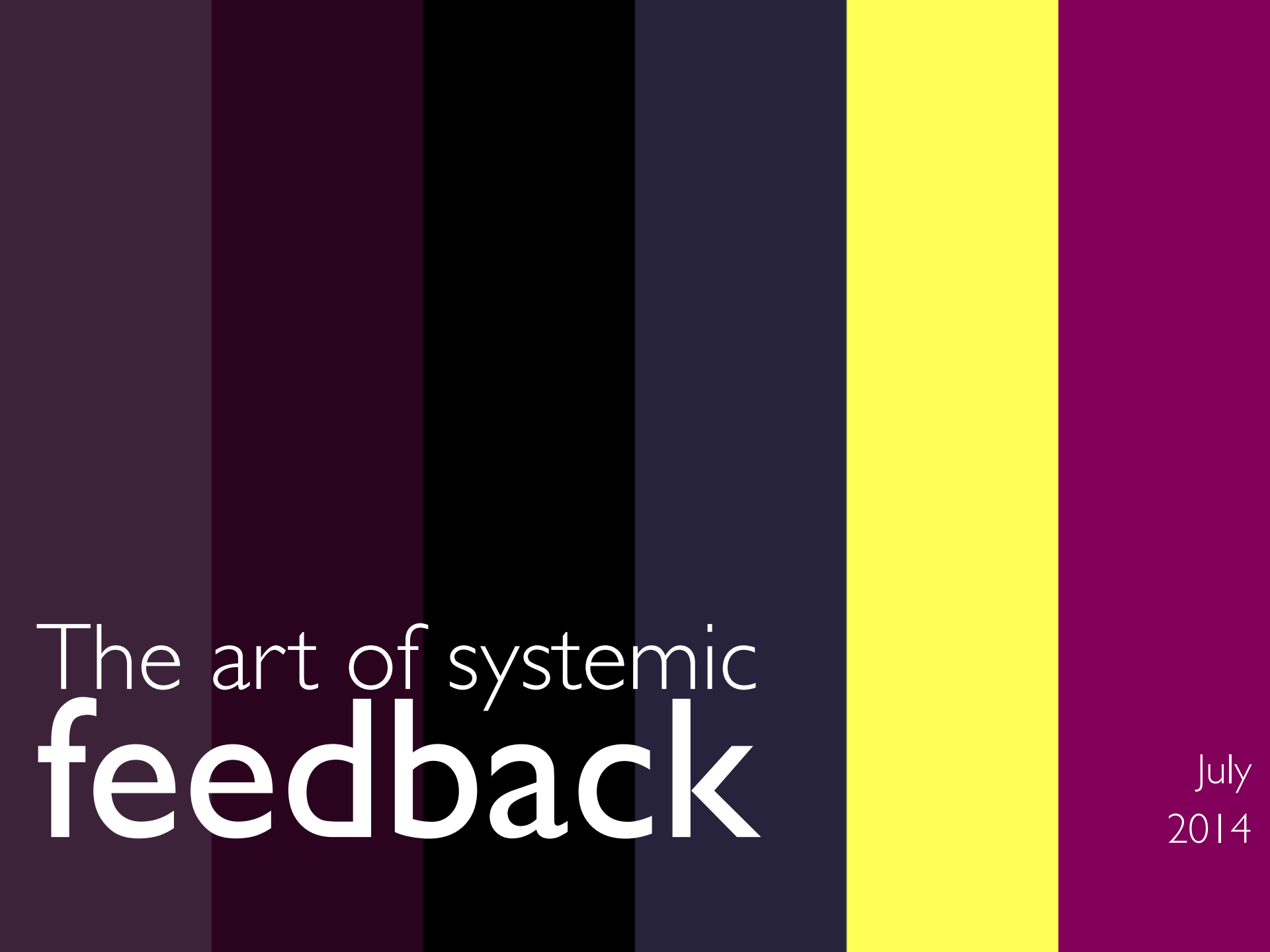




The art of systemic feedback

July
2014

I'm **Marcin Floryan**

<http://marcin.floryan.pl>



@mfloryan



Sergei

head of IT

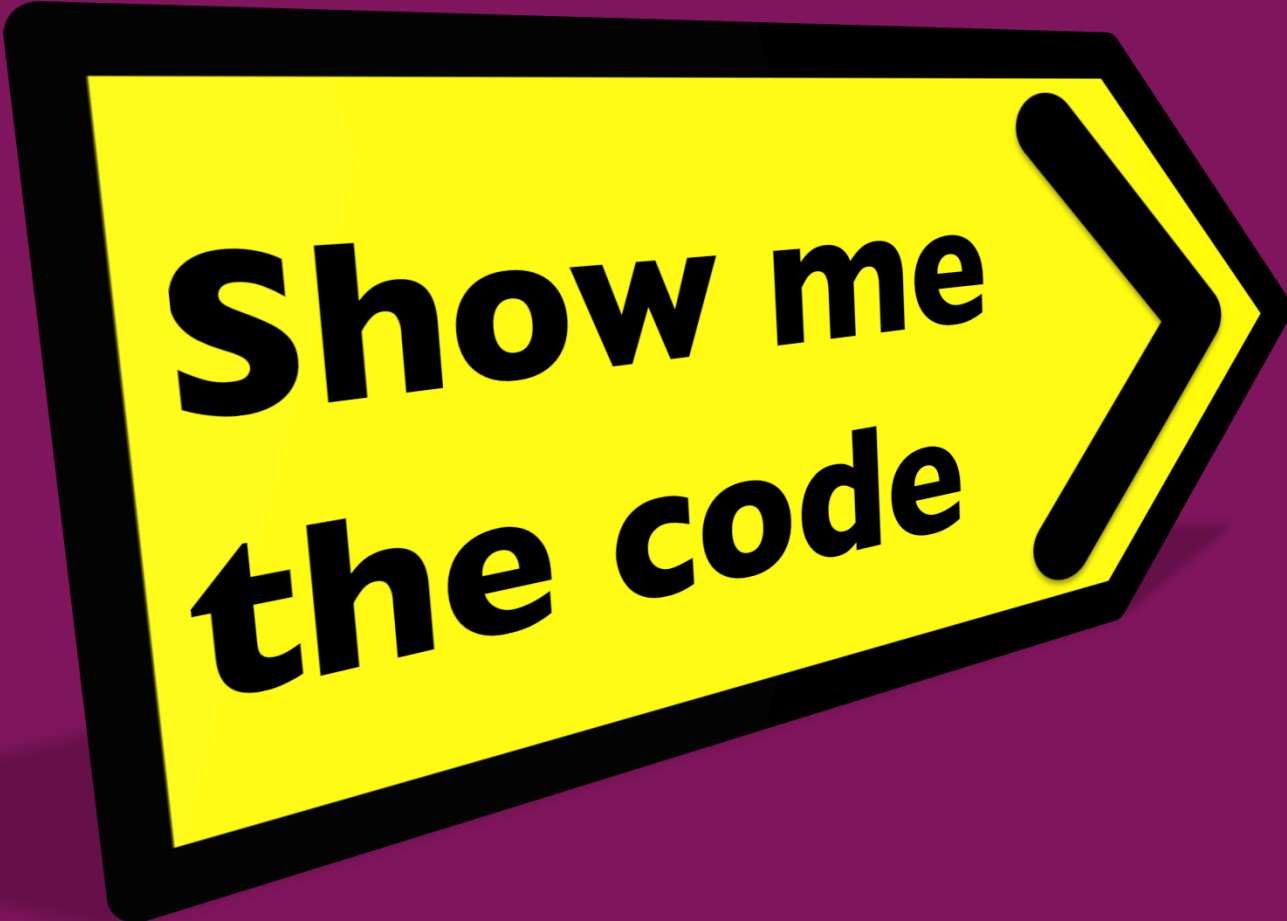


@mfloryan



Acceptance criteria

- ☐ Recognise feedback as a **tool**
- ☐ Identify properties and limitations of feedback
- ☐ Discover an approach to consider when using feedback



**Show me
the code**

```
public DistributorThrottler(  
    IDistributorProviderConfiguration distributorProviderConfiguration,  
    IPropertySource propertyStore) {  
  
    counter = new Counter();  
    numberOfProviders = distributorProviderConfiguration.NumberOfEnabledProviders;  
    var maximumPossibleRequestsInFlight =  
        HomeDistributorProviderProperty  
            .ProviderMaxRequestsPending  
            .From(propertyStore).ToInt() * numberOfProviders;  
}
```

```
public DistributorThrottler(  
    IDistributorProviderConfiguration distributorProviderConfiguration,  
    IPropertySource propertyStore) {  
  
    counter = new Counter();  
    numberOfProviders = distributorProviderConfiguration.NumberOfEnabledProviders;  
    var maximumPossibleRequestsInFlight =  
        HomeDistributorProviderProperty  
            .ProviderMaxRequestsPending  
            .From(propertyStore).ToInt() * numberOfProviders;  
}
```

```
var distributorProviderConfigurationMock =  
    MockRepository.GenerateMock<DistributorProviderConfiguration>  
        (propertyStoreMock, new FeatureSet());  
  
distributorProviderConfigurationMock  
    .Stub(c => c.NumberOfEnabledProviders)  
    .Return(2);  
  
throttler = new DistributorThrottler(  
    distributorProviderConfigurationMock,  
    propertyStoreMock);
```

```
throttler = new DistributorThrottler(propertyStoreMock, 2);
```

```
public DistributorThrottler(  
    IDistributorProviderConfiguration distributorProviderConfiguration,  
    IPropertySource propertyStore) {  
  
    counter = new Counter();  
    numberOfProviders = distributorProviderConfiguration.NumberOfEnabledProviders;  
    var maximumPossibleRequestsInFlight =  
        HomeDistributorProviderProperty  
            .ProviderMaxRequestsPending  
            .From(propertyStore).ToInt() * numberOfProviders;  
  

```

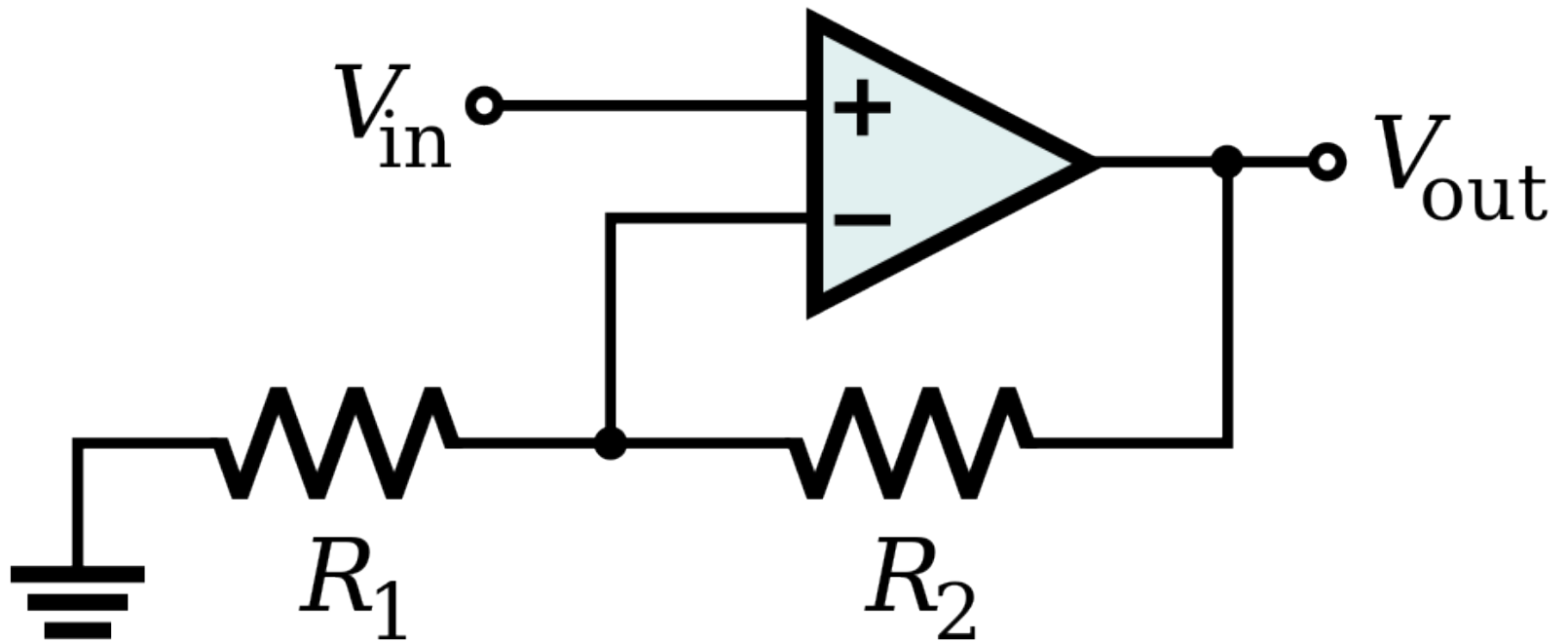
```
public DistributorThrottler(  
    int numberOfProviders,  
    IPropertySource propertyStore) {  
  
    counter = new Counter();  
    this.numberOfProviders = numberOfProviders;  
    var maximumPossibleRequestsInFlight =  
        HomeDistributorProviderProperty  
            .ProviderMaxRequestsPending  
            .From(propertyStore).ToInt() * this.numberOfProviders;  
  

```

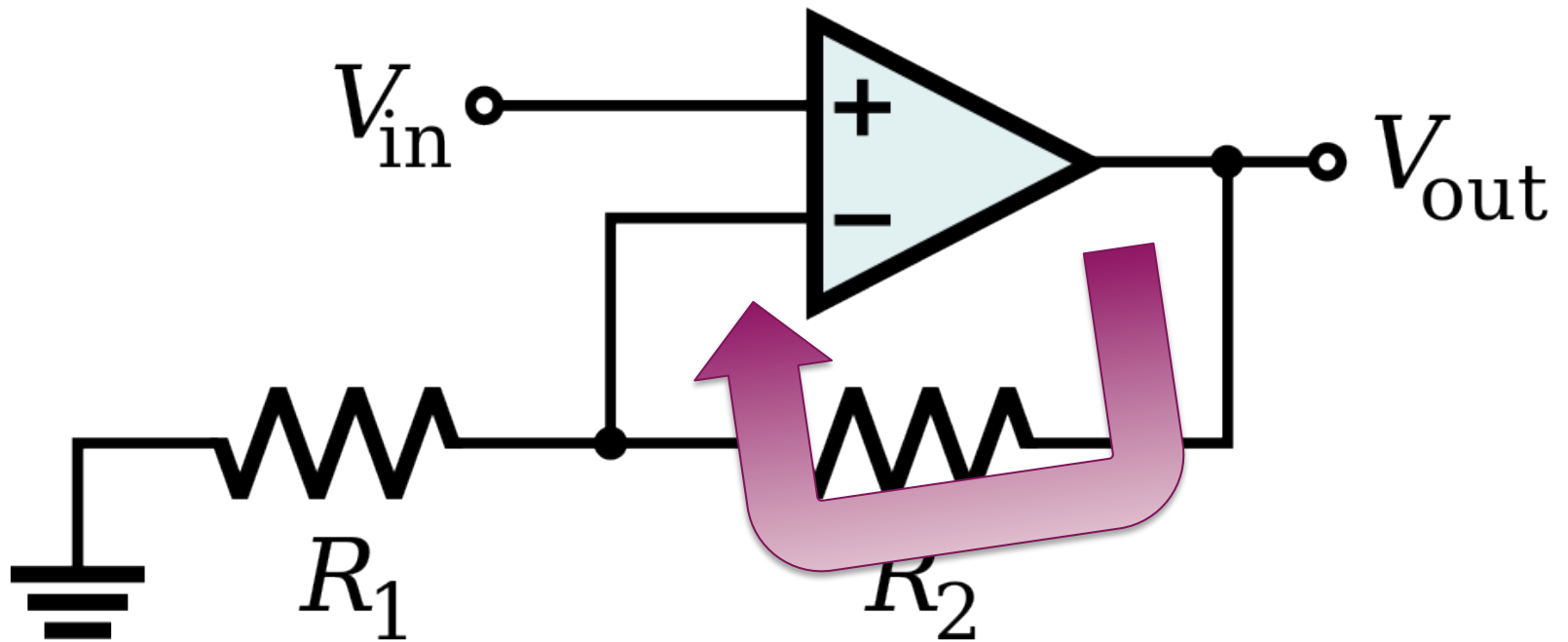


**What is
feedback**

Non-inverting amplifier



Non-inverting amplifier





feedback improves **performance**
(gain stability, linearity, frequency
response, step response) and
reduces sensitivity to parameter
variations due to manufacturing or
environment.

| **Wikipedia**



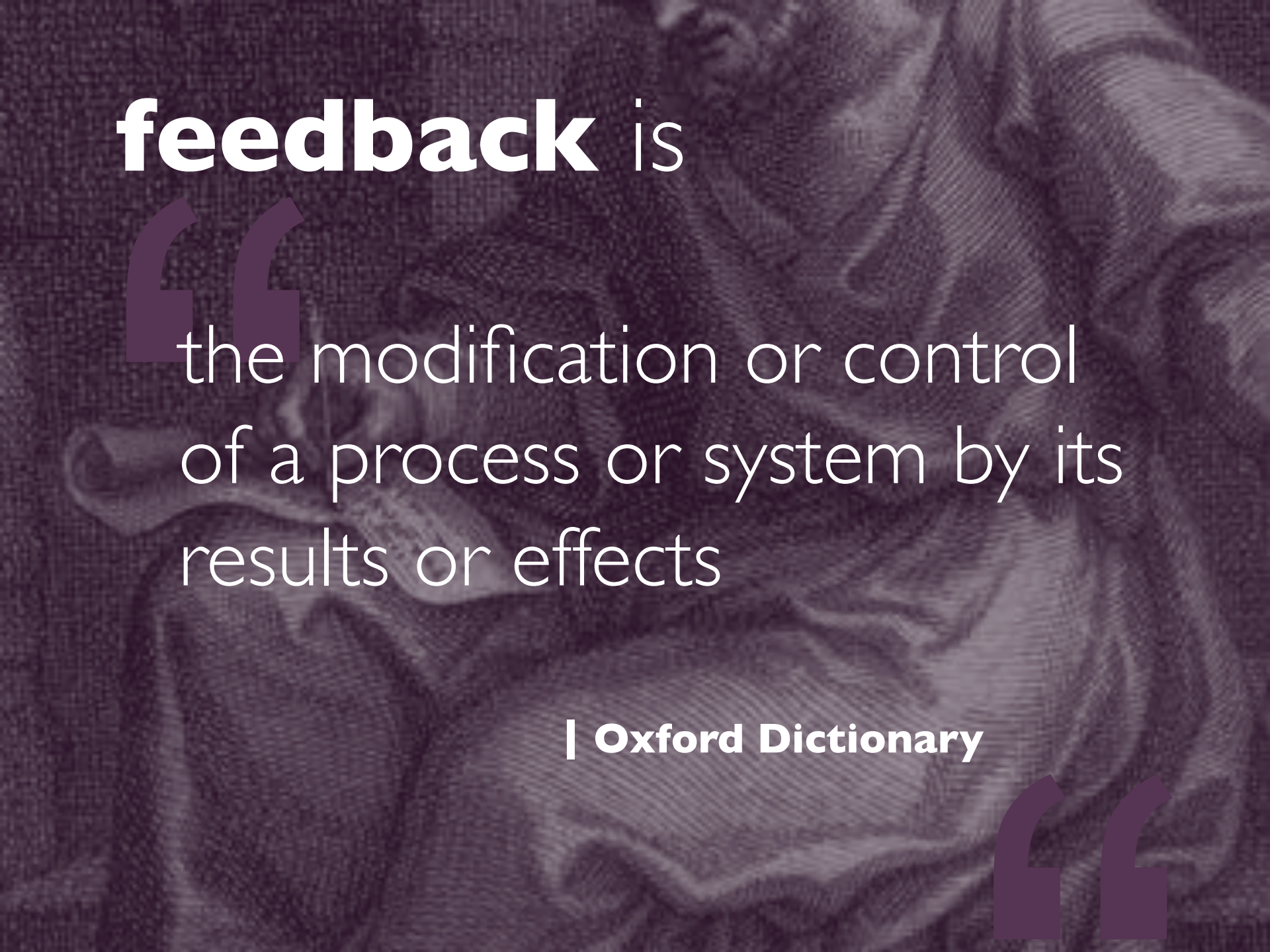
feedback is

“

information about reactions
to a product which is used
as a basis for improvement

| **Oxford Dictionary**

”



feedback is

“

the modification or control
of a process or system by its
results or effects

| **Oxford Dictionary**

”

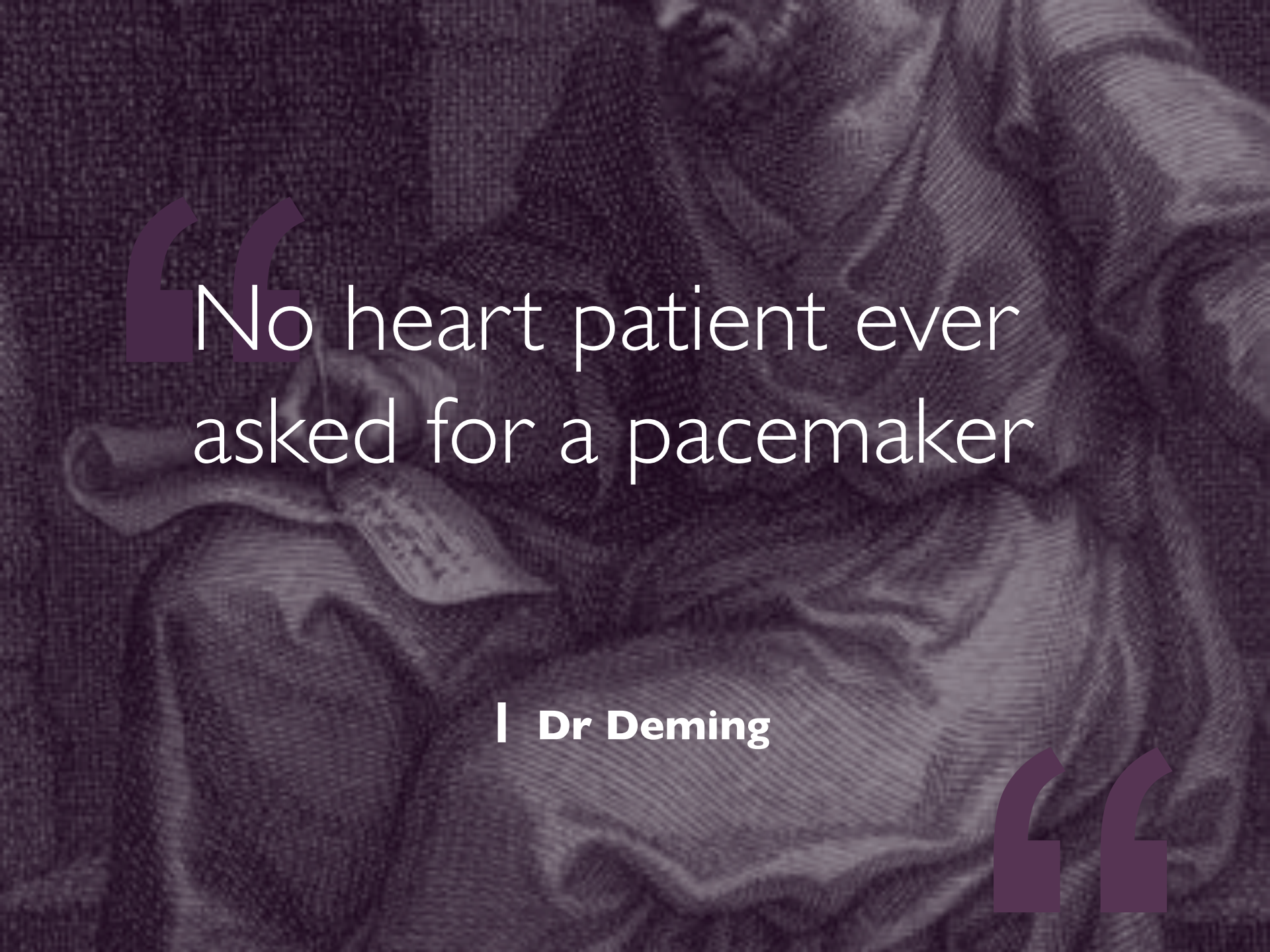


**Why we
need it**

3 things we wish were true:

1. The client knows what they want.
2. The developer knows how to build it.
3. Nothing will change along the way.

The client
knows
what they want



“No heart patient ever
asked for a pacemaker

| **Dr Deming**

”

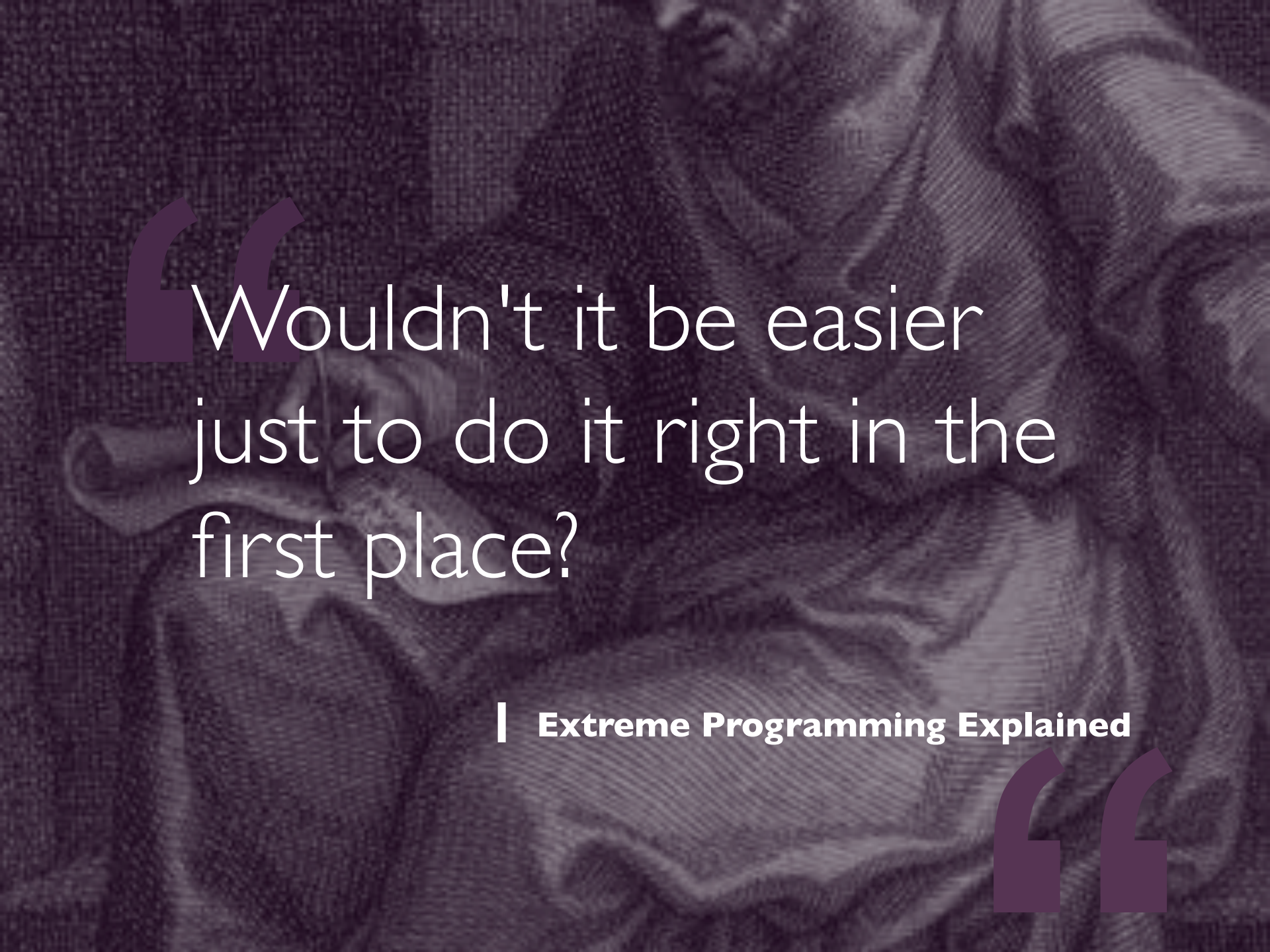


“People generally work better by starting with something concrete and tangible, such as examples, by altering rather than creating from scratch, by watching, and by getting feedback.”

| **Agile Software Development:
The Cooperative Game**



The developer
knows
how to build it



“Wouldn't it be easier
just to do it right in the
first place?”

| **Extreme Programming Explained**



Of course it would,
except
for three things:



We may not know how to
do it "right".

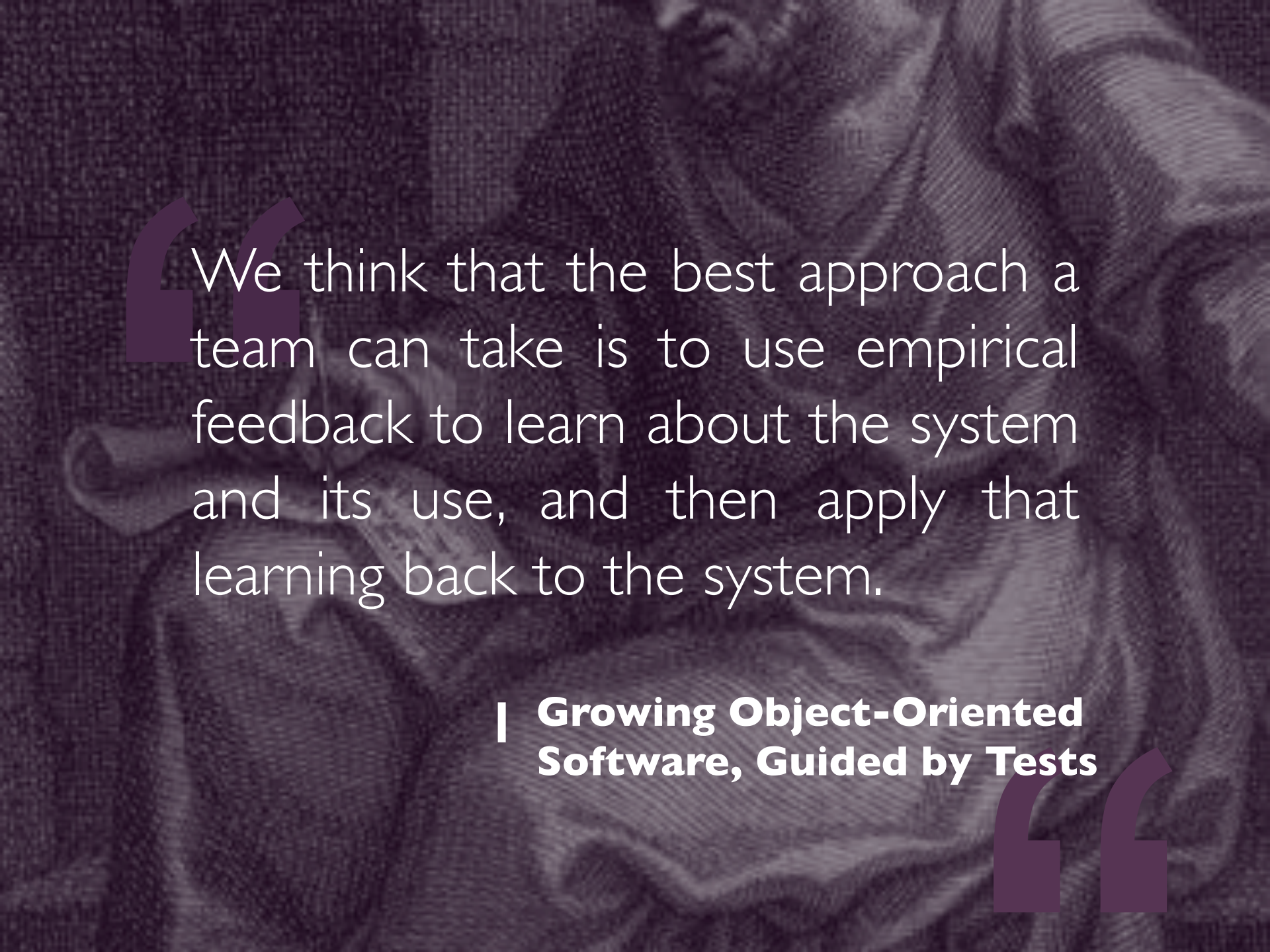




What's right for today may
be wrong for tomorrow.



Doing everything "right"
today might take so long
that changing
circumstances tomorrow
invalidate today's solution
before it is even finished.

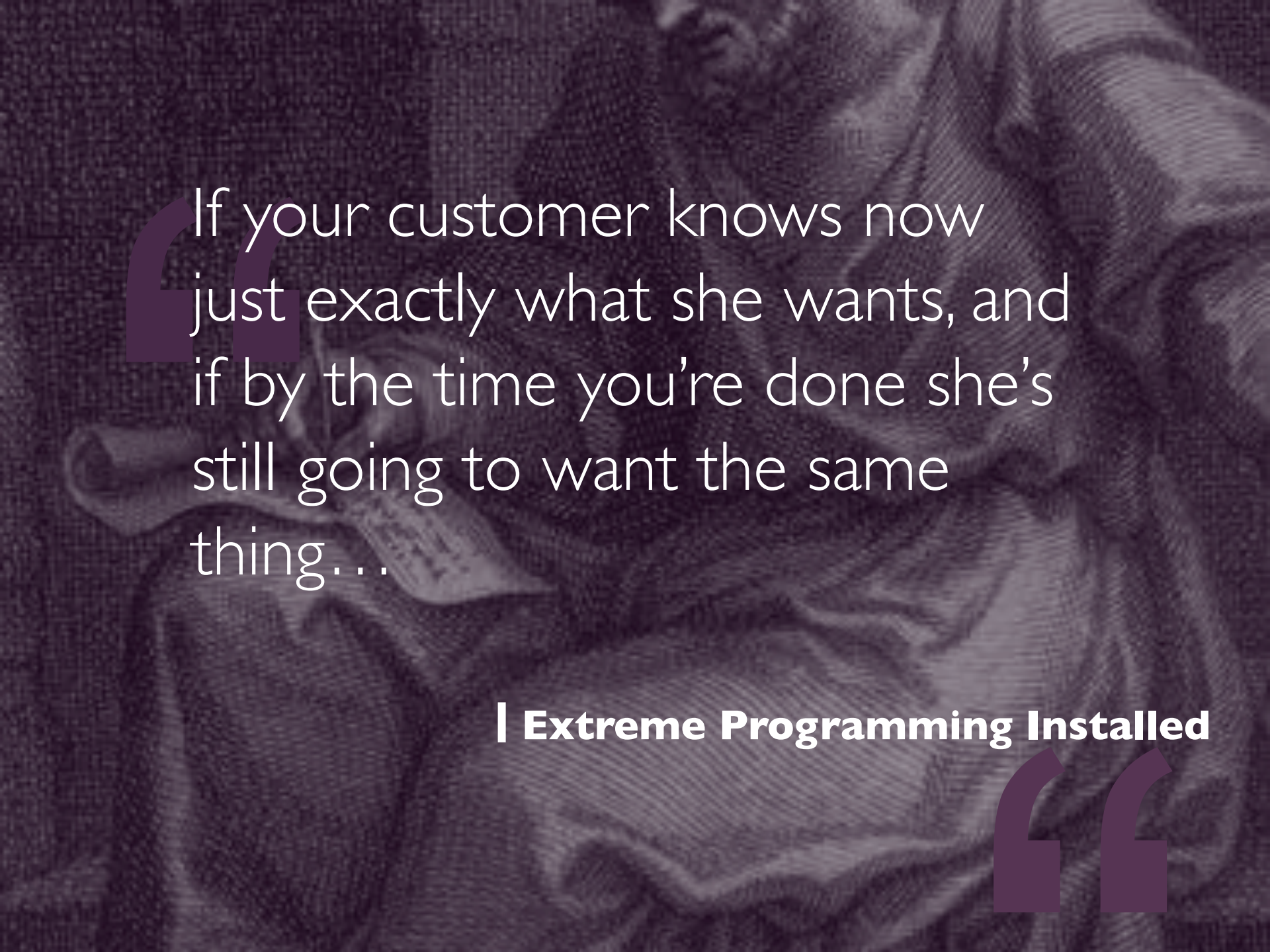
A person wearing a plaid shirt is looking at a laptop screen. The image is dark and serves as a background for the text.

“We think that the best approach a team can take is to use empirical feedback to learn about the system and its use, and then apply that learning back to the system.”

| **Growing Object-Oriented Software, Guided by Tests**

A large, stylized closing quotation mark in a dark purple color.

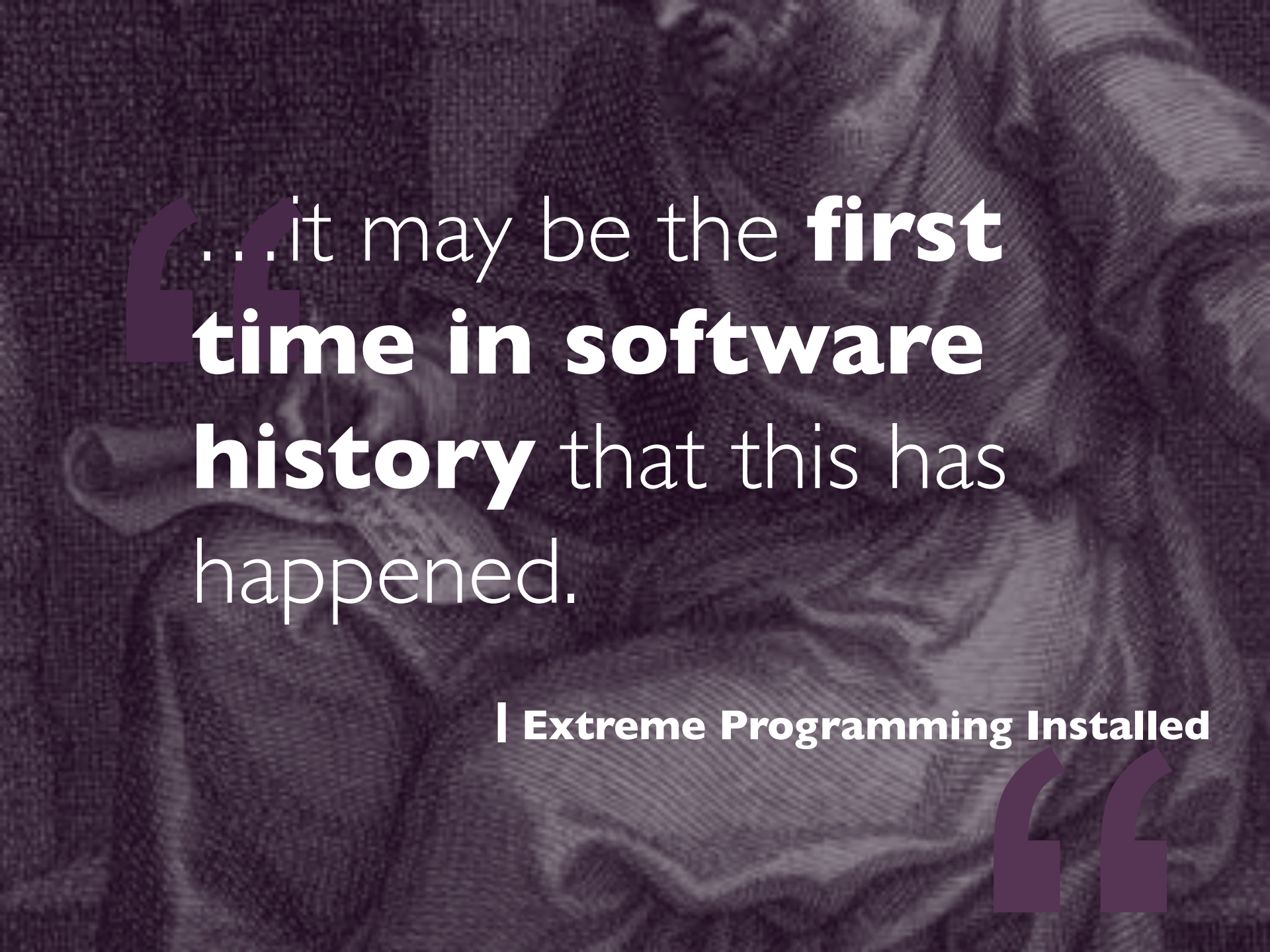
Nothing will
change
along the way

A person is lying down, possibly asleep, with their head tilted back. A rolled-up document and a pen are resting on their chest. The background is a textured, dark purple surface.

“If your customer knows now
just exactly what she wants, and
if by the time you’re done she’s
still going to want the same
thing...”

| Extreme Programming Installed

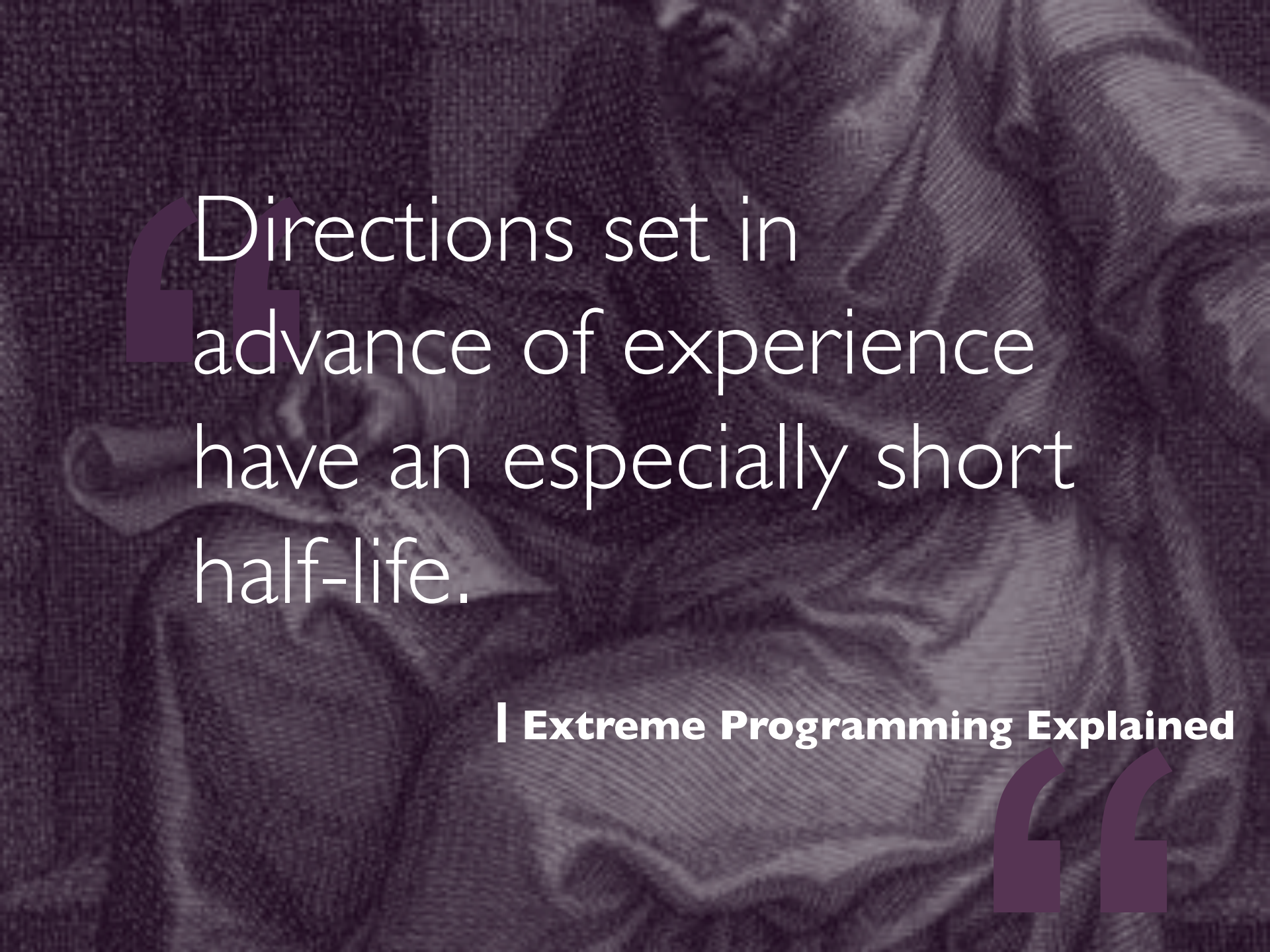
A large, stylized closing quotation mark in a dark purple color, positioned at the bottom right of the slide.



“...it may be the **first**
time in software
history that this has
happened.

| **Extreme Programming Installed**





“Directions set in
advance of experience
have an especially short
half-life.”

| **Extreme Programming Explained**





**Agile and
feedback**

MANIFESTO FOR AGILE SOFTWARE DEVELOPMENT

WE ARE UNCOVERING BETTER WAYS OF DEVELOPING
SOFTWARE BY DOING IT AND HELPING OTHERS DO IT.
THROUGH THIS WORK WE HAVE COME TO VALUE:



INDIVIDUALS AND INTERACTIONS OVER PROCESSES AND TOOLS
WORKING SOFTWARE OVER COMPREHENSIVE DOCUMENTATION
CUSTOMER COLLABORATION OVER CONTRACT NEGOTIATION
RESPONDING TO CHANGE OVER FOLLOWING A PLAN

THAT IS, WHILE THERE IS VALUE IN THE ITEMS ON
THE RIGHT, WE VALUE THE ITEMS ON THE LEFT MORE.



KENT BECK

MIKE BEEDLE

ARIE VAN BENNEKUM

ALISTAIR COCKBURN

WARD CUNNINGHAM

MARTIN FOWLER

JAMES GRENNING

JIM HIGHSMITH

ANDREW HUNT

RON JEFFRIES

JON KERN

BRIAN MARICK

ROBERT C. MARTIN

STEVE MELLOR

KEN SCHWABER

JEFF SUTHERLAND

DAVE THOMAS

©2001, THE ABOVE AUTHORS. THIS DECLARATION MAY BE FREELY COPIED IN ANY FORM,
BUT ONLY IN ITS ENTIRETY THROUGH THIS NOTICE.

We are **uncovering**
better ways
of developing software
by doing it and helping
others do it

Agile manifesto – principle 12

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.





“To mitigate the risk of uncertainty, Agile prescribes short iterations with **lots of feedback.**”

| **Lean-Agile Software Development**



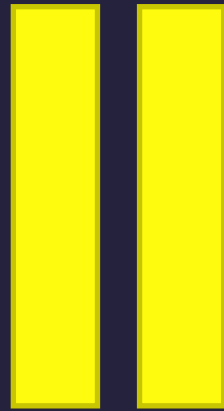


**Making
it work**

How's this for feedback?



feedback

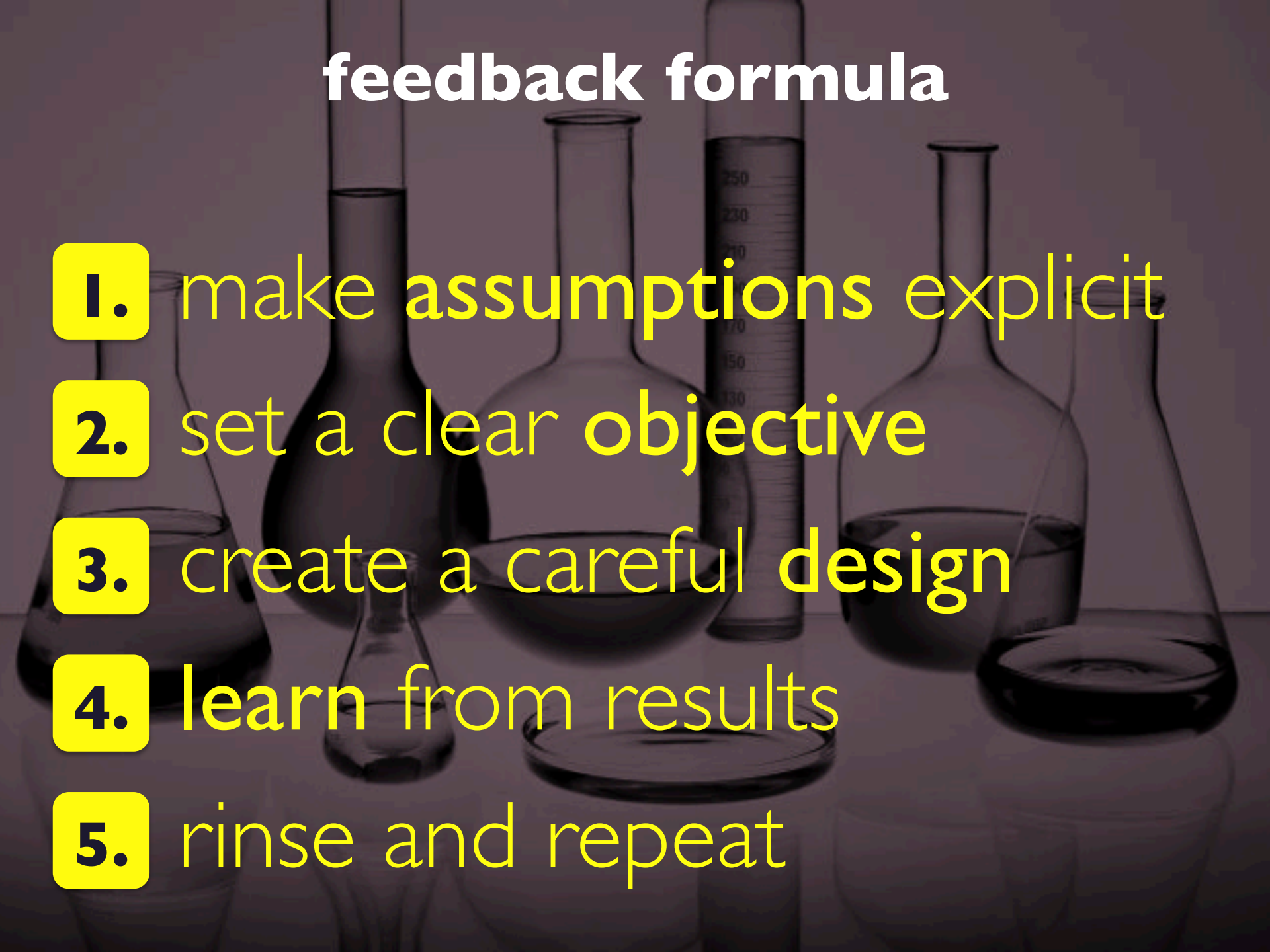


learning

deliberate learning



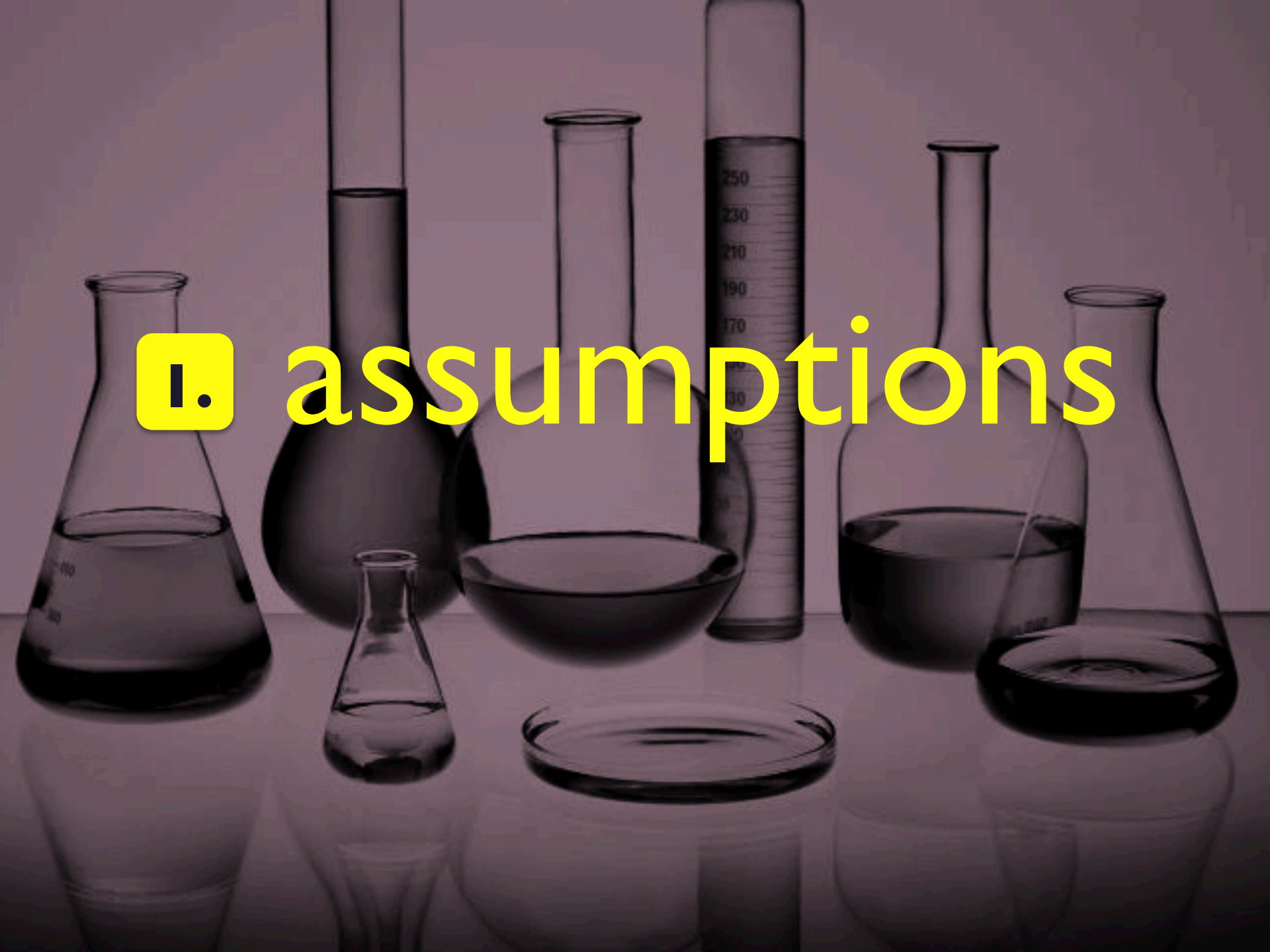
feedback formula

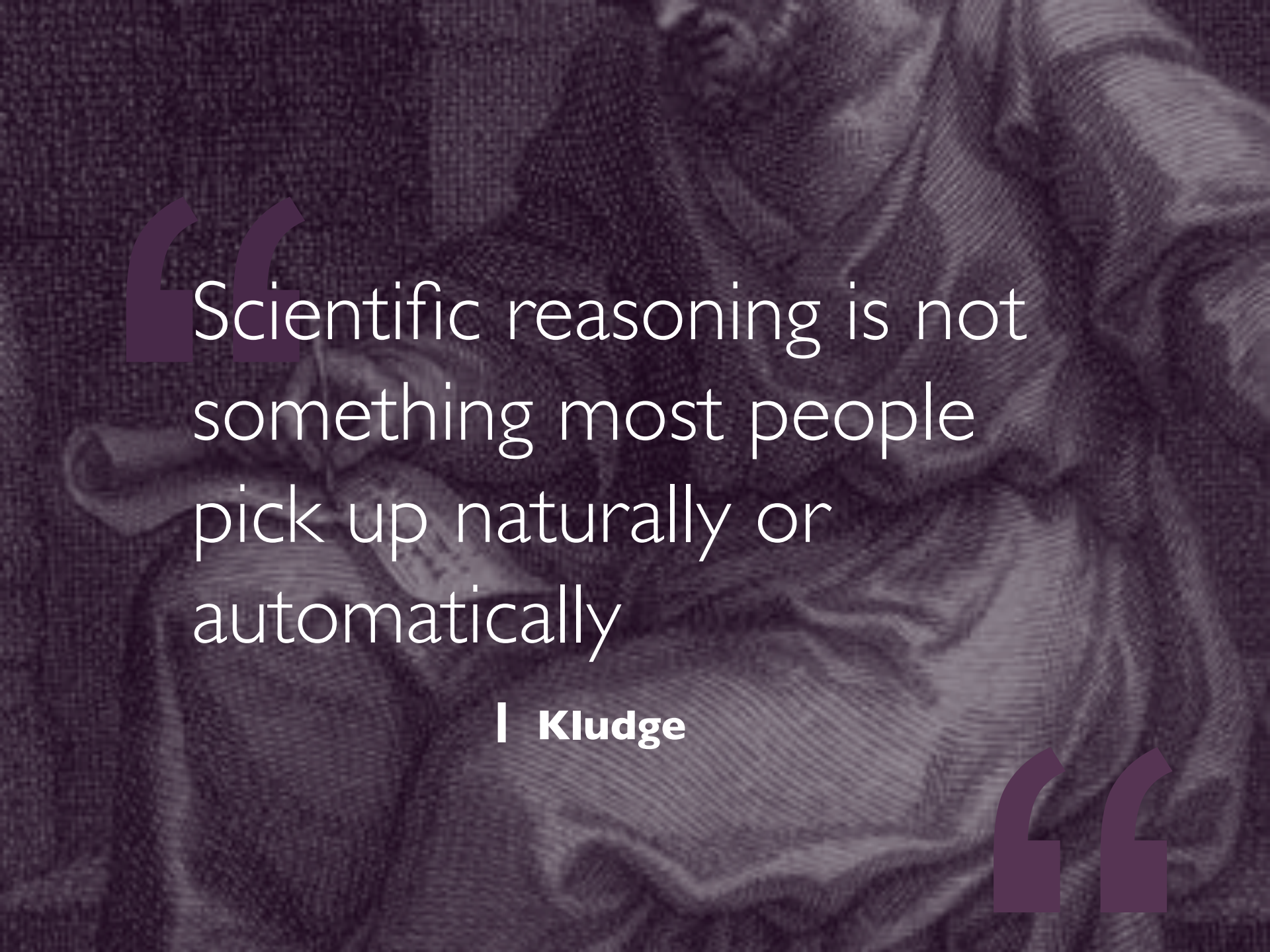


1. make **assumptions** explicit
2. set a clear **objective**
3. create a careful **design**
4. **learn** from results
5. rinse and repeat



assumptions





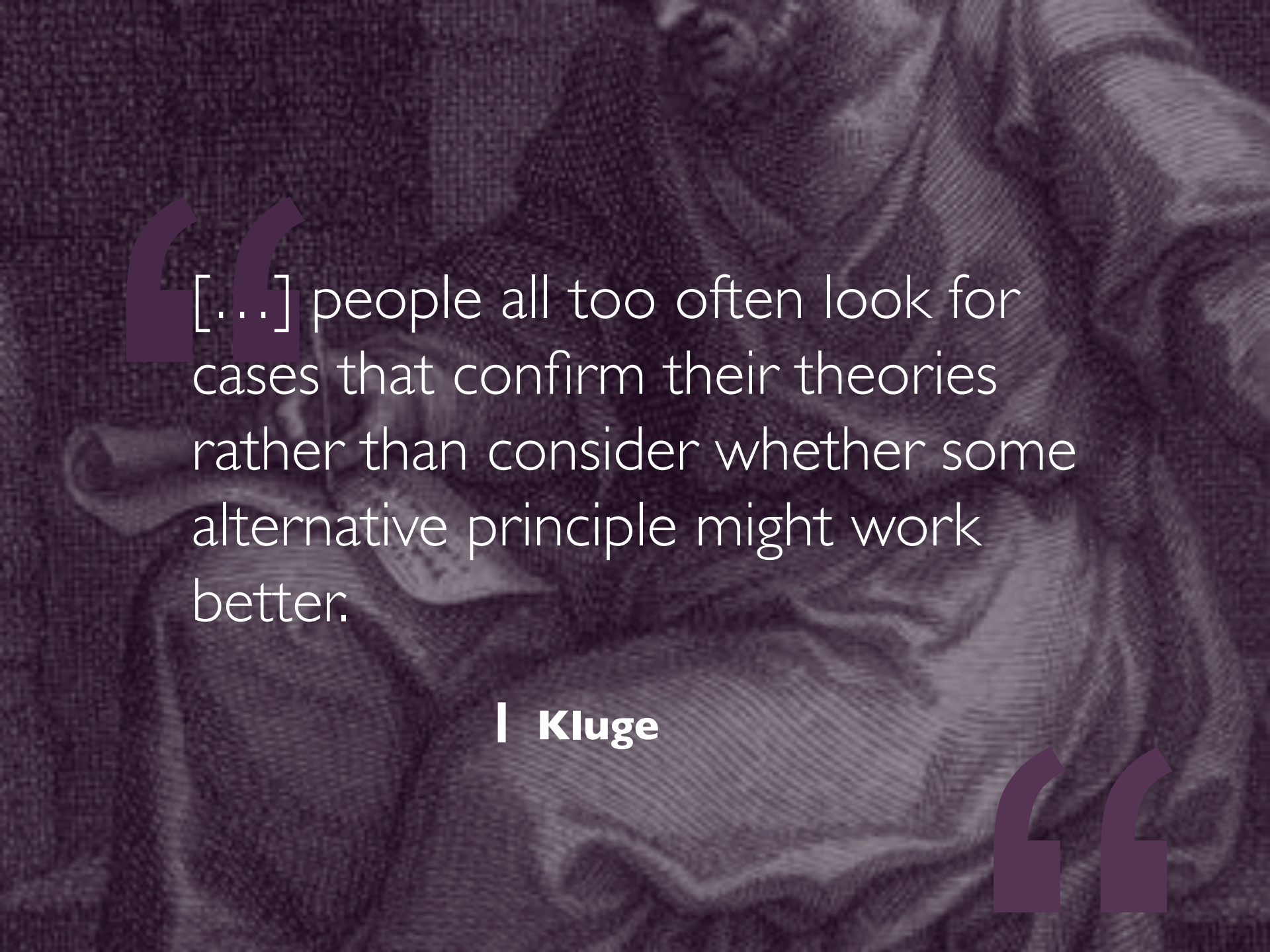
“Scientific reasoning is not
something most people
pick up naturally or
automatically

| **Kludge**

”



**confirmation
bias**



“ [...] people all too often look for cases that confirm their theories rather than consider whether some alternative principle might work better.

| **Kluge**



A collection of laboratory glassware, including Erlenmeyer flasks, a graduated cylinder, and a petri dish, arranged on a reflective surface. The glassware contains liquids of varying levels. The background is a solid dark purple color.

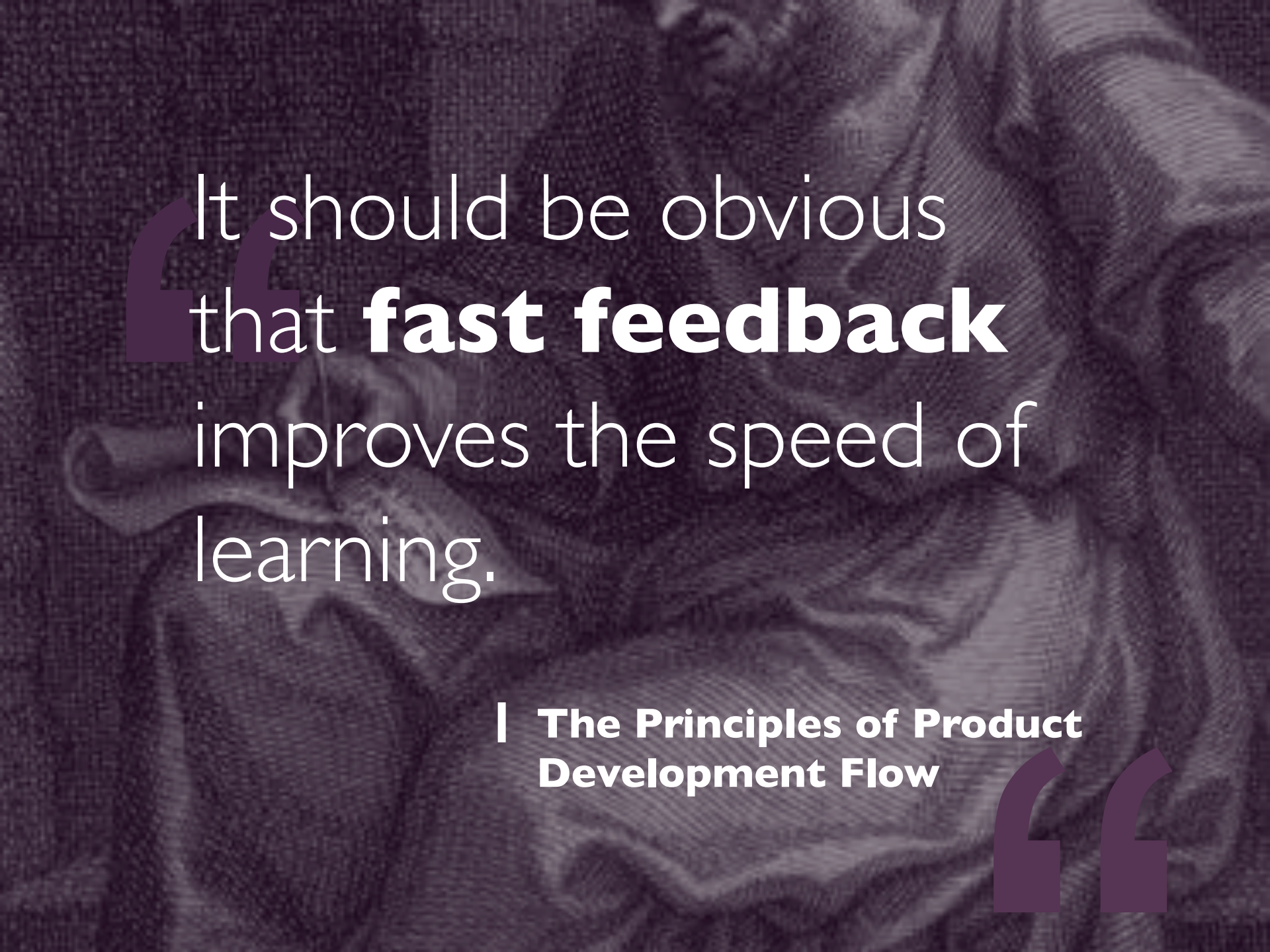
2. clear objective

A collection of laboratory glassware including Erlenmeyer flasks, test tubes, a graduated cylinder, a beaker, and a petri dish, all containing liquids of varying colors (clear, yellow, red, blue). The glassware is arranged on a reflective surface.

3. careful design

fast feedback cycles

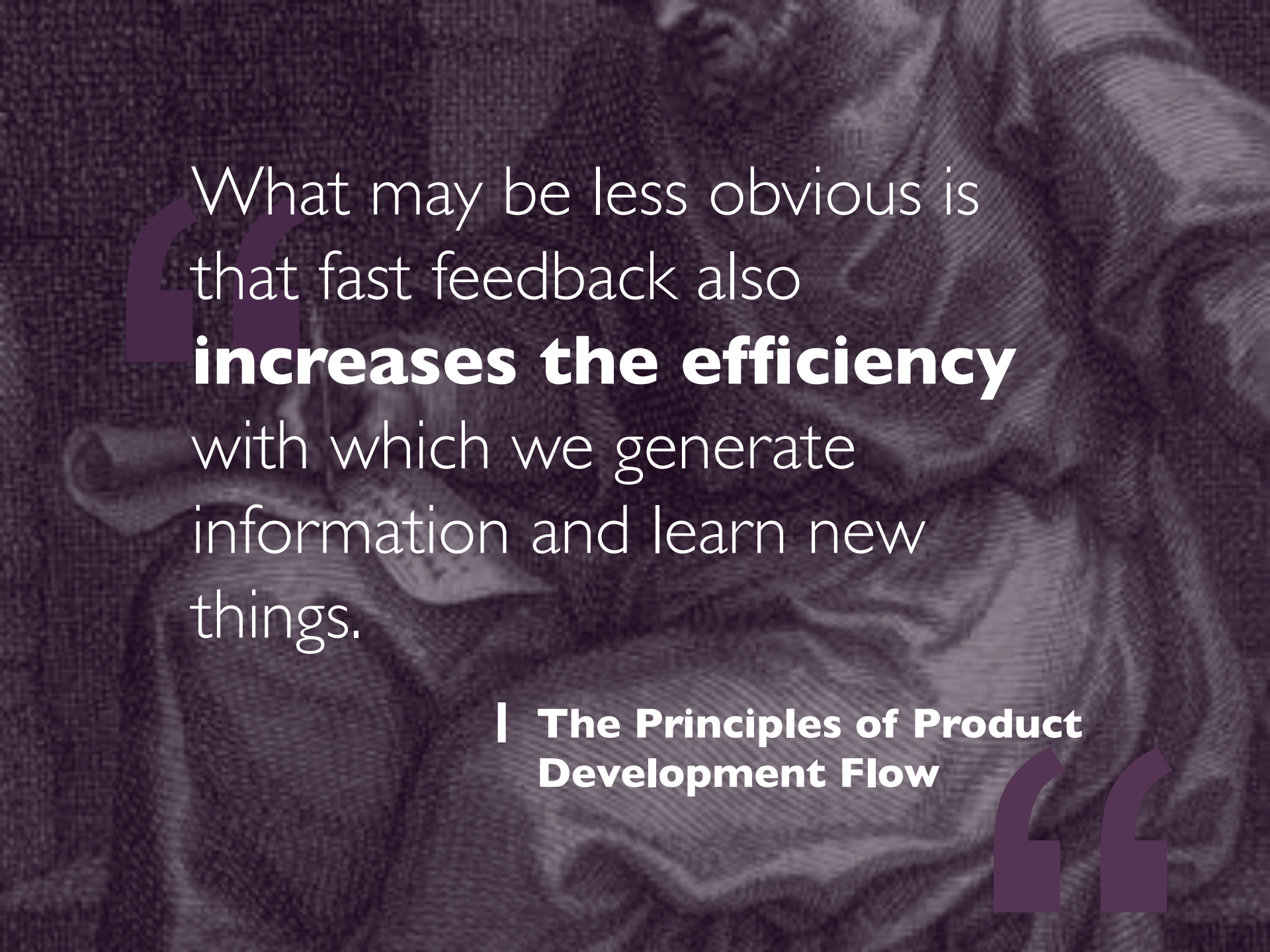
3.



It should be obvious
that **fast feedback**
improves the speed of
learning.

| **The Principles of Product
Development Flow**



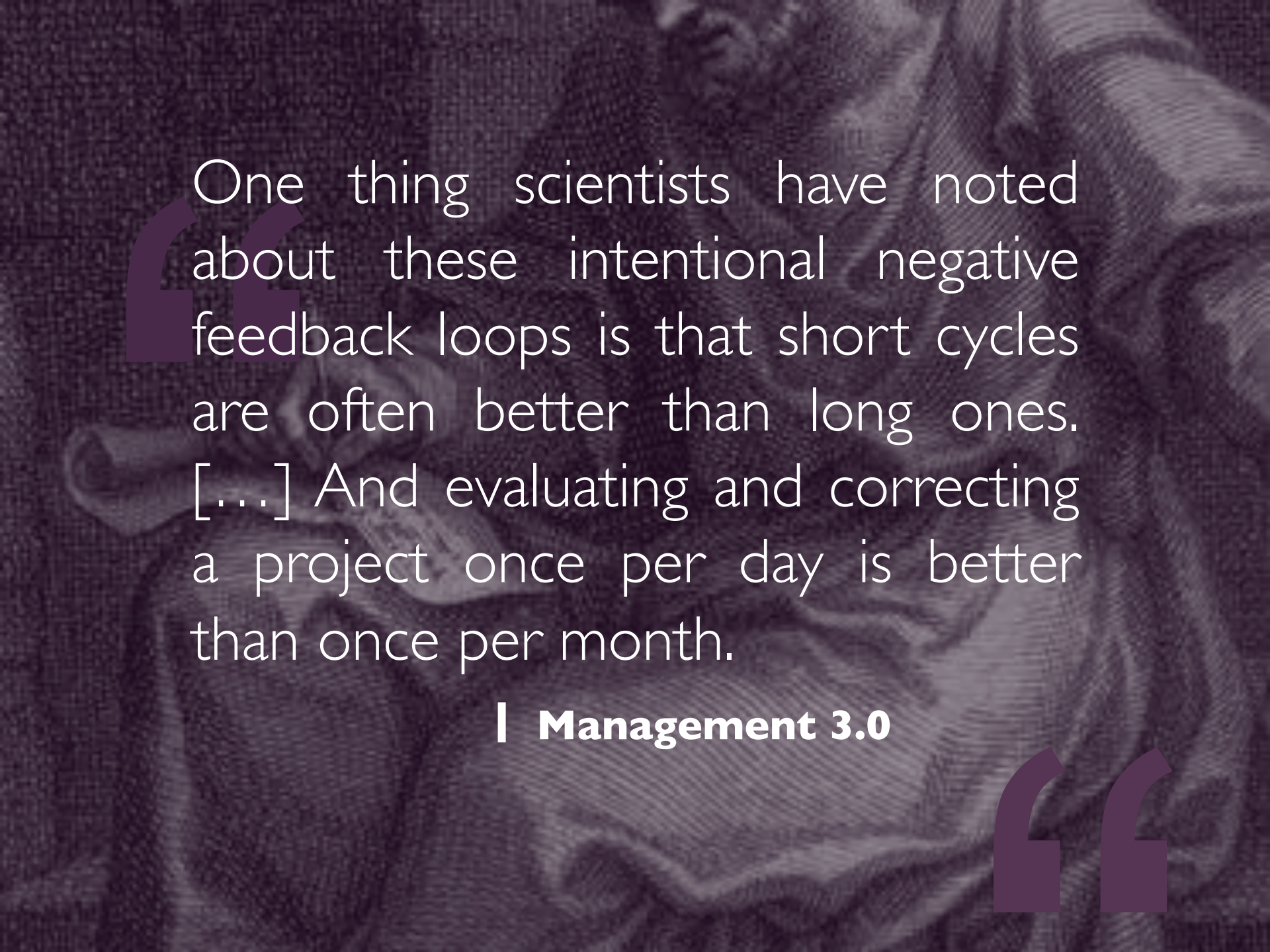


What may be less obvious is
that fast feedback also
increases the efficiency
with which we generate
information and learn new
things.

| **The Principles of Product
Development Flow**

short feedback loops

3.



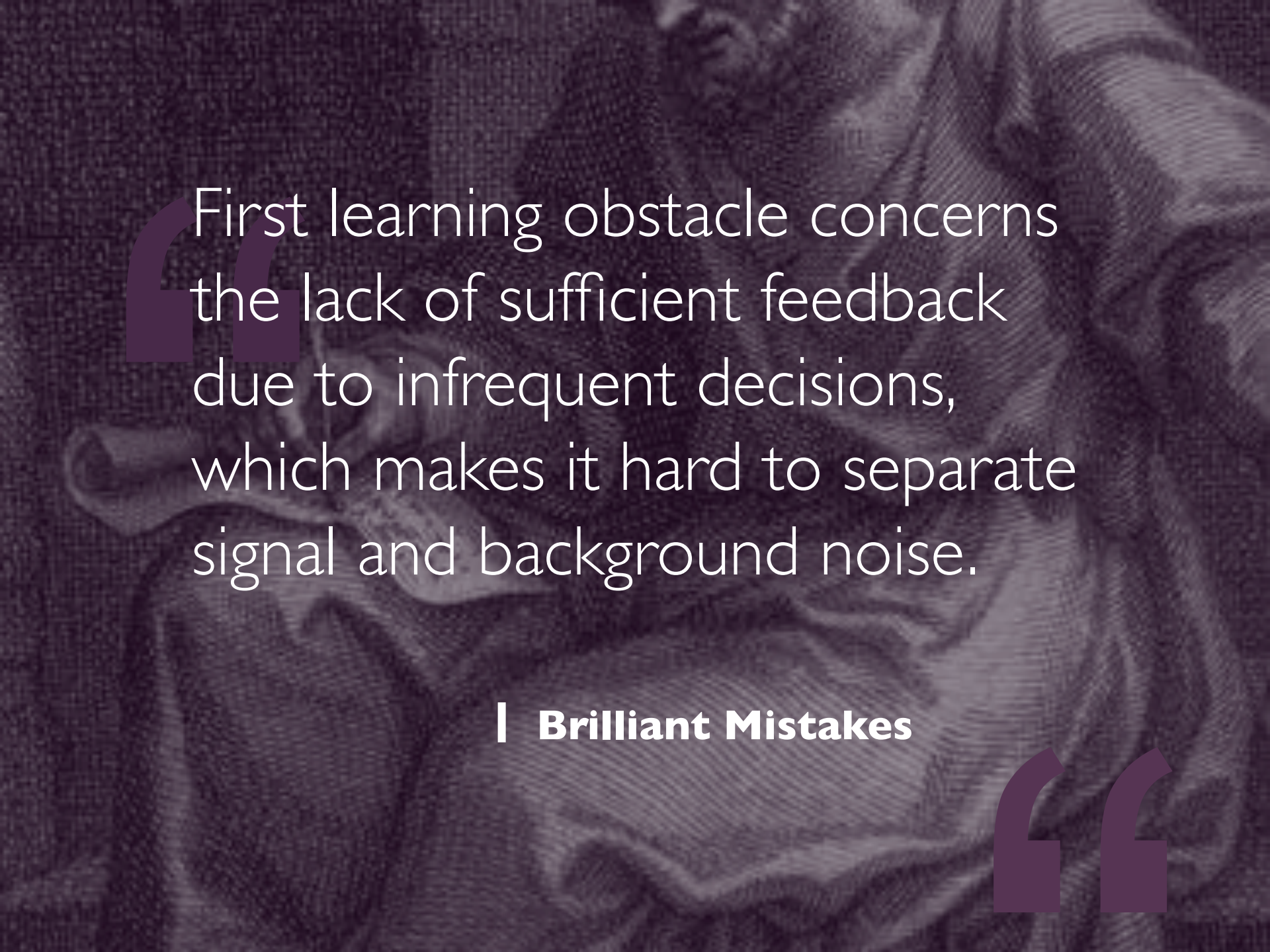
One thing scientists have noted about these intentional negative feedback loops is that short cycles are often better than long ones. [...] And evaluating and correcting a project once per day is better than once per month.

| Management 3.0



elimination of *noise*

3.



“First learning obstacle concerns the lack of sufficient feedback due to infrequent decisions, which makes it hard to separate signal and background noise.

| **Brilliant Mistakes**



AMERICA'S CUP



A collection of laboratory glassware including Erlenmeyer flasks, test tubes, a graduated cylinder, a beaker, and a petri dish, all containing liquids of varying colors (clear, yellow, red, blue). The glassware is arranged on a reflective surface.

4. learning



Whether professionals have a chance to develop intuitive expertise depends essentially on the quality and speed of feedback, as well as on sufficient opportunity to practice.

| **Thinking Fast And Slow**

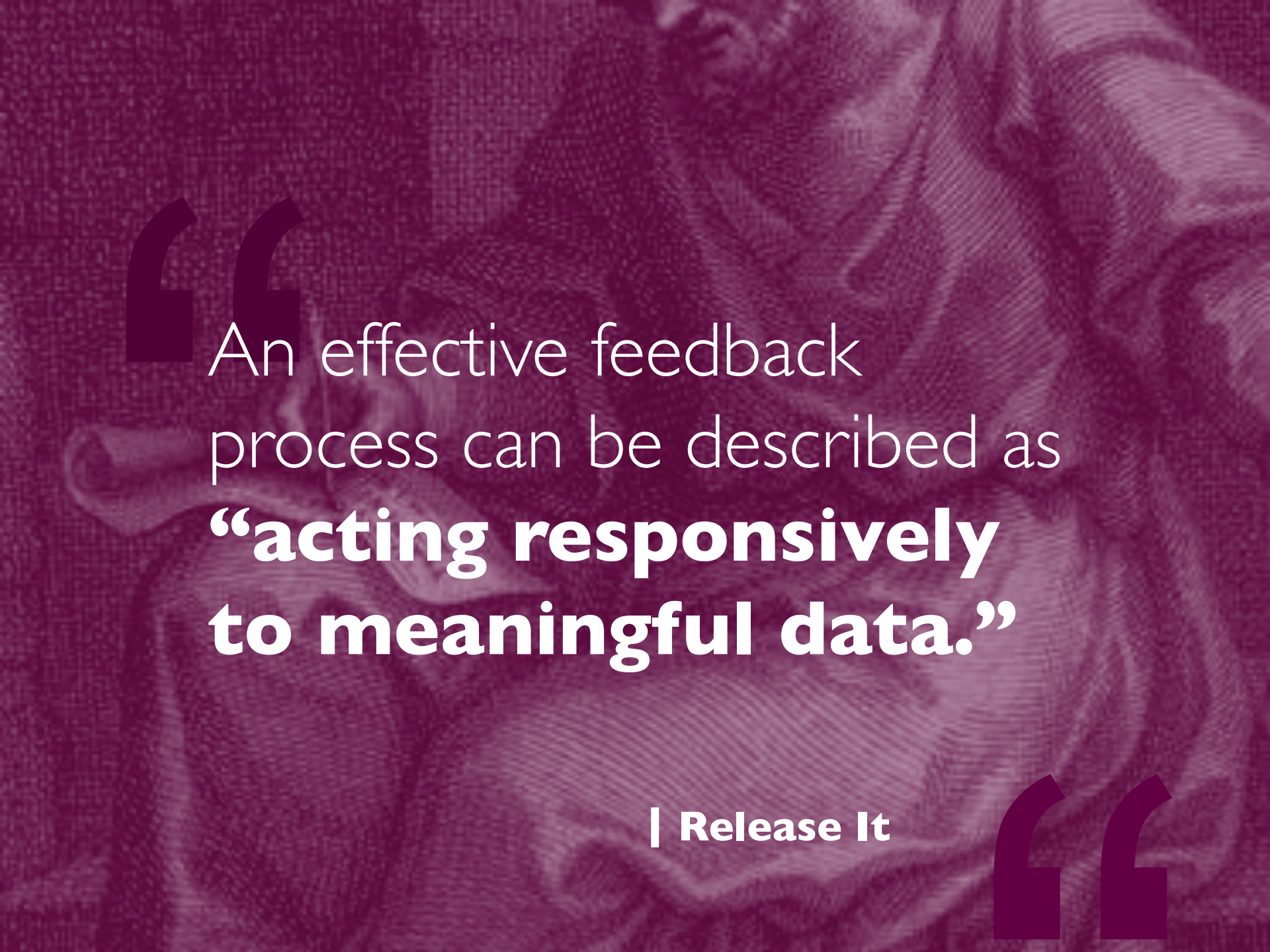
A collection of laboratory glassware including Erlenmeyer flasks, test tubes, a graduated cylinder, a beaker, and a petri dish, all containing liquids of varying colors (clear, yellow, red, blue). The glassware is arranged on a reflective surface.

5.

repetition

feedback formula

1. assumptions
2. clear objective
3. careful design
4. learning
5. repetition

A person with a beard, wearing a plaid shirt, is looking at a laptop screen. The image is overlaid with a dark purple gradient.

“

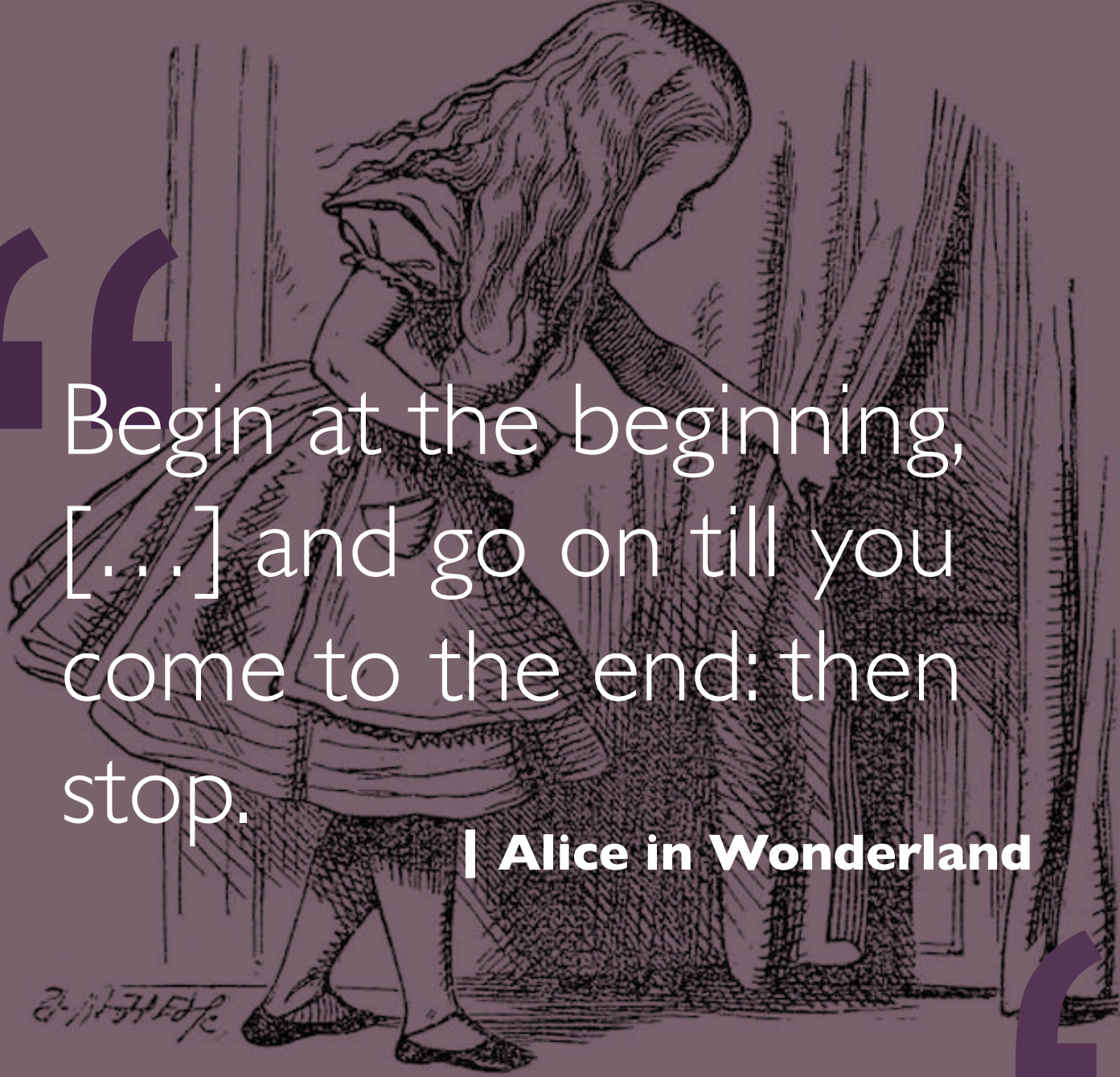
An effective feedback
process can be described as
**“acting responsively
to meaningful data.”**

| Release It

”



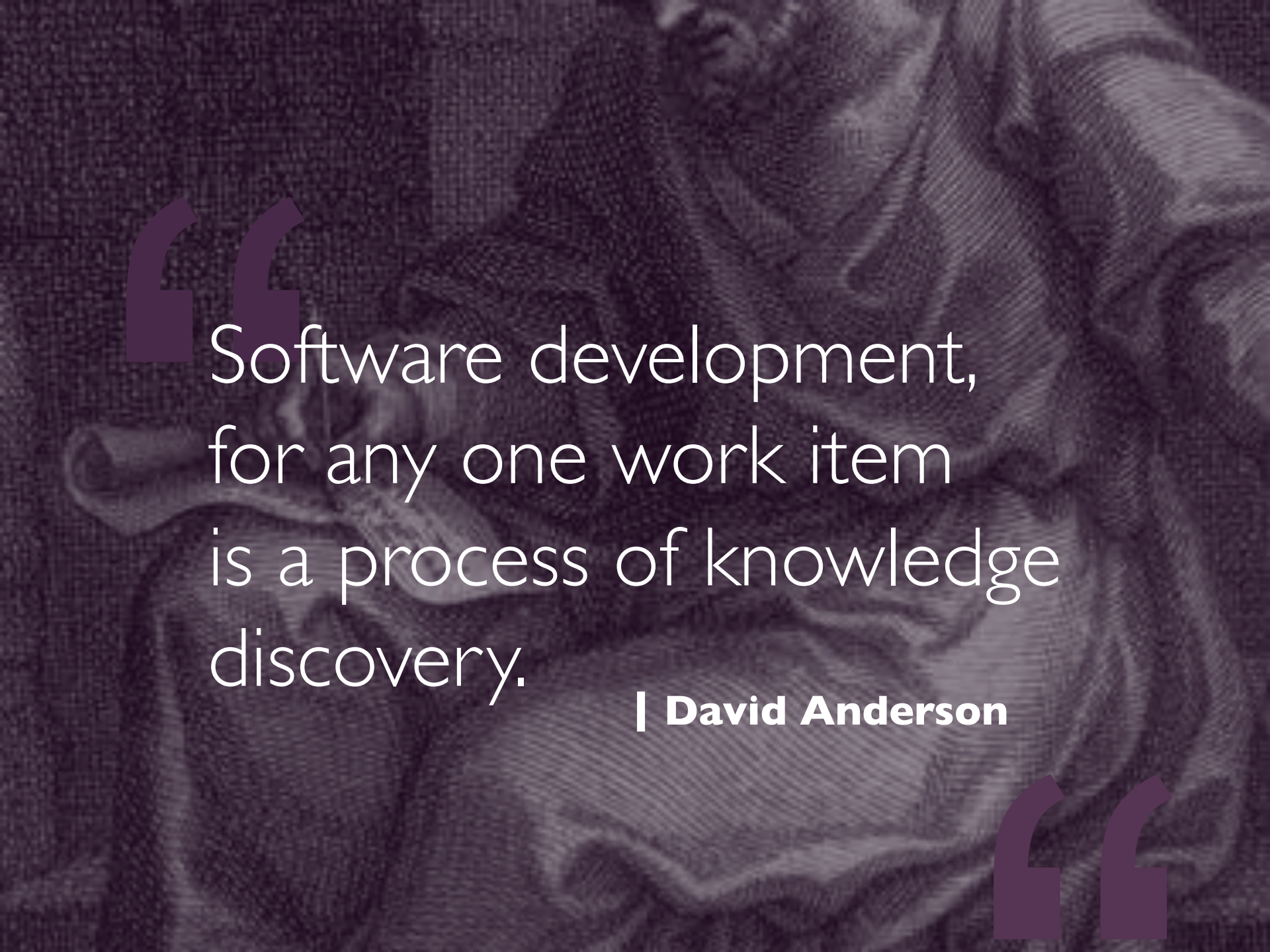
**Feedback
in SD**



“Begin at the beginning,
[...] and go on till you
come to the end: then
stop.

| **Alice in Wonderland**

”

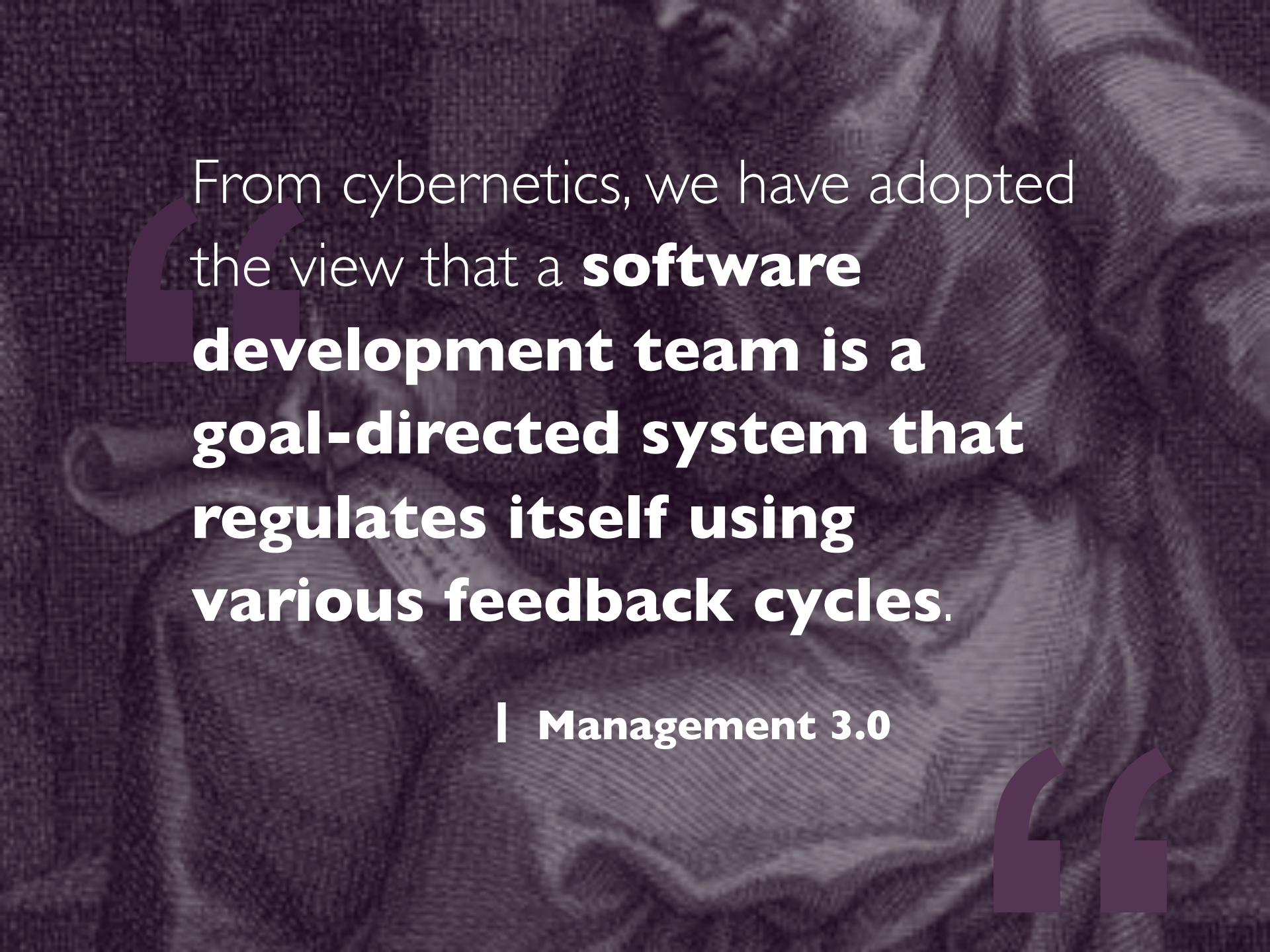


“

Software development,
for any one work item
is a process of knowledge
discovery.

| **David Anderson**

”



From cybernetics, we have adopted
the view that a **software
development team is a
goal-directed system that
regulates itself using
various feedback cycles.**

| **Management 3.0**

feedback about the **product**

feedback about the **process**

agile
is naturally
full of cycles

iterations



iterative development

iterations
produce
increments

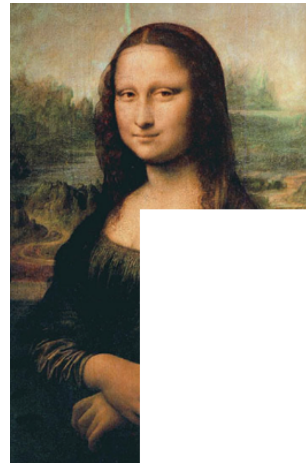
Incremental development



1



2



3



using feedback
requires

iterative development

Iterative development



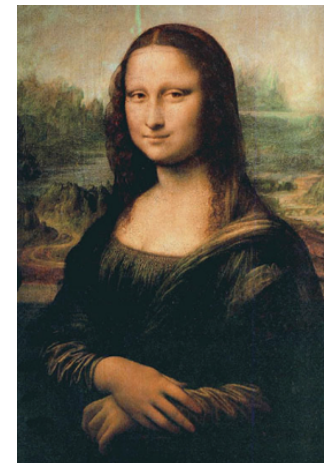
1



2



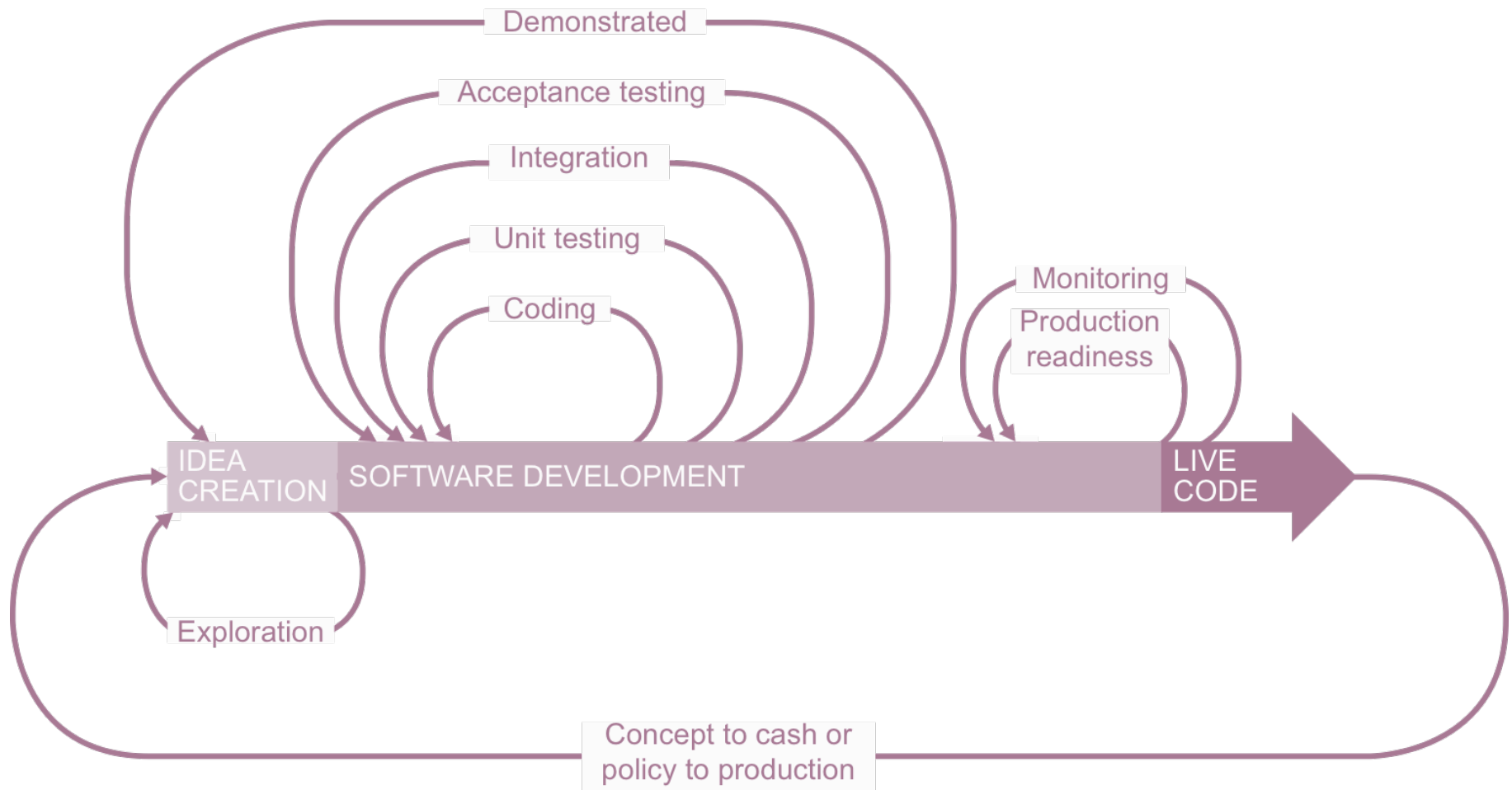
3

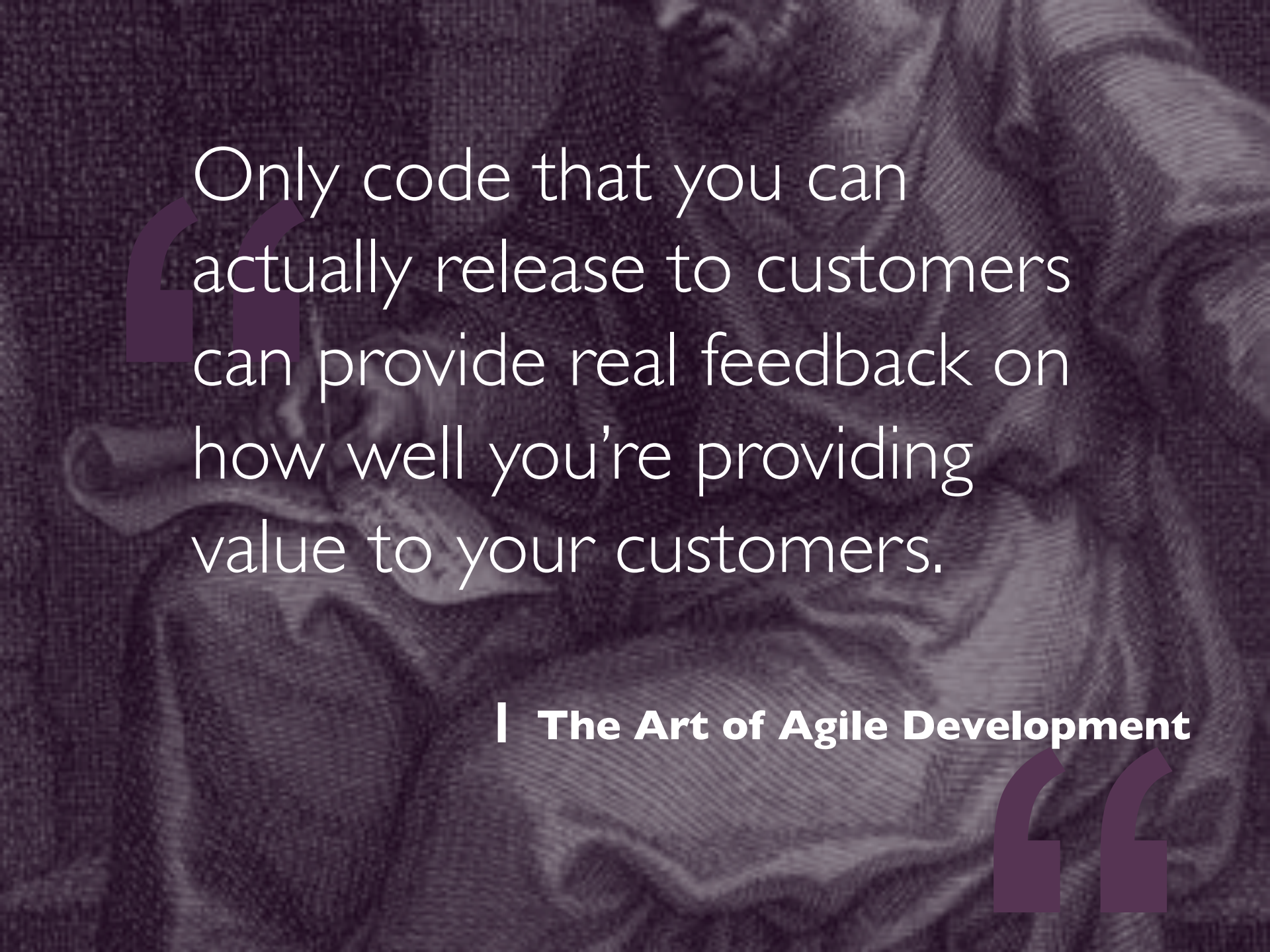


Iterative development
means doing
the same thing
more than once

feedback about the **product**

Loops in software development



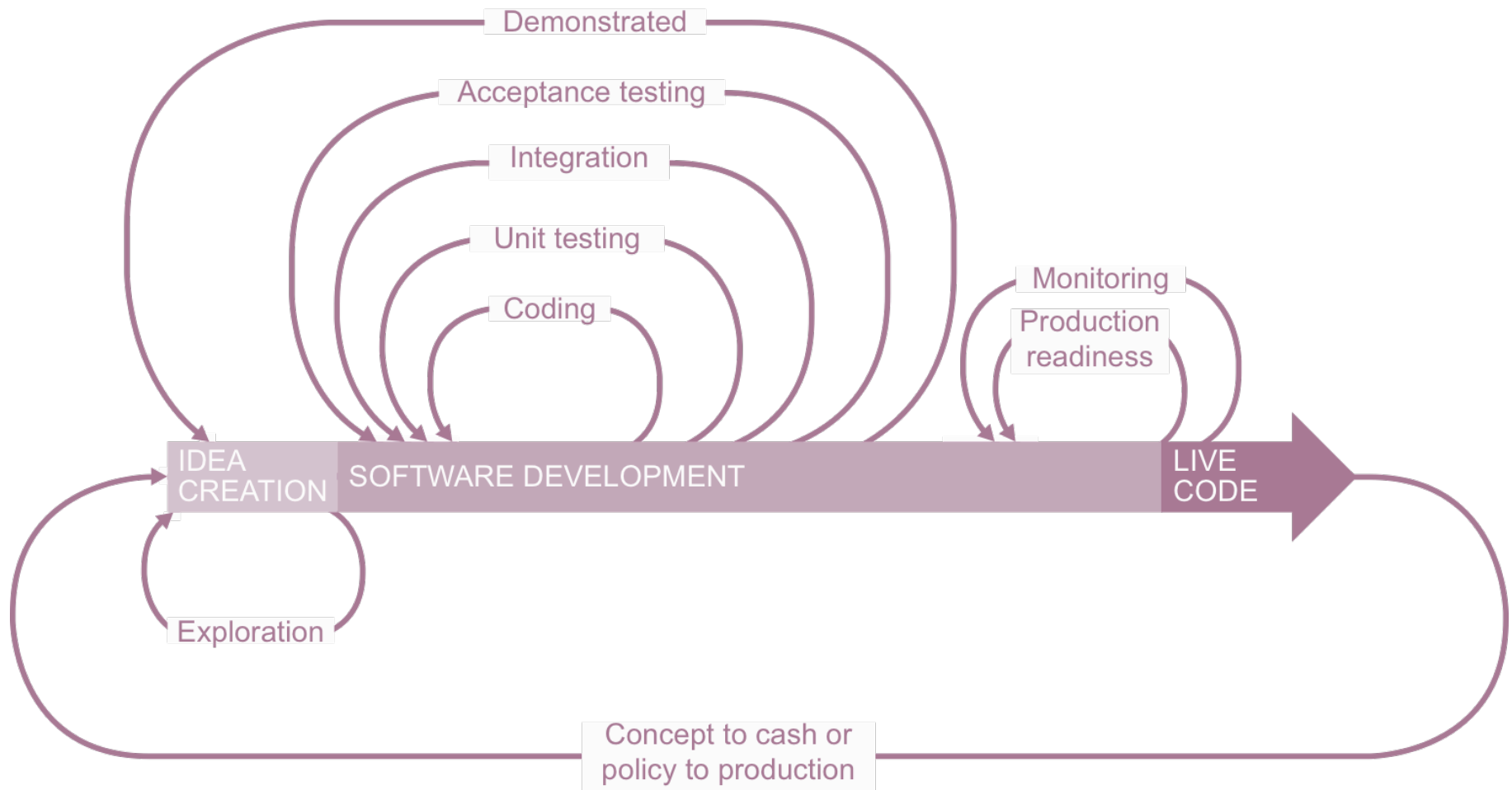


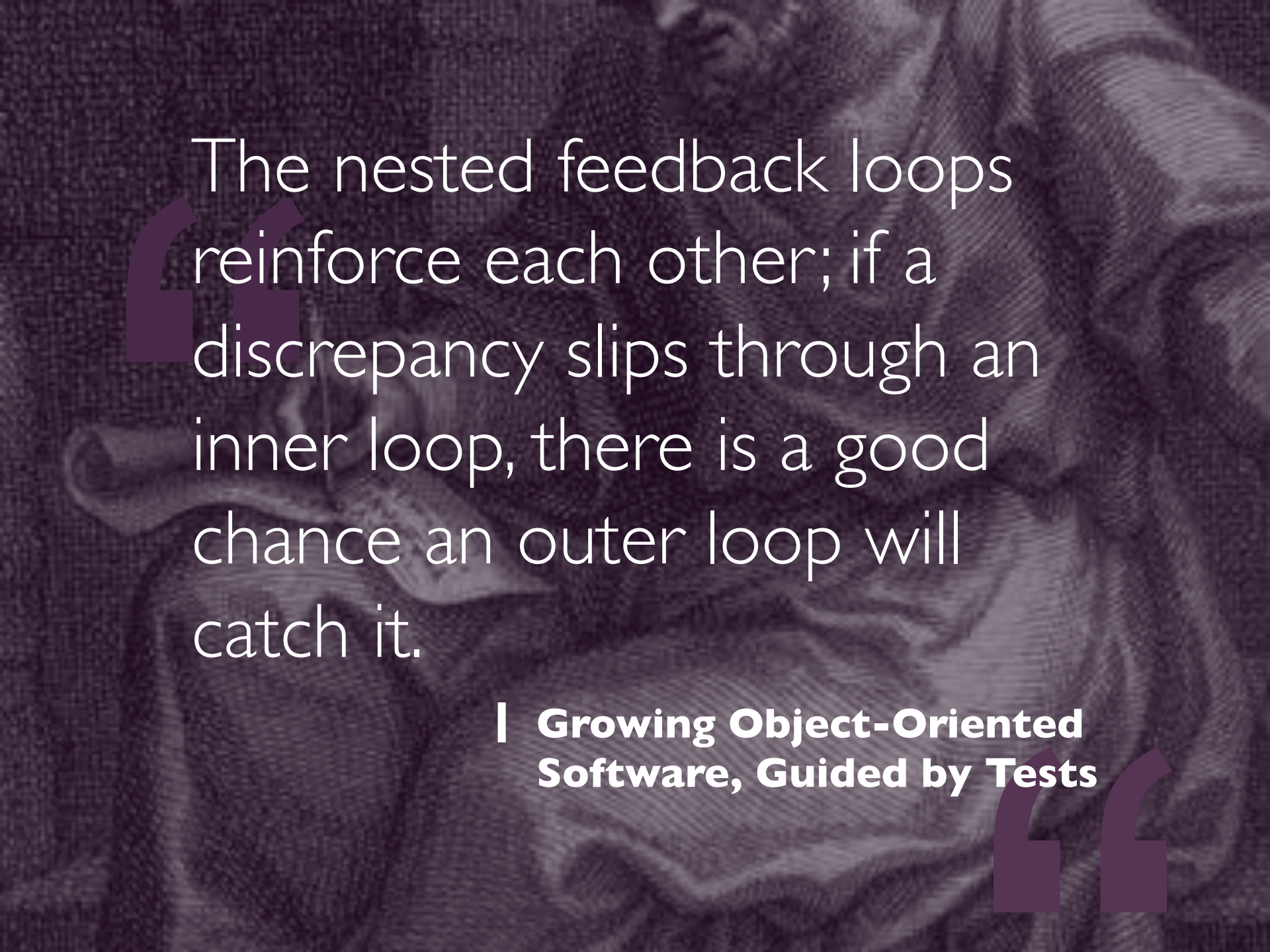
“Only code that you can
actually release to customers
can provide real feedback on
how well you’re providing
value to your customers.”

| **The Art of Agile Development**



Loops in software development





The nested feedback loops reinforce each other; if a discrepancy slips through an inner loop, there is a good chance an outer loop will catch it.

| **Growing Object-Oriented Software, Guided by Tests**

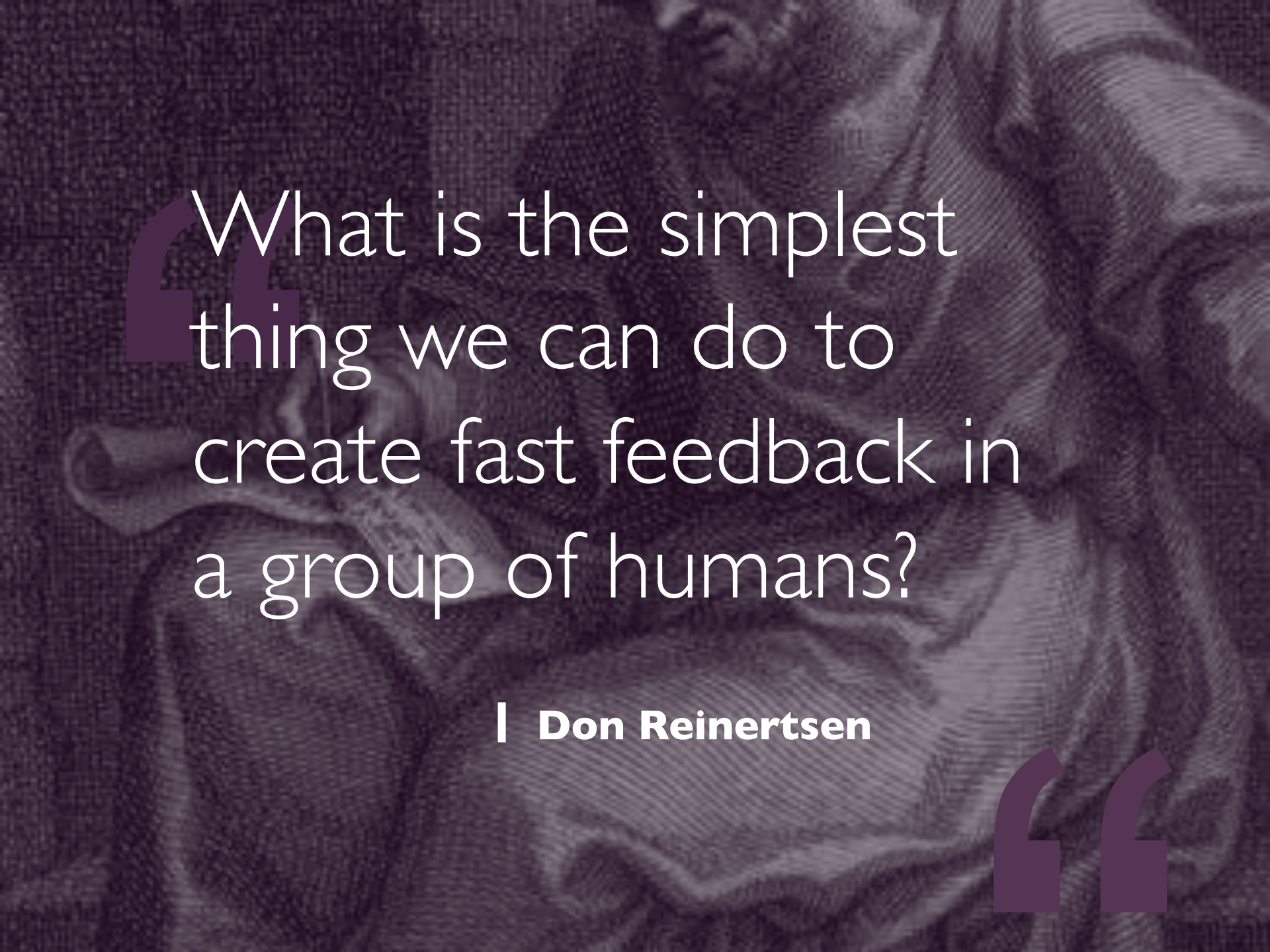
feedback about our **process**



“Everyone involved in a software project has to learn as it progresses.”

| **Growing Object-Oriented Software, Guided by Tests**





“What is the simplest
thing we can do to
create fast feedback in
a group of humans?”

| **Don Reinertsen**

”



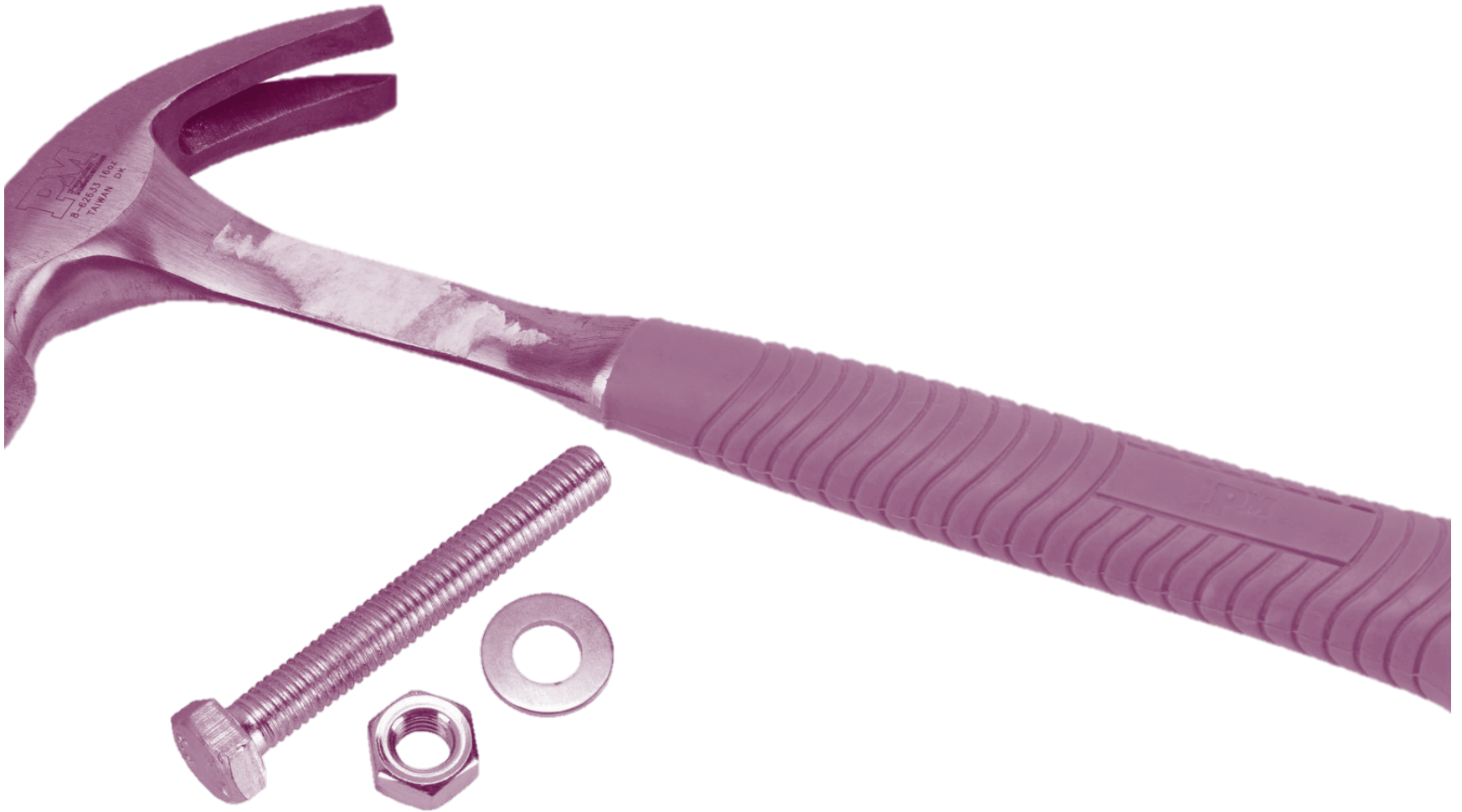
“

Increase face-to-face
communications.

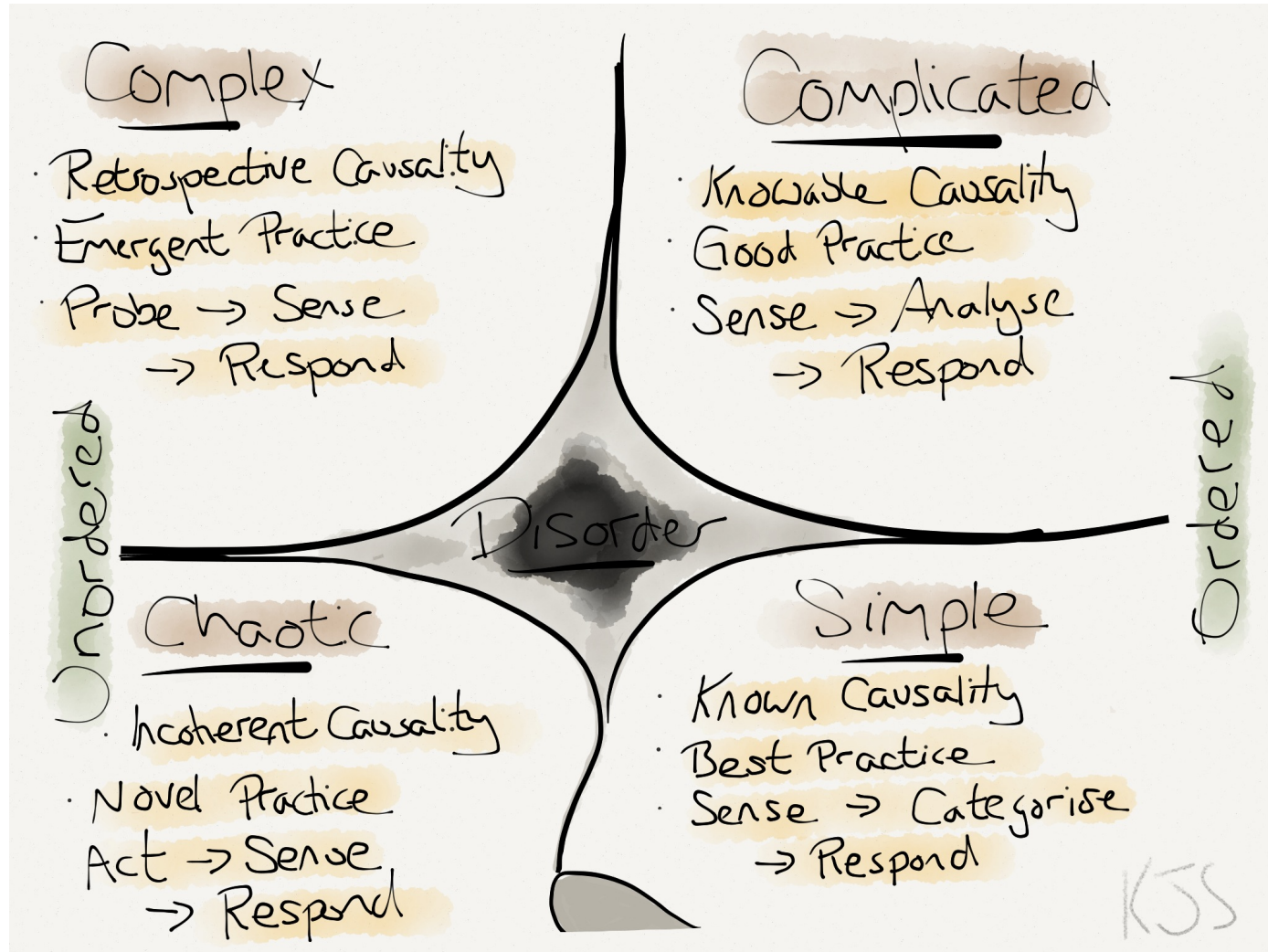
| **Don Reinertsen**

”

The right tool?



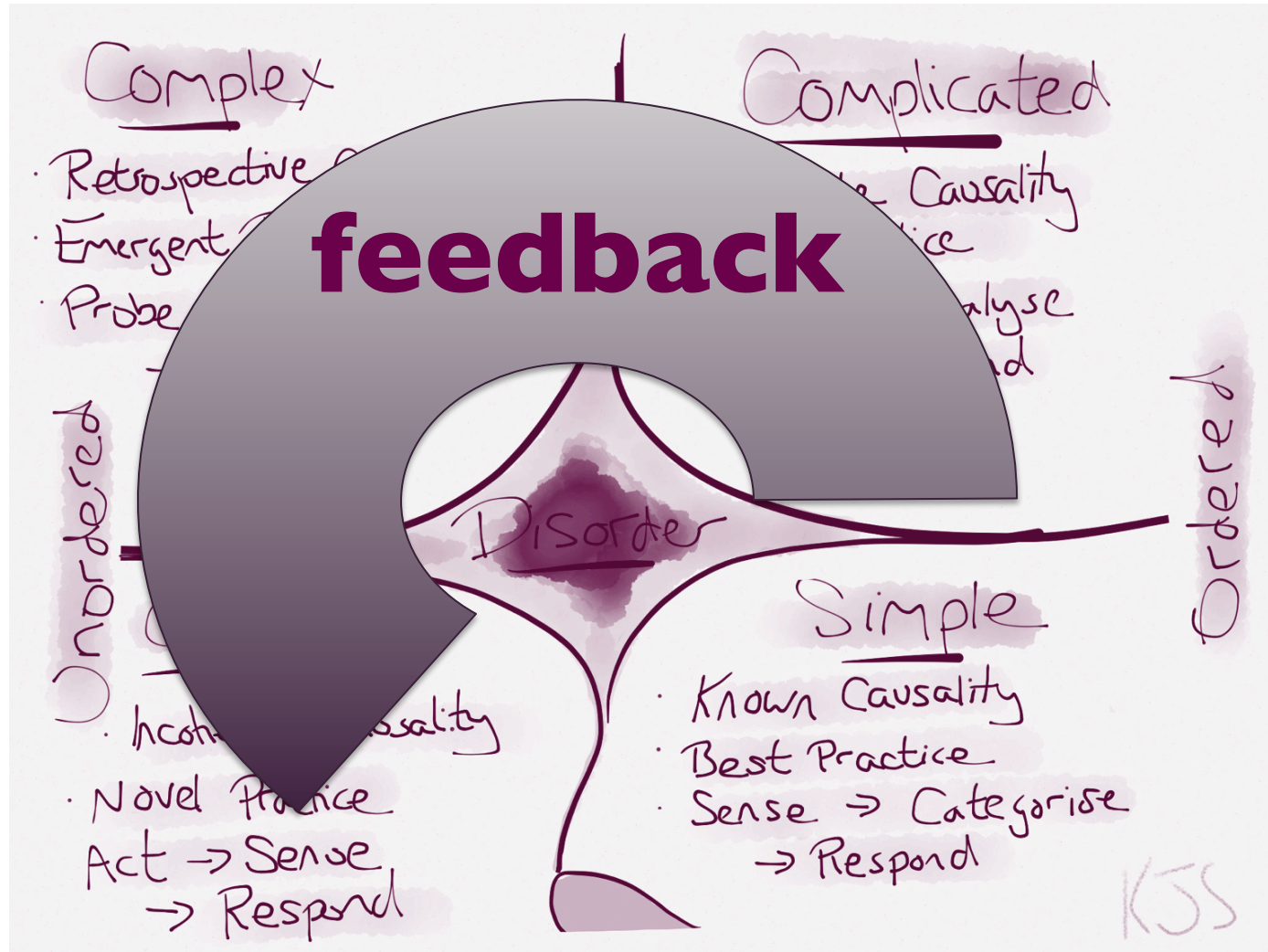
Cynefin



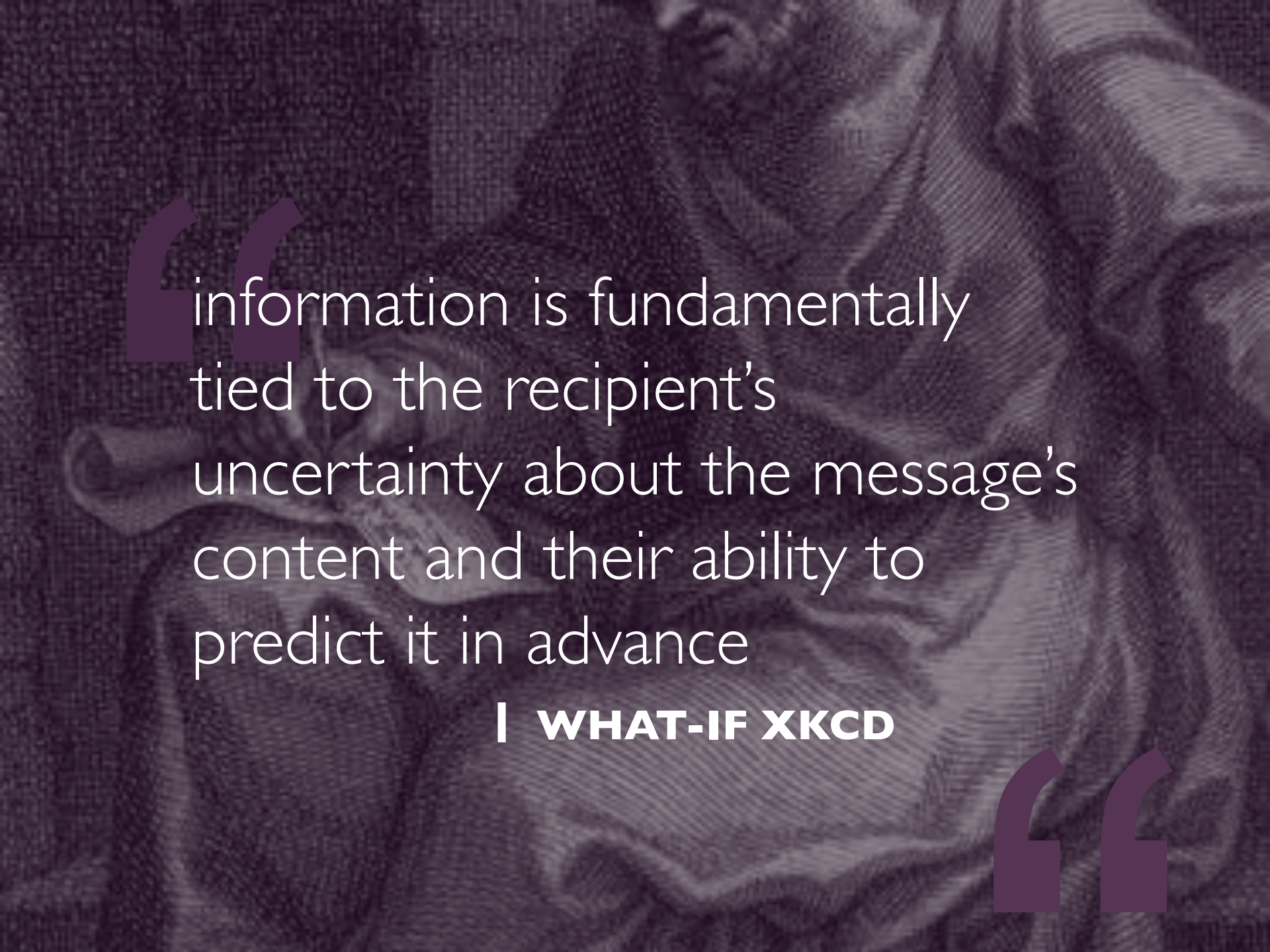
Cynefin simplified

1. Just about everyone in the world has done this.
2. Lots of people have done this, including several people in the company.
3. Someone in our company has done this.
4. Someone in the world did this, but not in our organization (and probably at a competitor).
5. Nobody has ever done this before.

Cynefin



Don't use feedback
if it's already
a **know problem**



“information is fundamentally
tied to the recipient’s
uncertainty about the message’s
content and their ability to
predict it in advance

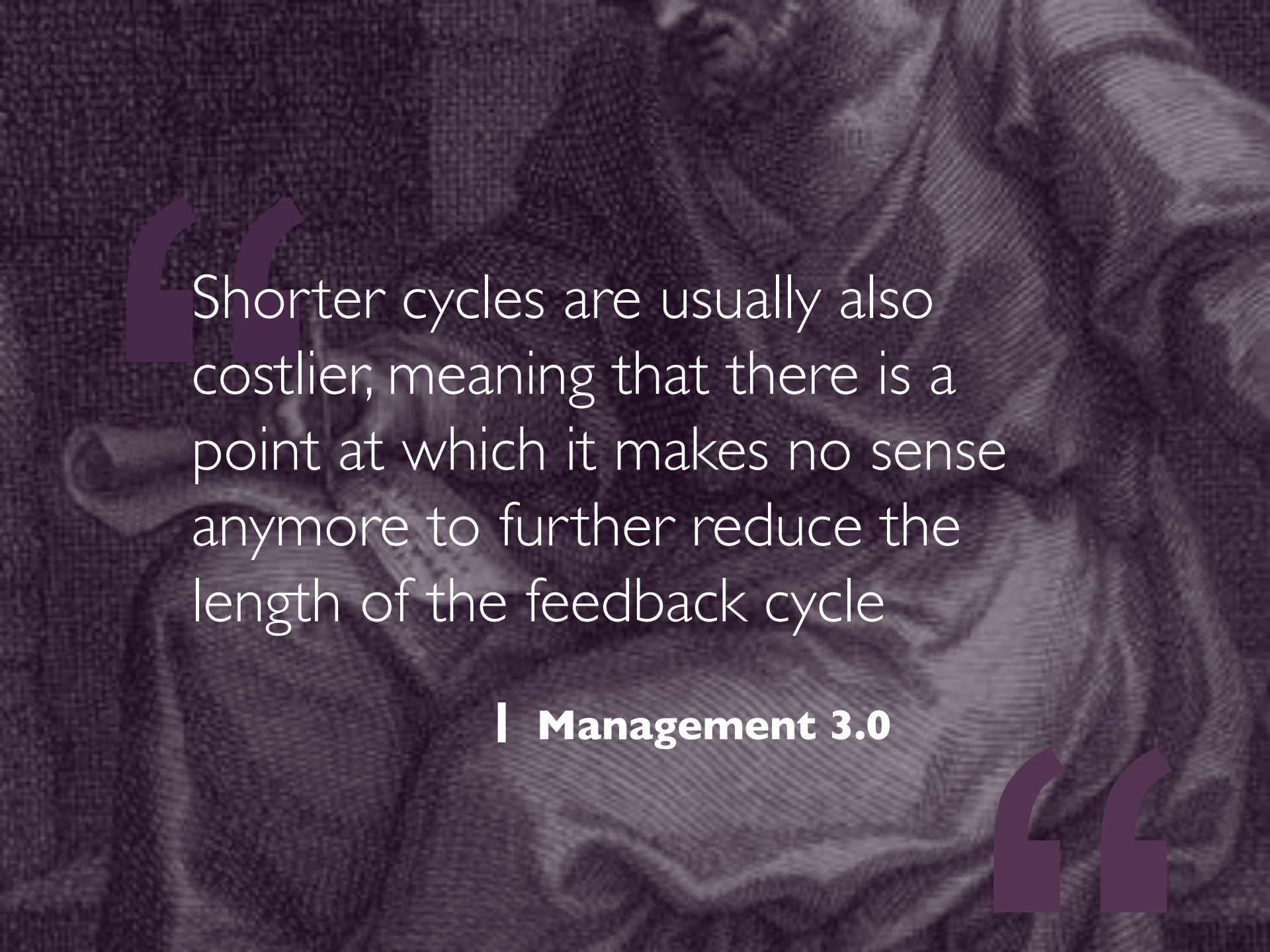
| **WHAT-IF XKCD**



**nearly
there**



cost of feedback

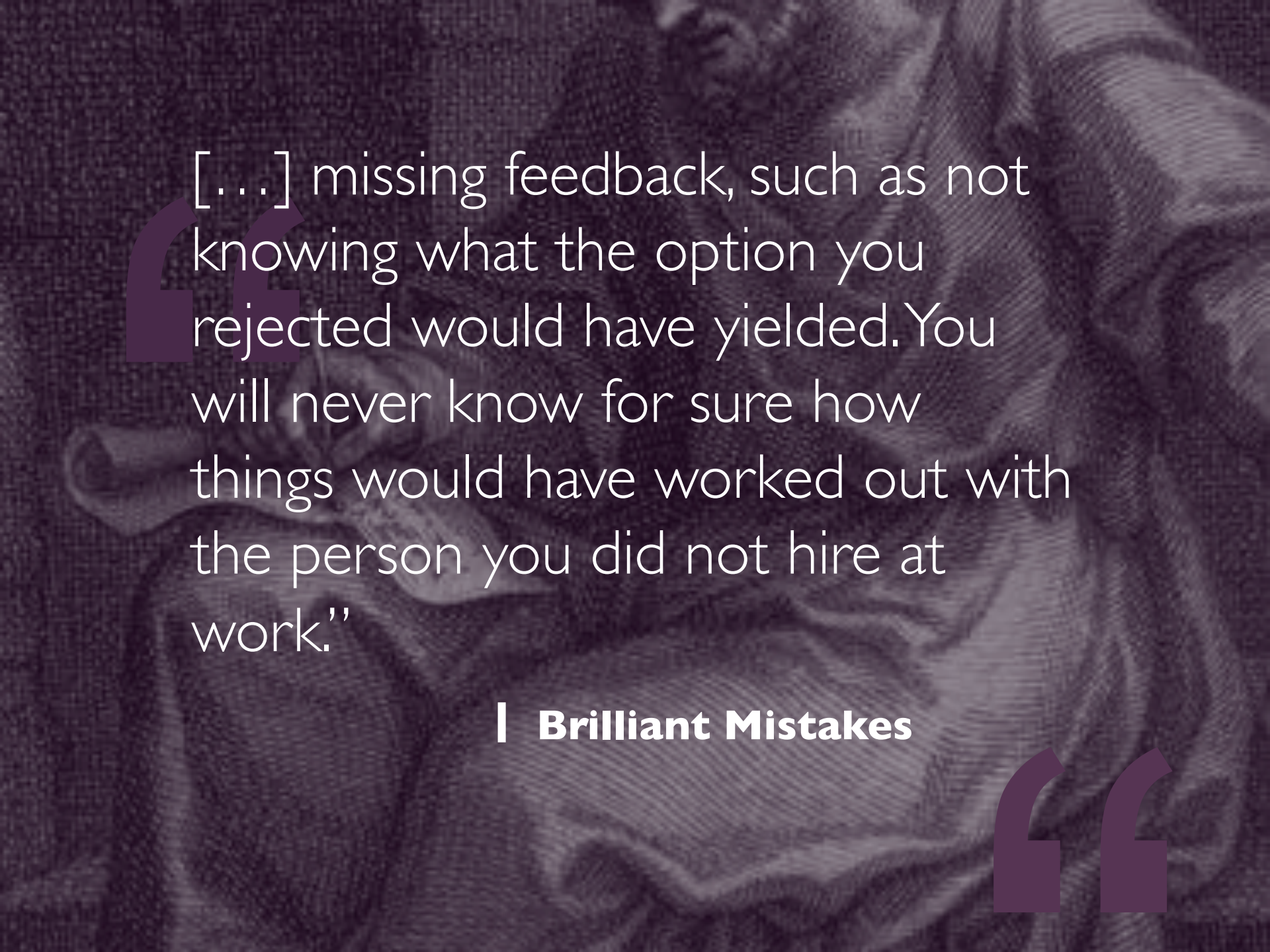


“Shorter cycles are usually also costlier, meaning that there is a point at which it makes no sense anymore to further reduce the length of the feedback cycle

| **Management 3.0**



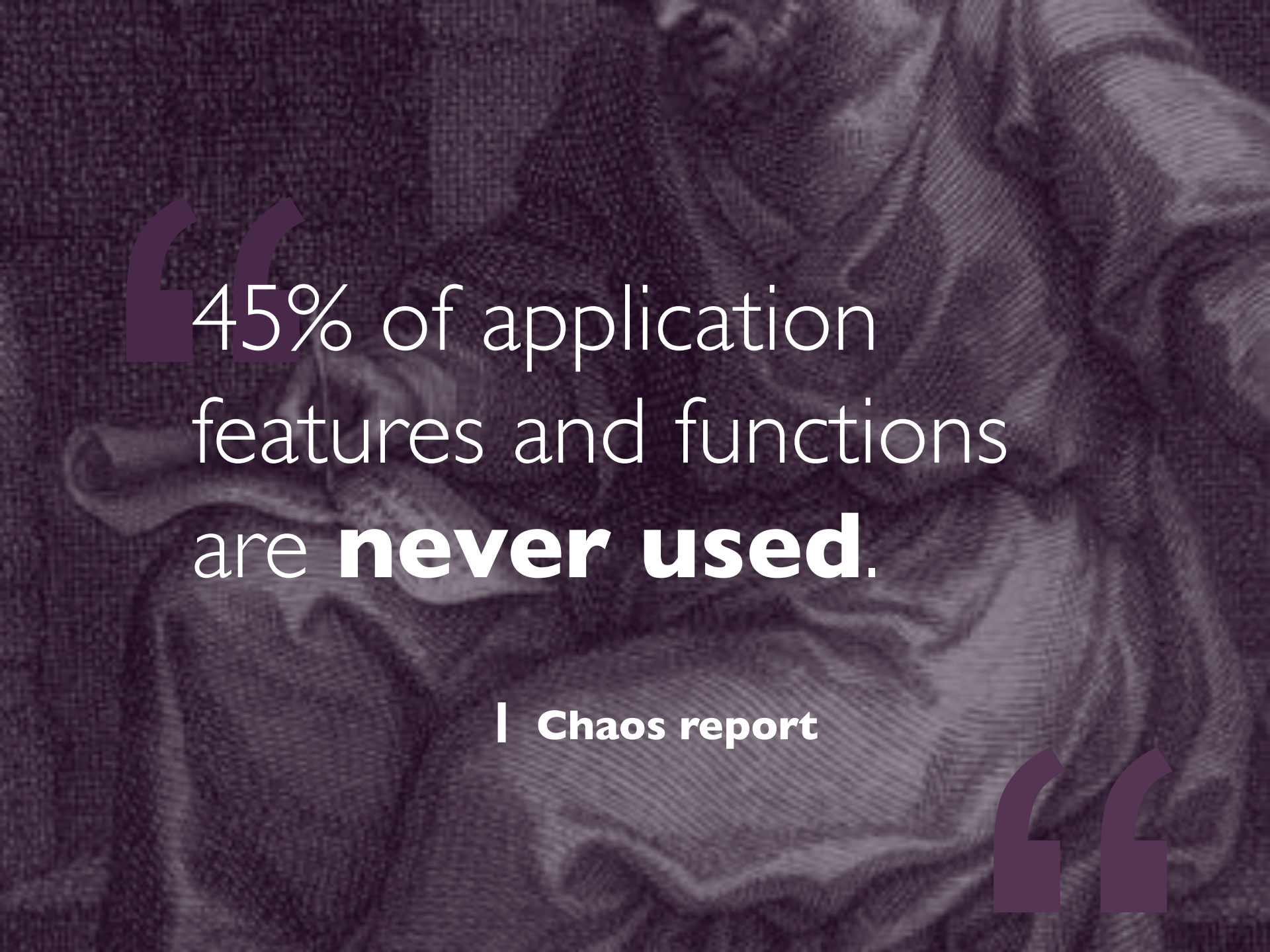
missing feedback



[...] missing feedback, such as not knowing what the option you rejected would have yielded. You will never know for sure how things would have worked out with the person you did not hire at work.”

| **Brilliant Mistakes**



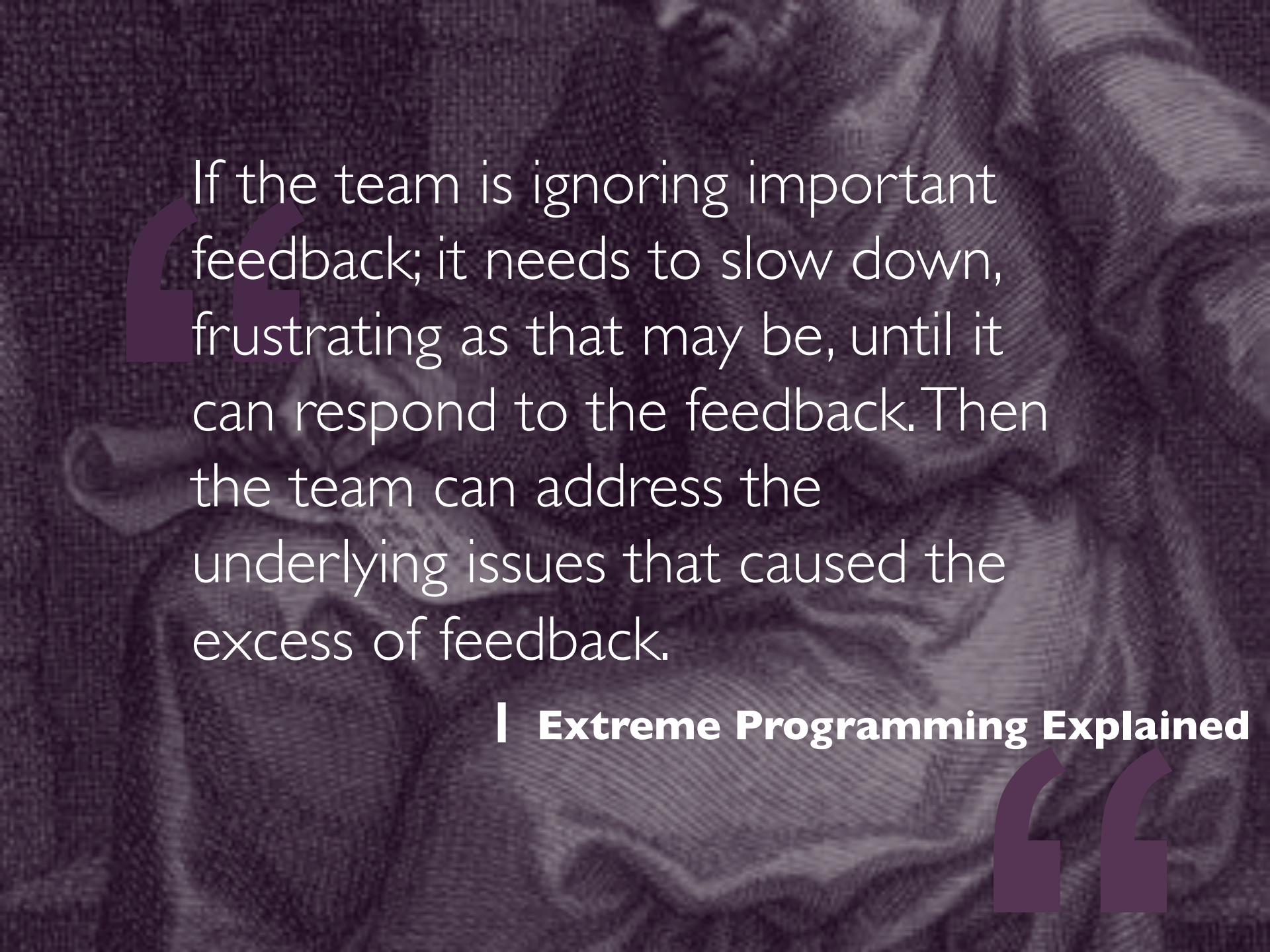


“45% of application
features and functions
are **never used**.”

| **Chaos report**



feedback overload



“If the team is ignoring important feedback; it needs to slow down, frustrating as that may be, until it can respond to the feedback. Then the team can address the underlying issues that caused the excess of feedback.”

| **Extreme Programming Explained**



So, how's this for feedback?



What did we learn?



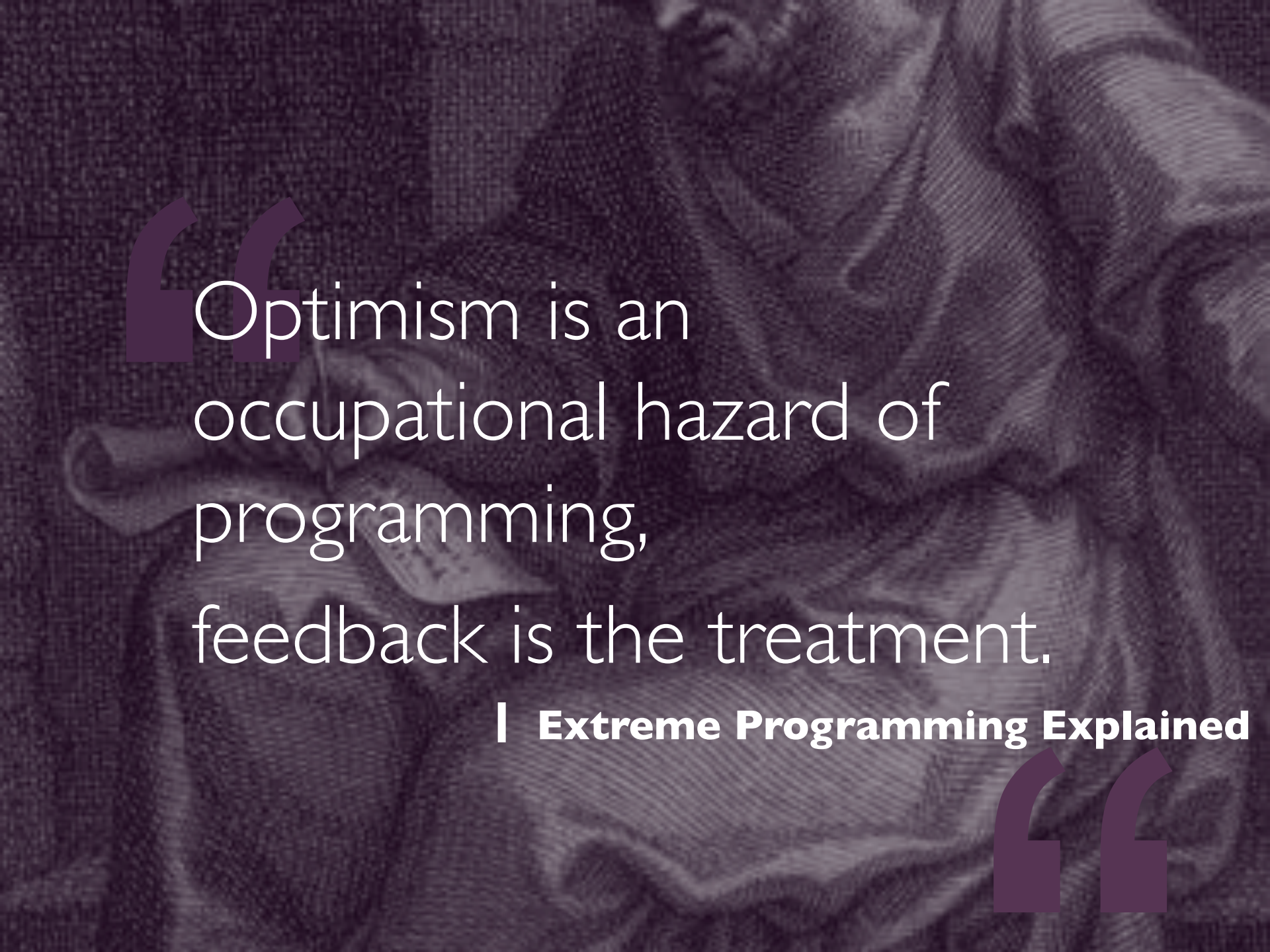
I guess we learnt
not to
do it again...

I guess we learnt

not to

do it again...

**use feedback deliberately
to learn more!**

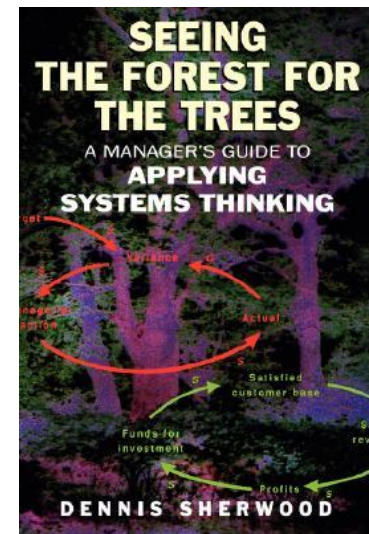
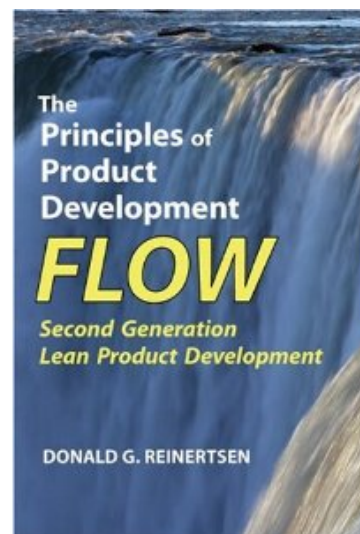
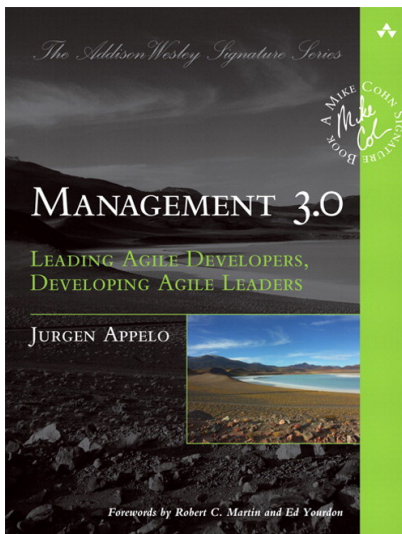
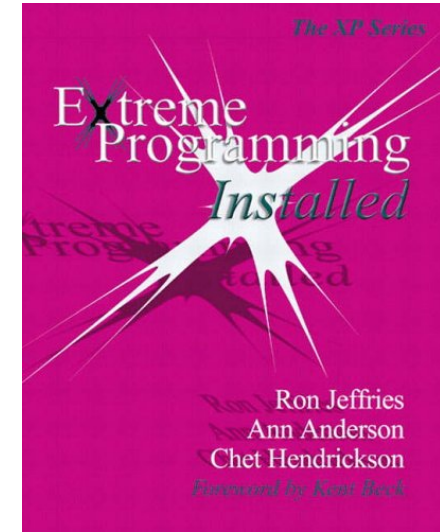
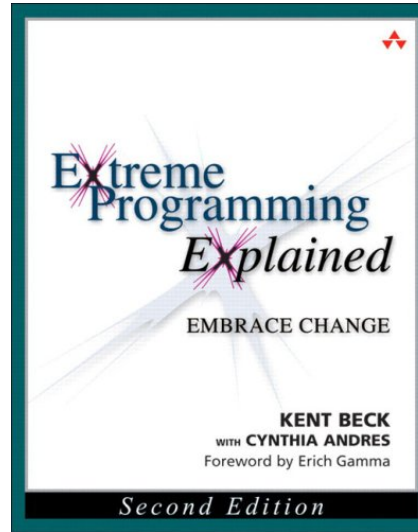
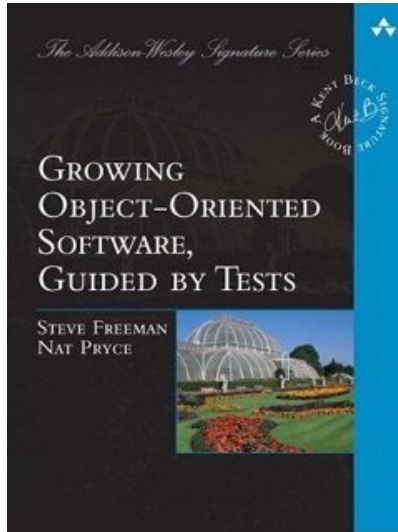


“Optimism is an
occupational hazard of
programming,
feedback is the treatment.

| **Extreme Programming Explained**



Further reading



The Remember Slide

- feedback is a learning tool
- it requires cycles
- loops can be nested and concurrent
- the feedback recipe:

**assumptions, objective,
design, learning, repetition**

So **Thank You**



License



This presentation is provided under the Creative Commons
Attribution - Share Alike 3.0 Unported License

You are free



to Share — to copy, distribute and transmit the work



to Remix — to adapt the work

Under the following conditions



Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).



Share Alike — If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.