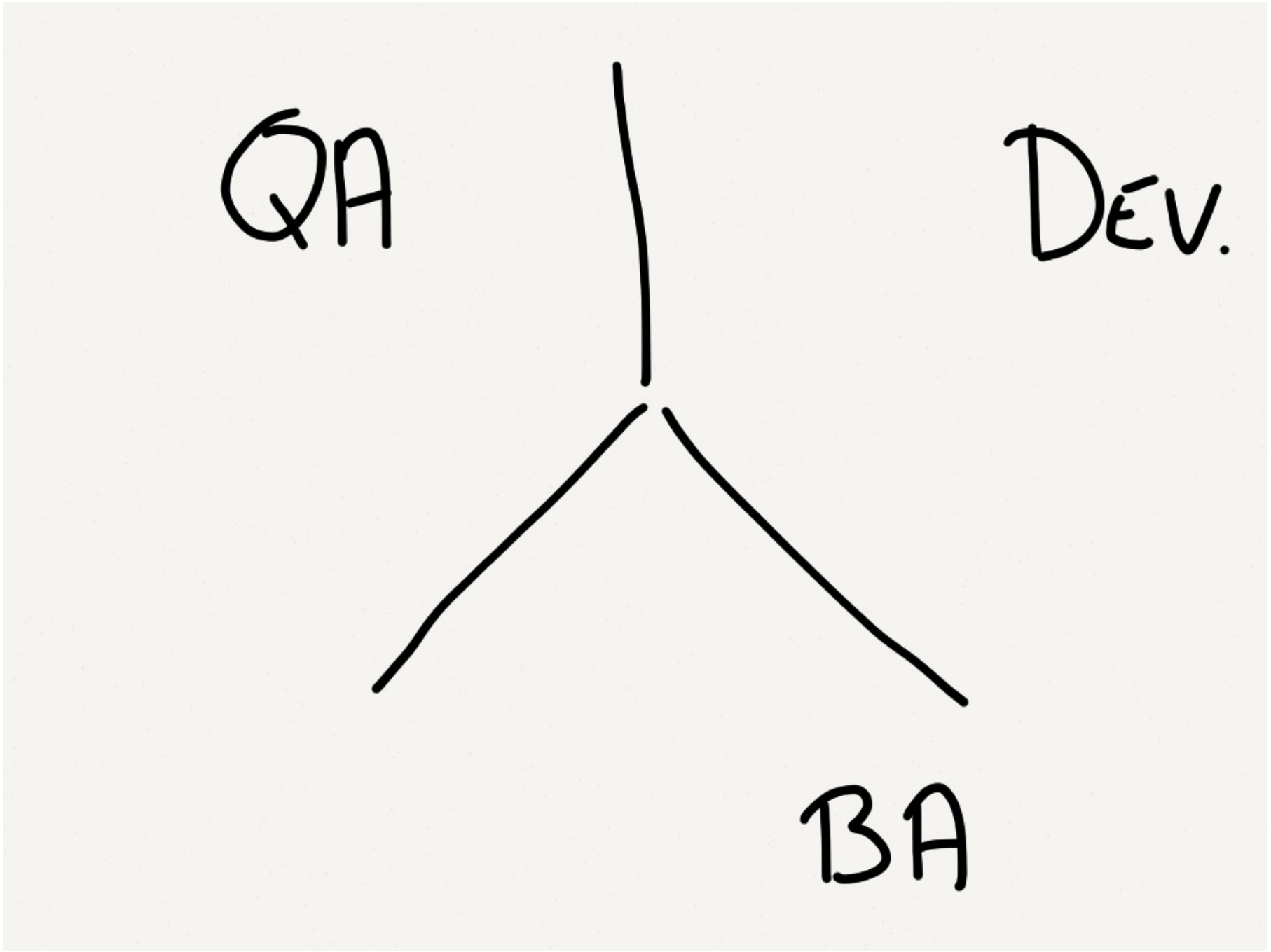
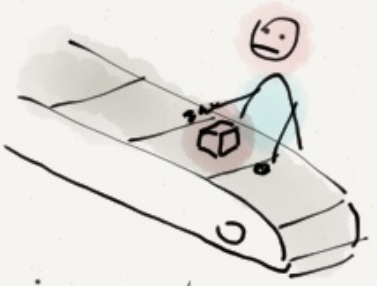

QA
DEV.
BA

"INSPECTION TO FIND DEFECTS
is WASTE.
INSPECTION TO PREVENT DEFECTS
is ESSENTIAL"

— MARY POPPENDIECK



SHIGEO SHINGO ZERO QUALITY CONTROL

"THE JOB OF TESTING IS NOT TO FIND DEFECTS.
THE JOB OF TESTING IS TO PREVENT DEFECTS.

— MARY POPPENDIECK

Disclaimer

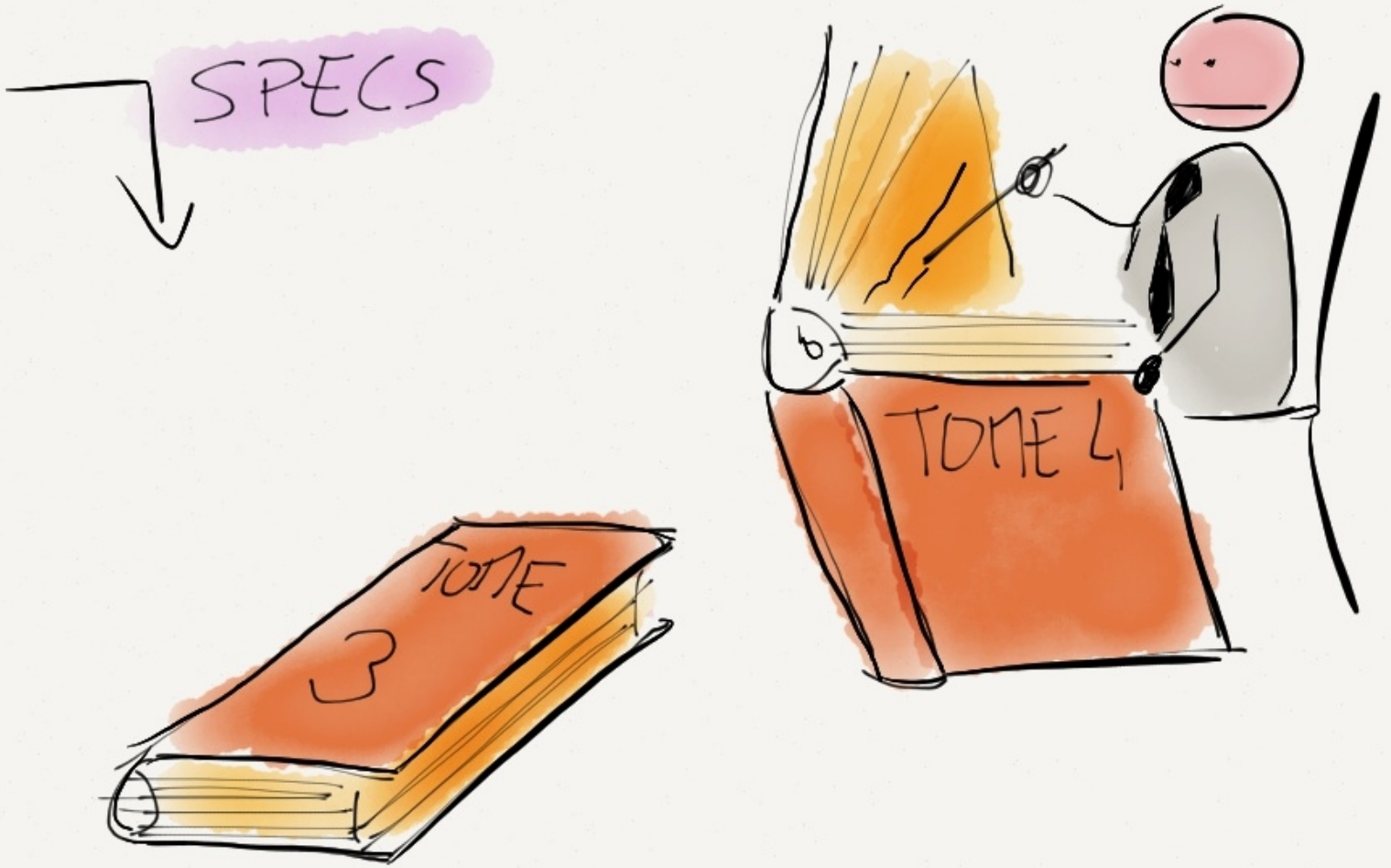
NOTHING IS NEW

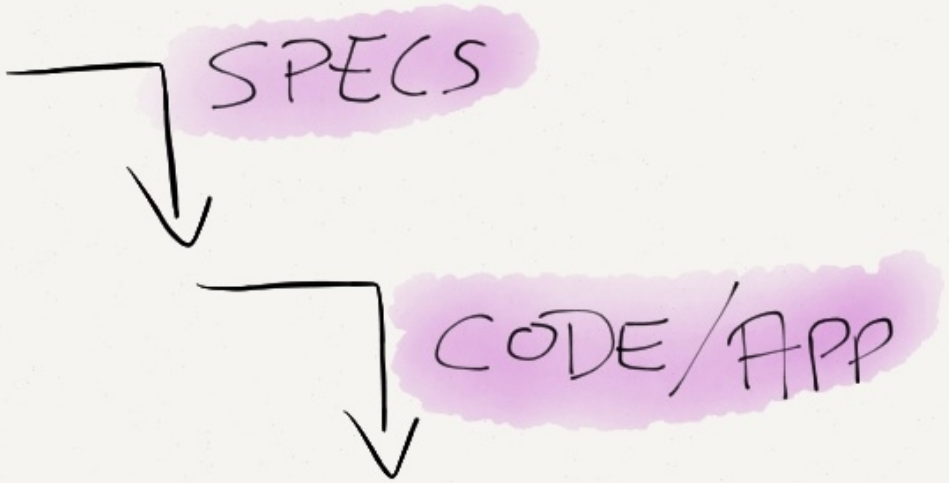
DAN NORTH - 2006

CYCLE

ORGANISATION

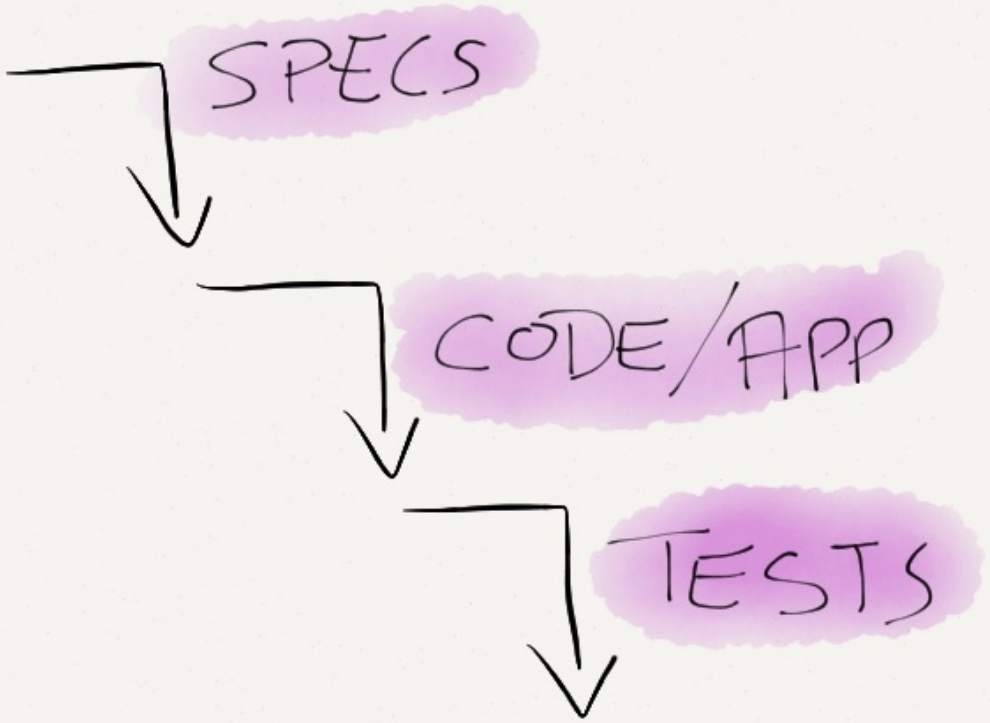

SPECS

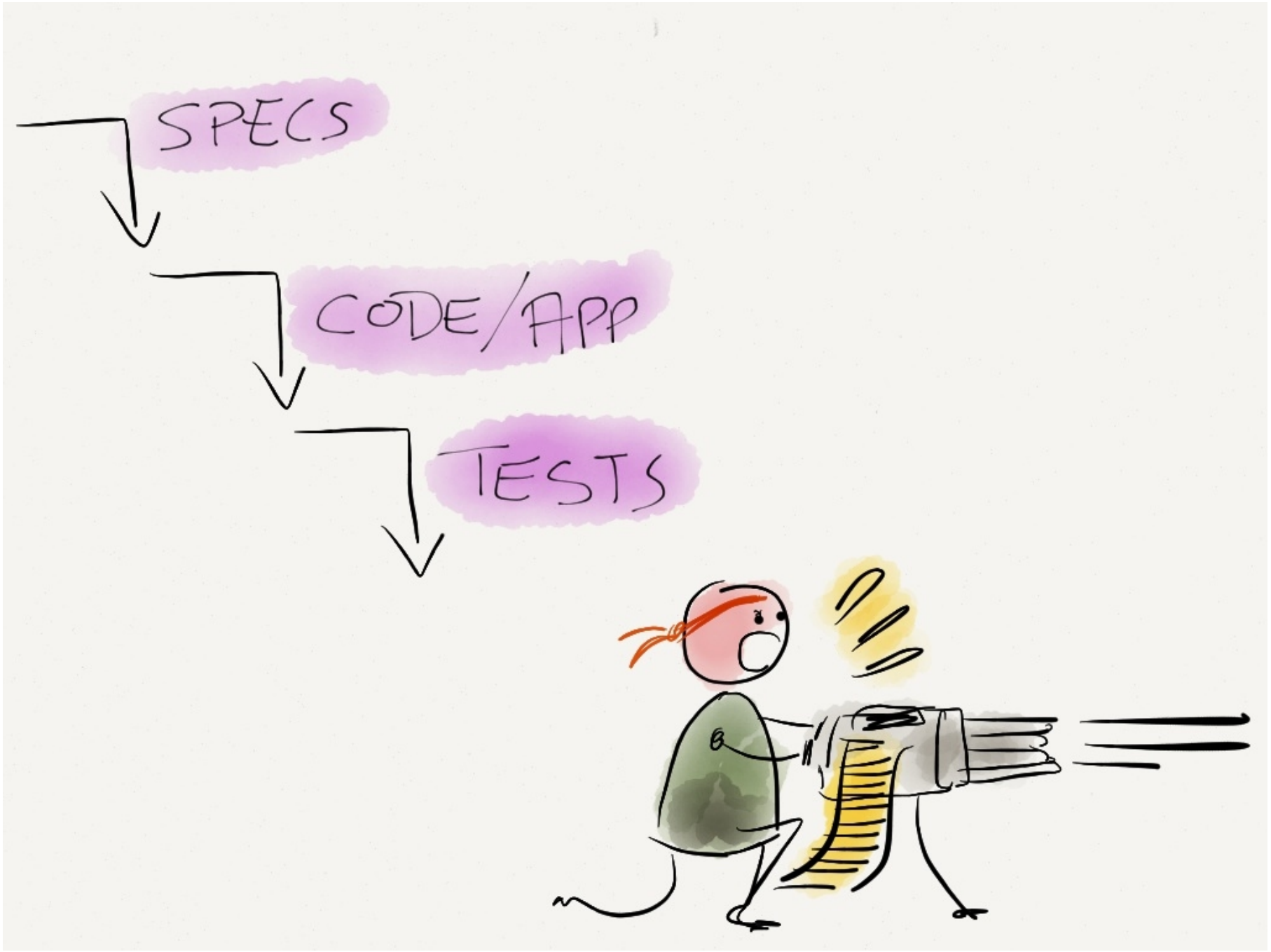

SPECS
TOME 3
TOME 4

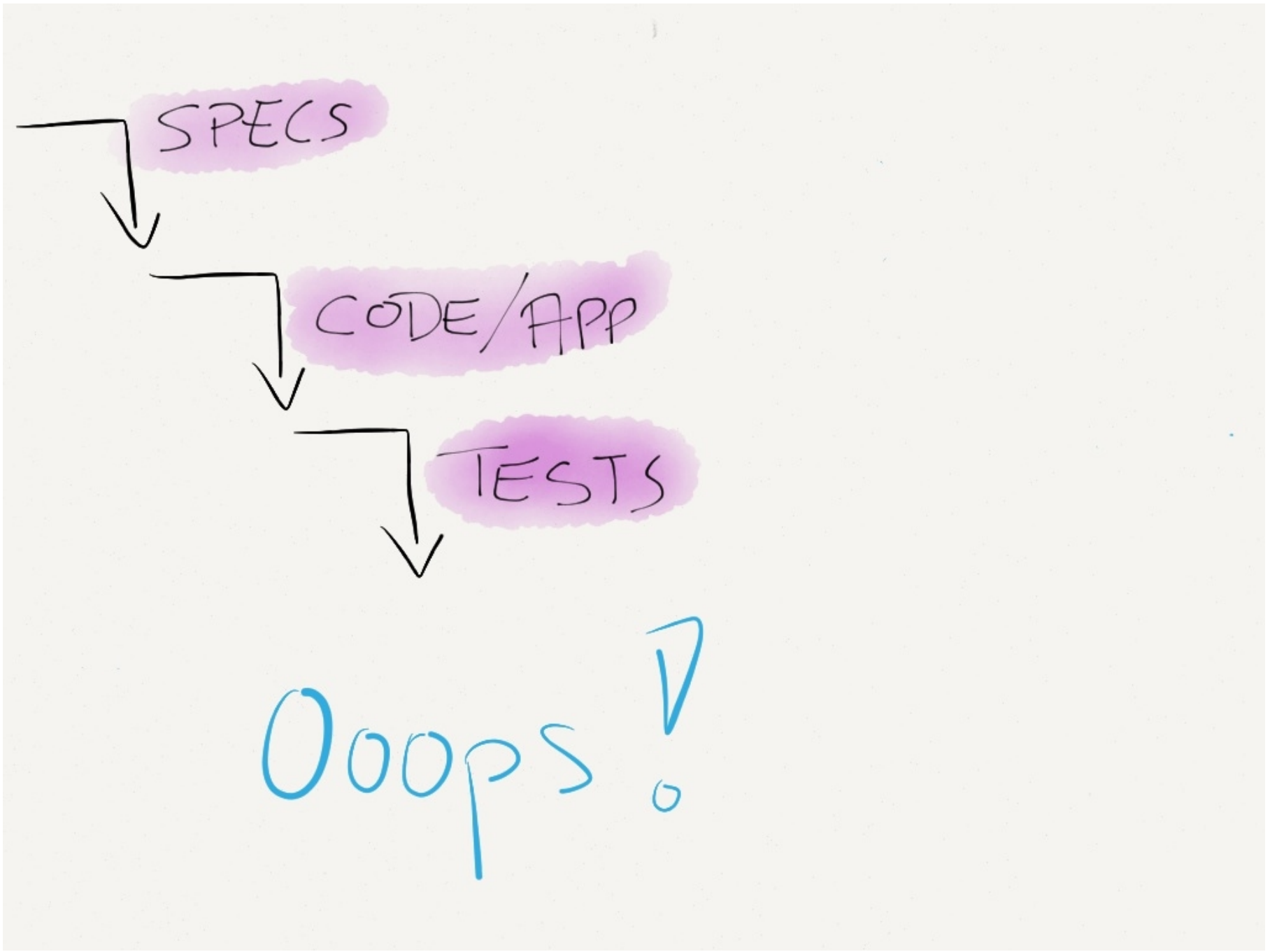

SPECS
CODE/APP

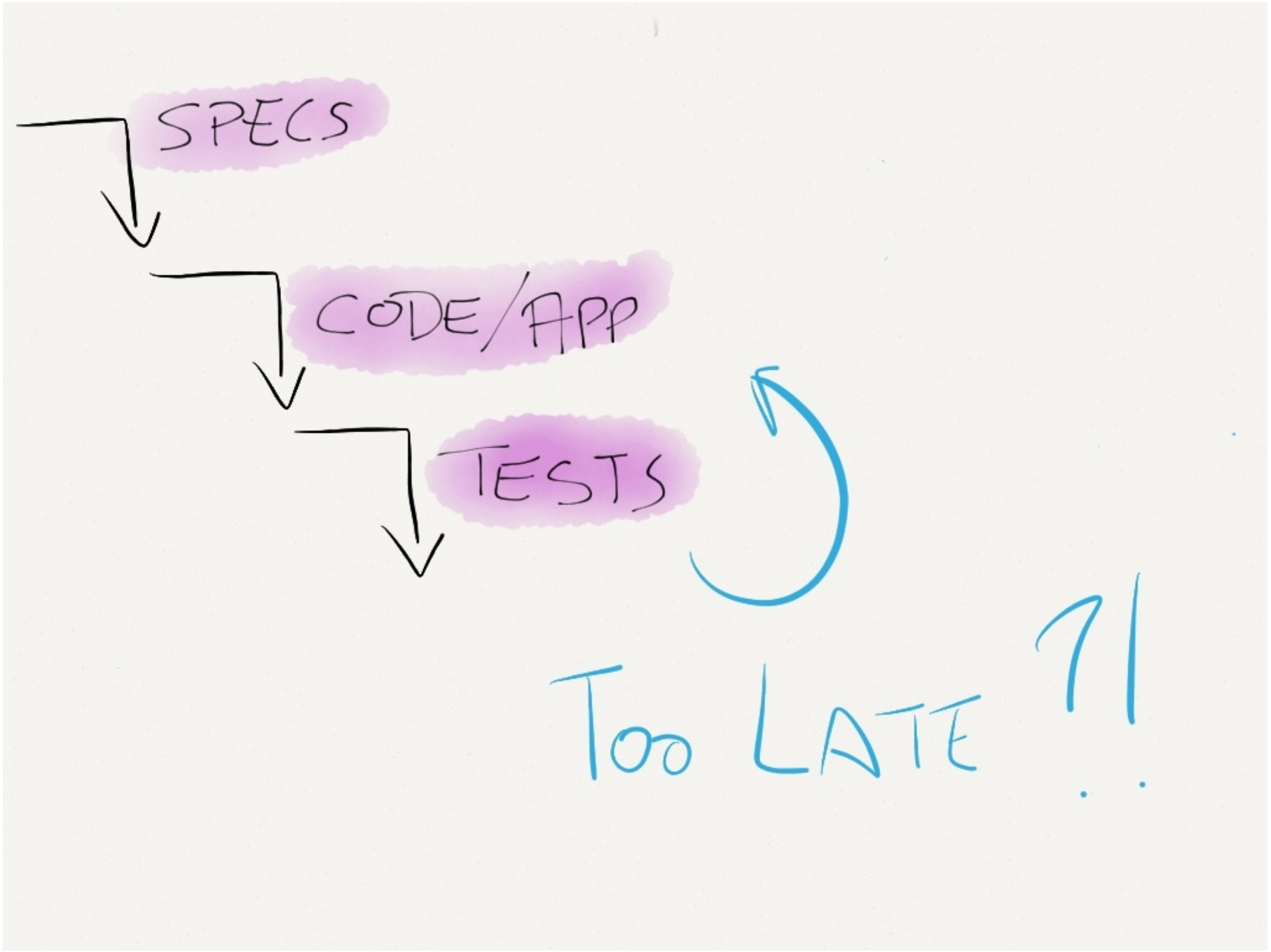
SPECS

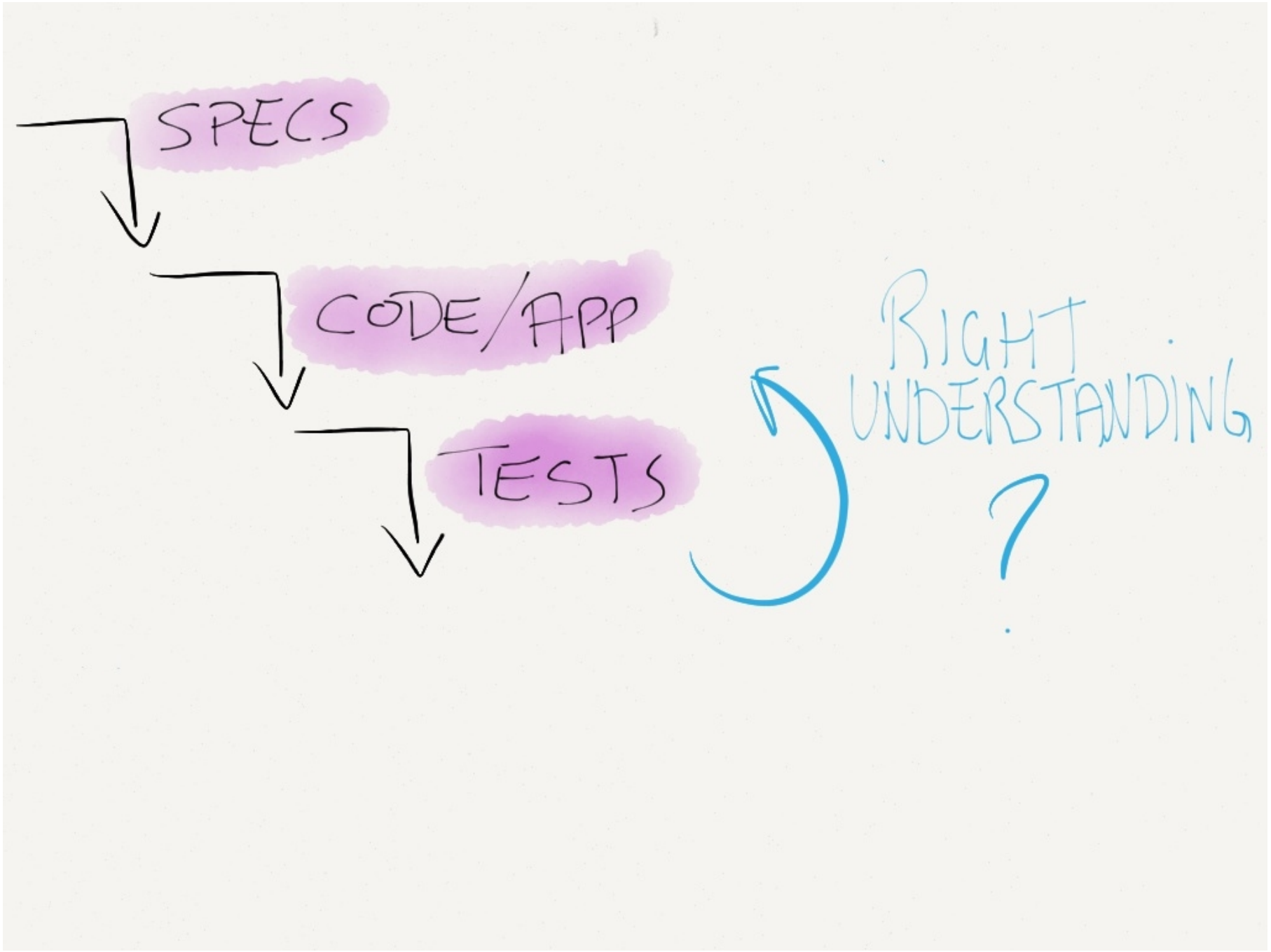
CODE/APP

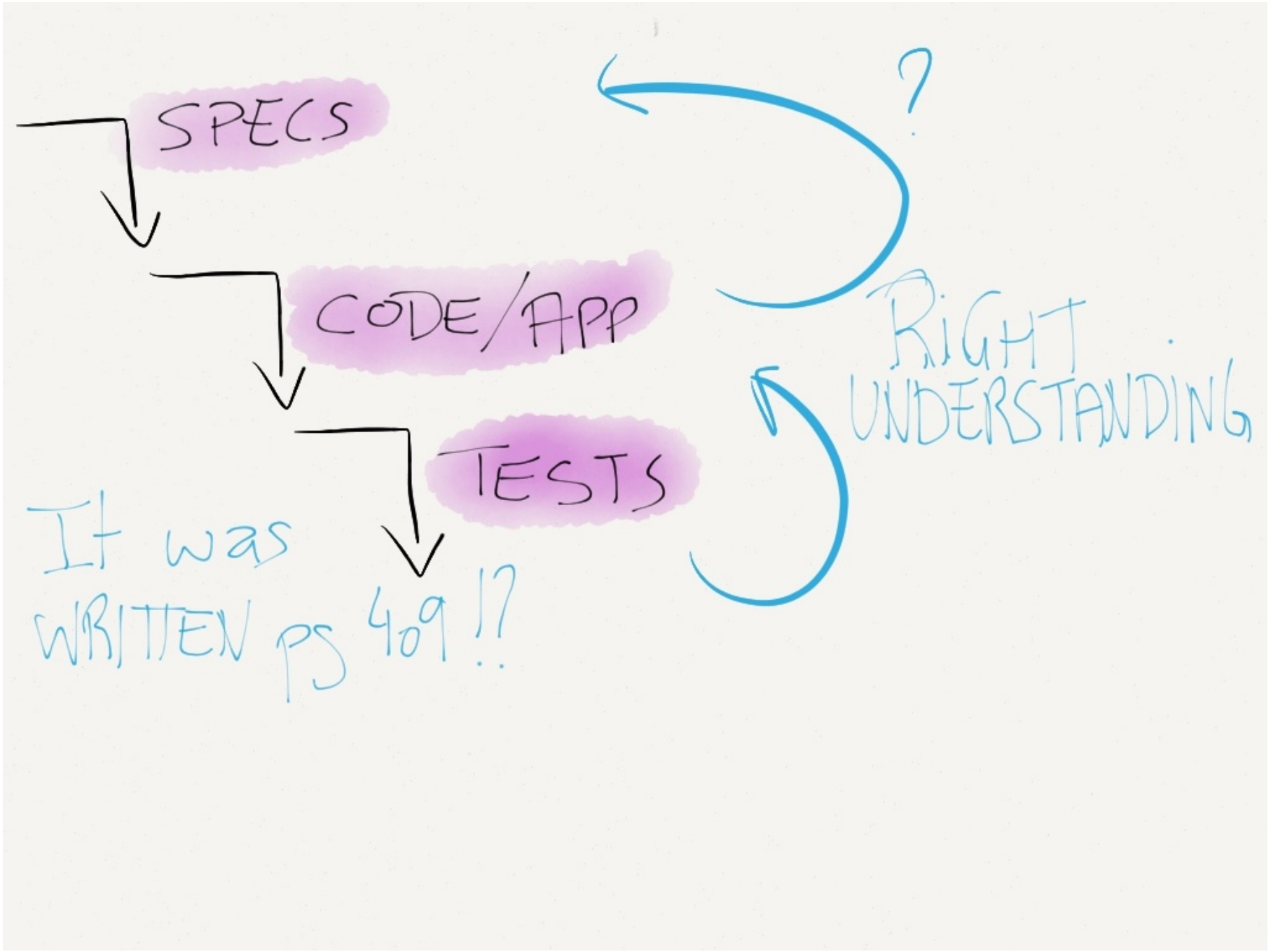

SPECS
CODE/APP
TESTS

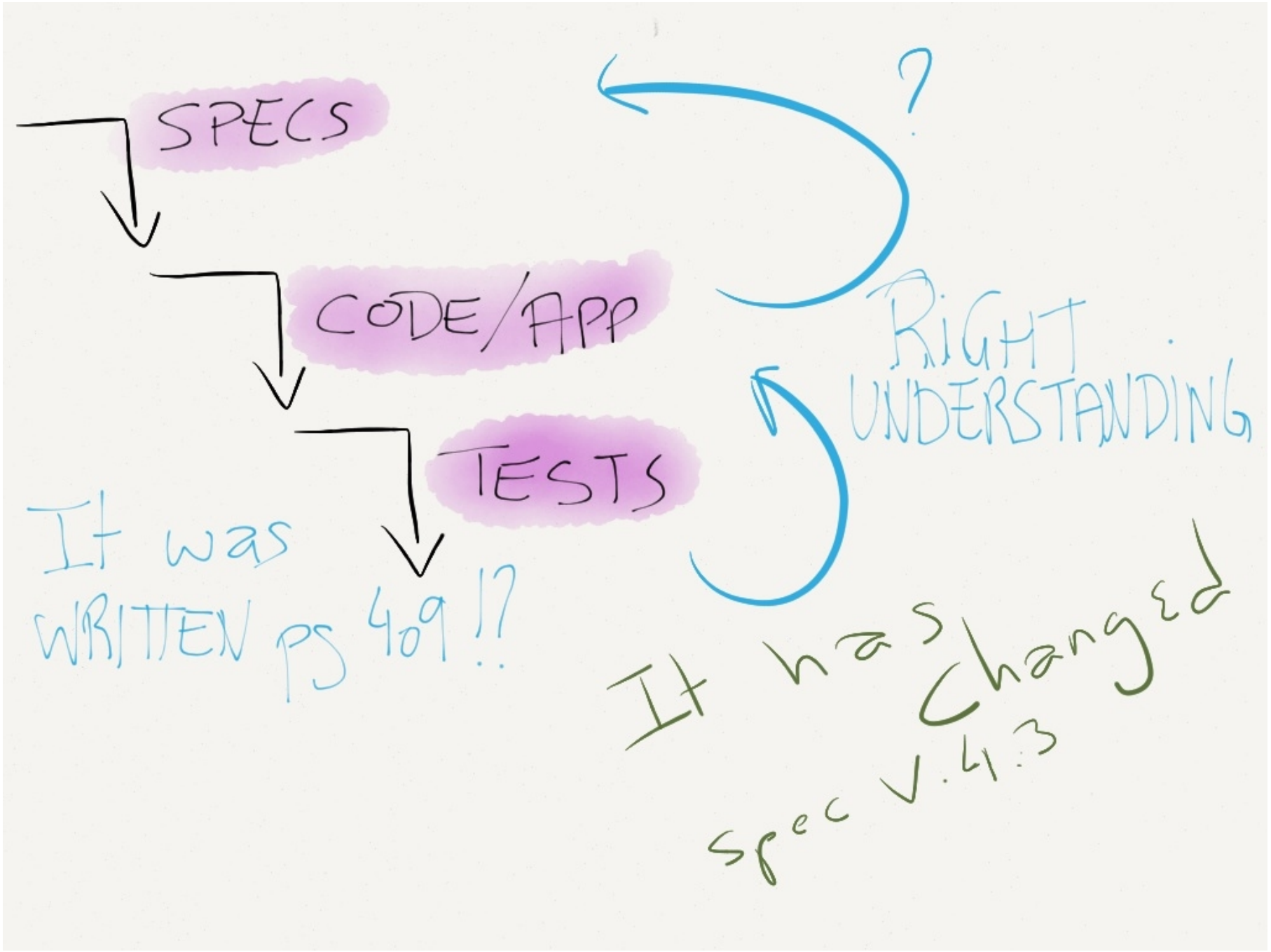

SPECS
CODE/APP
TESTS

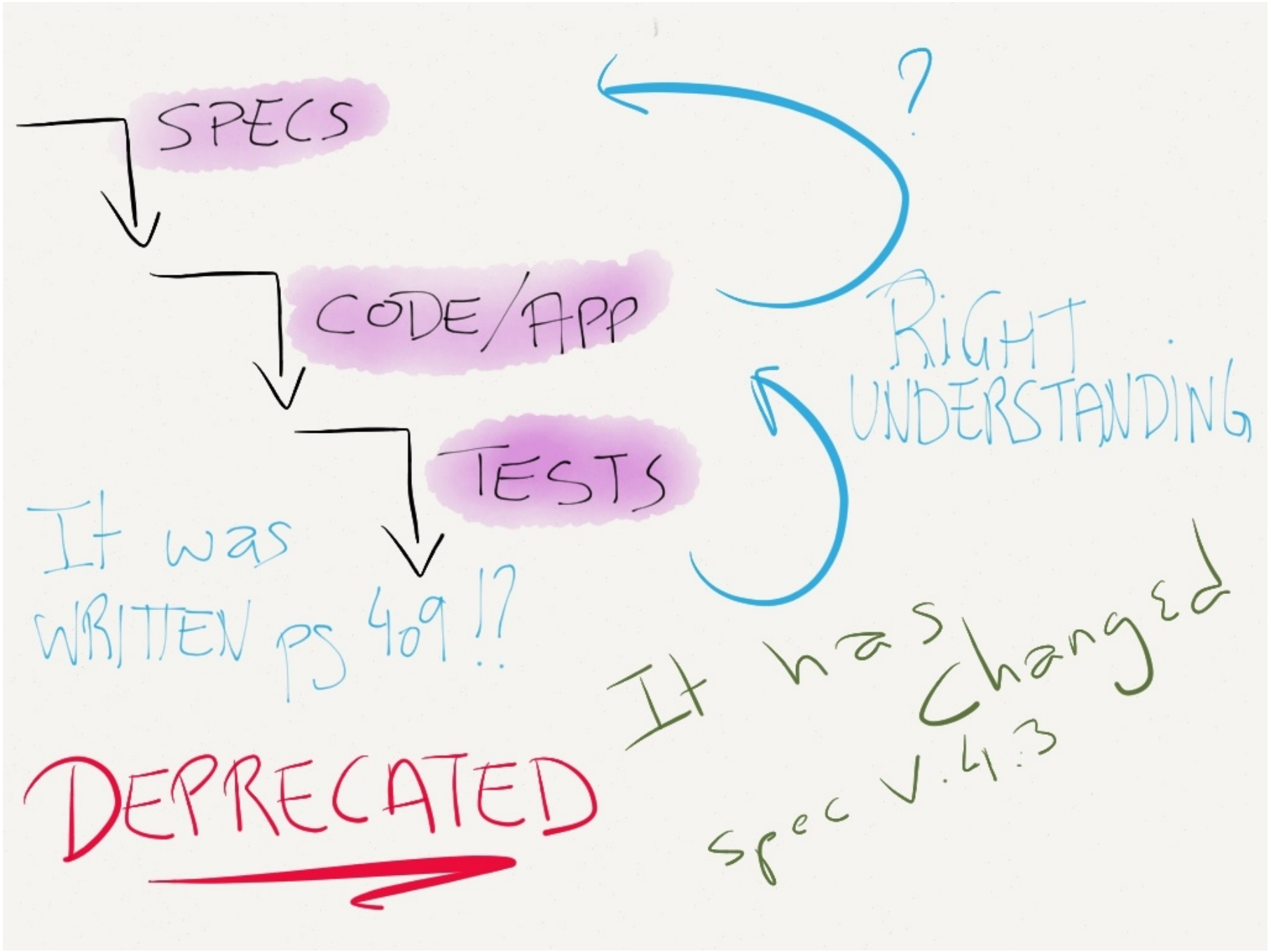

SPECS
CODE/APP
TESTS
Ooops!

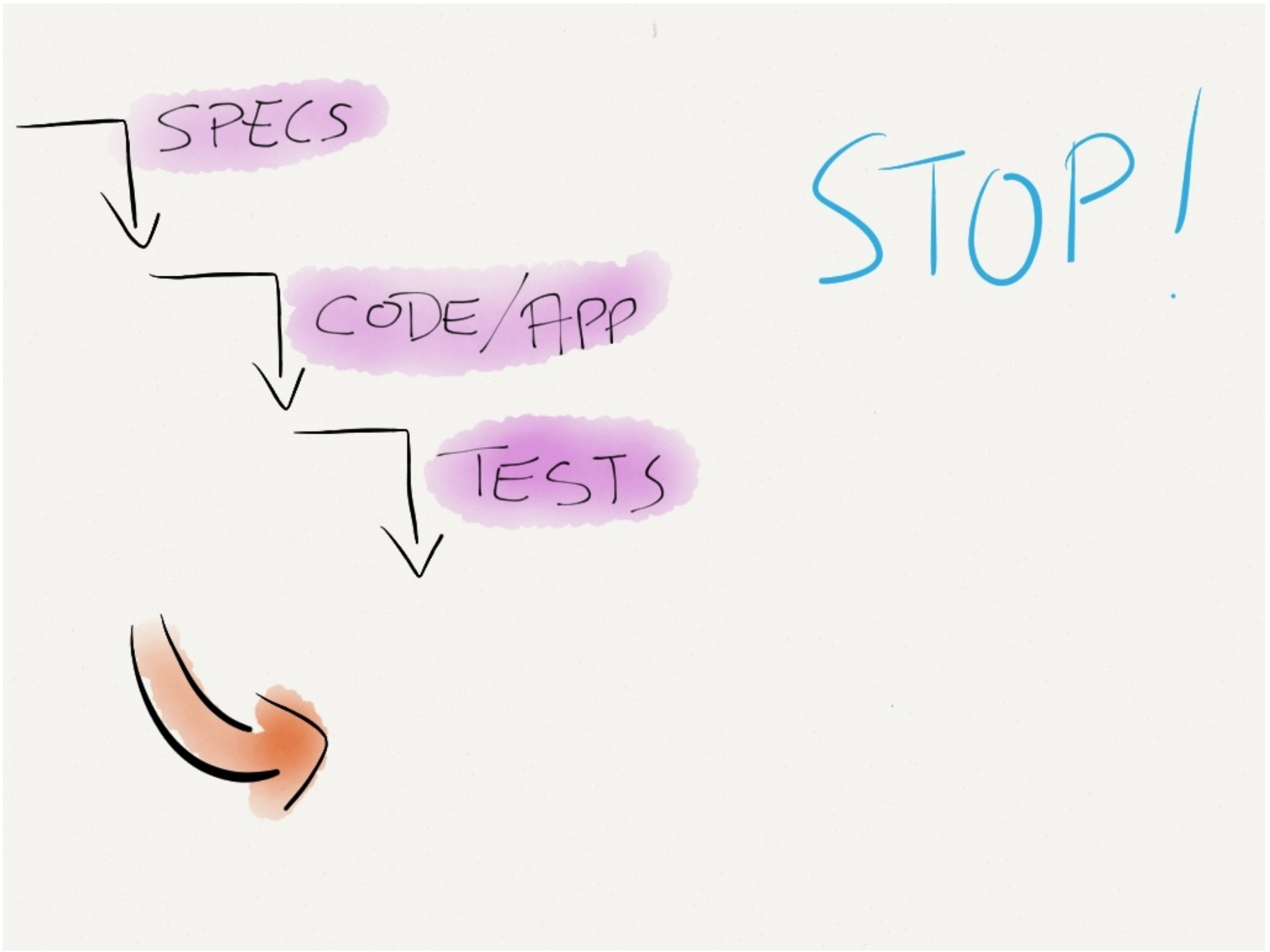

SPECS
CODE/APP
TESTS
TOO LATE ?!

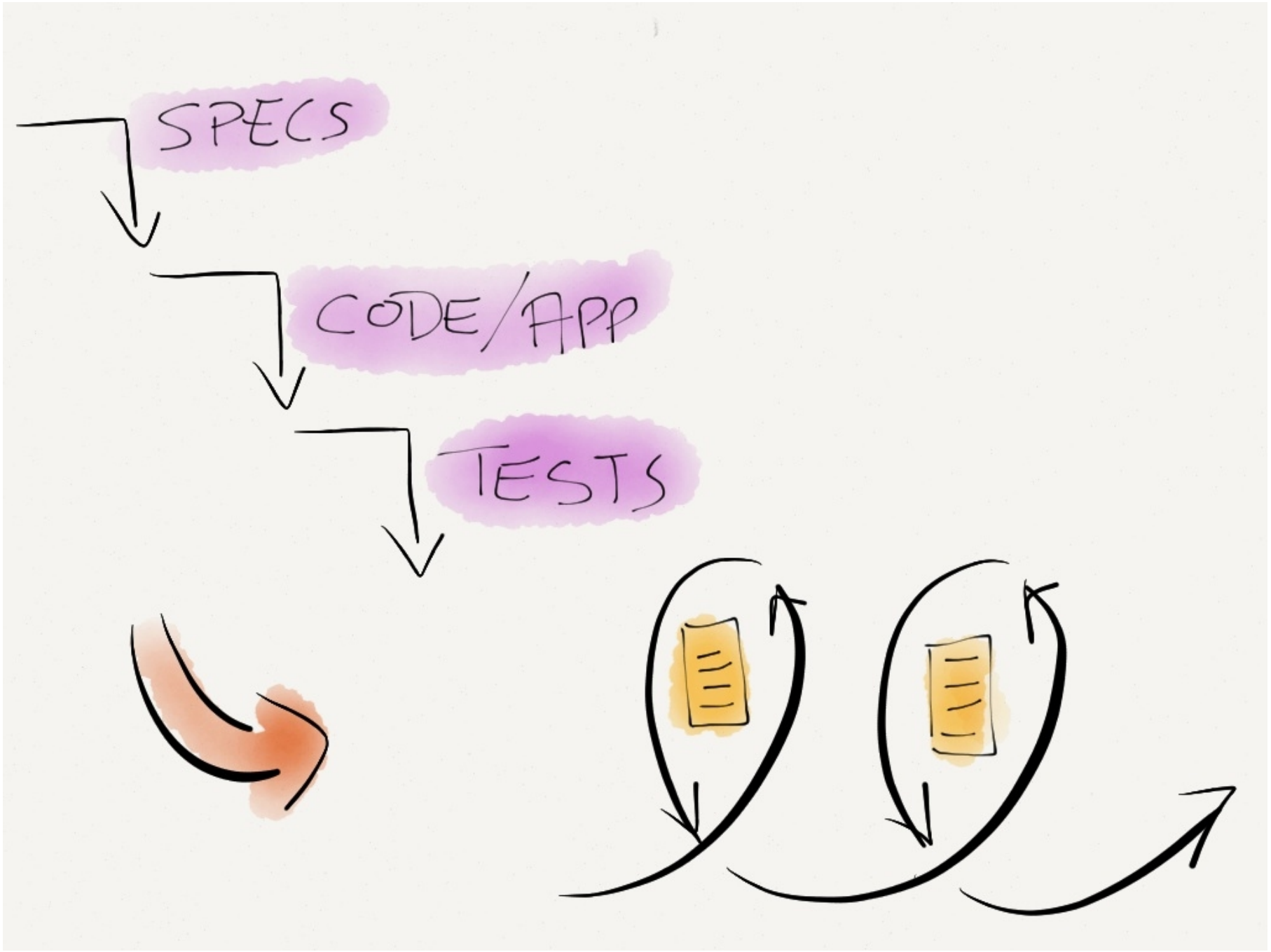

SPECS
CODE/APP
TESTS
RIGHT UNDERSTANDING
?

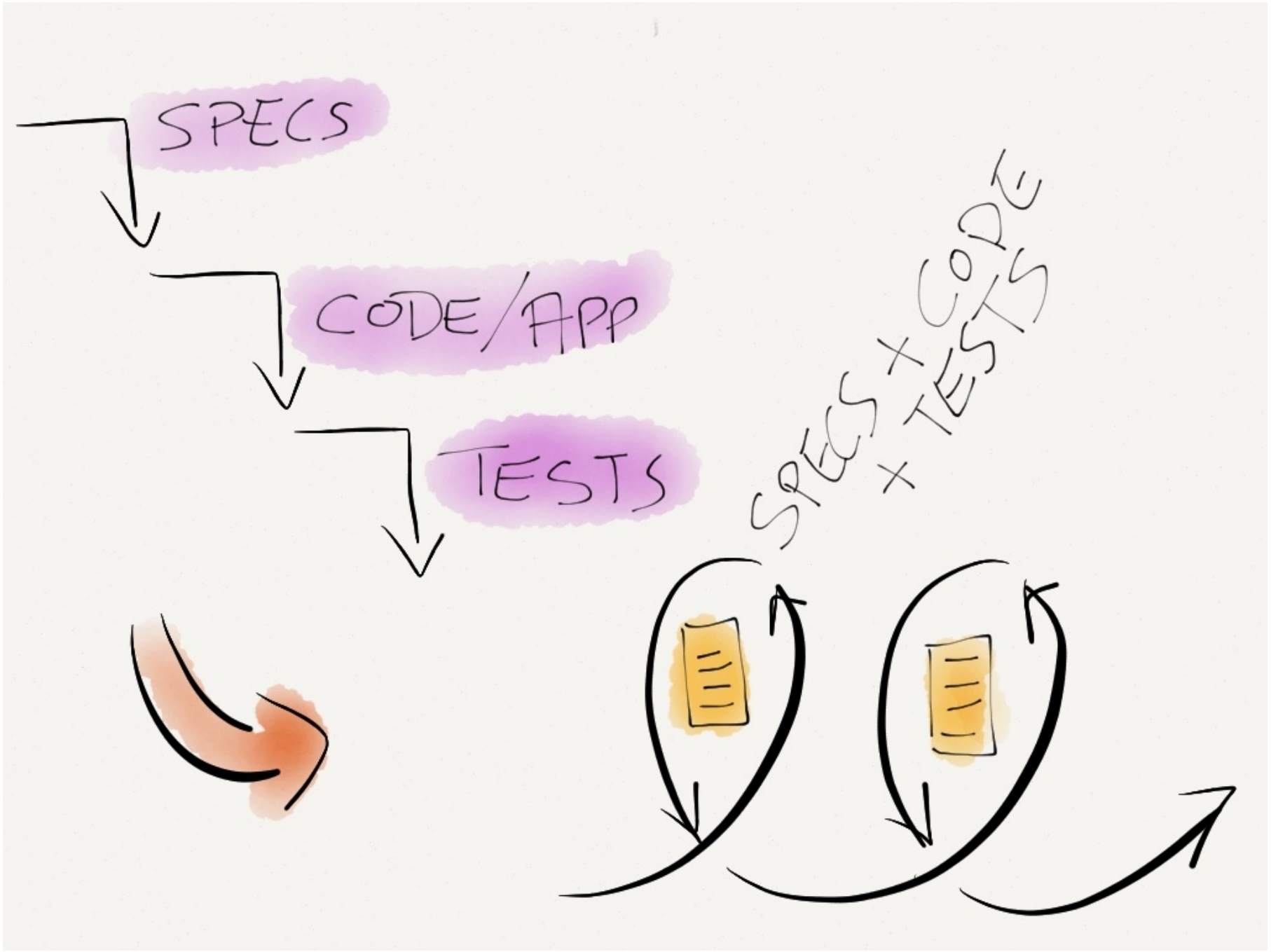

SPECS
CODE/APP
TESTS
?
RIGHT
UNDERSTANDING
It was
WRITTEN PS 409 !?

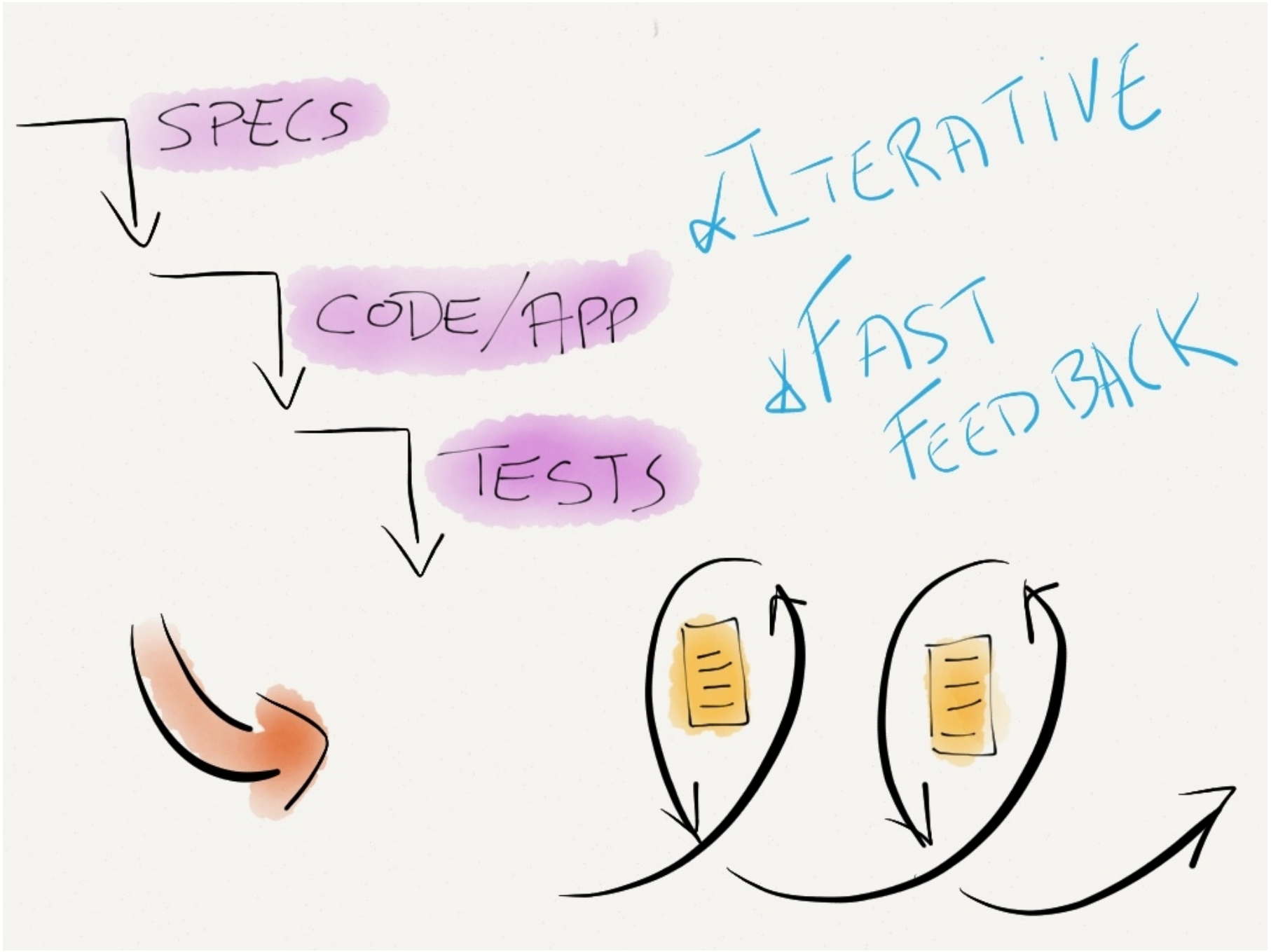

SPECS
CODE/APP
TESTS
?
RIGHT UNDERSTANDING
It was WRITTEN ps 409 !?
It has changed
spec V.4.3


SPECS
CODE/APP
TESTS
?
RIGHT
UNDERSTANDING
It was
WRITTEN PS 409 !?
DEPRECATED
It has changed
spec V.4.3

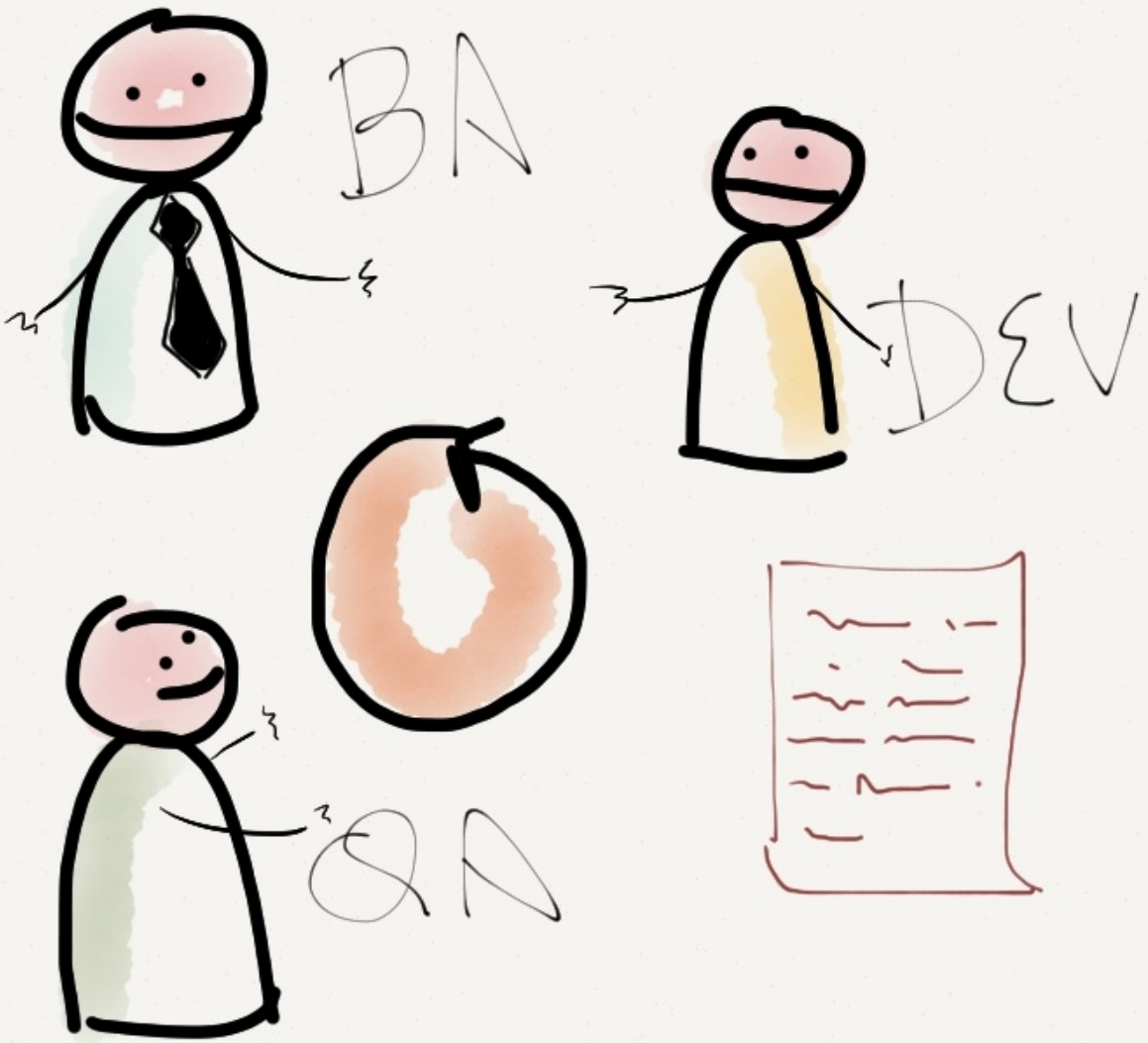

SPECS
CODE/APP
TESTS
STOP!

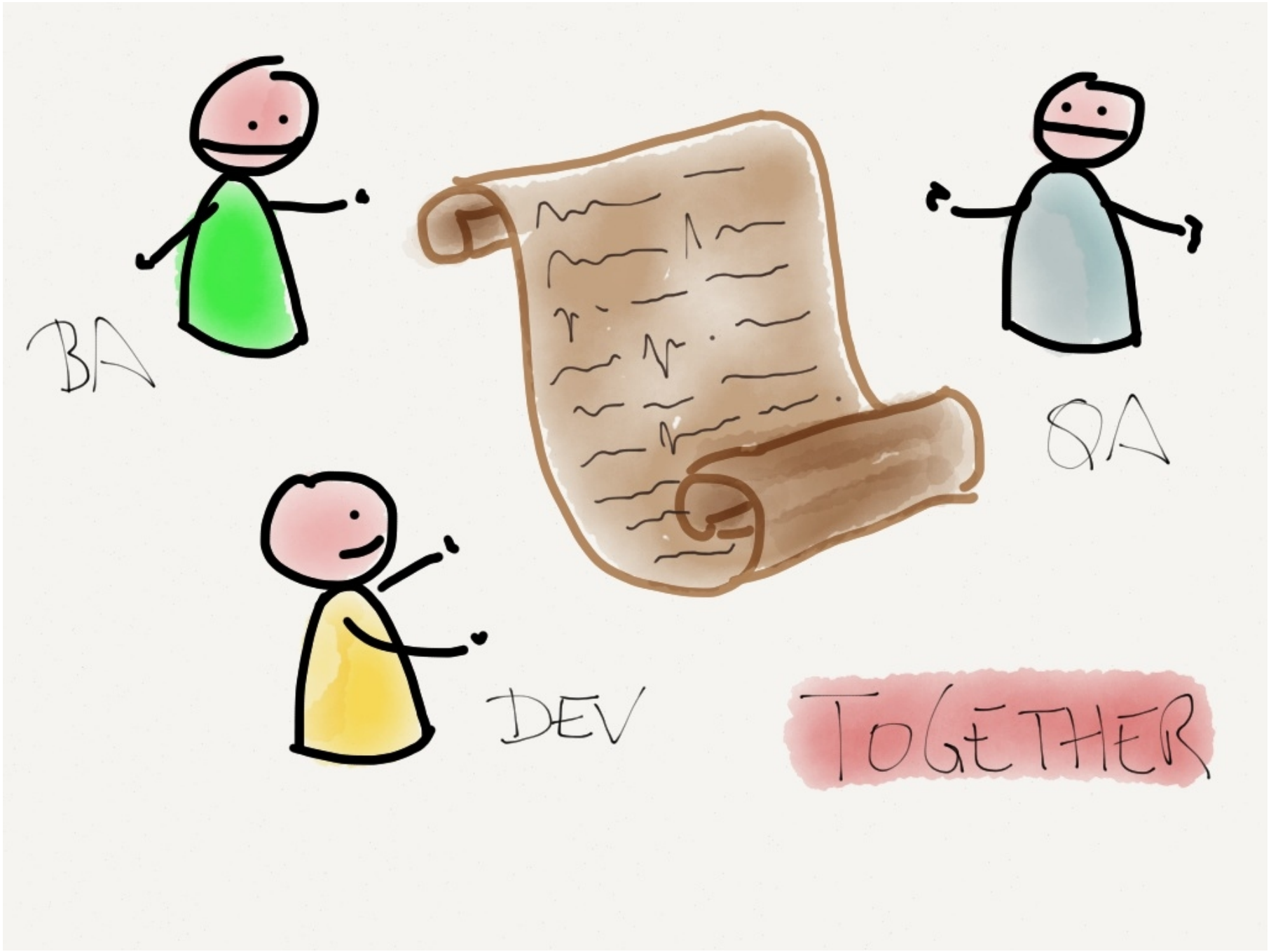

SPECS
CODE/APP
TESTS

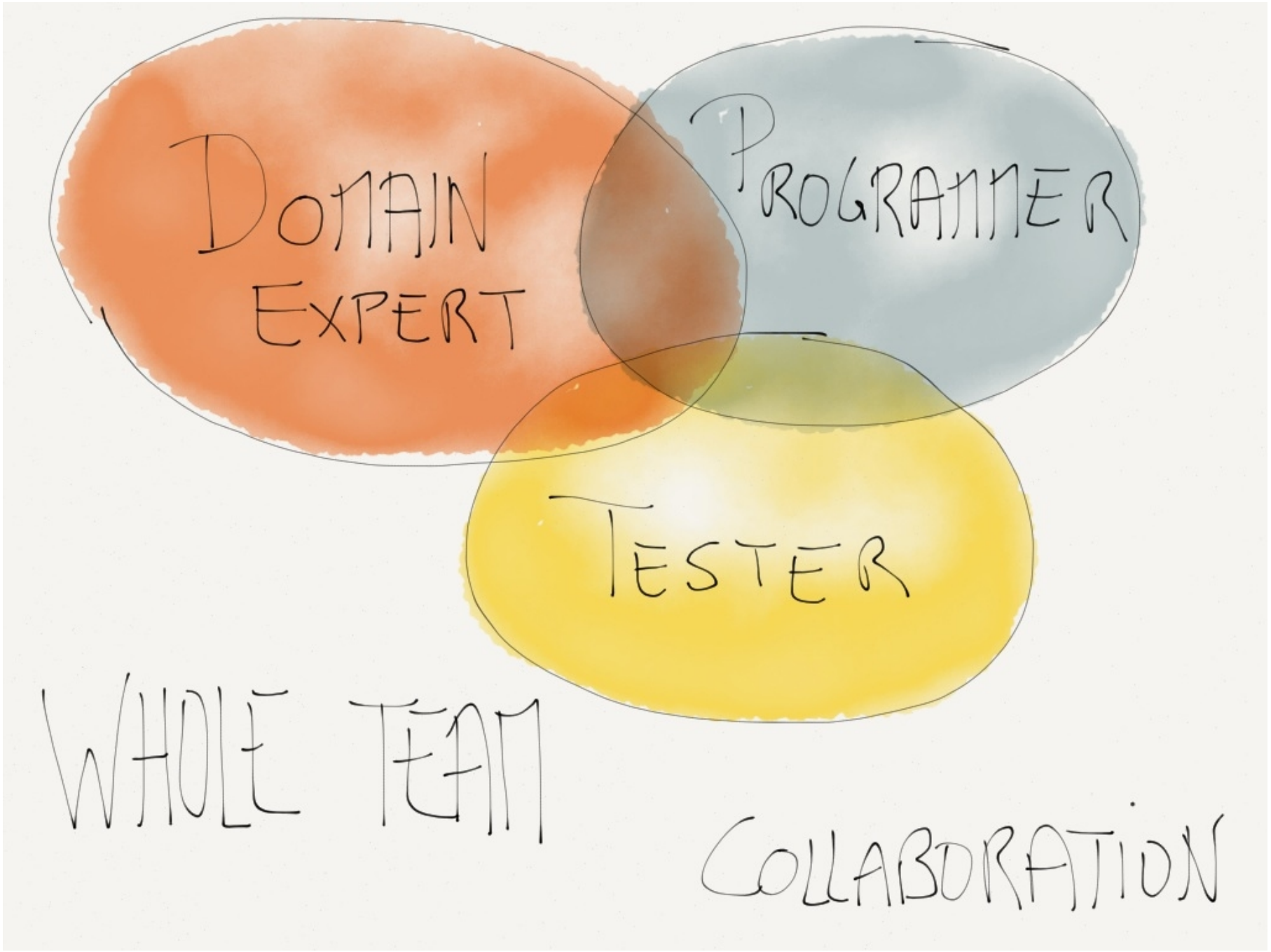

SPECS
CODE/APP
TESTS
SPECS x CODE x TESTS


SPECS
CODE/APP
TESTS
ITERATIVE
FAST
FEEDBACK

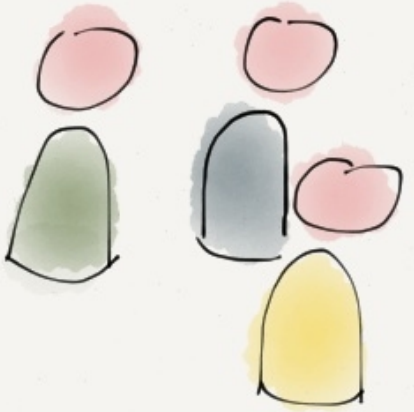
Three Amigos


BA
DEV
QA


BA
QA
DEV
TOGETHER


DOMAIN EXPERT
PROGRAMMER
TESTER
WHOLE TEAM
COLLABORATION

Three Amigos



"Power of Three" - Lisa Crispin & Janet Gregory

Ken Pugh's Triad

SCENARIO

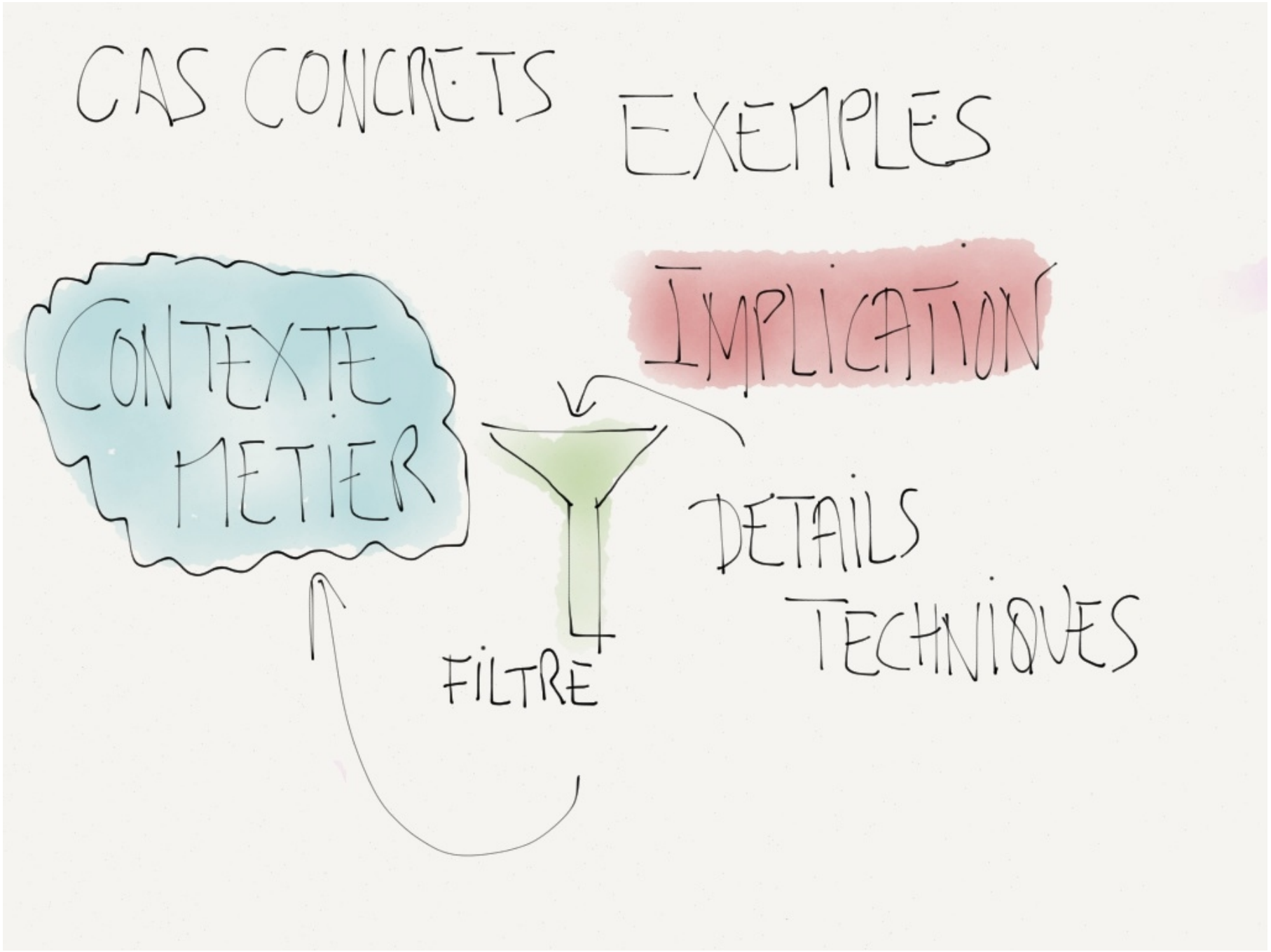
TEXT FILE

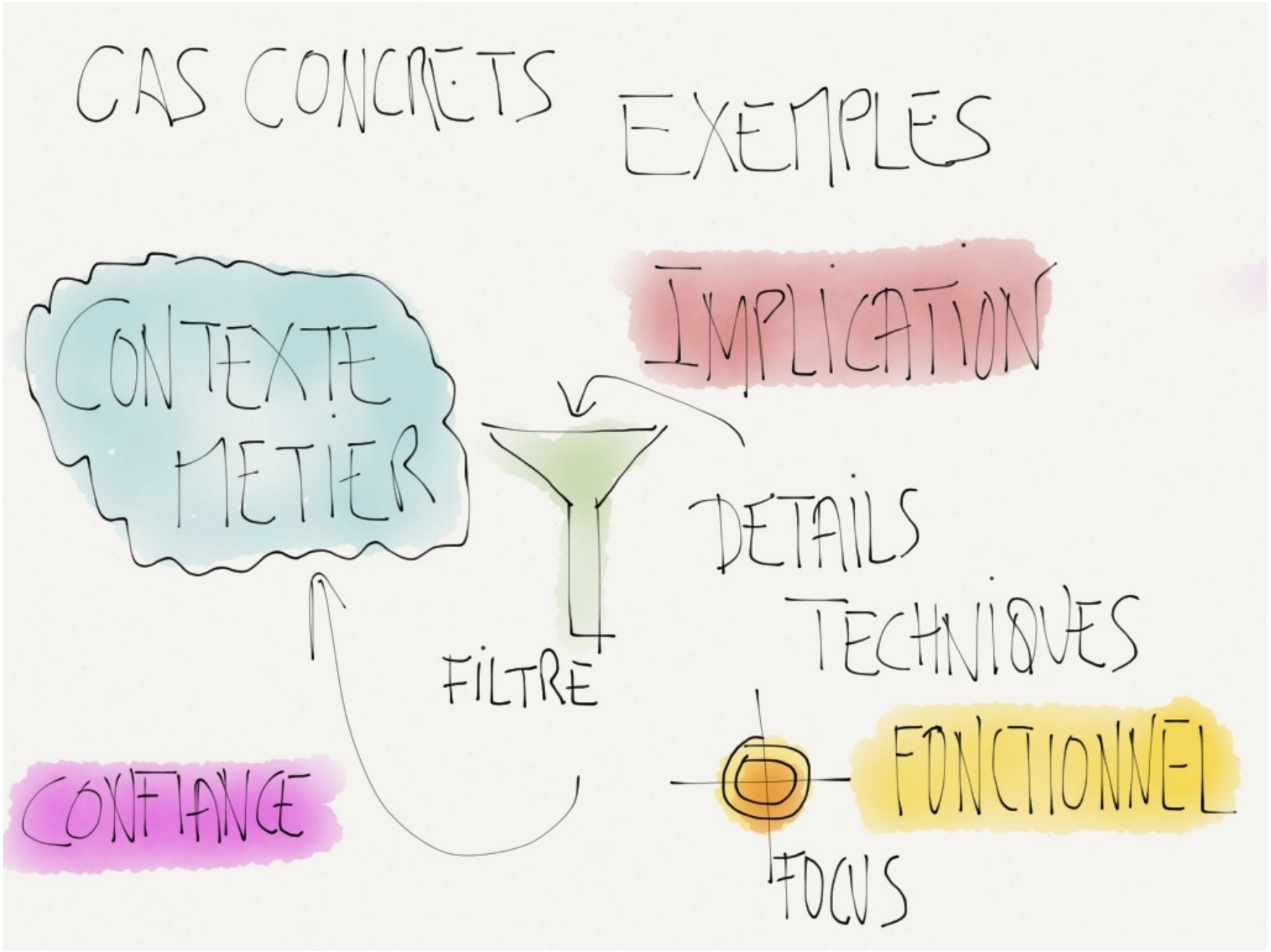
CAS CONCRETS

EXEMPLES

CONTEXTE METIER

IMPLICATION


CAS CONCRETS
EXEMPLES
CONTEXTE METIER
IMPLICATION
DETAILS TECHNIQUES
FILTRE

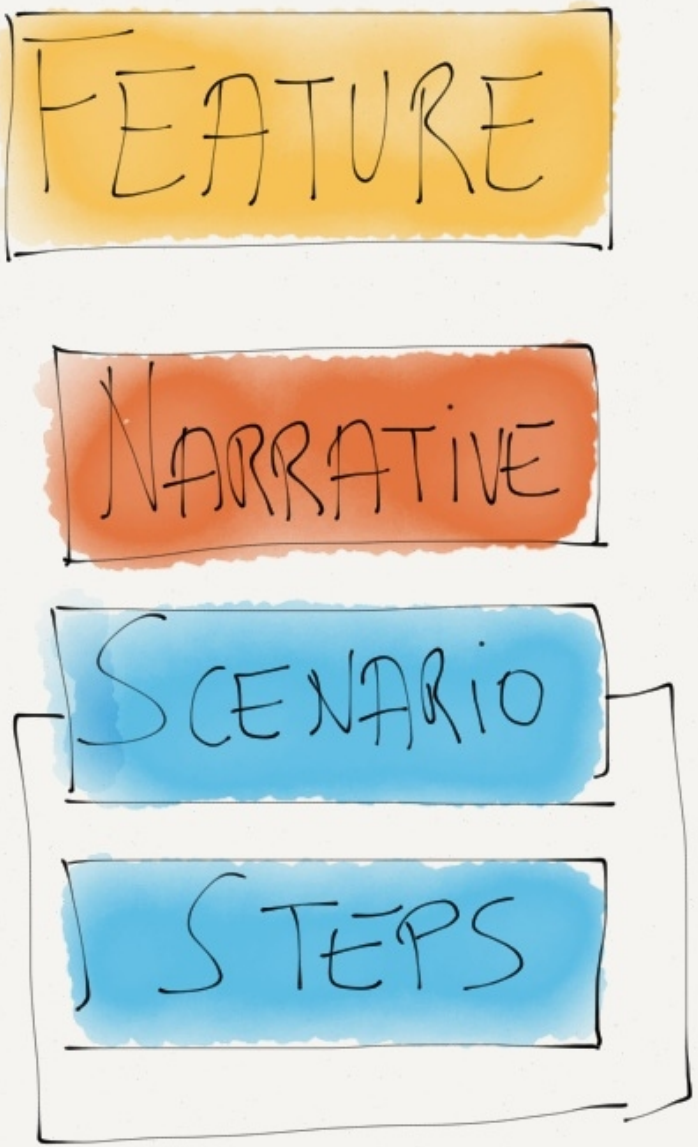

CAS CONCRETS
EXEMPLES
CONTEXTE METIER
IMPLICATION
DETAILS TECHNIQUES
FILTRE
CONFIANCE
FONCTIONNEL
FOCUS

SCENARIO

STRUCTURATION

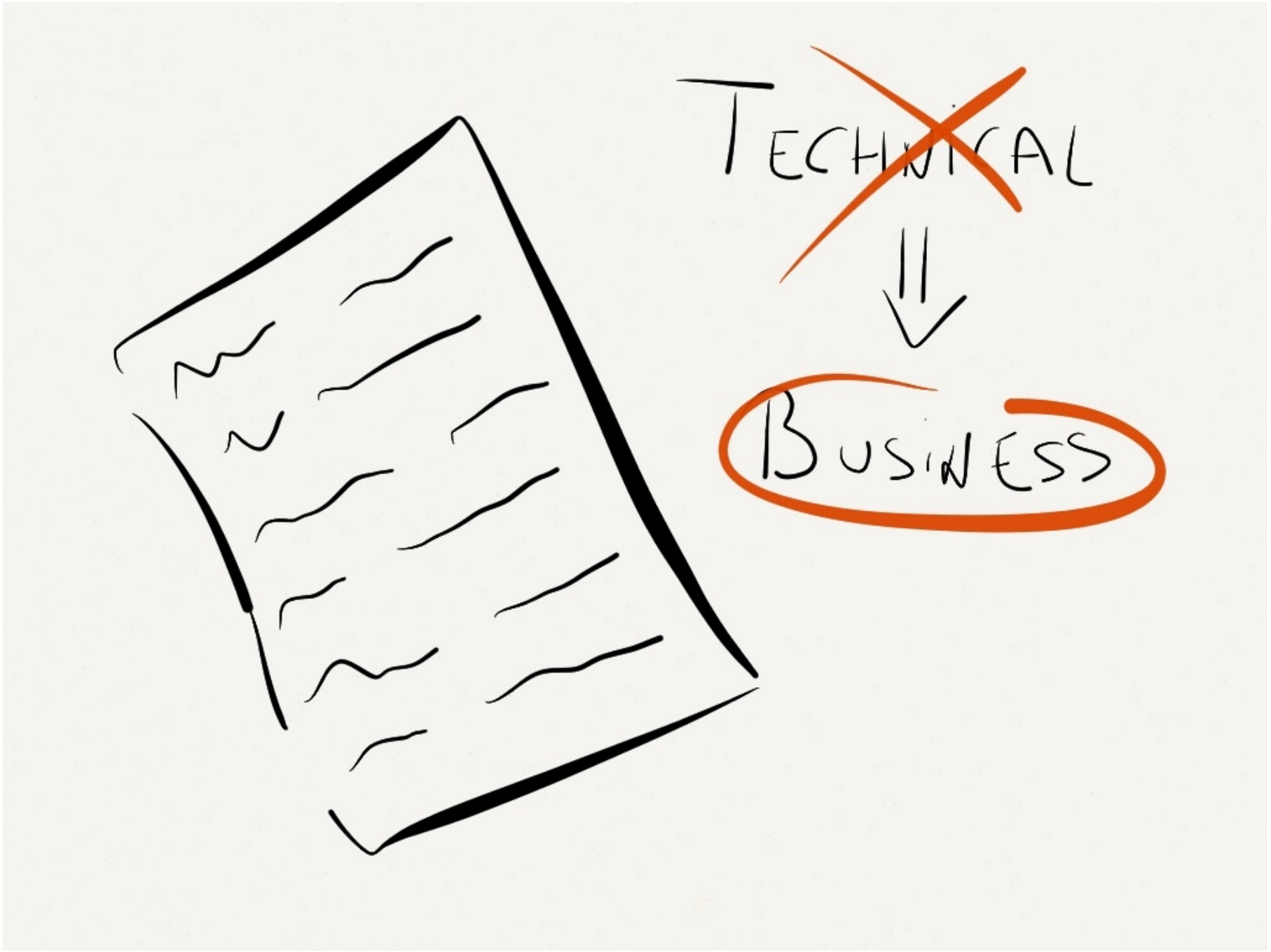
AS A ...
IN ORDER TO ...
I WANT ...

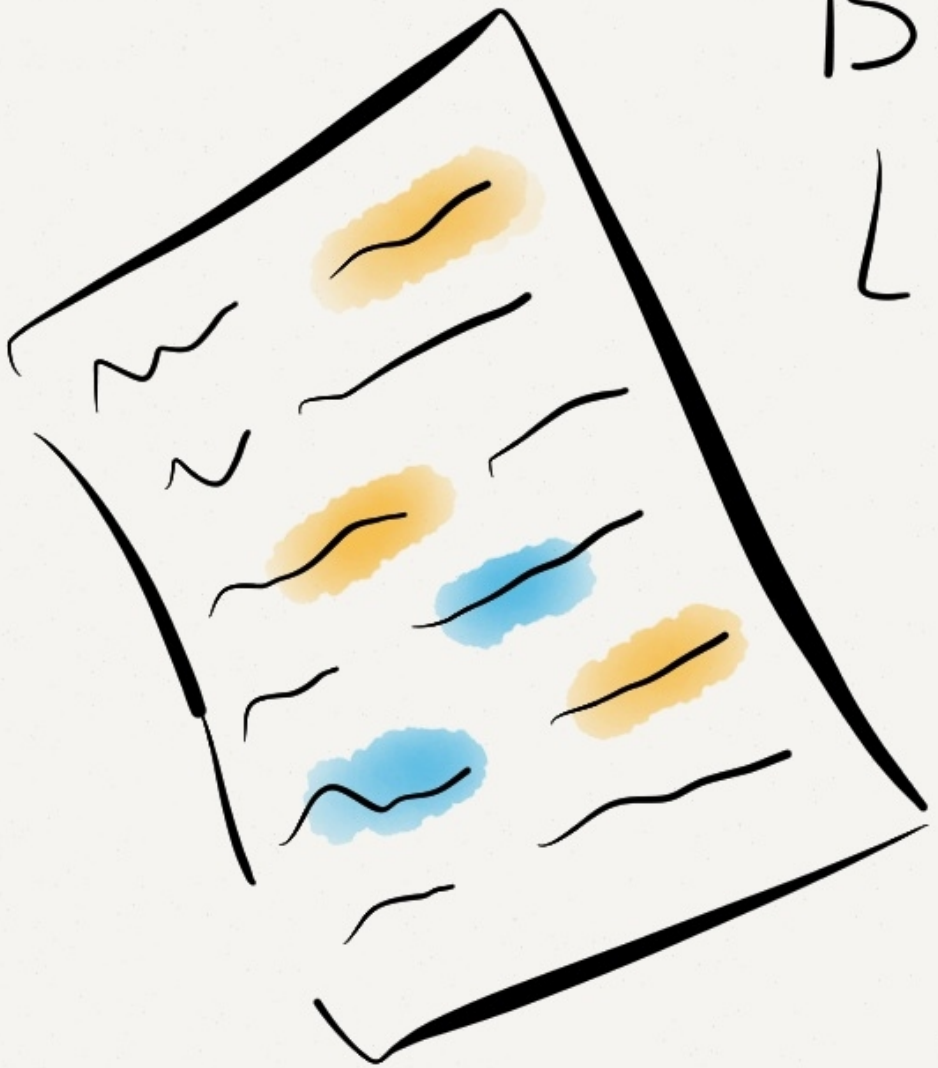
GIVEN ...
WHEN ...
THEN ...



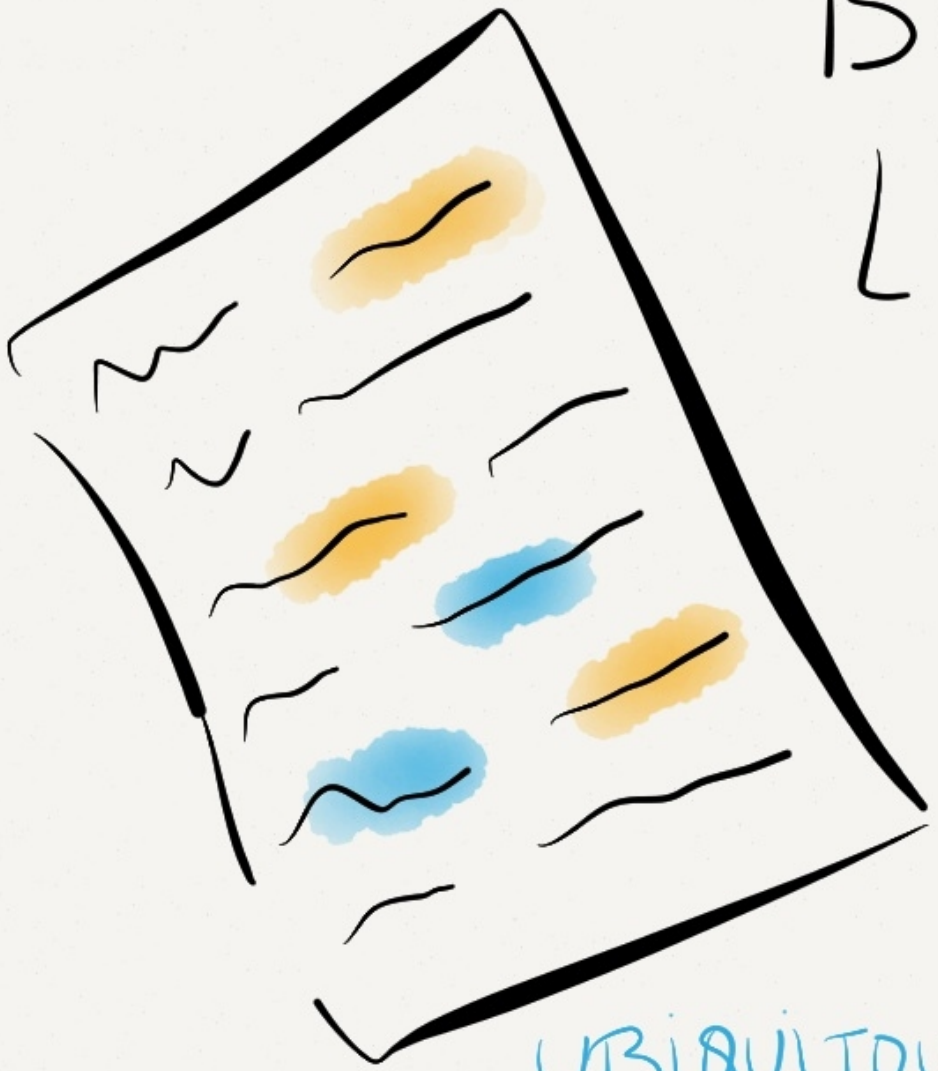
AS A ...
IN ORDER TO ...
I WANT ...

GIVEN ...
WHEN ...
THEN ...


TECHNICAL
BUSINESS



BUSINESS LANGUAGE EMERGES



BUSINESS LANGUAGE

↓

CODE

UBIQUITOUS LANGUAGE

DDD

Concrete Examples

Feature: Account Holder withdraws cash

As an Account Holder

I want to withdraw cash from an ATM

So that I can get money when the bank is closed

Scenario: Account has sufficient funds

Given the account balance is 100€

And the card is valid

And the machine contains enough money

When the Account Holder requests 20€

Then the ATM should dispense 20€

And the account balance should be 80€

And the card should be returned

Scenario: The ATM has insufficient funds

Given the account balance is 100€

And the card is valid

And the machine contains only 20€

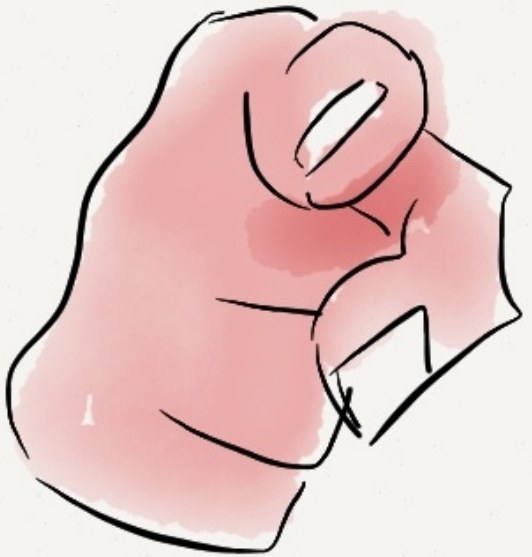
When the Account Holder requests 30€

Then the ATM should say it has insufficient funds

And the account balance should be 100€

And the card should be returned

Your Turn



Scenario: Account has insufficient funds

Given the account balance is 10€

And the card is valid

And the machine contains enough money

When the Account Holder requests 30€

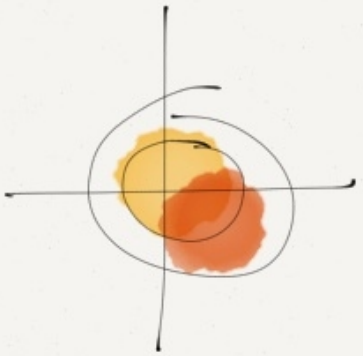
Then the ATM should not dispense any money

And the ATM should say there are insufficient funds

And the account balance should still be 10€

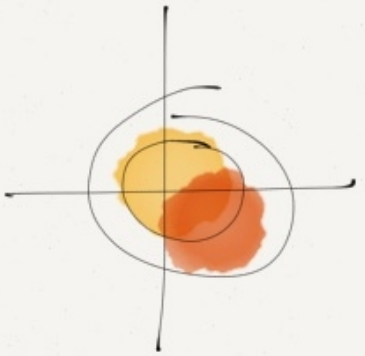
And the card should be returned

IMPLICIT
VS
EXPLICIT



Focus on the behavior described

Declarative
vs
Imperative



Focus on the
behavior described

Scenario: Wrong PIN

Given the card is valid

And its PIN number is "0000"

When the Account Holder enters "1234"

Then the ATM should say the PIN number is wrong

Scenario: Wrong PIN three times

Given the card is valid

And its PIN number is "0000"

When the Account Holder enters "1234"

And the Account Holder enters "4321"

And the Account Holder enters "2341"

Then the ATM should retain the card

And the ATM should say the card has been retained

Scenario: Card has been disabled

Given the card is disabled

When the Account Holder requests 30€

Then the ATM should retain the card

And the ATM should say the card has been retained

Scenario: Card has been disabled

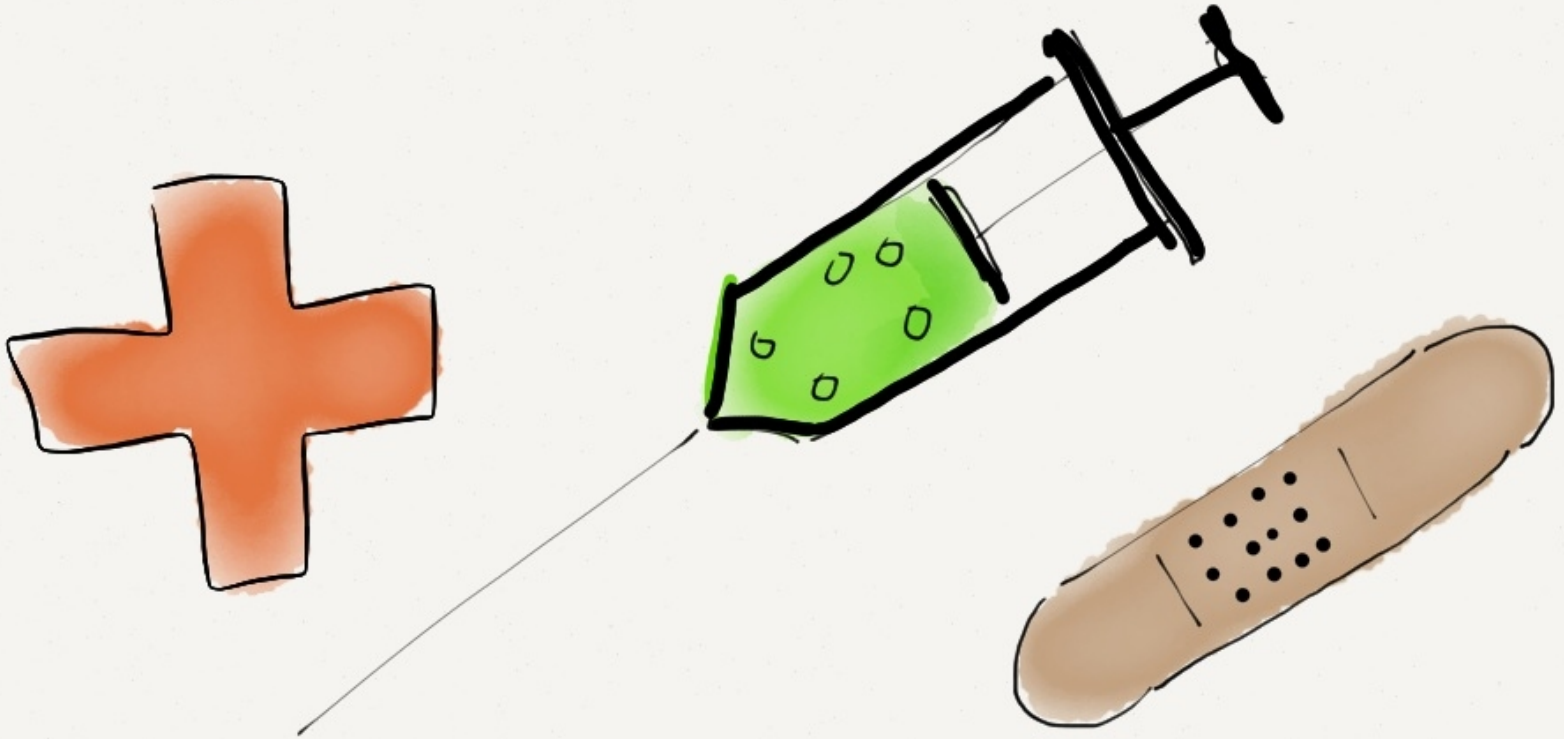
Given the card is disabled

When the Account Holder requests 30€

Then the ATM should retain the card

And the ATM should say the card has been retained

BDD - CLINIC



Why this feature ?

Feature: Interpolate

In order to interpolate values

As an Trader

I want to interpolate values in a range of Market data

Why this feature ?

Feature: Interpolate

In order to interpolate values

As an Trader

I want to interpolate values in a range of Market data

Why this feature ?

Feature: Linear Interpolation

In order to fill the gaps and provide a value for any maturity

As a trader responsible for market-marking

I want to interpolate linearly values within a range of points

And **I want** a flat extrapolation outside of the range of points

What about this scenario

Scenario: Change the negotiation price from positive to negative => soulte cashflow appears and premium cashflow is modified

Given an FUNKY_EXOTIC
And deal way is sell
And deal nature is TOMATO
And trade value date is 2012/07/01
And nominal is 100 JPY
And negotiation price is 0.20 JPY
When I validate the deal
Then there are 1 Price cashflows
And there are 0 fee cashflows
When I change the negotiation price to -0.3 JPY
And I validate the deal
Then there are 1 Price cashflows
And there are 1 fee cashflows
And the trade cashflow's payment date is 2012/07/01
And the trade cashflow's way is receive
And the trade cashflow's amount is 30 JPY
And the fee cashflow's payment date is 2012/07/01
And the fee cashflow's way is give
And the fee cashflow's amount is 60 JPY

What about this scenario

Scenario: Change the negotiation price from positive to negative => soulte cashflow appears and premium cashflow is modified

Given an FUNKY_EXOTIC
And deal way is sell
And deal nature is TOMATO
And trade value date is 2012/07/01
And nominal is 100 JPY
And negotiation price is 0.20 JPY
When I validate the deal
Then there are 1 Price cashflows
And there are 0 fee cashflows
When I change the negotiation price to -0.3 JPY
And I validate the deal
Then there are 1 Price cashflows
And there are 1 fee cashflows
And the trade cashflow's payment date is 2012/07/01
And the trade cashflow's way is receive
And the trade cashflow's amount is 30 JPY
And the fee cashflow's payment date is 2012/07/01
And the fee cashflow's way is give
And the fee cashflow's amount is 60 JPY

```
d = new Deal();
d.SetWay(Sell);
d.SetNature(Tomato);
d.SetValueDate(new Date(...));
d.SetNominal(100, JPY);
d.SetNegotiationPrice(0.20, JPY);
cf = d.GetCashFlows();
AssertThat(IsEqual(...);
...
```

What about this scenario

Scenario: Fee and Price cashflows when the negotiation price is set to a negative value

Given a sell for a nominal 100 JPY on FUNKY_EXOTIC TOMATO negotiation price 0.20 JPY traded on 2012/07/01

When the middle officer validates the deal

Then the trade has one Price cashflow and no Fee cashflow

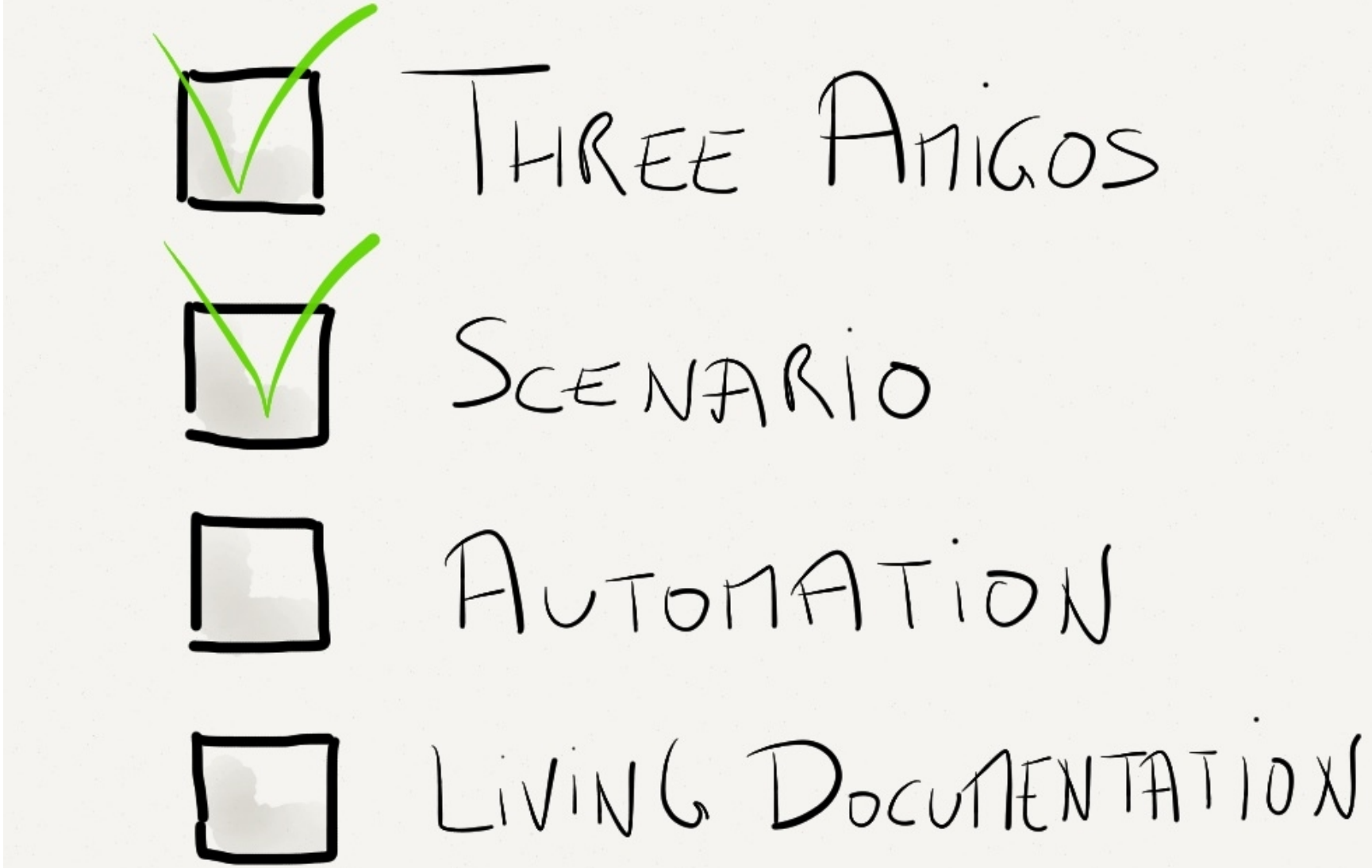
When the middle officer changes the negotiation price to -0.3 JPY

And the middle officer validates the deal

Then the trade has the following cashflows:

FlowType	Way	Amount	Currency	PaymentDate	Remarks
Price	Receive	30	JPY	2012/07/01	100*abs(-0.3)
Fee	Give	60	JPY	2012/07/01	100*2*abs(-0.3)

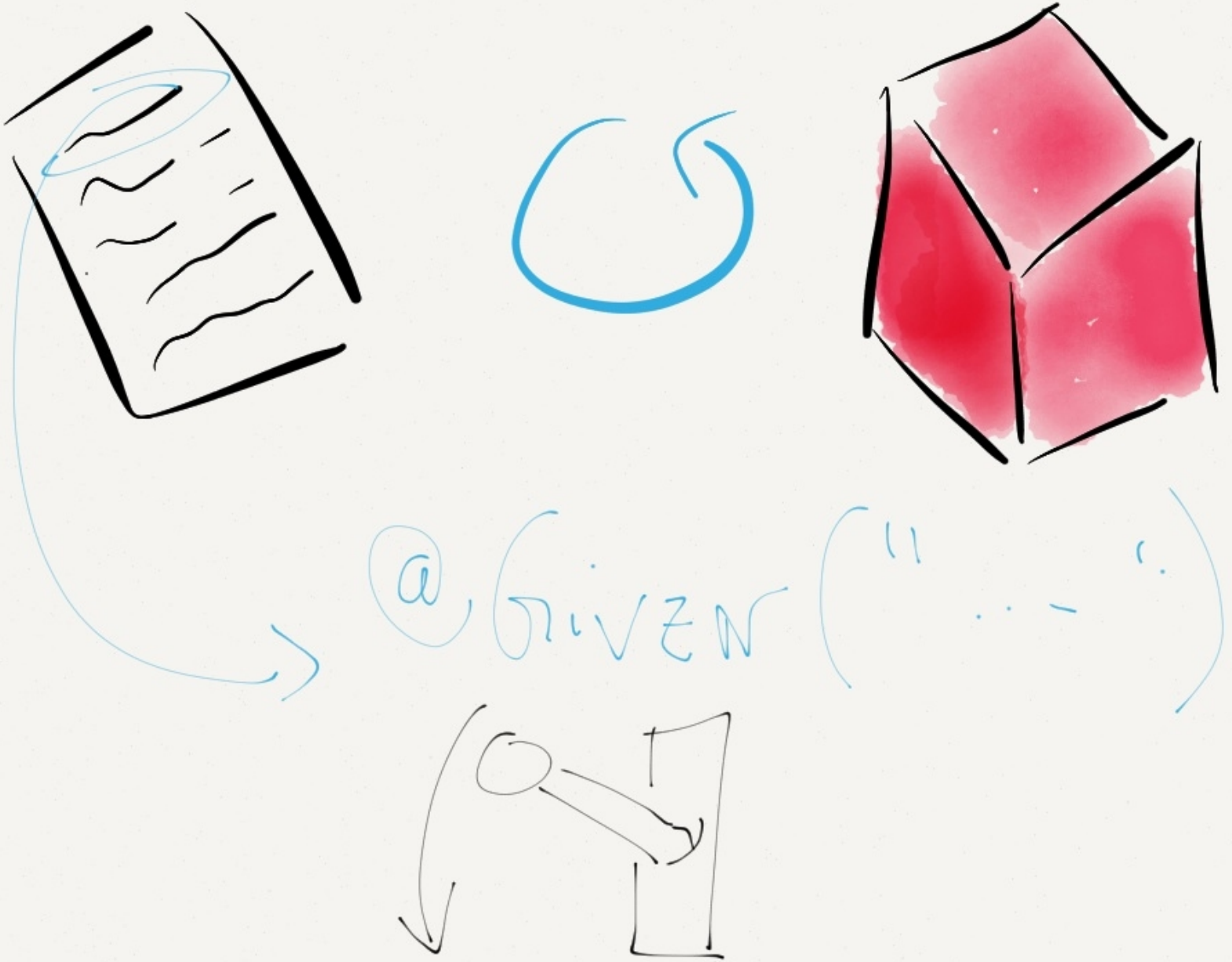
Communicate With the Business People !!!

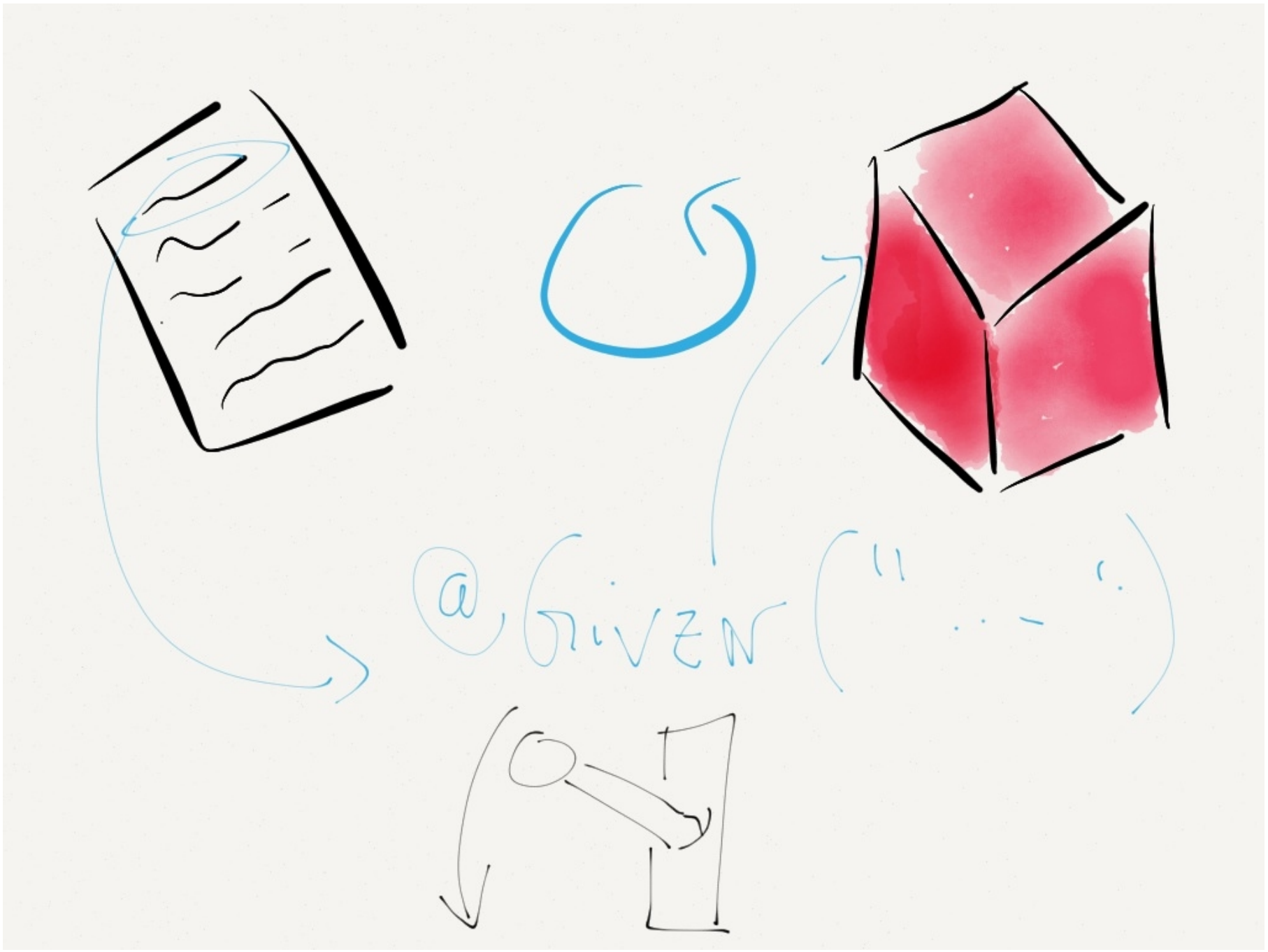

THREE AMIGOS
SCENARIO
AUTOMATION
LIVING DOCUMENTATION

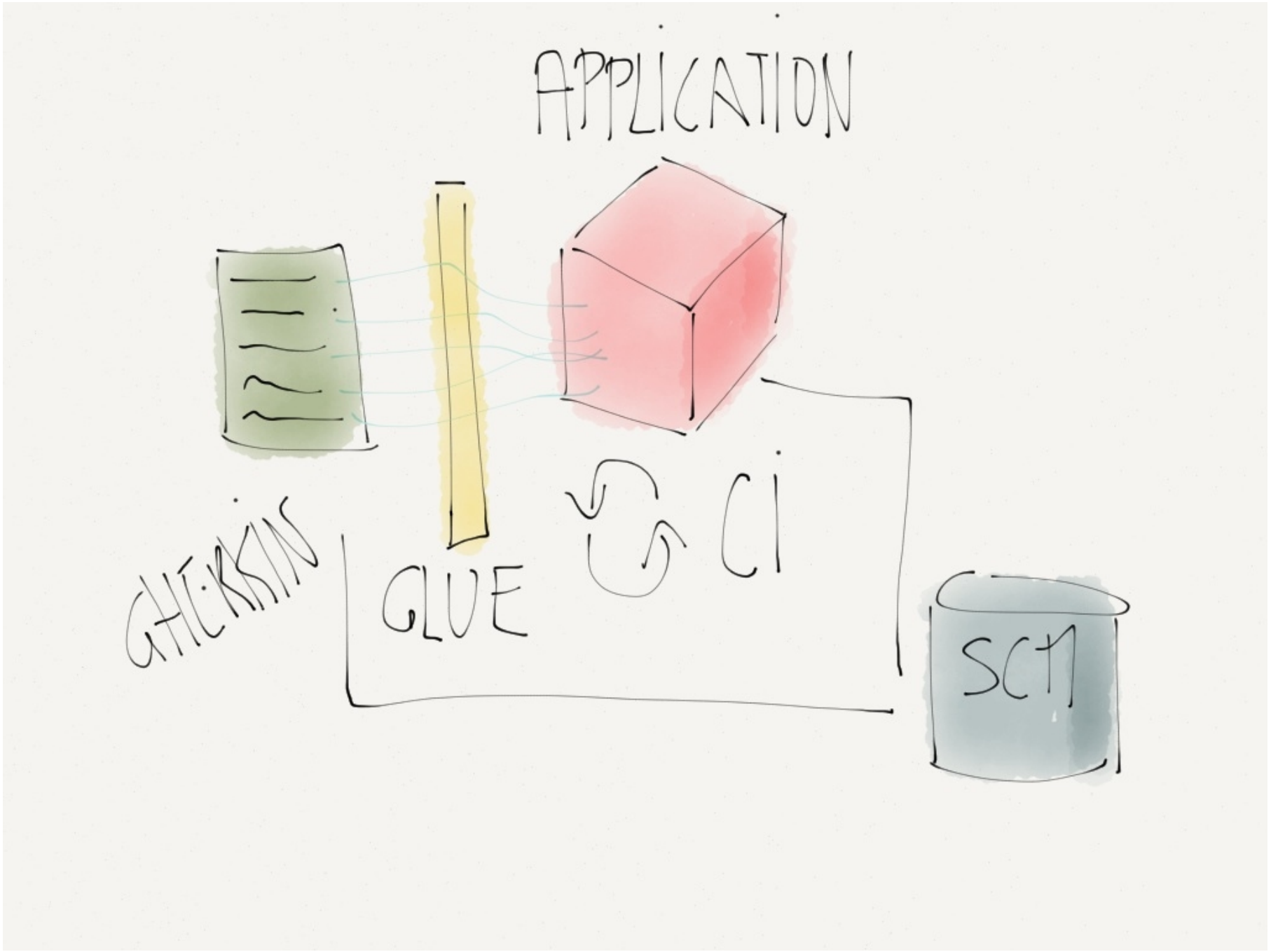
AUTOMATION

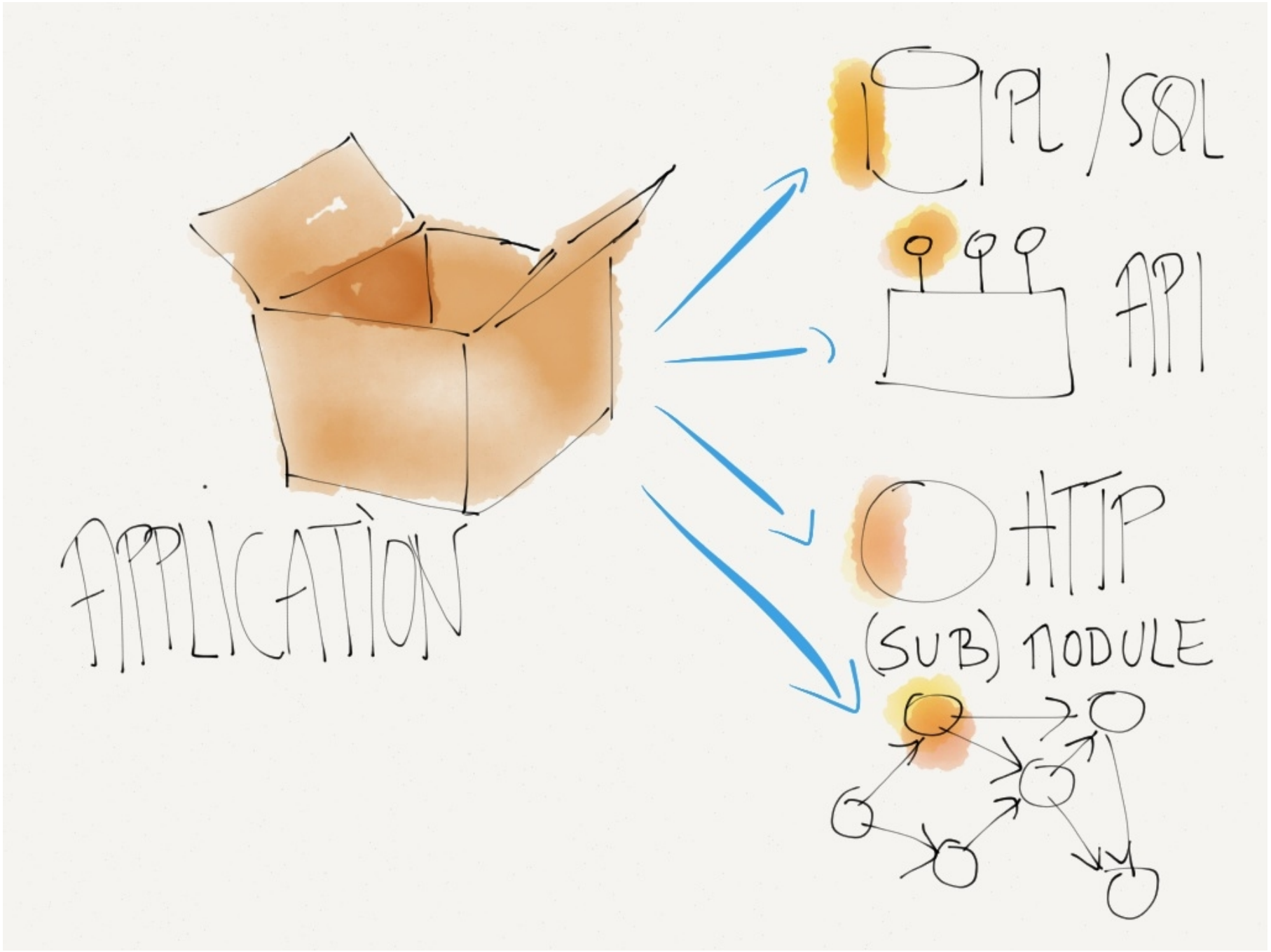
BRIDGE BETWEEN
SCENARIO & APP.


GLUE
CUCUMBER
SPEC FLOW


@given(" ... ")


@given(...)

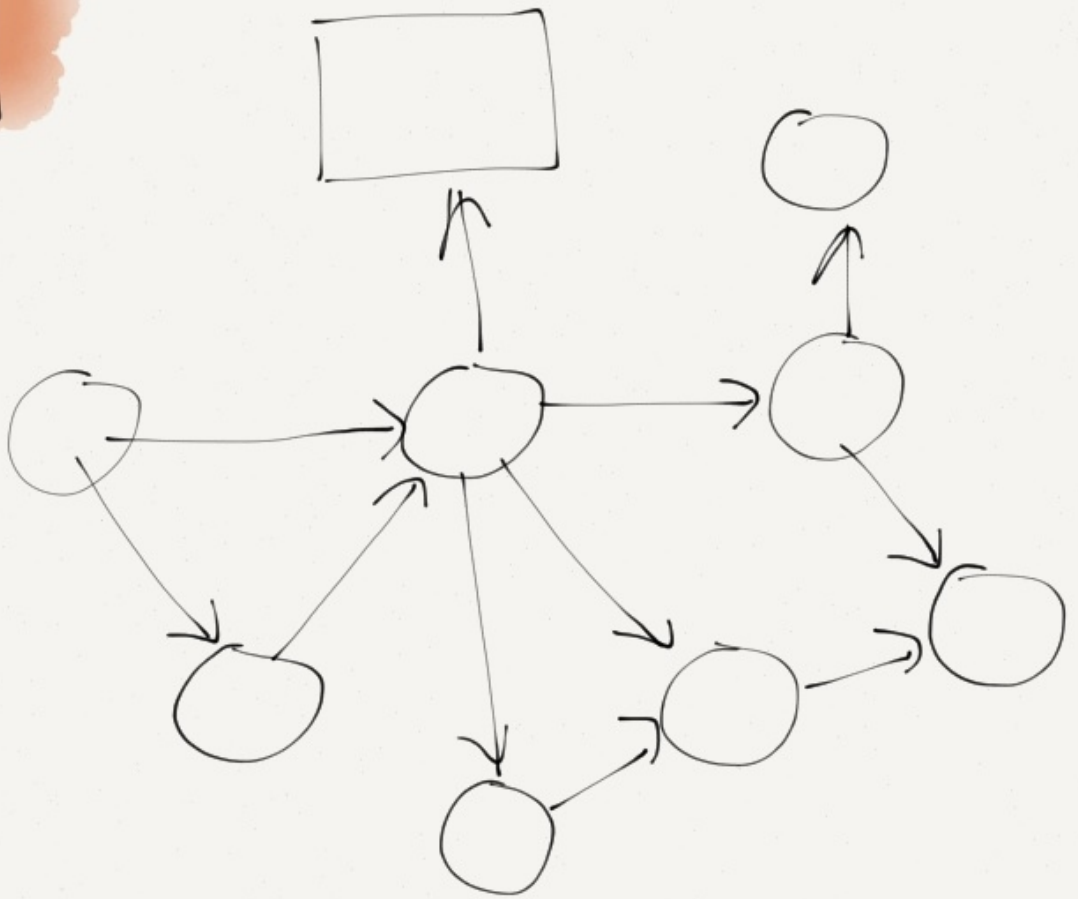

APPLICATION
GHERKIN
GLUE
CI
SCM

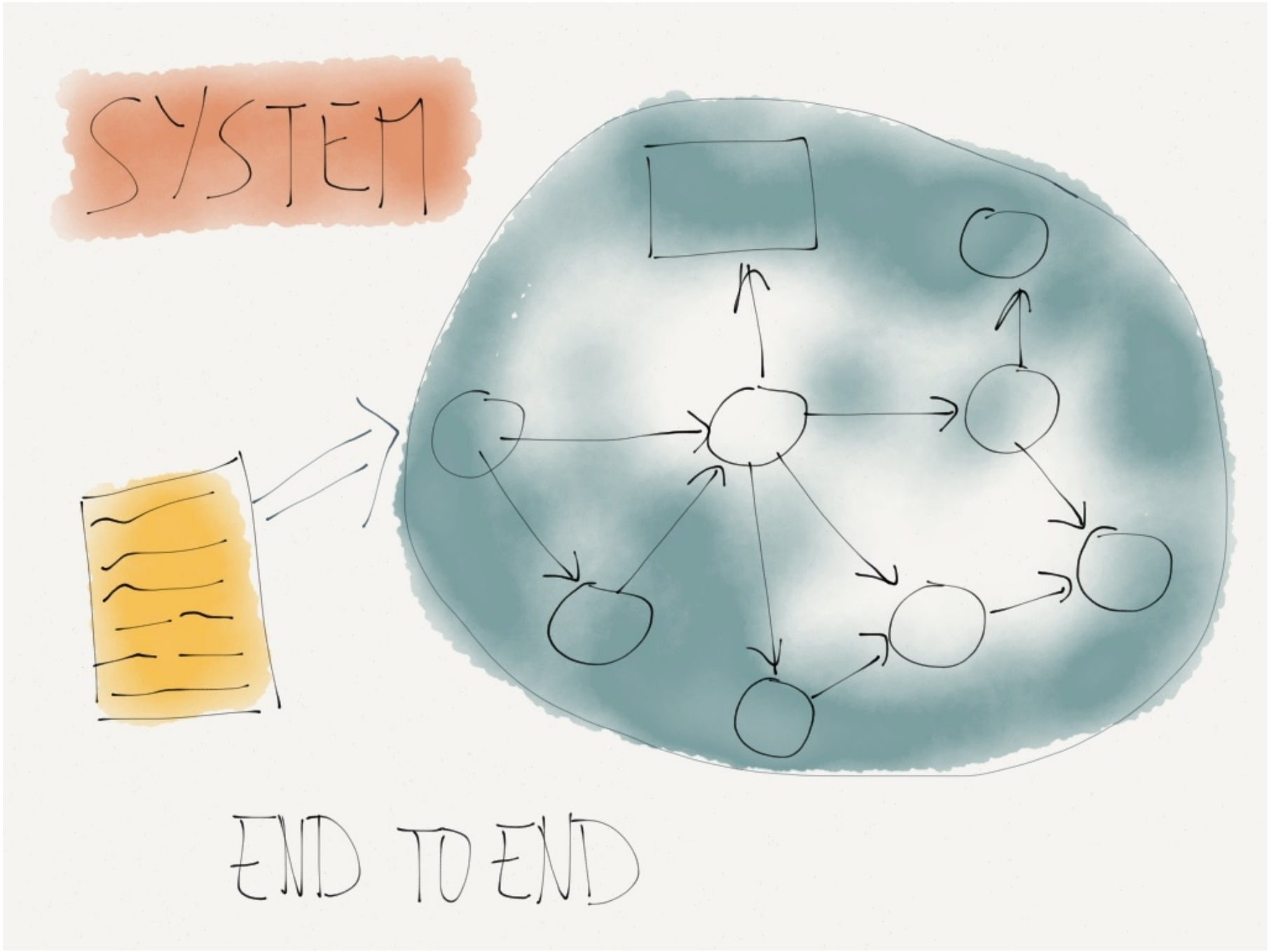

PL/SQL
API
HTTP
(SUB) MODULE
APPLICATION

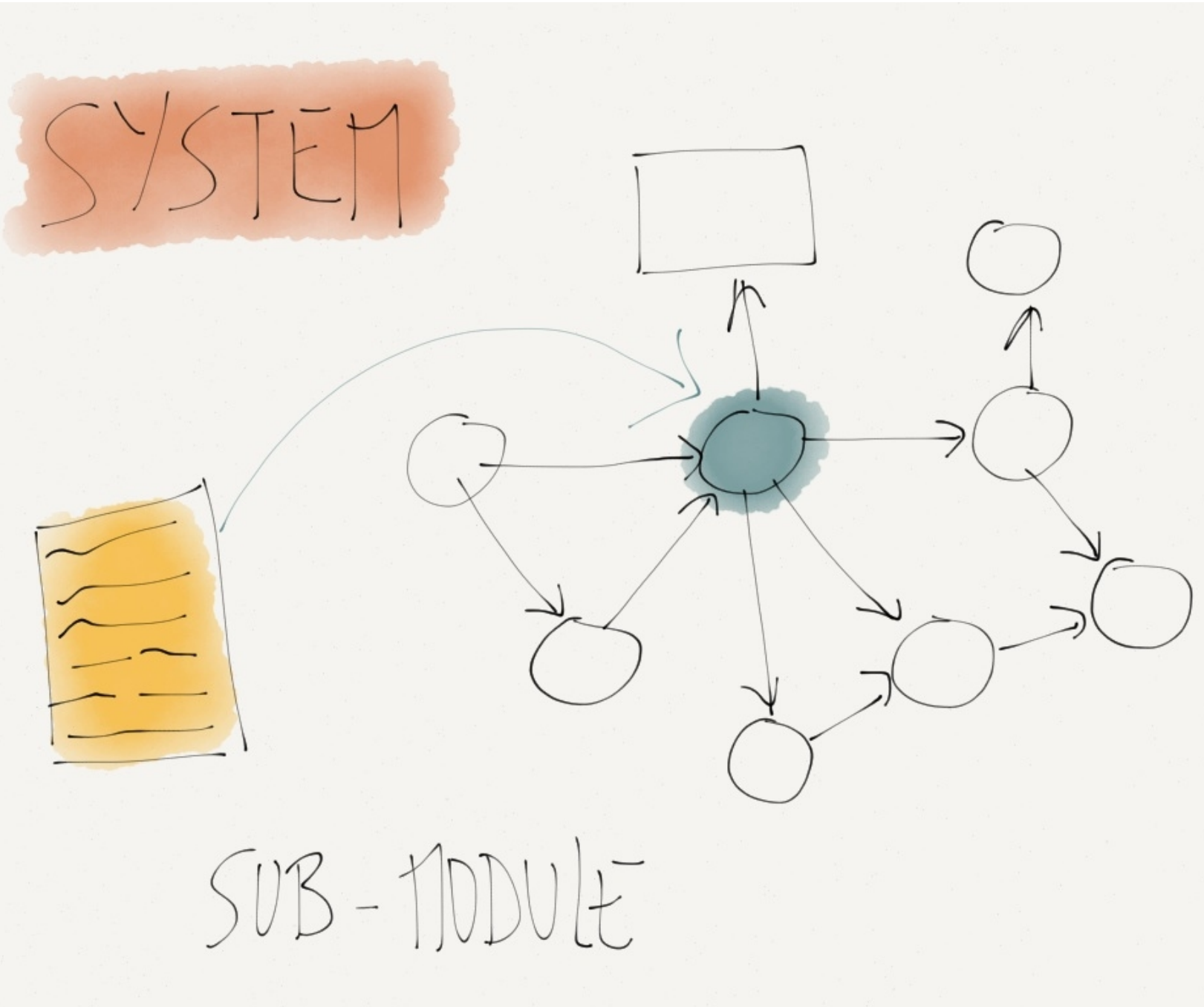
HOW

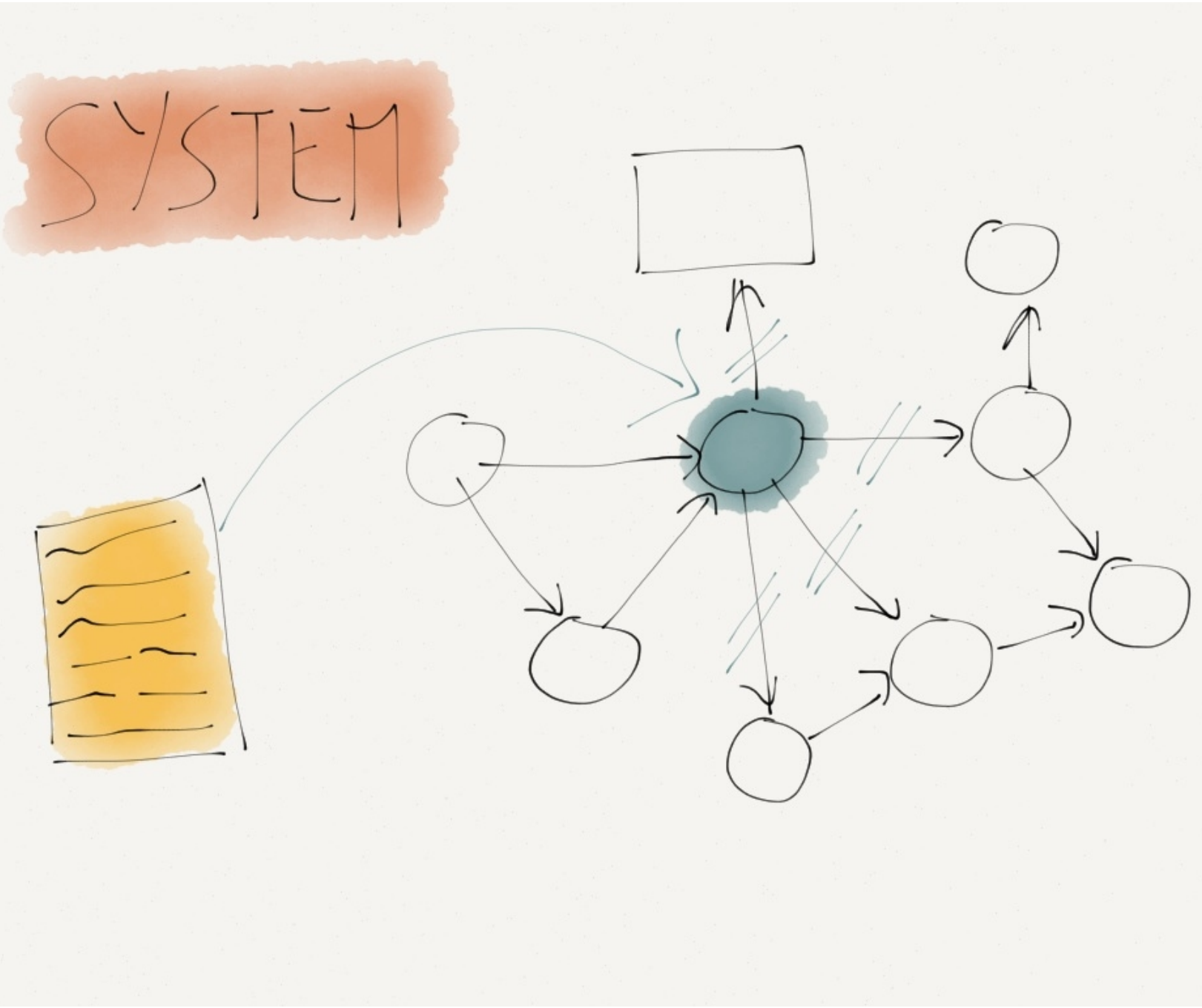
WHAT IS COVERED / TESTED
by SCENARIO

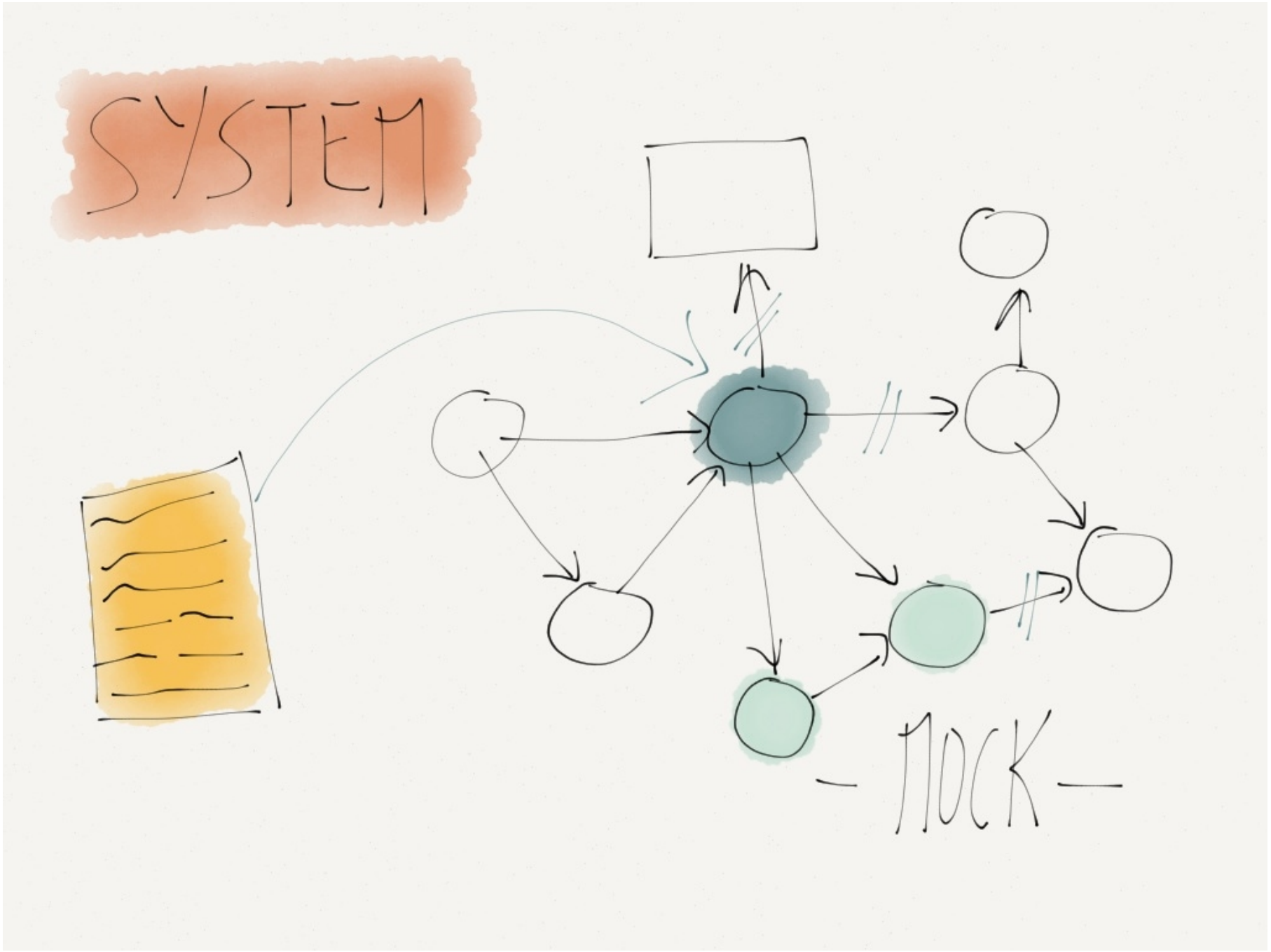
SYSTEM

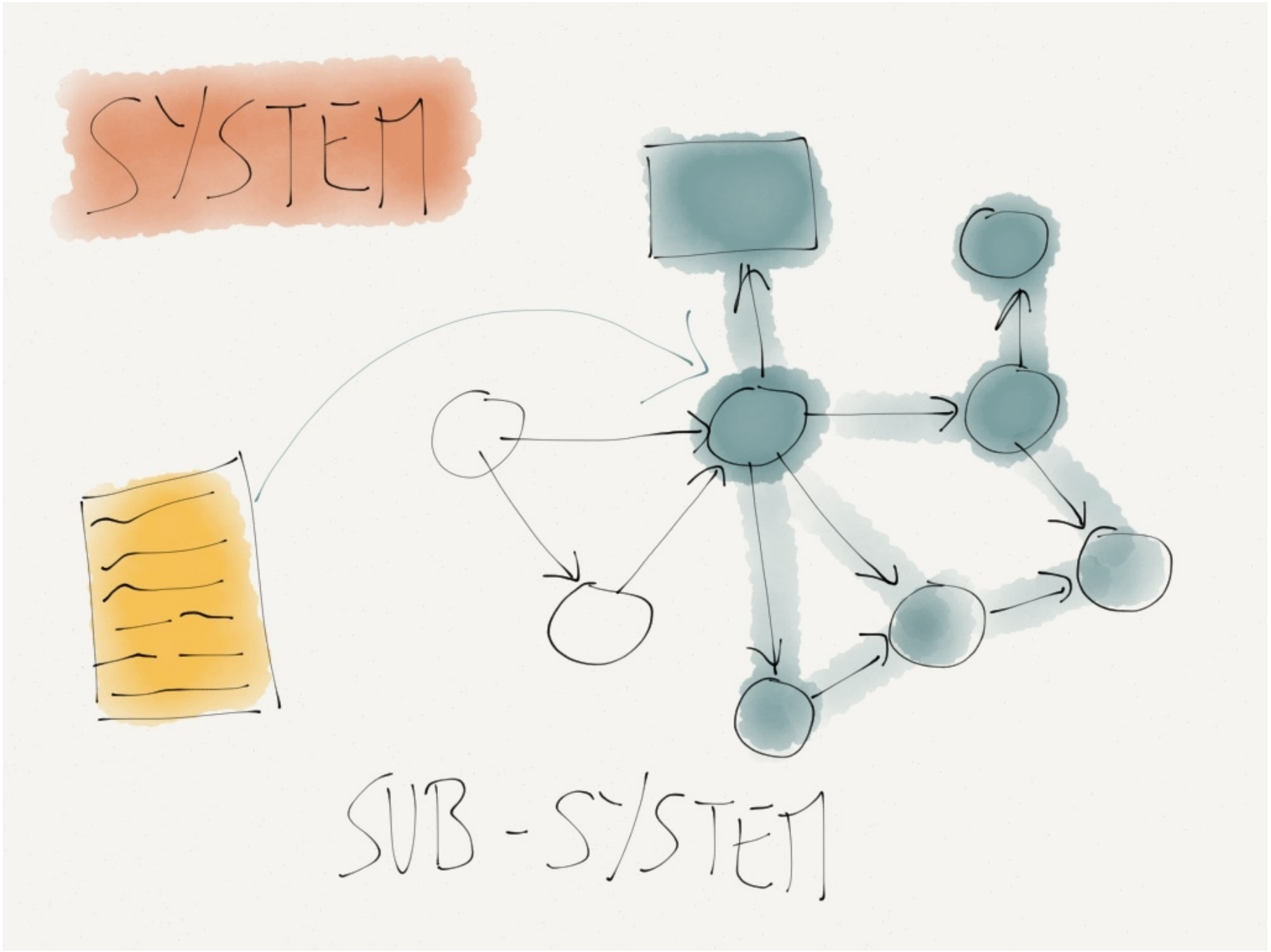



SYSTEM
END TO END

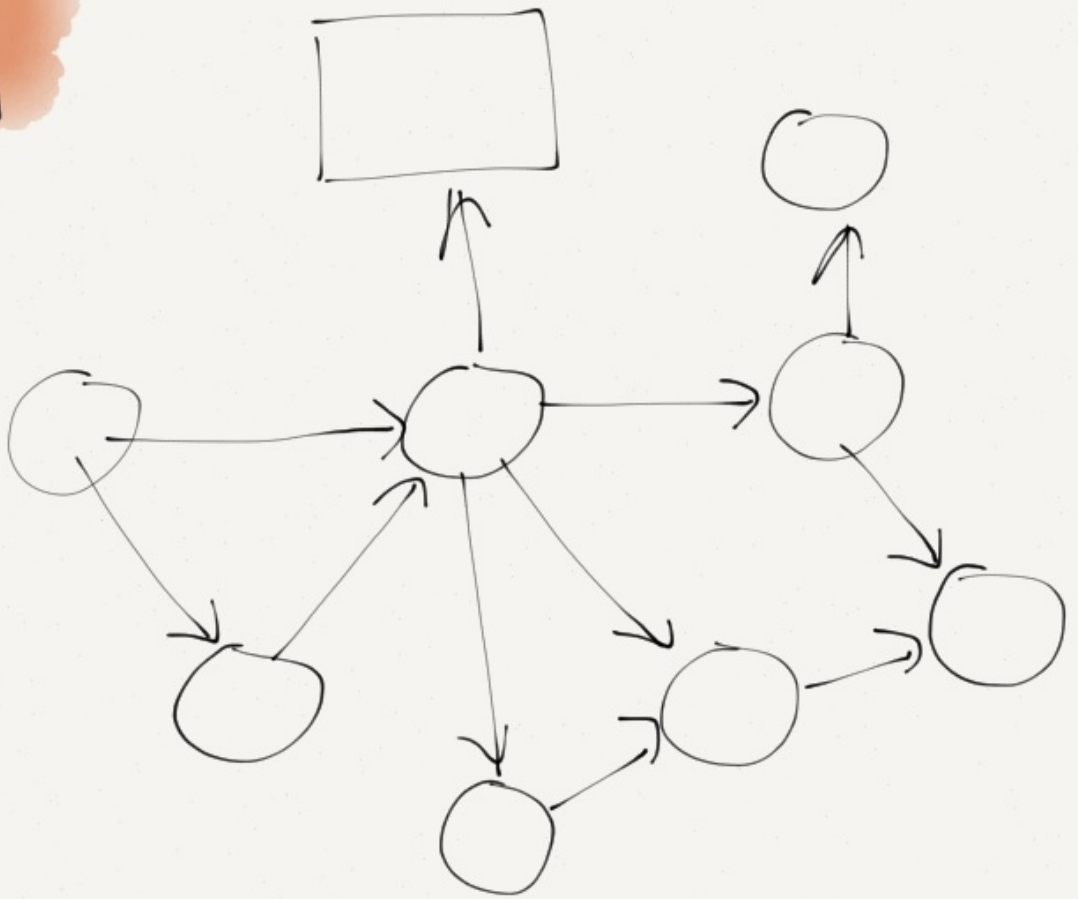

SYSTEM
SUB - MODULE

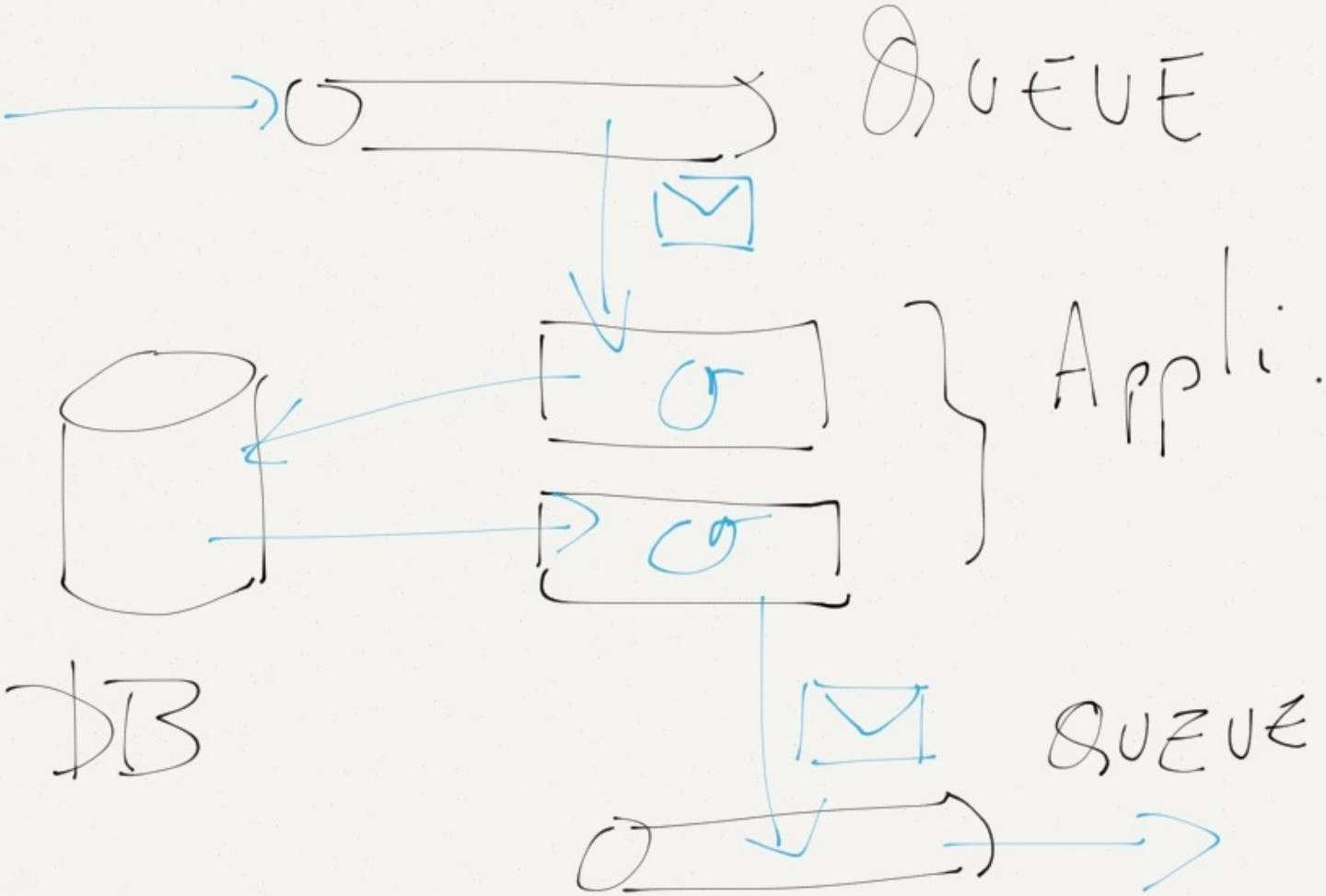

SYSTEM

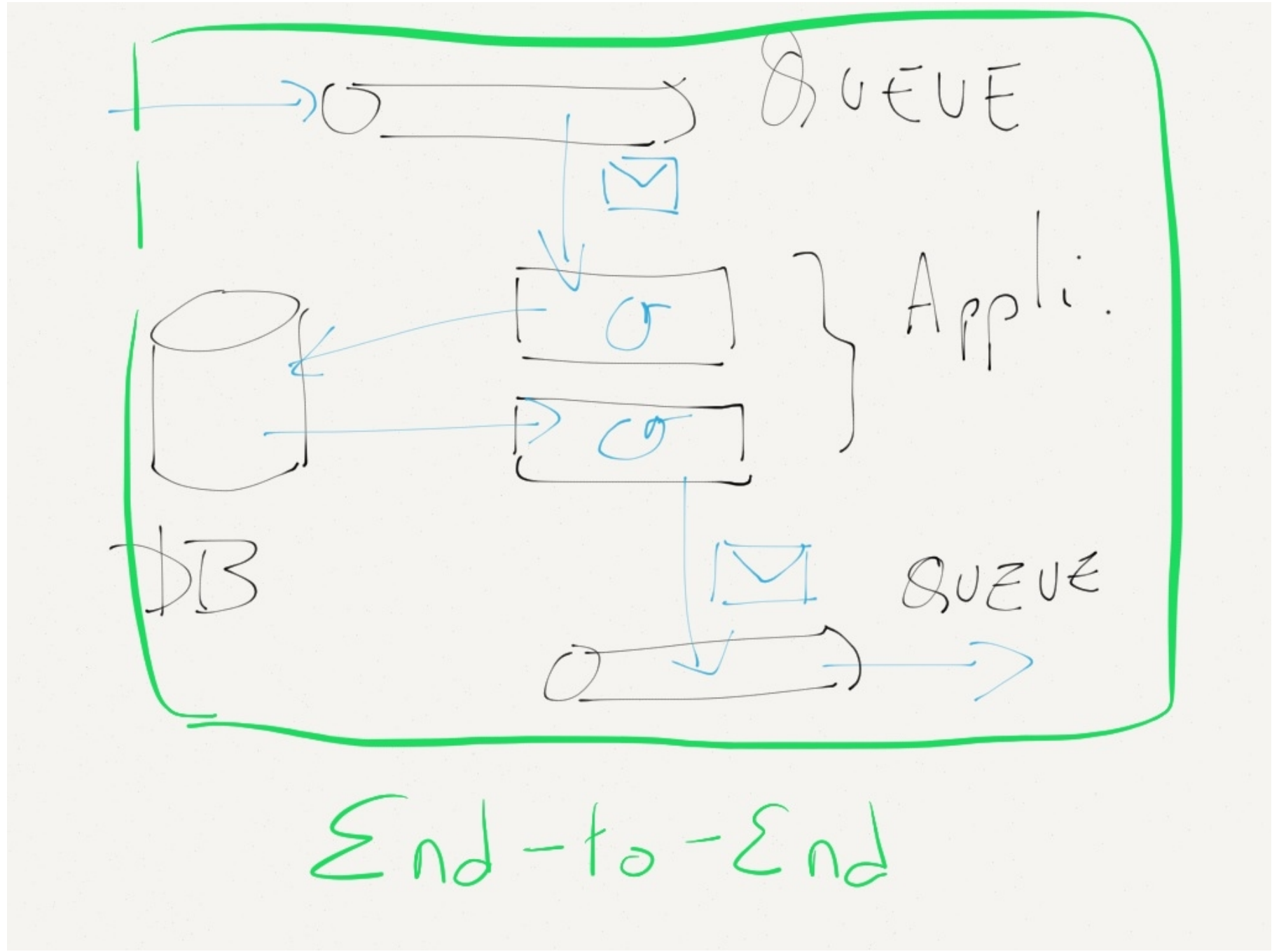

SYSTEM
- MOCK -

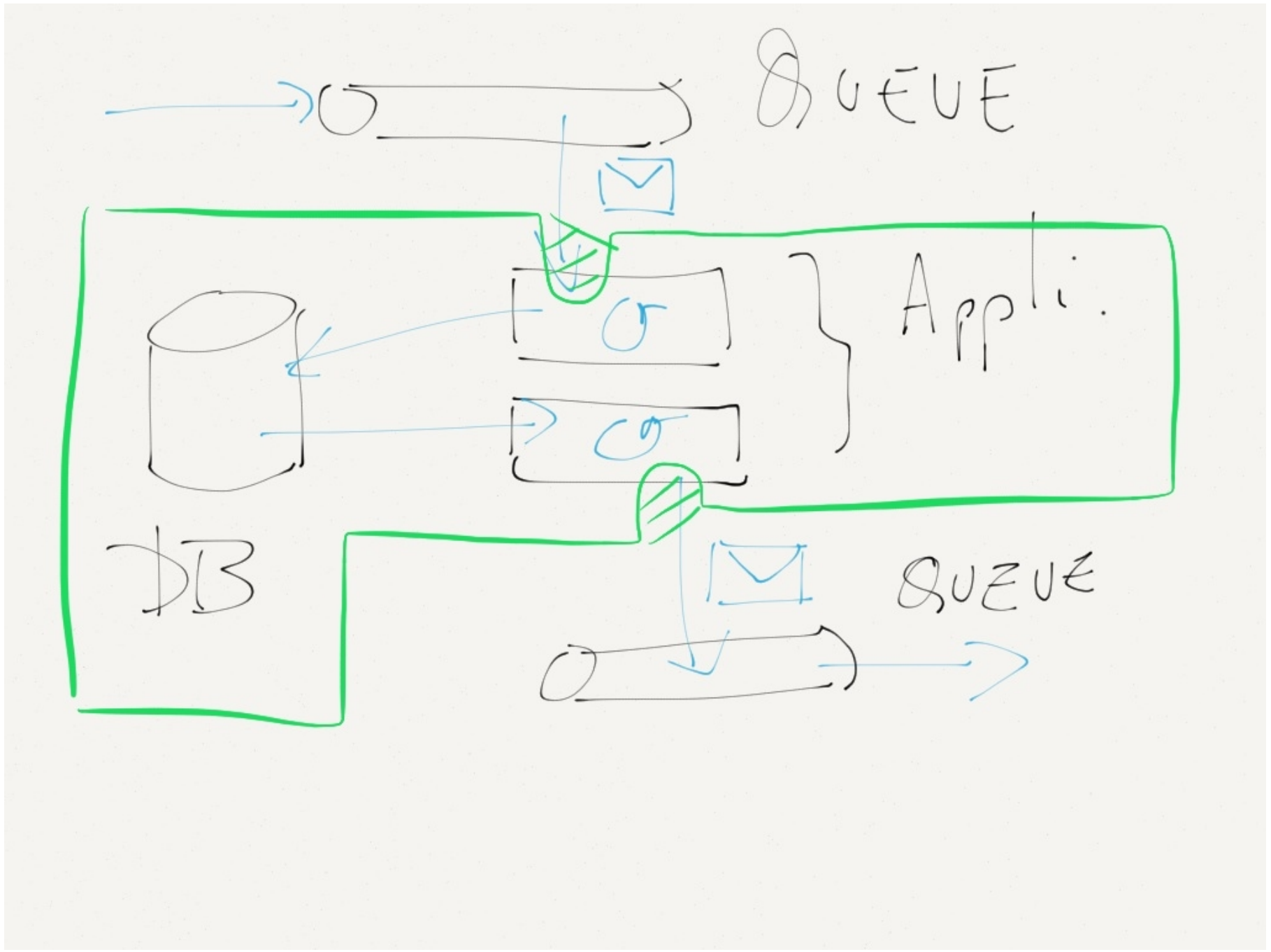

SYSTEM
SUB - SYSTEM

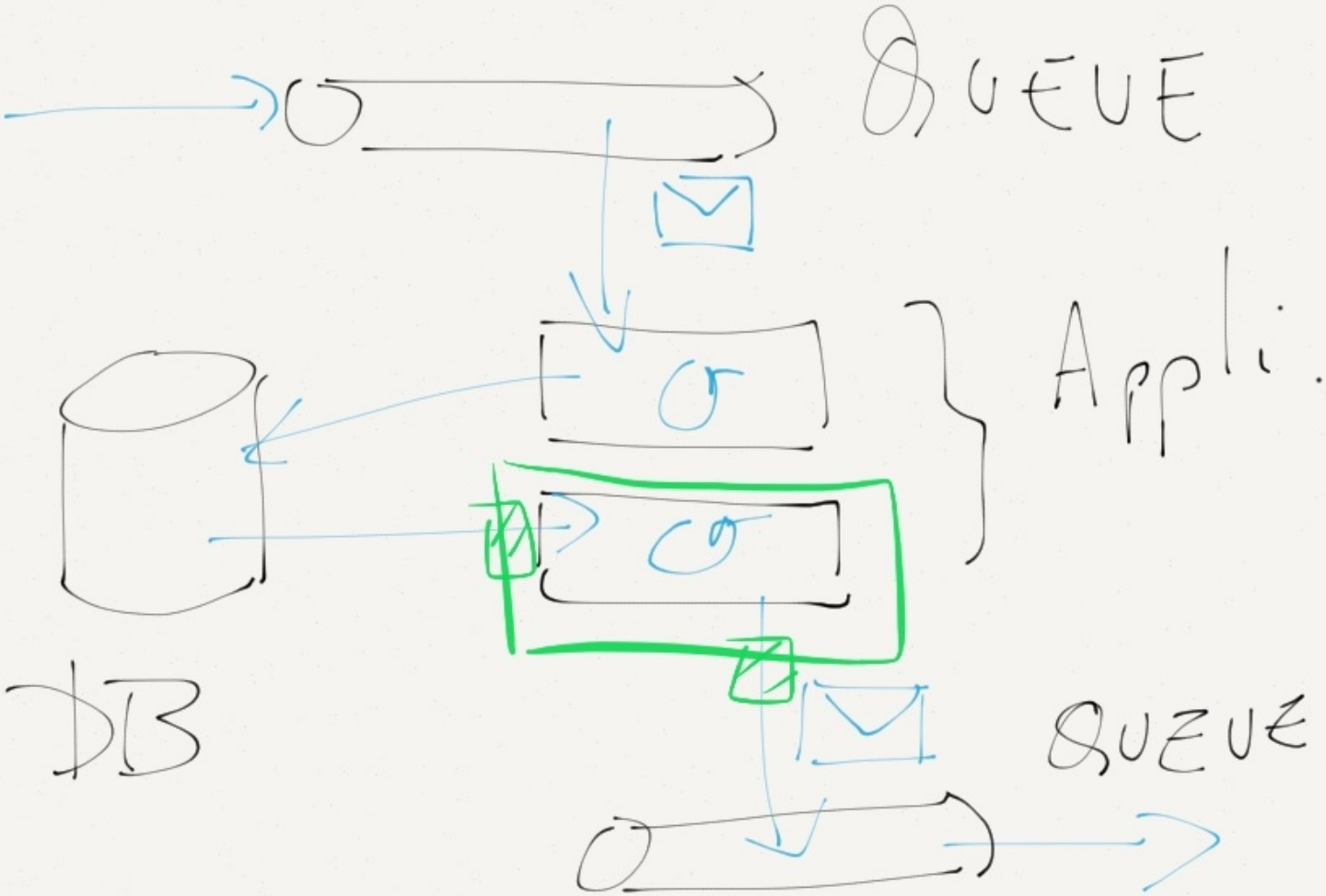
SYSTEM

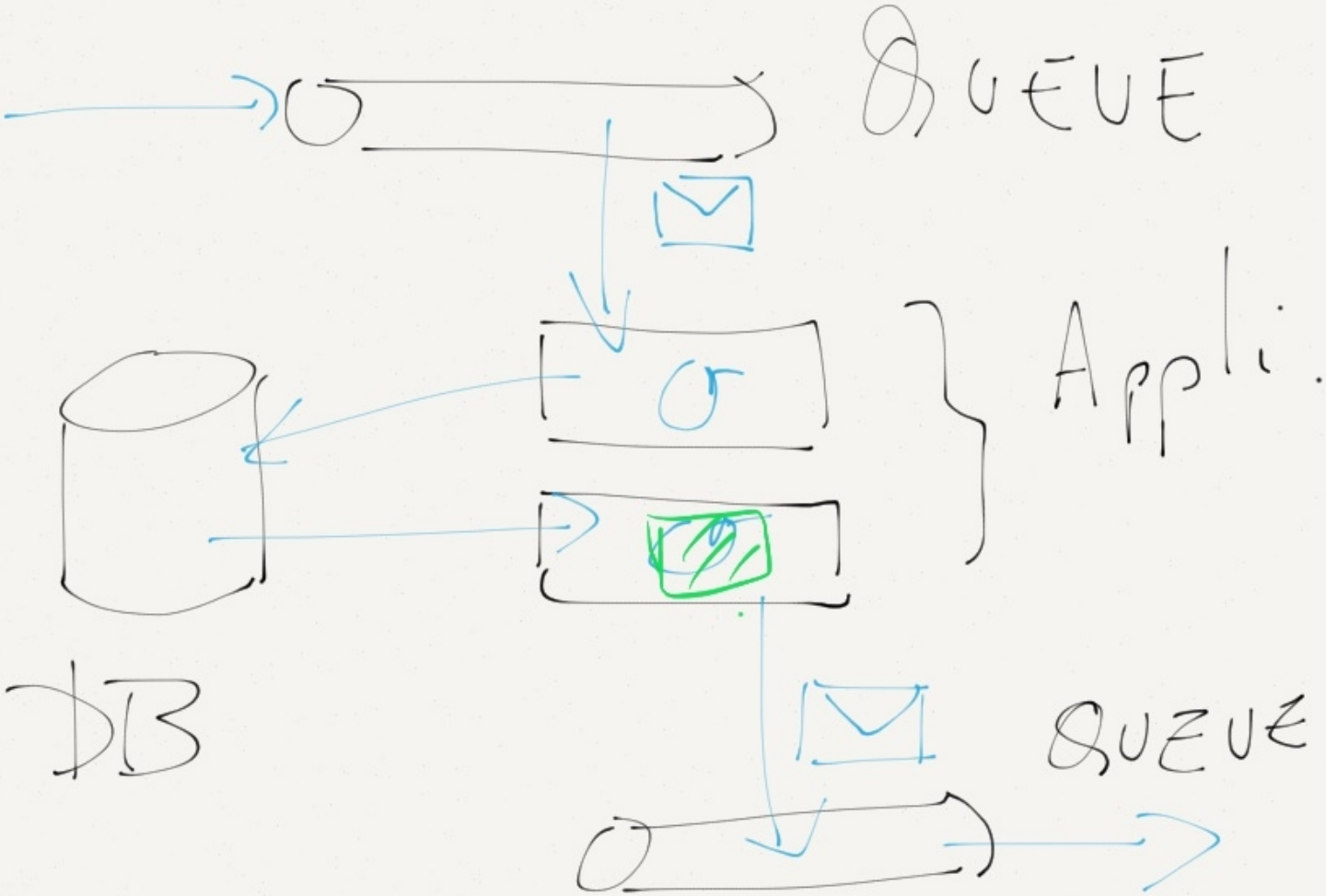


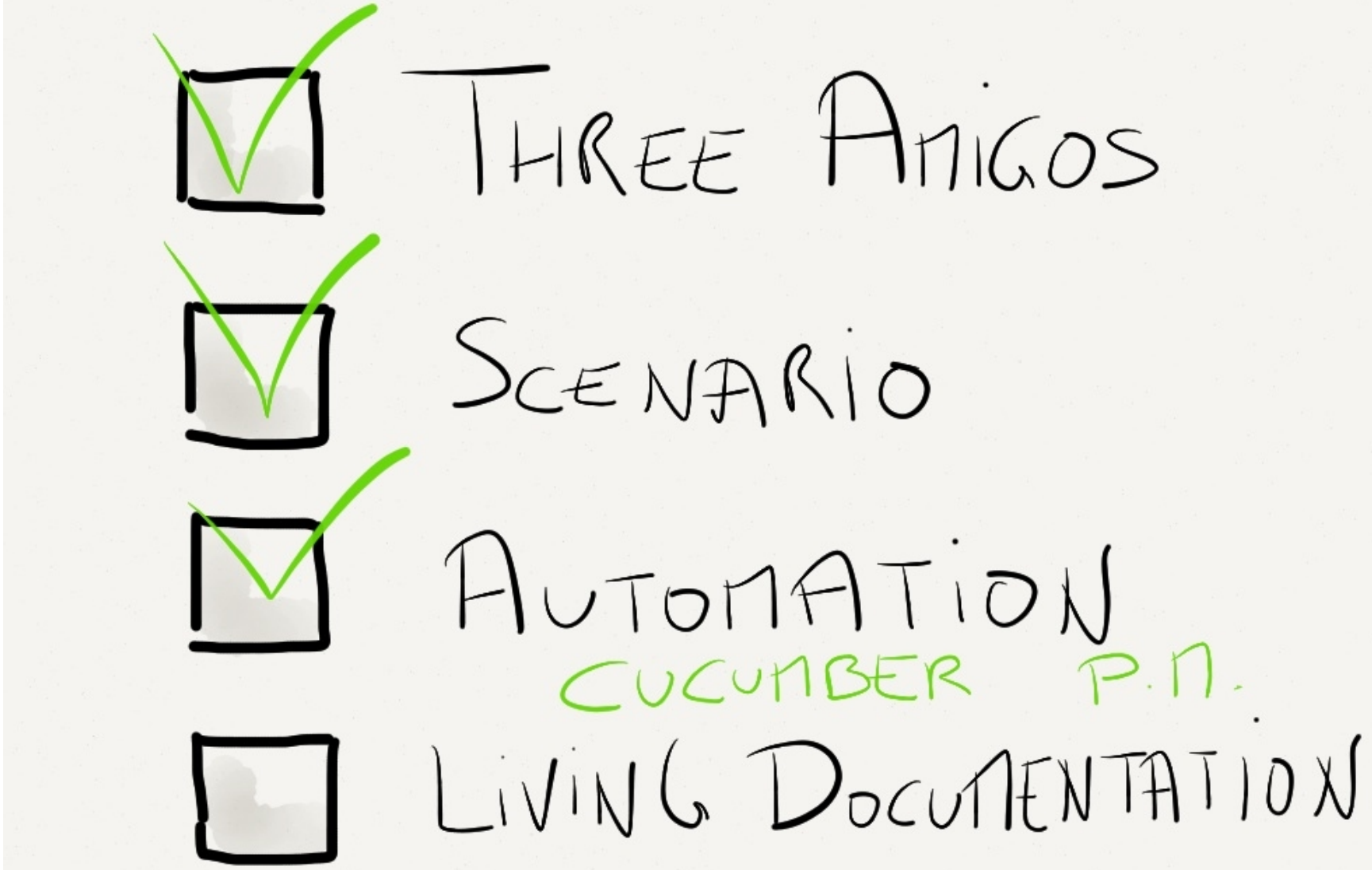

QUEUE
Appli.
DB
QUEUE

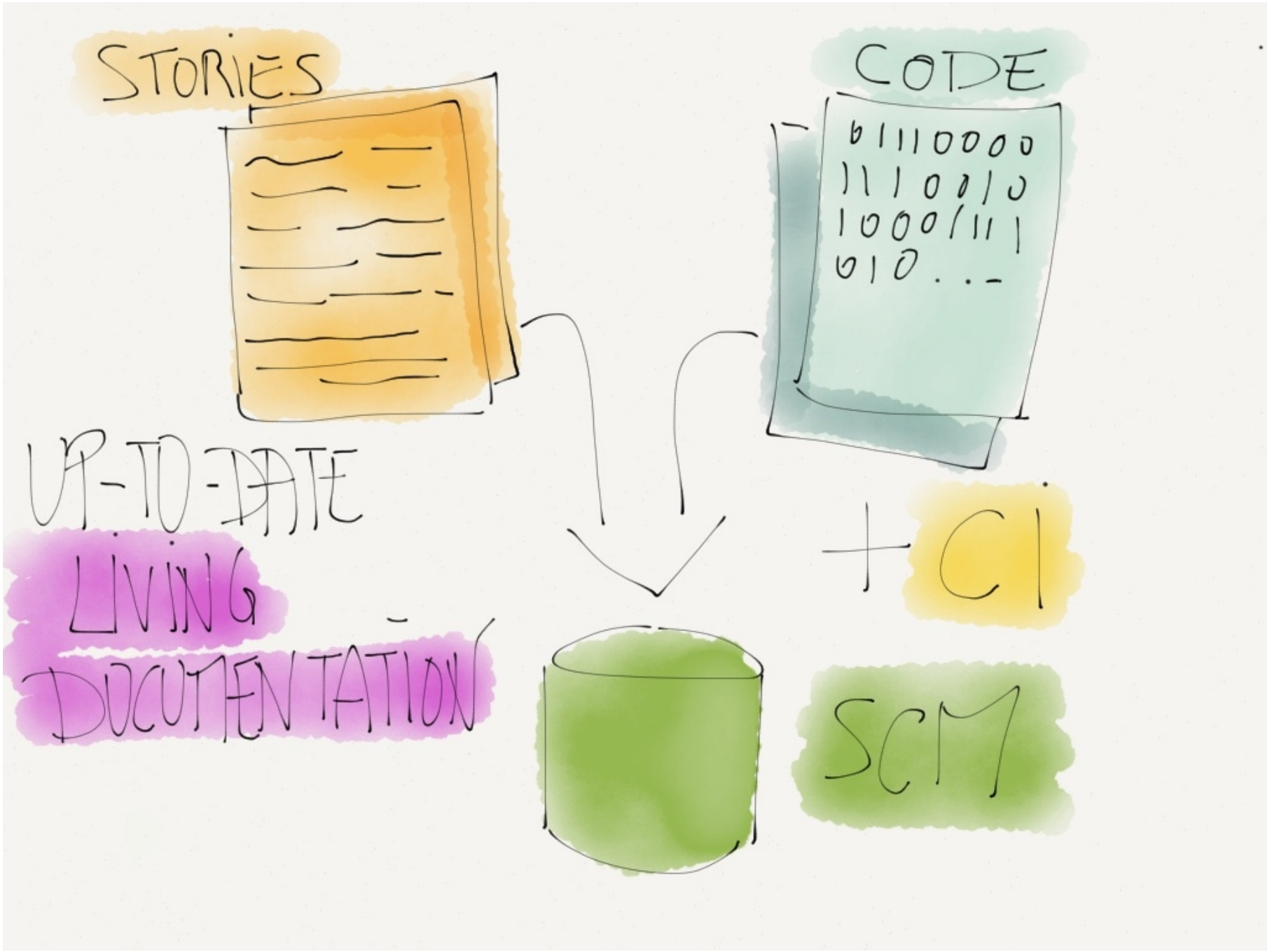

QUEUE
Appli.
DB
QUEUE
End-to-End


QUEUE
Appli.
DB
QUEUE


QUEUE
Appli.
DB
QUEUE

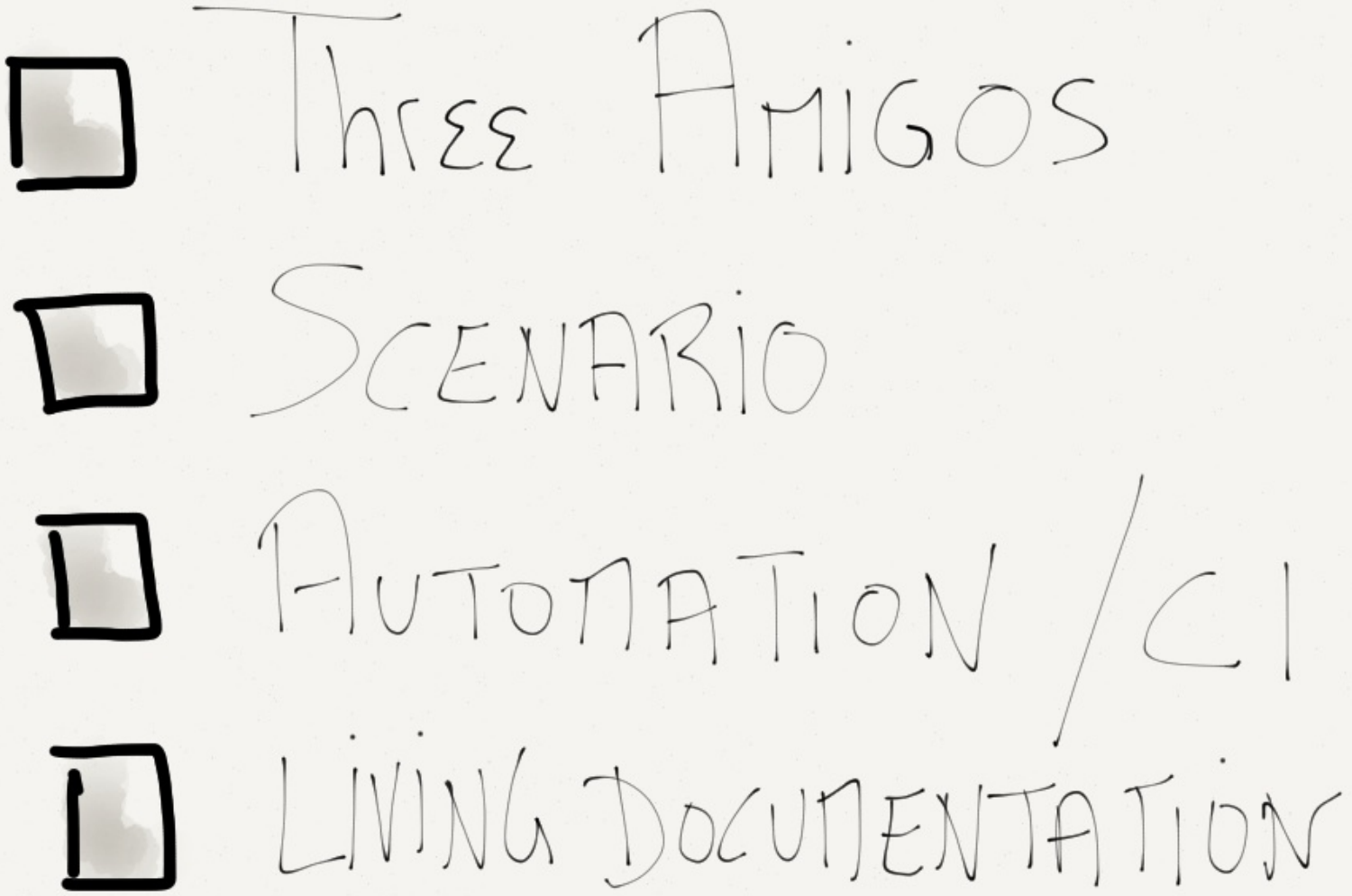

QUEUE
Appli.
DB
QUEUE


THREE AMIGOS
SCENARIO
AUTOMATION
CUCUMBER P.M.
LIVING DOCUMENTATION


STORIES
CODE
01110000
11101010
10001111
010...
UP-TO-DATE
LIVING
DOCUMENTATION
+CI
SCM

STEPS

STEP BY STEP

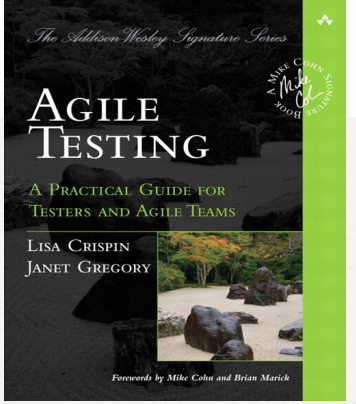

Three Amigos
Scenario
Automation / CI
Living Documentation

BOOKS

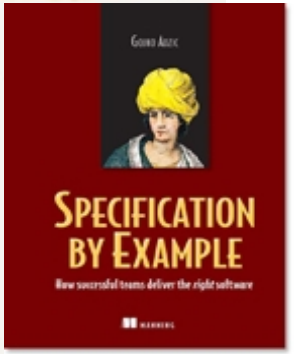
TO GO FURTHER

AGILE TESTING

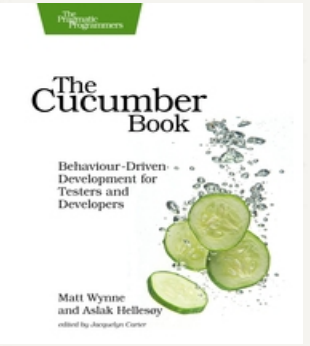
A PRACTICAL GUIDE FOR TESTERS AND AGILE TEAMS — LISA CRISPIN

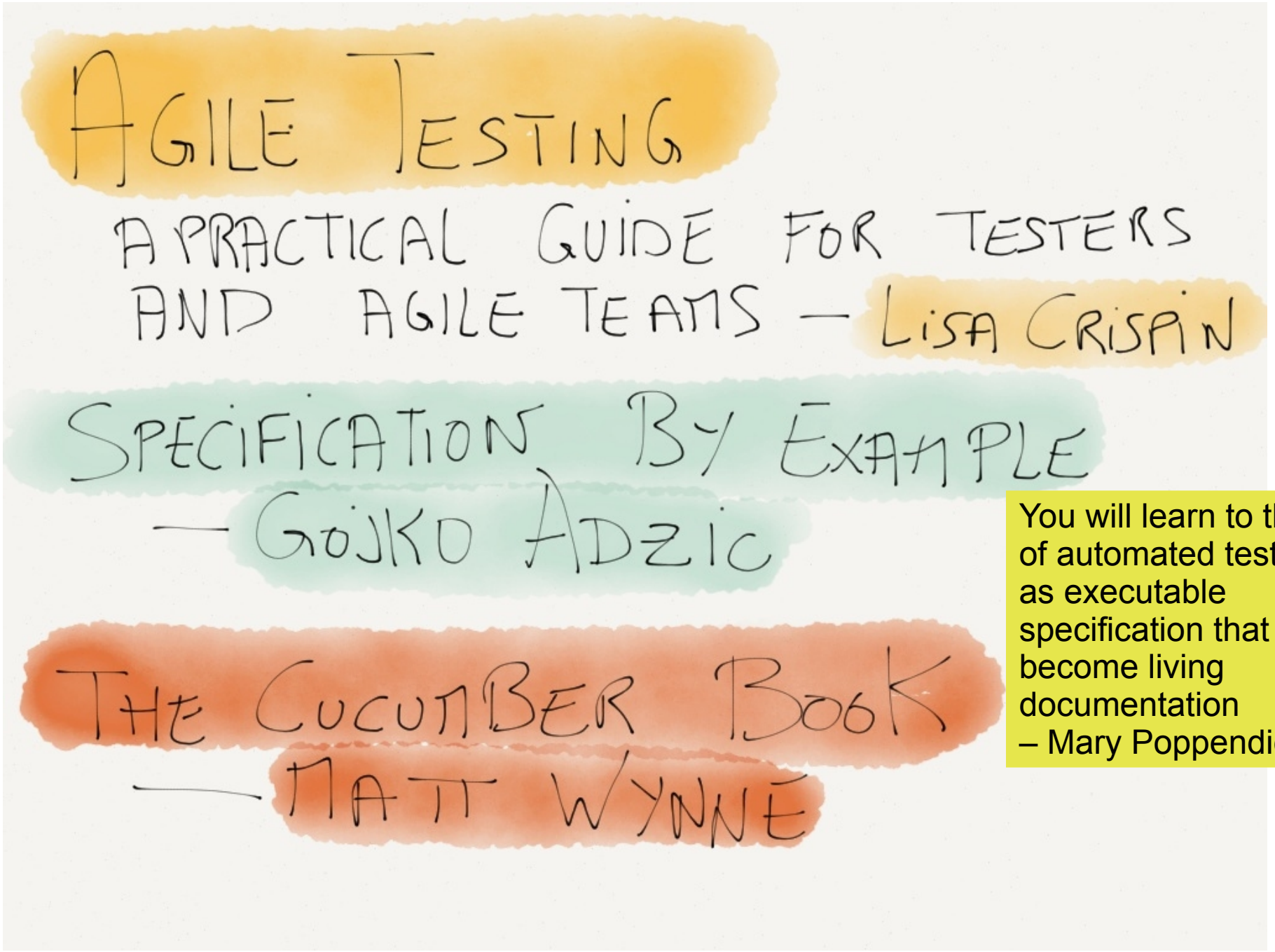


SPECIFICATION BY EXAMPLE — GOJKO ADZIC



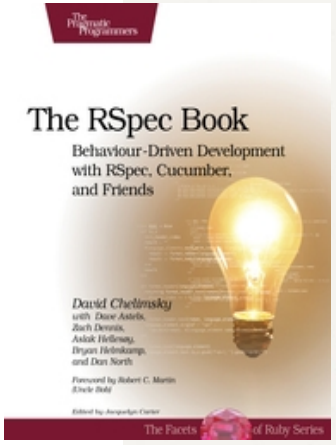
THE CUCUMBER BOOK — MATT WYNNE




AGILE TESTING
A PRACTICAL GUIDE FOR TESTERS AND AGILE TEAMS — LISA CRISPIN
SPECIFICATION BY EXAMPLE — GOJKO ADZIC
THE CUCUMBER BOOK — MATT WYNNE
You will learn to think of automated tests as executable specification that become living documentation – Mary Poppendieck

THE RSPEC BOOK : BEHAVIOUR-
-DRIVEN DEVELOPMENT WITH RSPEC,
CUCUMBER, AND FRIENDS —

D. CHELIMSKY, D. ASTELS, Z. DENNIS,
A. HELLESØY, B. HELMKAMP, DAN NORTH



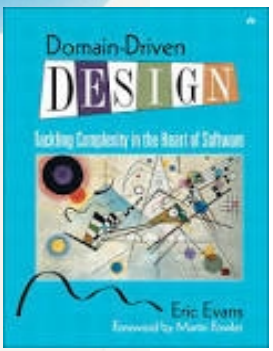
CUCUMBER

BDD'S FATHER

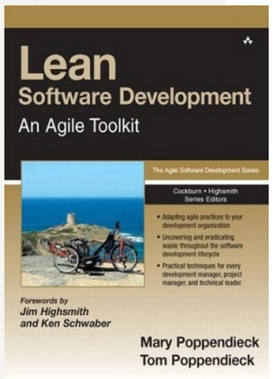
IIBA International Institute of Business Analysis

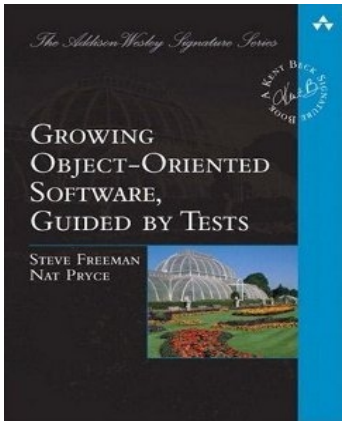
BABOK – AGILE EXTENSION

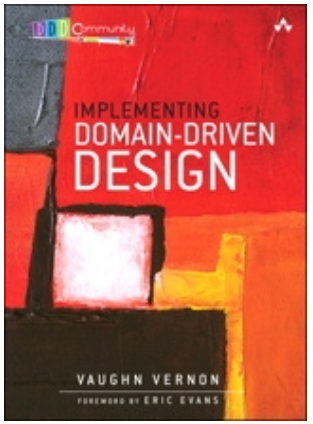
DOMAIN DRIVEN DESIGN – ERIC EVANS



LEAN SOFTWARE DEVELOPMENT:
AN AGILE TOOLKIT
– MARY & TOM POPPENDIECK




The Addison Wesley Signature Series
Kent Beck's Test-Driven Development Signature Book
Growing Object-Oriented Software, Guided by Tests
Steve Freeman
Nat Pryce


DDD Community
IMPLEMENTING
DOMAIN-DRIVEN
DESIGN
VAUGHN VERNON
FOREWORD BY ERIC EVANS

THE POINT

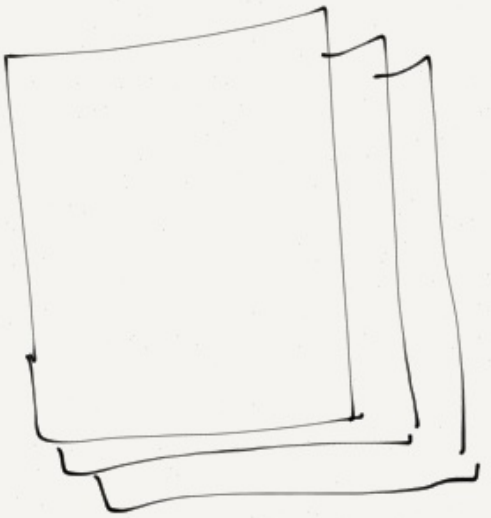
IF ONE THING MUST REMAIN

BDD = SHARED UNDERSTANDING
by DISCUSSING EXAMPLES

BDD = SHARED UNDERSTANDING

by DISCUSSING EXAMPLES

SCENARIO



BDD = SHARED UNDERSTANDING

by DISCUSSING EXAMPLES

SCENARIO

THREE AMIGOS