international PHP conference

AssertTrue(isDecoupled("MyTests"))

Dave Liddament @daveliddament

DECOUPLED TESTS REDUCE THE DEVELOPMENT AND MAINTENANCE COSTS OF THE TEST SUITE.

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

IS THIS TALK FOR YOU?

IS THIS TALK FOR YOU?

YES

- Some automated testing.
- You want high level concepts you can apply when testing applications via the UI or at integration level.

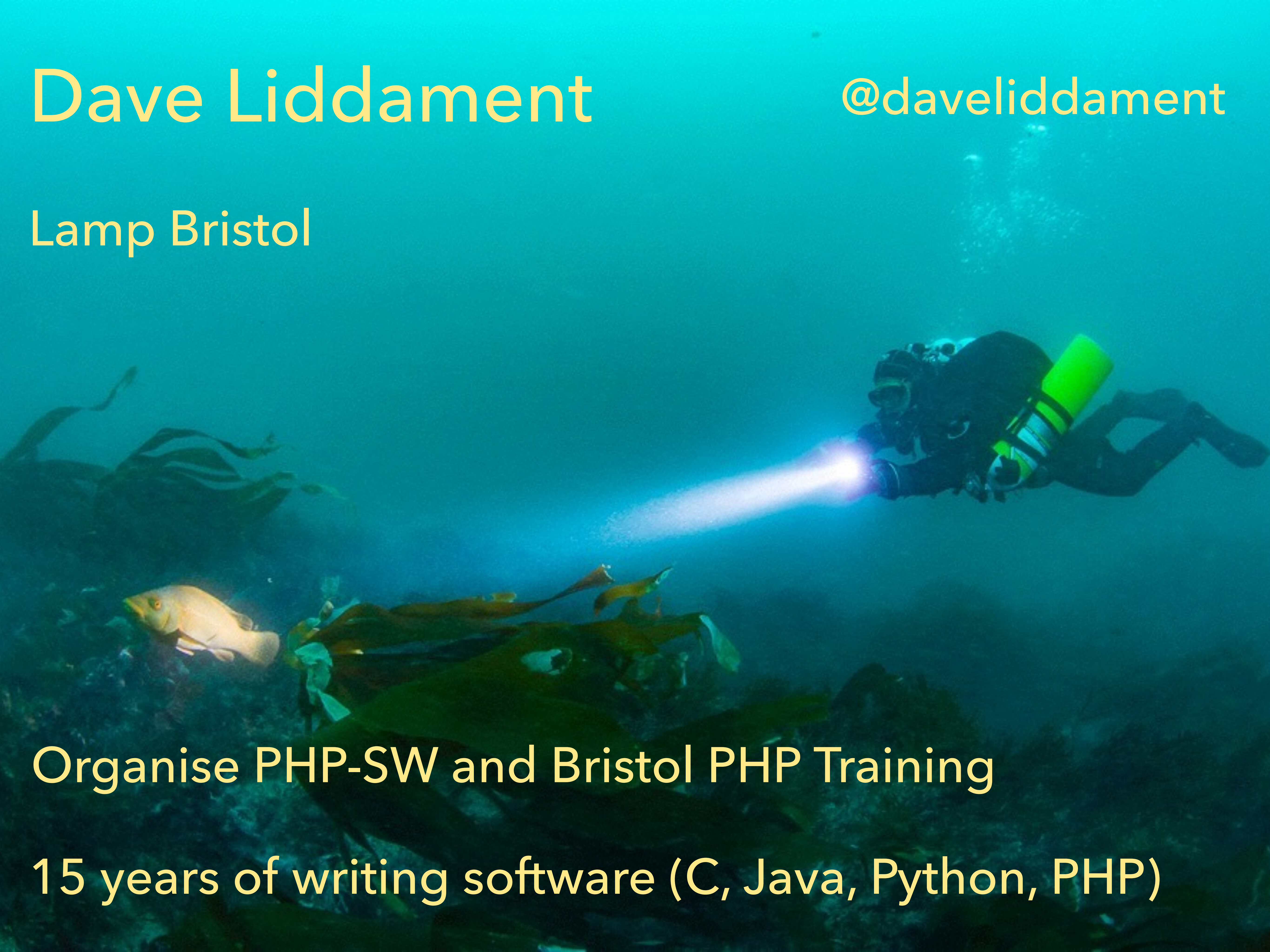
IS THIS TALK FOR YOU?

YES

- Some automated testing.
- You want high level concepts you can apply when testing applications via the UI or at integration level.

NO

- Experienced tester.
- You already write unit, integrations and end to end tests.
- You don't abstract talks.



Dave Liddament

@daveliddament

Lamp Bristol

Organise PHP-SW and Bristol PHP Training

15 years of writing software (C, Java, Python, PHP)

AGENDA

AGENDA

- Why

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)
```

```
................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.  63 / 444 ( 14%)
............................................................... 126 / 444 ( 28%)
.........................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF...... 189 / 444 ( 42%)
FFFFFFFFFFFF................................................... 252 / 444 ( 56%)
...................FF....FFFF.......FFFFFFFFFFFFFF............. 315 / 444 ( 70%)
..................................FFFFFFFFFFFFFFFFFFFFFFFFFFF.. 378 / 444 ( 85%)
FFFFFFFFFFFFFF........FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 441 / 444 ( 99%)
...

Time: 1.55 seconds, Memory: 24.75MB
```

@daveliddament

```
................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.  63 / 444 ( 14%)
............................................................... 126 / 444 ( 28%)
.........................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF...... 189 / 444 ( 42%)
FFFFFFFFFFFF................................................... 252 / 444 ( 56%)
...................FF....FFFF.......FFFFFFFFFFFFFF............. 315 / 444 ( 70%)
..................................FFFFFFFFFFFFFFFFFFFFFFFFFFF.. 378 / 444 ( 85%)
FFFFFFFFFFFFFF........FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 441 / 444 ( 99%)
...

Time: 1.55 seconds, Memory: 24.75MB

 There were lots of failures:
```

@daveliddament

```
................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.  63 / 444 ( 14%)
............................................................... 126 / 444 ( 28%)
.........................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF...... 189 / 444 ( 42%)
FFFFFFFFFFFF................................................... 252 / 444 ( 56%)
...................FF....FFFF.......FFFFFFFFFFFFFF............. 315 / 444 ( 70%)
..................................FFFFFFFFFFFFFFFFFFFFFFFFFFF.. 378 / 444 ( 85%)
FFFFFFFFFFFFFF........FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 441 / 444 ( 99%)
...

Time: 1.55 seconds, Memory: 24.75MB

 There were lots of failures:
```

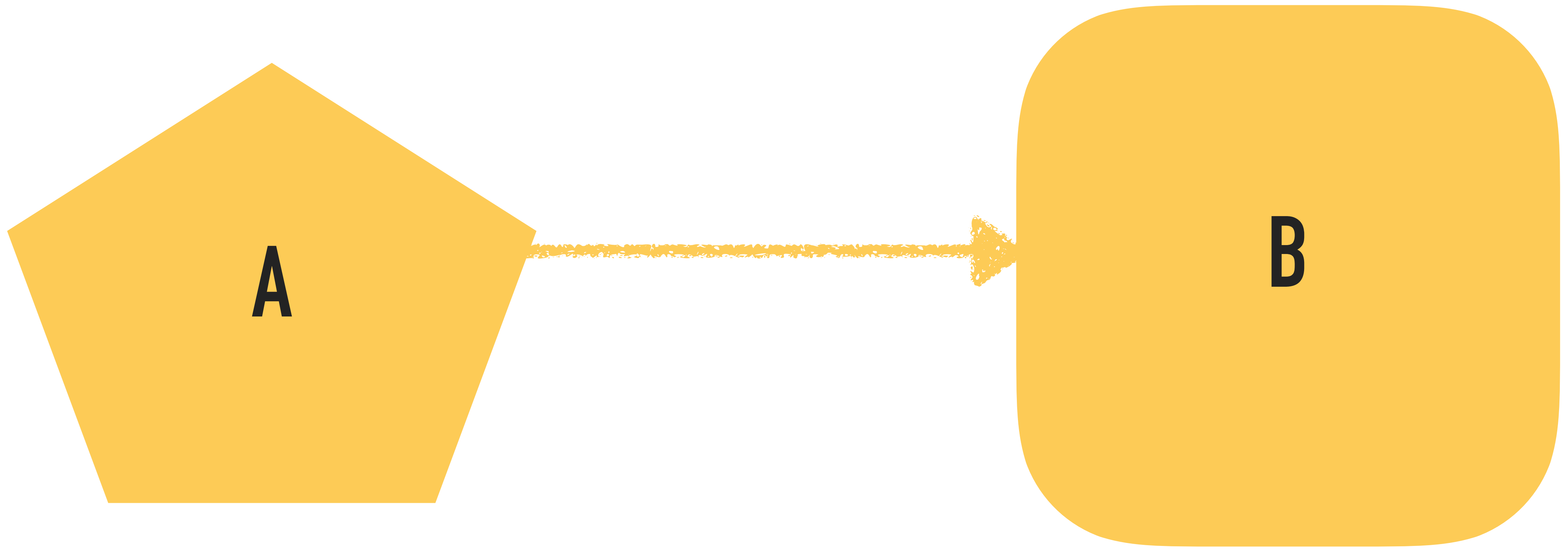
😞

@daveliddament

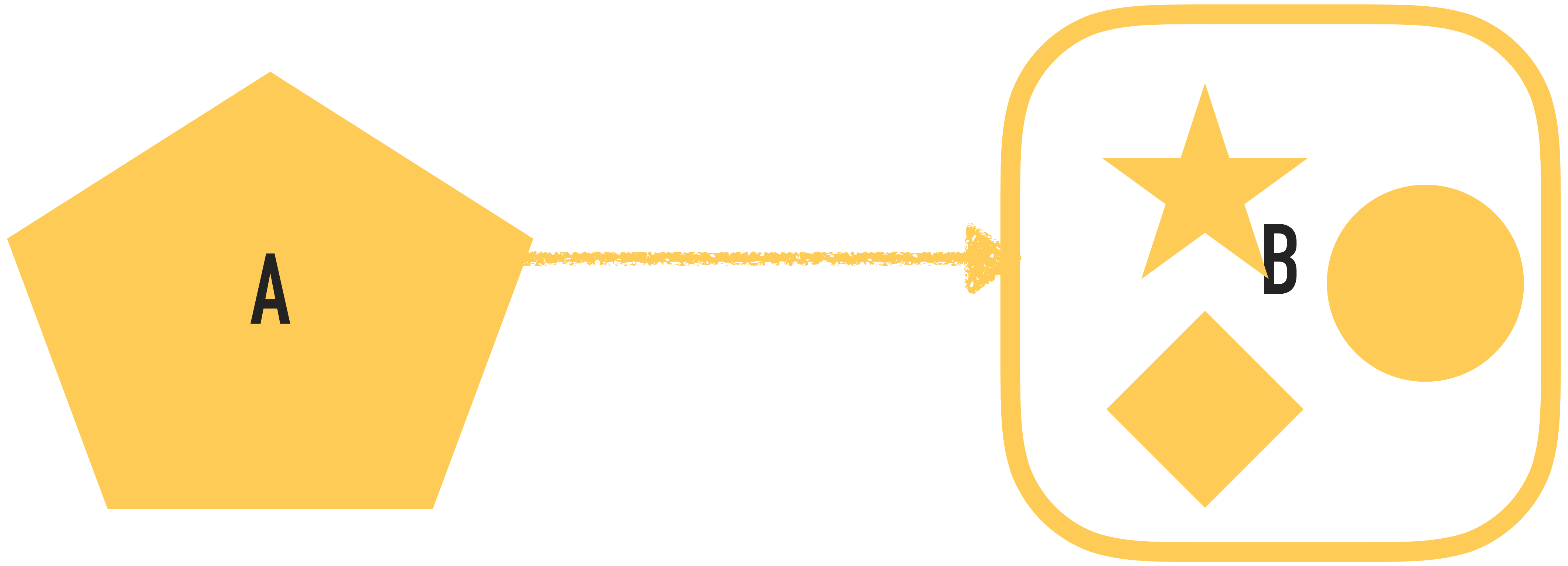
TERMINOLOGY

COUPLING

COUPLING



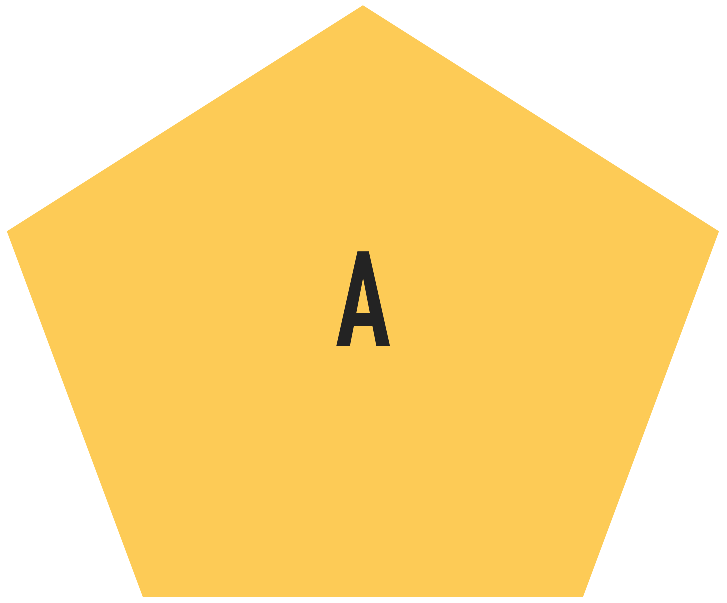
COUPLING



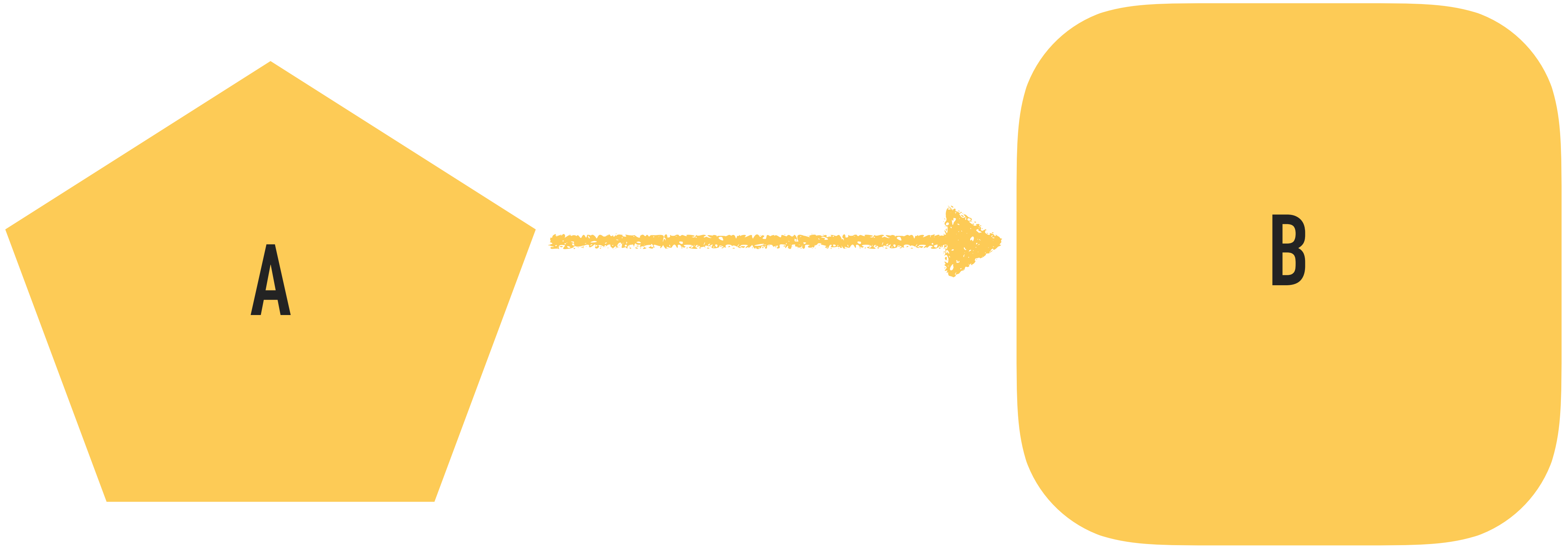
TERMINOLOGY

@daveliddament

TEST DOUBLES

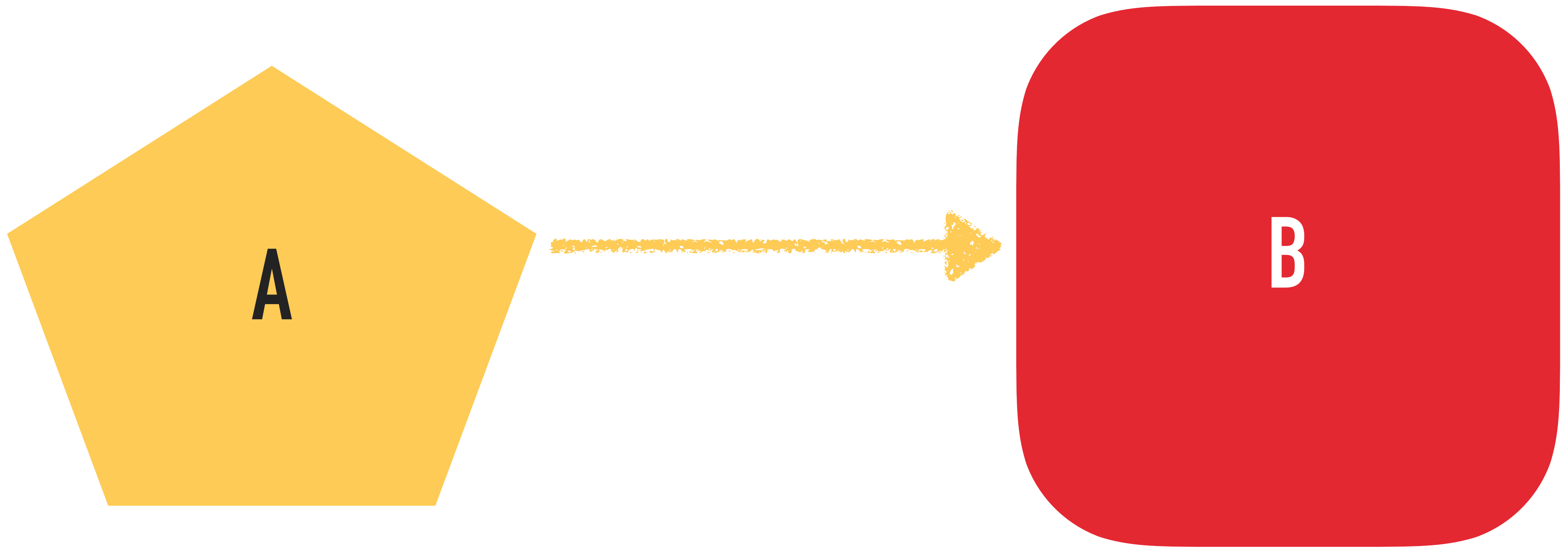


TEST DOUBLES



TERMINOLOGY

TEST DOUBLES

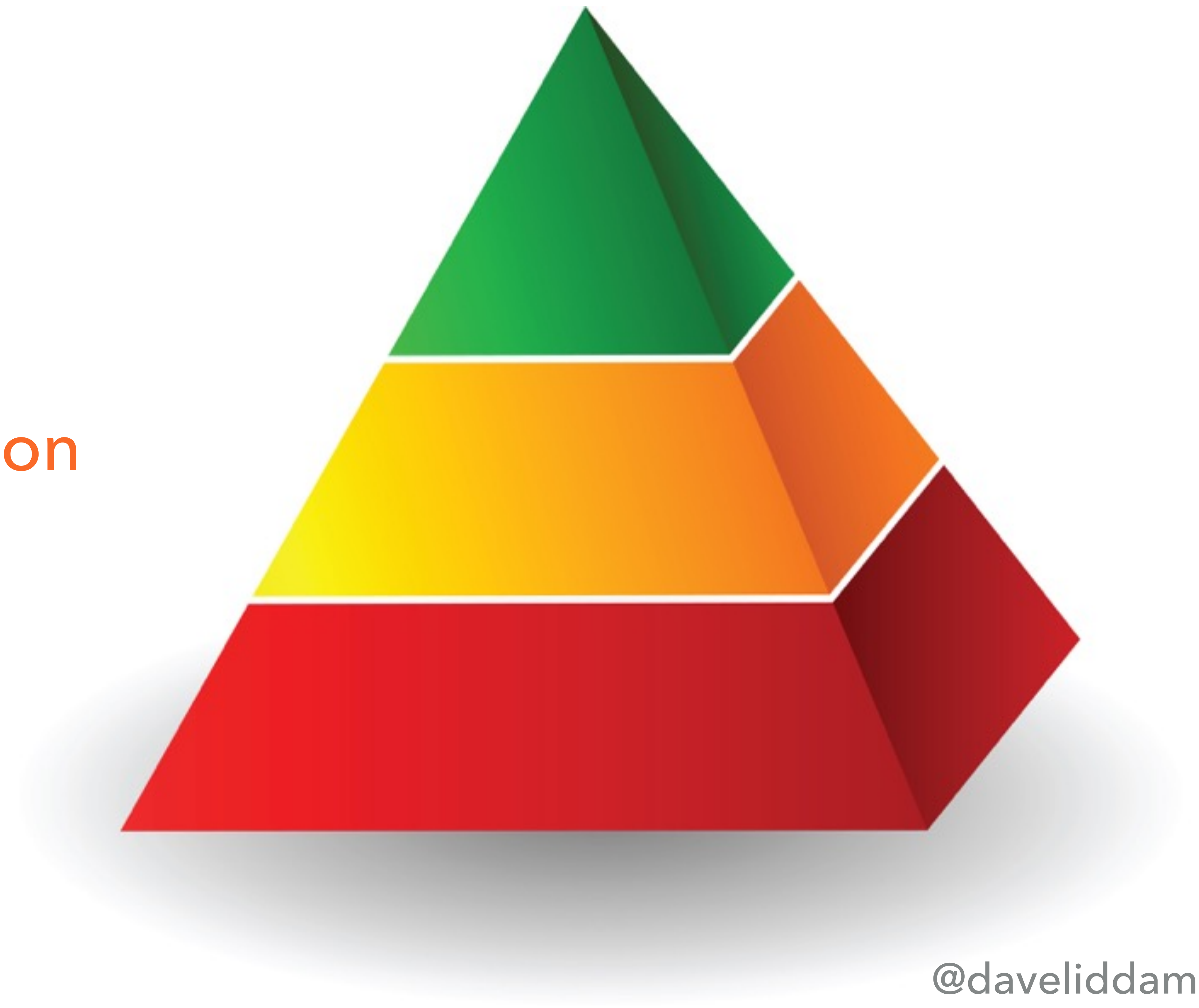


TEST PYRAMID

UI

Integration

Unit



DECOUPLED TESTS REDUCE THE DEVELOPMENT AND MAINTENANCE COSTS OF THE TEST SUITE.

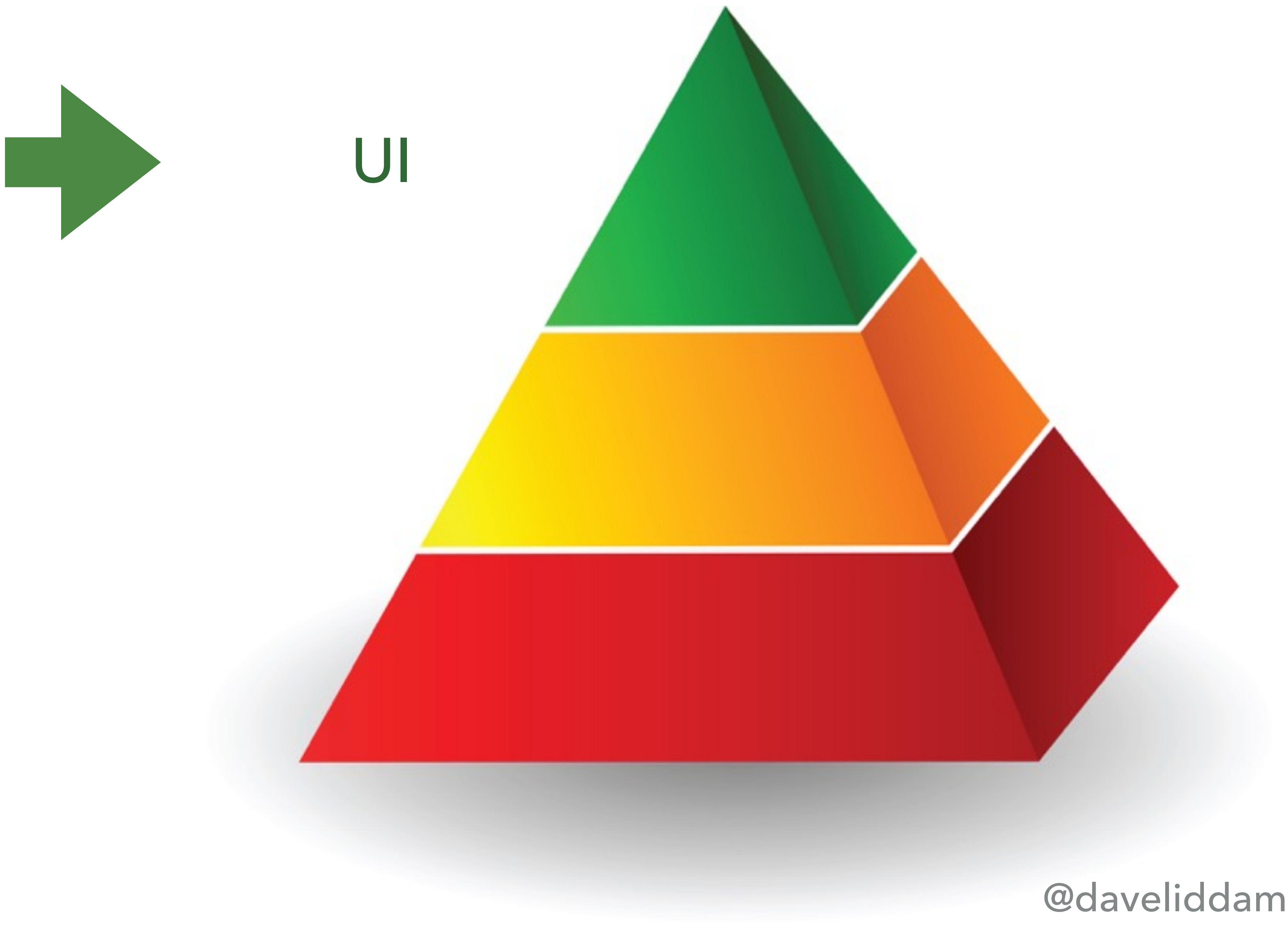
#1

@daveliddament

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

STORY 1



@daveliddament

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?

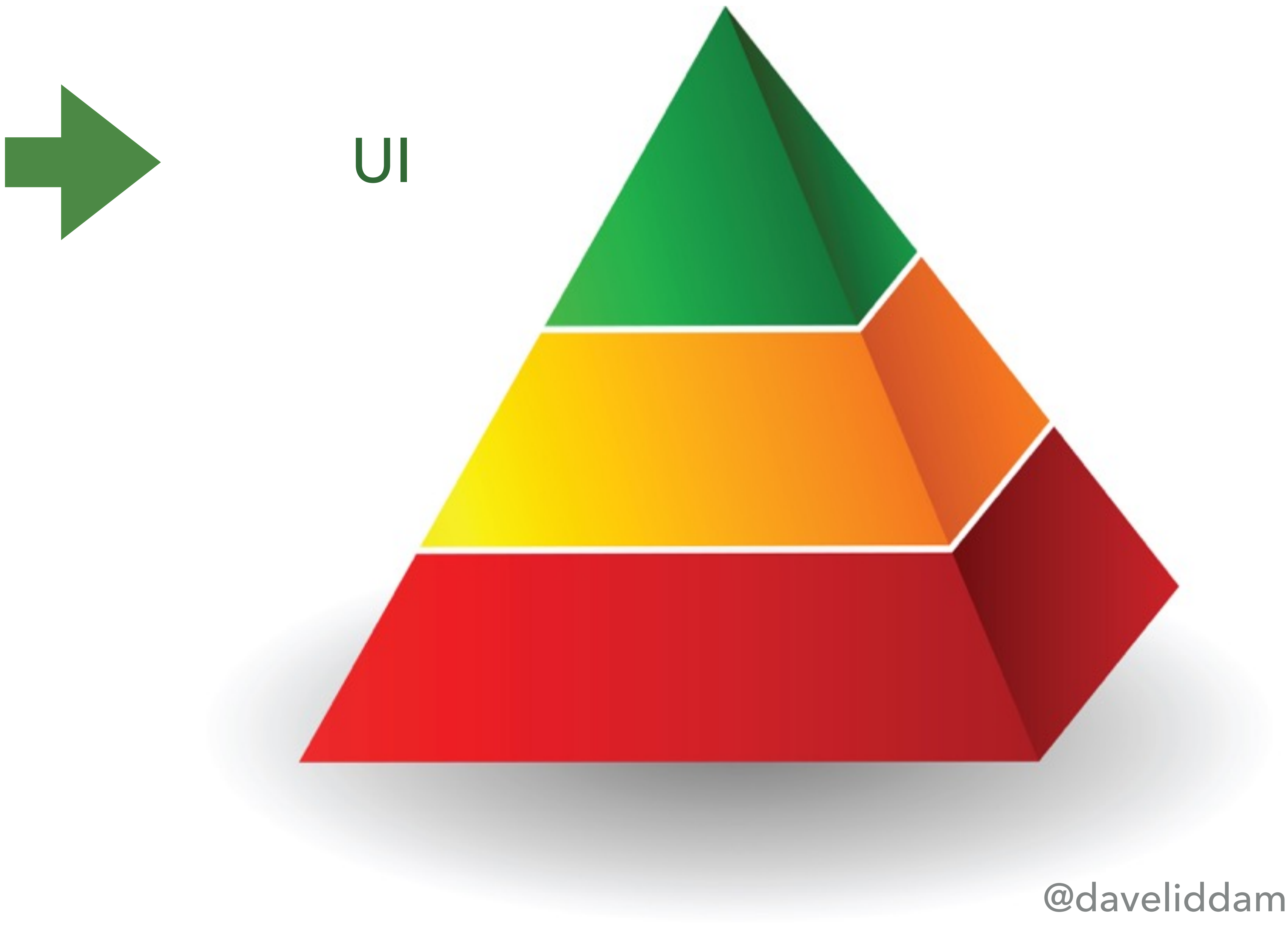
```
................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.  63 / 444 ( 14%)
............................................................... 126 / 444 ( 28%)
.........................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF...... 189 / 444 ( 42%)
FFFFFFFFFFFF................................................... 252 / 444 ( 56%)
...................FF....FFFF.......FFFFFFFFFFFFFF............. 315 / 444 ( 70%)
..................................FFFFFFFFFFFFFFFFFFFFFFFFFFF.. 378 / 444 ( 85%)
FFFFFFFFFFFFFF........FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 441 / 444 ( 99%)
...

Time: 20 minutes 54 seconds, Memory: 24.75MB

 There were lots of failures:
```

😞

@daveliddament

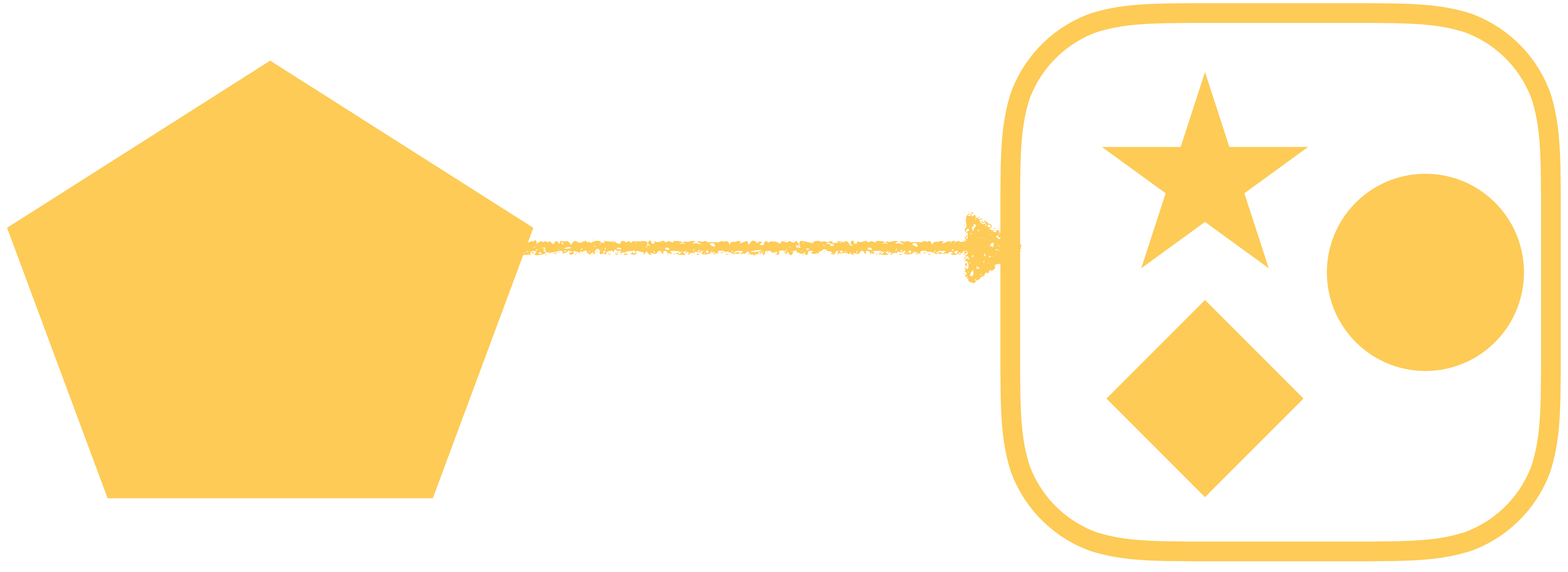
@daveliddament

STORY 1



@daveliddament

PROBLEM: TIGHT COUPLING

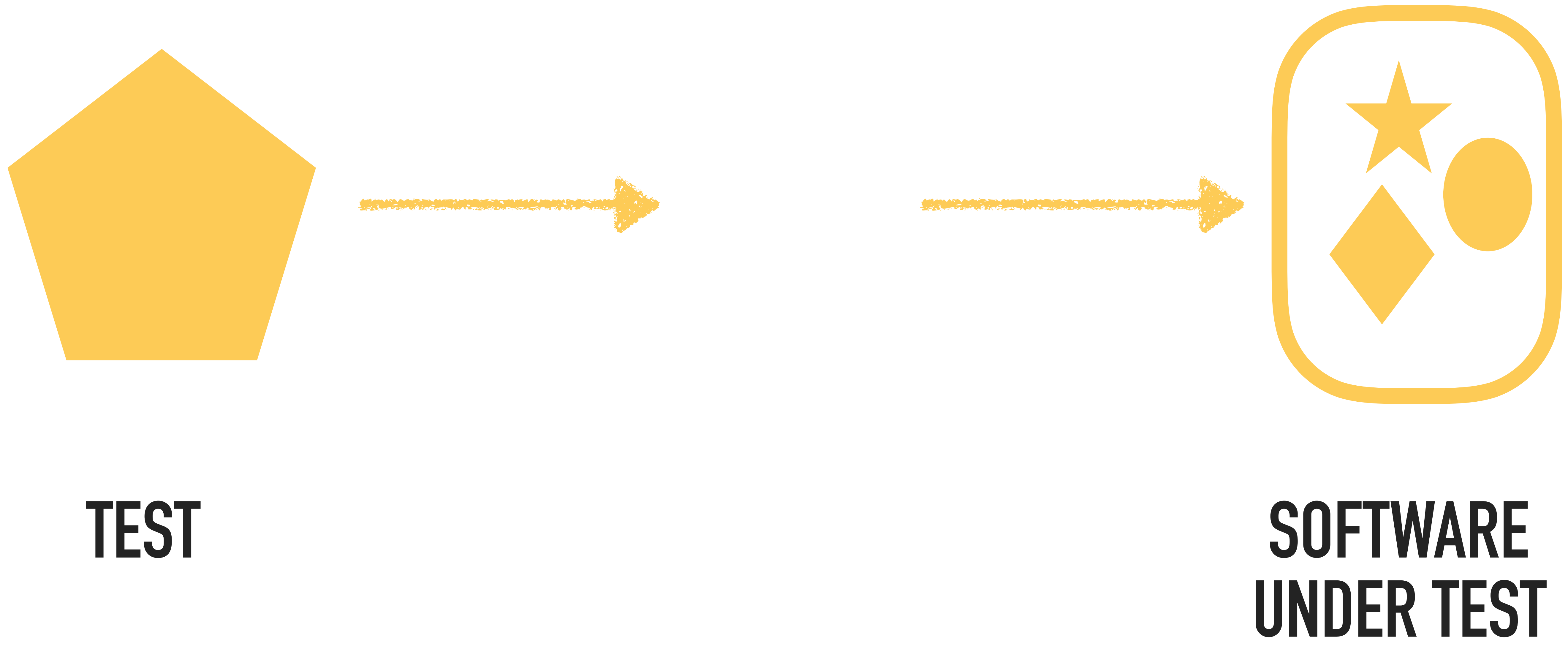


TEST

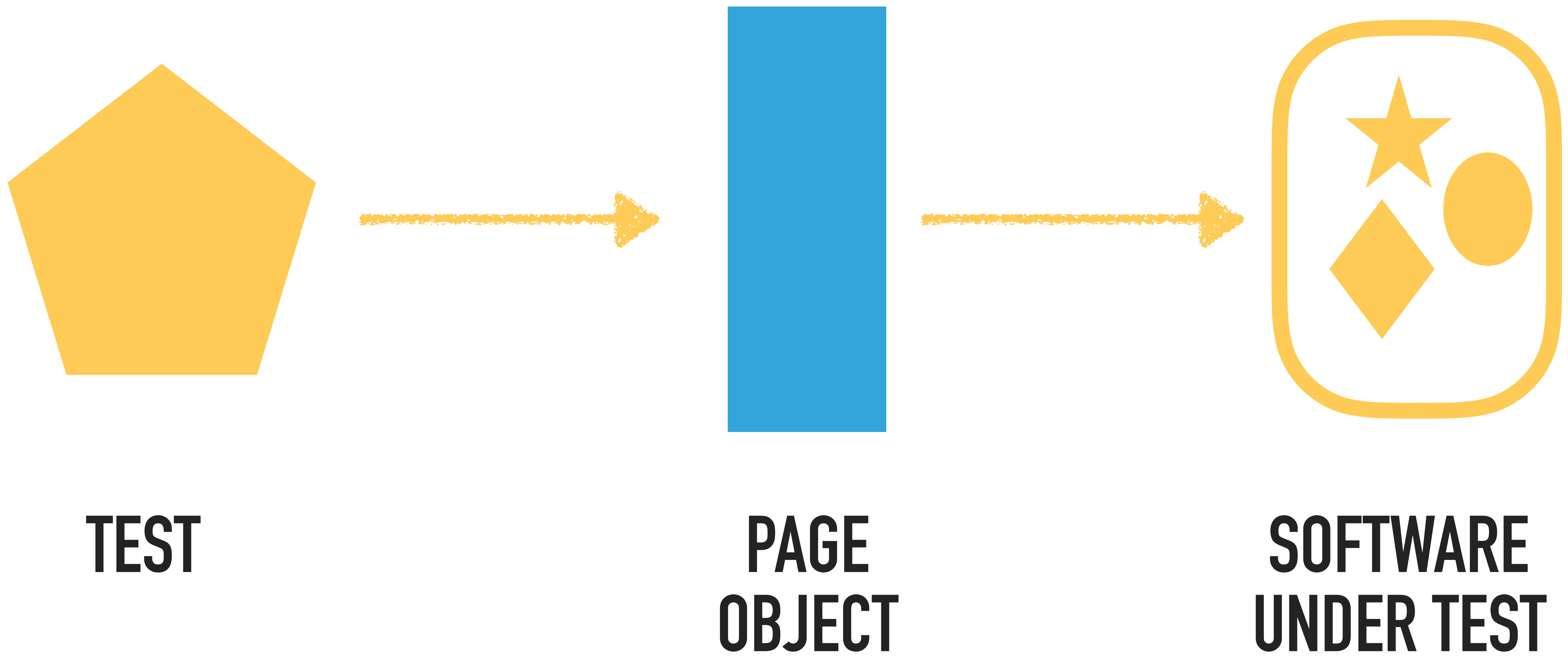
SOFTWARE UNDER TEST

@daveliddament

REDUCE COUPLING WITH PAGE OBJECT



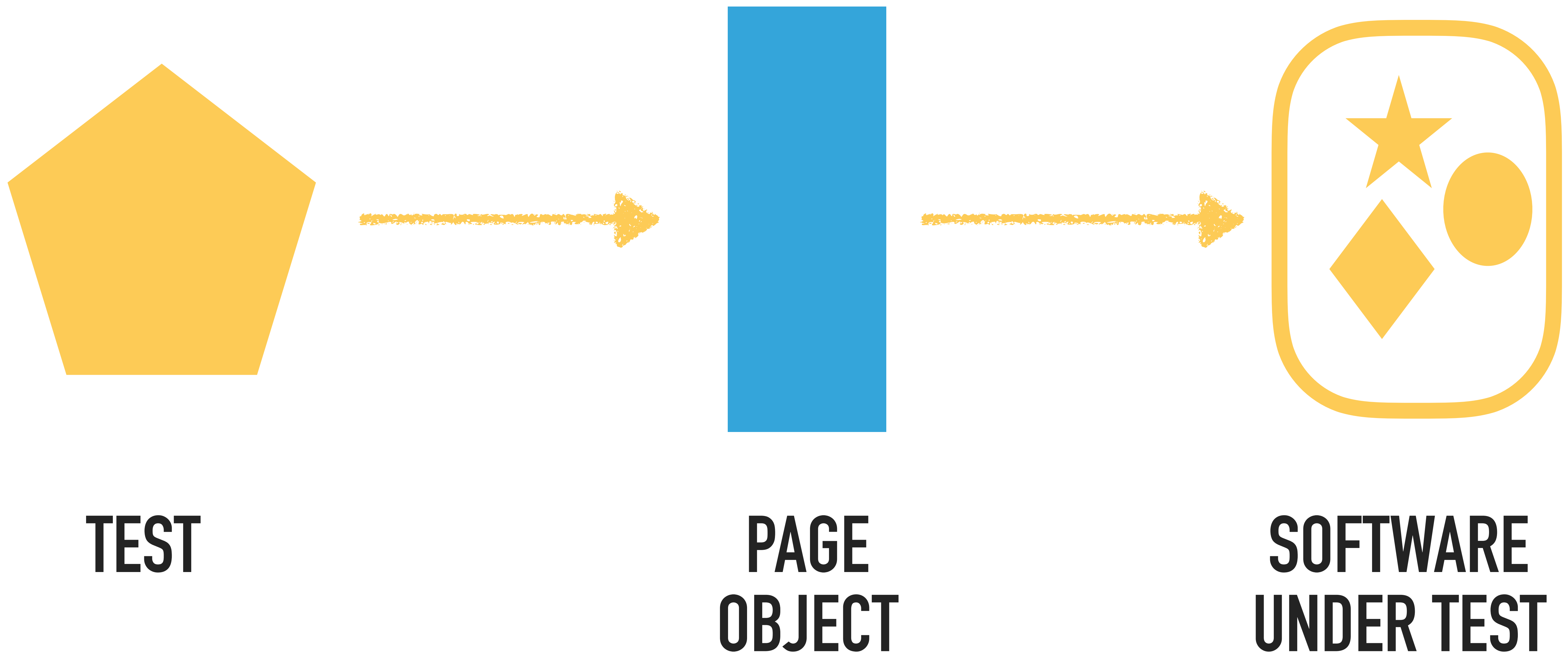
REDUCE COUPLING WITH PAGE OBJECT



REDUCE COUPLING WITH PAGE OBJECT

login ($username, $password)

answerQuestion ($answer)



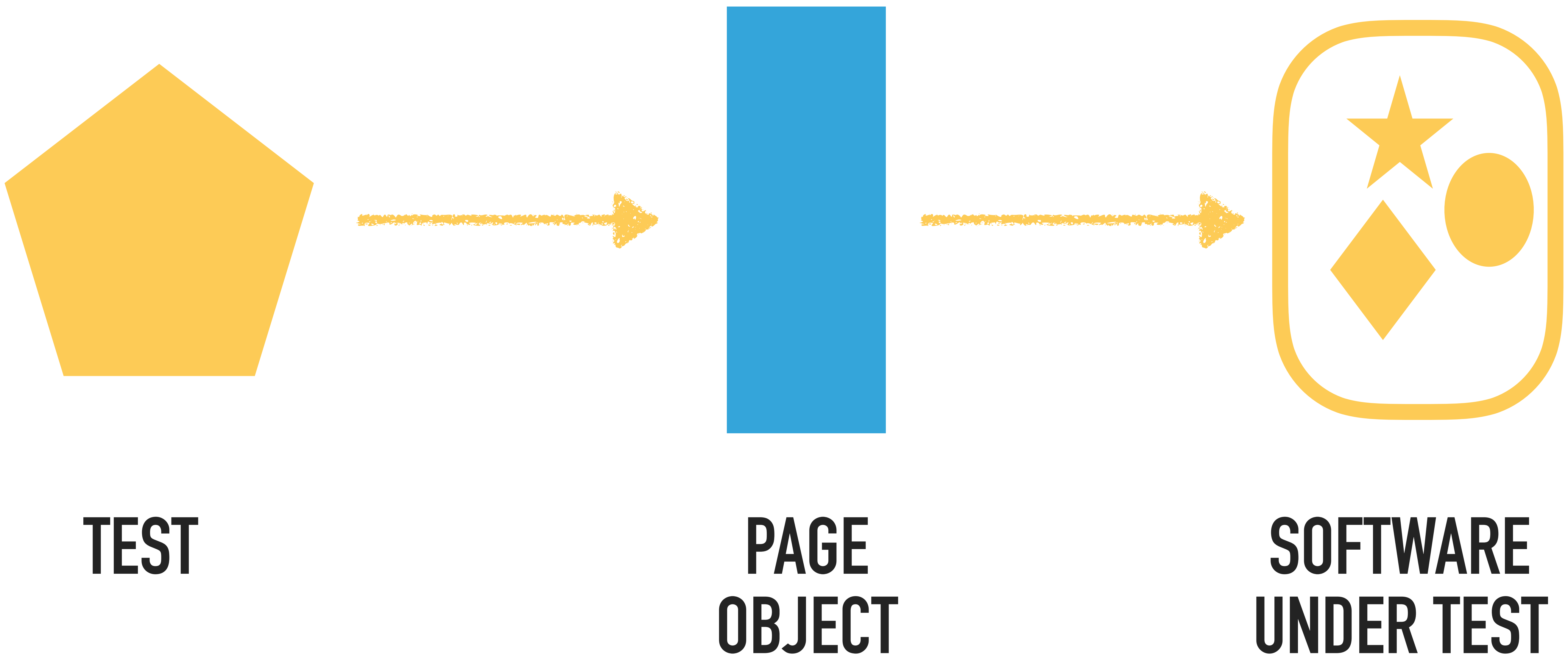
REDUCE COUPLING WITH PAGE OBJECT

login ($username, $password)

answerQuestion ($answer)

findElementByName ($name)

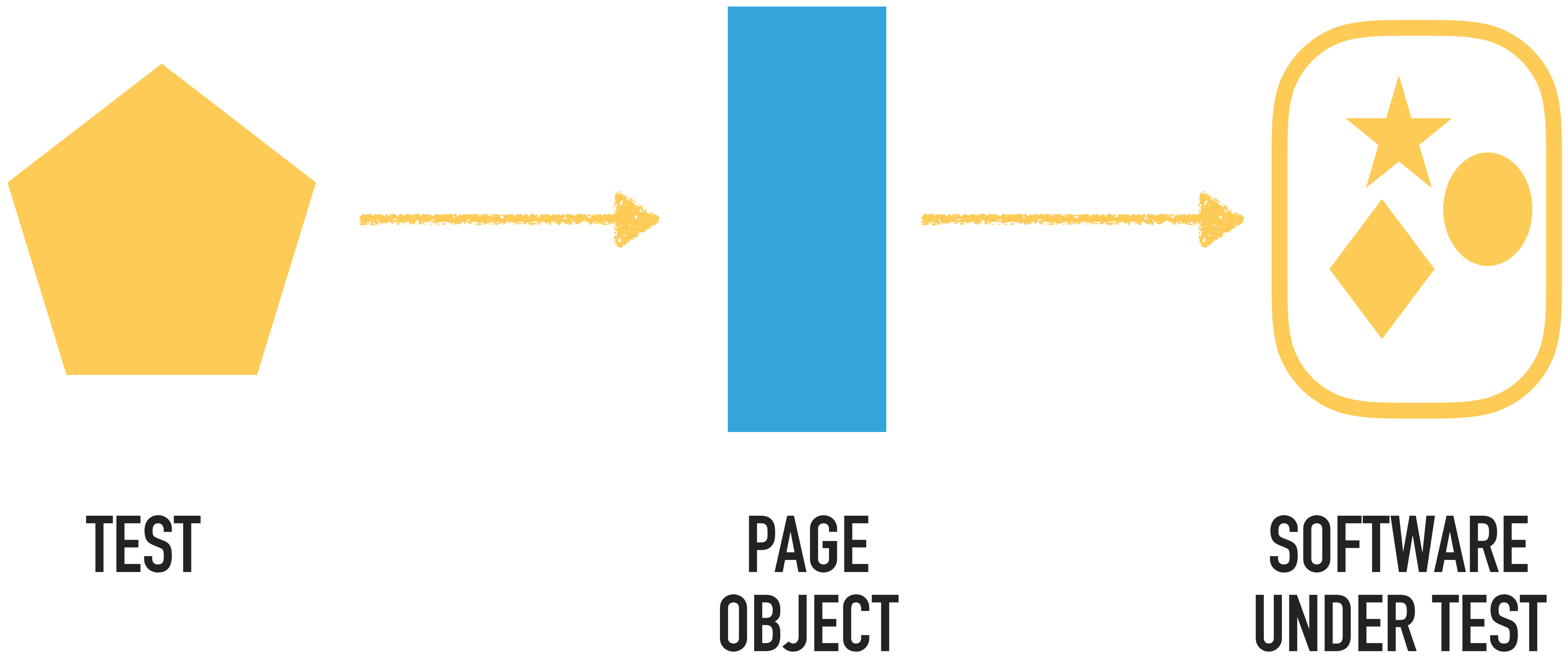
click ()



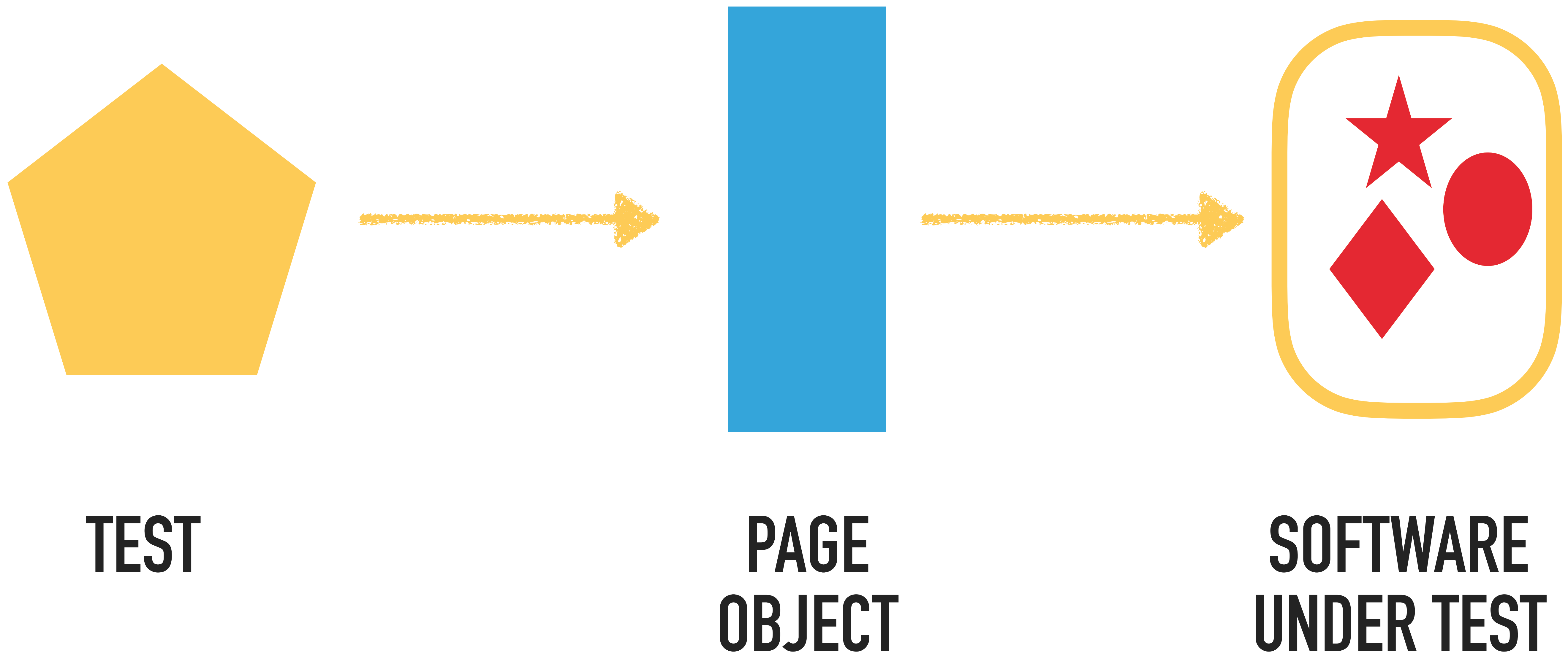
A PAGE OBJECT CAN…

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

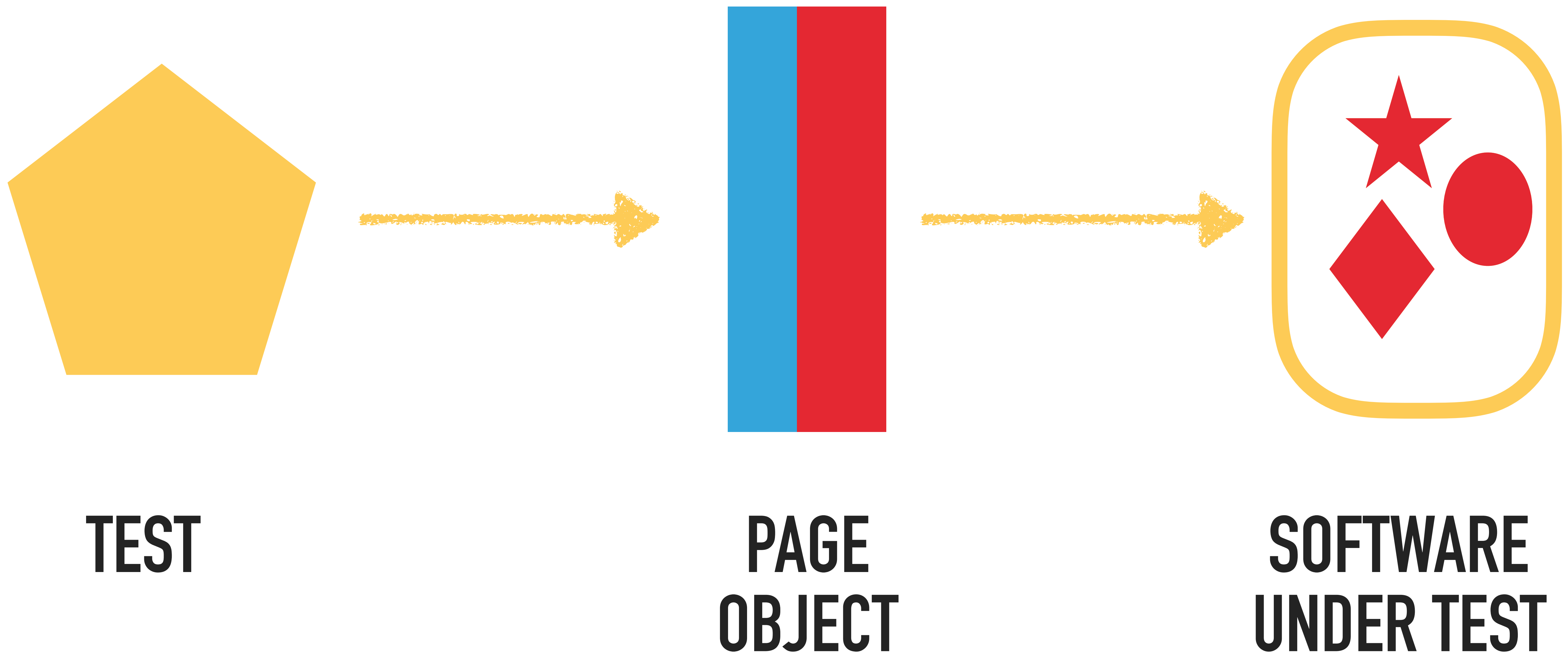
REDUCE COUPLING WITH PAGE OBJECT



REDUCE COUPLING WITH PAGE OBJECT



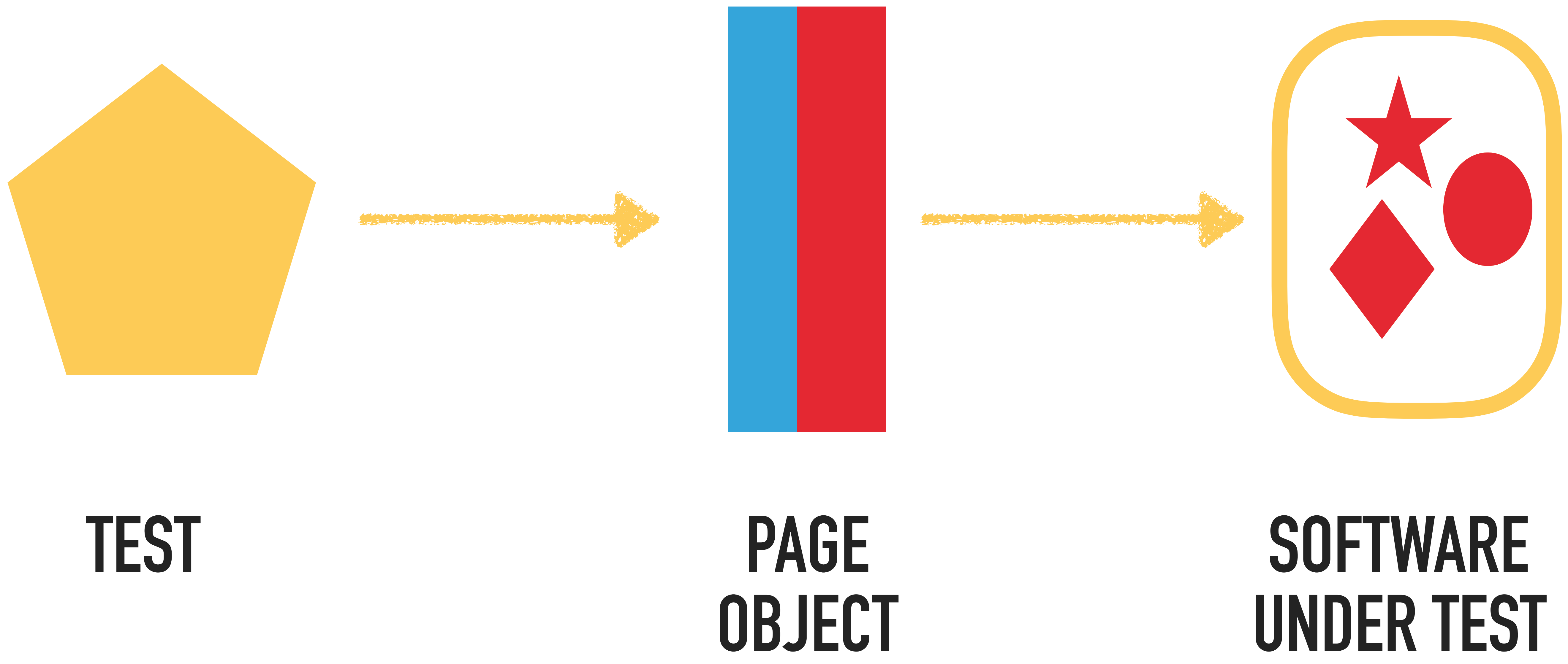
REDUCE COUPLING WITH PAGE OBJECT



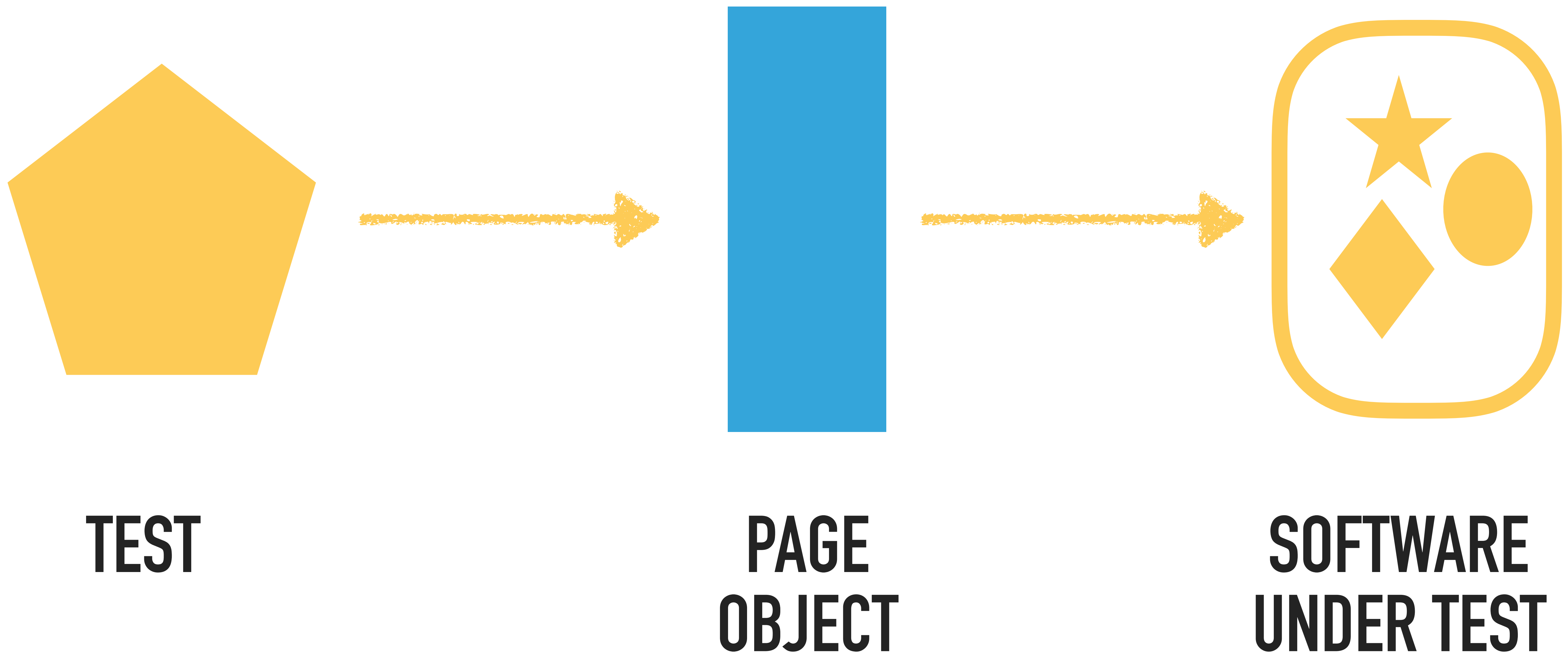
REDUCE COUPLING WITH PAGE OBJECT

login ($username, $password)

answerQuestion ($answer)

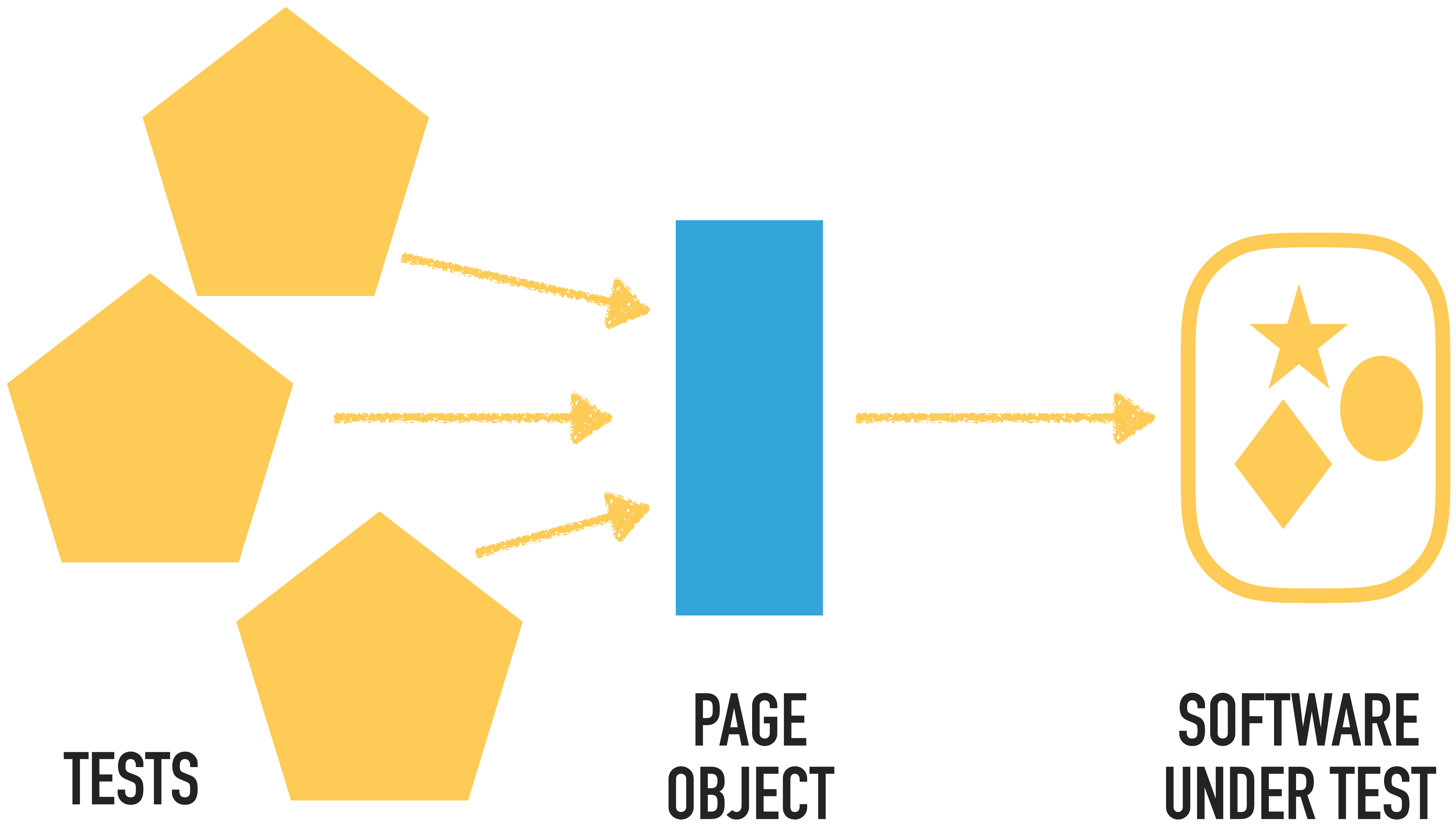


REDUCE COUPLING WITH PAGE OBJECT



STORY 1

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 …
```

THINGS I WANTED TO TEST…

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

A TINY CHANGE REQUEST….

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

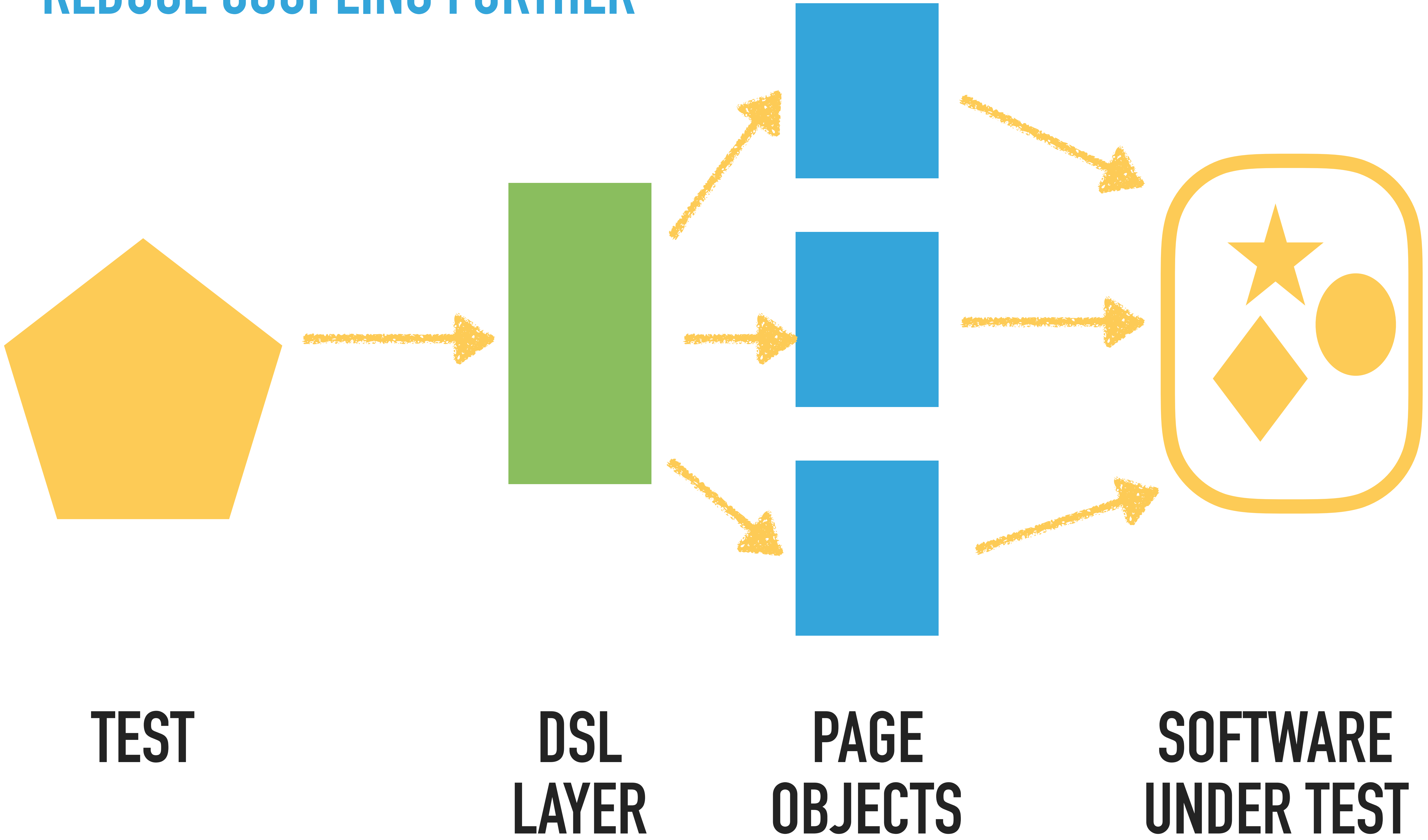
THE TESTS WILL BREAK

```
$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 …
```

```
................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.  63 / 444 ( 14%)
............................................................... 126 / 444 ( 28%)
.........................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF...... 189 / 444 ( 42%)
FFFFFFFFFFFF................................................... 252 / 444 ( 56%)
...................FF....FFFF.......FFFFFFFFFFFFFF............. 315 / 444 ( 70%)
..................................FFFFFFFFFFFFFFFFFFFFFFFFFFF.. 378 / 444 ( 85%)
FFFFFFFFFFFFFF........FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 441 / 444 ( 99%)
...

Time: 20 minutes 54 seconds, Memory: 24.75MB

 There were lots of failures:
```

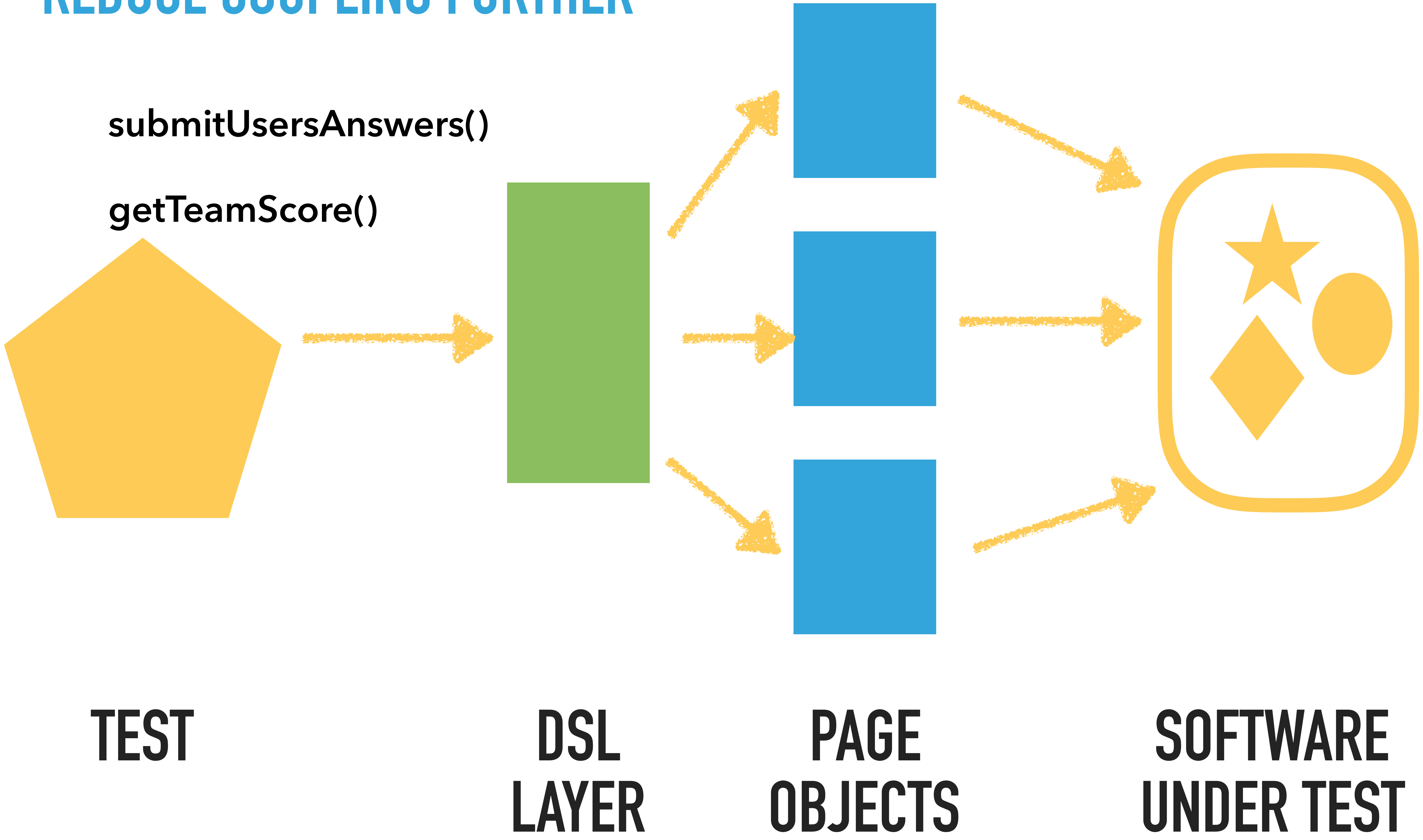
😞

REDUCE COUPLING FURTHER



STORY 1

REDUCE COUPLING FURTHER



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);
```

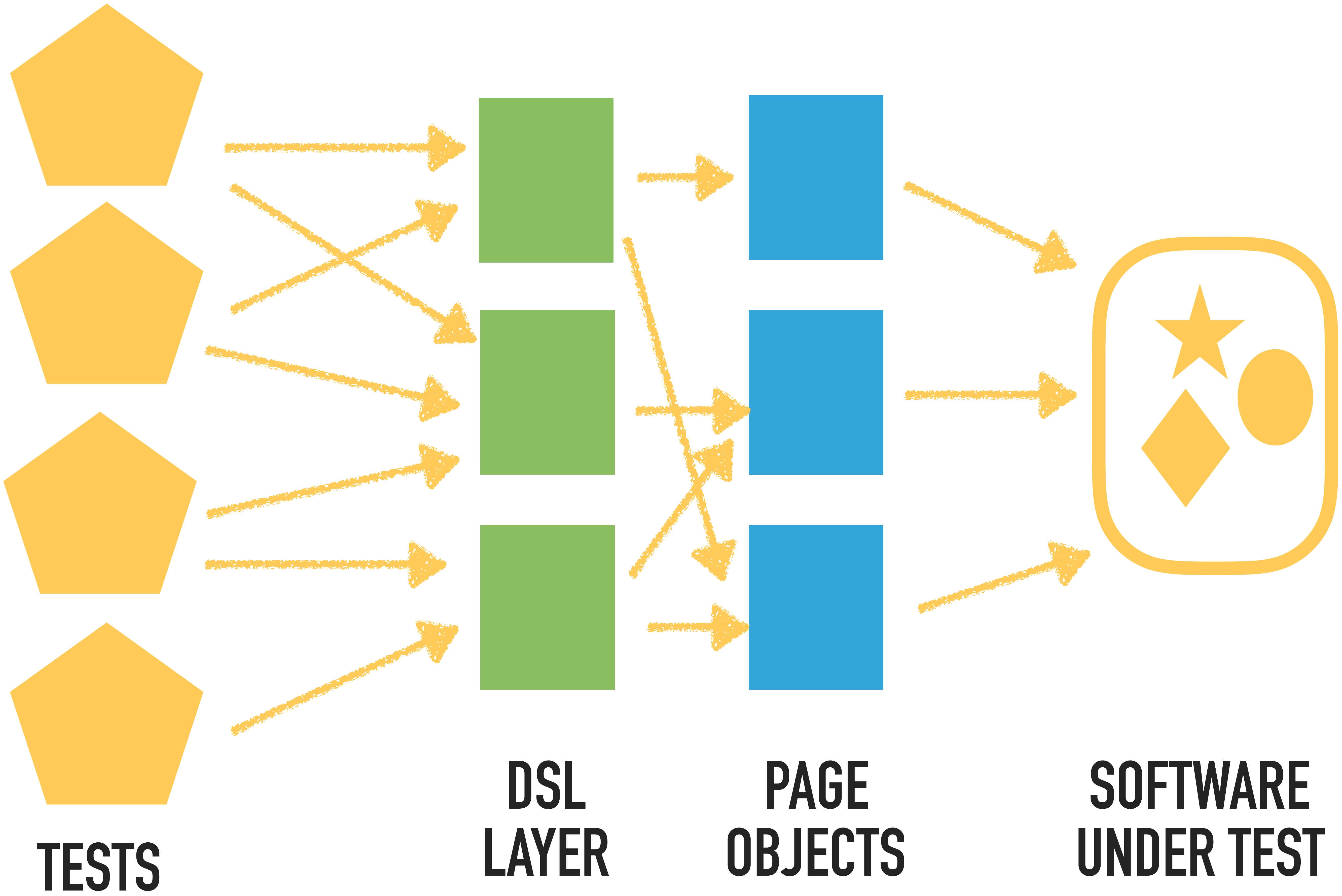
THINGS I WANTED TO TEST…

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

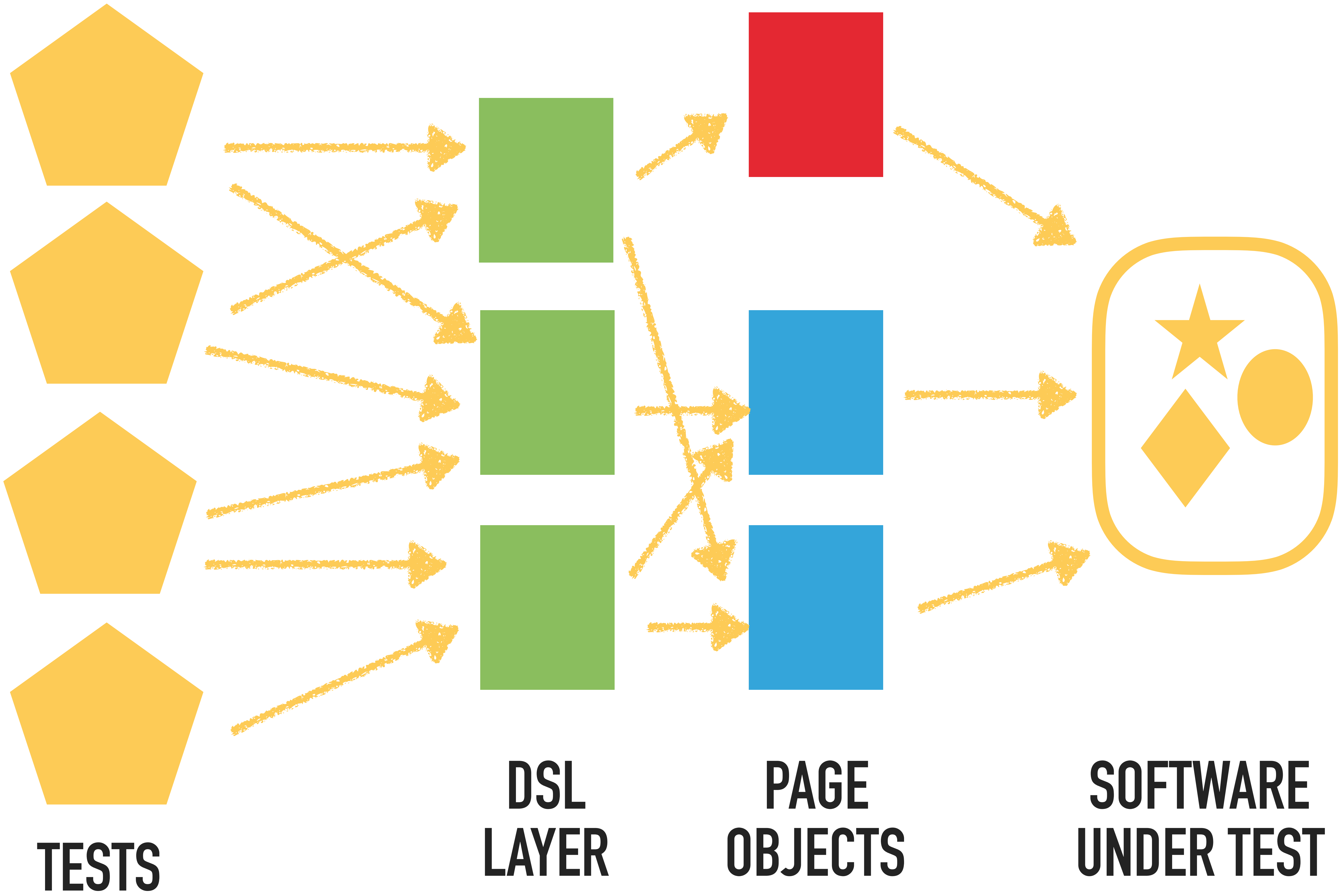
TEST LOOK A BIT MORE LIKE THIS

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

submitUsersAnswers($bob, self::QUIZ_1,

  ['engagement' => 'a',  'enjoyment' => 'b',  … etc … ]);

$score = getTeamScore($apple);

assertEquals(7, $score);
```

STORY 1



STORY 1



THE MORAL OF 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.

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.

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!

DECOUPLED TESTS REDUCE THE DEVELOPMENT AND MAINTENANCE COSTS OF THE TEST SUITE.

BUT WHAT IF . . .

BUT WHAT IF …

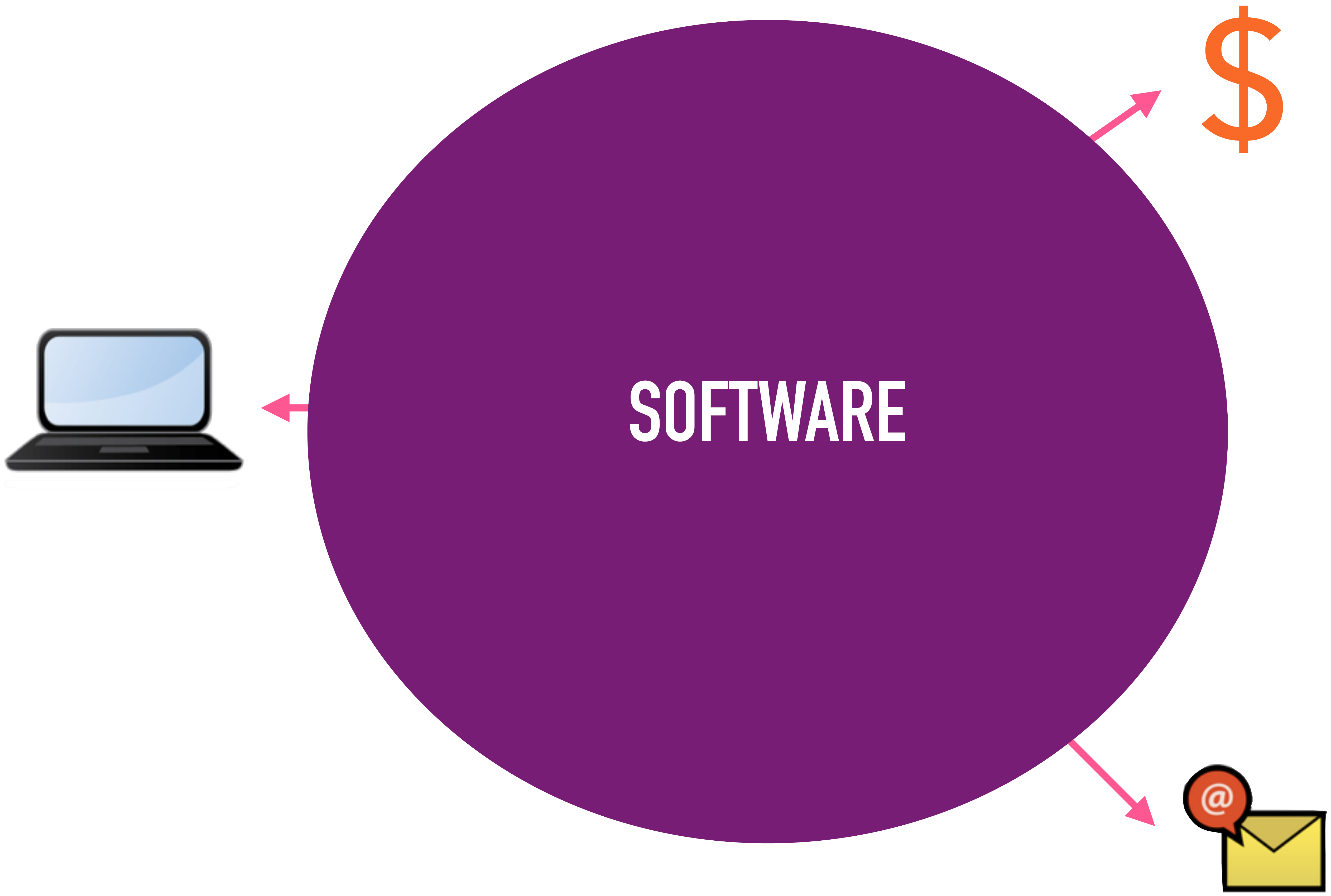
We replace the entire website with an app?

ALSO …

This feels like a lot of effort.

#2

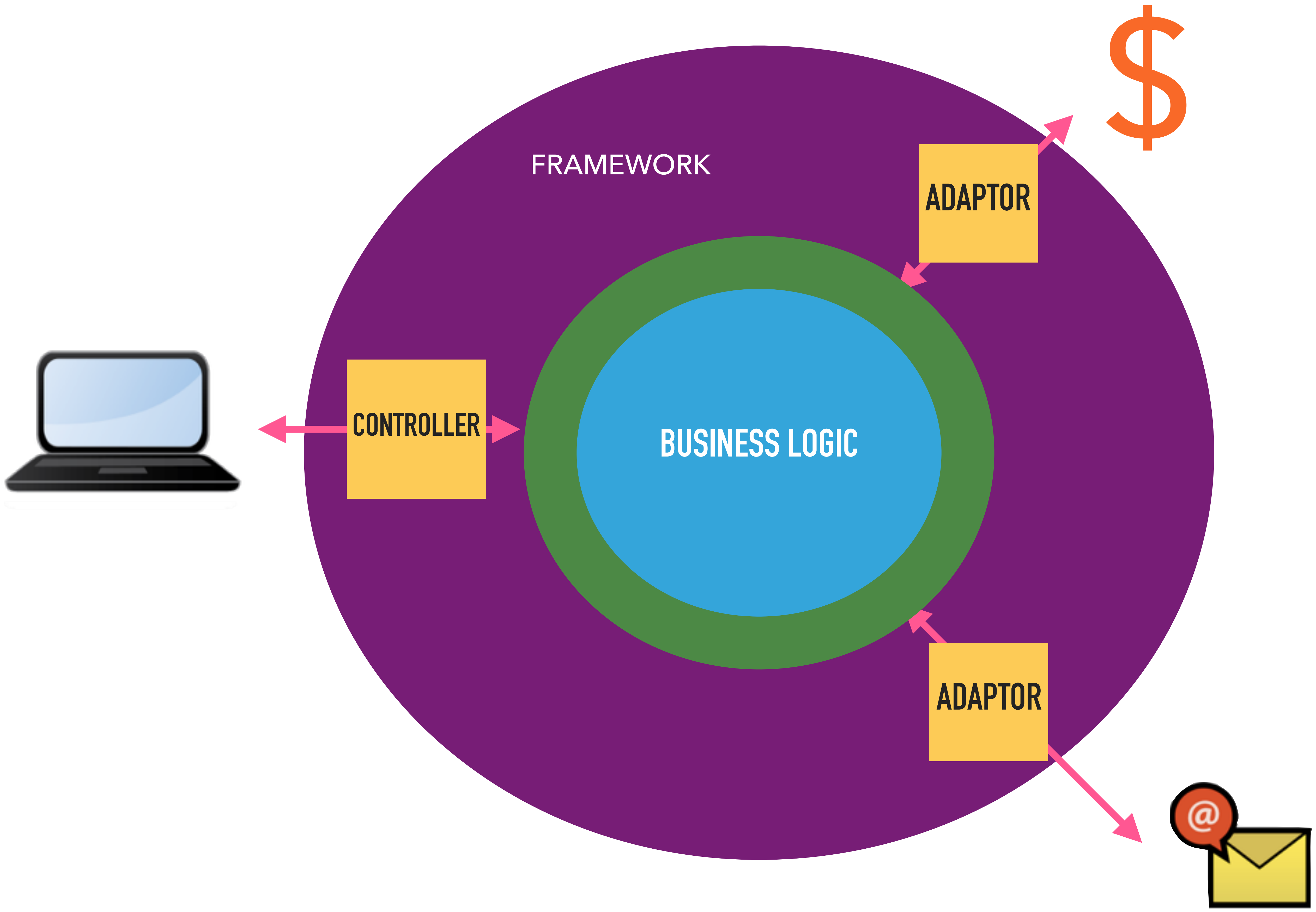
STORY 2



THERE MUST BE A BETTER WAY…

- Layered architecture
- Hexagonal architecture

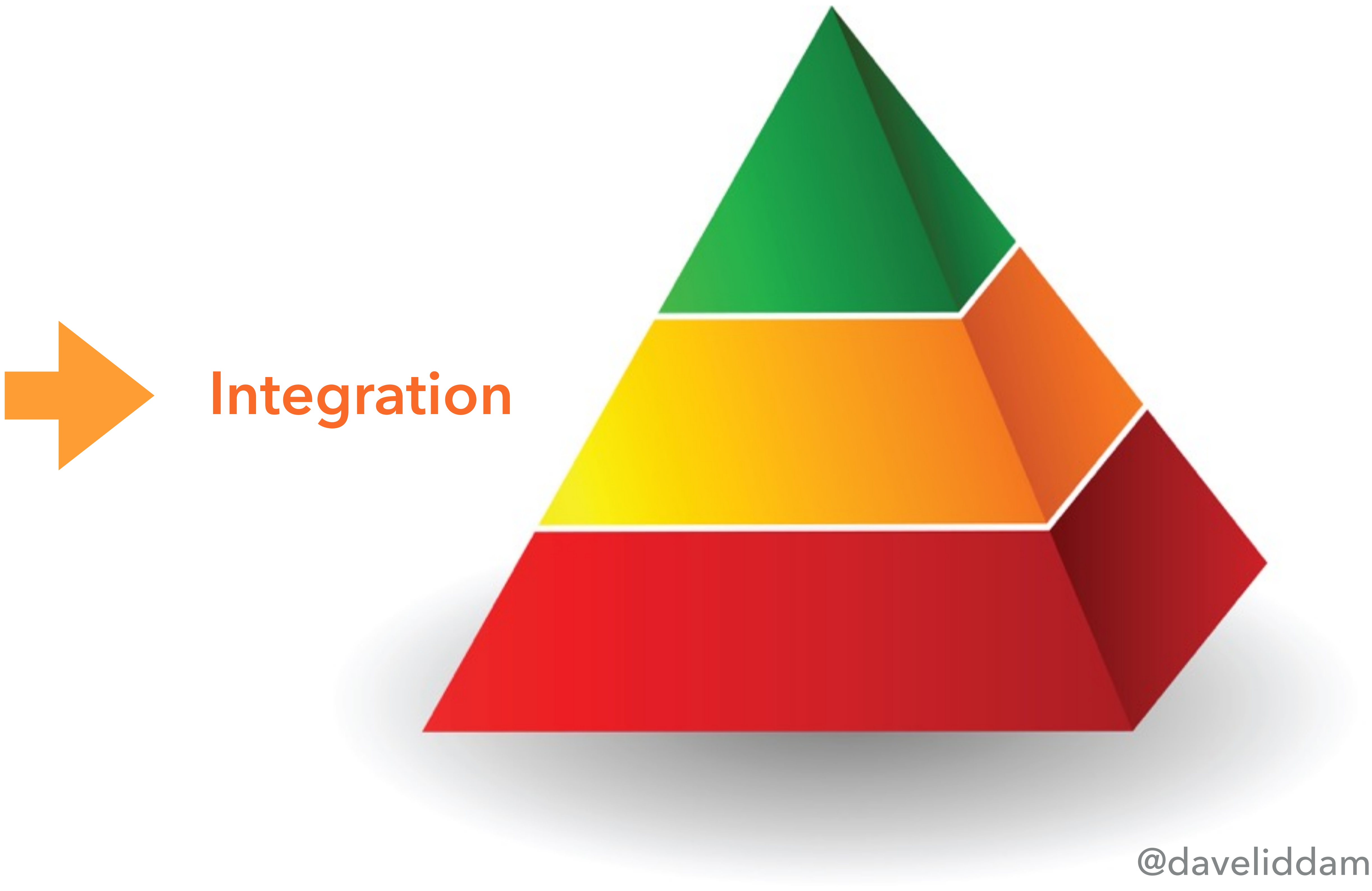
STORY 2



SERVICE LAYER

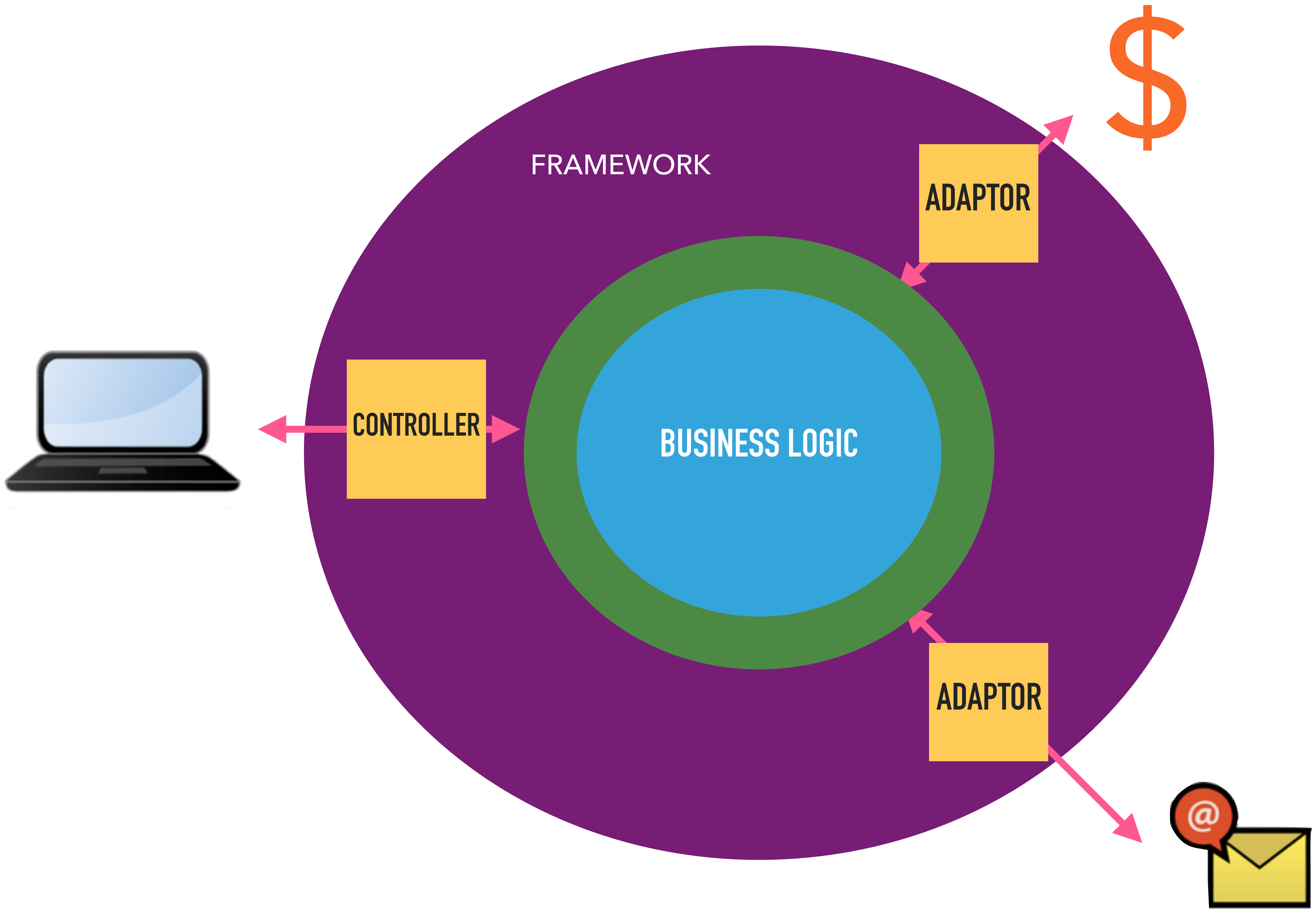
```
interface AnswerSubmissionService
{

  public function submitUsersAnswers(
        User $user,
        int $quizId,
        array $answers
  ): void;

}
```

STORY 2

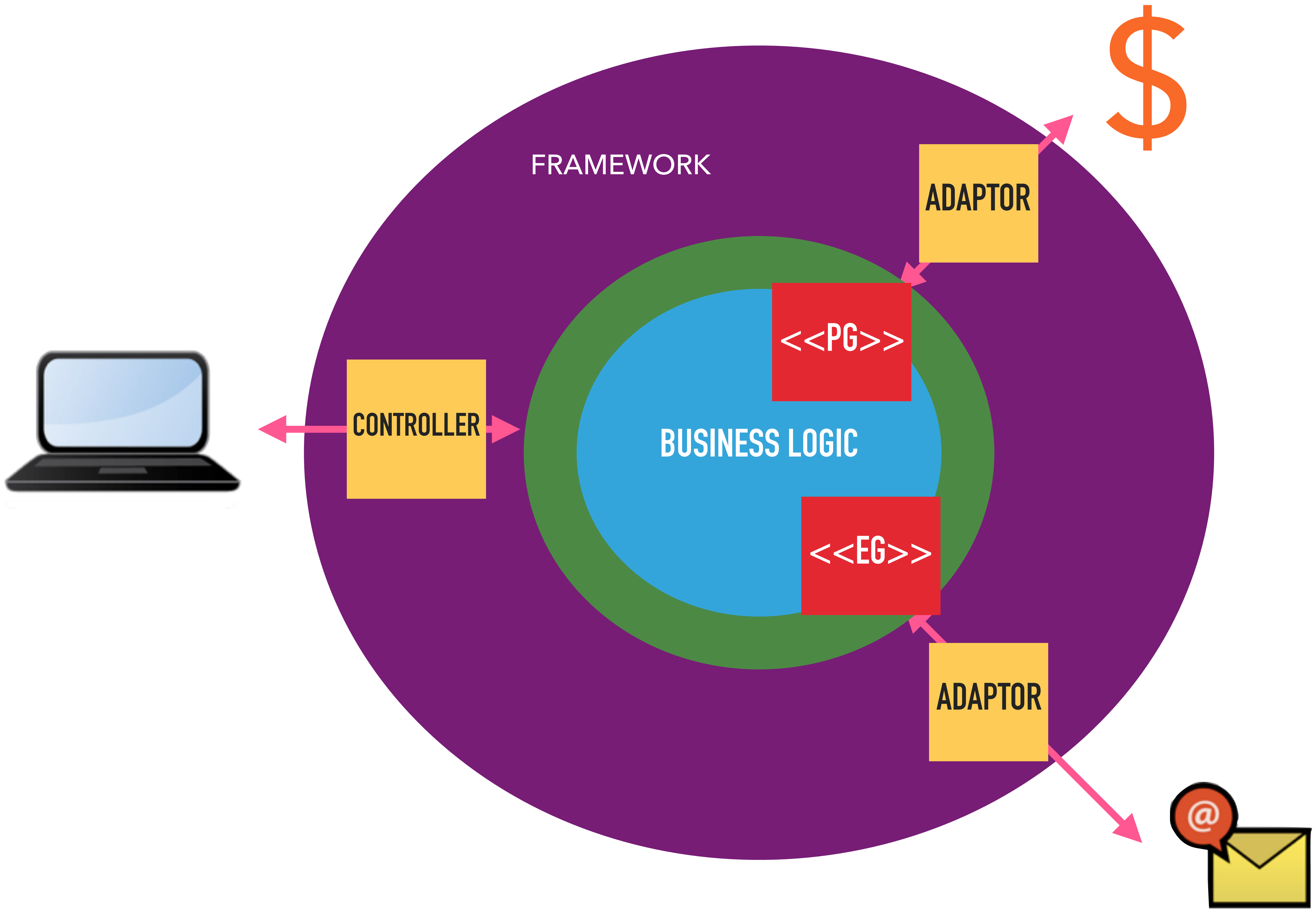


@daveliddament

STORY 2



STORY 2



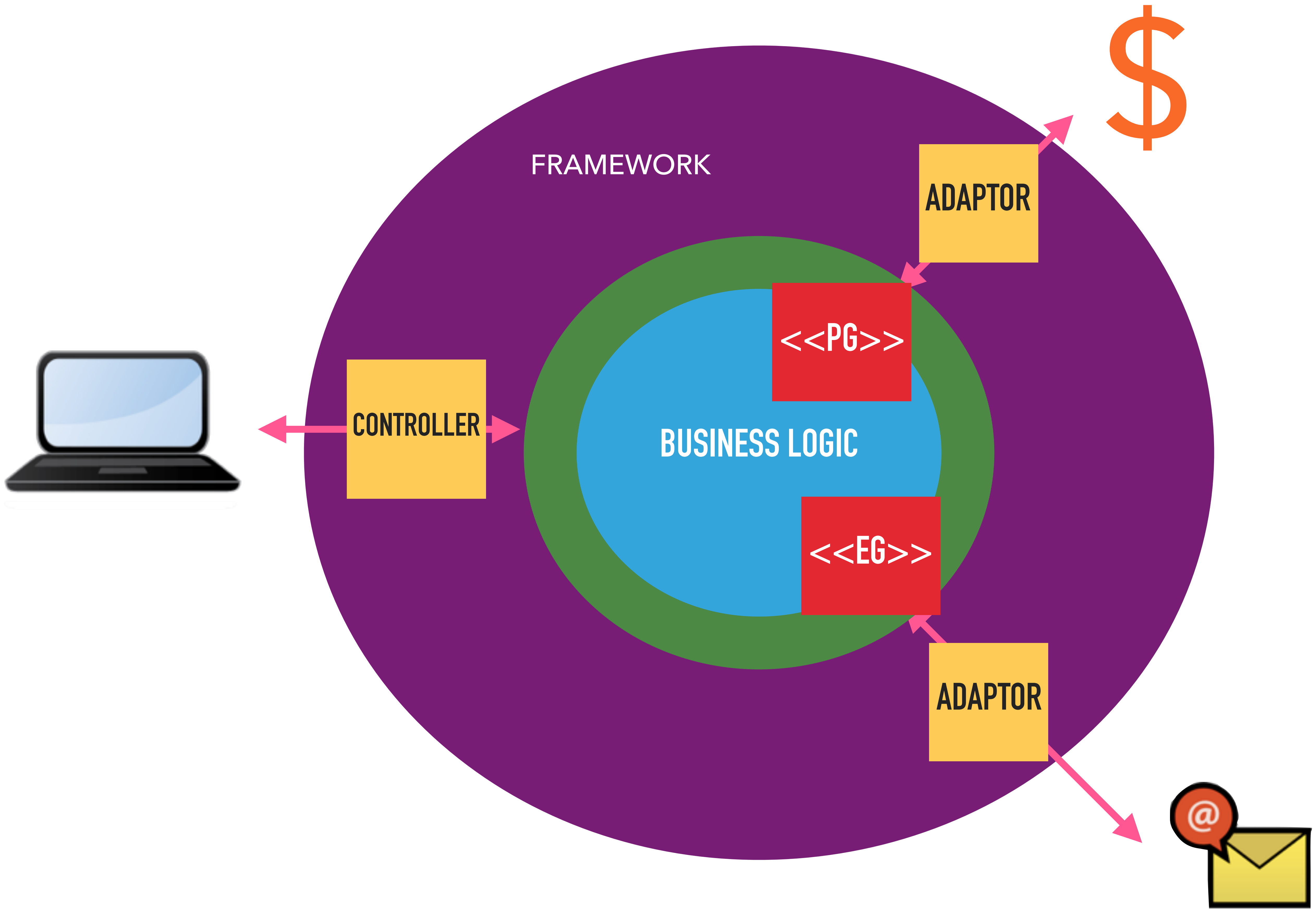
STORY 2

EMAIL GATEWAY

EMAIL GATEWAY

```
interface EmailGateway
{

  /**
   * Sends an email
   */
  public function sendEmail(
      $to,
      $from,
      $subject,
      $message
  ): void;

}
```

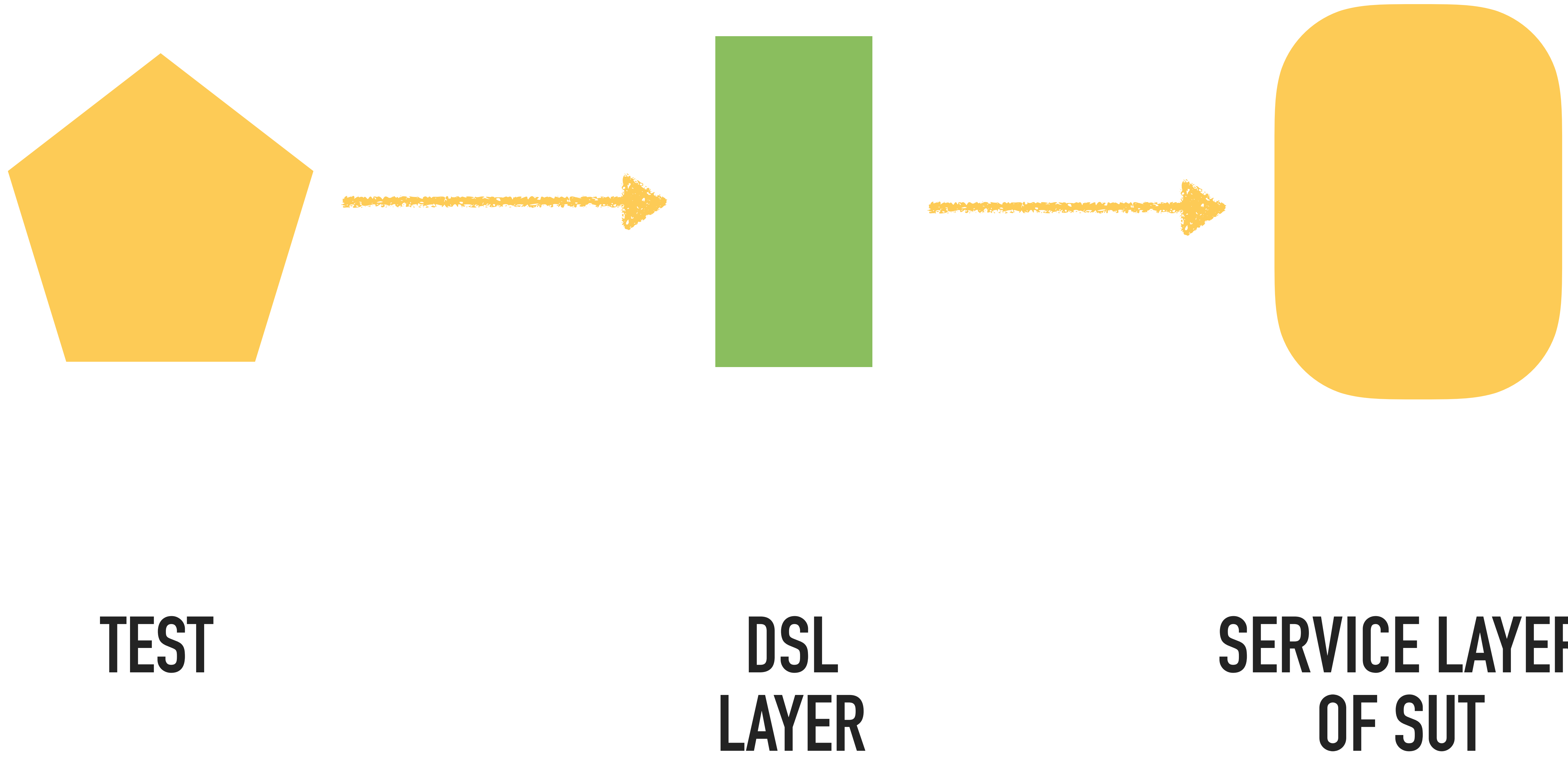
STORY 2



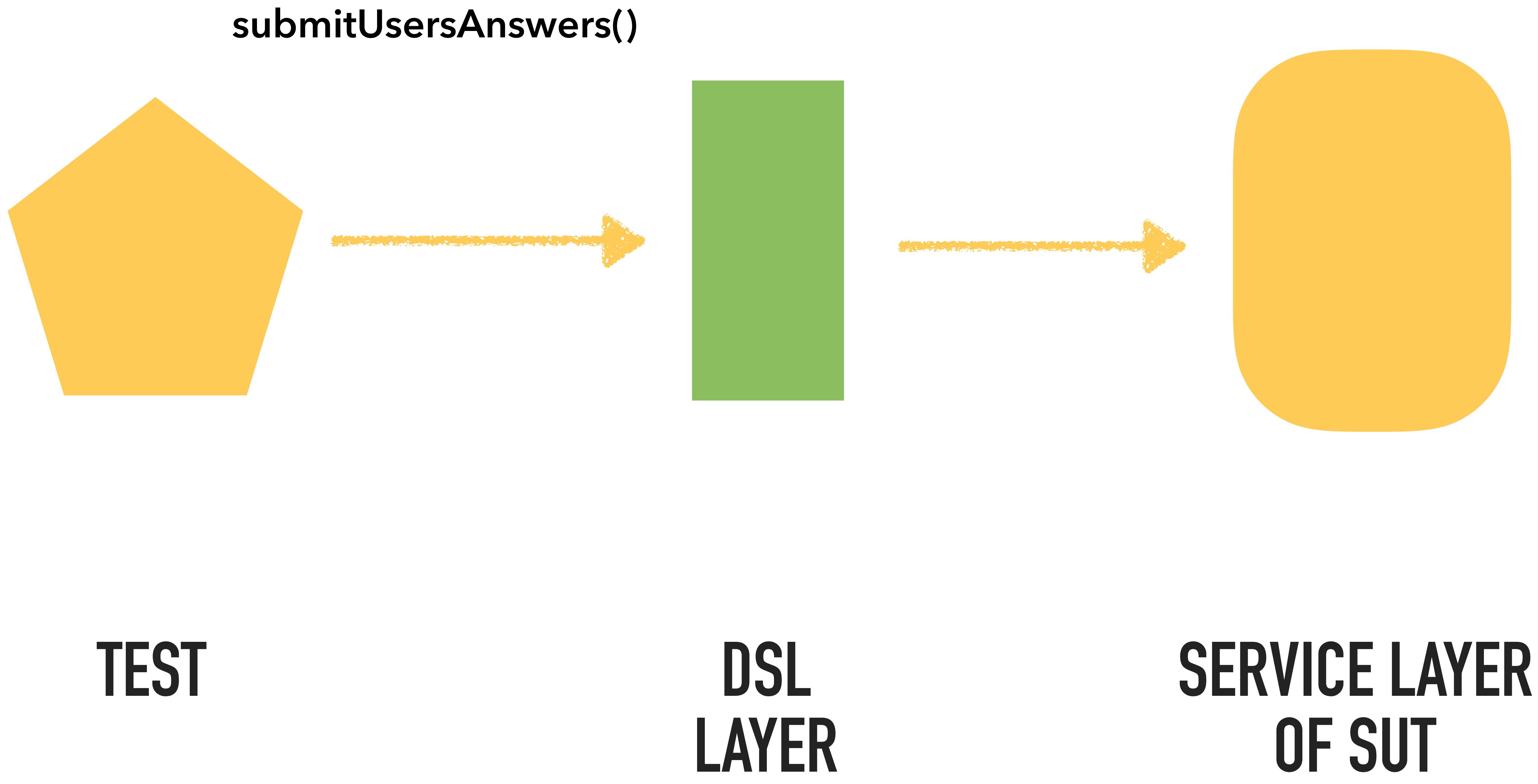
EMAIL GATEWAY TEST IMPLEMENTATION

```
EmailGatewaySpy implements EmailGateway
{

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

 public function getEmails() {
     return array of emails
 }

}
```

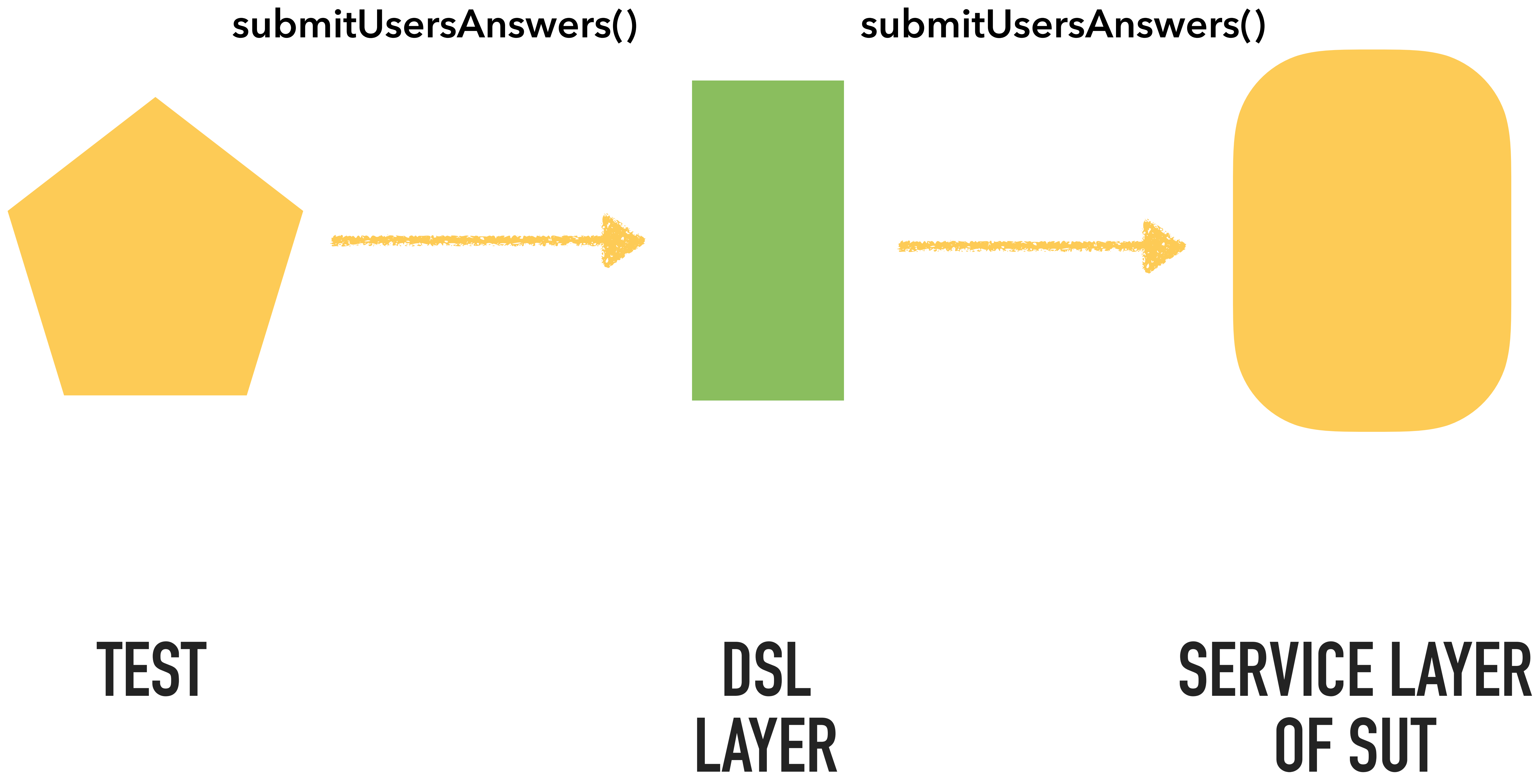
TESTING IS EASIER



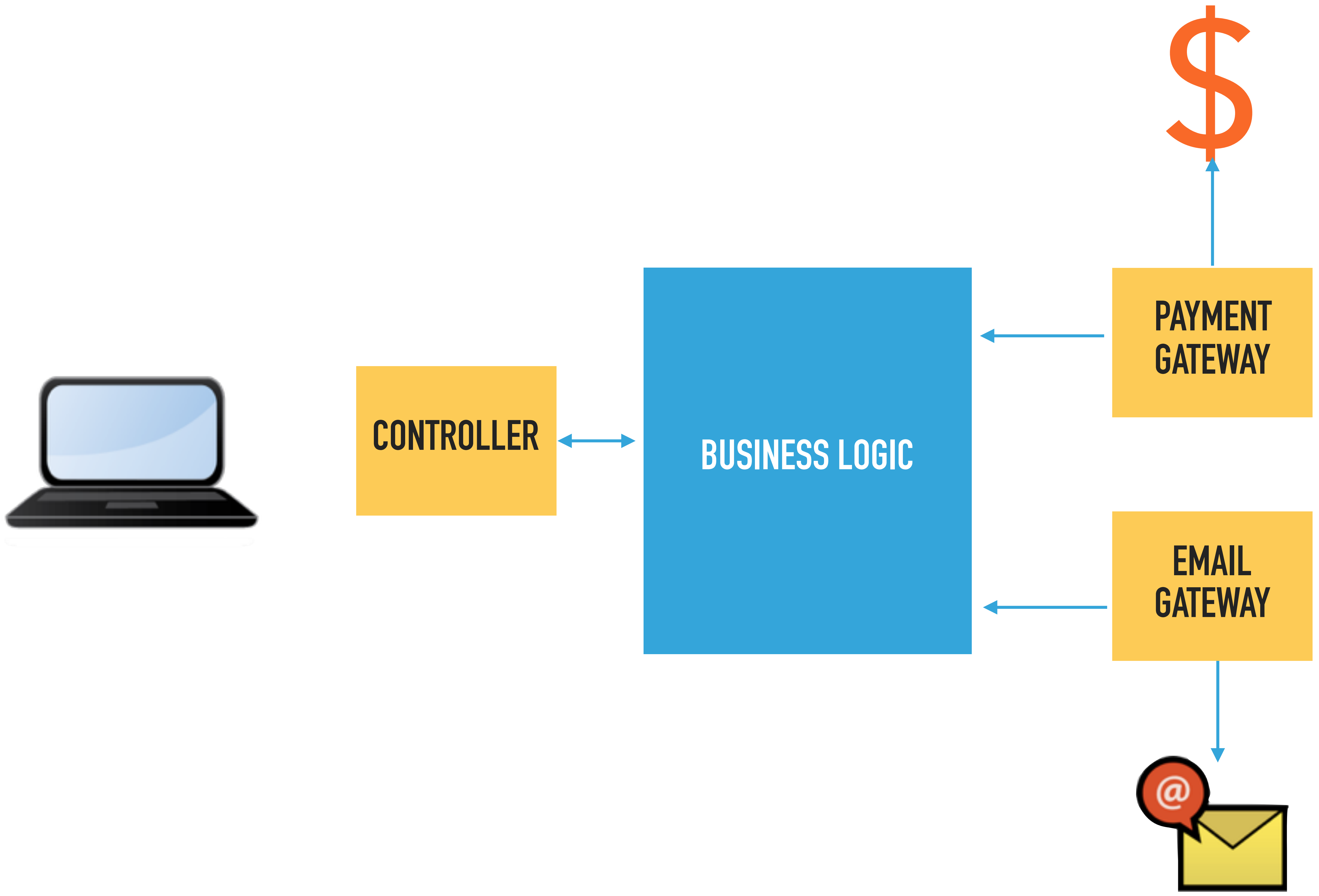
TESTING IS EASIER



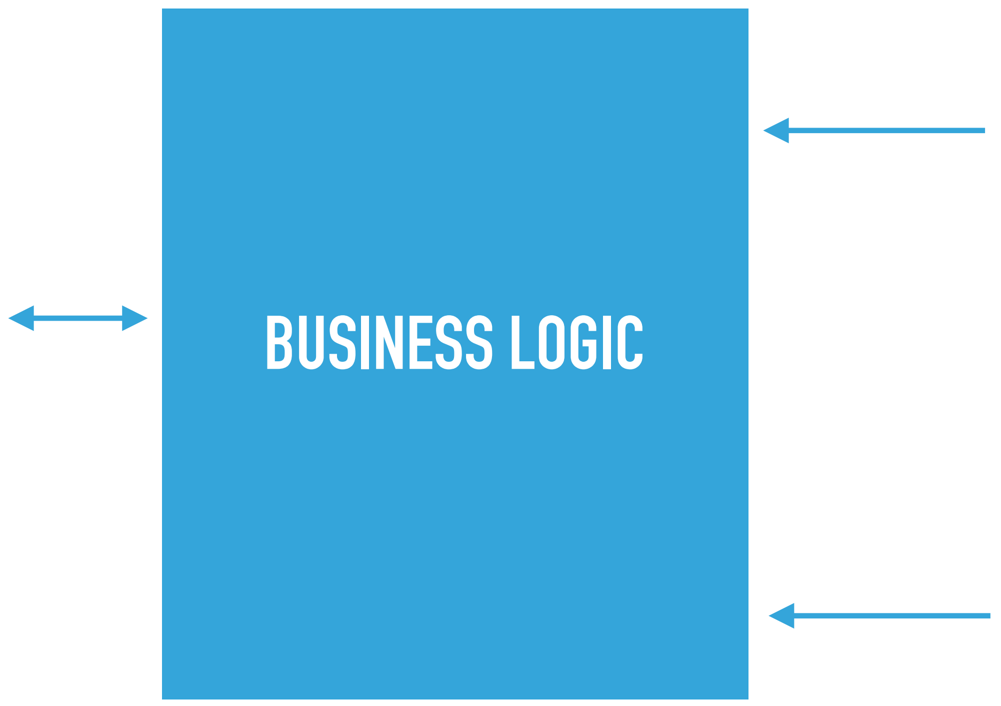
TESTING IS EASIER



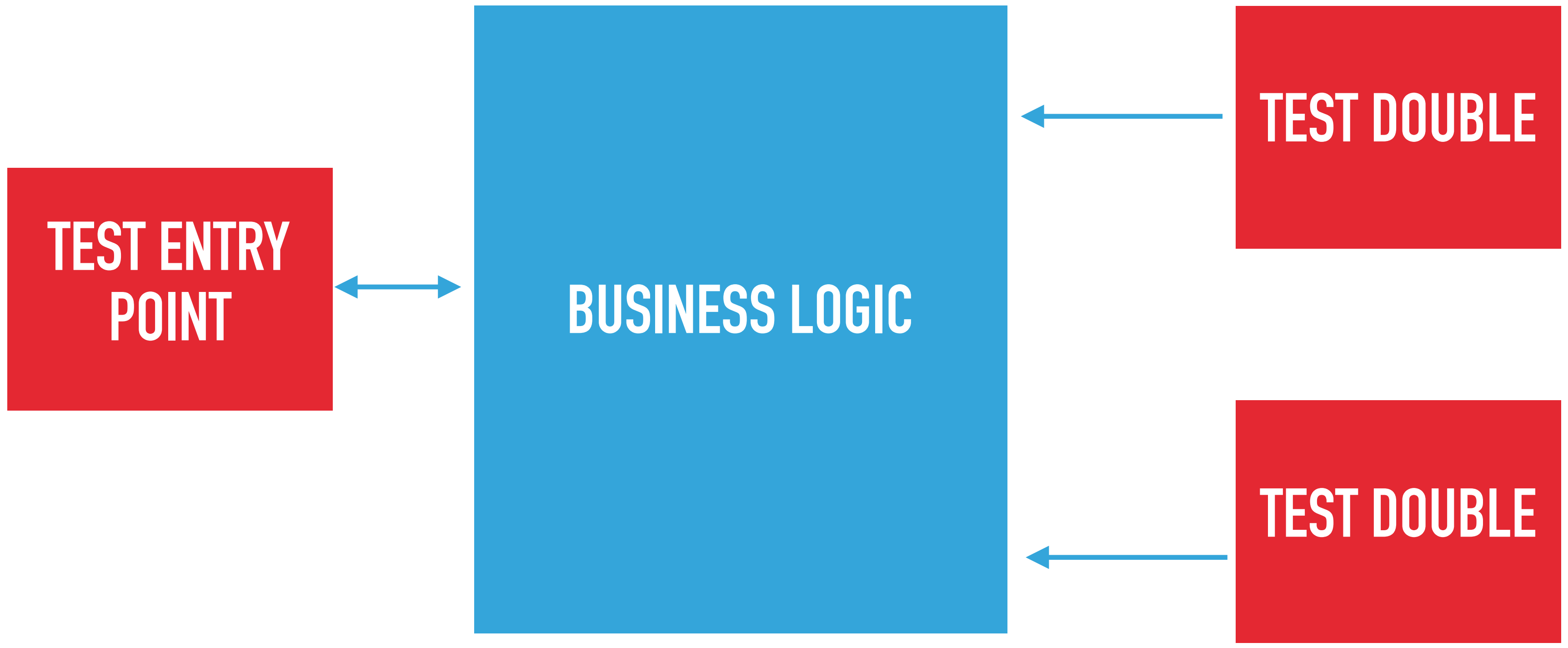
STORY 2



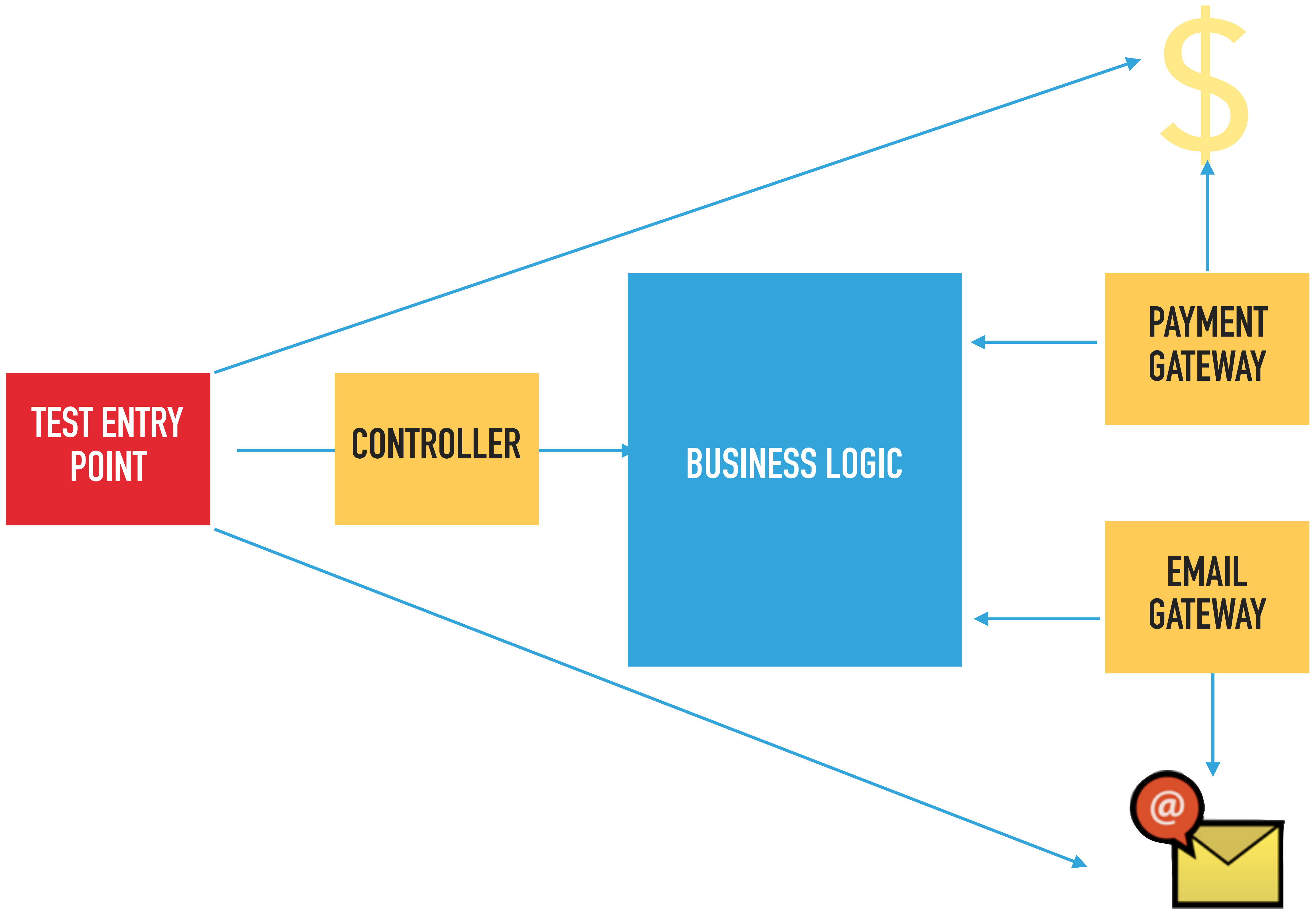
STORY 2



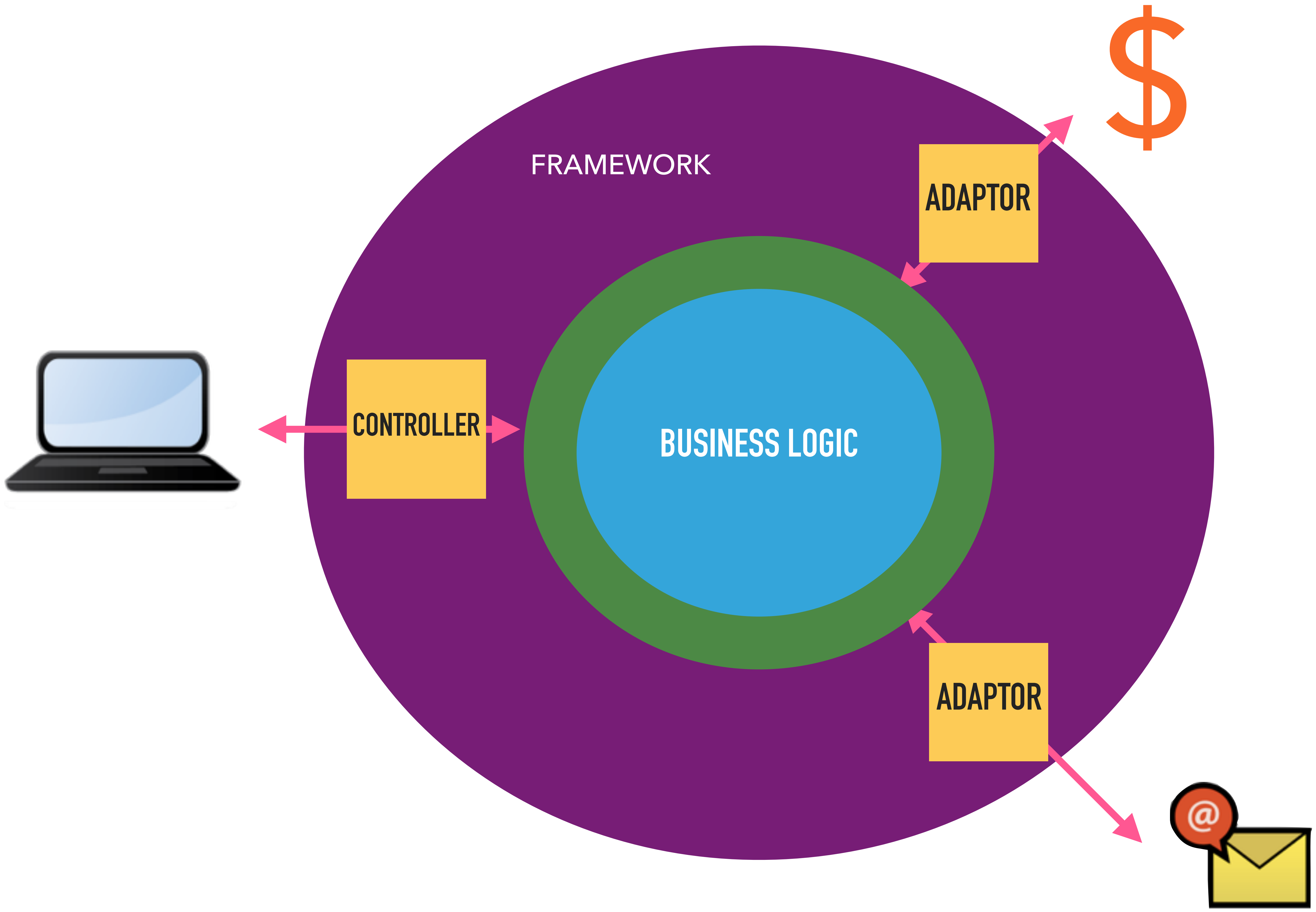
STORY 2



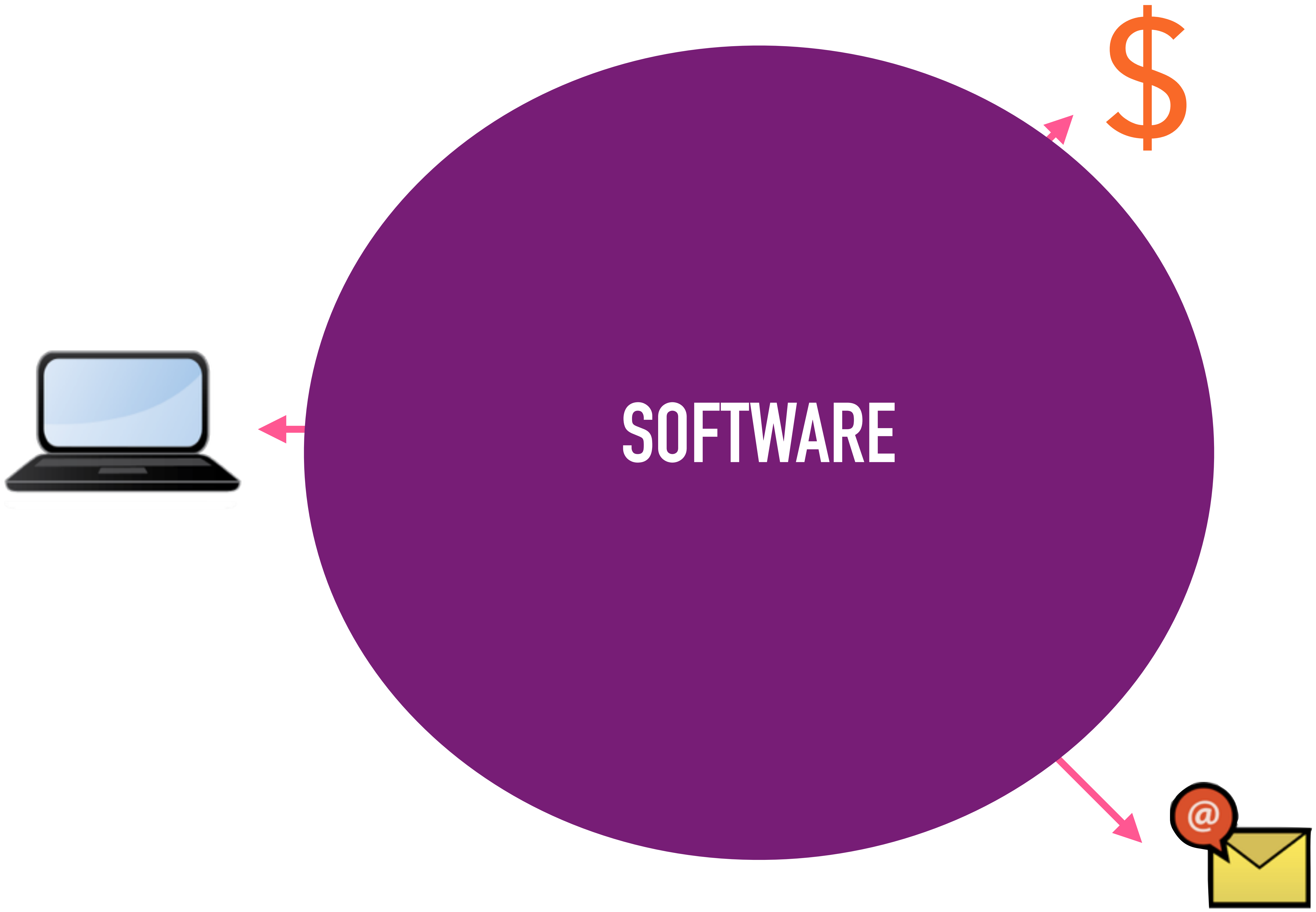
STORY 2


TEST ENTRY POINT
CONTROLLER
BUSINESS LOGIC
PAYMENT GATEWAY
EMAIL GATEWAY
$
@

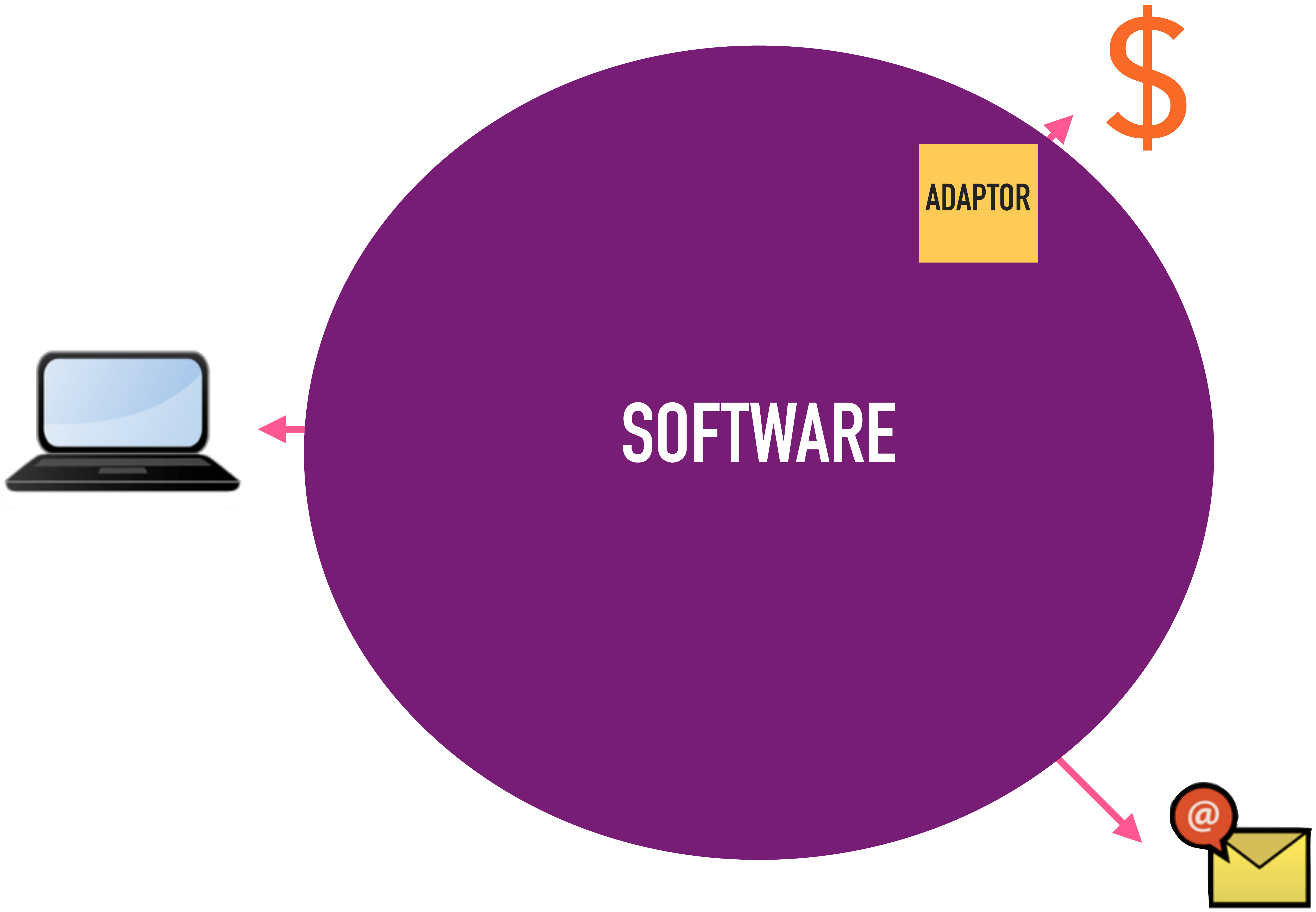
STORY 2



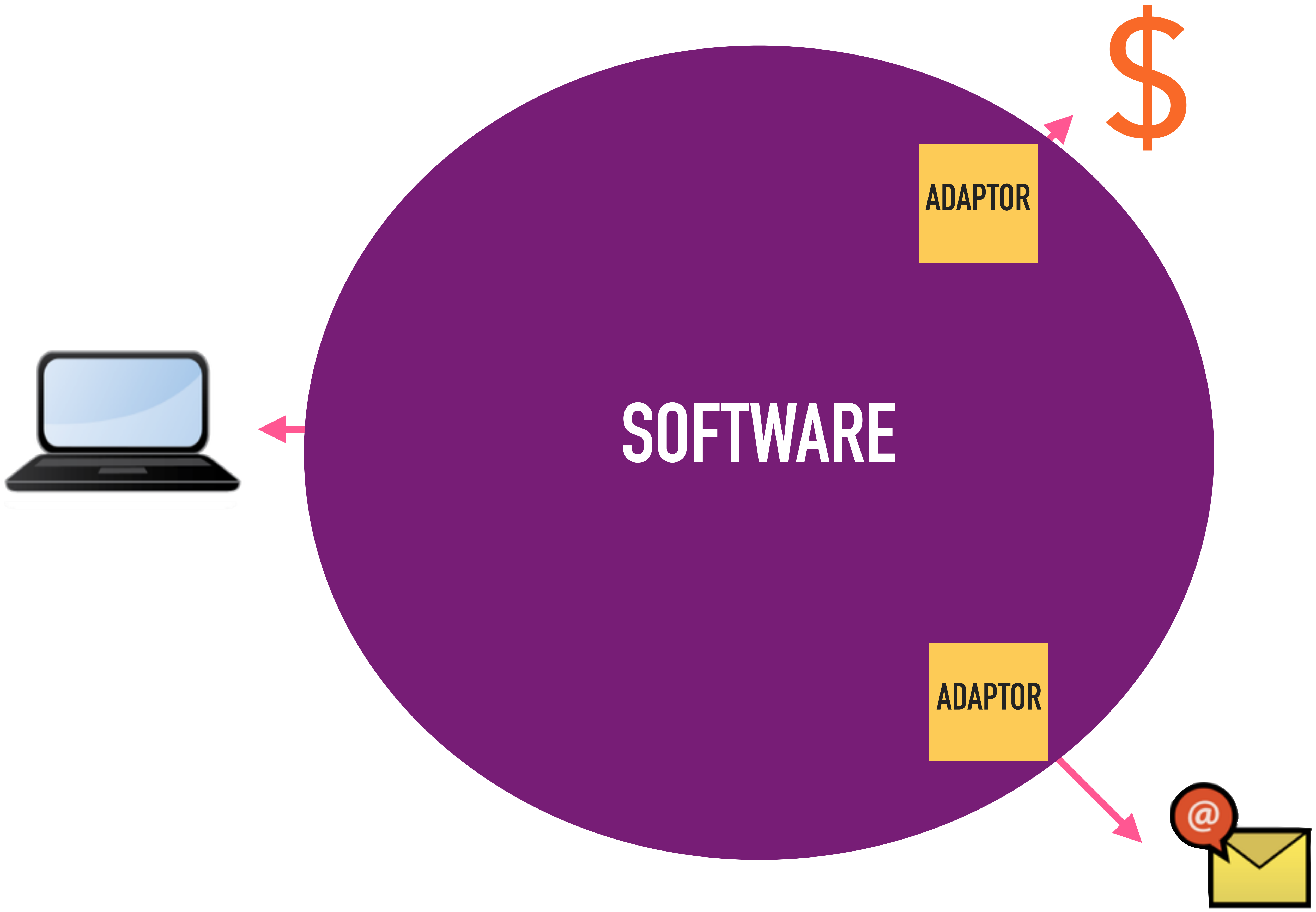
STORY 2



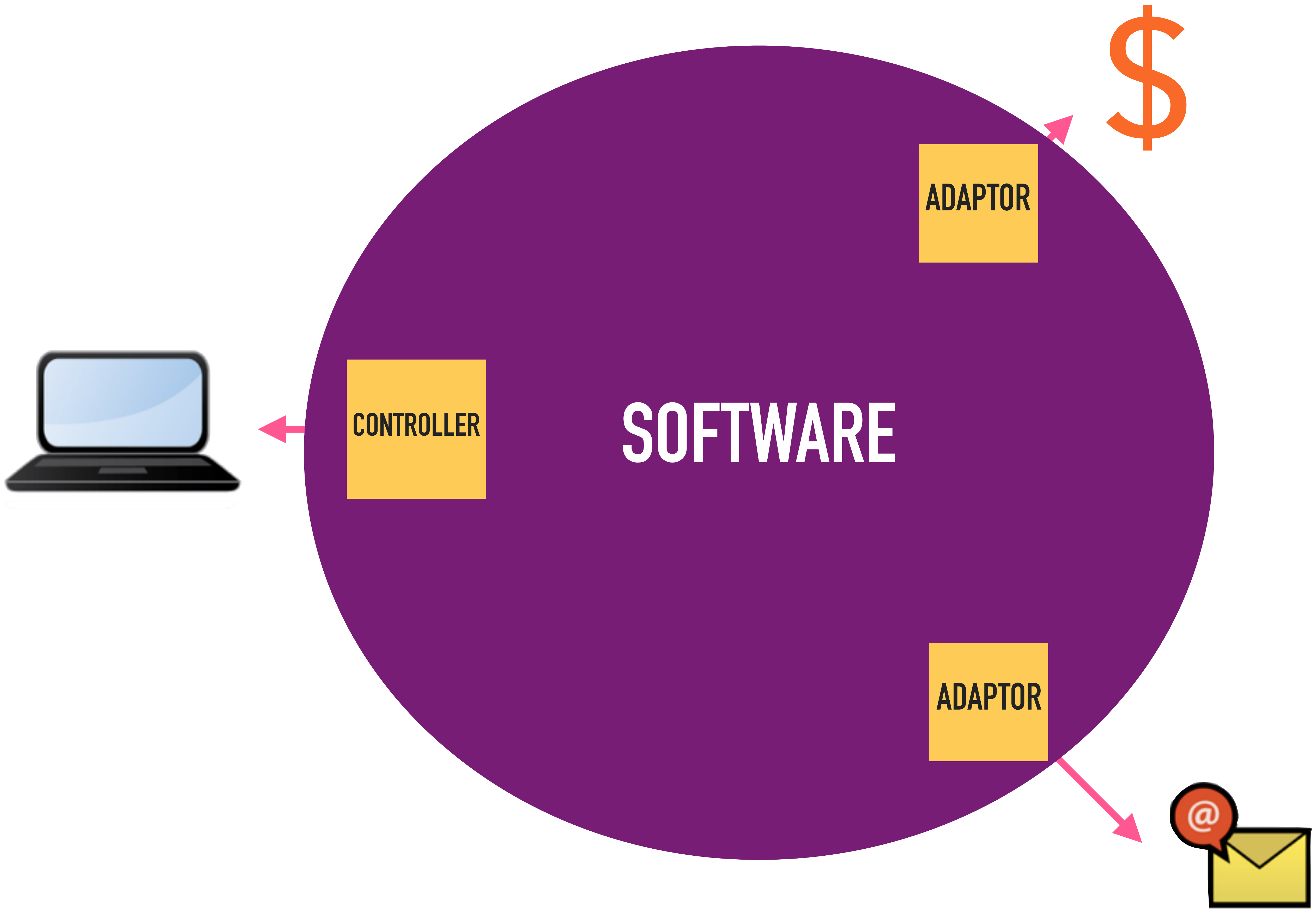
STORY 2



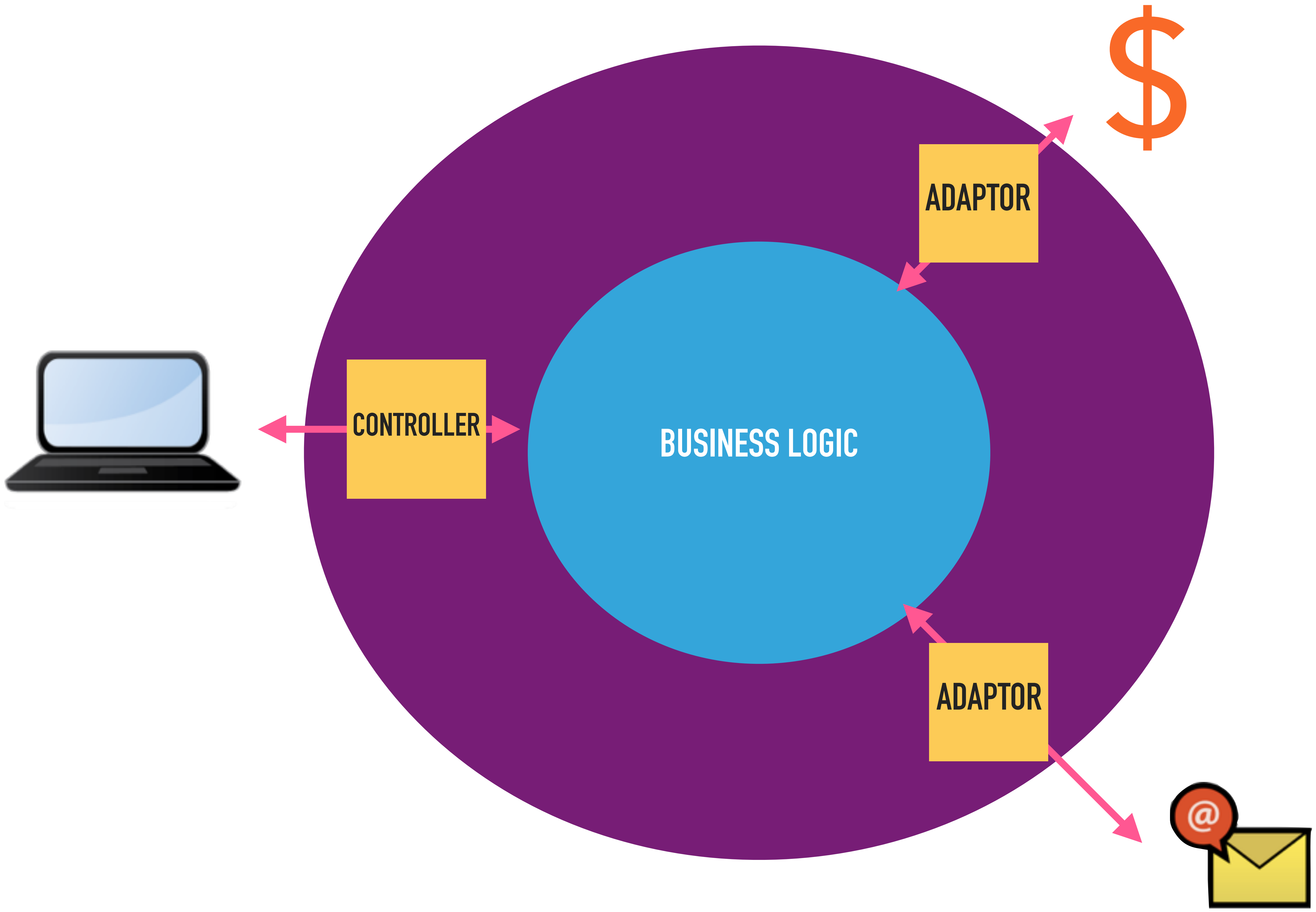
STORY 2



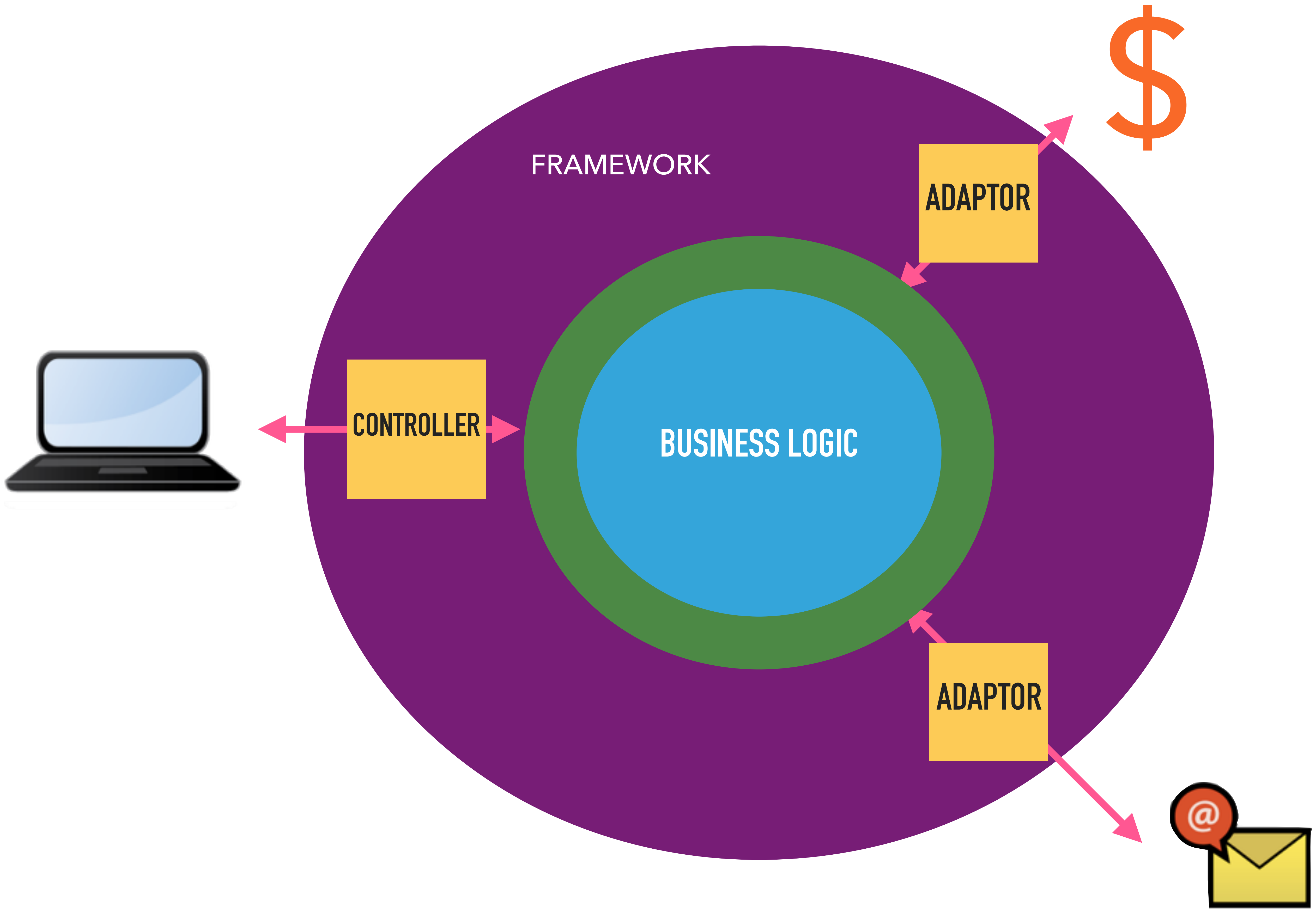
STORY 2



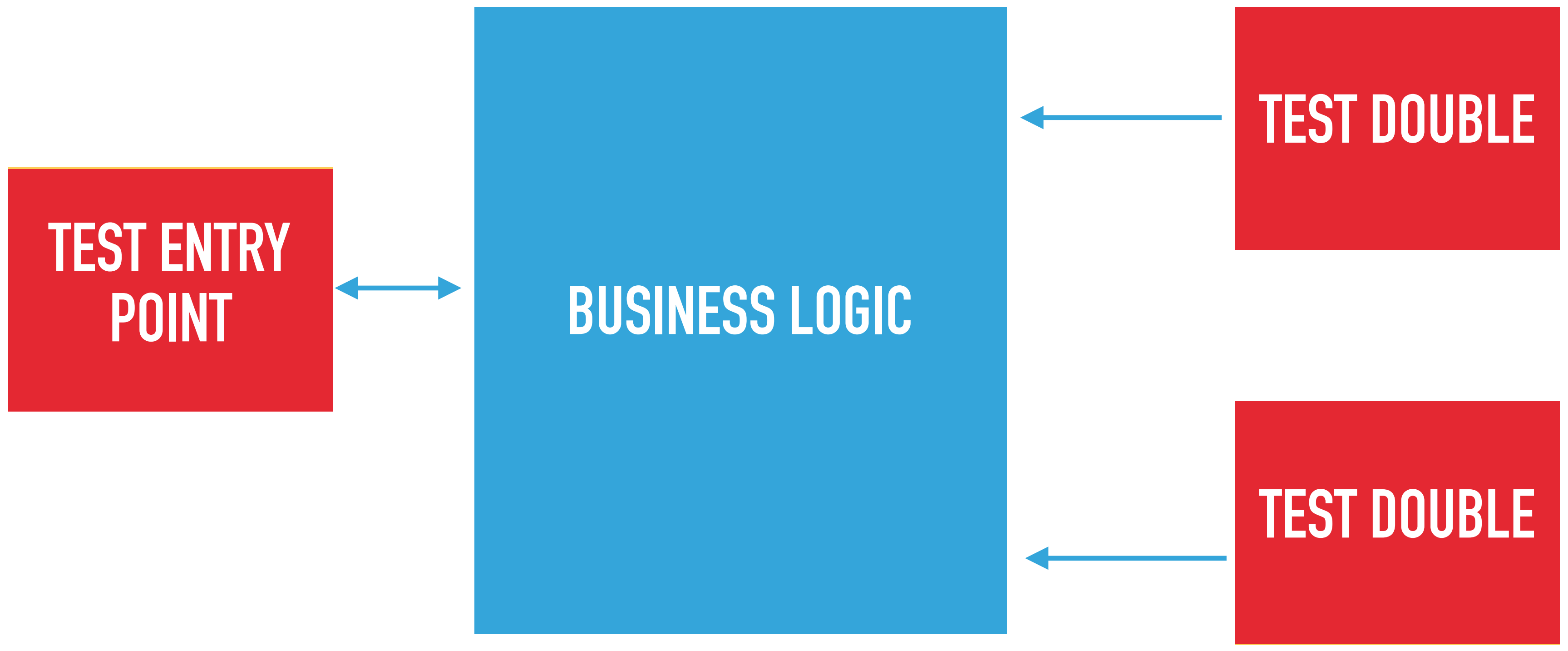
STORY 2



STORY 2

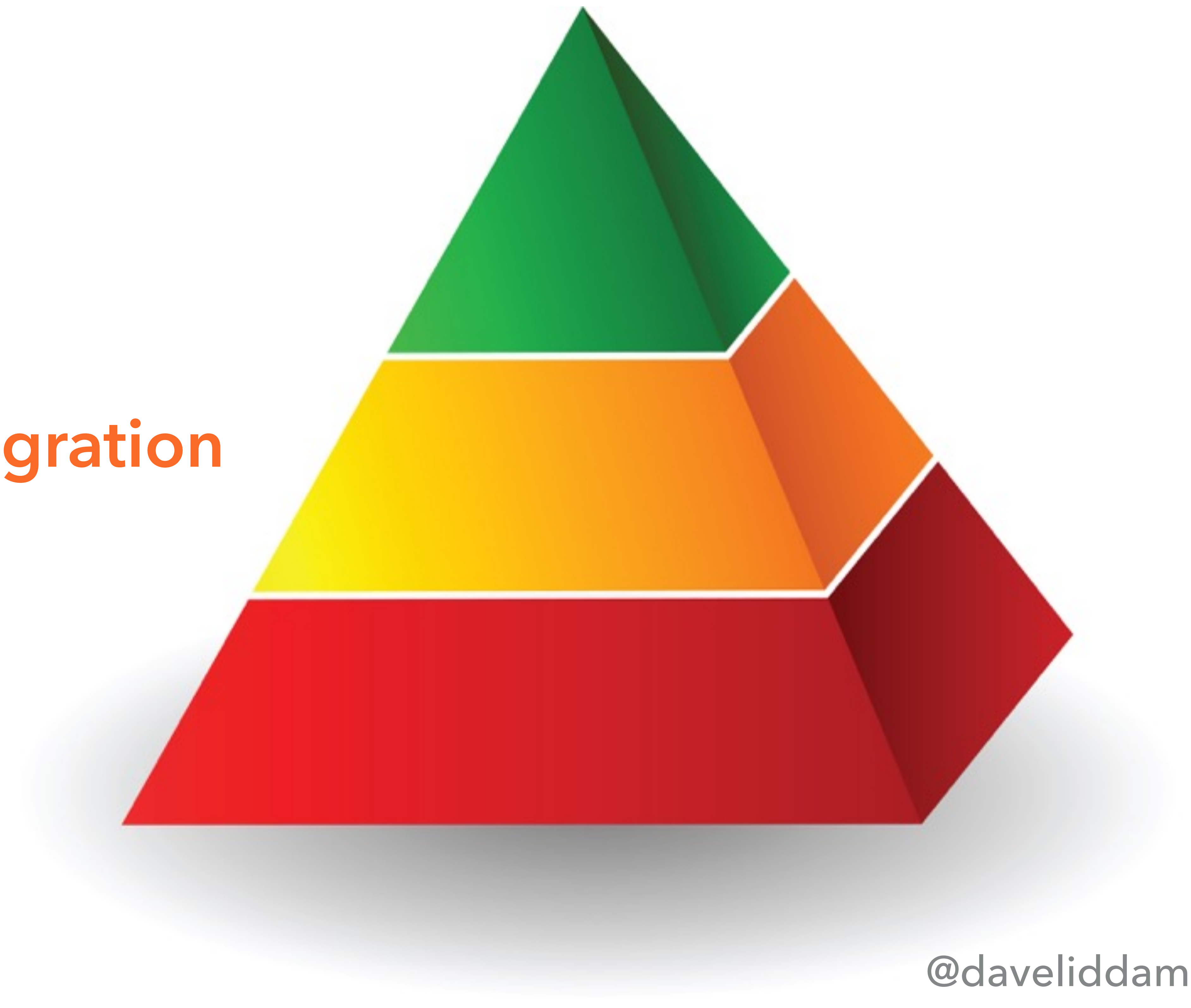


STORY 2



STORY 2

Integration

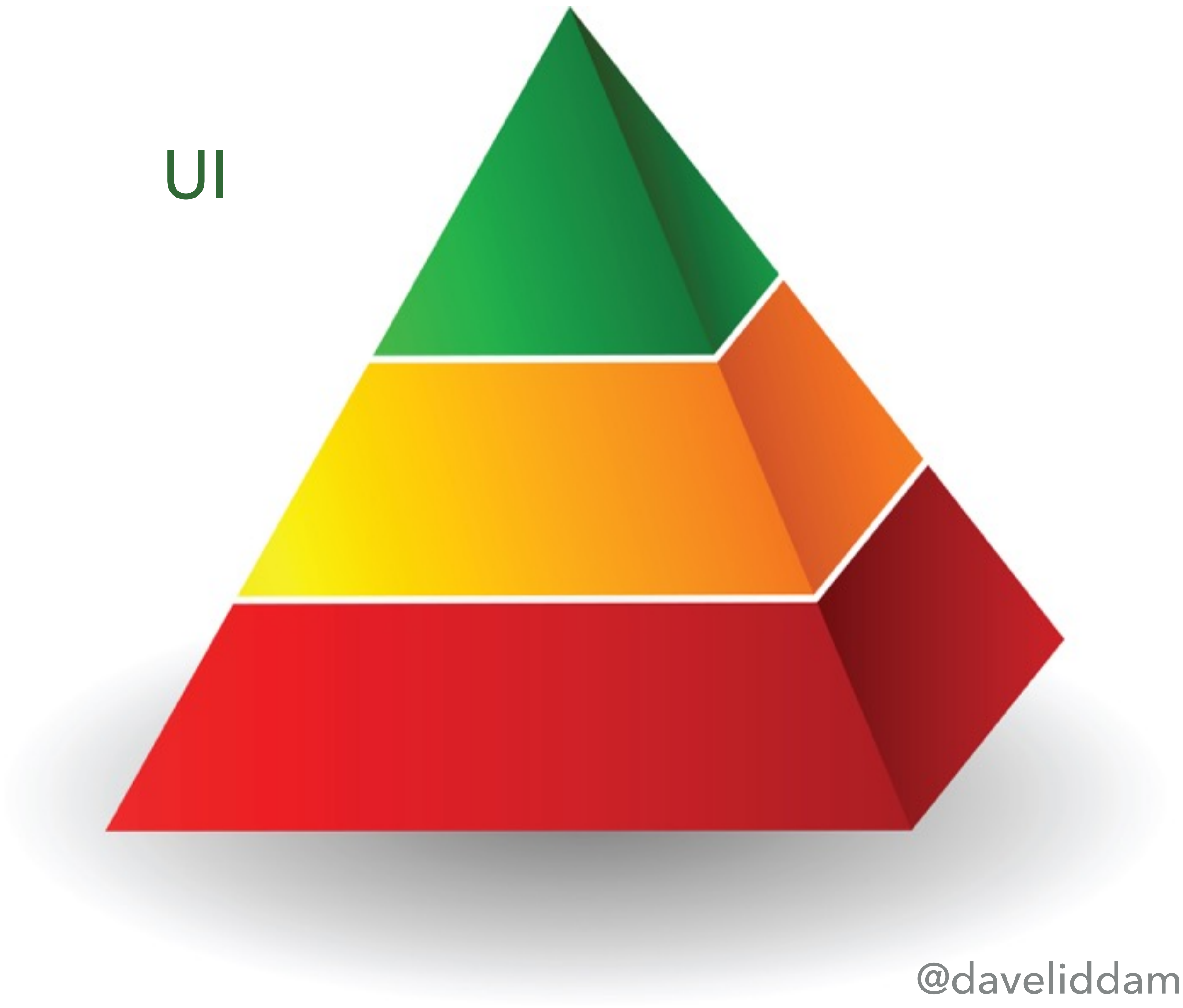


@daveliddament

STORY 2

?

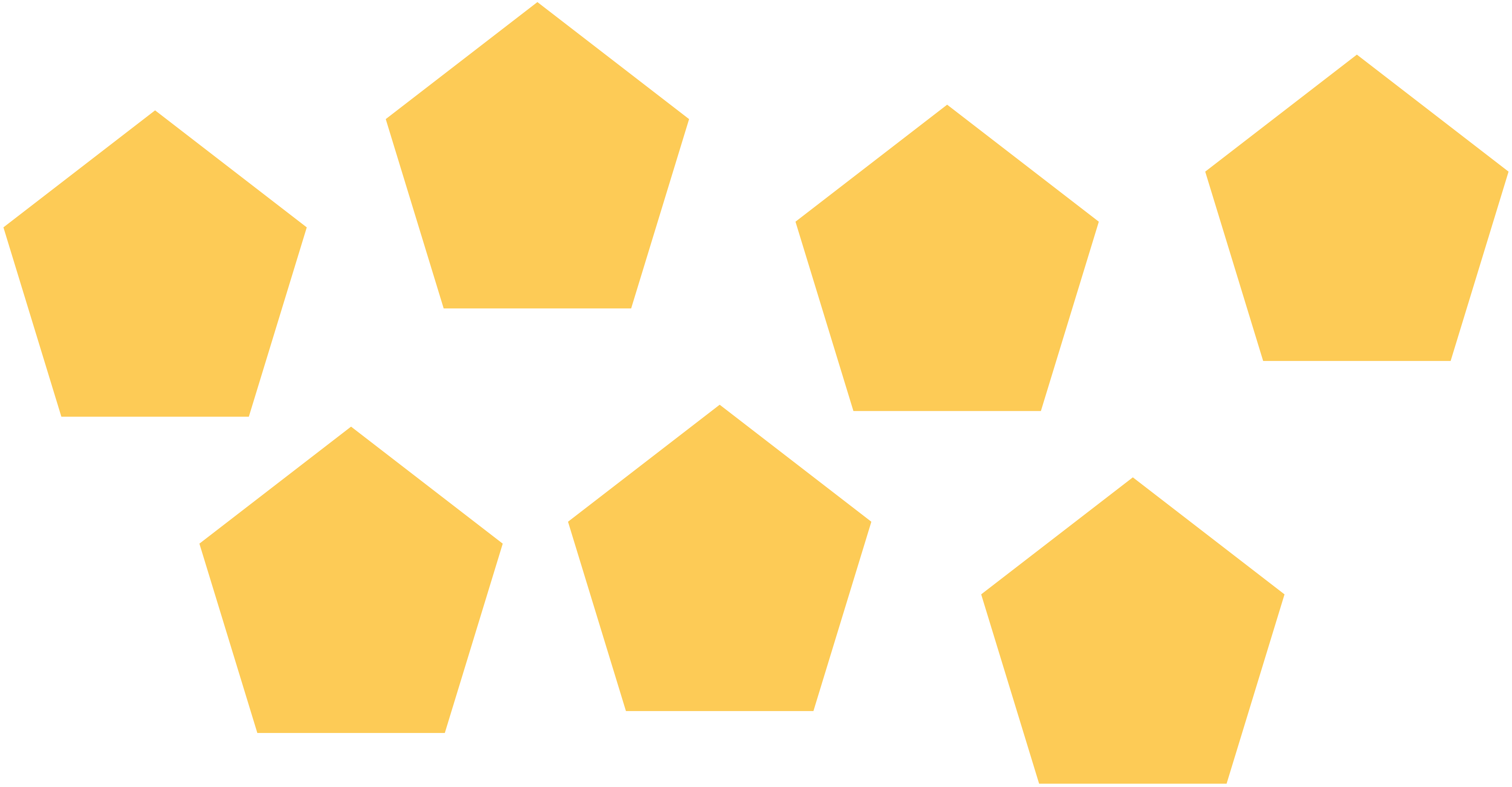
UI



@daveliddament

STORY 2

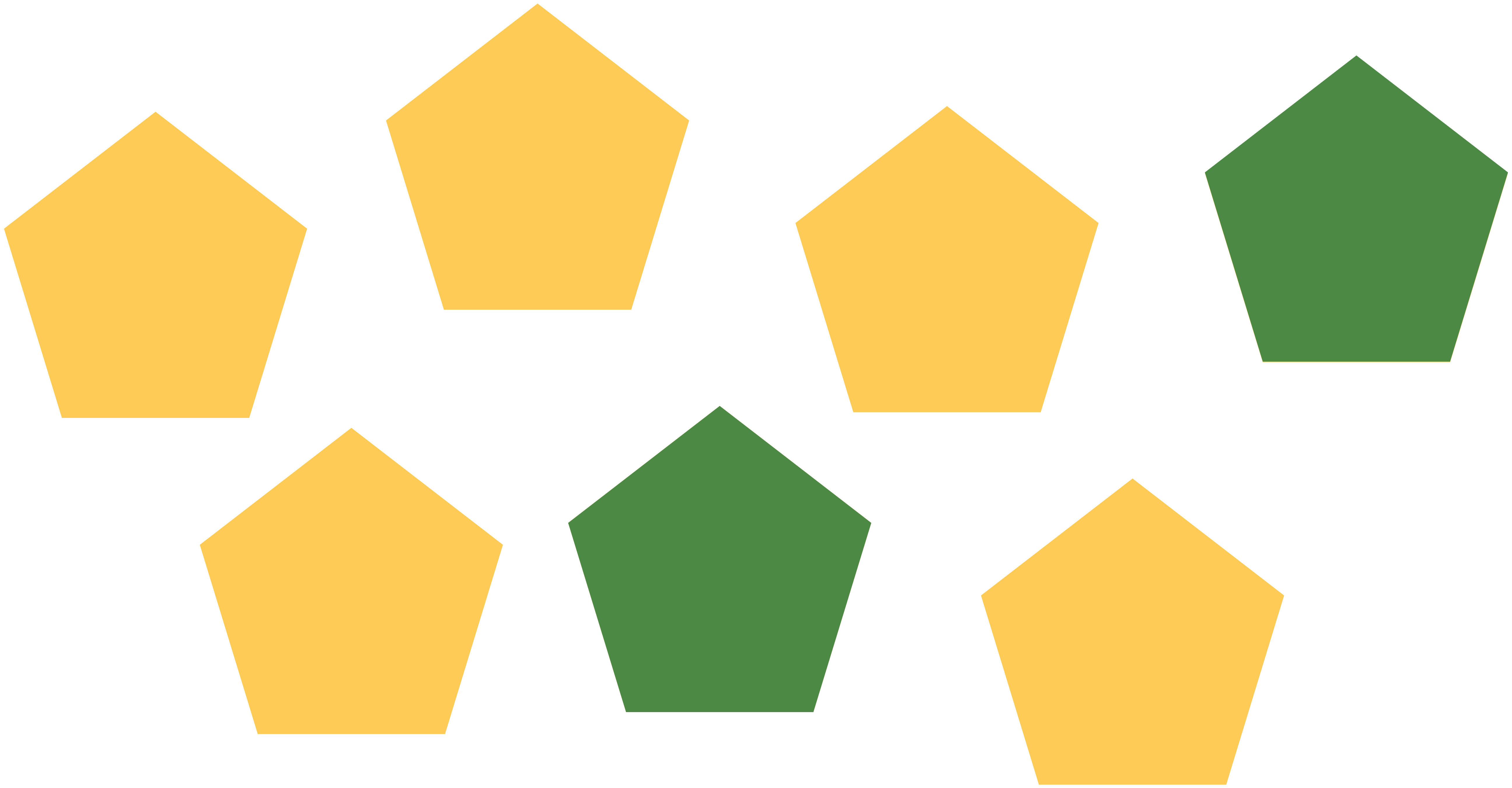
WHAT DO WE TEST AT THE UI LEVEL?



@daveliddament

STORY 2

WHAT DO WE TEST AT THE UI LEVEL?



@daveliddament

THE MORAL OF STORY 2…

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.

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.

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.

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!

STORY 1 CLIFF HANGERS

STORY 1 CLIFF HANGERS

- What happens if we replace the entire website with an app?

STORY 1 CLIFF HANGERS

- What happens if we replace the entire website with an app?
- This feels like a lot of effort.

DECOUPLED TESTS REDUCE THE DEVELOPMENT AND MAINTENANCE COSTS OF THE TEST SUITE.

STORY 2

BUT …

BUT …

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

#3

@daveliddament

STORY 3

WE EXPAND TO OFFER THE SERVICE TO MULTIPLE COMPANIES

WE EXPAND TO OFFER THE SERVICE TO MULTIPLE COMPANIES

- Each company has a branded page on their own subdomain.

WE EXPAND TO OFFER THE SERVICE TO MULTIPLE COMPANIES

- Each company has a branded page on their own subdomain.
- Could could only login from your company’s subdomain.

WE EXPAND TO OFFER THE SERVICE TO MULTIPLE COMPANIES

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

```
................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.  63 / 444 ( 14%)
............................................................... 126 / 444 ( 28%)
.........................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF...... 189 / 444 ( 42%)
FFFFFFFFFFFF................................................... 252 / 444 ( 56%)
...................FF....FFFF.......FFFFFFFFFFFFFF............. 315 / 444 ( 70%)
..................................FFFFFFFFFFFFFFFFFFFFFFFFFFF.. 378 / 444 ( 85%)
FFFFFFFFFFFFFF........FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 441 / 444 ( 99%)
...

Time: 20 minutes 54 seconds, Memory: 24.75MB

 There were lots of failures:
```

😞

@daveliddament

ONE OF THE MANY FAILING TESTS…

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

STORY 3

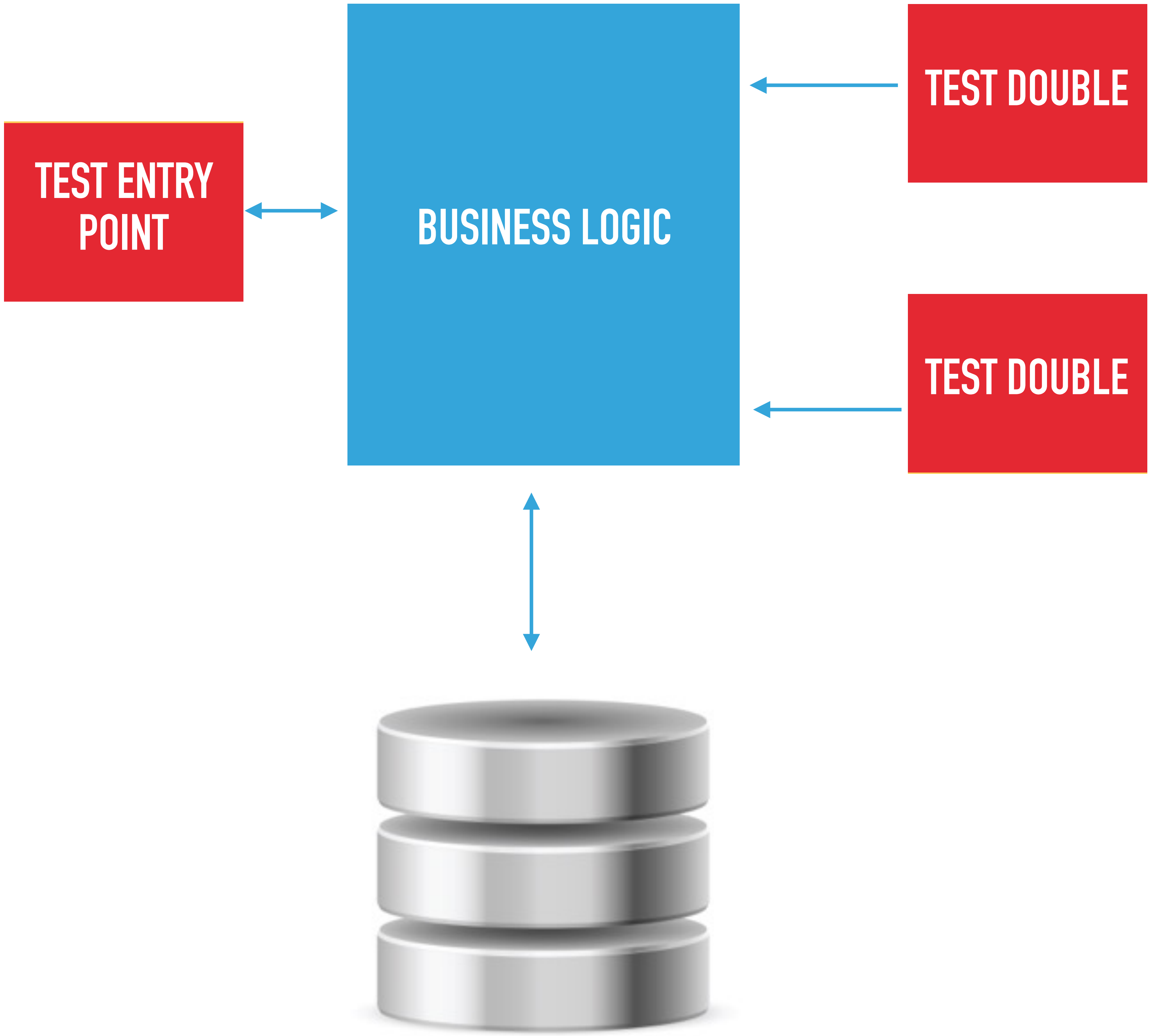
SEEDING A DATABASE

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


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

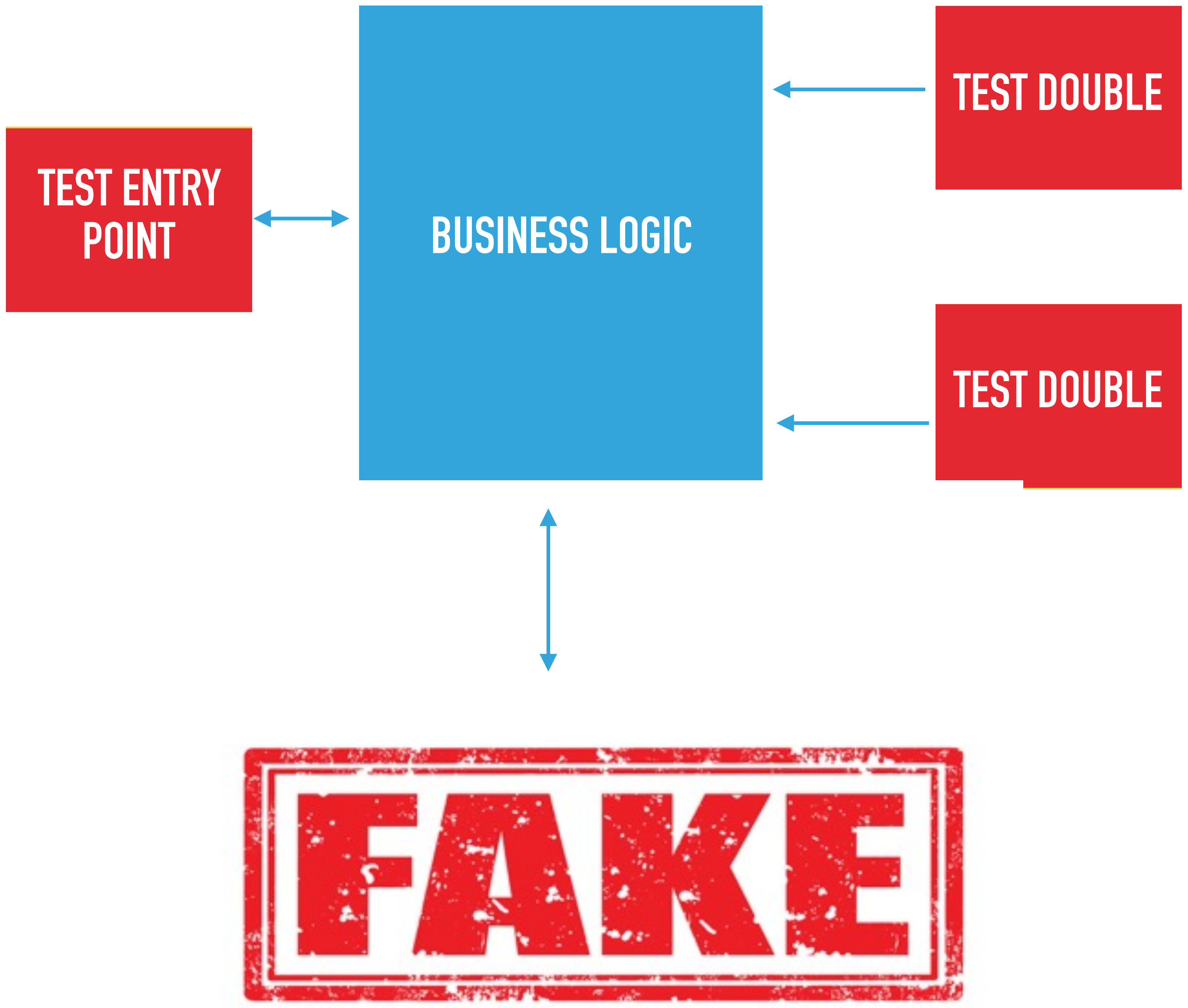
SEEDING A DATABASE

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


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

STORY 3



STORY 3


TEST ENTRY POINT
BUSINESS LOGIC
TEST DOUBLE
TEST DOUBLE
FAKE

BUILDING DATA FIXTURES

HAND BUILDING

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

HAND BUILDING

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

HAND BUILDING

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

XXX

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 = $this->getUserBuilder();
$user = $userBuilder->build();

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

USING A TEST BUILDER (2)

```
$userBuilder = $this->getUserBuilder();
$user = $userBuilder
          ->name(“Annabelle”)
          ->password(“Passw4rd”)
          ->team(“Banana”)
          ->build();
```

DEFER TO OTHER OBJECT MOTHERS / BUILDERS

```
class UserObjectMother {

   public function getAnna(): User {

      $company = $this->companyObjectMother()
          ->getAcmeCompany();

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

      return $user;
}
```

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…

MORAL OF STORY 3…

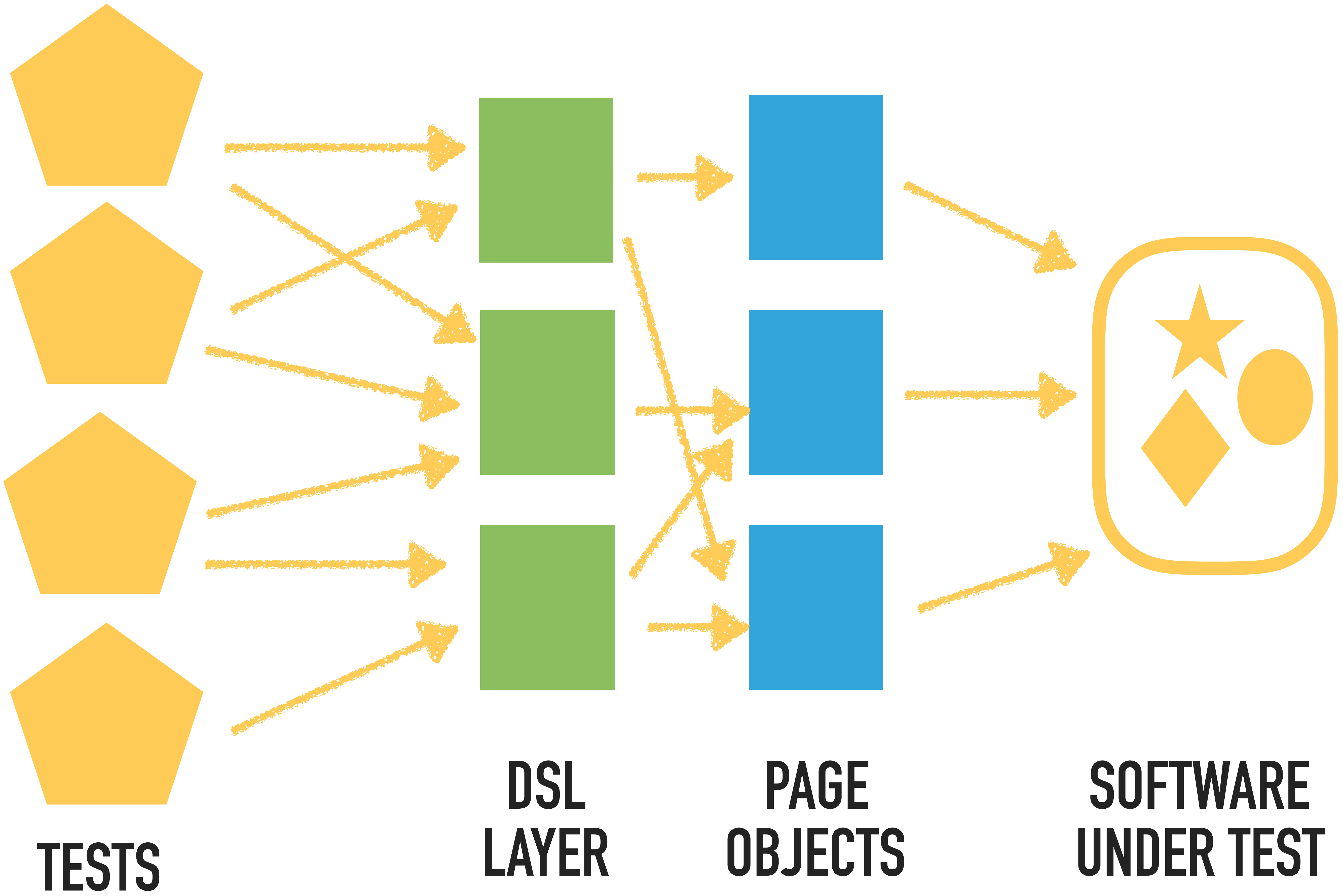
- Use patterns like Object Mothers / Test Builders for building data fixtures.
  - Makes tests more robust to change.
  - Allows us to test with a fake in memory database.

MORAL OF STORY 3…

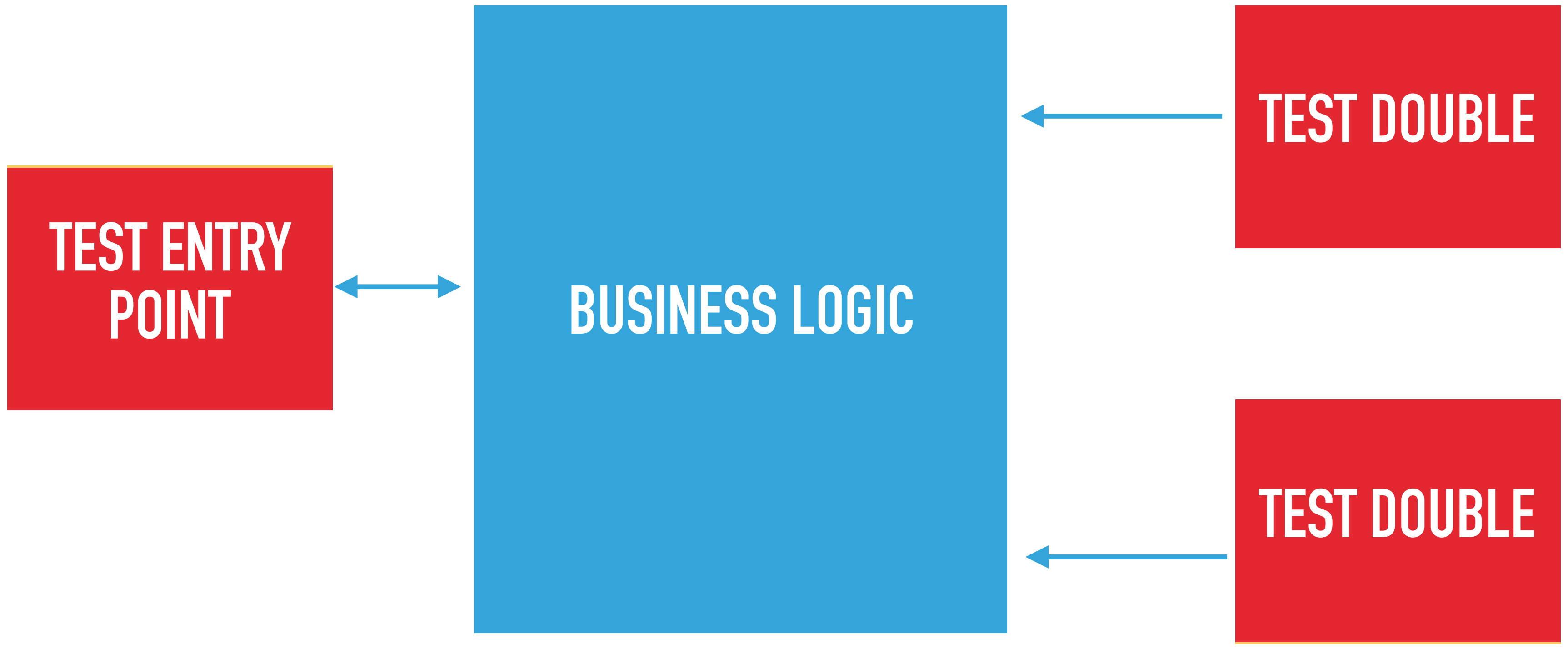
- Use patterns like Object Mothers / Test Builders for building data fixtures.
  - Makes tests more robust to change.
  - Allows us to test with a fake in memory database.
- Decoupling our tests from the software under test.

DECOUPLED TESTS REDUCE THE DEVELOPMENT AND MAINTENANCE COSTS OF THE TEST SUITE.

STORY 1



STORY 2



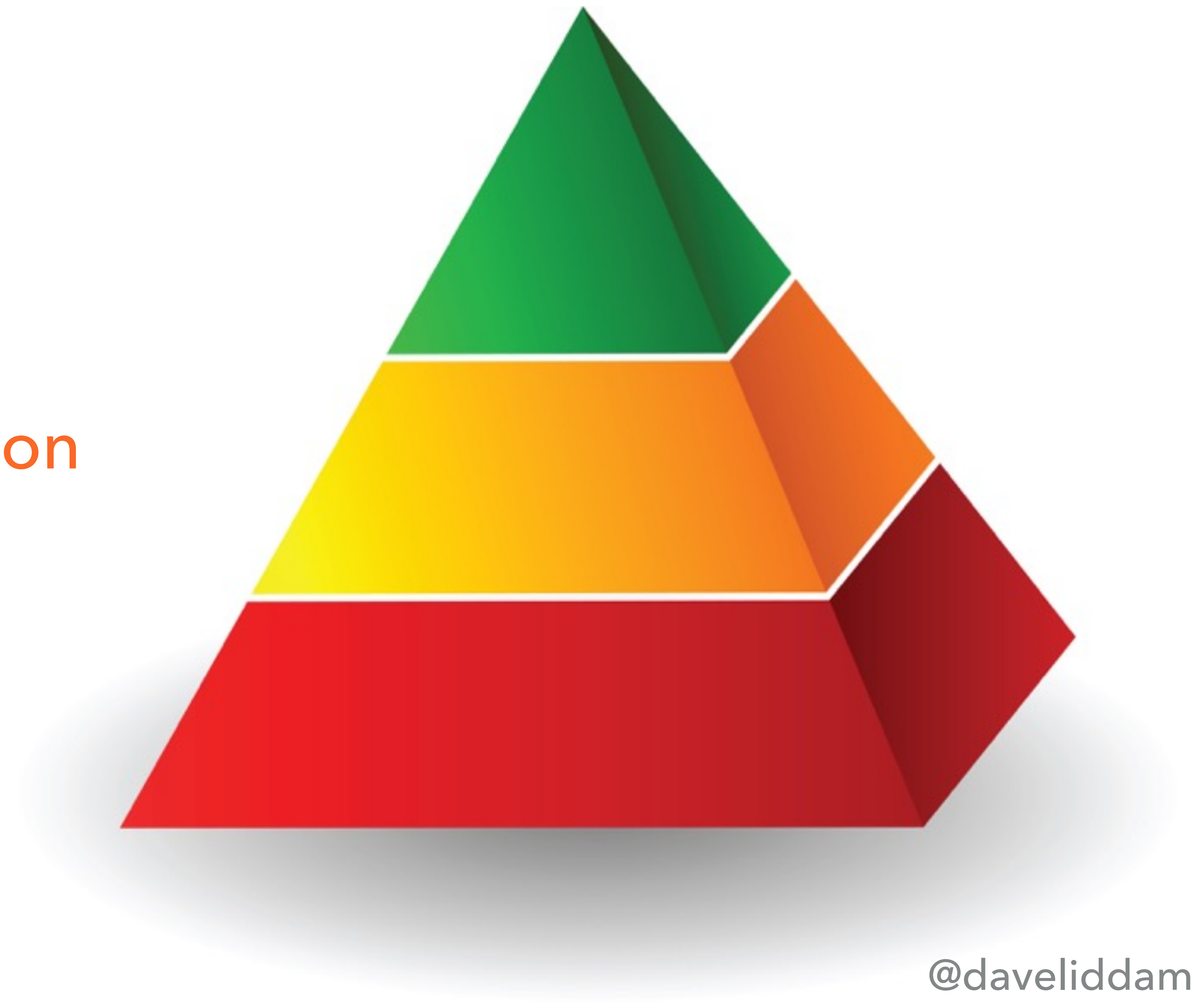
STORY 3

TEST PYRAMID

UI

Integration

Unit



SUMMARY

SUMMARY

- Decoupling is good

SUMMARY

- Decoupling is good
  - Reduces development and maintenance costs

SUMMARY

- Decoupling is good
  - Reduces development and maintenance costs
- Do the right kind of tests at the right level

SUMMARY

- Decoupling is good
  - Reduces development and maintenance costs
- Do the right kind of tests at the right level
  - Architect the code correctly

SUMMARY

- Decoupling is good
  - Reduces development and maintenance costs
- Do the right kind of tests at the right level
  - Architect the code correctly
  - Test business logic at the service layer

SUMMARY

- Decoupling is good
  - Reduces development and maintenance costs
- Do the right kind of tests at the right level
  - Architect the code correctly
  - Test business logic at the service layer
  - Test UI to check it is correctly wired up to service layer

SUMMARY

- Decoupling is good
  - Reduces development and maintenance costs
- Do the right kind of tests at the right level
  - Architect the code correctly
  - Test business logic at the service layer
  - Test UI to check it is correctly wired up to service layer
- Building objects using Object Mother / Builder patterns

Thanks for listening

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