AssertTrue(isDecoupled(“my tests”))

Dave Liddament

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

AGENDA

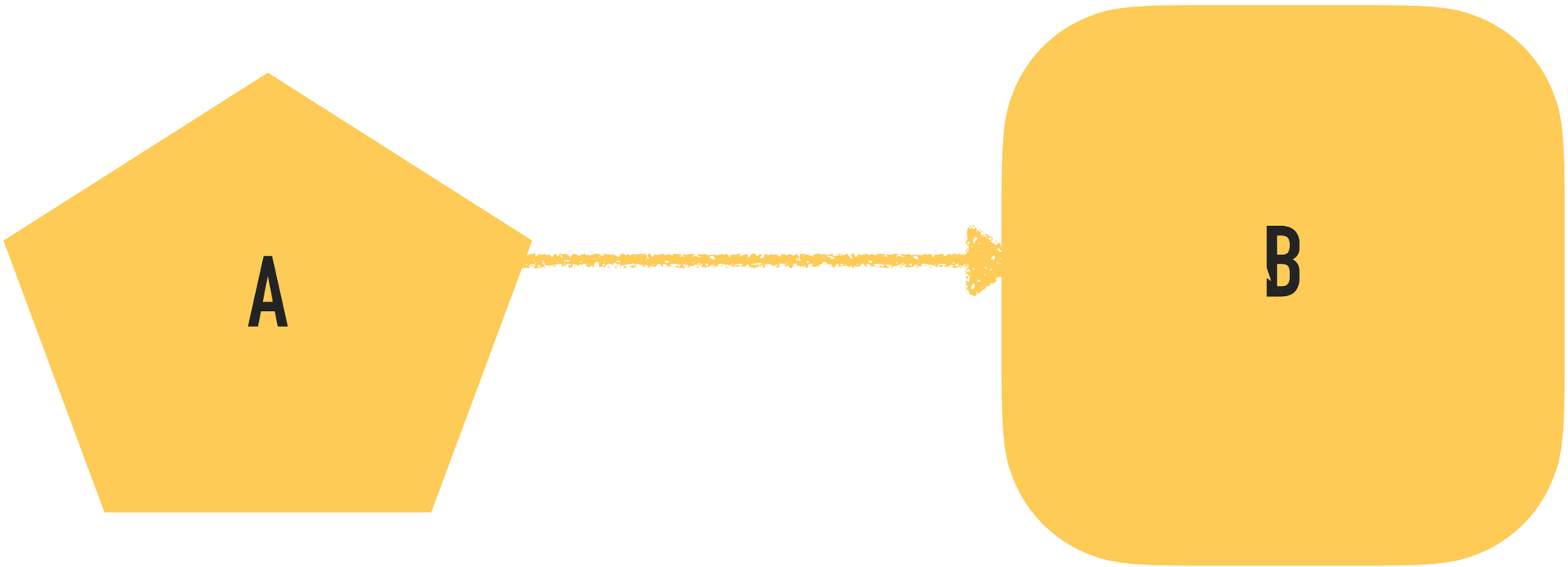
- Why
- Terminology

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

Time: 20 minutes 54 seconds, Memory: 24.75MB

OK (444 tests, 1201 assertions)
```

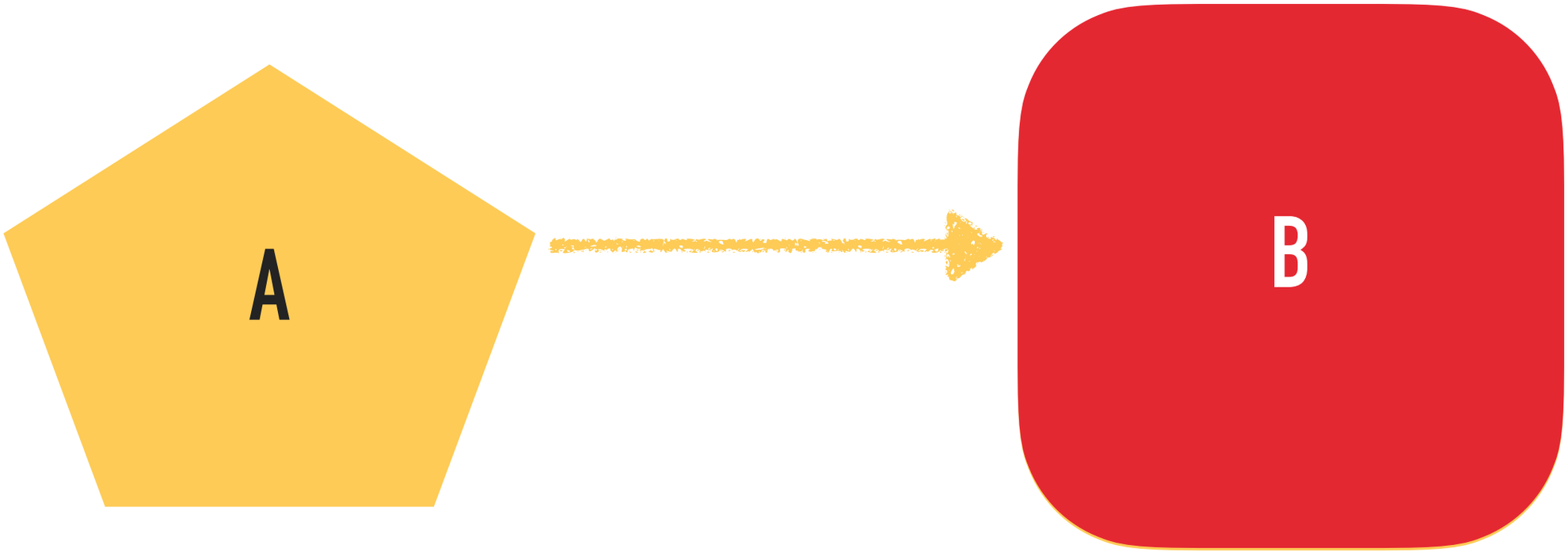
```
..................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.  63 / 444 ( 14%)
.................................................................. 126 / 444 ( 28%)
.........................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.......... 189 / 444 ( 42%)
FFFFFFFFFFFF.............................................................. 252 / 444 ( 56%)
..................FF....FFFF.......FFFFFFFFFFFFFF.............. 315 / 444 ( 70%)
..............................FFFFFFFFFFFFFFFFFFFFFFFFFFFF.. 378 / 444 ( 85%)
FFFFFFFFFFFFFF........FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 441 / 444 ( 99%)
...

Time: 20 minutes 54 seconds, Memory: 24.75MB

 There were lots of failures:
```

@daveliddament

COUPLING



TEST DOUBLES



TEST PYRAMID

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

#1

@daveliddament

TYPICAL USER JOURNEY

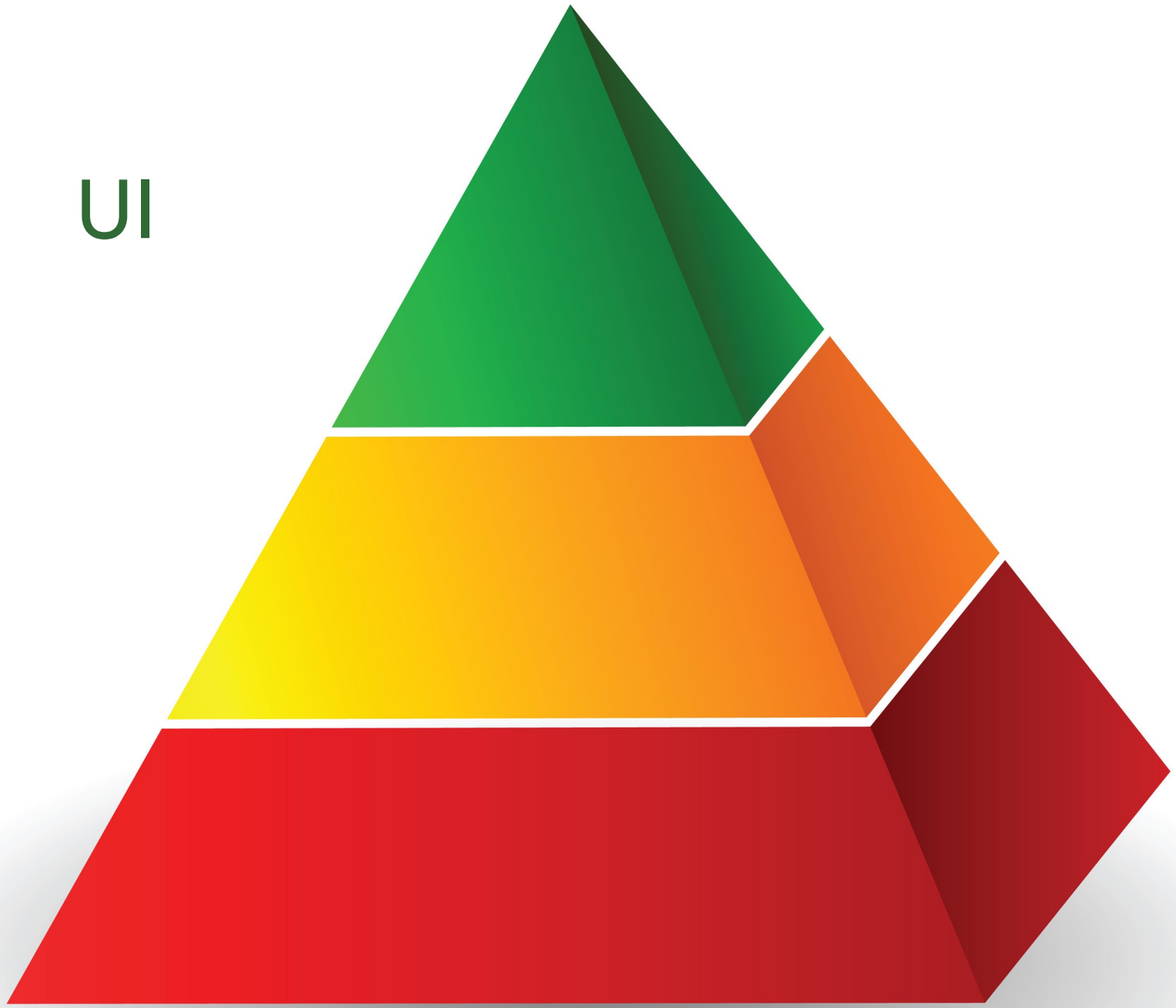
- Bob would log in
- Bob see a list of quizzes
- Pick one he hadn’t done
- Complete the quiz
- See his score
- His team’s score would be updated

I WANT TO TEST . . .

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

STORY 1

UI



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. . . .

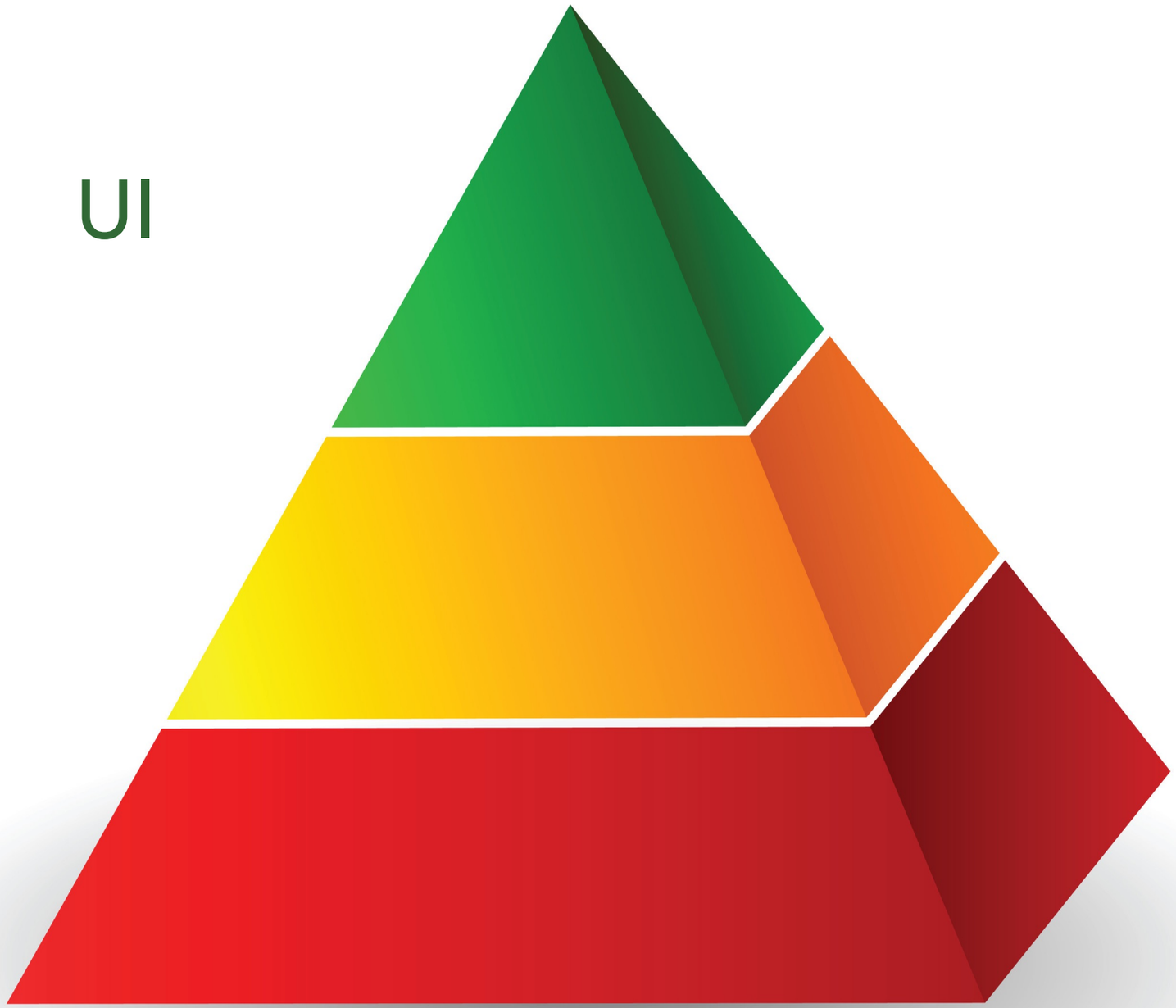
Can we change the layout of the page showing the lists of quizzes?

```
..................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.  63 / 444 ( 14%)
.................................................................. 126 / 444 ( 28%)
.........................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.......... 189 / 444 ( 42%)
FFFFFFFFFFFF.............................................................. 252 / 444 ( 56%)
..................FF....FFFF.......FFFFFFFFFFFFFF.............. 315 / 444 ( 70%)
..............................FFFFFFFFFFFFFFFFFFFFFFFFFFFF.. 378 / 444 ( 85%)
FFFFFFFFFFFFFF........FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 441 / 444 ( 99%)
...

Time: 20 minutes 54 seconds, Memory: 24.75MB

 There were lots of failures:
```

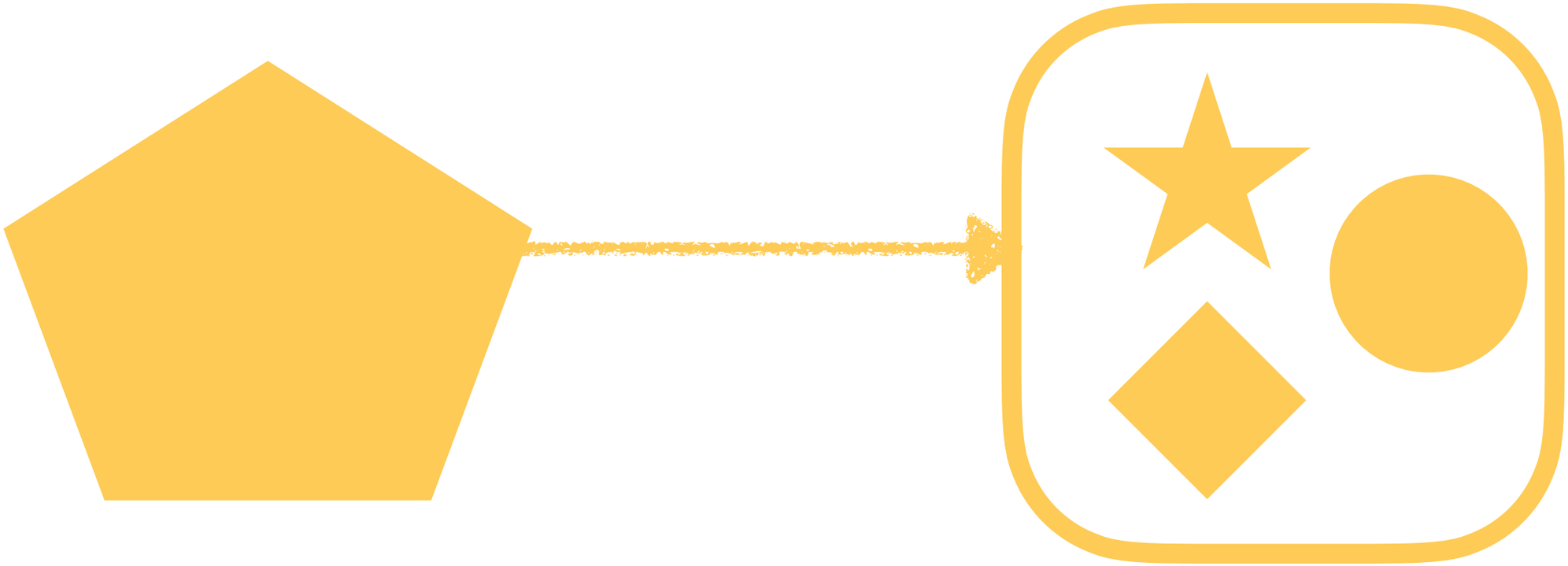
@daveliddament

STORY 1

UI



PROBLEM: TIGHT COUPLING



TEST

SOFTWARE UNDER TEST

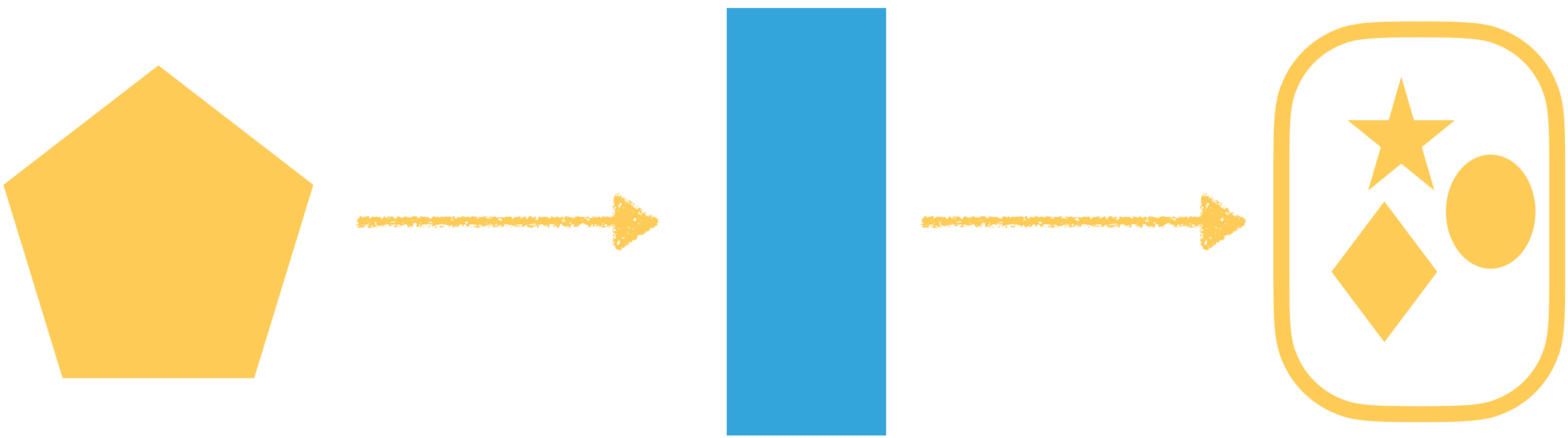
REDUCE COUPLING WITH PAGE OBJECT

login ($username, $password)

answerQuestion ($answer)

findElementByName ($name)

click ()



TEST

PAGE OBJECT

SOFTWARE UNDER TEST

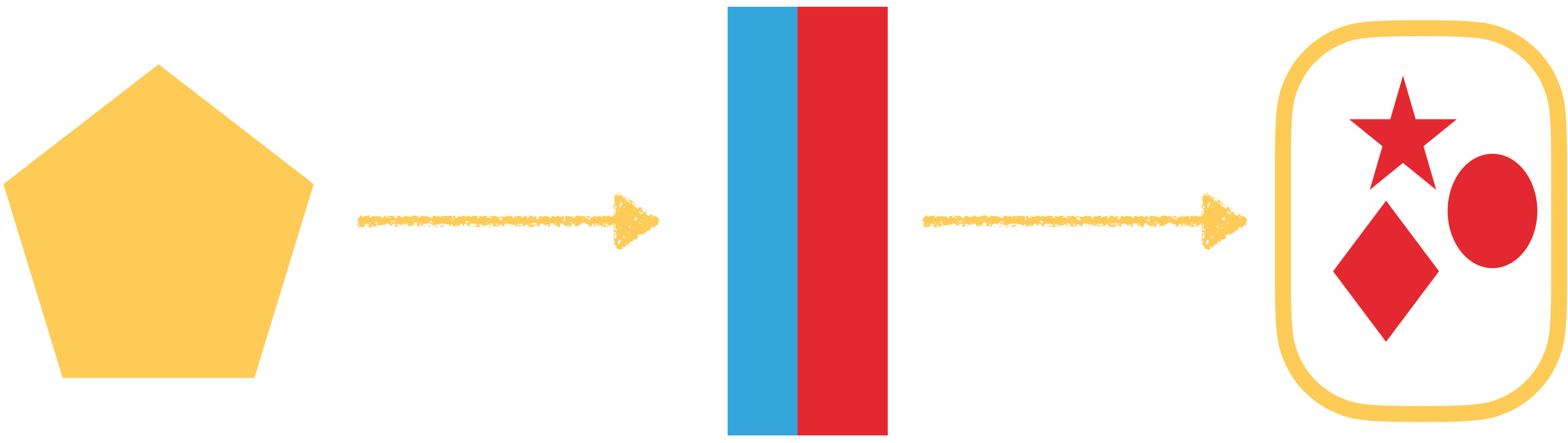
A PAGE OBJECT CAN...

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

REDUCE COUPLING WITH PAGE OBJECT

login ($username, $password)

answerQuestion ($answer)

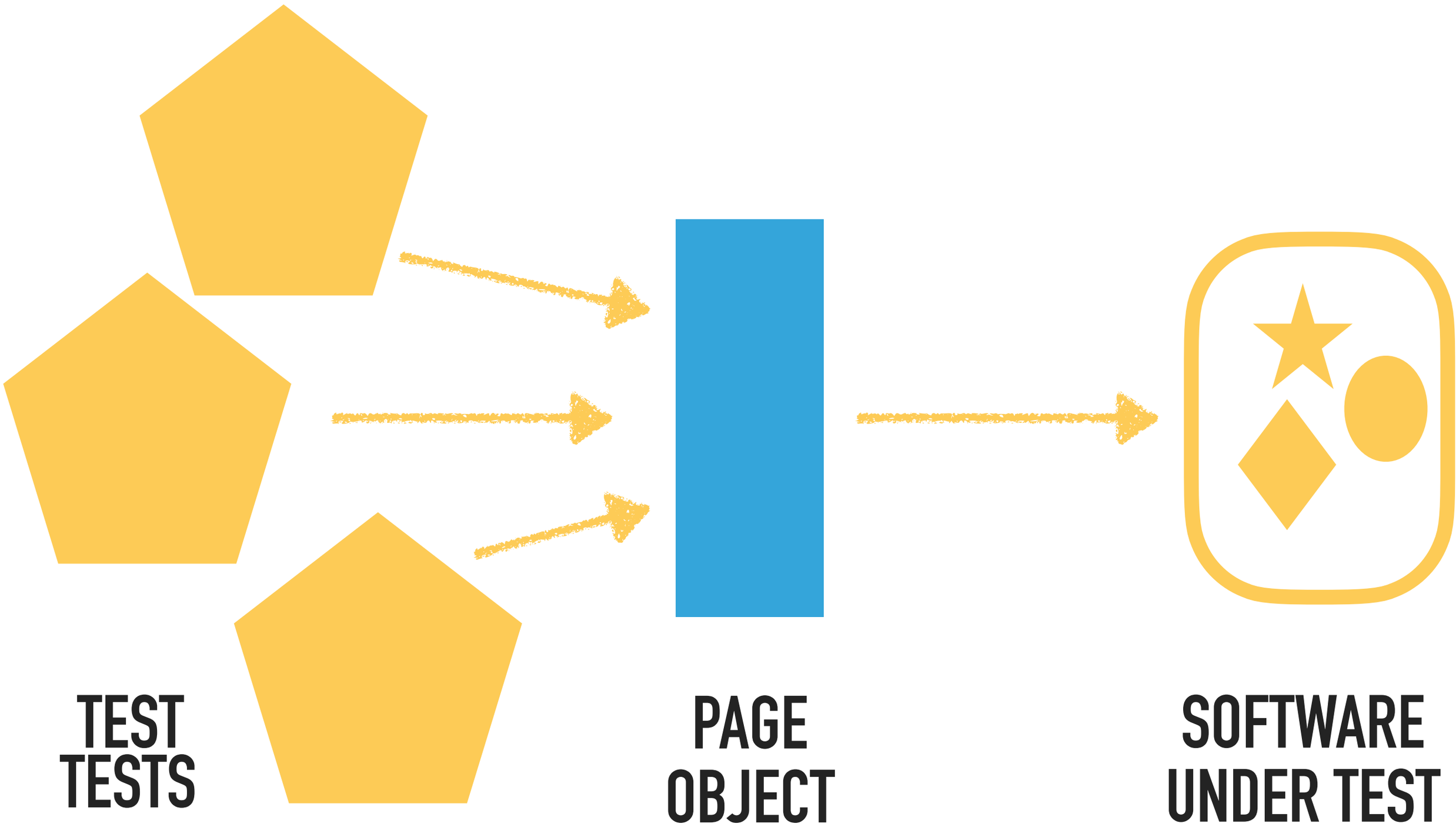


TEST

PAGE OBJECT

SOFTWARE UNDER TEST

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?

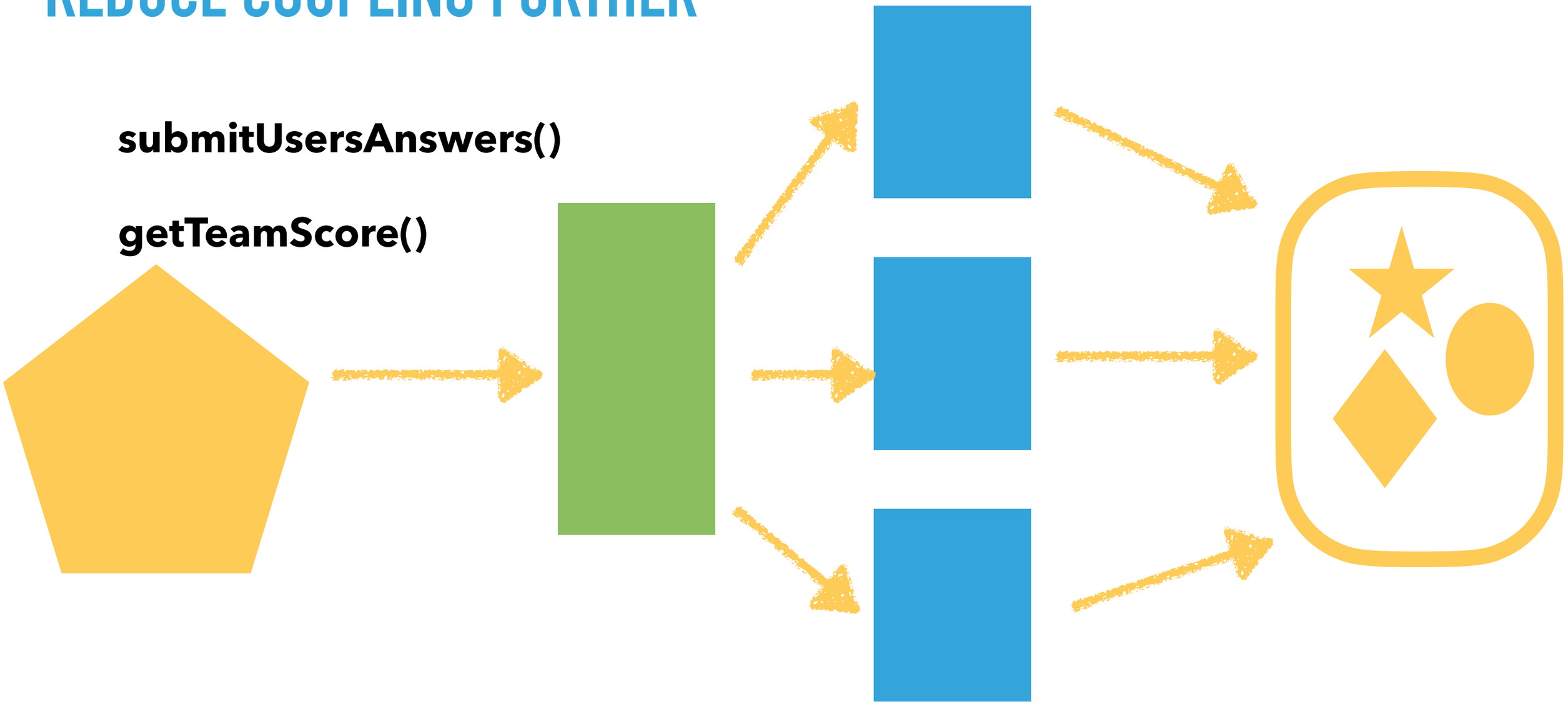
THE TESTS WILL BREAK

```
$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 ...
```

```
..................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.  63 / 444 ( 14%)
.................................................................. 126 / 444 ( 28%)
.........................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.......... 189 / 444 ( 42%)
FFFFFFFFFFFF.............................................................. 252 / 444 ( 56%)
..................FF....FFFF.......FFFFFFFFFFFFFF.............. 315 / 444 ( 70%)
..............................FFFFFFFFFFFFFFFFFFFFFFFFFFFF.. 378 / 444 ( 85%)
FFFFFFFFFFFFFF........FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 441 / 444 ( 99%)
...

Time: 20 minutes 54 seconds, Memory: 24.75MB

 There were lots of failures:
```

@daveliddament

REDUCE COUPLING FURTHER

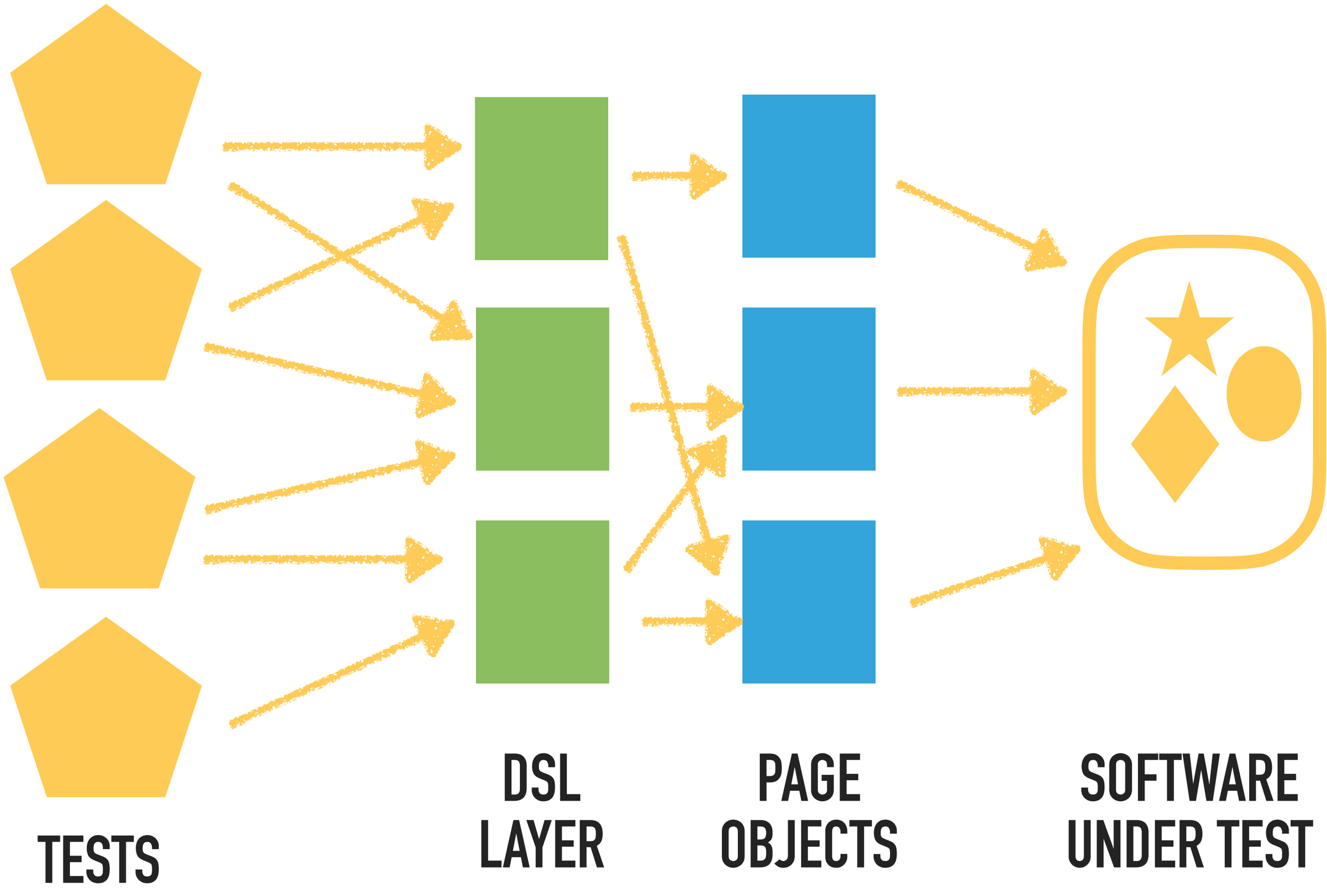


TEST LOOK A BIT MORE LIKE THIS

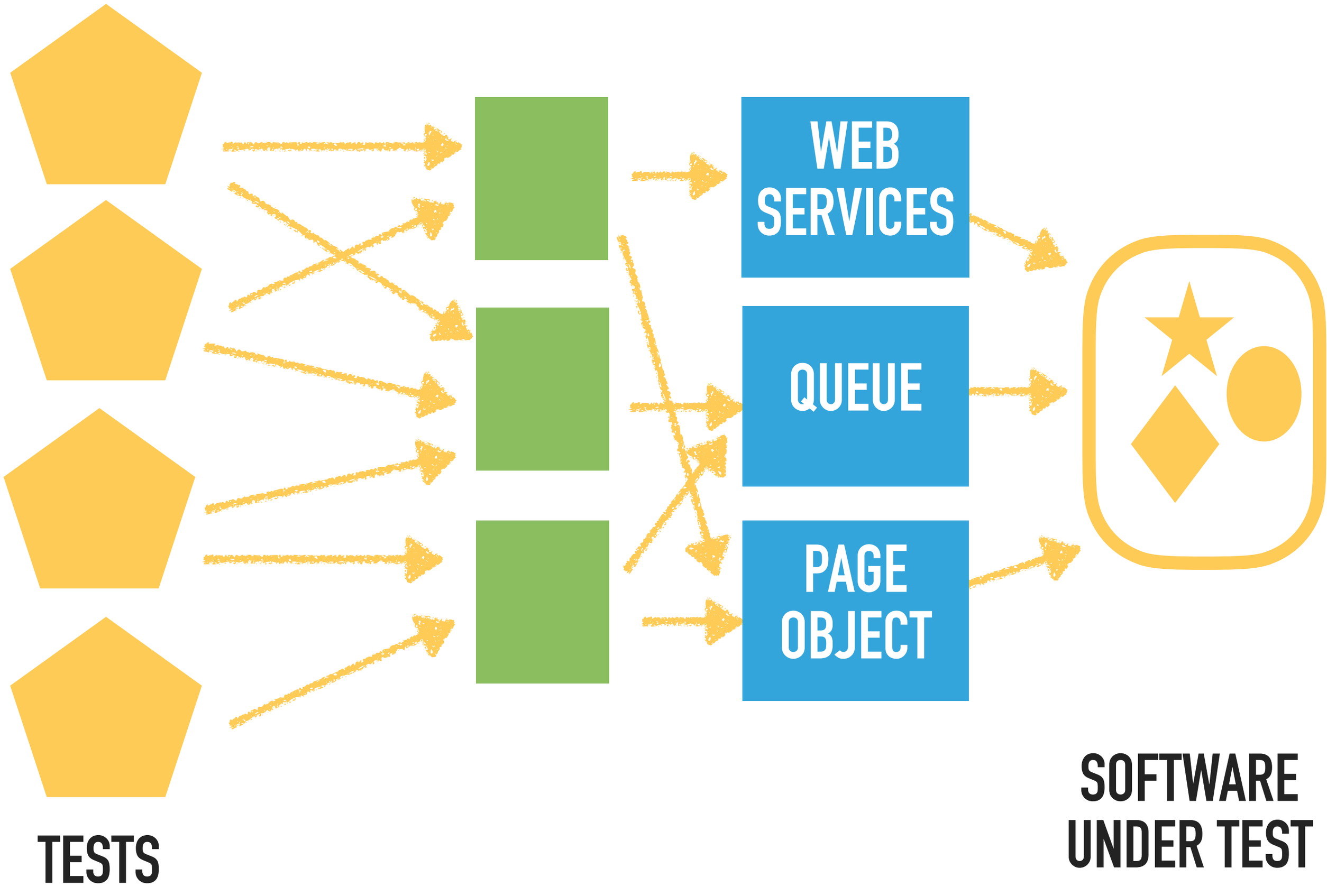
```
assignUserToTeam($bob, $teamApple);

submitUsersAnswers($bob, self::QUIZ_1,

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

$score = getTeamScore($apple);

assertEquals(7, $score);
```

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!

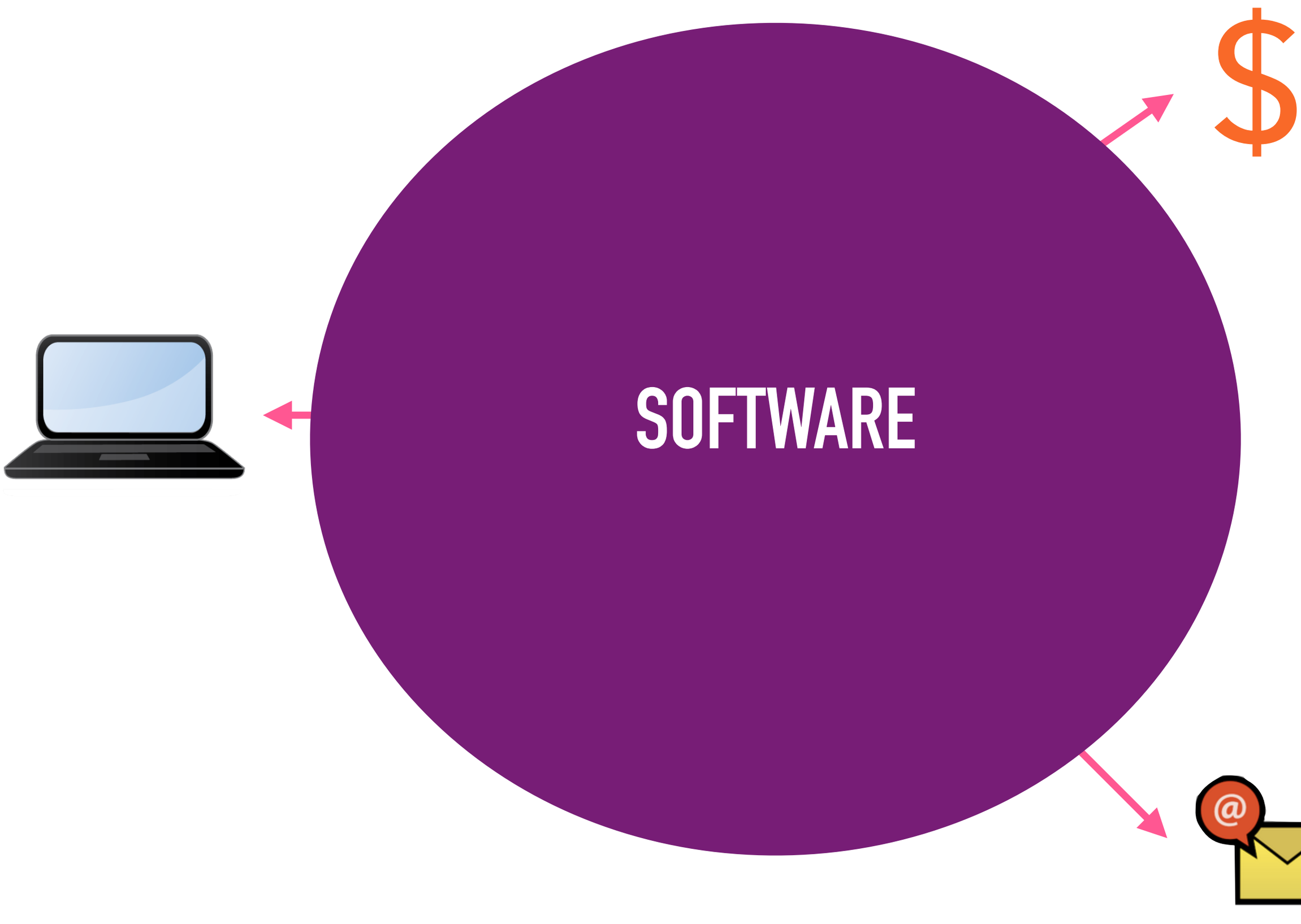
DECOUPLED TESTS REDUCE THE DEVELOPMENT AND MAINTENANCE COSTS OF THE TEST SUITE.

BUT WHAT IF . . .

We replace the entire website with an app?

#2

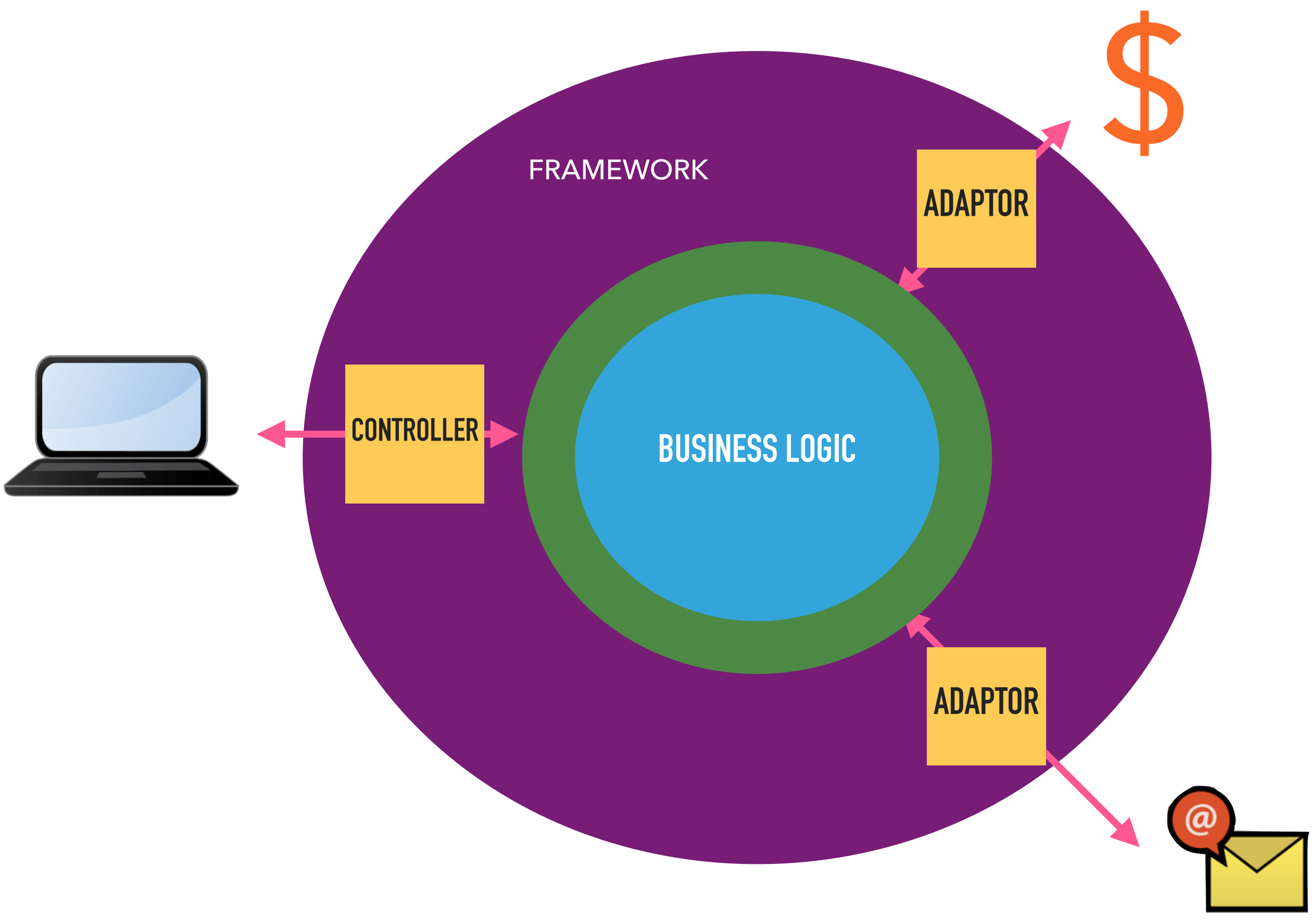
@daveliddament


$
SOFTWARE

THERE MUST BE A BETTER WAY . . .

- Layered architecture
- Hexagonal architecture

STORY 2



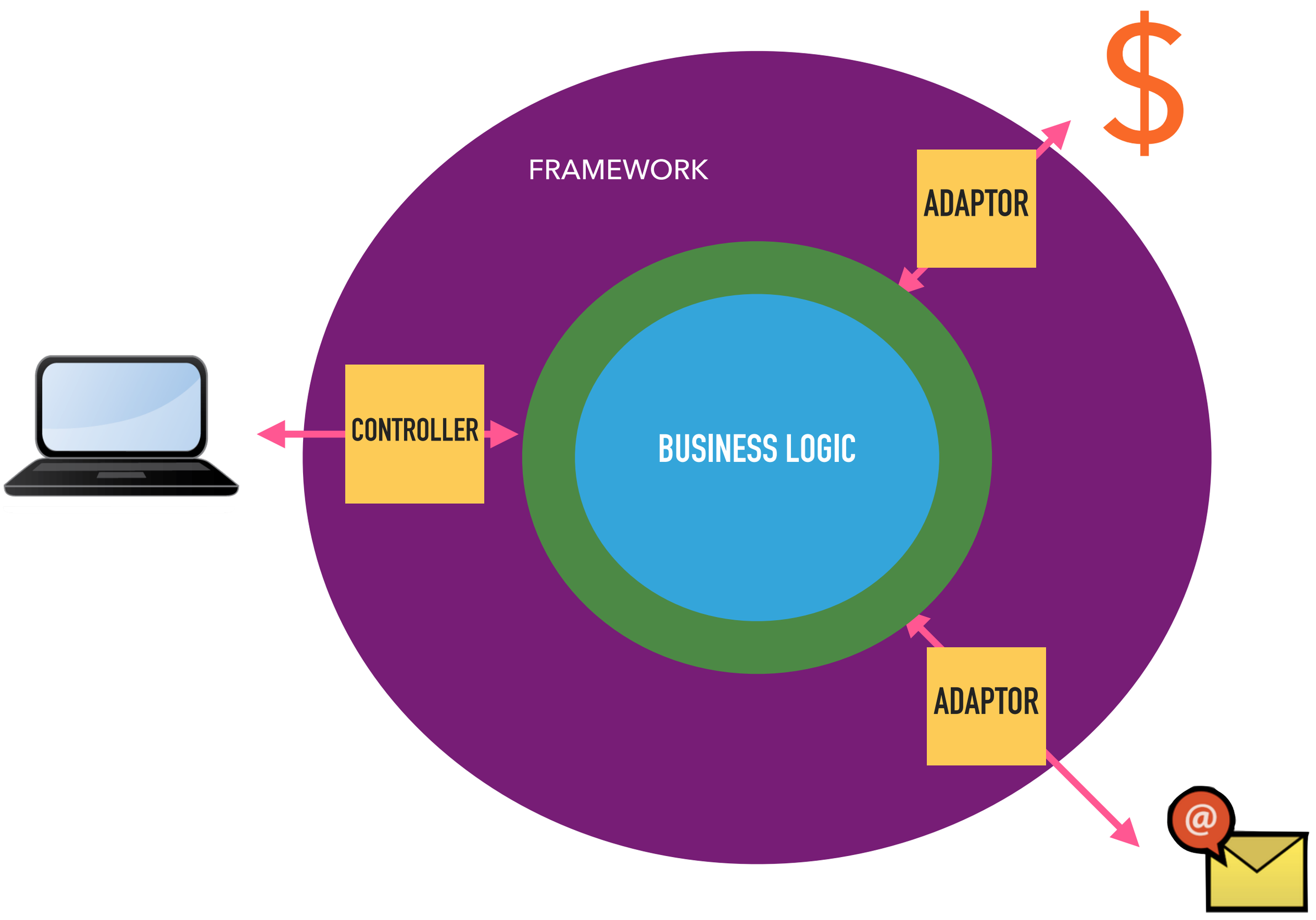
SERVICE LAYER

```
interface AnswerSubmissionService
{

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

}
```

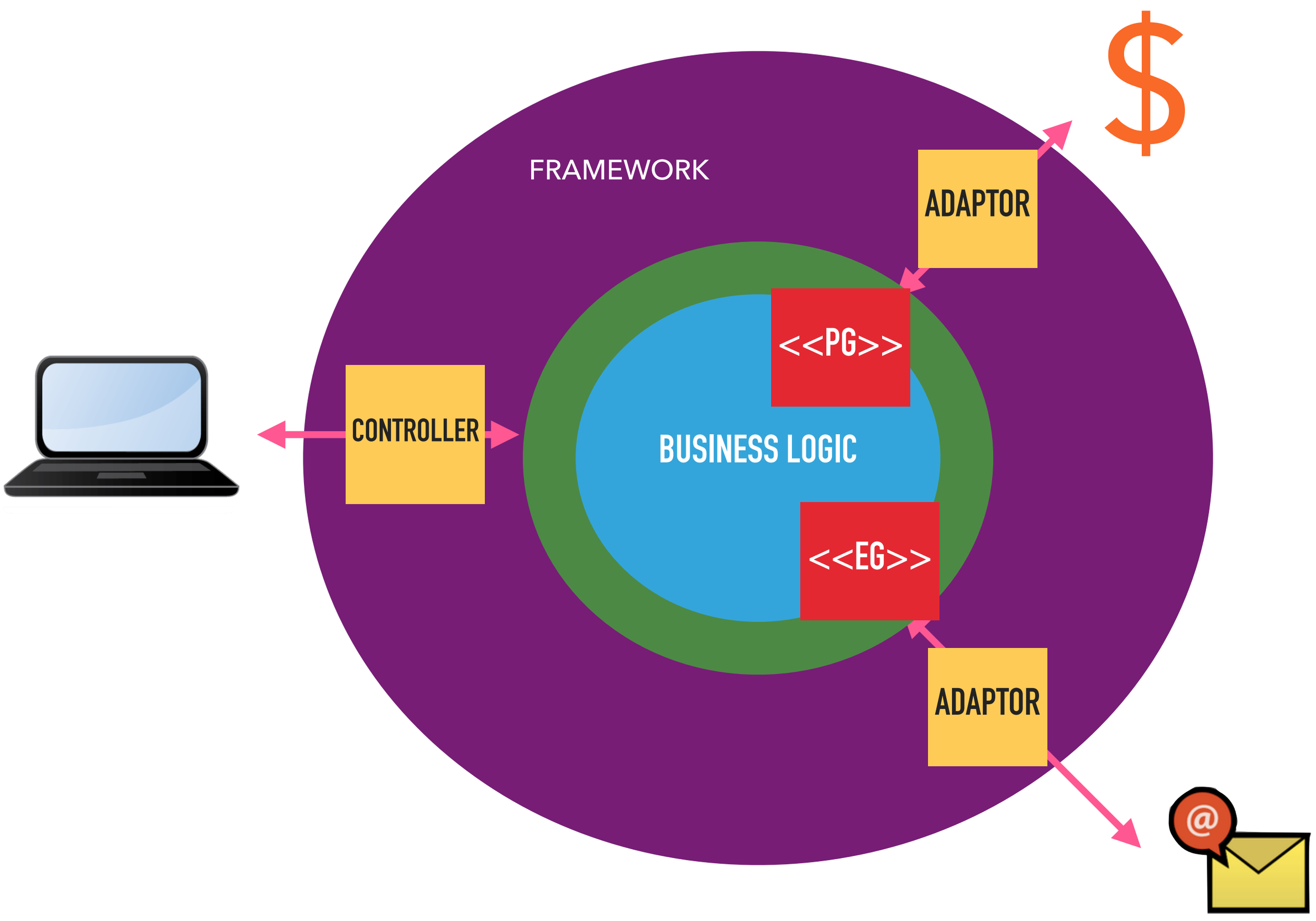
STORY 2

Integration

STORY 2



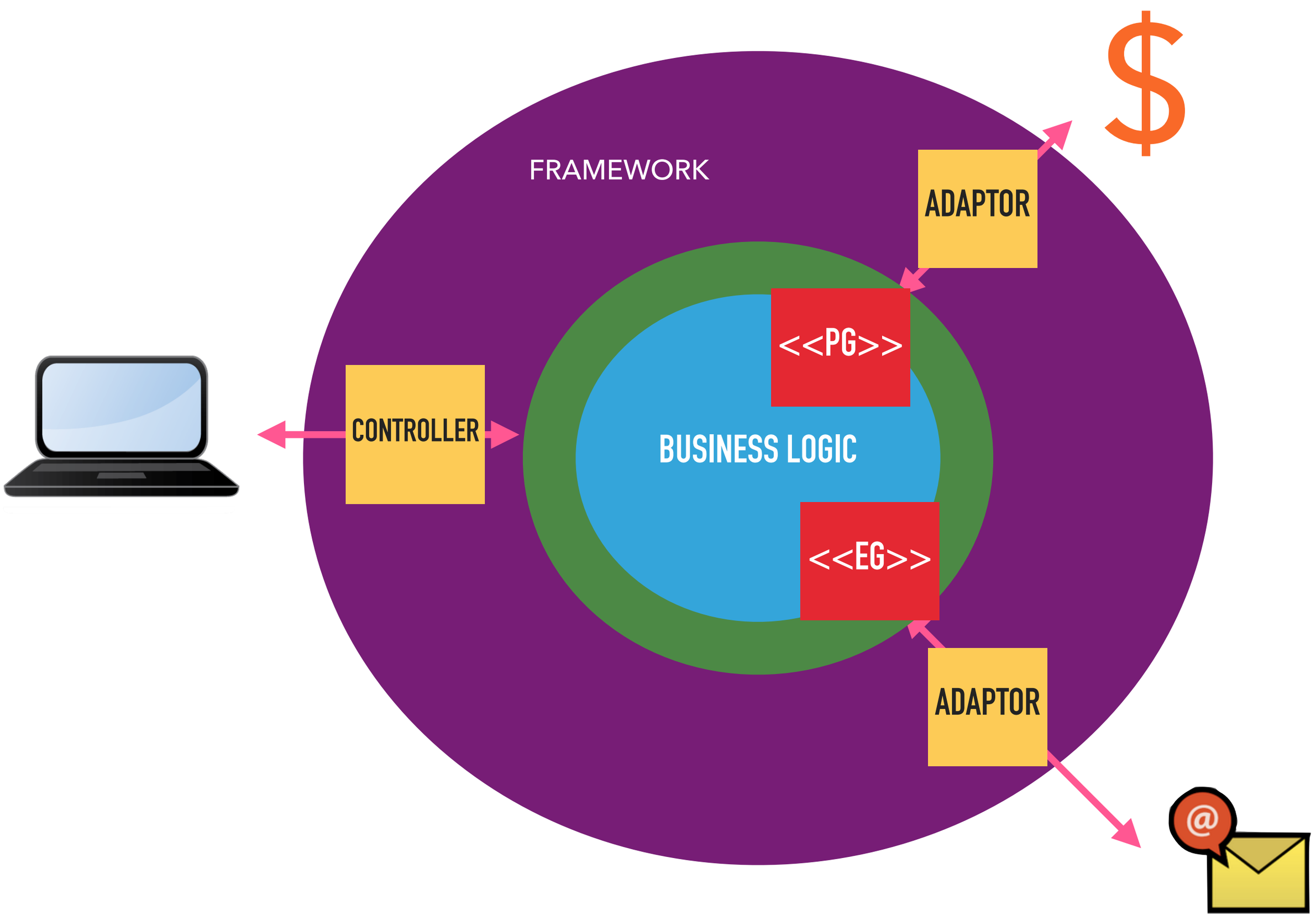
STORY 2


FRAMEWORK
ADAPTOR
$
<<PG>>
CONTROLLER
BUSINESS LOGIC
<<EG>>
ADAPTOR

EMAIL GATEWAY

```
interface EmailGateway
{

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

}
```

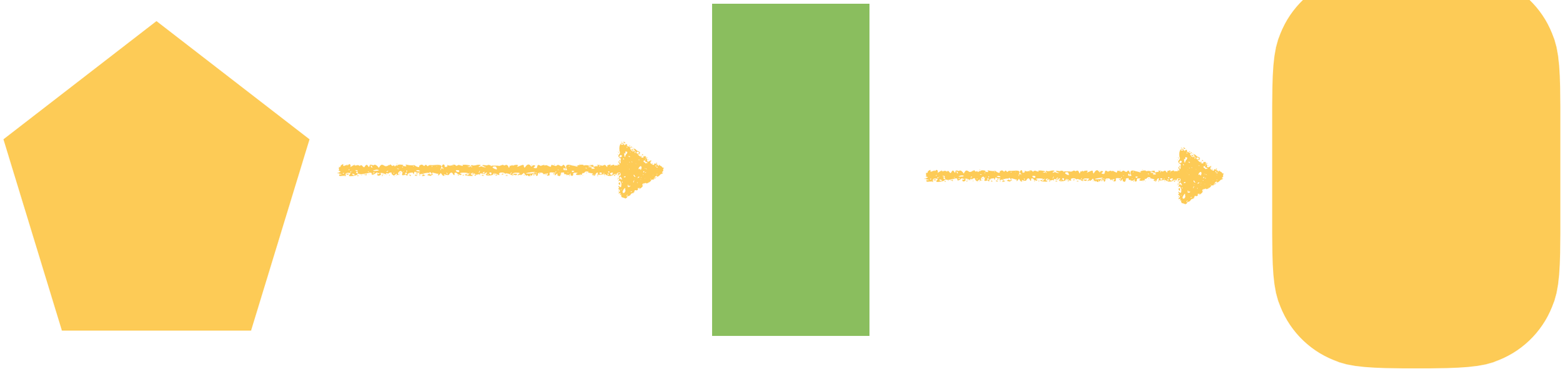
STORY 2


FRAMEWORK
ADAPTOR
$
<<PG>>
CONTROLLER
BUSINESS LOGIC
<<EG>>
ADAPTOR

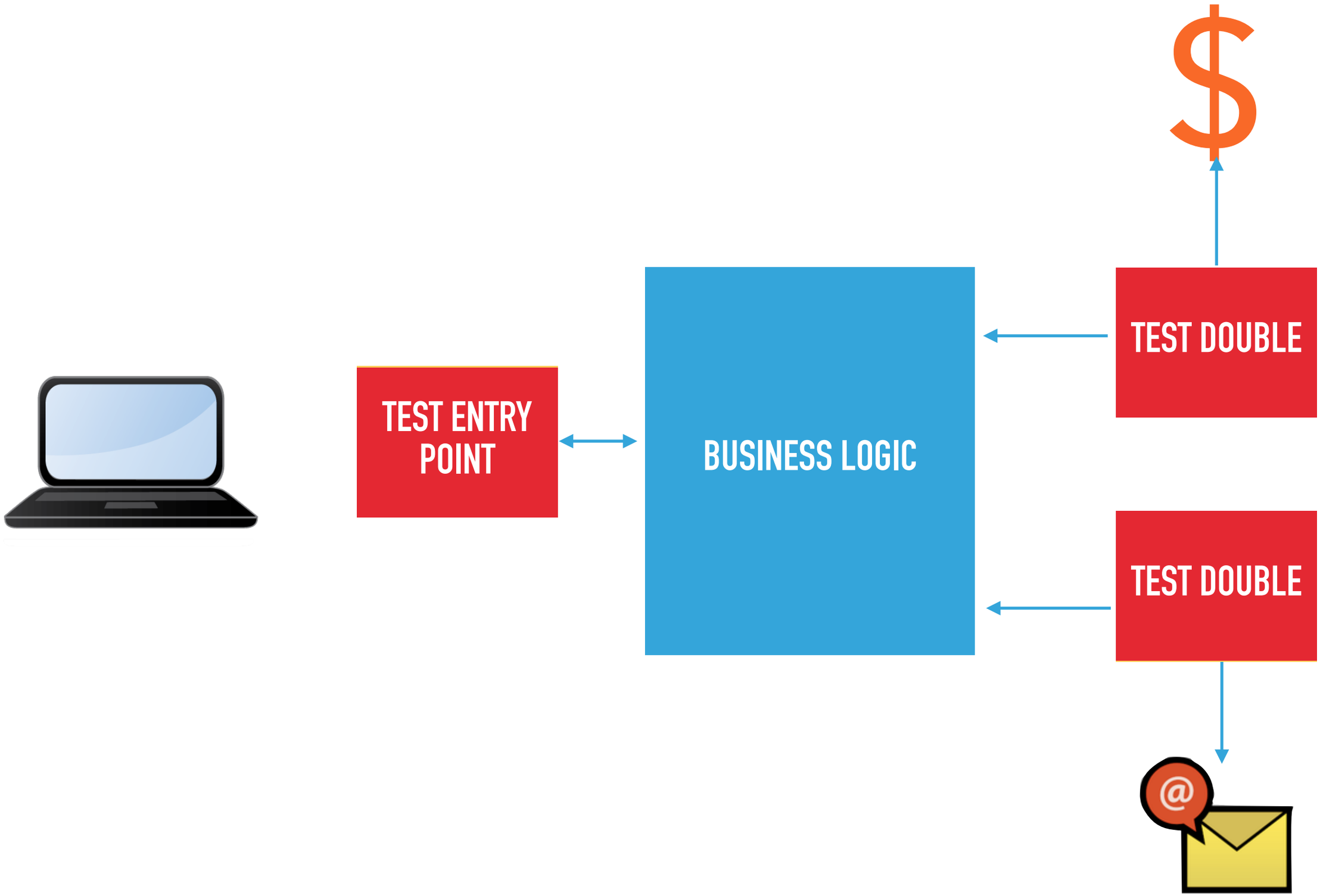
EMAIL GATEWAY TEST IMPLEMENTATION

```
EmailGatewaySpy implements EmailGateway
{

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

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

}
```

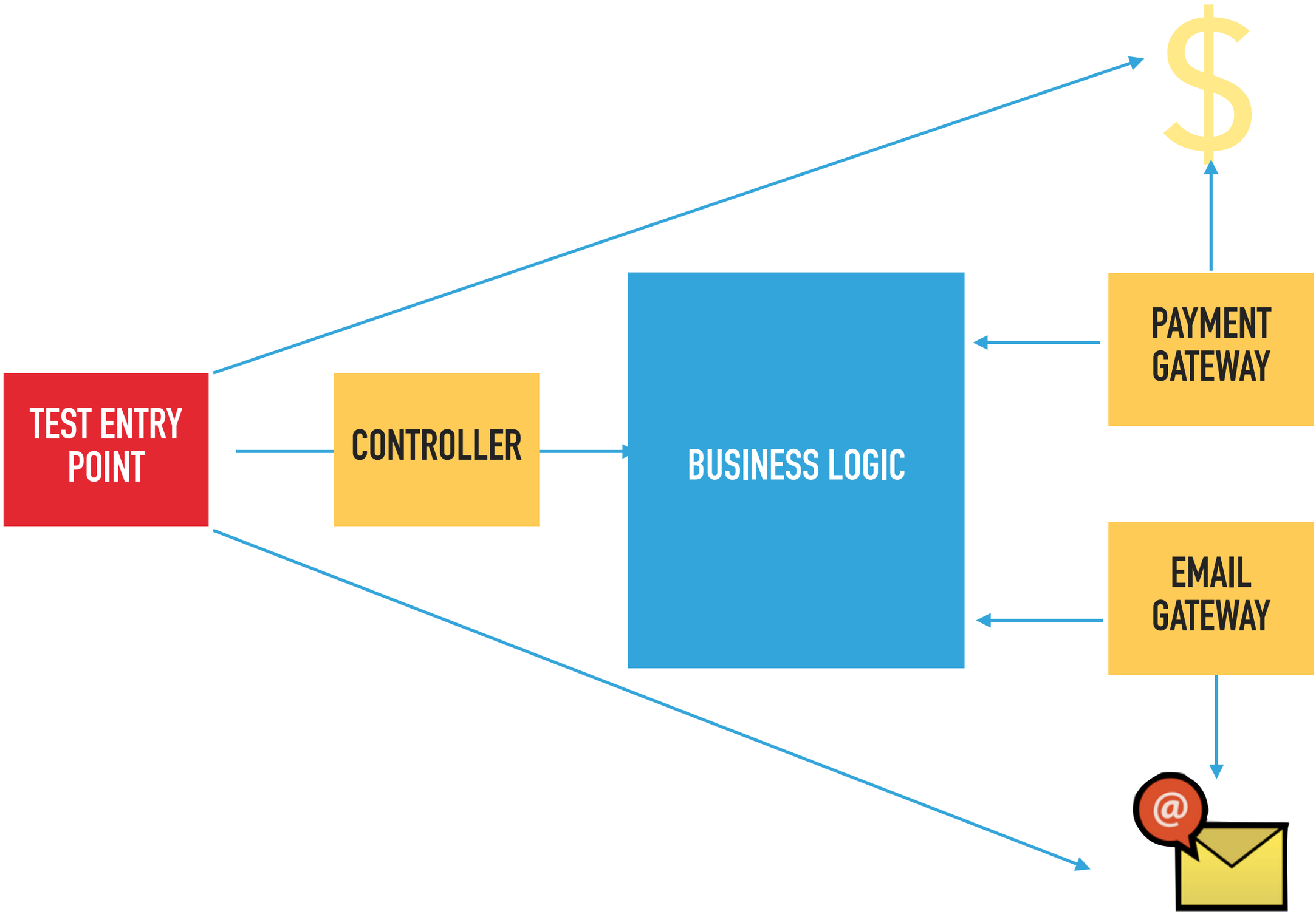
TESTING IS EASIER



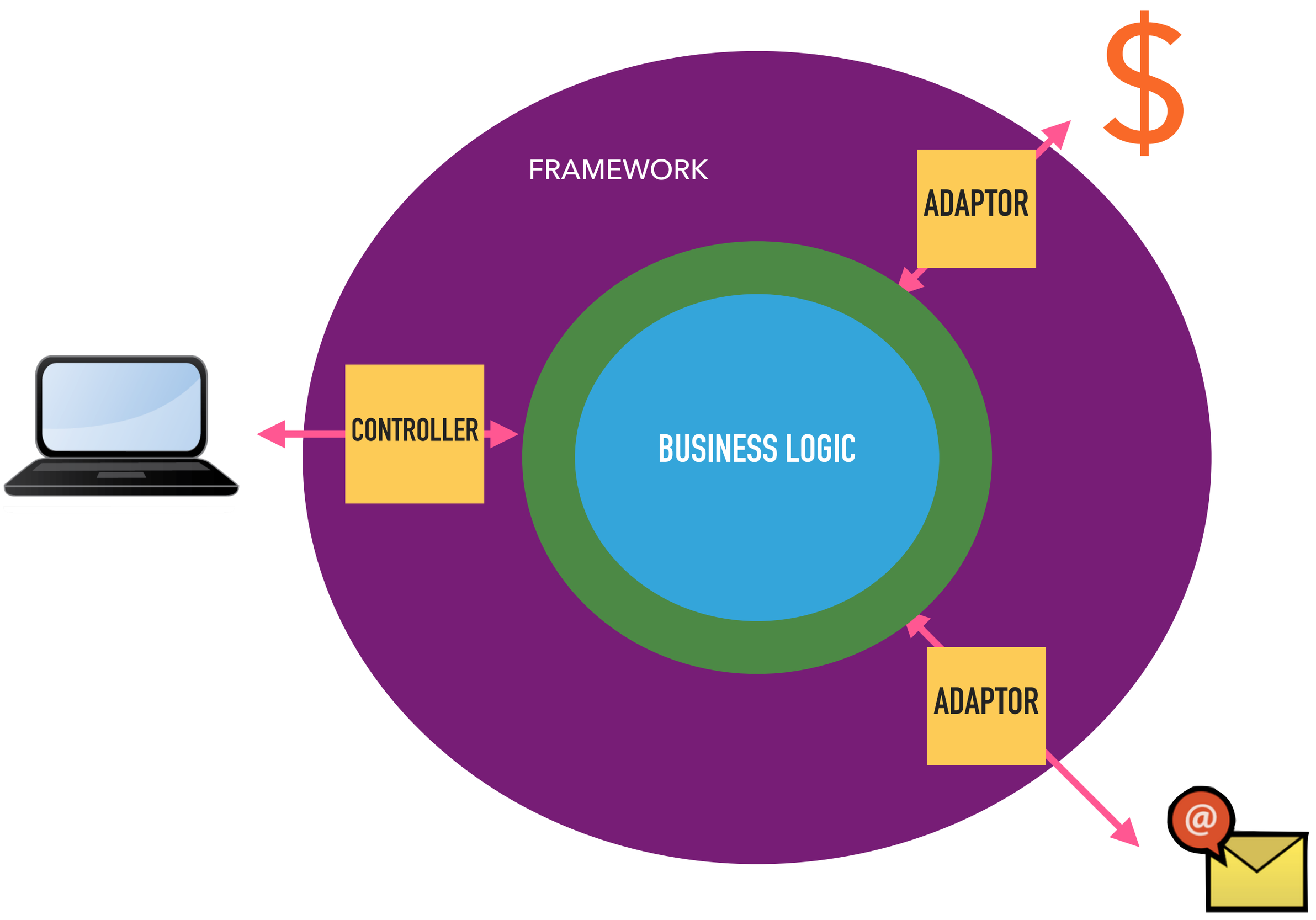
STORY 2



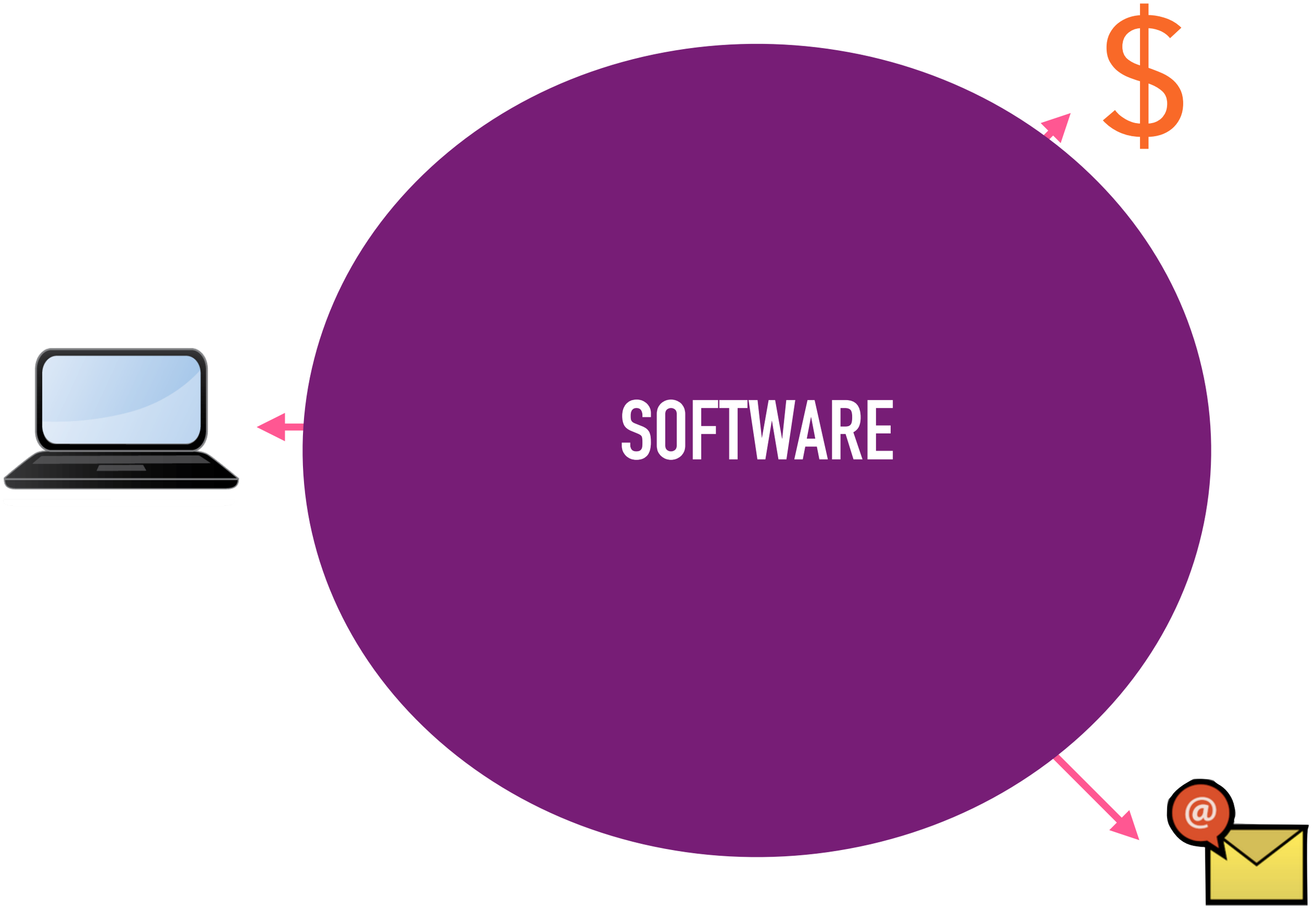
STORY 2


TEST ENTRY POINT
CONTROLLER
BUSINESS LOGIC
PAYMENT GATEWAY
EMAIL GATEWAY
@

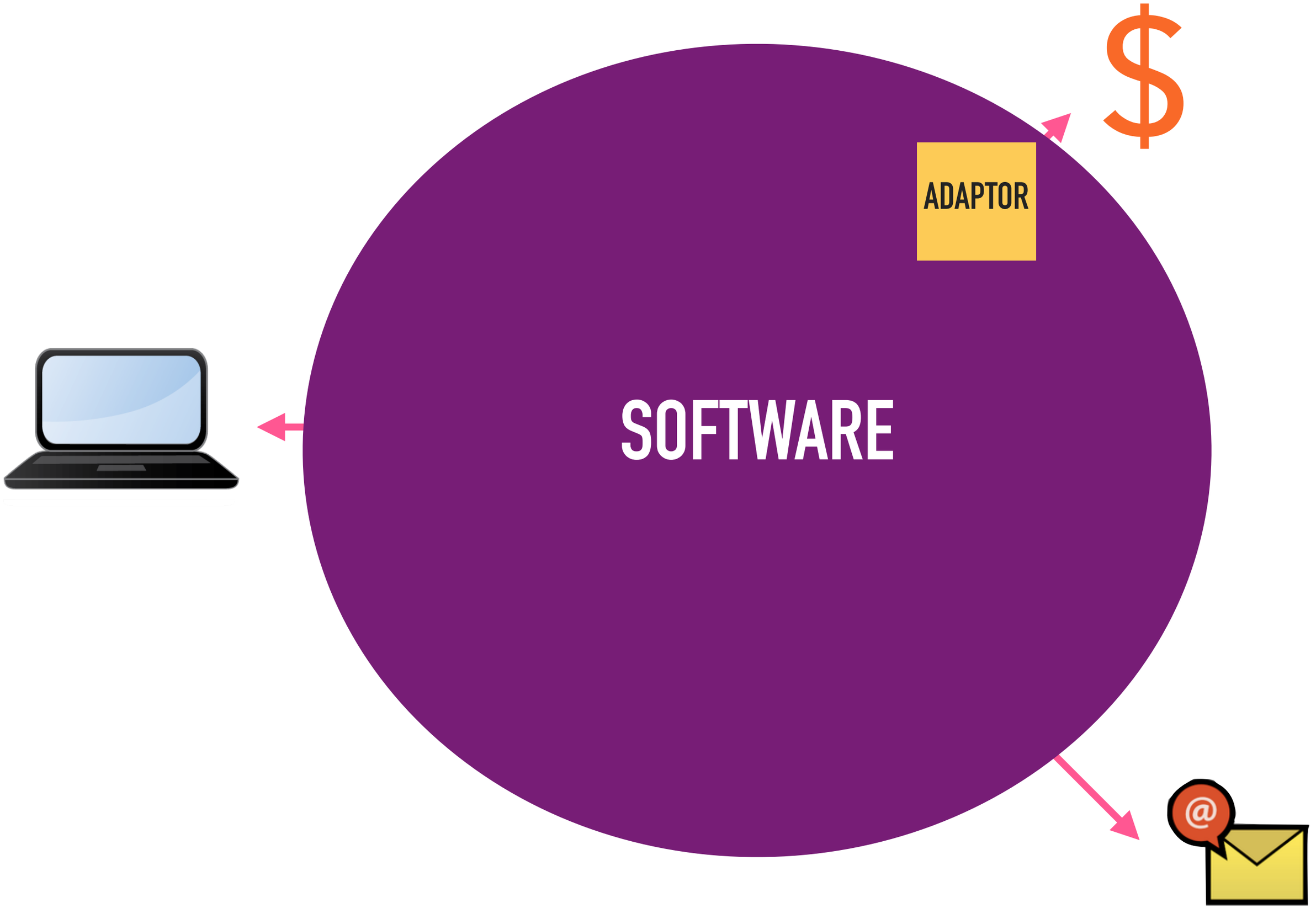
STORY 2

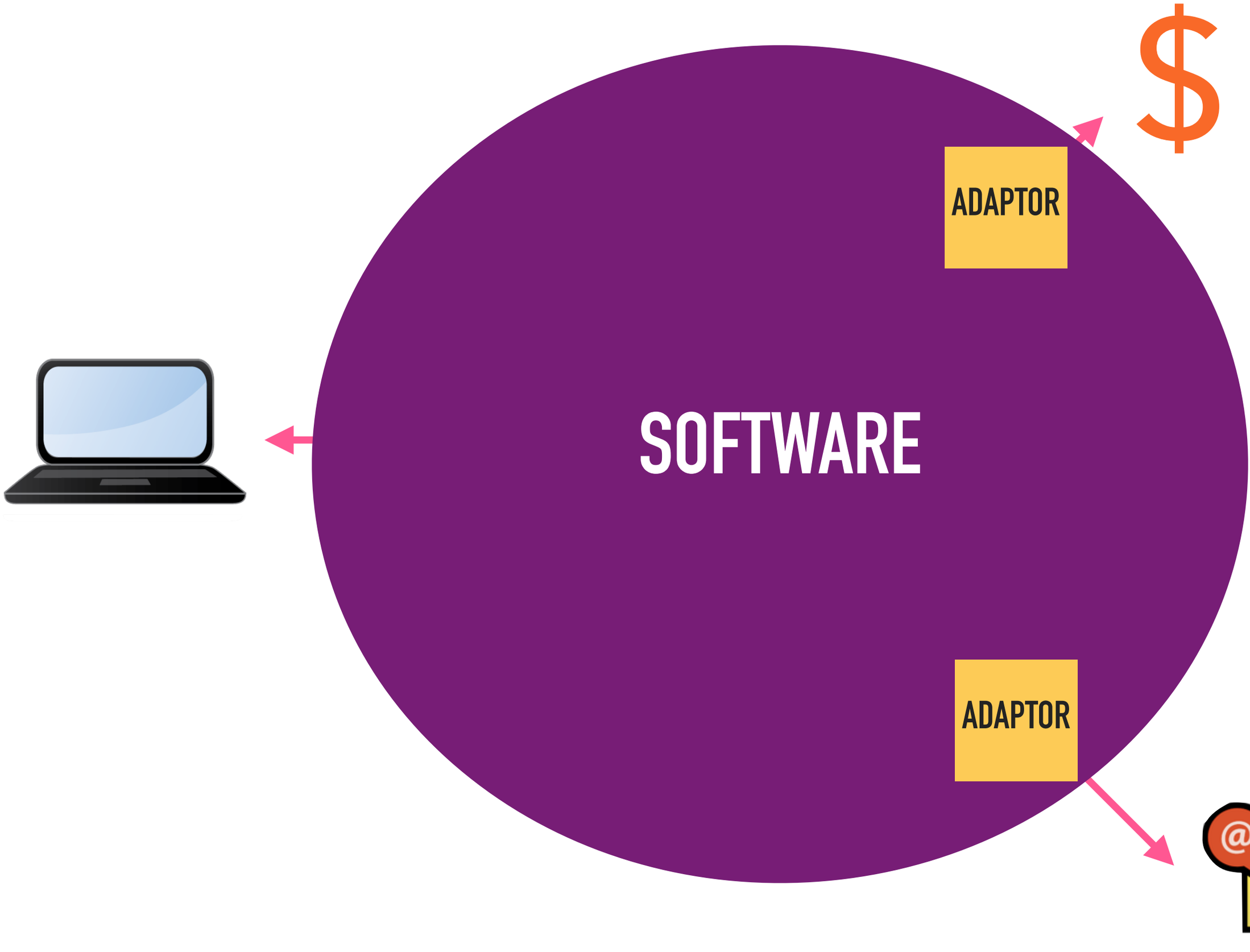


STORY 2

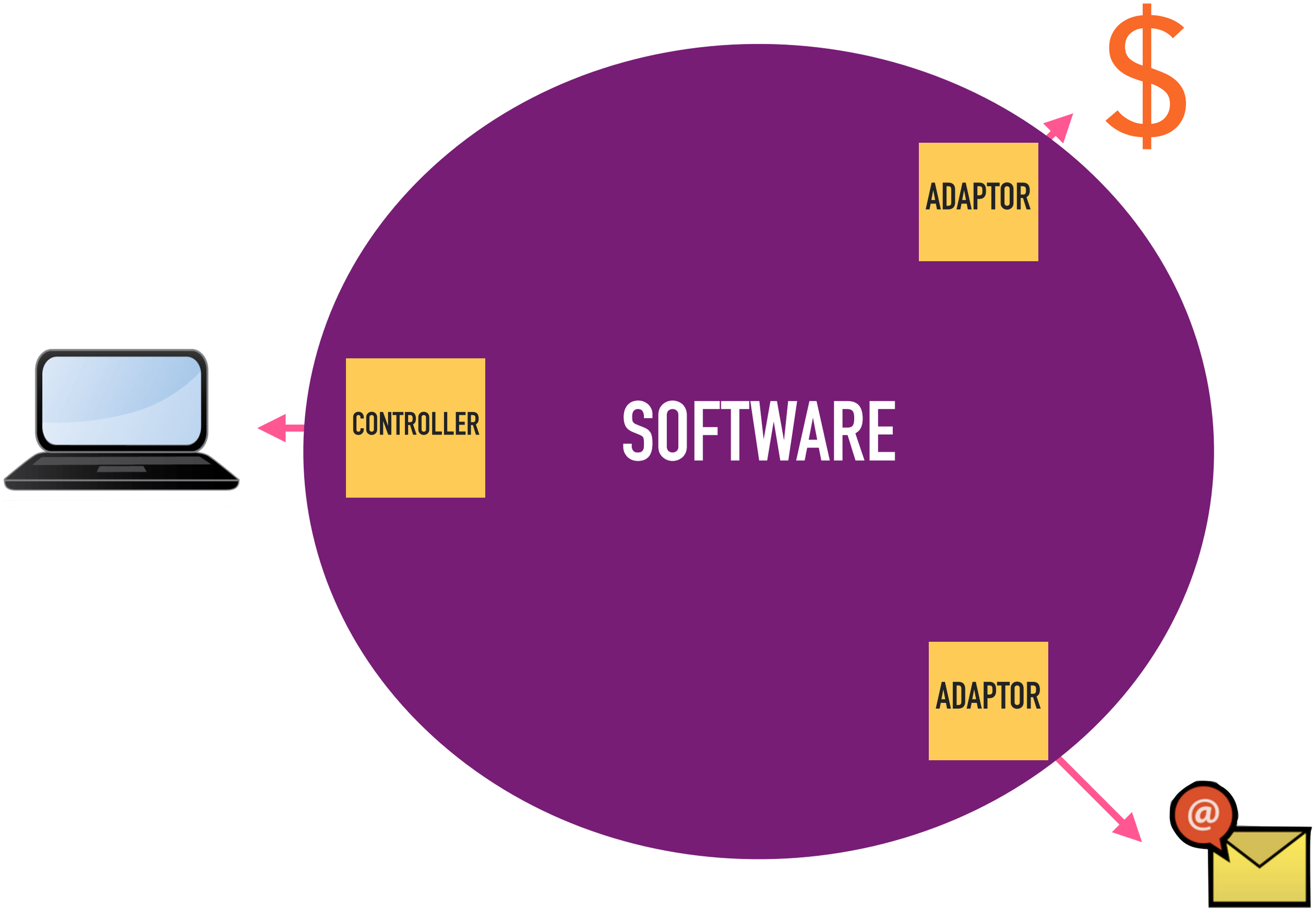


STORY 2

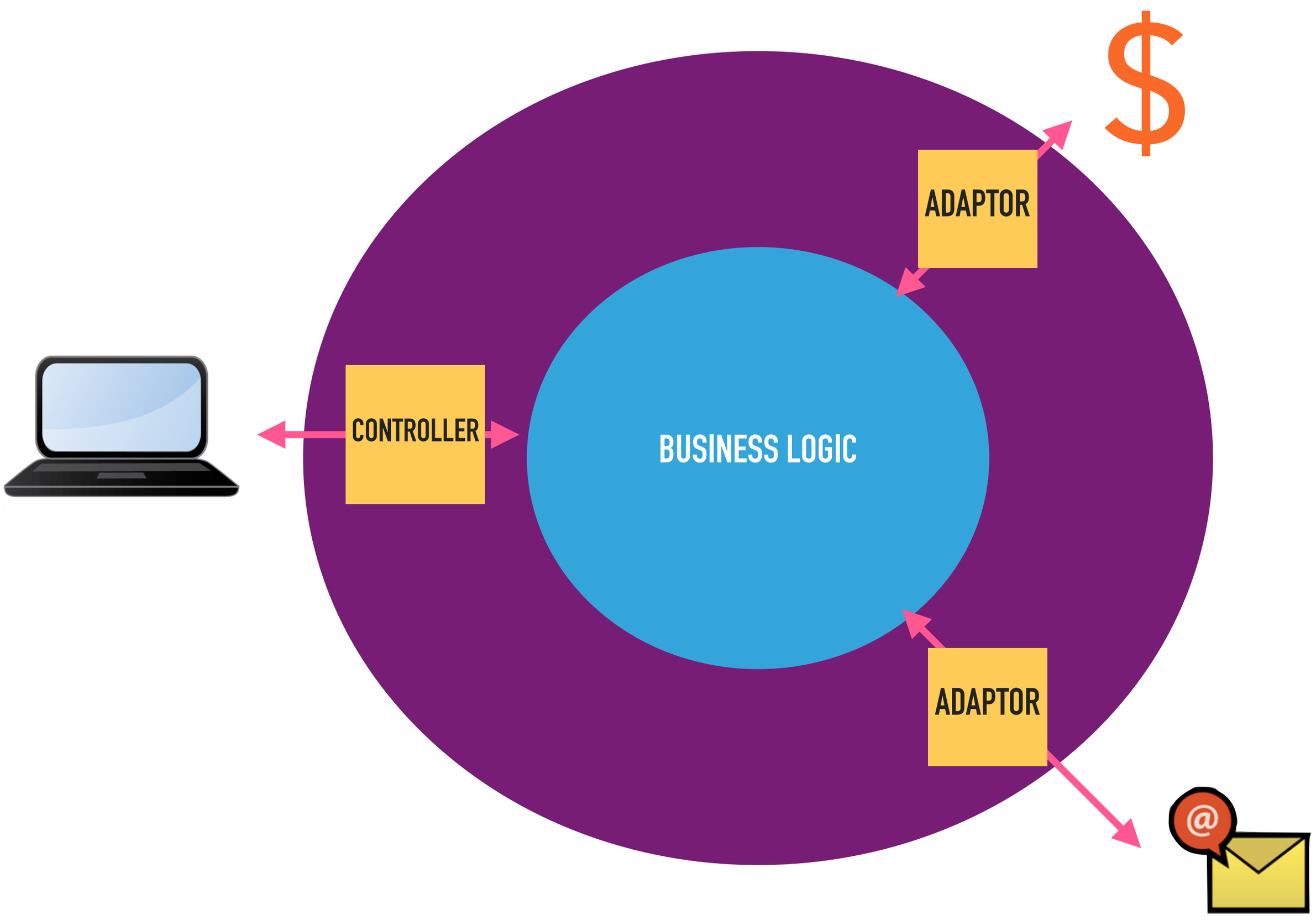



$
ADAPTOR
SOFTWARE
ADAPTOR

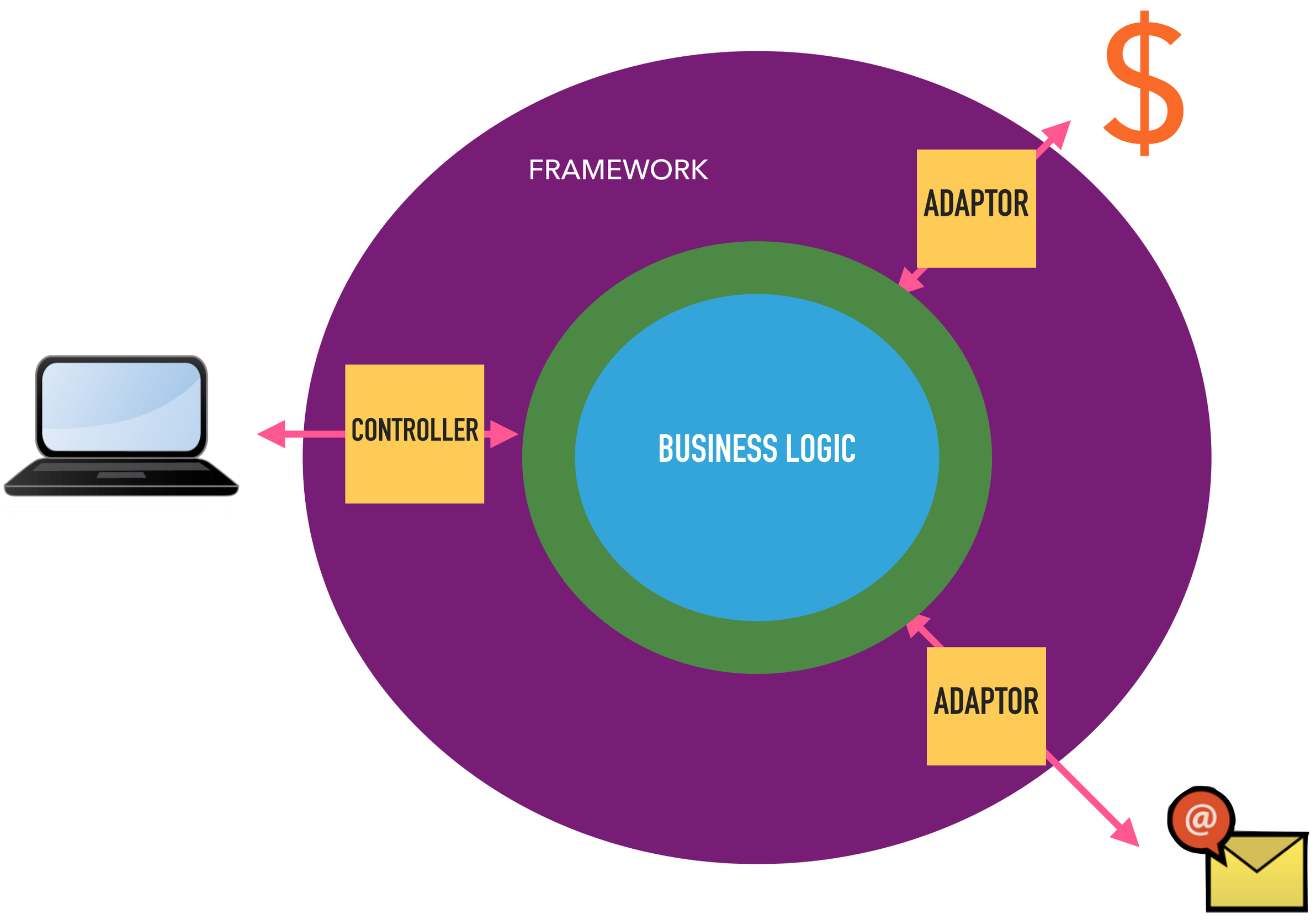
STORY 2



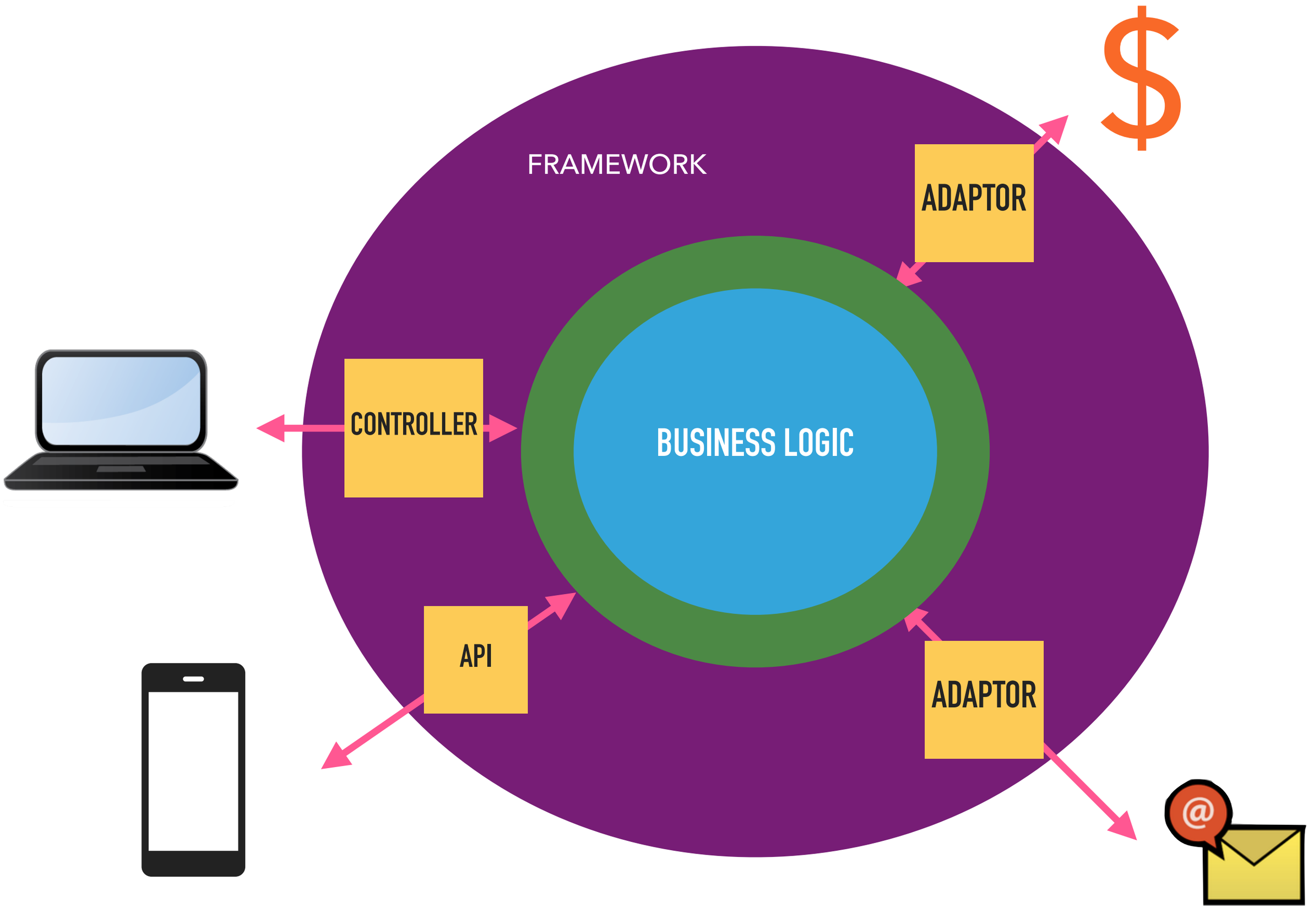
STORY 2



STORY 2



STORY 2



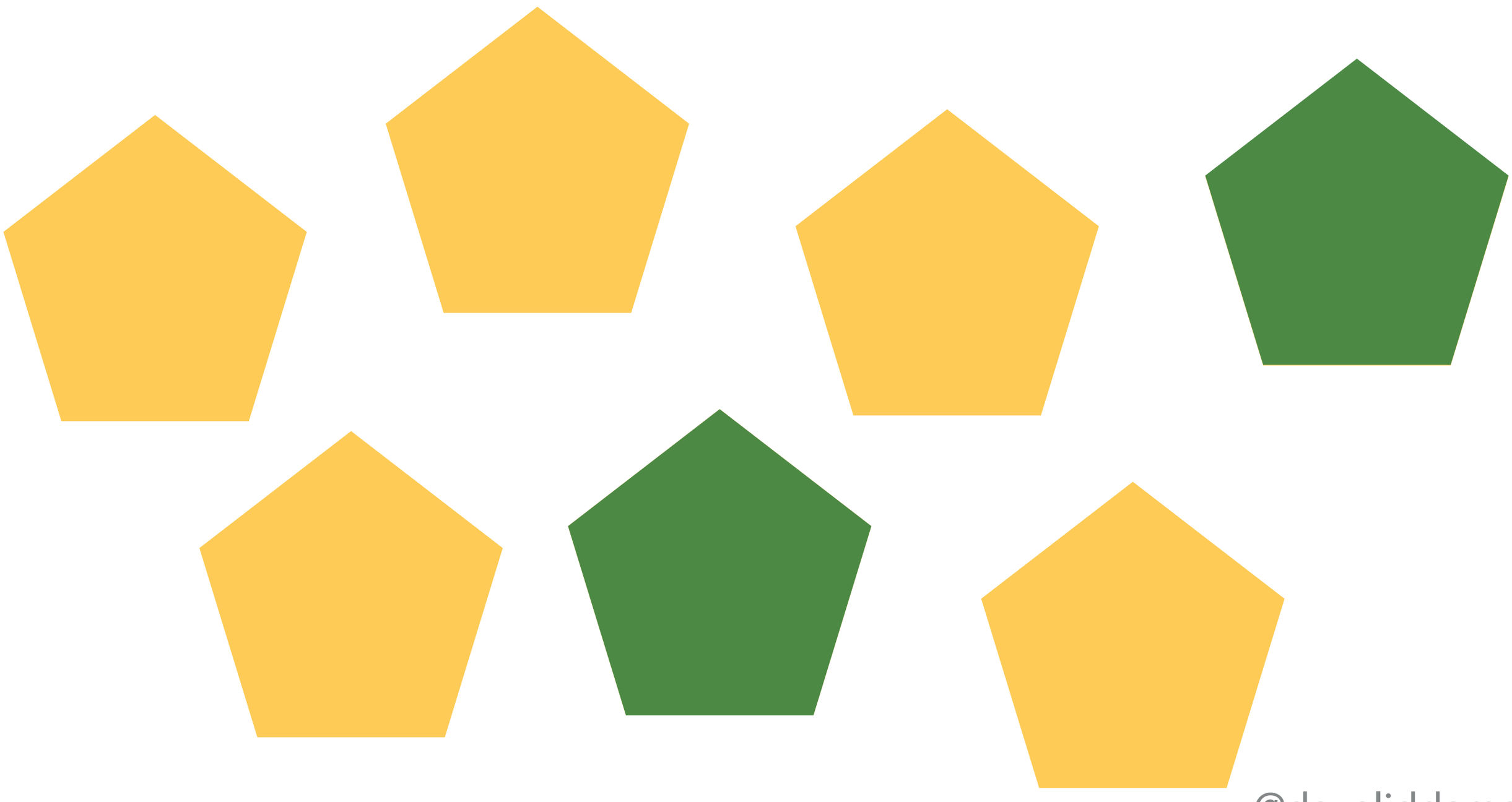
STORY 2

Integration

?

UI

WHAT DO WE TEST AT THE 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.
- I really like doing this kind of testing!

DECOUPLED 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

@daveliddament

STORY 3

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

```
..................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.  63 / 444 ( 14%)
.................................................................. 126 / 444 ( 28%)
.........................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.......... 189 / 444 ( 42%)
FFFFFFFFFFFF.............................................................. 252 / 444 ( 56%)
..................FF....FFFF.......FFFFFFFFFFFFFF.............. 315 / 444 ( 70%)
..............................FFFFFFFFFFFFFFFFFFFFFFFFFFFF.. 378 / 444 ( 85%)
FFFFFFFFFFFFFF........FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 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
```

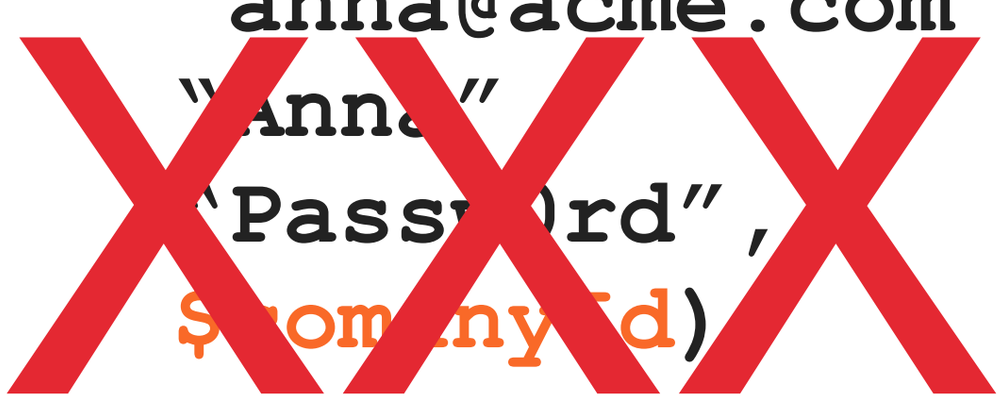
BUILDING DATA FIXTURES

HAND BUILDING

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

HAND BUILDING

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



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;
}
```

HYBRID

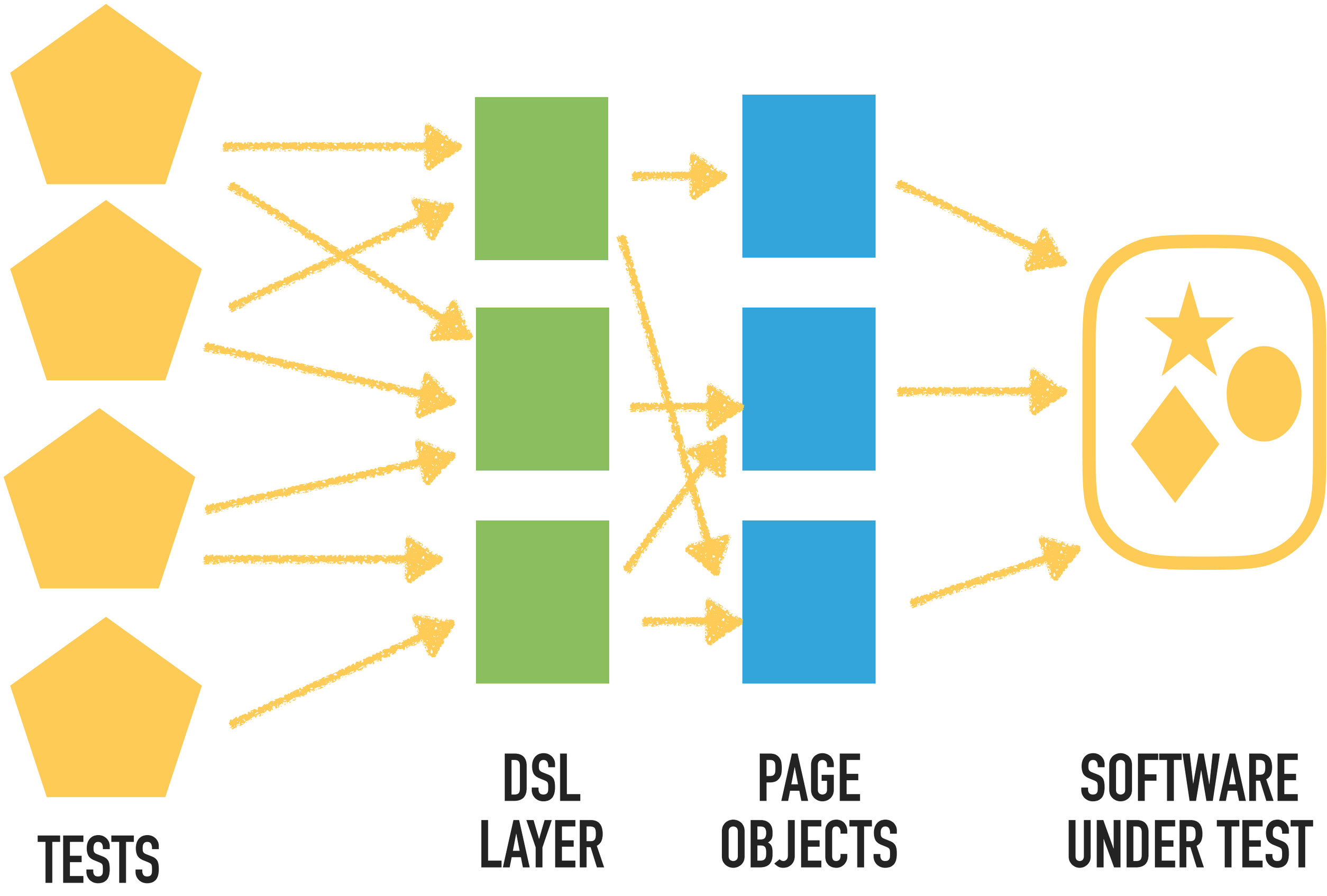
```
users:
   - name: Anna
     email: anna@acme.com
     password: Passw1rd
     team: Apple

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

MORAL OF STORY 3...

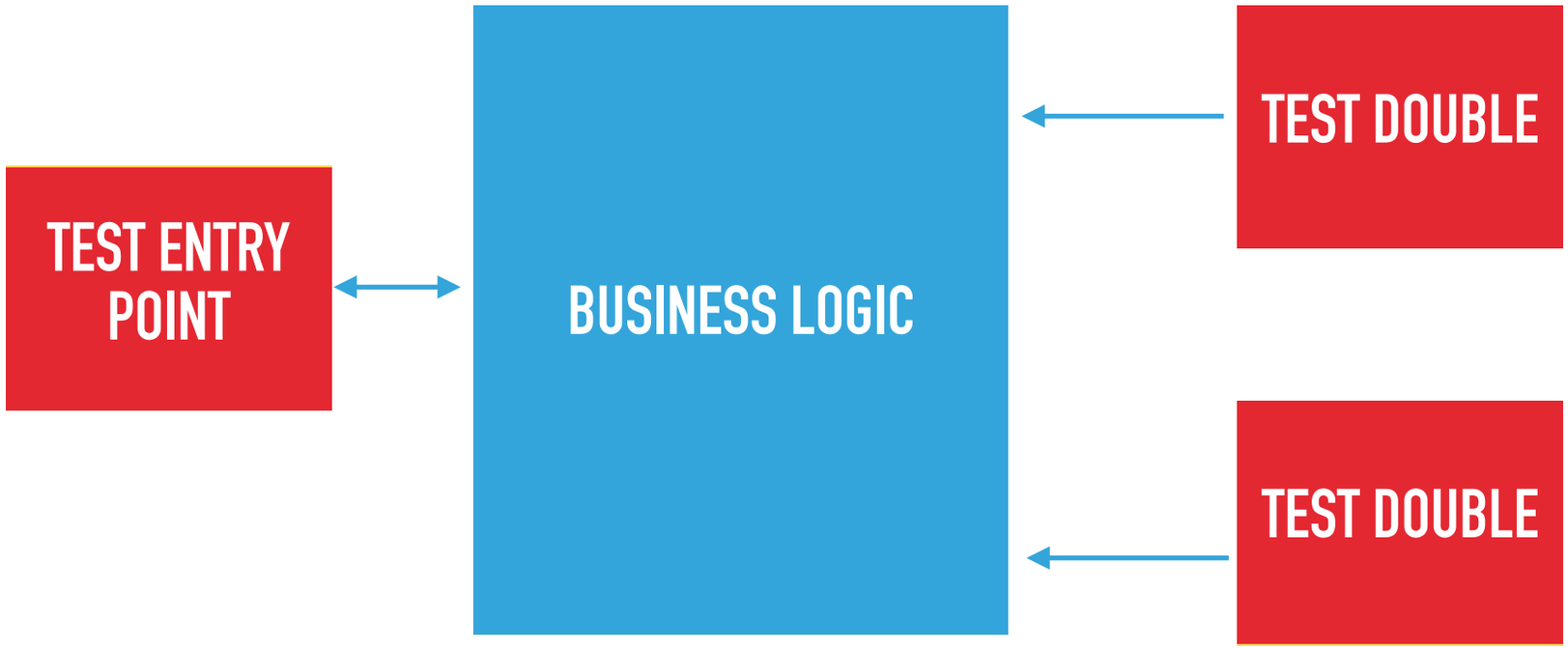
- Use patterns like Object Mothers / Test Builders for building data fixtures.
  - Makes tests more robust to change.
- Decoupling our tests from the software under test.

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

STORY 1



STORY 2



STORY 3

VALUE OF TESTS =

COST OF BUGS FOUND BY TESTS

– COST OF TEST SUITE

VALUE OF TESTS =

COST OF BUGS FOUND BY TESTS

− COST OF TEST SUITE

+ MORE ADAPTABLE CODEBASE

Dave Liddament

@daveliddament

@daveliddament.bsky.social

github.com/DaveLiddament

www.linkedin.com/in/daveliddament/

Me when I'm not coding!