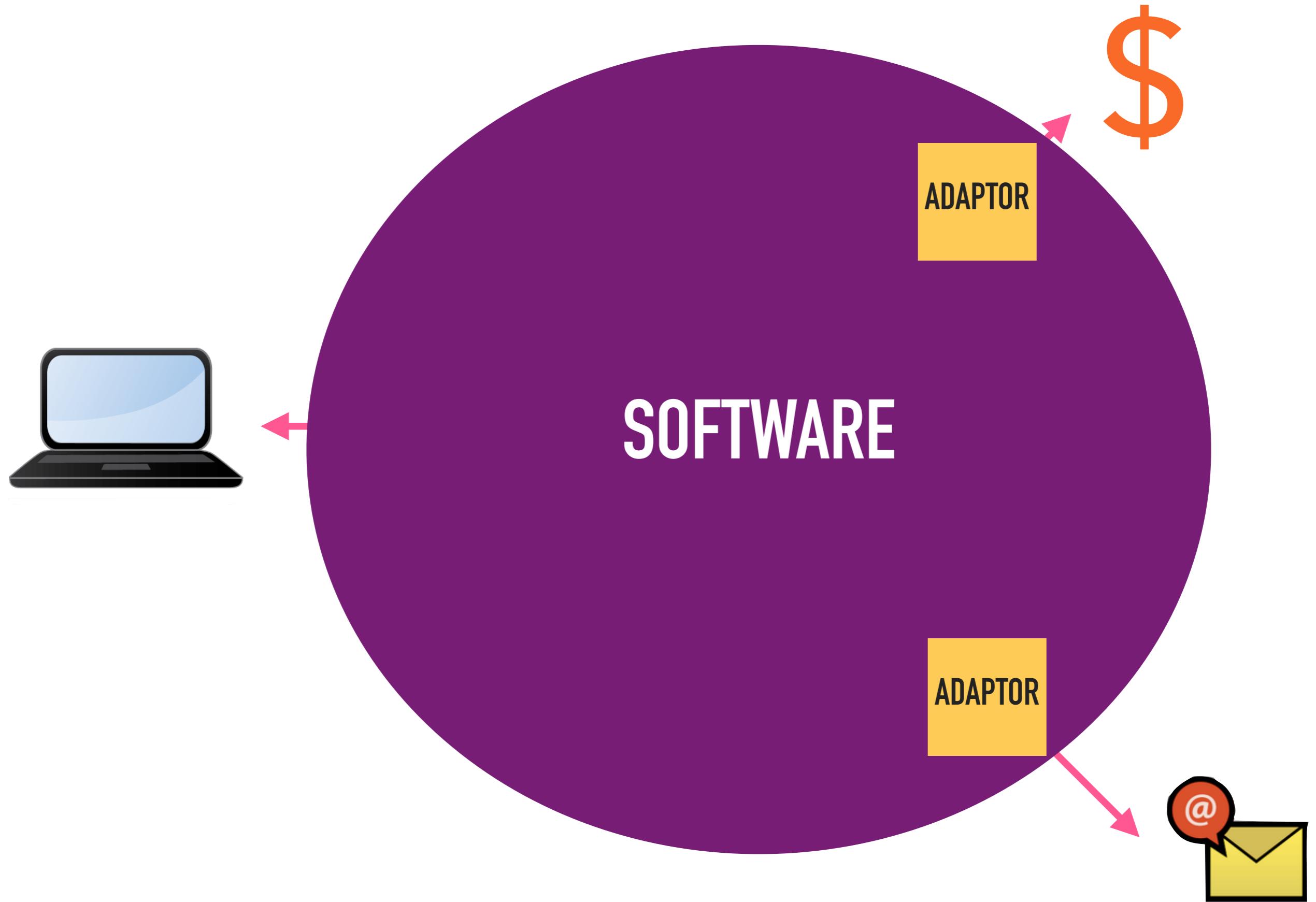
STORY 2



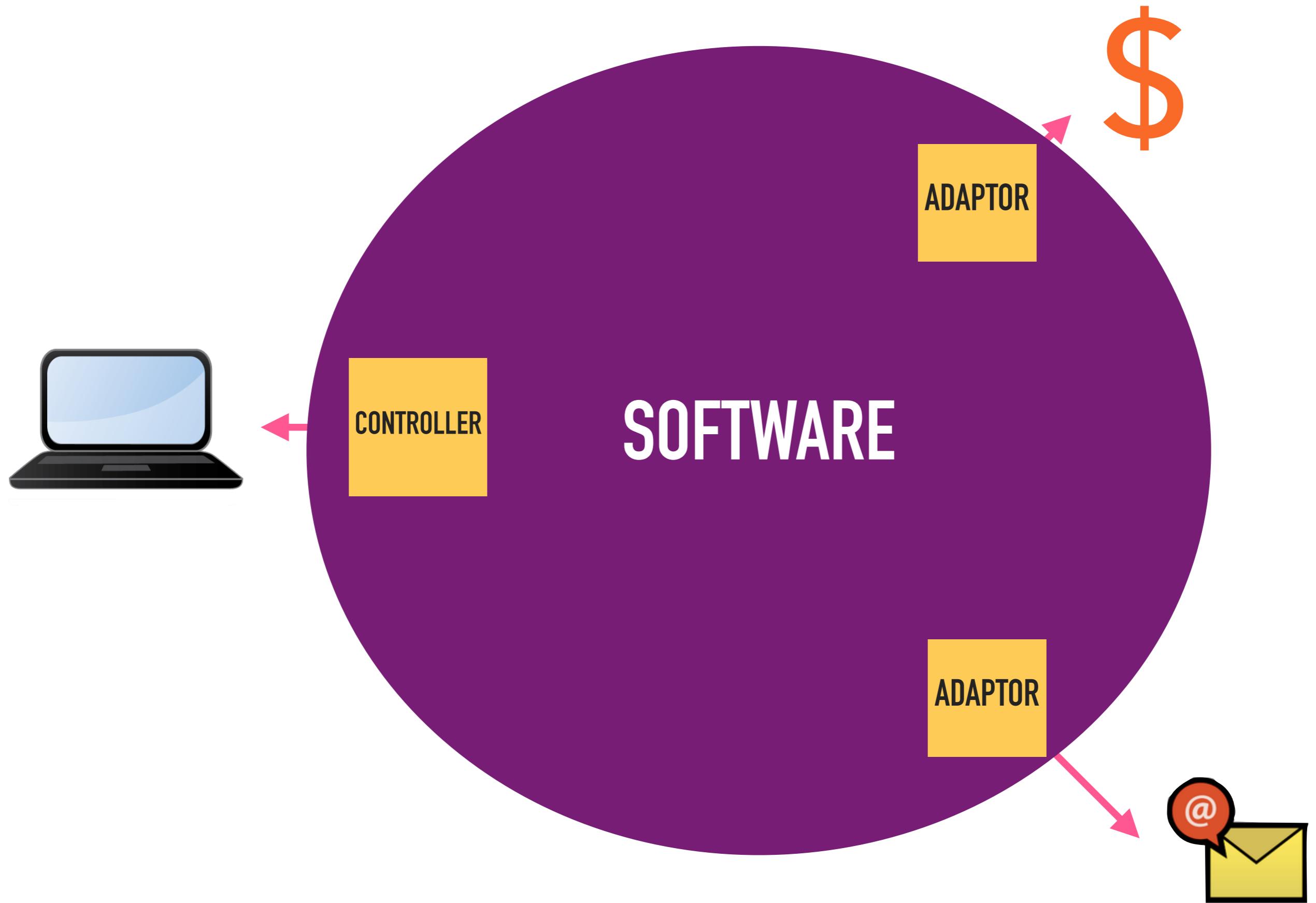
STORY 2



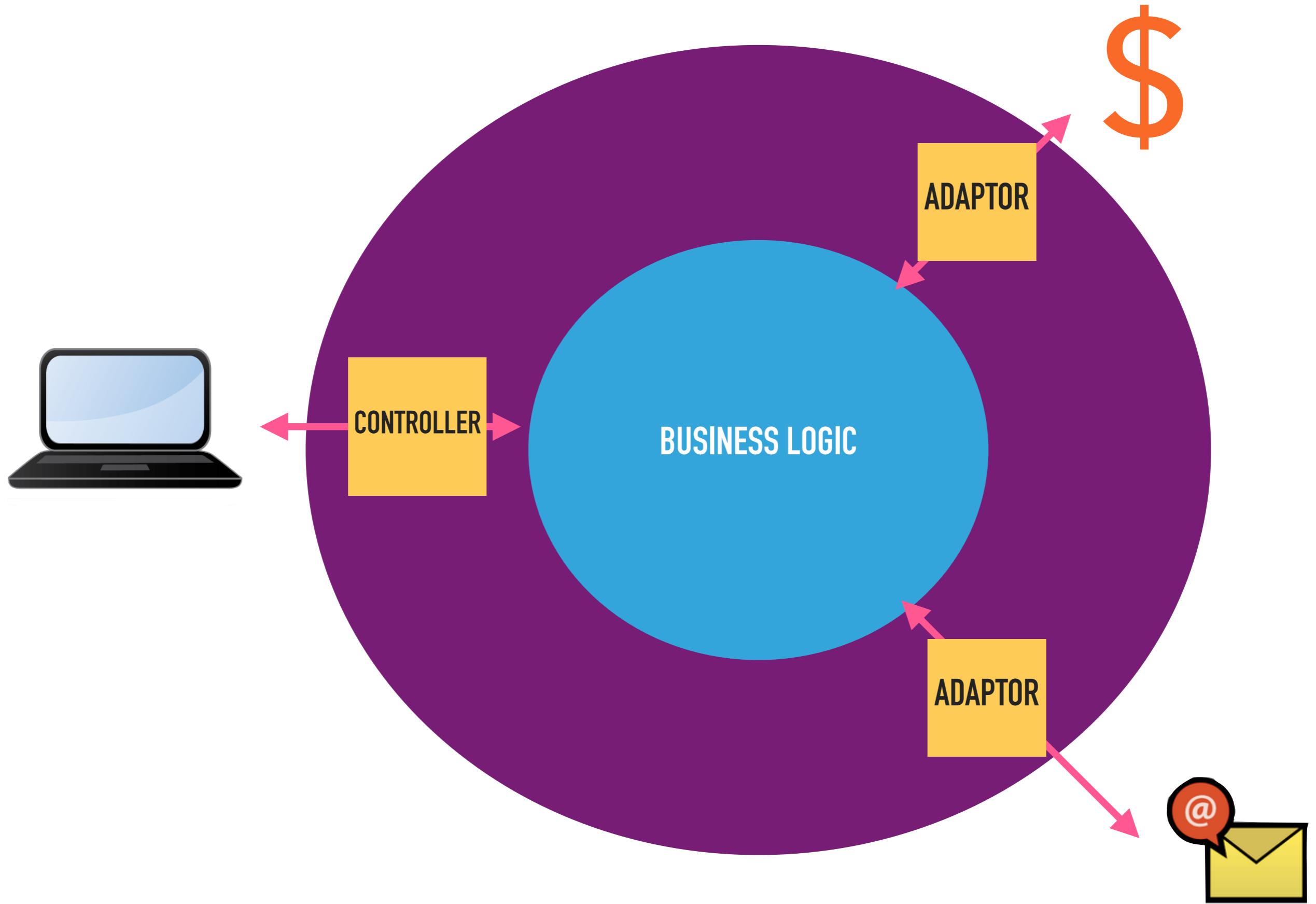
STORY 2



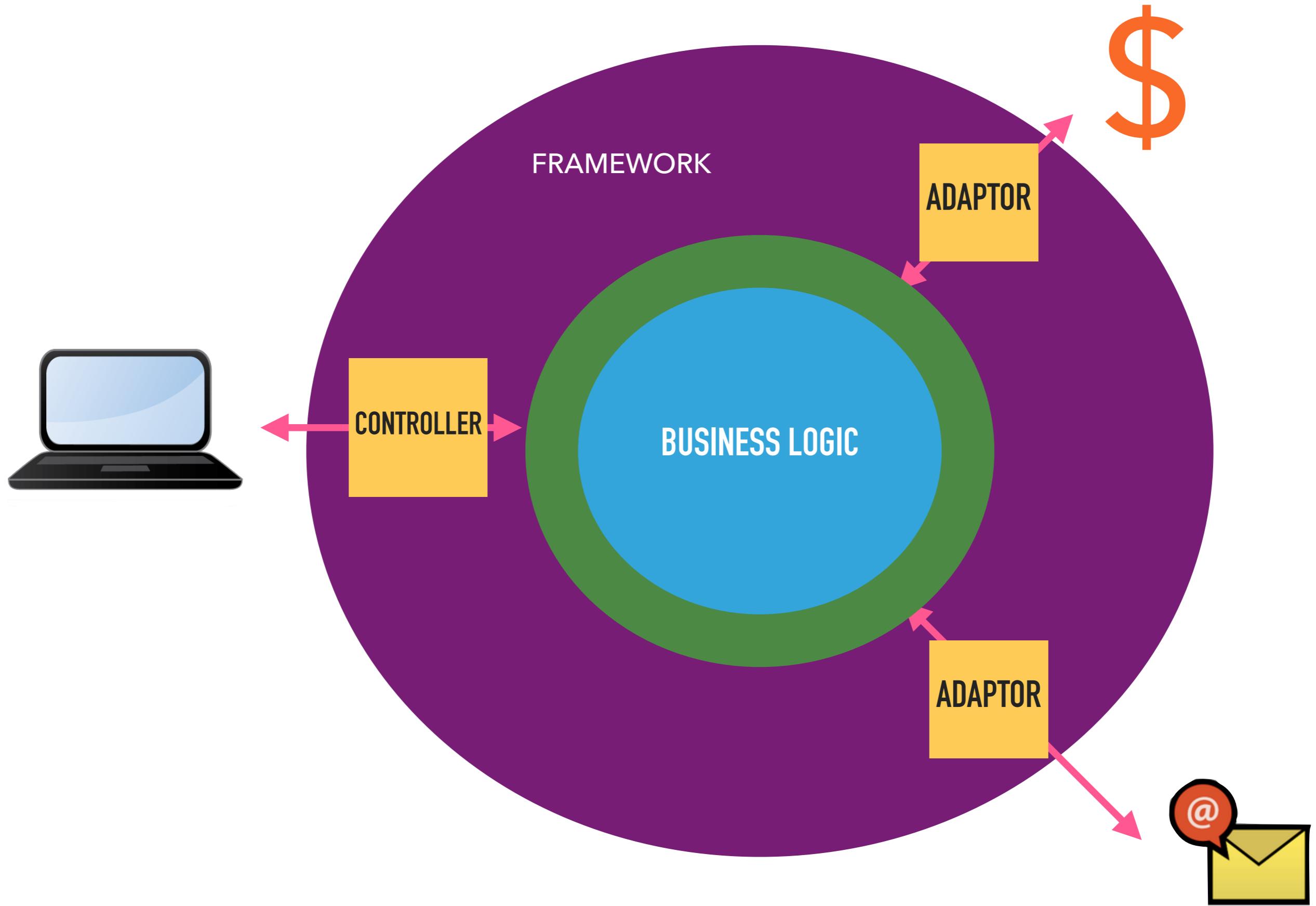
STORY 2



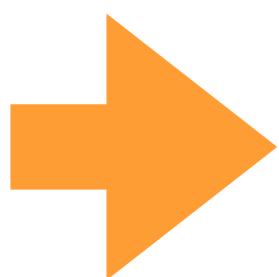
STORY 2



STORY 2



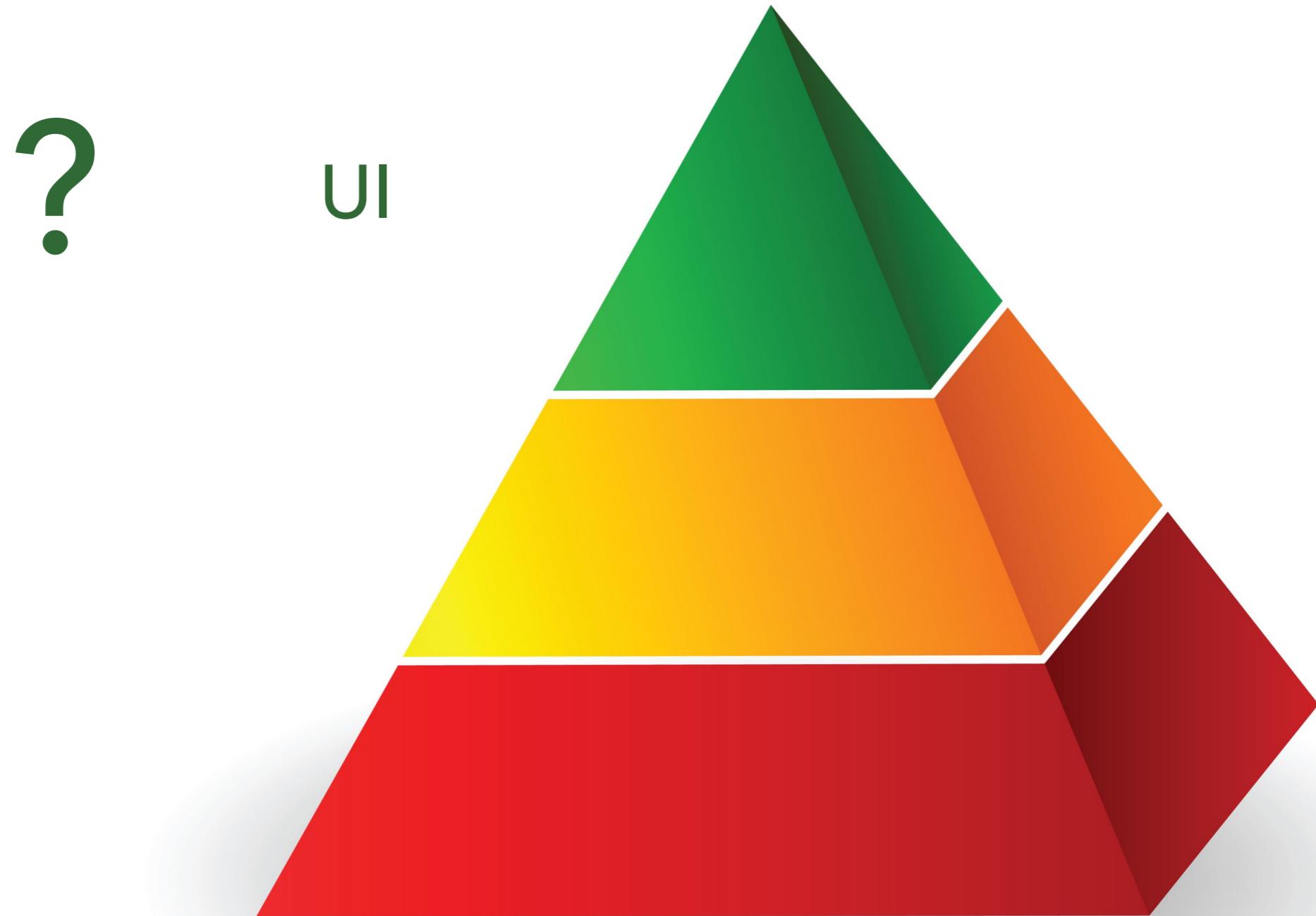
STORY 2



Integration

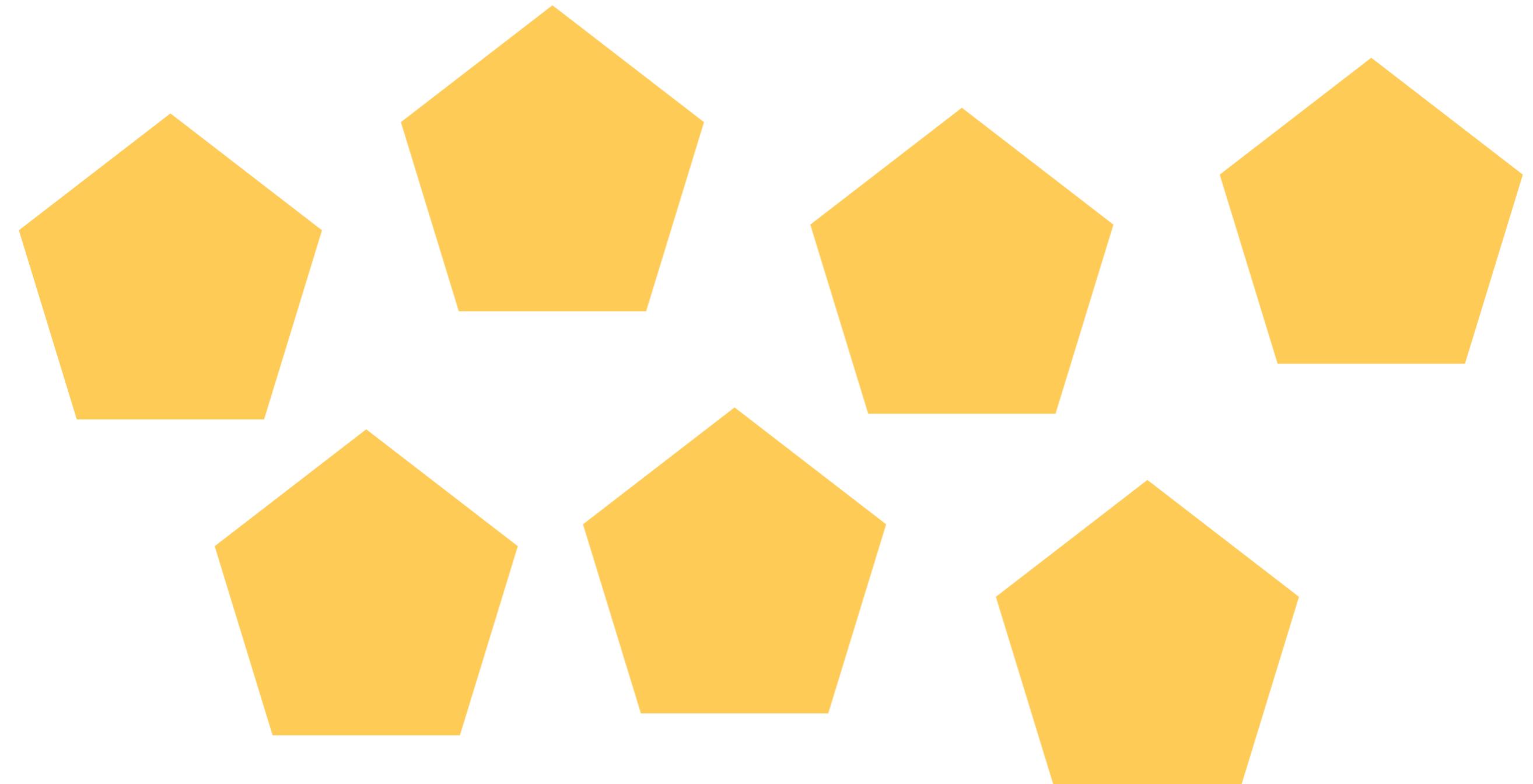


STORY 2



STORY 2

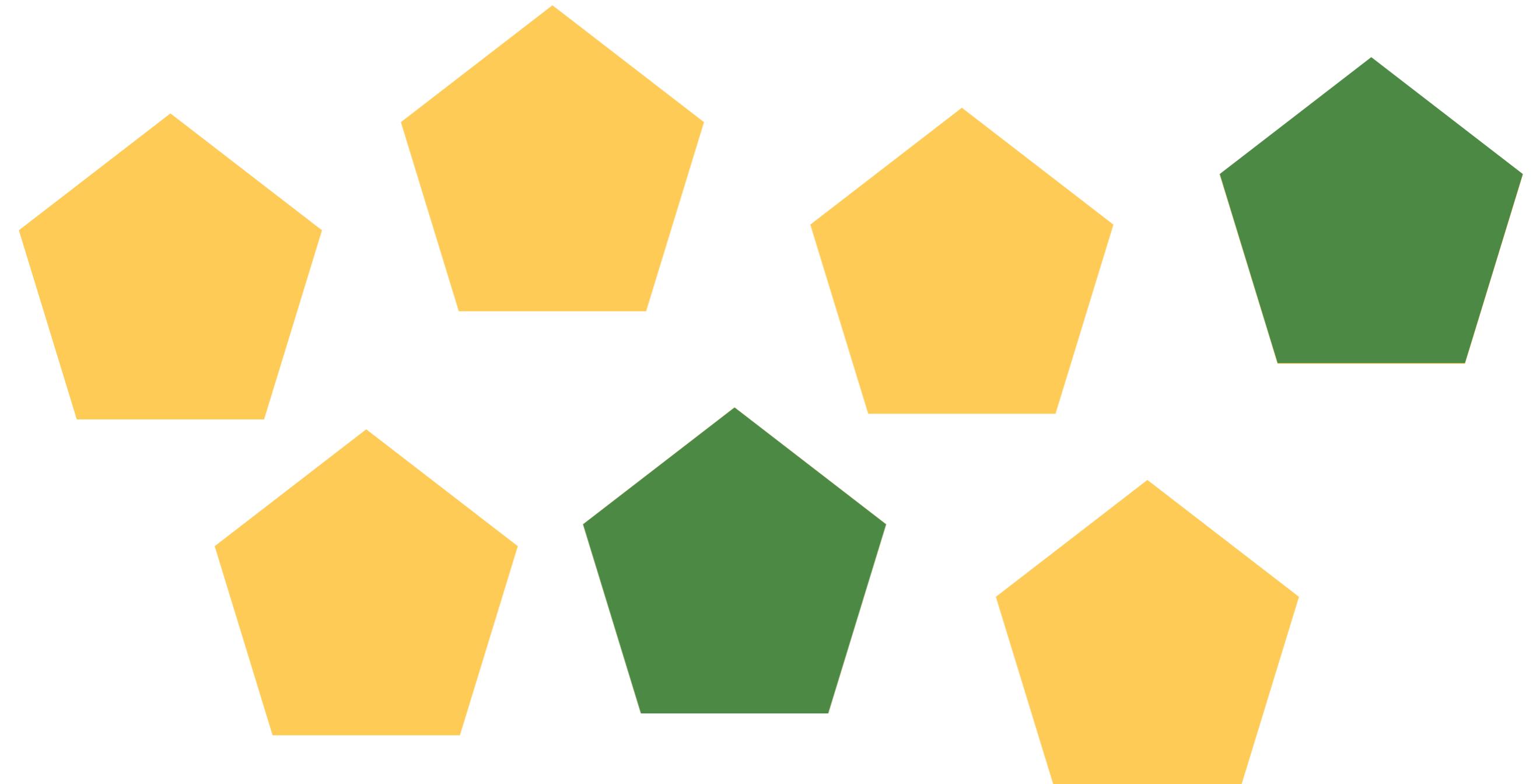
WHAT DO WE TEST AT THE UI LEVEL?



@daveliddament

STORY 2

WHAT DO WE TEST AT THE UI LEVEL?



@daveliddament

THE MORAL OF STORY 2...

- ▶ Testing an application's business logic via at integration level is much easier than at the UI level.
 - ▶ Coupling between test and SUT via the Service Layer.
 - ▶ Still need some testing at UI level.
 - ▶ We need to architect our code in a way to make this possible.
 - ▶ Business logic has no knowledge of the world around it.
 - ▶ I really like doing this kind of testing!

**DECOPLED TESTS REDUCE THE
DEVELOPMENT AND MAINTENANCE
COSTS OF THE TEST SUITE.**

BUT ...

Parts of my test suite are still tightly coupled to the software I'm testing...



#3

WE EXPAND TO OFFER THE SERVICE TO MULTIPLE COMPANIES

- ▶ Each company has a branded page on their own subdomain.
- ▶ Could only login from your company's subdomain.
- ▶ Behind the scenes authentication now requires:
 - ▶ username
 - ▶ password
 - ▶ subdomain

..... **FFFFFFFFFFFFFFFFFFF**..... 63 / 444 (14%)
..... **FFFFFFFFFFF**..... 126 / 444 (28%)
..... **FFFFFFFFFFF**..... 189 / 444 (42%)
FFFFFFFFFFF..... 252 / 444 (56%)
..... **FF** .. **FFF** .. **FFFF** .. **FFFF** .. **FFFF** .. 315 / 444 (70%)
..... **FFFFFFFFFFF**..... 378 / 444 (85%)
FFFFFFFFFFF..... **FFFFFFFFFFF**..... 441 / 444 (99%)
...

Time: 20 minutes 54 seconds, Memory: 24.75MB

There were lots of failures:



ONE OF THE MANY FAILING TESTS...

Does an individual's score get
correctly allocated to their team?

STORY 3



SEEDING A DATABASE

users:

- **name:** Anna
email: anna@acme.com
password: Passw1rd
team: Apple
- **name:** Bob
email: bob@example.com
password: Passw5rd
team: Apple

SEEDING A DATABASE

users:

- **name:** Anna
~~**email:** ann@acme.com~~
~~**password:** Passw1rd~~
~~**team:** Apple~~
- **name:** Bob
email: bob@example.com
password: Passw5rd
team: Apple

STORY 3

BUILDING DATA FIXTURES



@daveliddament

HAND BUILDING

```
$user = $this->userService->registerUser(  
    "anna@acme.com",  
    "Anna",  
    "Passw0rd");
```

HAND BUILDING

```
$user = $this->userService->registerUser(  
    "anna@acme.com",  
    "Anna",  
    "Passw0rd",  
    $companyId);
```

HAND BUILDING

```
$user = $this->userService->registerUser(  
    "anna@acme.com",  
    "Anna",  
    "PassWord",  
    $password)
```



OBJECT MOTHER

```
$user = $this->userObjectMother->getAnna();  
  
// User will have default values for name,  
// email, etc
```

OBJECT MOTHER: IMPLEMENTATION

```
class UserObjectMother {  
  
    public function getAnna(): User {  
  
        ... return user if already created ...  
  
        $user = $userService->registerUser(  
            "anna@acme.com",  
            "Anna",  
            "Passw0rd");  
  
        return $user;  
    }  
}
```

OBJECT MOTHER: IMPLEMENTATION

```
class UserObjectMother {  
  
    public function getAnna(): User {  
  
        ... return user if already created ...  
  
        $user = $userService->registerUser(  
            "anna@acme.com",  
            "Anna",  
            "Passw0rd"  
            $companyId);  
  
        return $user;  
    }  
}
```

TEST BUILDER: 1

```
$userBuilder = new UserBuilder();
$user = $userBuilder->build();

// User will have default values for
// name, email, etc
```

USING A TEST BUILDER (2)

```
$userBuilder = new UserBuilder();  
$user = $userBuilder  
    ->name("Annabelle")  
    ->password("Passw4rd")  
    ->team("Banana")  
    ->build();
```

DEFER TO OTHER OBJECT MOTHERS / BUILDERS

```
class UserObjectMother {  
  
    public function getAnna(): User {  
  
        $companyId = $this->companyObjectMother()  
            ->getAcmeCompany();  
  
        $user = $userService->registerUser(  
            "anna@acme.com",  
            "Anna",  
            "Passw0rd"  
            $companyId);  
  
        return $user;  
    }  
}
```

STORY 3

HYBRID

users:

- name: Anna
email: anna@acme.com
password: Passw1rd
team: Apple

- name: Bob
email: bob@example.com
password: Passw5rd
team: Apple

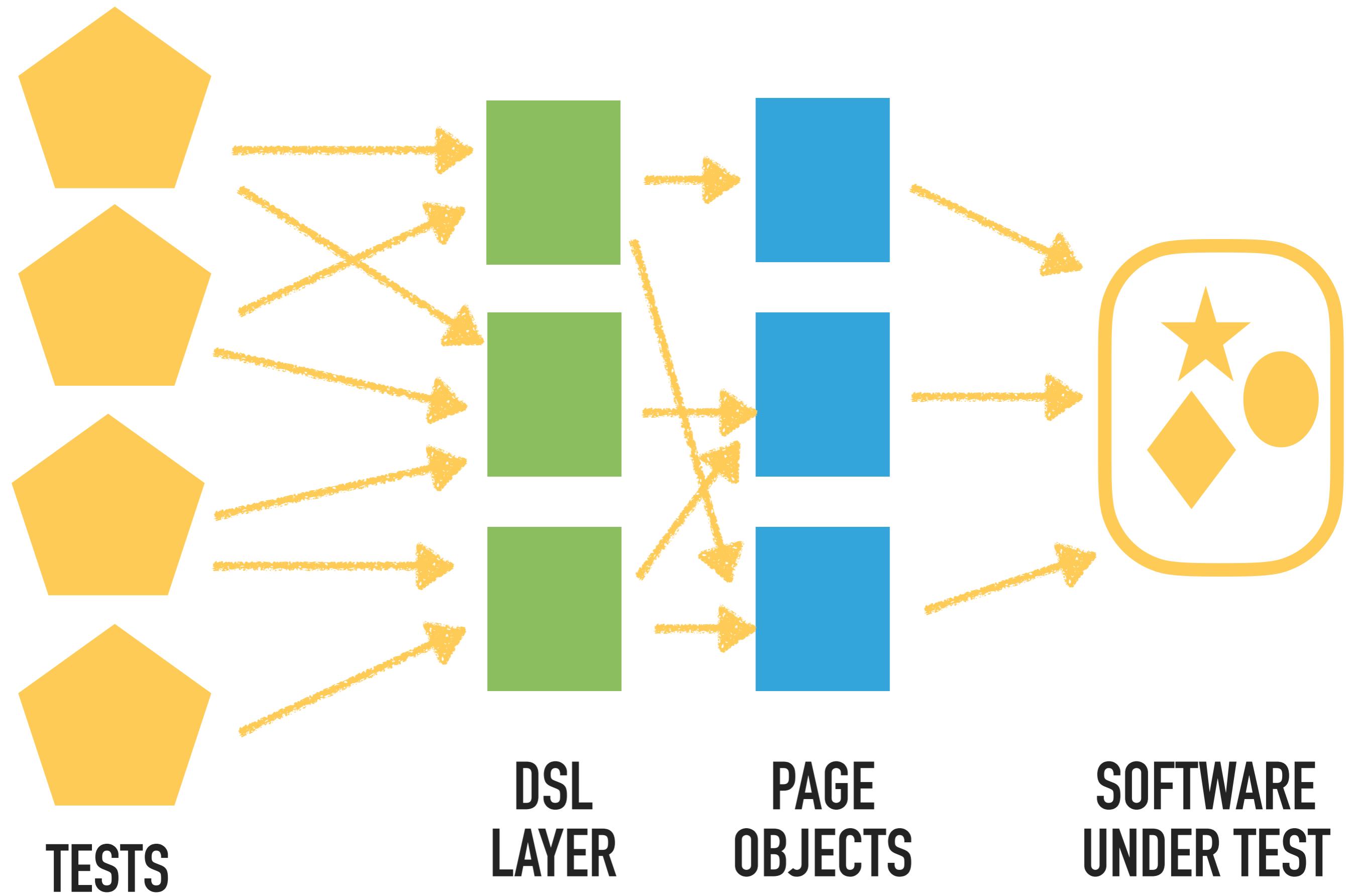


MORAL OF STORY 3...

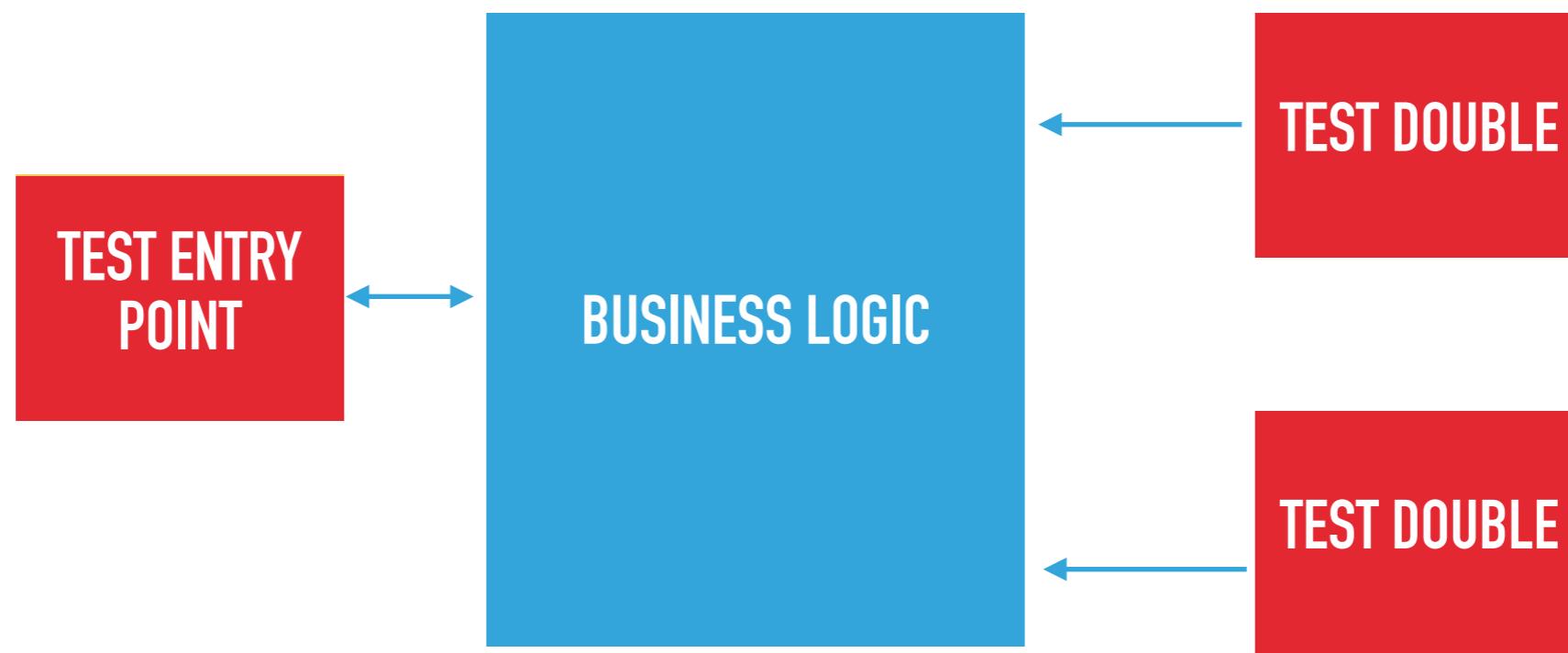
- ▶ Use patterns like Object Mothers / Test Builders for building data fixtures.
 - ▶ Makes tests more robust to change.
- ▶ Decoupling our tests from the software under test.

**DECOPLED TESTS REDUCE THE
DEVELOPMENT AND MAINTENANCE
COSTS OF THE TEST SUITE.**

STORY 1



STORY 2



STORY 3



@daveliddament

Dave Liddament

Lamp Bristol

@daveliddament

A photograph taken underwater. A scuba diver is on the right, wearing a black wetsuit and a green tank. They are holding a bright white dive light that is pointing towards the left. In the water, there is a yellowish-orange fish swimming towards the light. Below the fish, there are large, dark green, leafy plants called kelp. The water is a deep teal color.

Organise PHP-SW
Author of Static Analysis Results Baseline
20 years of writing software (C, Java, Python, PHP)

Dave Liddament
Lamp Bristol

@daveliddament

Thank you for
listening

Organise PHP-SW
Author of Static Analysis Results Baseline
20 years of writing software (C, Java, Python, PHP)



IMAGE CREDITS

- ▶ Decouple © Can Stock Photo / iqoncept
- ▶ Story © Can Stock Photo / Palto
- ▶ Man On Moon: © Can Stock Photo / openlens
- ▶ Confession © Can Stock Photo / lenm
- ▶ Pyramid © Can Stock Photo / Arcady
- ▶ Feedback © Can Stock Photo / kikkerdirk
- ▶ Scripts © Can Stock Photo / LoopAll
- ▶ Tools © Can Stock Photo / dedMazay
- ▶ Builder © Can Stock Photo / aleksangel
- ▶ Database © Can Stock Photo / dvarg
- ▶ Fake © Can Stock Photo / carmendorin
- ▶ People chatting © Can Stock Photo / studioworkstock
- ▶ Seeding: © Can Stock Photo / italianestro
- ▶ Banking app © Can Stock Photo / tashka2000
- ▶ Old Telephone © Can Stock Photo / barneyboogles
- ▶ Bank © Can Stock Photo / dolgachov
- ▶ Coupler © Can Stock Photo / ArtImages
- ▶ Bank Building © Can Stock Photo / dvarg
- ▶ Online Shopping © Can Stock Photo / Wetzkaz