ThoughtWorks®

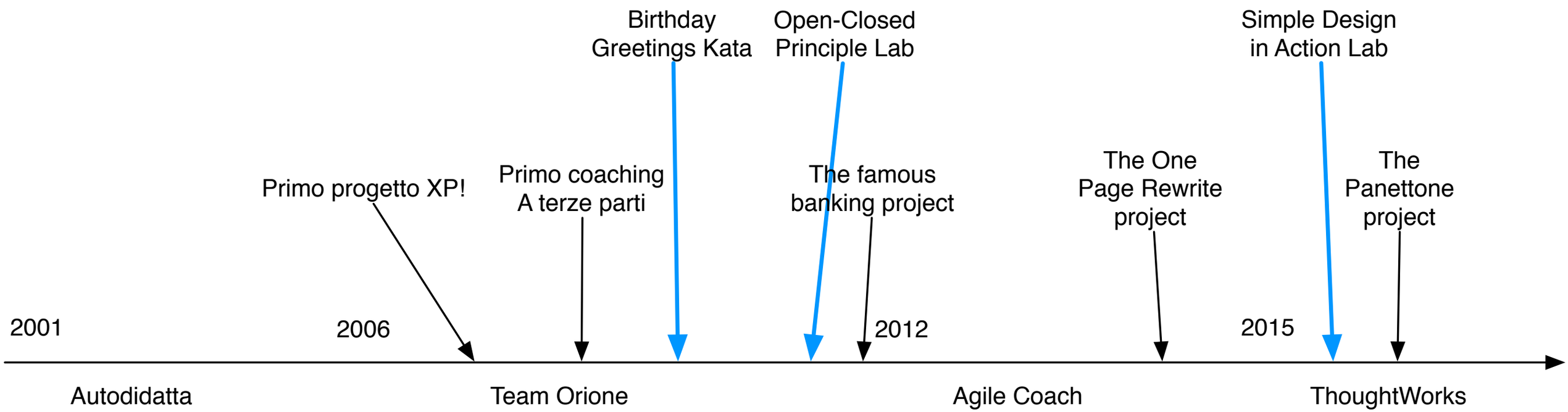
TDD: COSE CHE HO IMPARATO

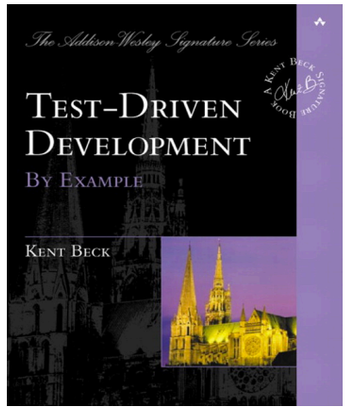
Matteo Vaccari

UNA SFIDA:

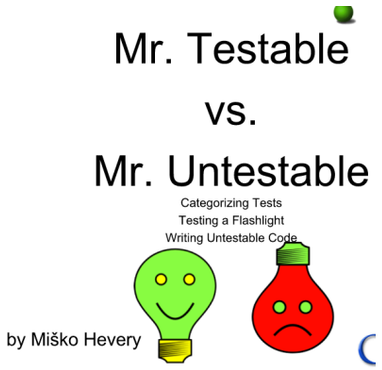
Scrivi un programma che produce la lista dei numeri da 0 a 1 milione

Come lo testi?



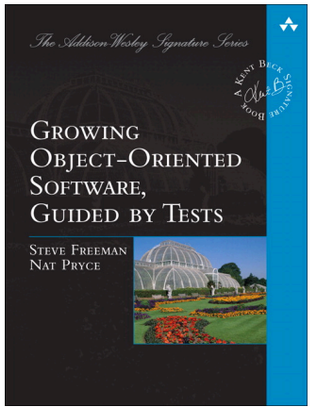


Kent Beck



Miško Hevery
Testable Code videos

Workshop
Design Emergente
Francesco Cirillo



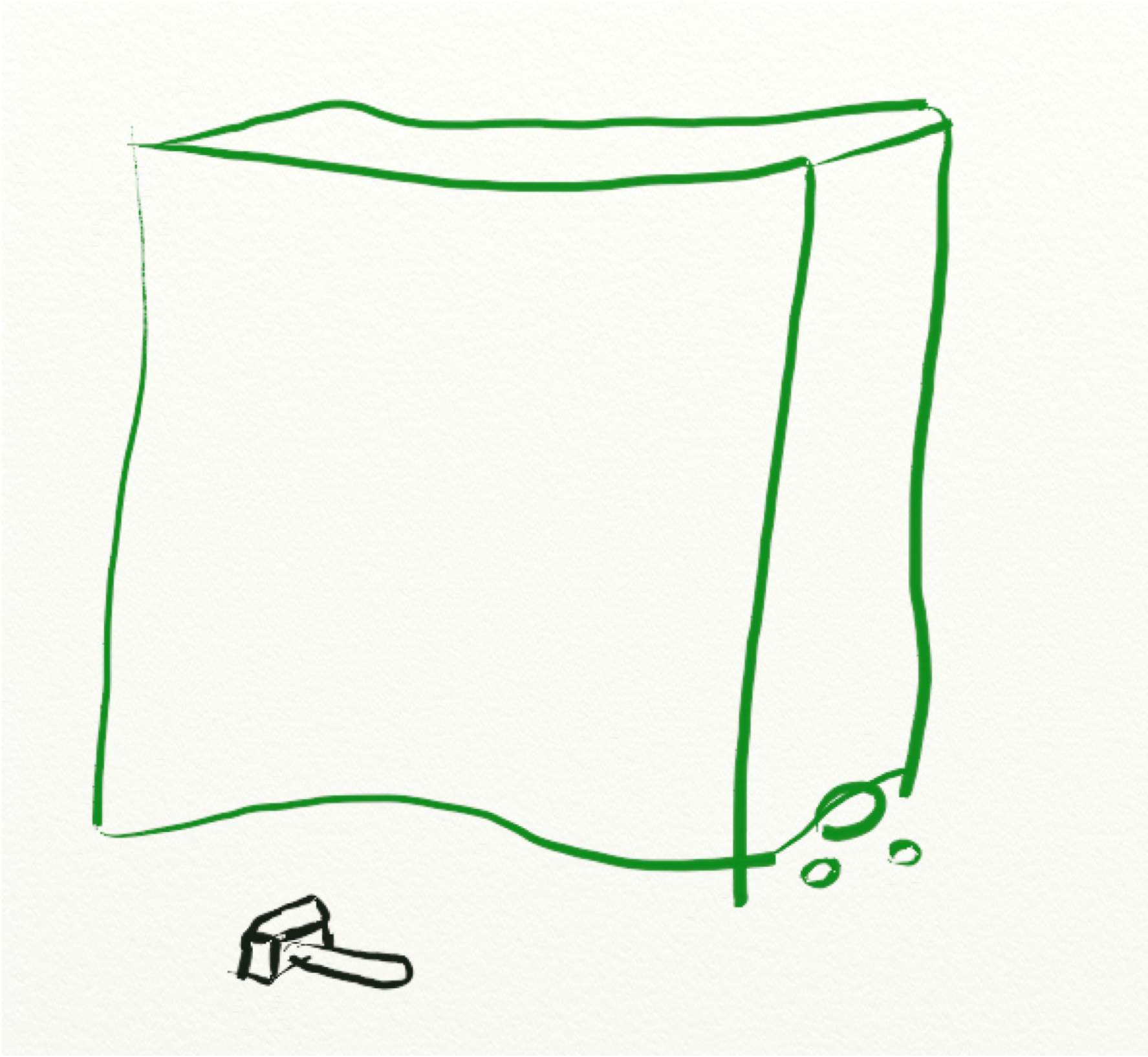
Freeman & Pryce

DOMANDE

- Il design emergente esiste davvero?
- Il TDD richiede più tempo?
- Come si fa con la UI?
- Come si fa con il DB?
- Mock sì, mock no?
- Posso fidarmi di Uncle Bob?
- Il TDD è morto?!?

Altre domande? Le tue domande?

1. COME INIZIARE



TEST LIST

1. **Esempi**
2. Del **problema** da risolvere

Pag. 3 di "TDD By Example" di Kent Beck

Multi-Currency Money

We'll start with the object that Ward created at WyCash, multi-currency money (refer to the Introduction). Suppose we have a report like this:

Instrument	Shares	Price	Total
IBM	1000	25	25000
GE	400	100	40000
		Total	65000

To make a multi-currency report, we need to add currencies:

Instrument	Shares	Price	Total
IBM	1000	25 USD	25000 USD
Novartis	400	150 CHF	60000 CHF
		Total	65000 USD

We also need to specify exchange rates:

From	To	Rate
CHF	USD	1.5

$5 + 10 CHF = $10 if rate is 2:1
$5 * 2 = $10

$5 + 10CHF = $10 if rate is 2:1
$5 * 2 = $10

UNA SFIDA:

Scrivi un programma che produce la lista dei numeri da 0 a 1 milione

Come lo testi?

VARI TENTATIVI/1

```
@Test
public void onlyExtremes() {
  List result = generateZeroToOneMillion();
  assertEquals(0, result.get(0));
  assertEquals(1, result.get(1));
  assertEquals(999999, result.get(999999));
  assertEquals(1000000, result.get(1000000));
}
```

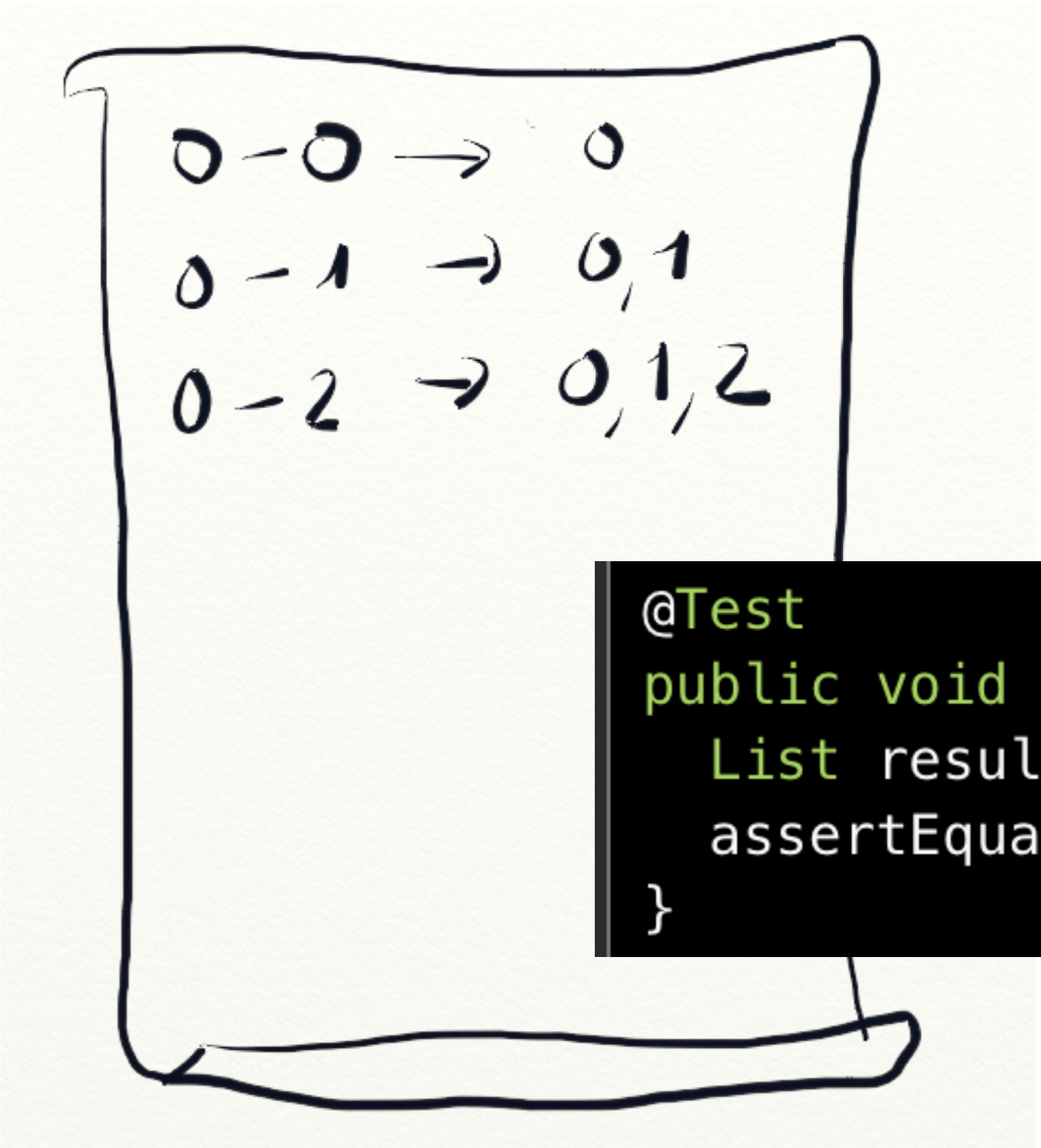
VARI TENTATIVI/2

```
@Test
public void algorithmically() {
  List result = generateZeroToOneMillion();
  for (int i=0; i<=1000000; i++) {
    assertEquals(i, result.get(i));
  }
}
```

VARI TENTATIVI/3

```
@Test
public void compareWithExternalFiles() {
  BufferedReader reader = readExternalFile("testData.txt")
  List result = generateZeroToOneMillion();
  while ((line = reader.readLine()) != null)
    assertEquals(parseInt(line), result.get(i));
  }
}
```

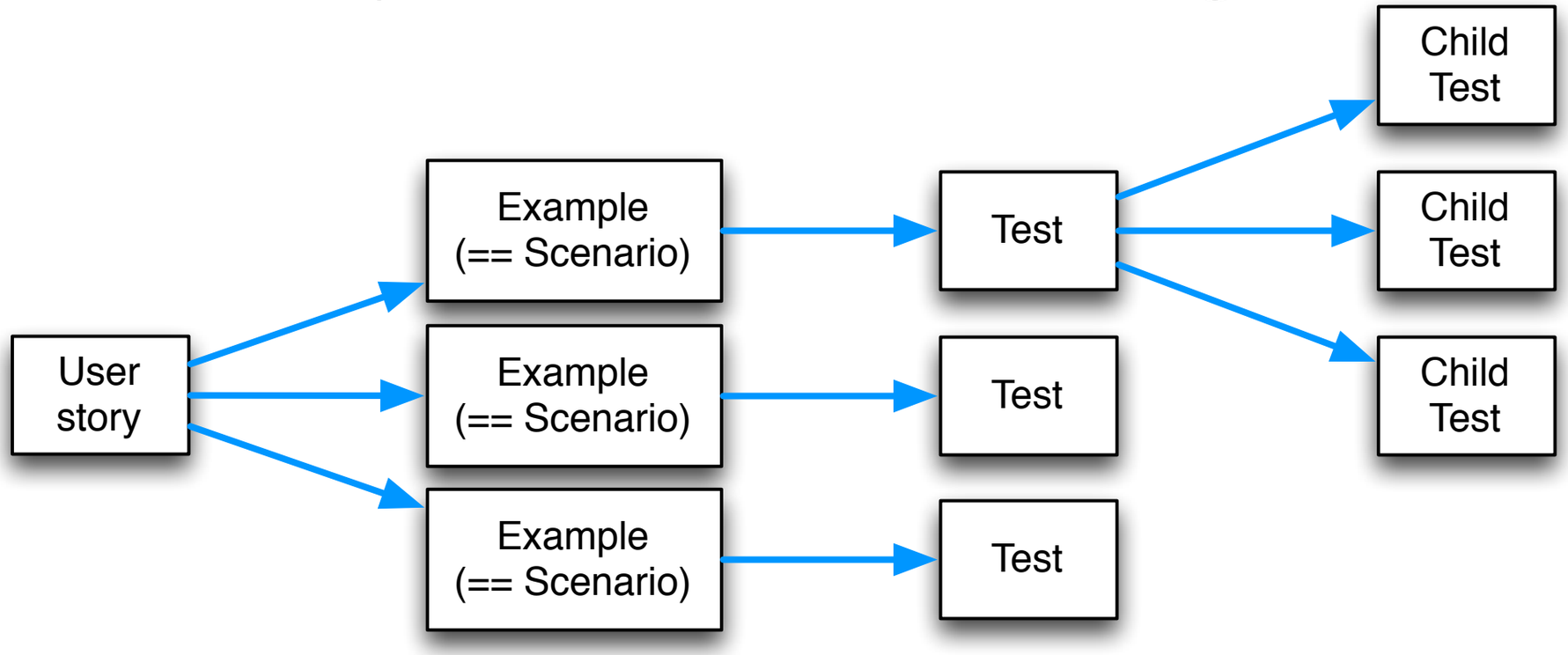
GENERALIZE AND SIMPLIFY



```
@Test
public void zeroToTwo() {
  List result = generate(0, 2);
  assertEquals(asList(0,1,2), result);
}
```

LA TEST LIST DEVE:

- ☐ Contenere esempi
- ☐ Del problema da risolvere
- ☐ Che coprano tutte le variazioni (cioè gli scenari)
- ☐ Senza implicare soluzioni di design



DA QUALE TEST PARTIRE?

A. Da un test sulla UI? (outside-in)

B. Da un test sulla logica di dominio? (inside-out)

C. ?

Red Bar Patterns

These patterns are about when you write tests, where you write tests, and when you stop writing tests.

One Step Test

Which test should you pick next from the list? Pick a test that will teach you something and that you are confident you can implement.

Each test should represent one step toward your overall goal. Looking at the following Test List, then, which test should we pick next?

Plus
Minus
Times
Divide

Which test should you pick next from the list?

Pick a test that

- *will teach you something*
- *you are confident you can implement*

"From known to unknown"

TDD By Example, p. 133

2. ASSI ORTOGONALI

How should the running of tests affect one another? Not at all.

— Kent Beck, TDD By Example

If I had one test broken, I wanted one problem. If I had two tests broken, I wanted two problems.

— Kent Beck, TDD By Example

A second implication of isolated tests is that you have to work, sometimes work hard, to break your problem into little orthogonal dimensions.

— Kent Beck, TDD By Example

Questa gemma è molto ben nascosta in fondo a pag. 125 di "TDD By Example"

Isolated Test

How should the running of tests affect one another? Not at all.

When I was a young programmer—long, long ago when we had to dig our own bits out of the snow and carry heavy buckets of them barefooted back to our cubicles, leaving bloody little footprints for the wolves to follow. . . . Sorry, just reminiscing. My first experience of automated tests was having a set of long-running, overnight, GUI-based tests (you know, record the keystrokes and mouse events and play them back) for a debugger I was working on. (Hi Jothy, hi John!) Every morning when I came in, there would be a neat stack of paper on my chair describing last night's test runs. (Hi Al!) On good days there would be a single sheet summarizing that nothing broke. On bad days there would be many, many sheets, one for each broken test. I began to dread days when I saw a pile of paper on my chair.

I took two lessons from this experience. First, make the tests so fast to run that I can run them myself, and run them often. That way I can catch errors before anyone else sees them, and I don't have to dread coming in in the morning. Second, I noticed after a while that a huge stack of paper didn't usually mean a huge list of problems. More often it meant that one test had broken early, leaving the system in an unpredictable state for the next test.

We tried to get around this problem by starting and stopping the system between each test, but it took too long, which taught me another lesson about seeking tests at a smaller scale than the whole application. But the main lesson I took was that tests should be able to ignore one another completely. If I had one test broken, I wanted one problem. If I had two tests broken, I wanted two problems.

One convenient implication of isolated tests is that the tests are order independent. If I want to grab a subset of tests and run them, then I can do so without worrying that a test will break now because a prerequisite test is gone.

Performance is the usual reason cited for having tests share data. A second implication of isolated tests is that you have to work, sometimes work hard, to break your problem into little orthogonal dimensions, so setting up the environment for each test is easy and quick. Isolating tests encourages you to compose solutions out of many highly cohesive, loosely coupled objects. I always heard this was a good idea, and I was happy when I achieved it, but I never knew

UN ESEMPIO DI TEST NON ISOLATI

```
public void testSingleNewReleaseStatement() {
  customer.addRental(new Rental(new Movie("The Cell", Movie.NEW_RELEASE), 3));
  String expected = "Rental Record for Fred\n\tThe Cell\t9.0\n" +
    "You owed 9.0\nYou earned 2 frequent renter points\n";
  assertEquals(expected, customer.statement());
}

public void testDualNewReleaseStatement() {
  customer.addRental(new Rental(new Movie("The Cell", Movie.NEW_RELEASE), 3));
  customer.addRental(new Rental(new Movie("The Tigger Movie", Movie.NEW_RELEASE), 3));
  String expected = "Rental Record for Fred\n\tThe Cell\t9.0\n" +
    "\tThe Tigger Movie\t9.0\nYou owed 18.0\nYou earned 4 frequent renter points\n";
  assertEquals(expected, customer.statement());
}

public void testSingleChildrensStatement() {
  customer.addRental(new Rental(new Movie("The Tigger Movie", Movie.CHILDRENS), 3));
  String expected = "Rental Record for Fred\n\tThe Tigger Movie\t1.5\n" +
    "You owed 1.5\nYou earned 1 frequent renter points\n";
  assertEquals(expected, customer.statement());
}
```

Martin Fowler, *Refactoring*

```java
public void testSingleChildrensStatement() {
  customer.addRental(new Rental(new Movie("The Tigger Movie", Movie.CHILDRENS), 3))
  String expected = "Rental Record for Fred\n\tThe Tigger Movie\t1.5\n" +
    "You owed 1.5\nYou earned 1 frequent renter points\n";
  assertEquals(expected, customer.statement());
}
```

```java
@Test
public rentFor_childrensMovie_notOverdue() {
  Customer customer = new Customer();
  int days = 3;
  Movie movie = new Movie("The Tigger Movie", Movie.CHILDRENS);
  Rental rental = new Rental(movie, days));

  customer.addRental(rental);

  assertEquals(3.99, customer.statementTotal());
}
```

```java
@Test
public rentFor_childrensMovie_notOverdue() {
  Customer customer = new Customer();
  int days = 3;
  Movie movie = new Movie("The Tigger Movie", Movie.CHILDRENS);
  Rental rental = new Rental(movie, days));

  customer.addRental(rental);

  assertEquals(3.99, customer.statementTotal());
}
```

```java
@Test
public rentFor_childrensMovie_notOverdue() {
  int days = 3;
  Movie movie = new Movie("The Tigger Movie", Movie.CHILDRENS);
  Rental rental = new Rental(movie, days));
  assertEquals(3.99, rental.rentalAmount());
}
```

UN ALTRO ESEMPIO DI TEST NON ISOLATI

Password-strength checker

In order to be an acceptable password, a string must:

* Have a length greater than 7 characters.
* Contain at least one alphabetic character.
* Contain at least one digit.

Rob Myers, on the TDD mailing list

Password-strength checker

In order to be an acceptable password, a string must:

* **Have a length greater than 7 characters**.
* Contain at least one alphabetic character.
* Contain at least one digit.

```
@Test
public void passwordTooShort() {
  assertEqual(false, isValidPassword(""));
  assertEqual(false, isValidPassword("aaa"));
  assertEqual(false, isValidPassword("aaaaaaa"));
}

@Test
public void passwordIsGood() {
  assertEqual(true, isValidPassword("abcdefgh"));
}
```

Password-strength checker

In order to be an acceptable password, a string must:

* **Have a length greater than 7 characters**.
* **Contain at least one alphabetic character**.
* Contain at least one digit.

```
@Test
public void passwordTooShort() {
  assertEqual(false, isValidPassword(""));
  assertEqual(false, isValidPassword("aaa"));
  assertEqual(false, isValidPassword("aaaaaaa"));
}

@Test
public void passwordIsGood() {
  assertEqual(true, isValidPassword("abcdefgh"));
}

@Test
public void passwordContainsNoAlpha() {
  assertEqual(false, isValidPassword("12345678"));
}
```

Password-strength checker

In order to be an acceptable password, a string must:

* **Have a length greater than 7 characters**.
* **Contain at least one alphabetic character**.
* **Contain at least one digit**.

Ora sono invalidi per due motivi

```
@Test
public void passwordTooShort() {
  assertEqual(false, isValidPassword(""));
  assertEqual(false, isValidPassword("aaa"));
  assertEqual(false, isValidPassword("aaaaaaa"));
}

@Test
public void passwordIsGood() {
  assertEqual(true, isValidPassword("abcdefgh"));
}

@Test
public void passwordContainsNoAlpha() {
  assertEqual(false, isValidPassword("12345678"));
}

@Test
public void passwordContainsNoDigit() {
  assertEqual(false, isValidPassword("abcdefgh"));
}
```

Ora fallisce

Password-strength checker

In order to be an acceptable password, a string must:

* **Have a length greater than 7 characters**.
* **Contain at least one alphabetic character**.
* **Contain at least one digit**.

```
@Test
public void passwordTooShort() {
  assertEqual(false, isValidPassword(""));
  assertEqual(false, isValidPassword("7aa"));
  assertEqual(false, isValidPassword("9aaaaaa"));
}

@Test
public void passwordIsGood() {
  assertEqual(true, isValidPassword("1234abcd"));
}

@Test
public void passwordContainsNoAlpha() {
  assertEqual(false, isValidPassword("12345678"));
}

@Test
public void passwordContainsNoDigit() {
  assertEqual(false, isValidPassword("abcdefgh"));
}
```

Password-strength checker

In order to be an acceptable password, a string must:

* **Have a length greater than 7**
* **Contain at least one alphabetic**
* **Contain at least one digit**.
* Contain at least one uppercase
* Contain at least one lowercase

New requirement!!!

Ora tutti i vecchi test sono da rivedere!

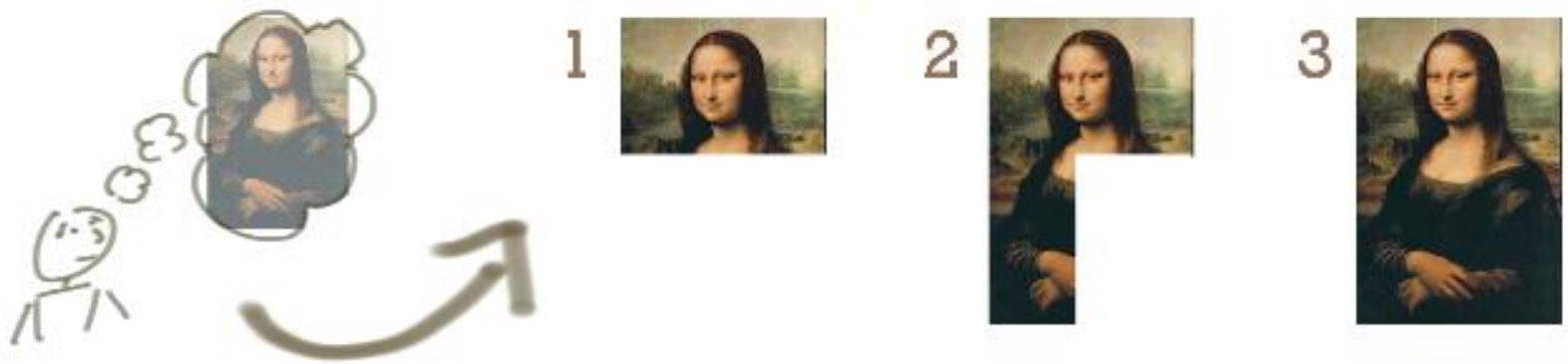
```
@Test
public void passwordTooShort() {
  assertEqual(false, isValidPassword(""));
  assertEqual(false, isValidPassword("7aa"));
  assertEqual(false, isValidPassword("9aaaaaa"));
}

@Test
public void passwordIsGood() {
  assertEqual(true, isValidPassword("1234abcd"));
}

@Test
public void passwordContainsNoAlpha() {
  assertEqual(false, isValidPassword("12345678"));
}

@Test
public void passwordContainsNoDigit() {
  assertEqual(false, isValidPassword("abcdefgh"));
}
```

TWO WAYS TO DECOMPOSE A PROBLEM



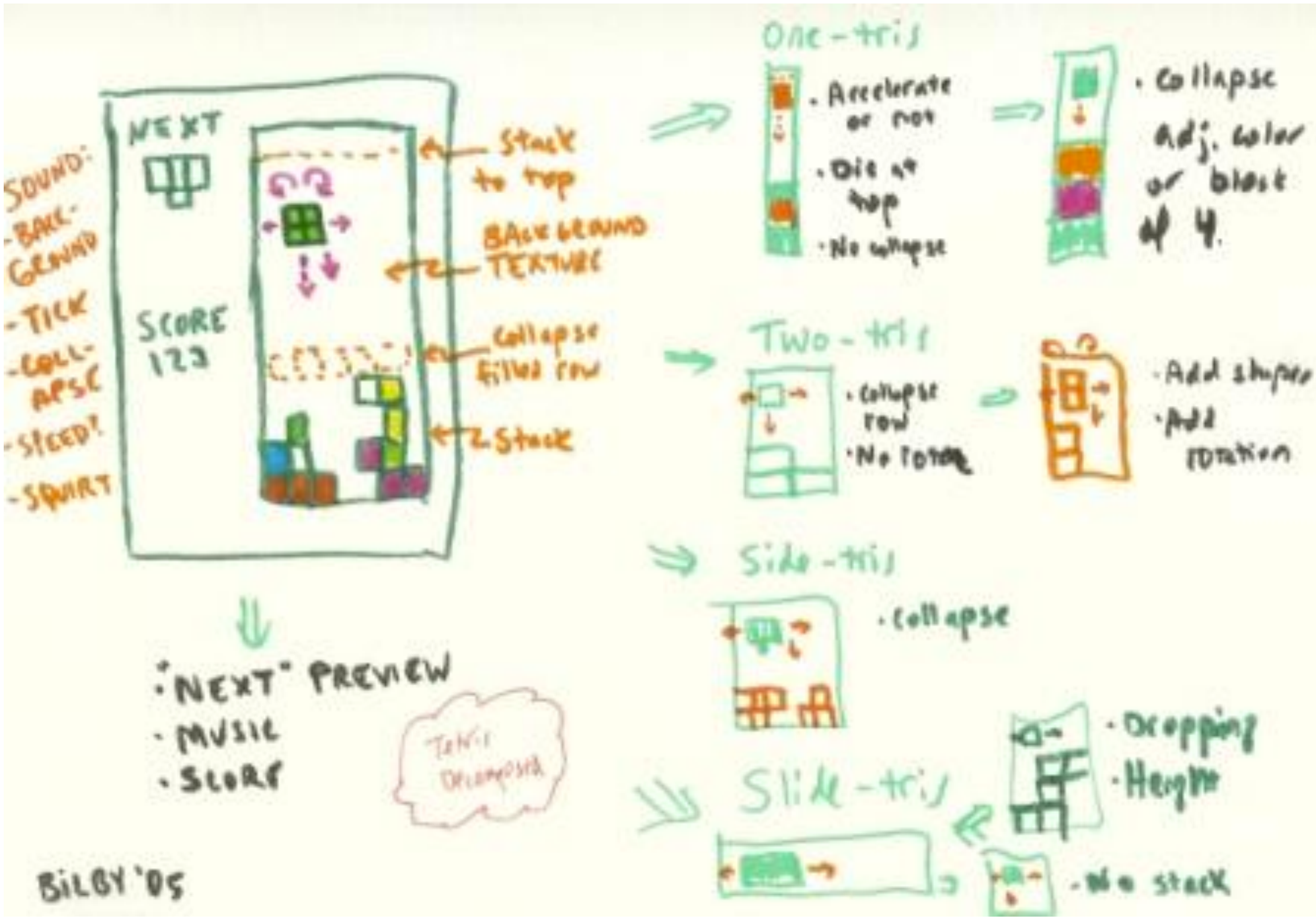
Incremental

Iterative

Jeff Patton, http://jpattonassociates.com/dont_know_what_i_want/

Your level: 0
Full lines: 1
SCORE 60
H E L P
7:Left
9:Right
8:Rotate
1:Draw next
6:Speed up
4:Drop
SPACE:Drop
Next:
STATISTICS
1
0
1
1
0
2
1
Σ : 6
Play TETRIS !

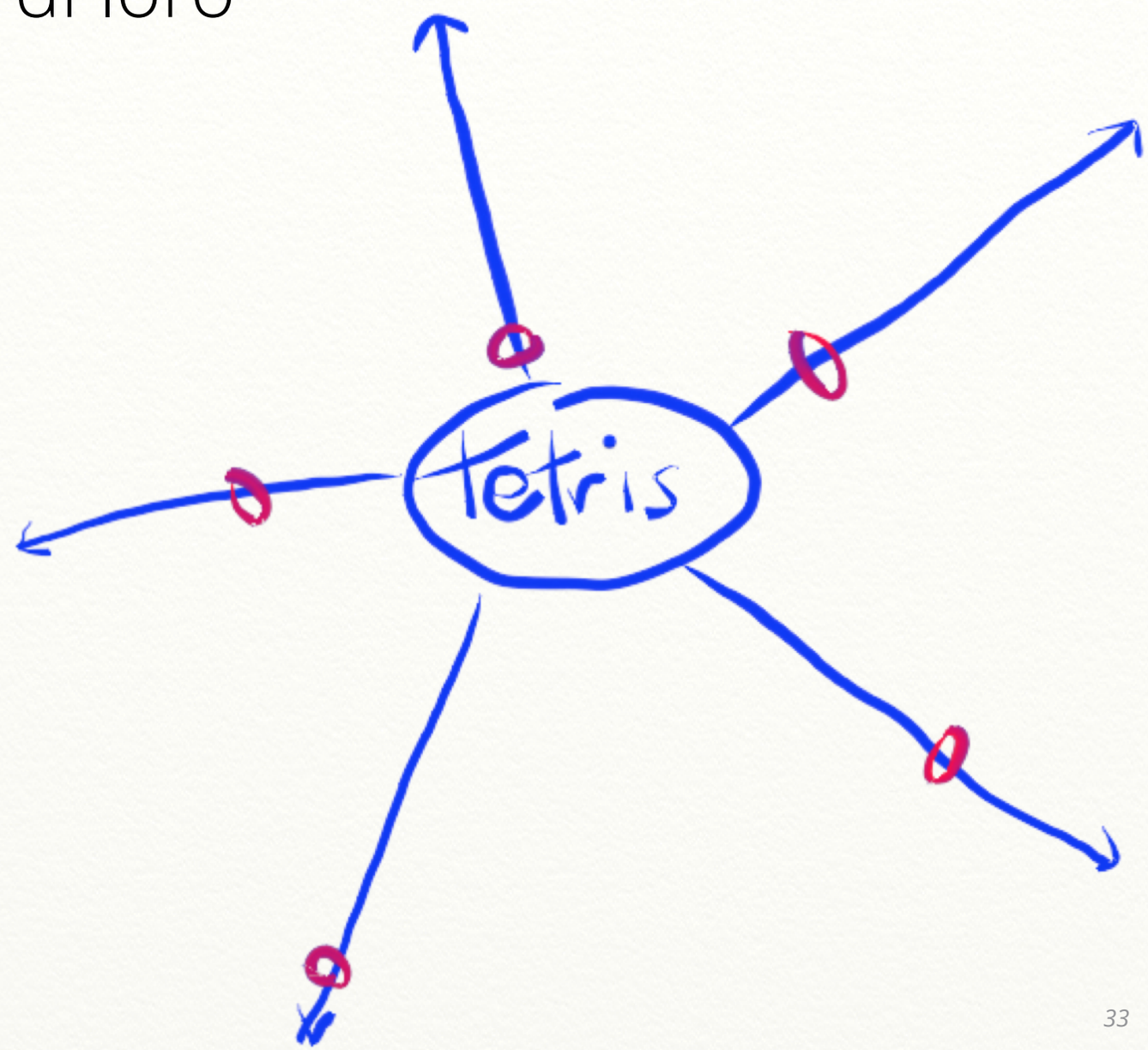
DECOMPOSE THE PROBLEM



Bill Wake, xp123.com/articles/slicing-functionality-alternate-paths

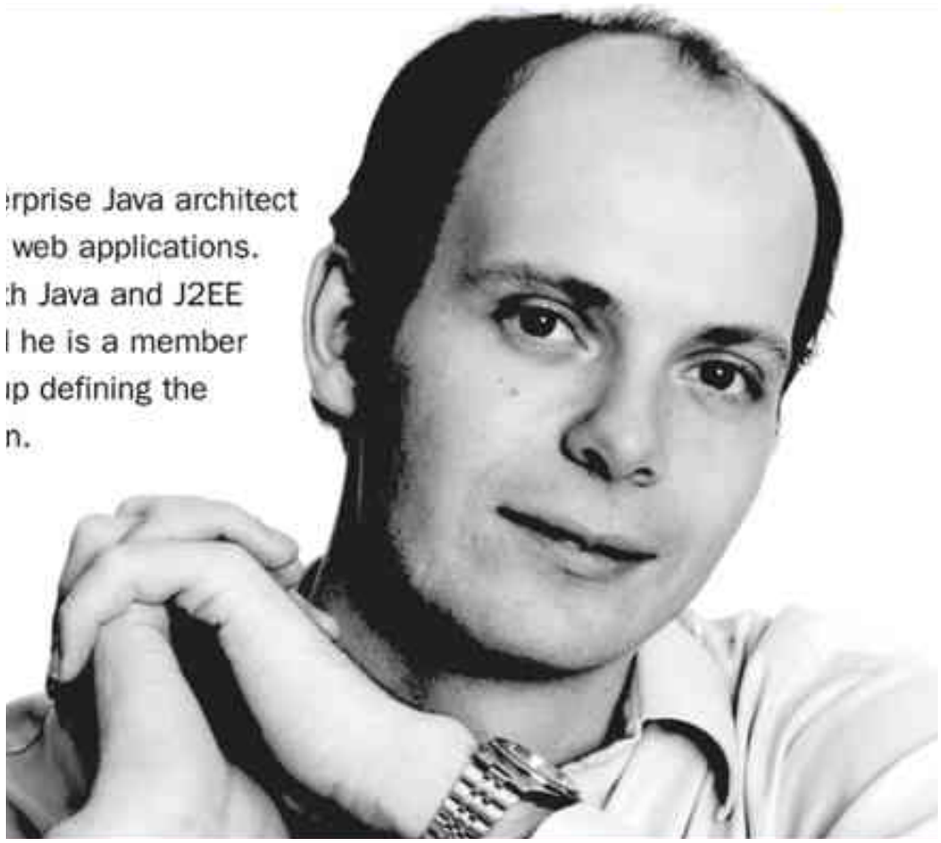
GLI ASSI ORTOGONALI DI TETRIS

- Movimento dei pezzi
- Forma dei pezzi
- Interazione dei pezzi fra di loro
- Comandi del giocatore
- Punteggio
- Avanzamento di livello
- Suoni e musica
- High score e social
- …



3. DESIGN EMERGENTE

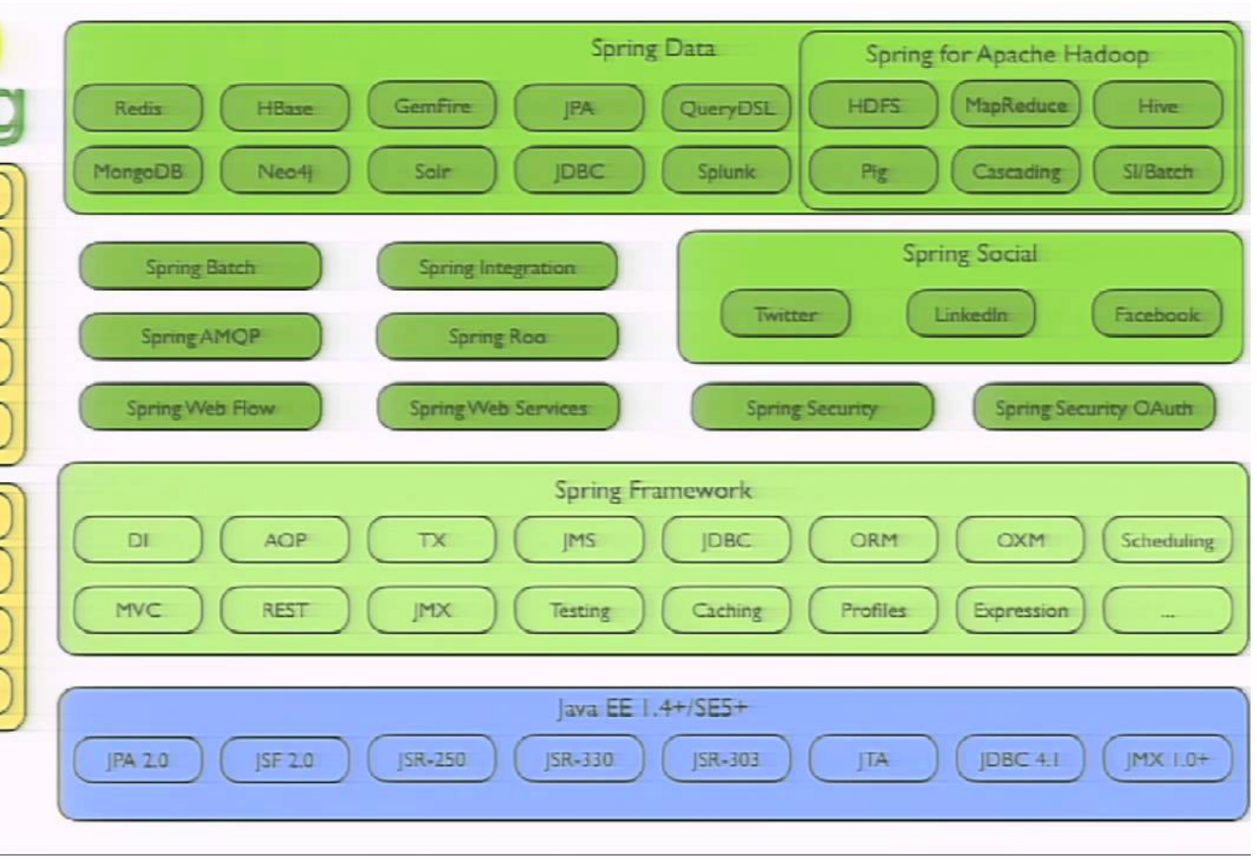
DUE TIPI DI PROGRAMMATORI



L'architetto

Lo smanettone

DUE TIPI DI PROGRAMMATORI




```
<?
if ('POST' == $REQUEST_METHOD) {

  if (isset($HTTP_POST_VARS["cancel"])) {
    redirect("list_travel.php");
  }

  //  $debug_sql = true;
  if (!$errors) {
    $HTTP_POST_VARS["aaa"] = isset($HTTP_POST_VARS["aaa"])
    $HTTP_POST_VARS["bbb"] = isset($HTTP_POST_VARS["bbb"])
    update_model($HTTP_POST_VARS) or die("Errore nell'inser
    redirect("list_model.php");
  } else {
    stripslashes_all($HTTP_POST_VARS);
  }
}
```

L'architetto

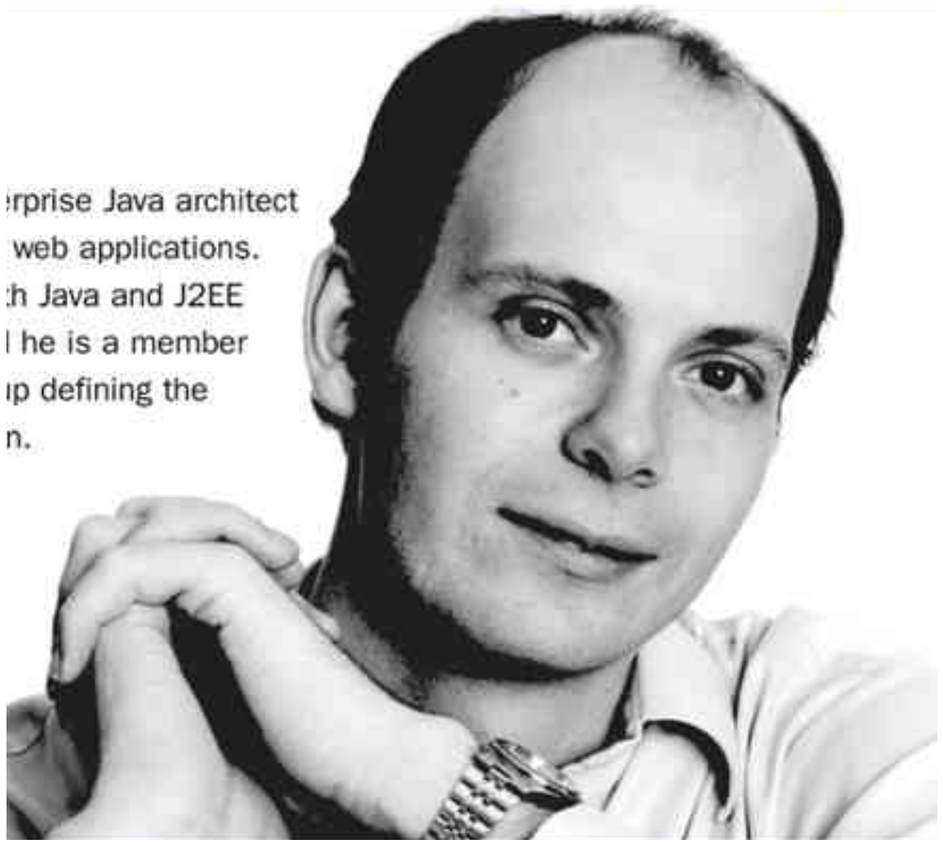
Lo smanettone

NON FA REFACTORING

Lo smanettone

FA DESIGN-DRIVEN TESTING

Ho esperienza! So dove andiamo a parare!



L’architetto

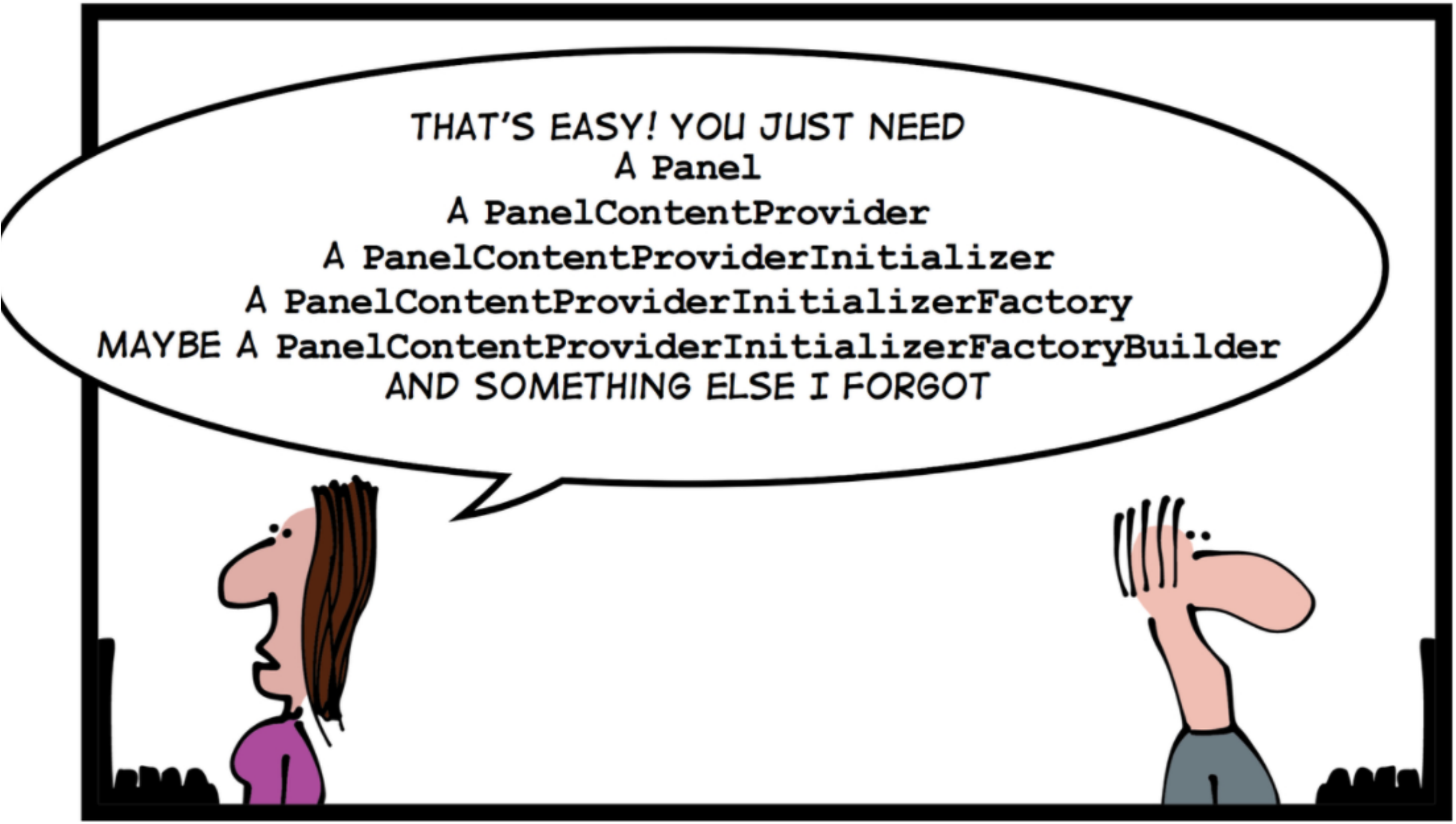
DESIGN-DRIVEN TESTING

Pseudo-TDD

1. Think of a solution
2. Imagine a bunch of classes and functions that you *just know* you'll need to implement (1)
3. Write some tests that assert the existence of (2)
4. Run all tests and fail
5. Implement a bunch of stuff
6. Run all tests and fail
7. Debug
8. Run the tests and succeed
9. Write a `TODO` saying to go back and refactor some stuff later.

Keith Braithwaite, *TDD as if you Meant It*

ENTRAMBI FARANNO "BAD OOP"



http://geek-and-poke.com/geekandpoke/2014/11/8/frameworks

TDD SECONDO KENT BECK:

1. Quickly add a test.
2. Run all tests and see the new one fail.
3. Make a little change.
4. Run all tests and see them all succeed.
5. Refactor to remove duplication.

Pag. 1 di
“TDD By Example”

Do The Simplest Thing That Could Possibly Work

- (if you're not sure what to do yet)

Beware: The history and therefore meaning of this phrase is lost on many people who wield it. The 'if you're not sure what to do yet' part tends to be forgotten. See: http://www.artima.com/intv/simplest3.html

Particularly:

I think the advice got turned into a command: "Do the simplest thing that could possibly work." That's a little more confusing, because there isn't this notion that as soon as you've done it, we'll evaluate it. People ask, "Well, how do you know it's the simplest?" In my case, we didn't know. We were just going to get it on the screen and look at it. But as soon as it becomes a comm

of a

http://wiki.c2.com/?DoTheSimplestThingThatCouldPossiblyWork

Please note that the article quoted is ten years old (from 2004) and mentions Smalltalk as a primary driver for a lot of the experimentation done by the principles working on the project. Software engineering has come a long way in the last decade.

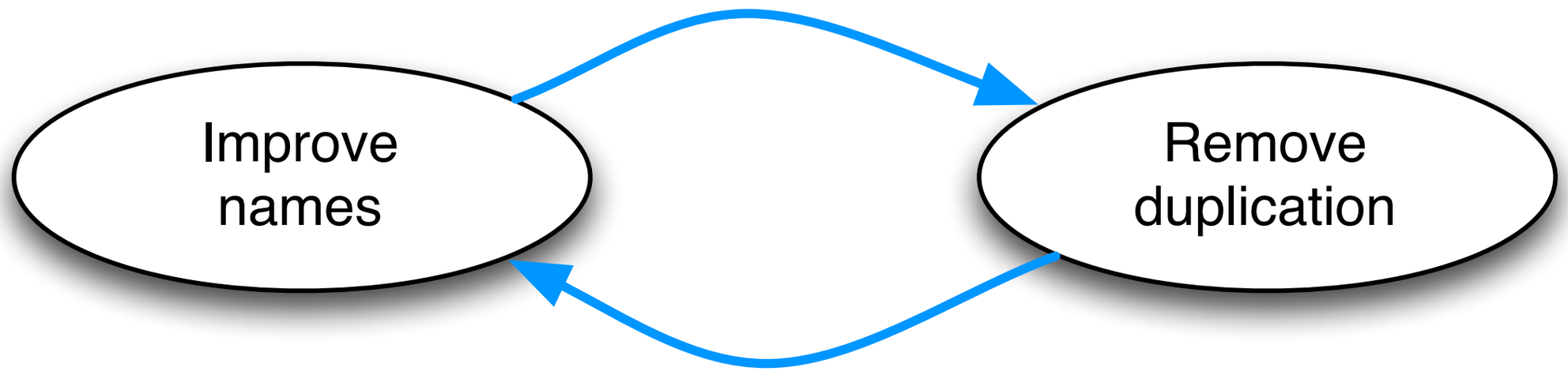
KENT BECK'S "FOUR RULES OF SIMPLE DESIGN"

Code is simple enough when:

1. *It passes all the tests*
2. *It clearly expresses intent*
3. *It contains no duplication*
4. *With the minimum number of elements*

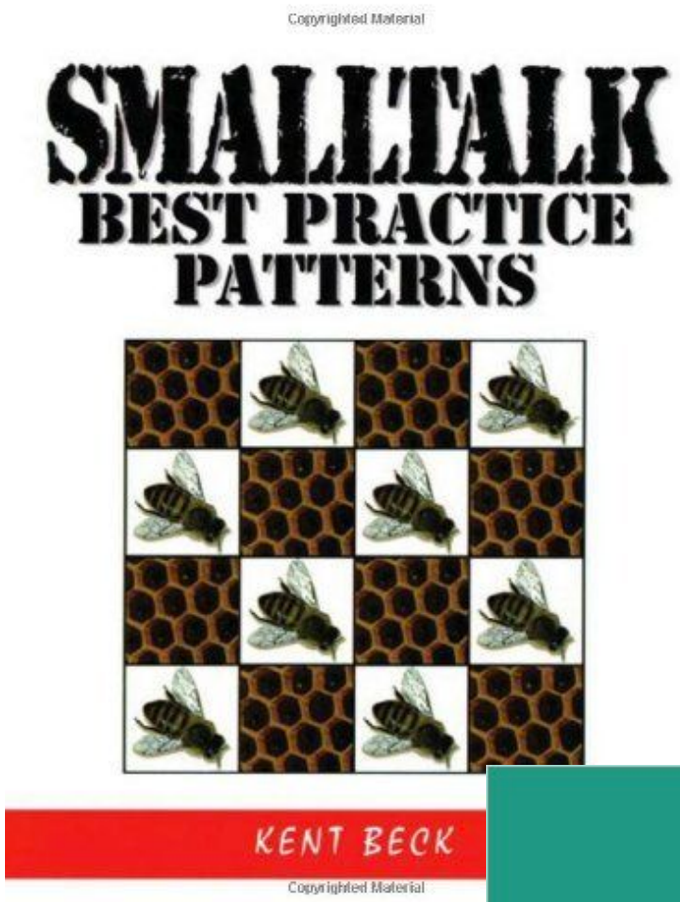
Da
XP Explained

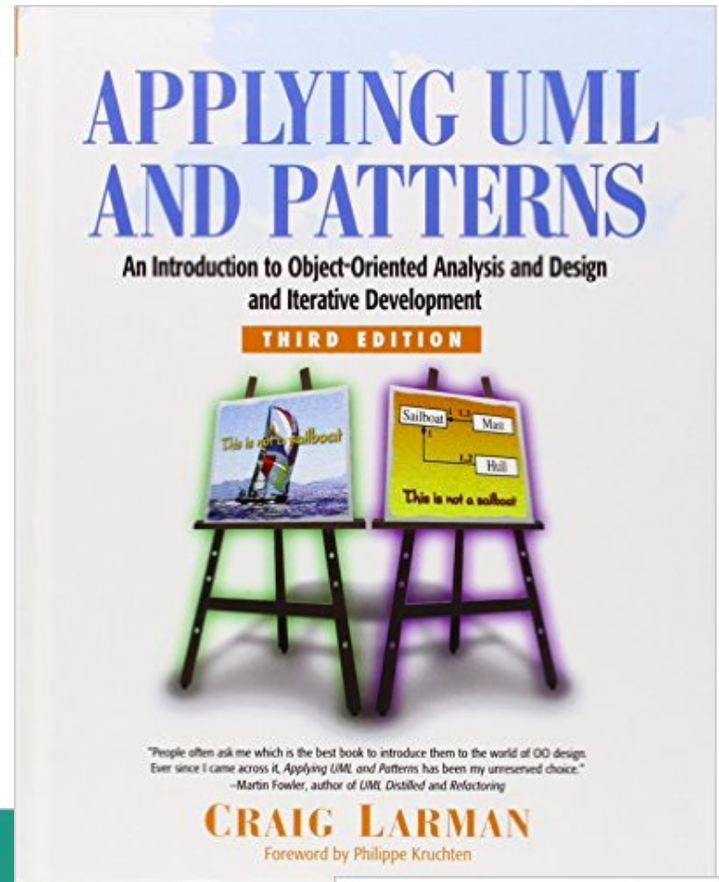
SECONDO J.B. RAINSBERGER:

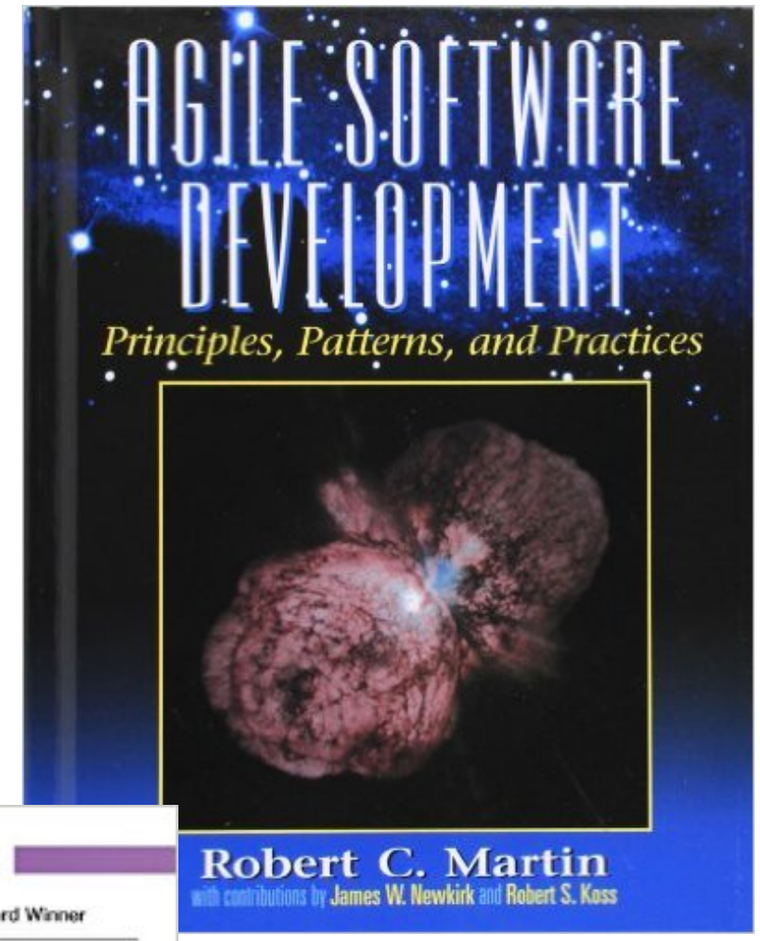


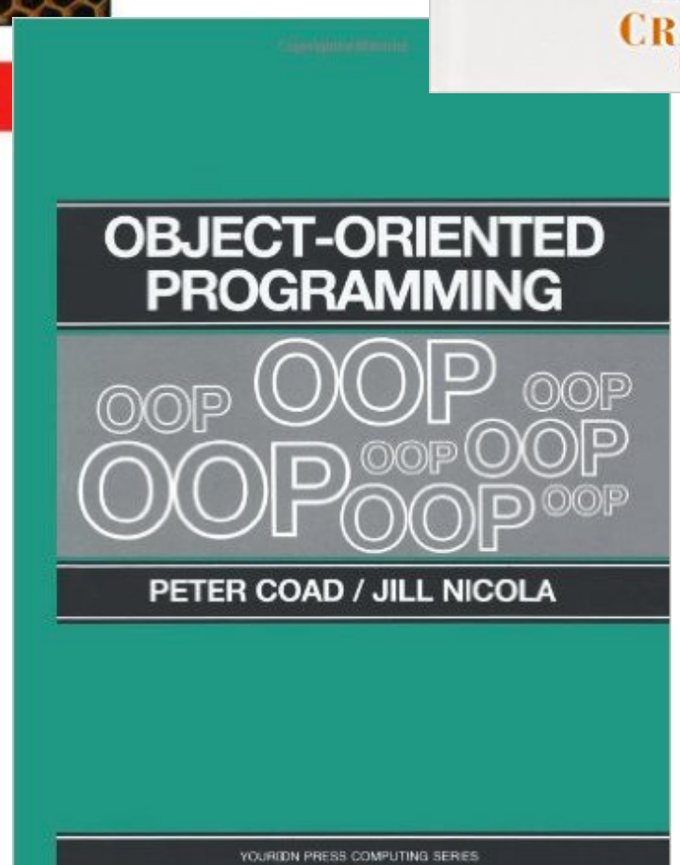
http://blog.thecodewhisperer.com/permalink/putting-an-age-old-battle-to-rest/

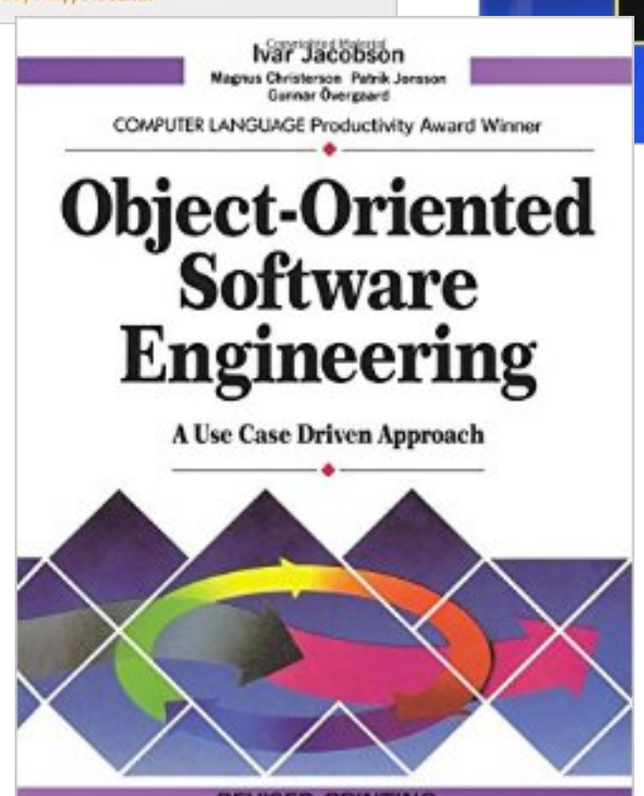
STUDIARE IL SOFTWARE DESIGN È IMPORTANTE



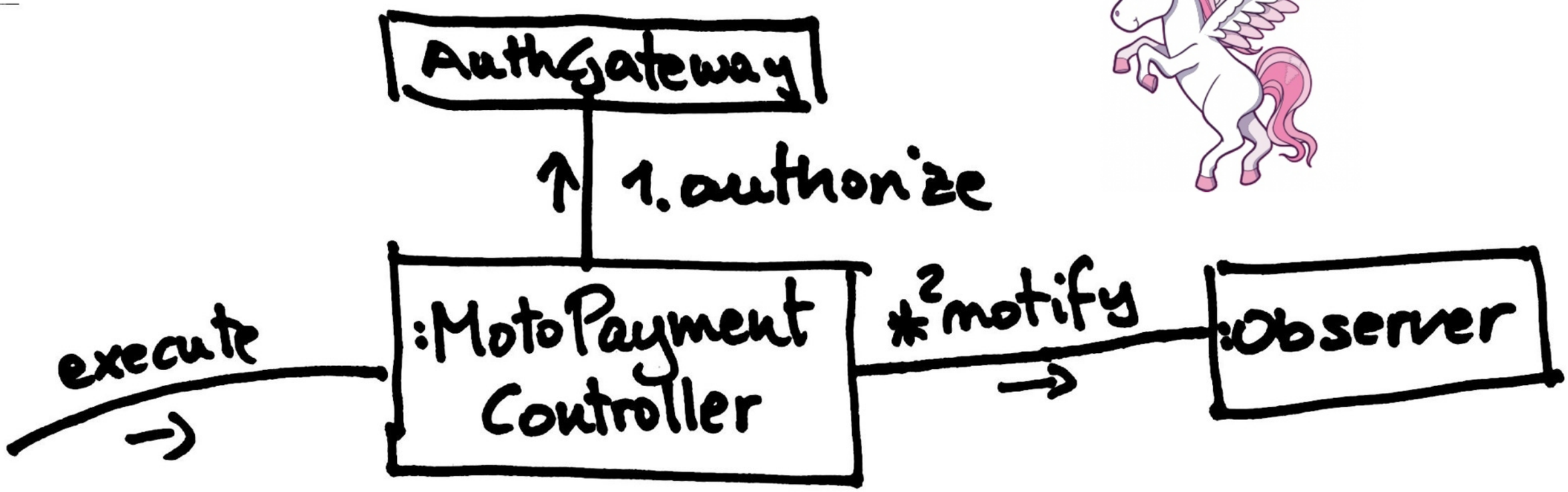






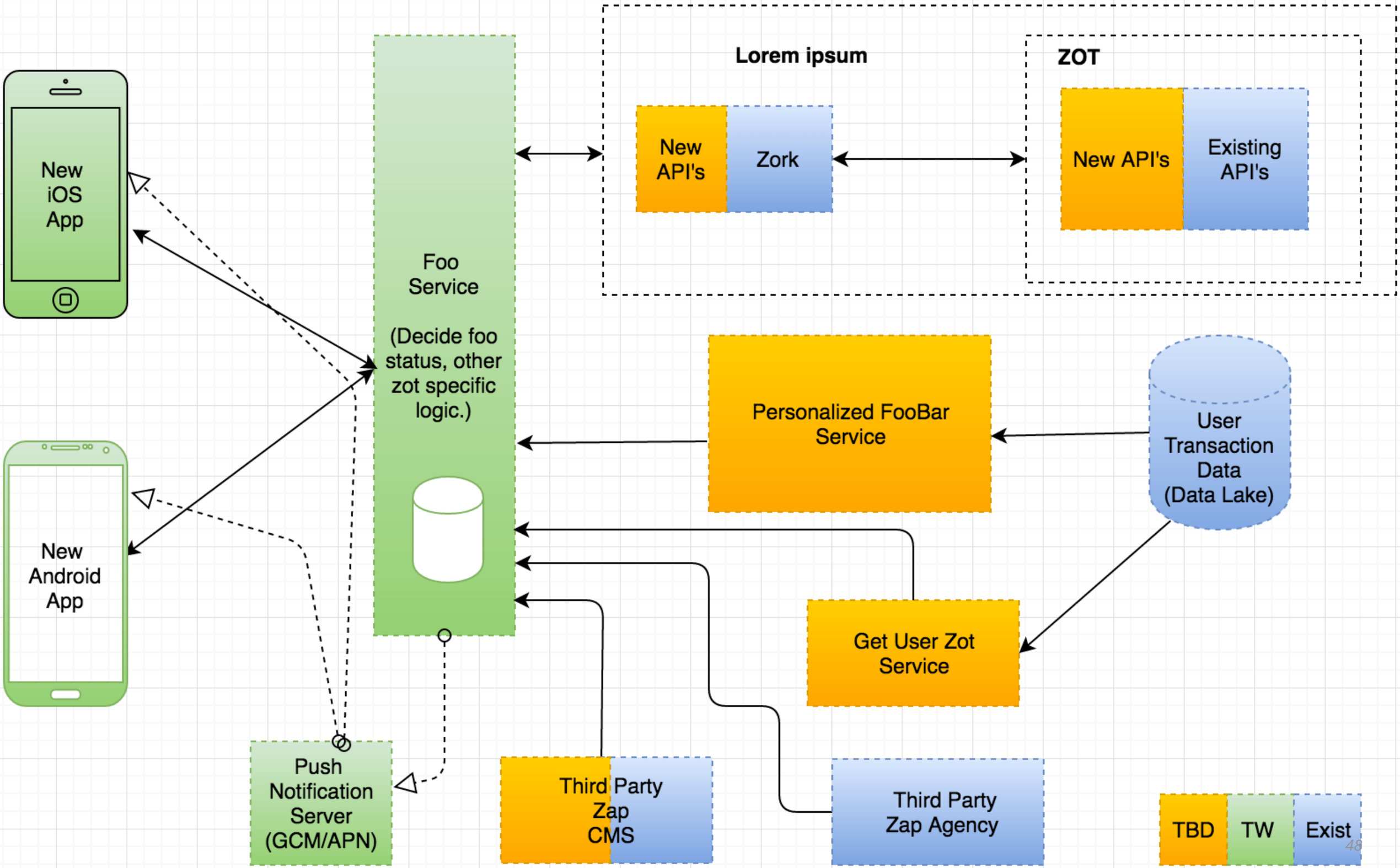


E POI, A UN CERTO PUNTO, COMINCI A CAPIRE...



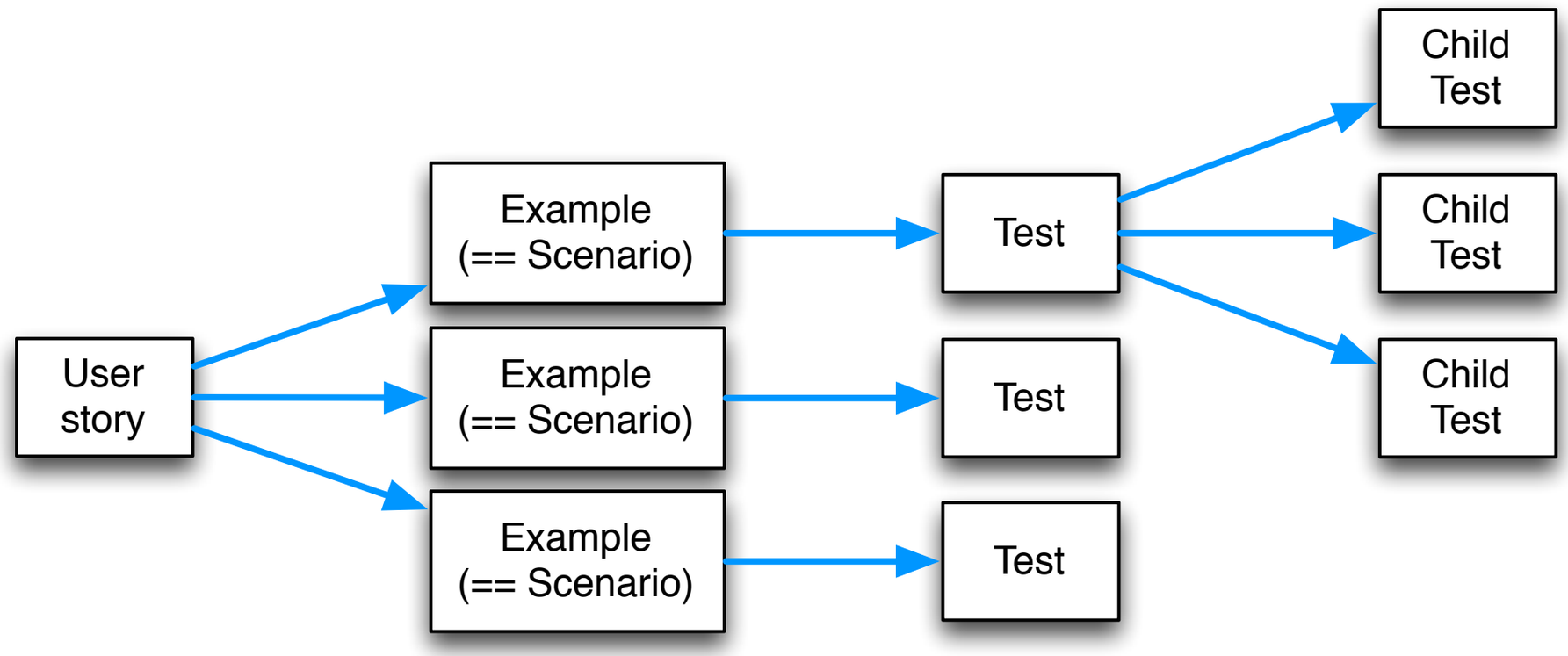
Che cosa faccio io in pratica?

CONDIVIDERE CON GLI STAKEHOLDER

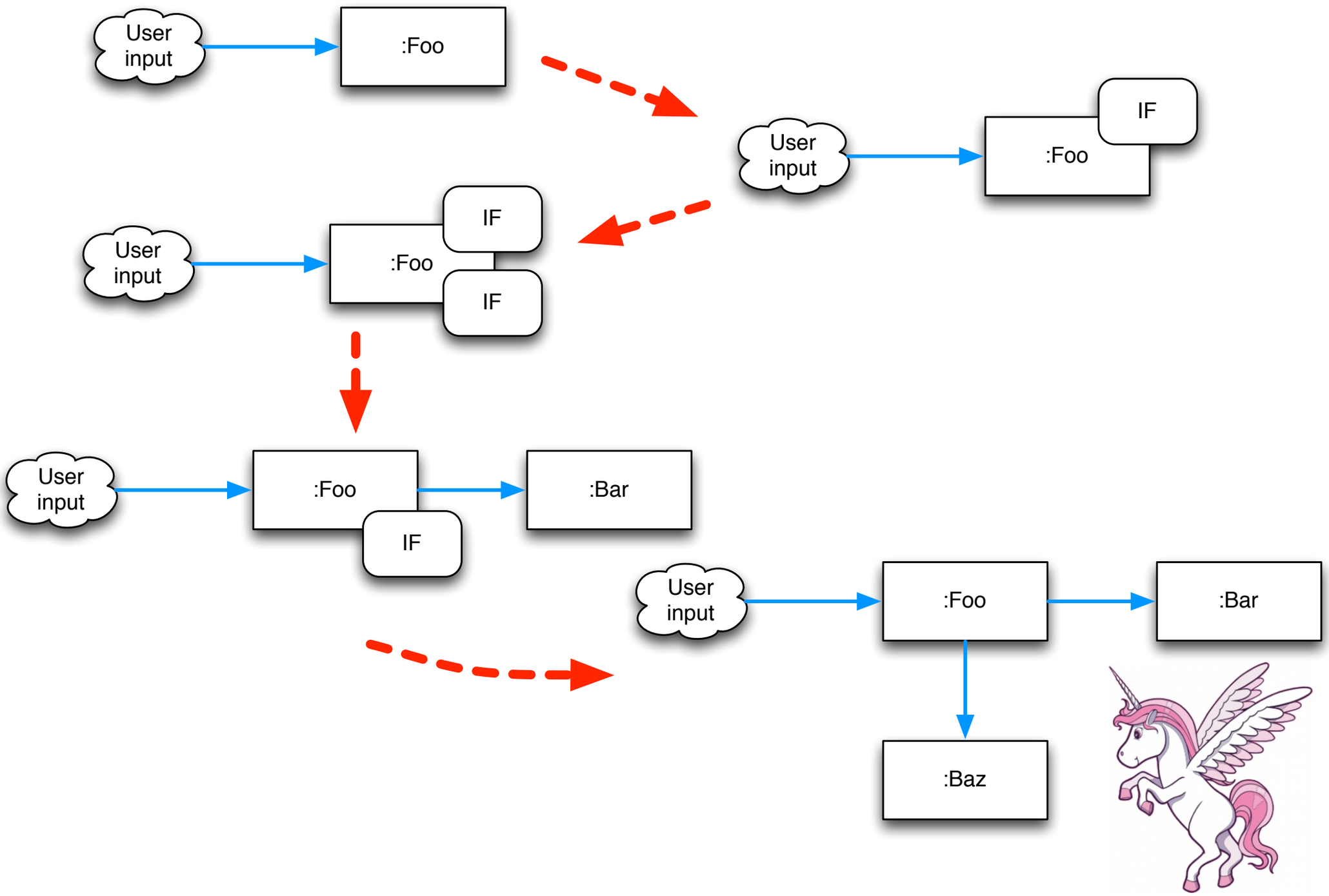


ALLINEARSI SU UNA NUOVA FEATURE

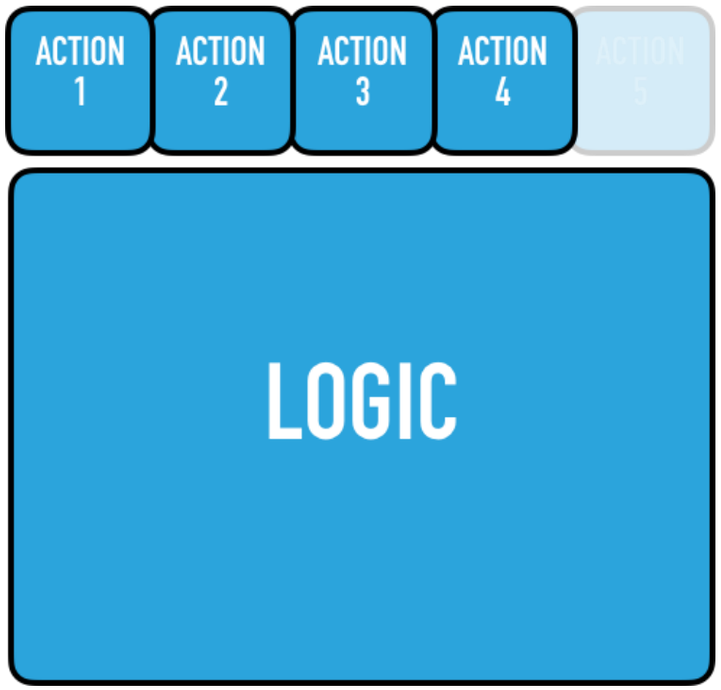
UN TEST PER VOLTA

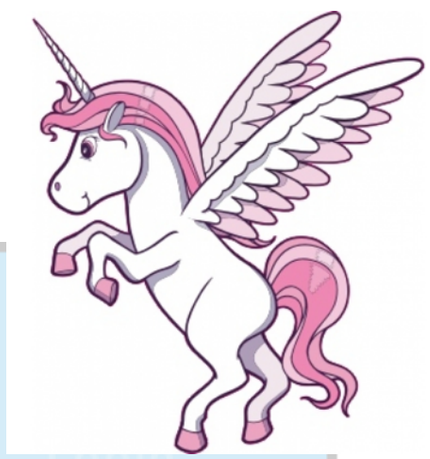


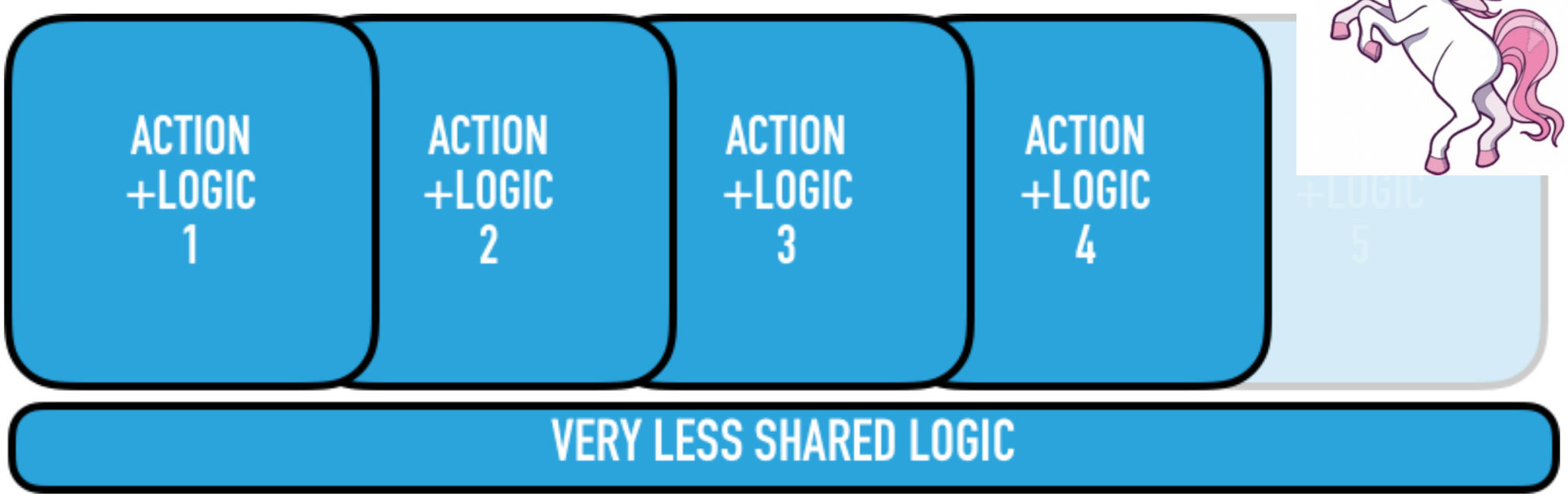
UN TEST PER VOLTA



ASPETTA DI TROVARE LE ASTRAZIONI BUONE







https://medium.com/@rdsubhas/10-modern-software-engineering-mistakes-bc67fbef4fc8

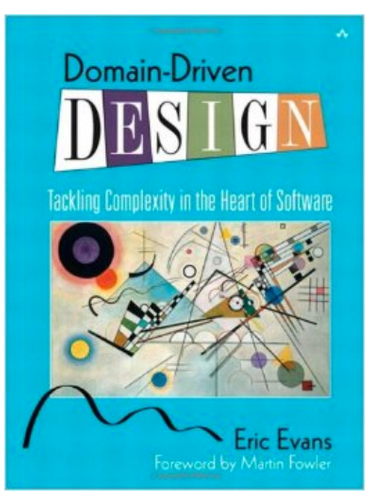
4. CHE FARE CON IL DB?

Object-Oriented

Parte dai *comportamenti*

Disaccoppia il DB

TDD: OK!



DB-Oriented

Parte dai *dati*

Strettamente accoppiato al DB

TDD è difficile

RAILS

django

play!

THE "SANDWICH" PATTERN

```
// we start the use case in the world of infrastructure
CounterRepository repository = ...;
Counter counter = repository.findCounter(id);

// here we enter the world of pure logic
counter.increment();

// here we return to the world of infrastructure
dao.save(counter);
```

The database is just a detail that you don't need to figure out right away.

SIAMO ALLA FINE...

HO RISPOSTO?

- Il design emergente esiste davvero?
- Il TDD richiede più tempo?
- Come si fa con la UI?
- Come si fa con il DB?
- Mock sì, mock no?
- Posso fidarmi di Uncle Bob?
- Il TDD è morto?!?

Altre domande? Le tue domande?

WHERE TO FIND MORE HELP

- Milano XPUG
- Rileggi “TDD By Example” una volta all’anno
- J.B. Rainsberger’s video training
- *Growing Object-Oriented Software* book and mailing list
- TDD Mailing list on Yahoo

PER IL PROSSIMO EVENTO TDD A MILANO...

matteo.vaccari.name

Matteo Vaccari

It is possible to write software that solves real business problems, cheaply and reliably. The recipe is well known, even though it's not easy to do.

I am a software developer. I'm happy to work as a programmer for ThoughtWorks, mostly in the Milano area. I used to teach at the Università dell'Insubria.

- Newsletter. Subscribe to receive updates every one or two months or so.

Please subscribe!

GRAZIE

```
matteo@localhost $ karma start --single-run
16 11 2016 17:34:03.534:INFO [karma]: Karma v1.3.0 server started at http://localhost:9876/
16 11 2016 17:34:03.537:INFO [launcher]: Launching browser Chrome with unlimited concurrency
16 11 2016 17:34:03.547:INFO [launcher]: Starting browser Chrome
16 11 2016 17:34:04.372:INFO [Chrome 54.0.2840 (Mac OS X 10.11.6)]: Connected on socket /#4FEBm
AAAA with id 16485270
.....................................................................
Chrome 54.0.2840 (Mac OS X 10.11.6): Executed 68 of 68 SUCCESS (0.07 secs / 0.016 secs)
matteo@localhost $
```

matteo.vaccari.name

ThoughtWorks®

Assumiamo!