Top Legacy Sins

Eberhard Wolff
Fellow
@ewolff

Eberhard Wolff

Microservices

Grundlagen flexibler Softwarearchitekturen

dpunkt.verlag

http://microservices-buch.de/

Microservices

Flexible Software Architectures

Eberhard Wolff

http://microservices-book.com/

What should a Software Architect do?

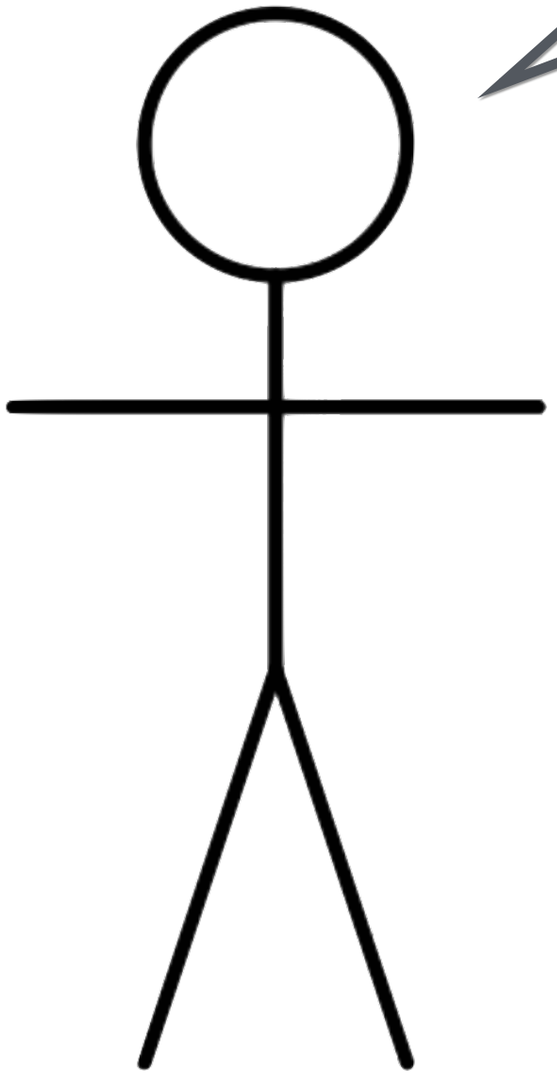
Create a Proper Architecture.

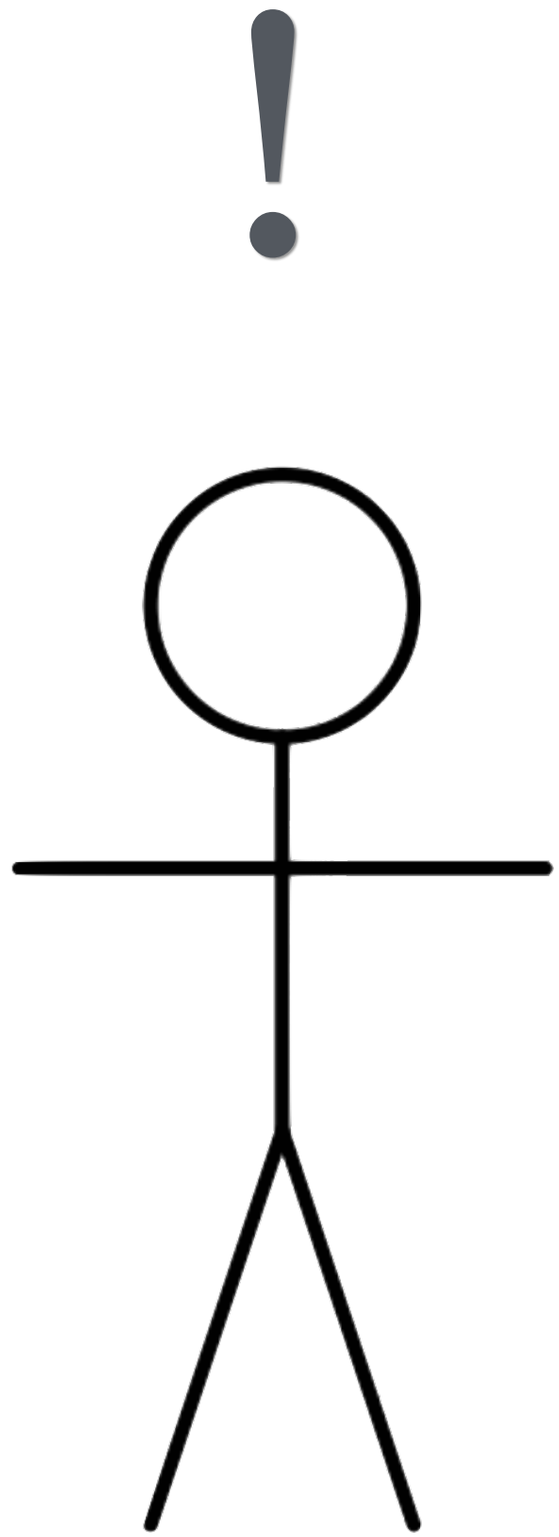
Create a Proper Architecture.

Agree??

I'm a Software Architect.

But I'm not doing architecture.





How important is Software Architecture?

Why would we care about Software Architecture?

Software Architecture =Structure

Architecture Goals: Quality

- All kinds of quality
- Performance
- Security
- ...

- Focus of this presentation: Changeability
- Architects must understand customers & priority

Architects must understand customer

Software Architecture

Set of

structures

comprise

software elements,

relations among them, and

properties of both.

Why?

- Can work on modules in isolation
- Can work on collaboration of modules
- Fits in my head

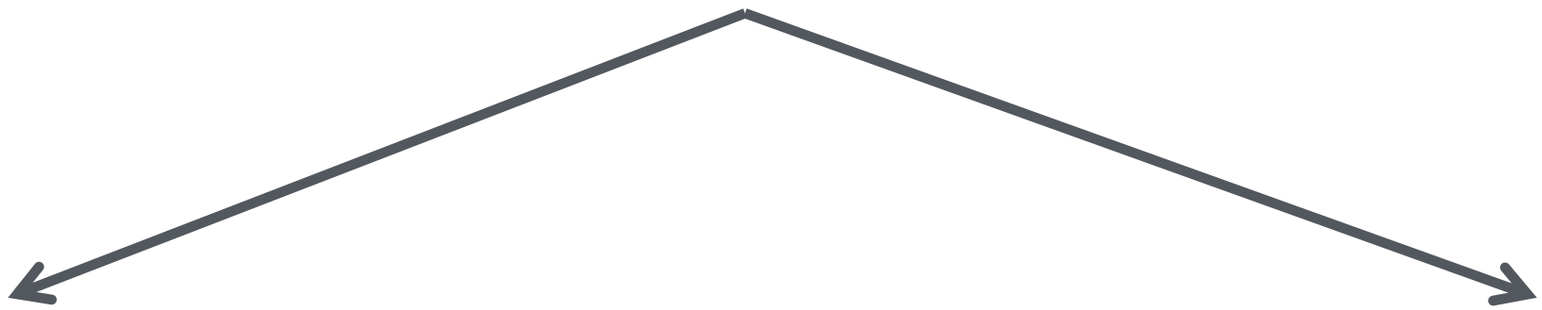
Dan North

Does He Care?

Why?

Great
Architecture
↓
Software

Easy to change



High
productivity

Low
Cost

He Cares

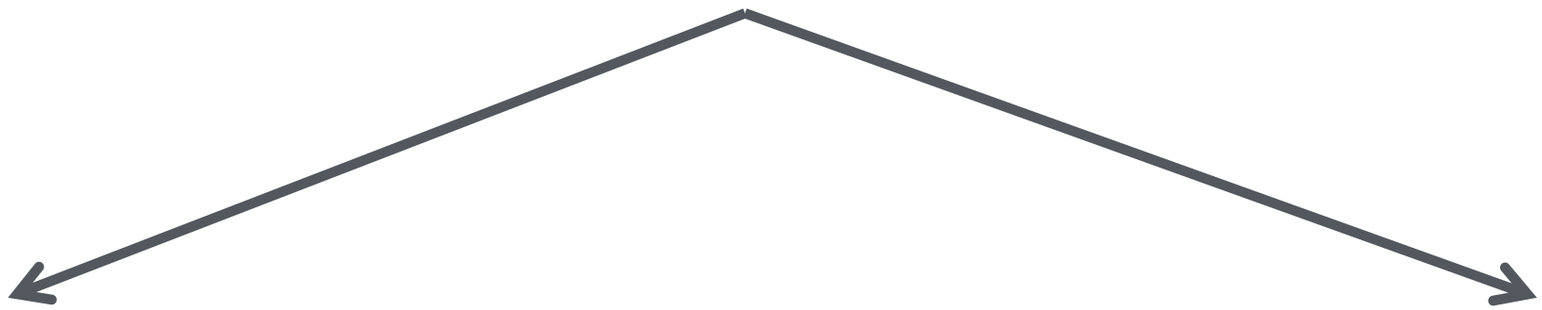
Why?

Great Architecture

↓

Software

Easy to change



High productivity

Low Cost

What He Cares About

Great Architecture

Software Easy to change

High productivity

Low Cost

Is architecture the only factor for changeability?

What is the best thing we can have in a legacy code?

Great architecture

or

automated tests?

No Automated Tests

- No way to find bugs
- Changes almost impossible

- Legacy code = code without tests
- Michael Feathers
  Working Effectively with Legacy Code

Let’s add some automated UI tests

Automated UI Tests

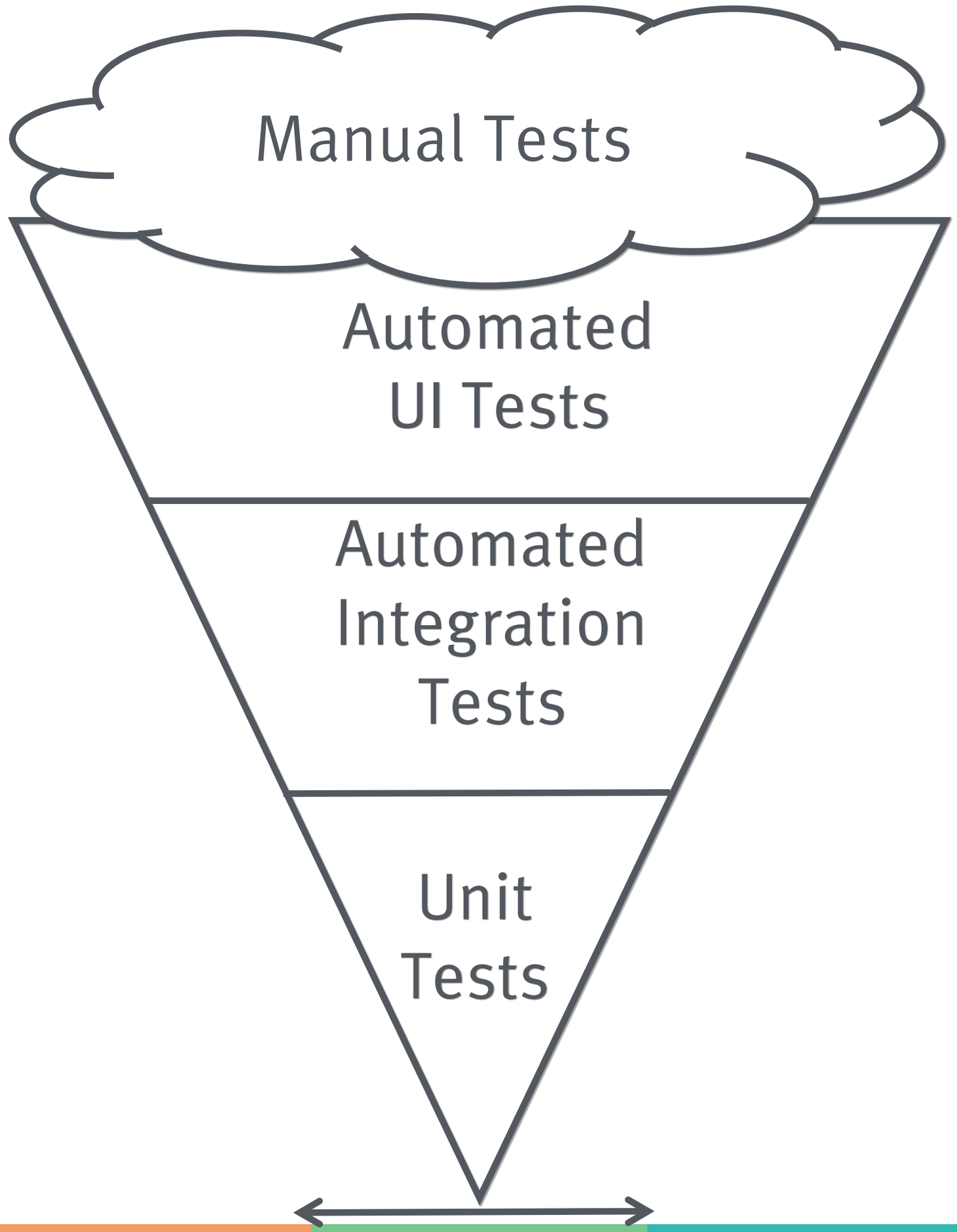
- Easy to implement
- Exactly what testers do manually
- Easy to understand for customers
- Test business processes
- Safety net

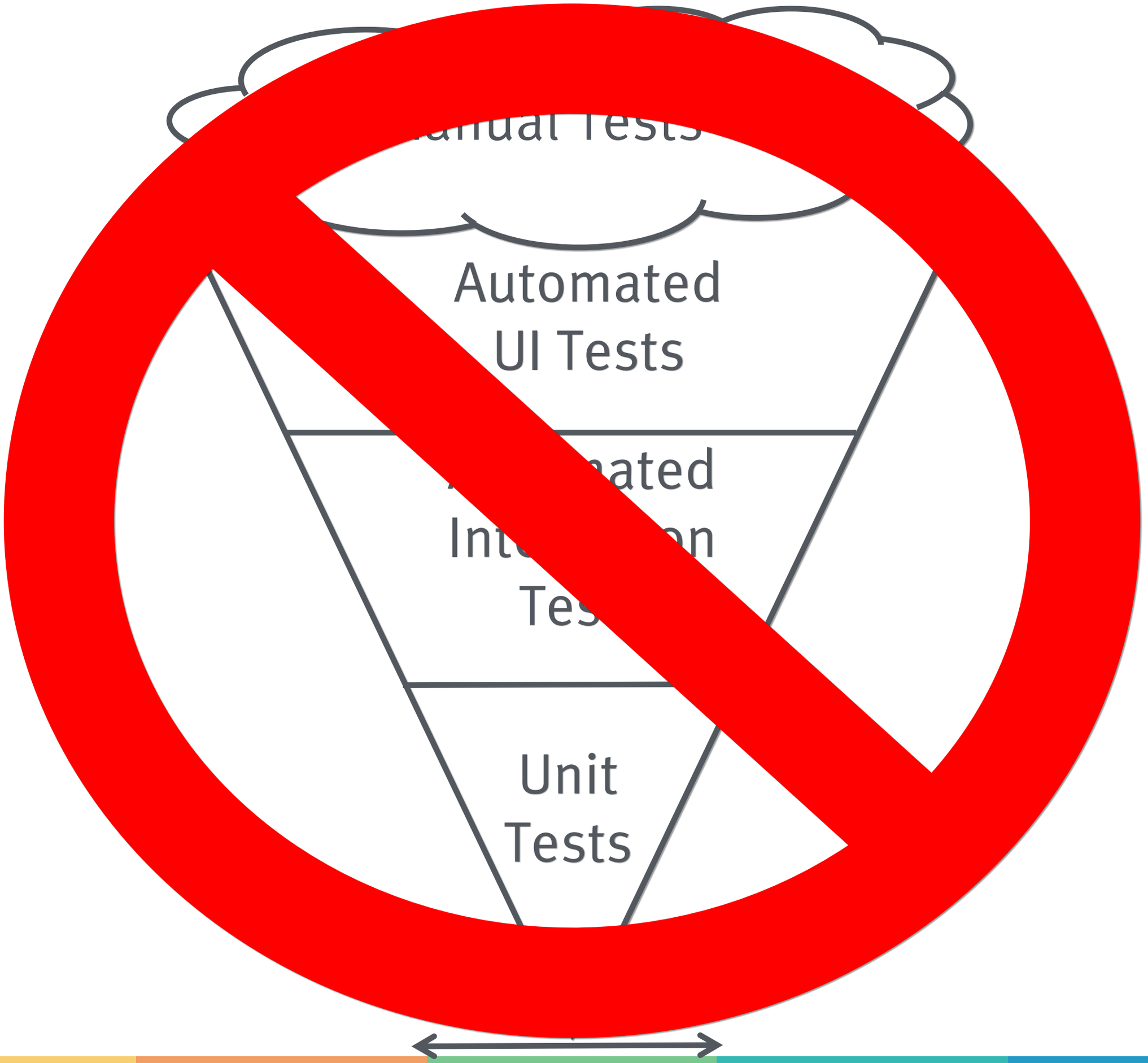
Unit Tests

or automated UI tests?

Many UI Tests Worse

- Fragile: Changes to UI break test
- Business meaning of tests easily lost
- Takes long
- Often not reproducible


Manual Tests
Automated
UI Tests
Automated
Integration
Tests
Unit
Tests


Automated
UI Tests
Unit
Tests

Slow

Unreliable

Expensive

Alternatives to automated UI tests?

Textueller Akzeptanztest

Szenario

- Möglicher Ablauf in einer Story

- Standardisierte Bestandteile:
  - Gegeben… (Kontext)
  - Wenn… (Ereignis)
  - Dann… (erwartetes Ergebnis)

Szenario: Beispiel

Szenario: Kunde registriert sich erfolgreich

Gegeben ein neuer Kunde mit EMail eberhard.wolff@gmail.com Vorname Eberhard Name Wolff

Kontext

Wenn der Kunde sich registriert

Ereignis

Dann sollte ein Kunde mit der EMail eberhard.wolff@gmail.com existieren

Erwartetes Ergebnis

Und es sollte kein Fehler gemeldet werden

Erwartetes Ergebnis

```
RegistrationService registrationService;
// Initialisierung RegistrationService
// ausgelassen

private User kunde;
private boolean fehler = false;

@Given("ein neuer Kunde mit "+
 "EMail $email Vorname $vorname"+
 " Name $name")
public void gegebenKunde(String email,
 String vorname, String name) {
  kunde = new User(vorname, name,
   email);
}
```

```
@When("der Kunde sich registriert")
public void registerKunde() {
  try {
    registrationService.register(kunde);
  } catch (IllegalArgumentException ex) {
    fehler = true;
  }
}

  @Then("sollte ein Kunde mit der EMail $email existieren")
  public void existiert(String email) {
    assertNotNull(registrationService.getByEMail(email));
  }
@Then("es sollte kein Fehler gemeldet werden")
  public void keinFehler() {
    assertFalse(fehler);
  }}
```

Test is about handling risk

Tests for Risks

- Algorithm / calculation wrong:
  Unit test
- System failures:
  Unit tests
- Wiring / collaboration:
  Integration tests
- Business process:
  Integration test
- UI: UI test

Example: User Registration

- Unit test

Validations

Database failure

- Integration test

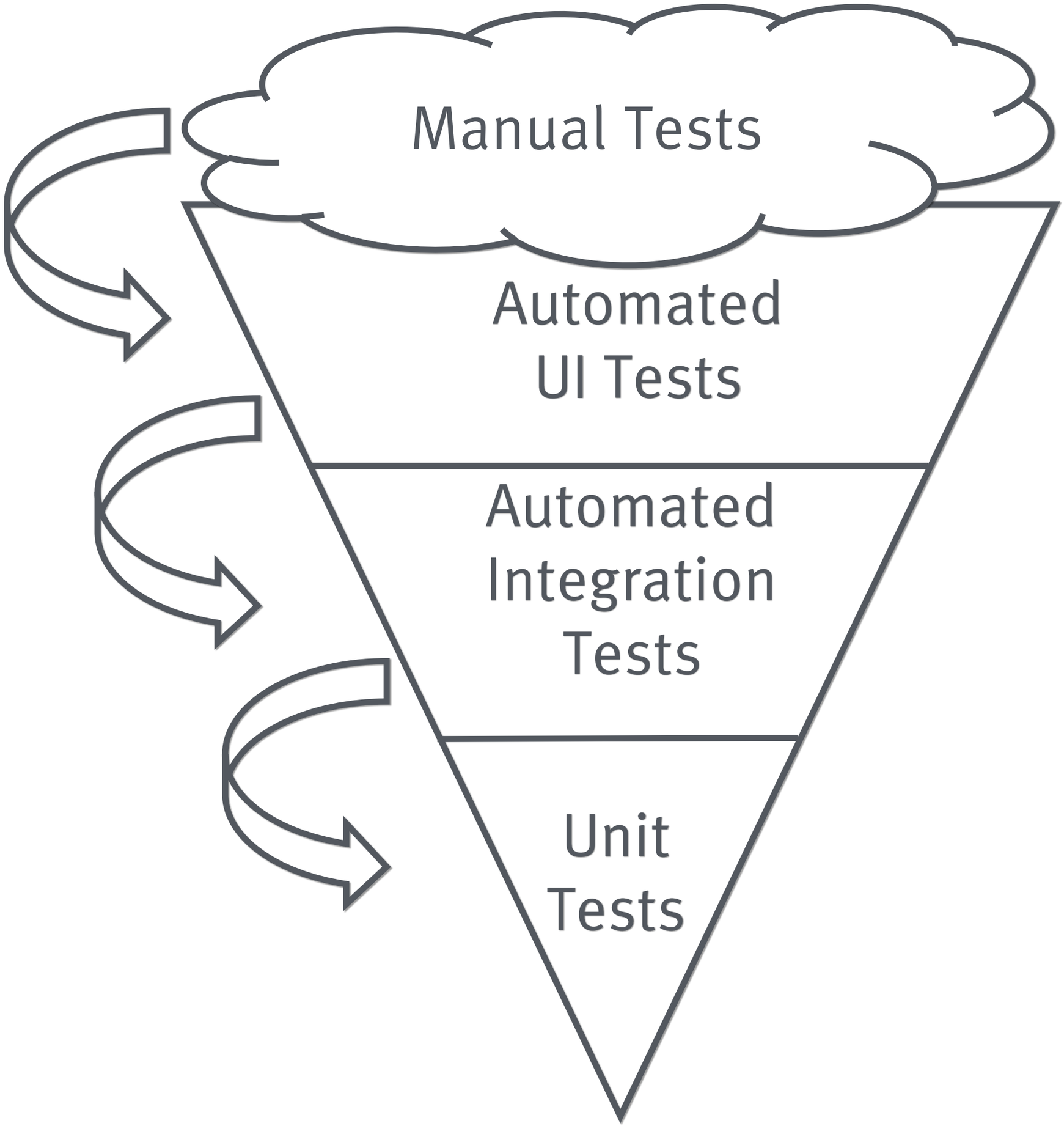
Process

- UI test

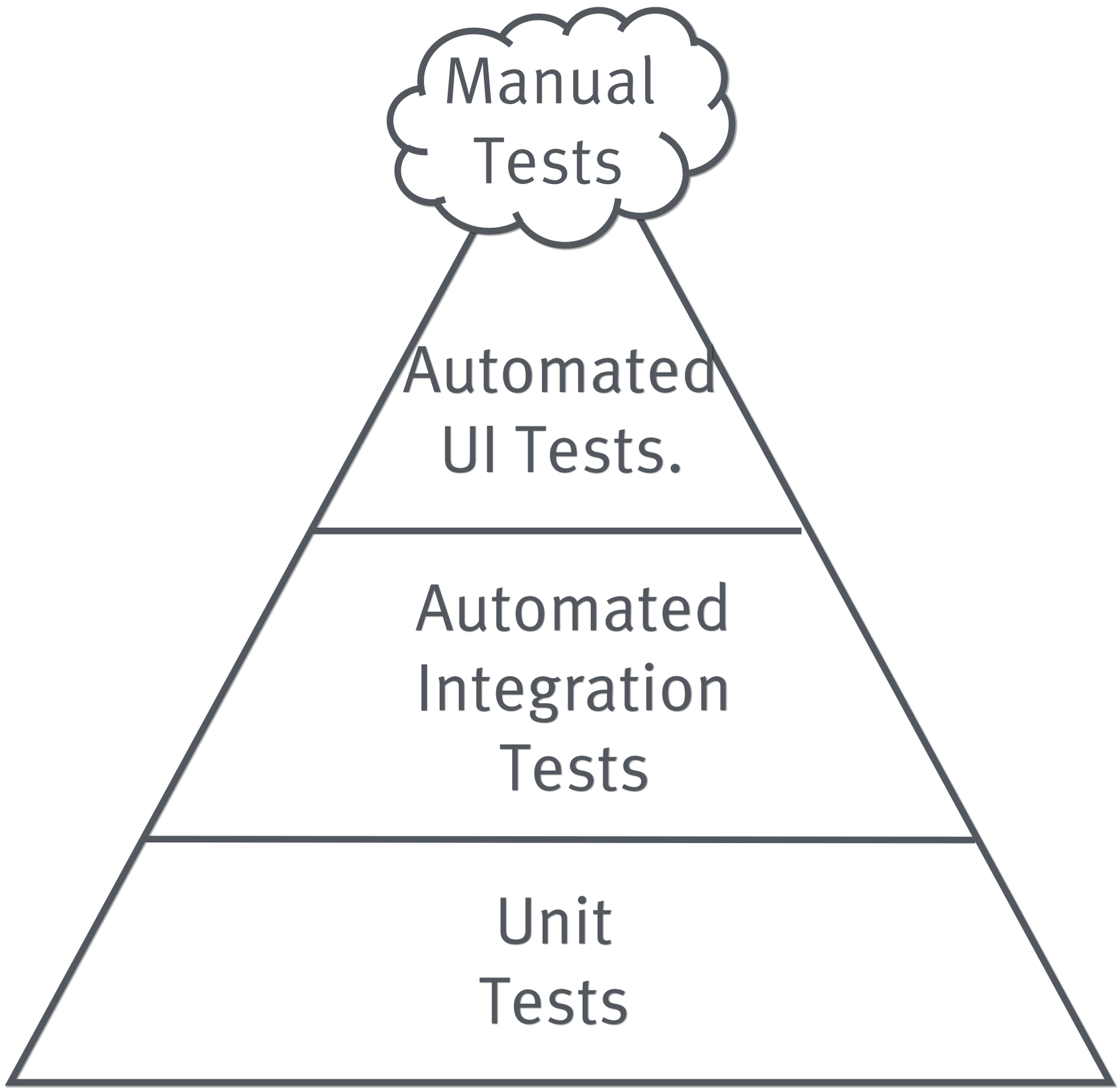
Everything shown?

Not Tested

- UI won't test validation
- ...or algorithms
- ...or the process
- Risks handled elsewhere


Manual Tests
Automated
UI Tests
Automated
Integration
Tests
Unit
Tests

Test Pyramid



Test Pyramid instead of Automated UI Tests

Great architecture

or fast & easy deployment?

Deployment

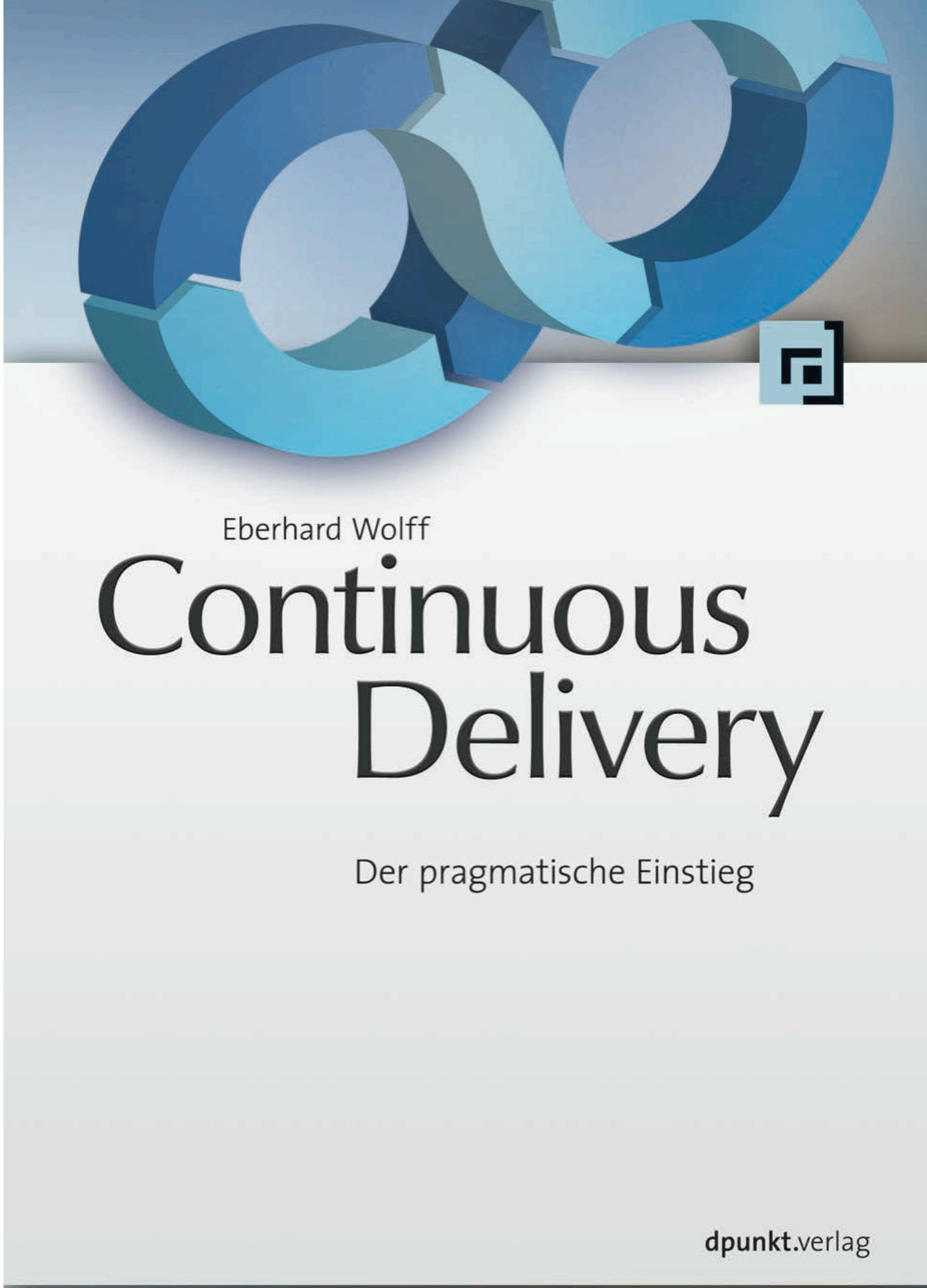
- Manual deployment is error prone
- Slow deployment

- Lots of code developed but not deployed
- i.e. more lean waste

- Slow feedback
- Slow time to recovery

Leseprobe:

http://bit.ly/CD-Buch



Continuous Delivery: Build Pipeline

Continuous Delivery

- Testing
- + automated deployment

- Testing: reduce probability of errors
- Automated deployment: better mean time to repair

Continuous Delivery

- Make software easier to change
- & deploy
- Reliable and reproducible tests
- Automated deployed
- Fast & reliable

What is a great architecture?

UI

Logic

Database

Is Three Tier a great architecture?

Three Tier

- Actually layer: no distribution
- By technology
- Layers can be replaced
- Layers can be developed independently

Do you replace e.g. the persistence layer?

Is it really simple to add e.g. mobile clients?

A better reason:

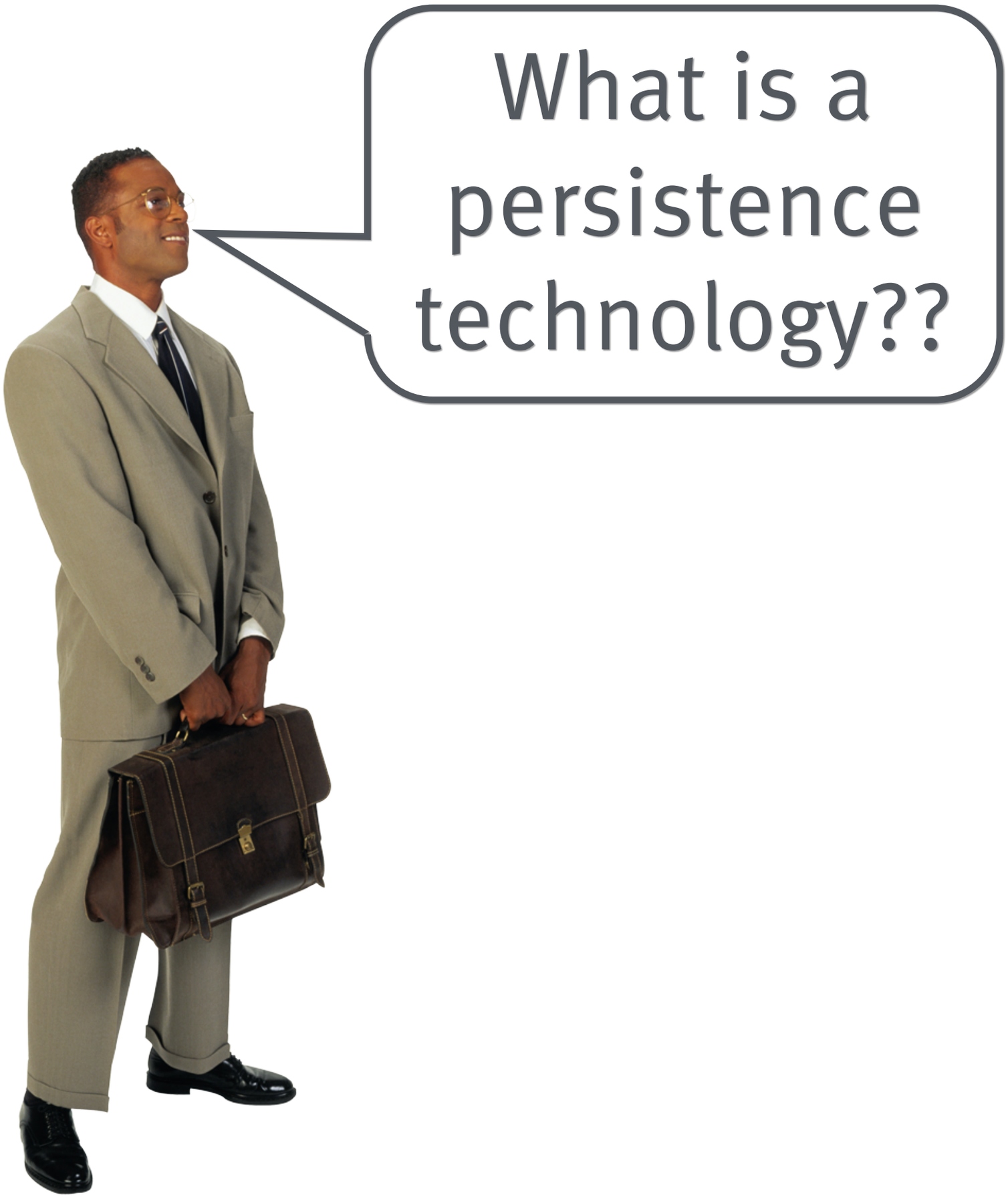
Fits in my head.


Redo the order processing!

Add feature to the registration!

Can we change the persistence technology instead?

Please?


What is a persistence technology??

UI

Logic

Registration

Order

Billing

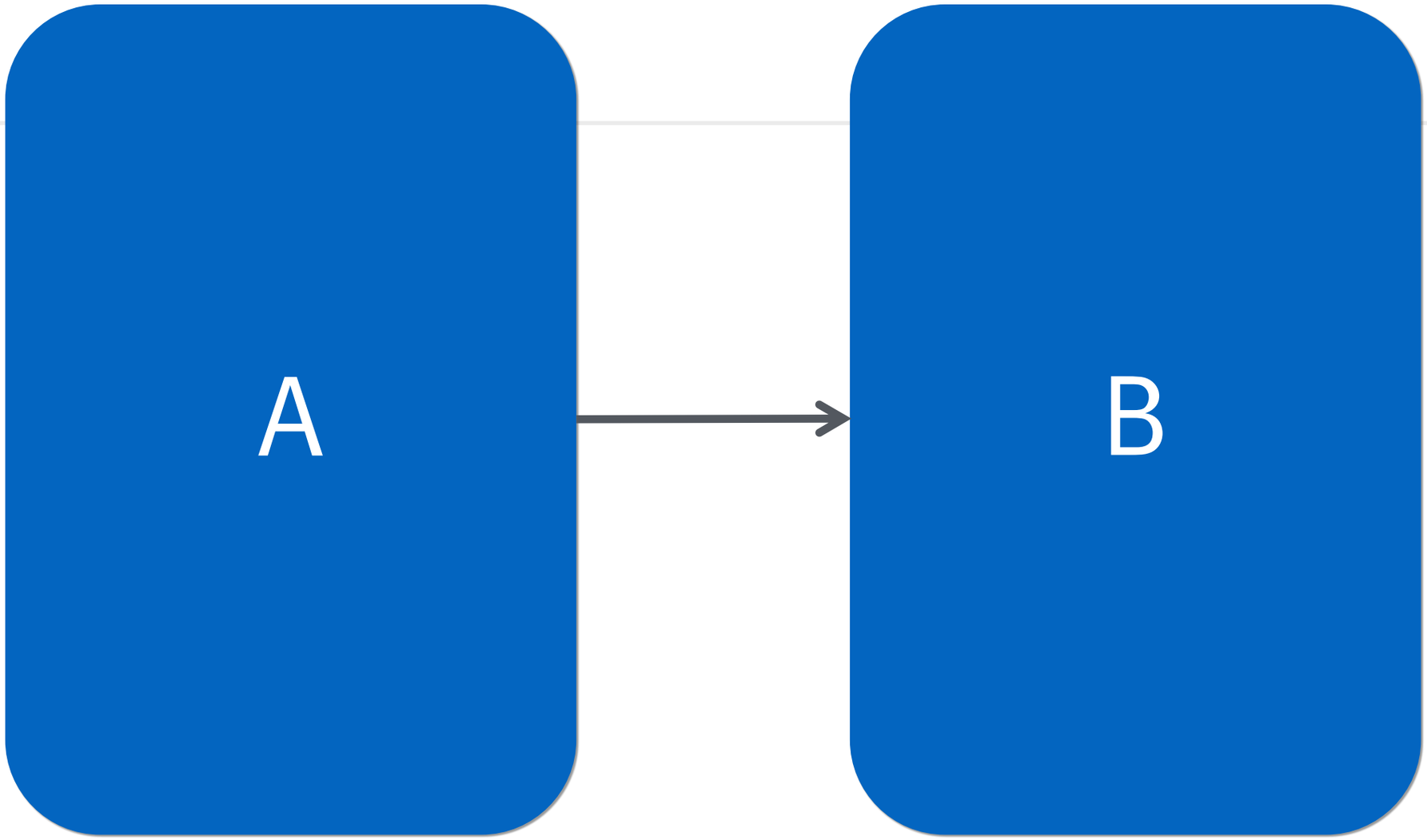
Architecture

- Should support changes
- ...with business value

- Needs to model the domain

- Hard to get right
- Architect needs to understand the domain

Is a great architecture free of cyclic dependencies?

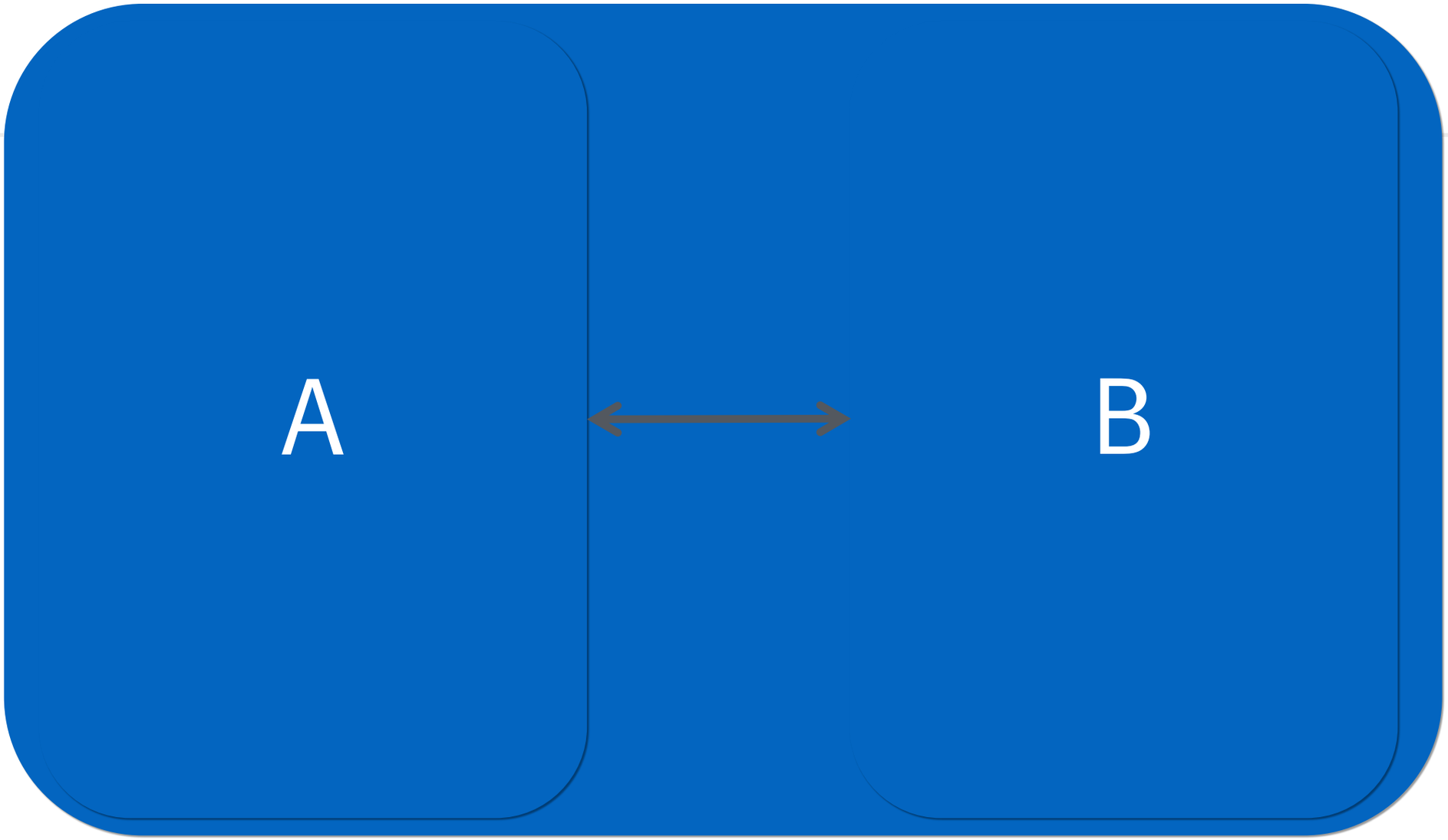


A depends on B

Changes to B influence A

Should be two components

In fact one component



A depends on B

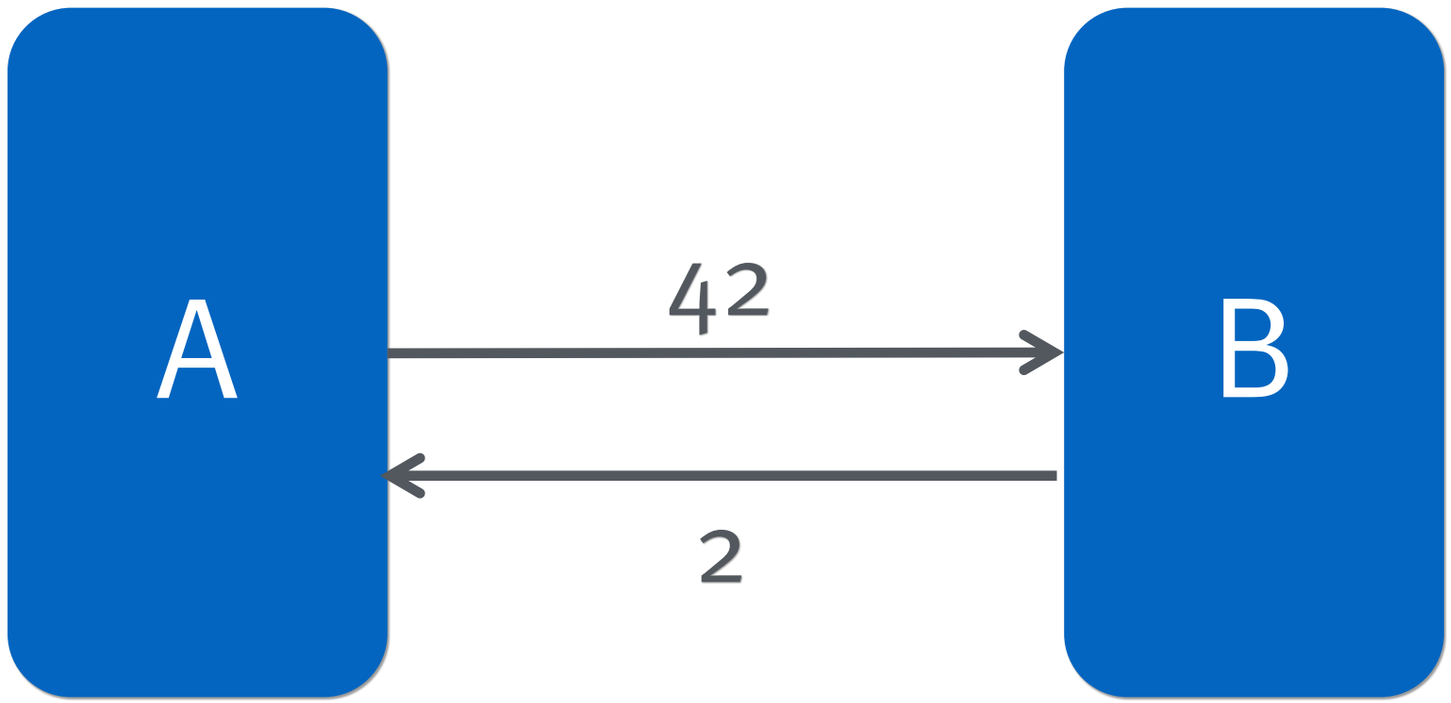
B depends on A

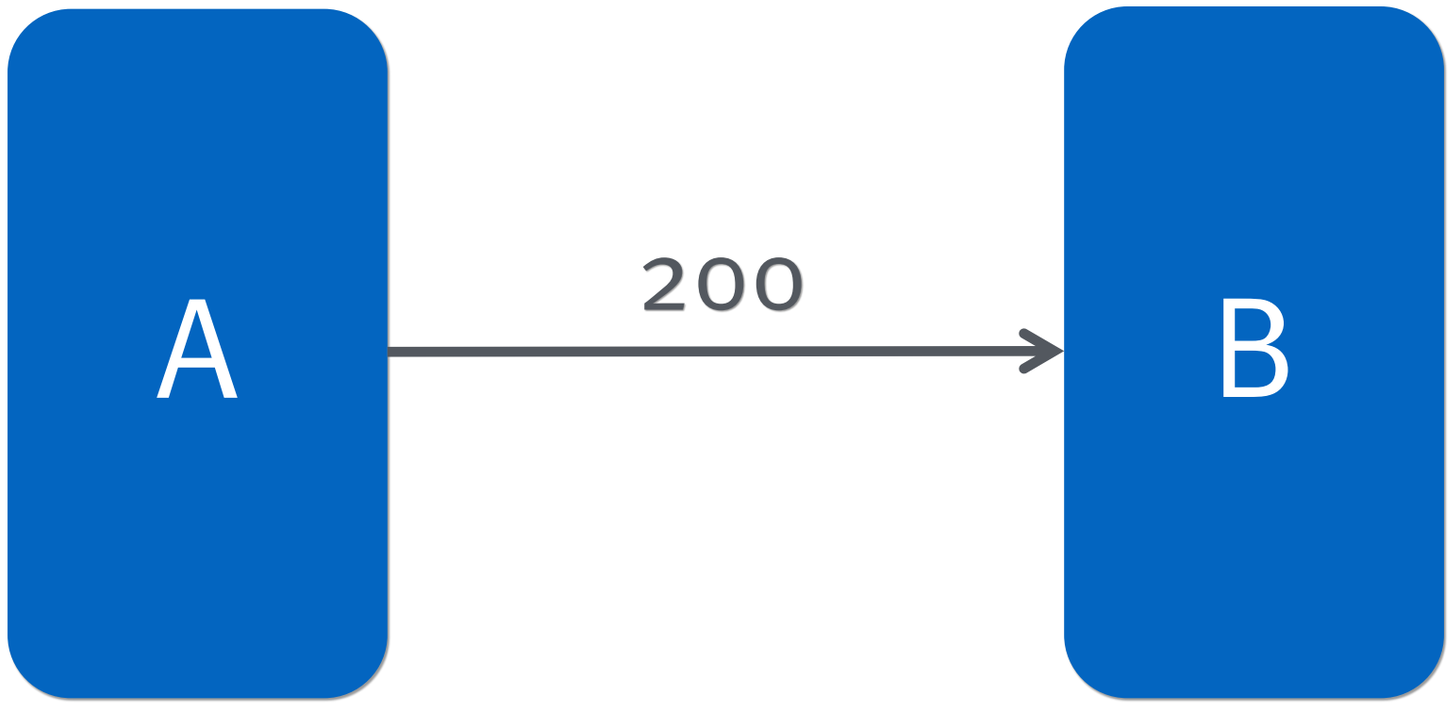
Changes to B influence A

Changes to A influence B

Is a great architecture free of cyclic dependencies?

Cyclic dependencies: Architects' Mortal Sin


A
42
B
2


A
200
B

Other Architecture Metrics

- High cohesion
- Elements of a module should belong together

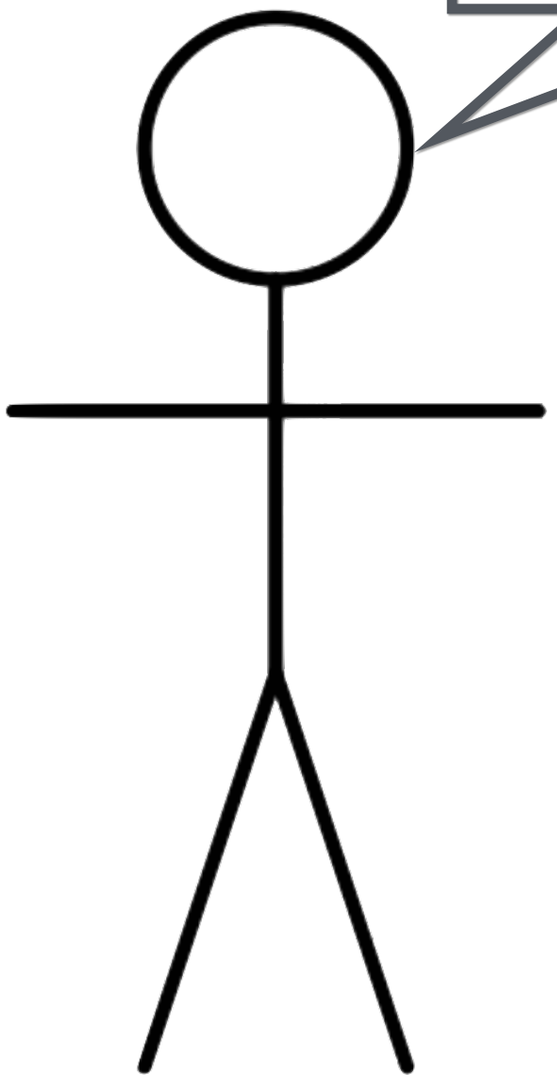
- Low coupling
- Modules should not depend on each other

Great Architecture

- Don’t overrate cyclic dependencies!
- Consider other metrics
- Architecture by domain

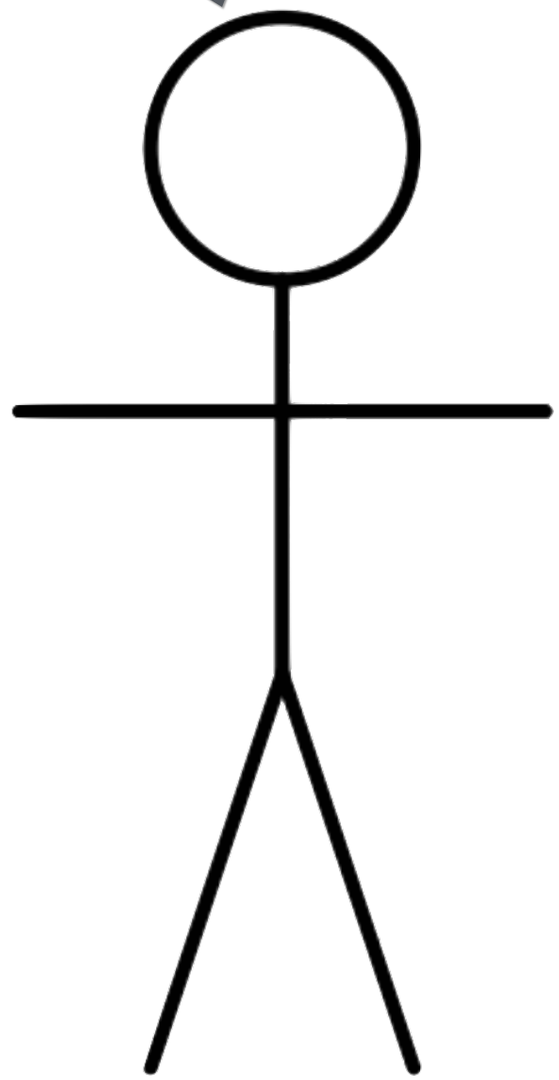
The worst legacy problems?

The project has a lot of cyclic dependencies!



I know.

...but that doesn't cause a lot of trouble



Architects must understand customer & his quality demands

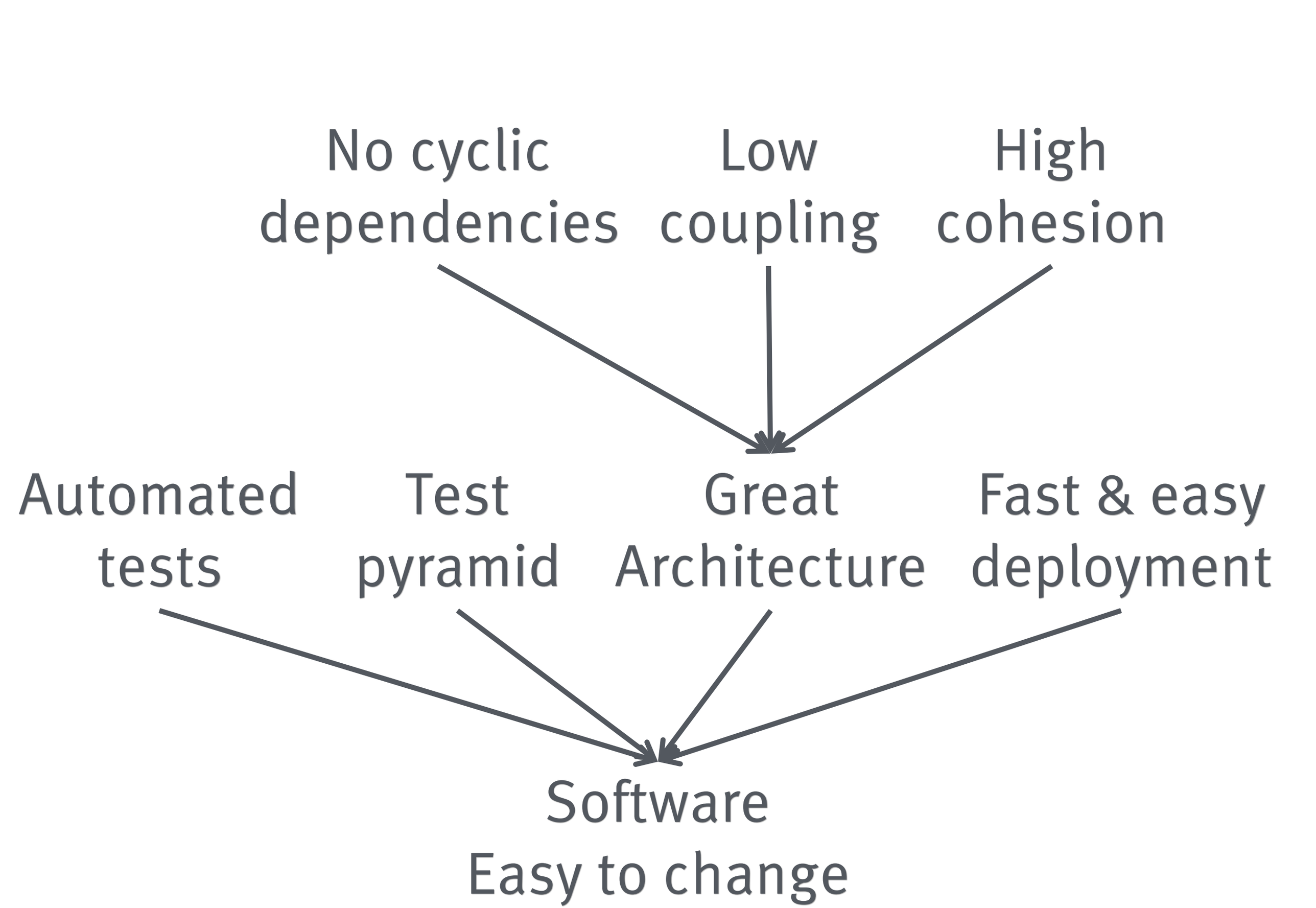
Quality

- Changeability
- Performance
- Security
- ...

No two projects are alike.

No general rules.

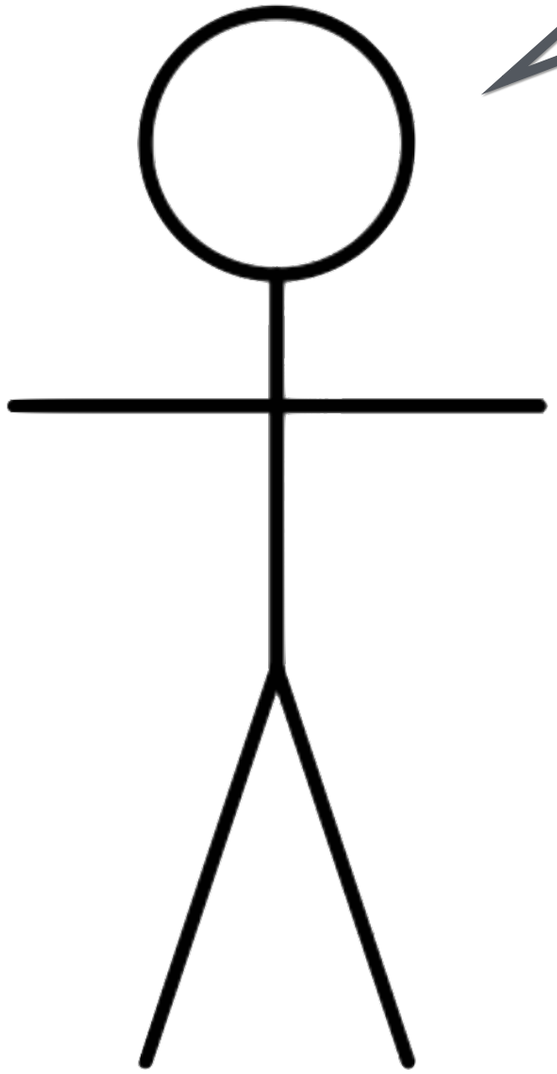
Sorry!


No cyclic dependencies
Low coupling
High cohesion
Automated tests
Test pyramid
Great Architecture
Fast & easy deployment
Software
Easy to change

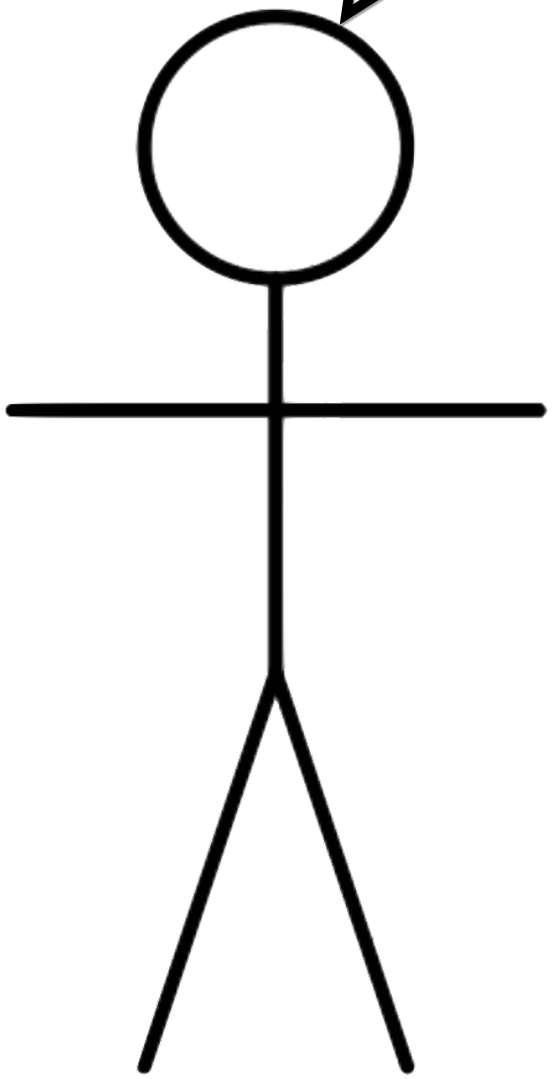
What should a Software Architect do?

Create a Proper Architecture.

I'm a Software Architect.
But I'm not doing architecture.



There is more to changeable software than architecture.



Thank You!

@ewolff