The Agile Architect

Oxymoron or Savior?

Howard Deiner

Agilist. Pragmatist. Craftsman.
Scrum. eXtreme Programming.
Consultant. Coach. Instructor.

+1 (203) 243-1515

howard.deiner@deinersoft.com
http://www.deinersoft.com

Software Architect 2011

Mommy, When I Grow Up, I Want To Be An Architect!

Mommy, When I Grow Up, I Want To Be An Architect!

Getting a title on a business card of "Architect" is exhilarating

Your company trusts you with money to develop your dreams

You are being told that you are smart and capable

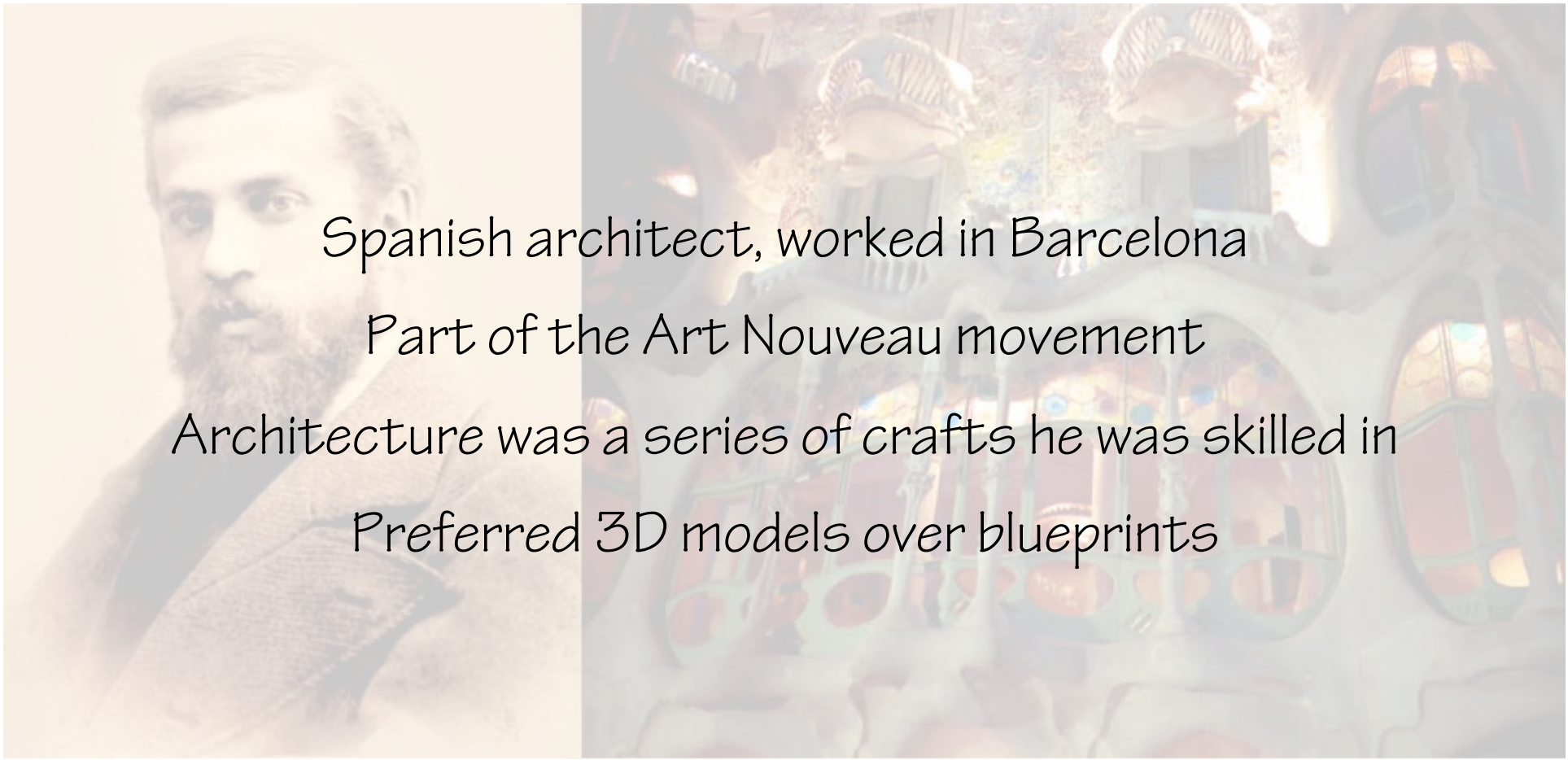
Quickly moves you up the Maslovian Scale!

Invention & creation – make things happen, not just talk about it

Be analytic as well as creative – beauty as well as practicality

Famous Architect #1 - Antonio Gaudi (1852-1926)

Famous Architect #1 - Antonio Gaudi (1852-1926)



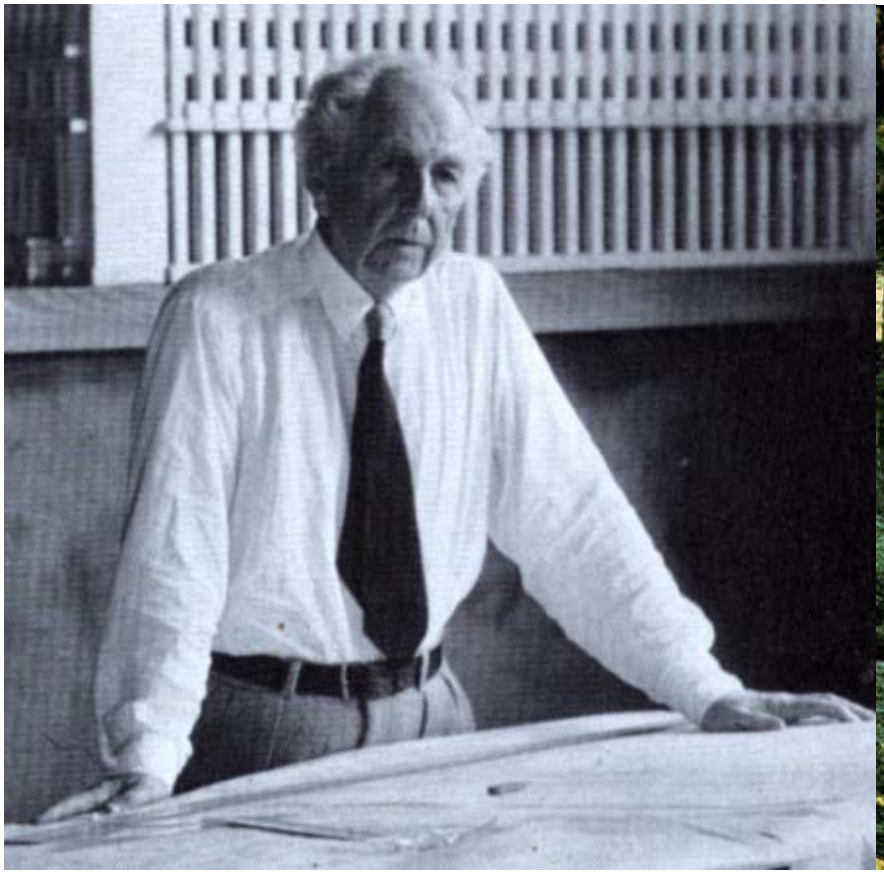
Spanish architect, worked in Barcelona

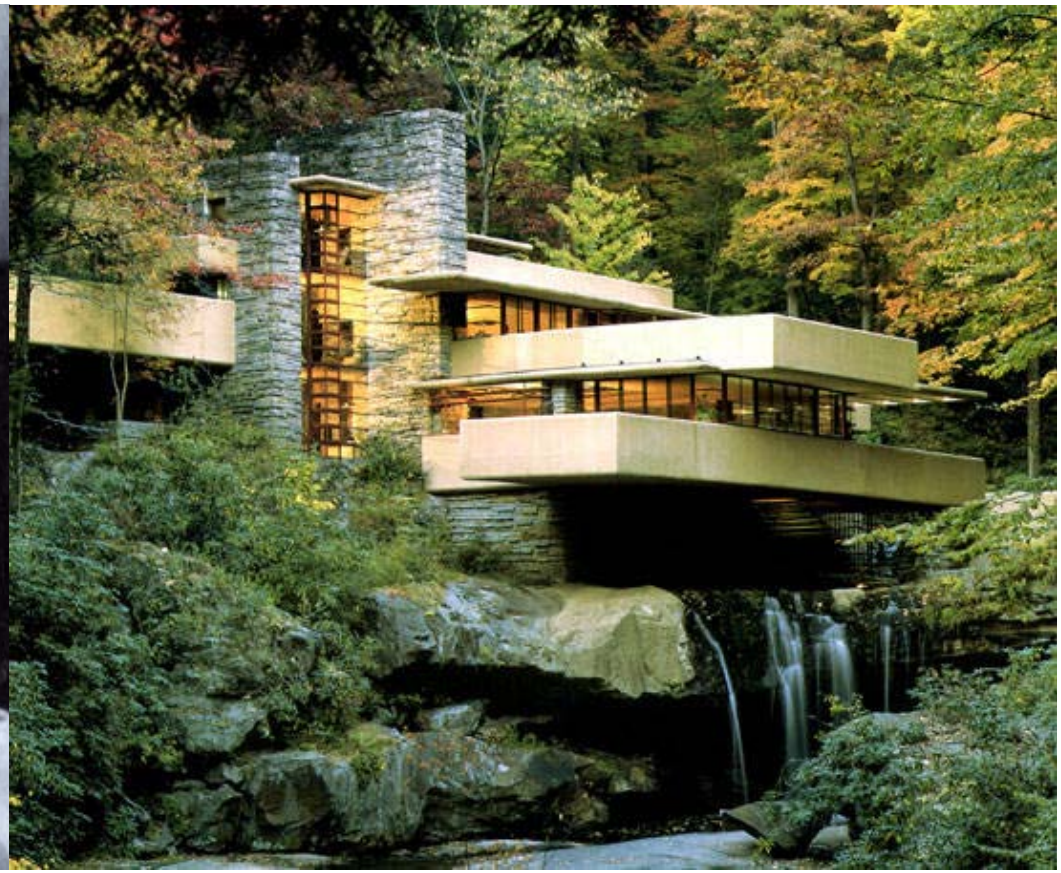
Part of the Art Nouveau movement

Architecture was a series of crafts he was skilled in

Preferred 3D models over blueprints

Famous Architect #2 - Frank Lloyd Wright (1867 - 1959)





Famous Architect #2 - Frank Lloyd Wright (1867 – 1959)

American architect, designed over 1,000 projects

Promoted organic architecture, exemplified by Fallingwater

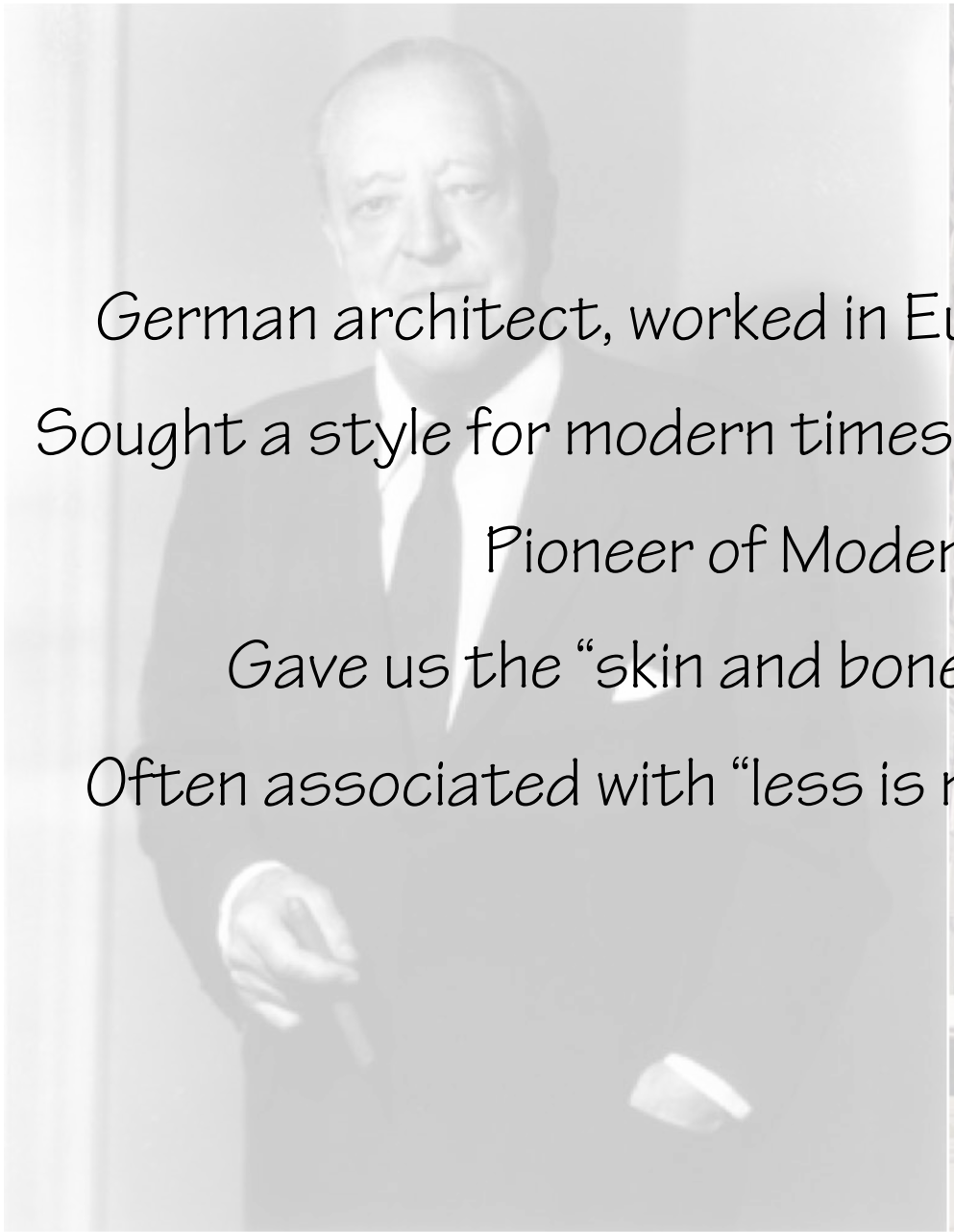
Leader of Prairie School movement – "Craftsman" homes

Developed Usonian home – distinctly American

Unencumbered by previous architectural conventions

Famous Architect #3 - Ludwig Mies van der Rohe (1886 - 1969)

Famous Architect #3 - Ludwig Mies van der Rohe (1886 - 1969)



German architect, worked in Europe and the United States

Sought a style for modern times to replace classical and gothic

Pioneer of Modern architecture

Gave us the "skin and bones" style of architecture

Often associated with "less is more" & "God is in the details"

Um, Am I in the Wrong Conference?

Um, Am I in the Wrong Conference?

Worried? You and me both… ☺

There's a reason that software architects are called architects

All architects look to define themselves in their work

Out of passion, our greatest goals are realized

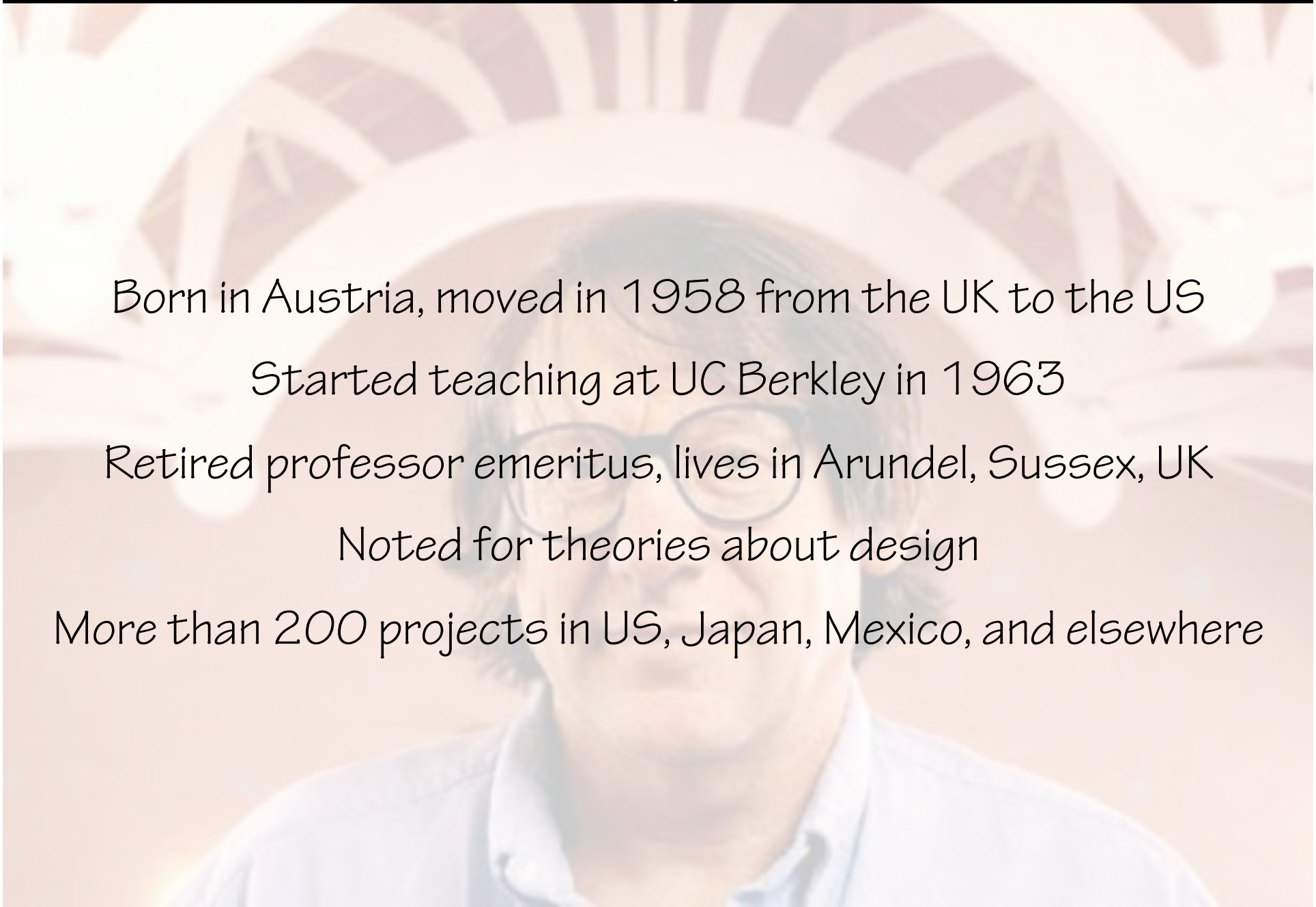
Architects must live in two worlds to be successful

The world of the what, why, and when

And the world of who and how

One More Architect - Christopher Alexander (born 1936)

One More Architect - Christopher Alexander (born 1936)



Born in Austria, moved in 1958 from the UK to the US

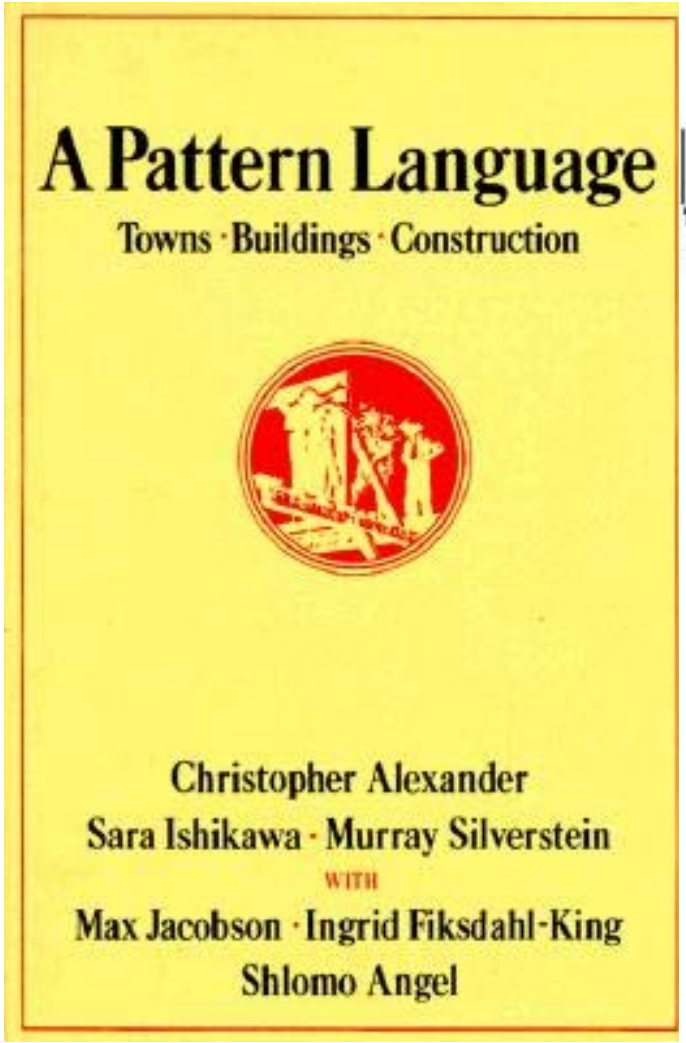
Started teaching at UC Berkley in 1963

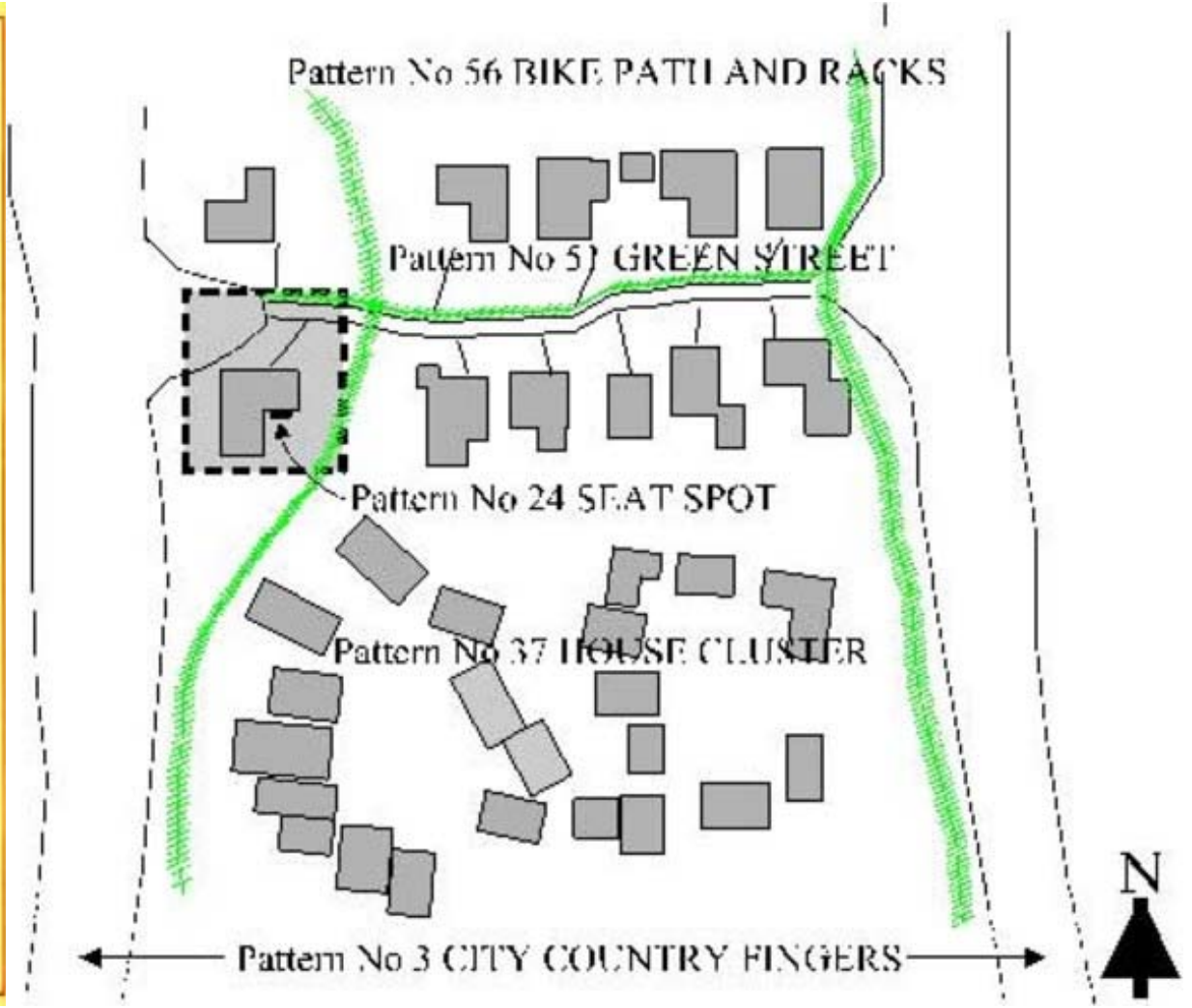
Retired professor emeritus, lives in Arundel, Sussex, UK

Noted for theories about design

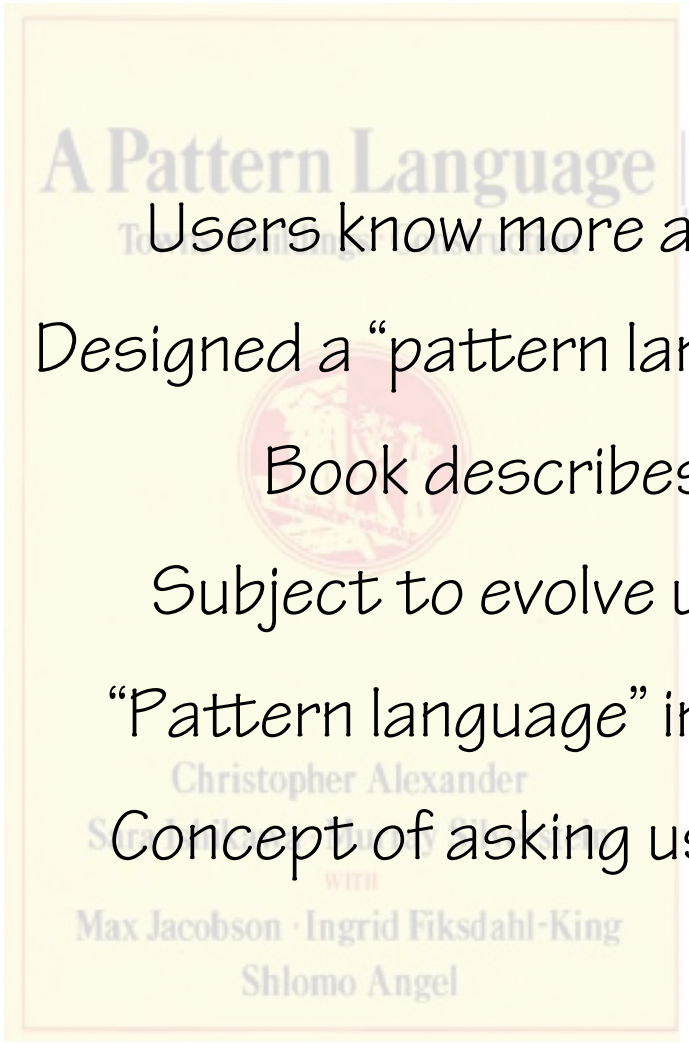
More than 200 projects in US, Japan, Mexico, and elsewhere

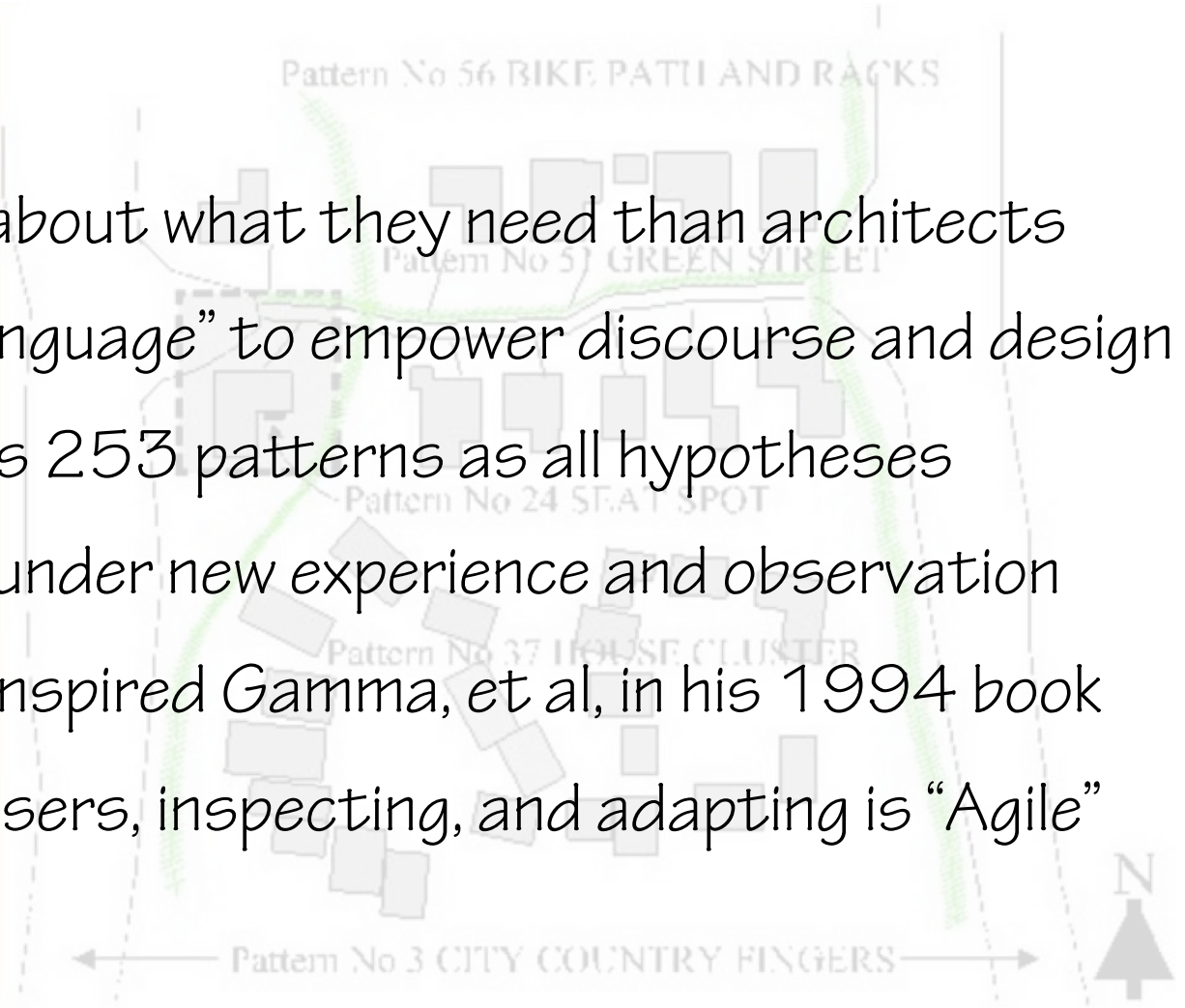
And the Book That Started A Lot of Modern Software Thinking





And the Book That Started A Lot of Modern Software Thinking





Users know more about what they need than architects

Designed a "pattern language" to empower discourse and design

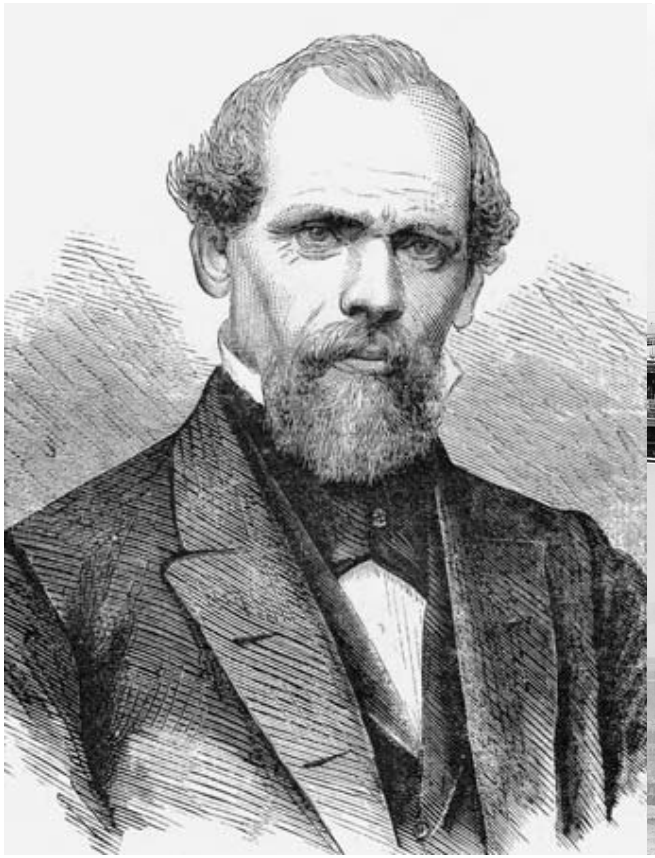
Book describes 253 patterns as all hypotheses

Subject to evolve under new experience and observation

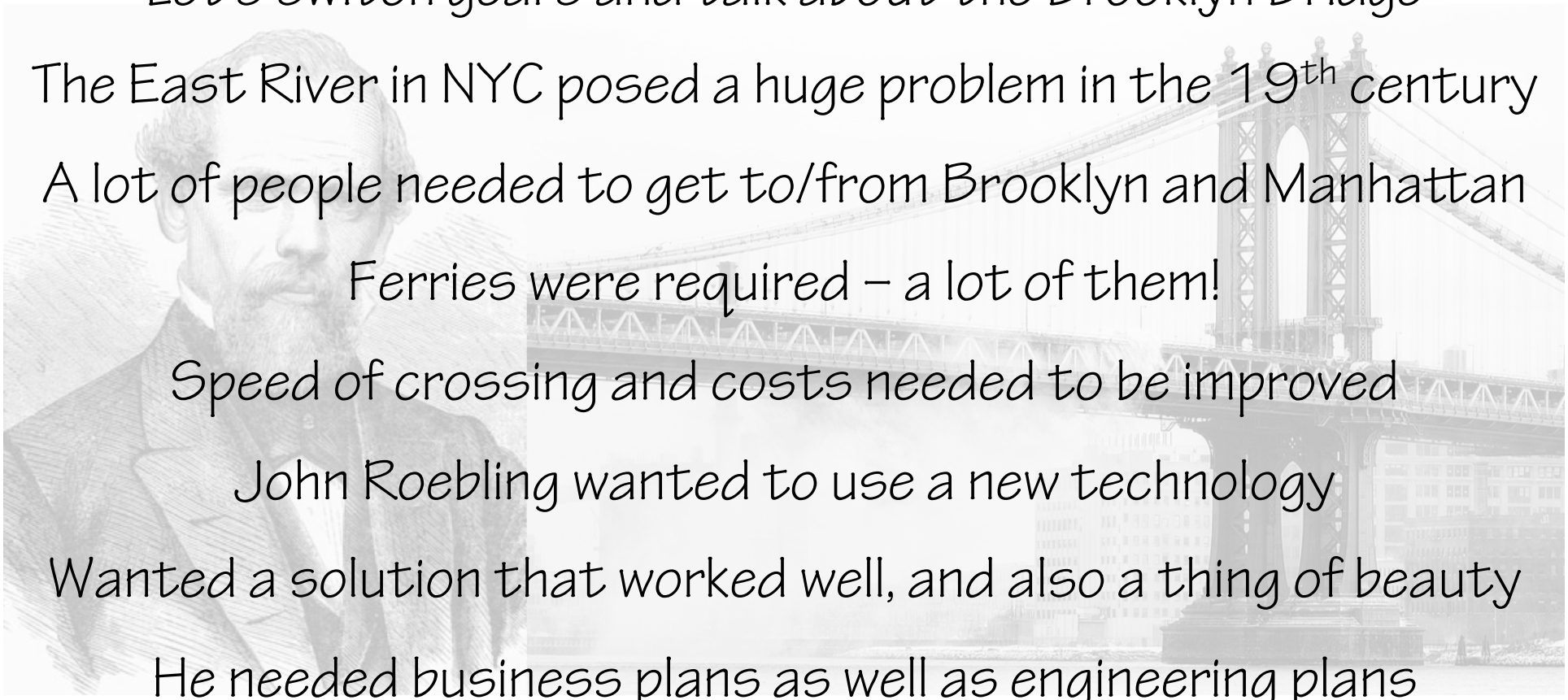
"Pattern language" inspired Gamma, et al, in his 1994 book

Concept of asking users, inspecting, and adapting is "Agile"

Building the Brooklyn Bridge



Building the Brooklyn Bridge



Let's switch gears and talk about the Brooklyn Bridge

The East River in NYC posed a huge problem in the 19th century

A lot of people needed to get to/from Brooklyn and Manhattan

Ferries were required – a lot of them!

Speed of crossing and costs needed to be improved

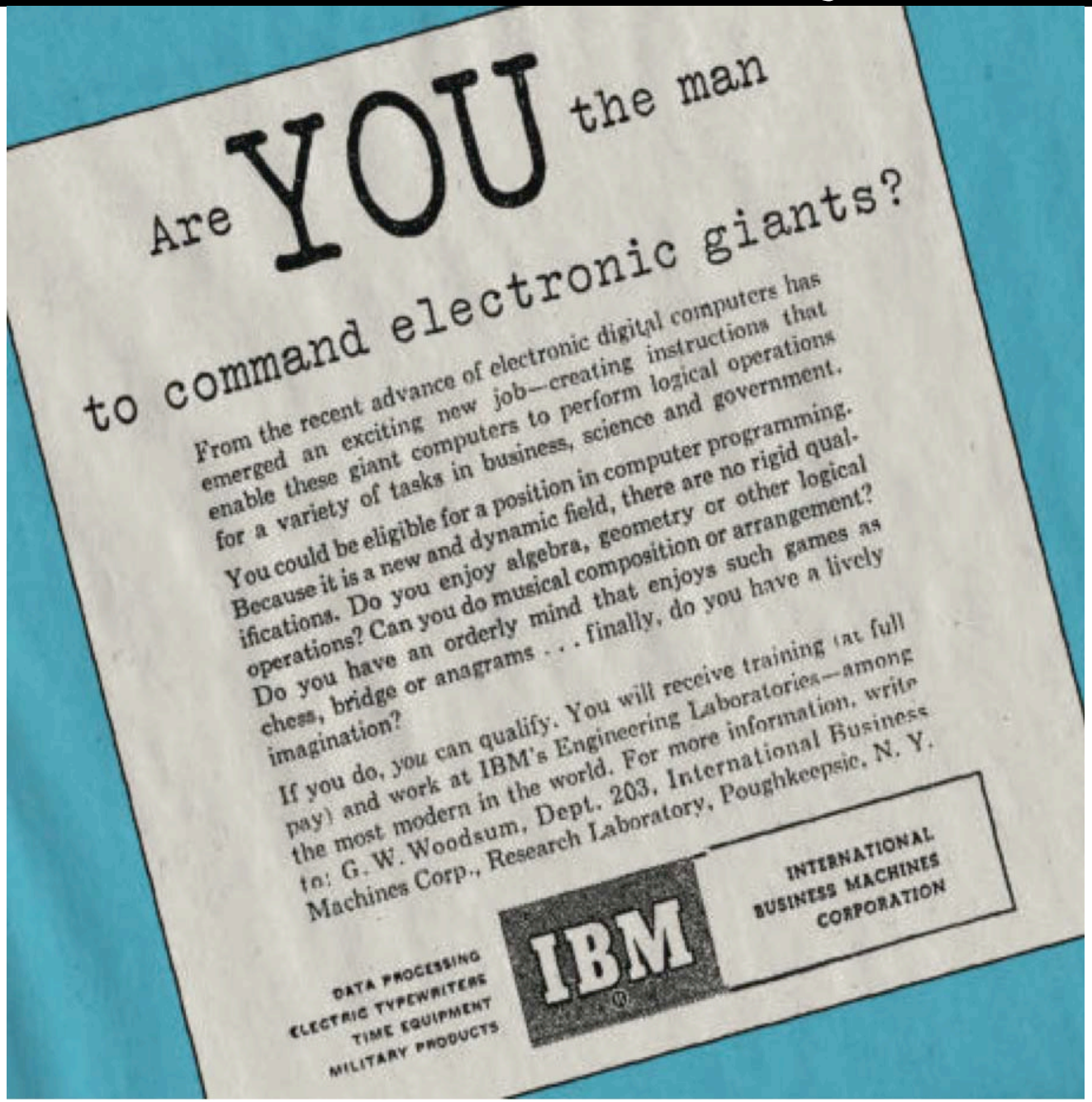
John Roebling wanted to use a new technology

Wanted a solution that worked well, and also a thing of beauty

He needed business plans as well as engineering plans

And both were subject to inspect and adapt

1956 - Are You Man Enough?



Are YOU the man

to command electronic giants?

From the recent advance of electronic digital computers has emerged an exciting new job—creating instructions that enable these giant computers to perform logical operations for a variety of tasks in business, science and government.

You could be eligible for a position in computer programming. Because it is a new and dynamic field, there are no rigid qualifications. Do you enjoy algebra, geometry or other logical operations? Can you do musical composition or arrangement? Do you have an orderly mind that enjoys such games as chess, bridge or anagrams . . . finally, do you have a lively imagination?

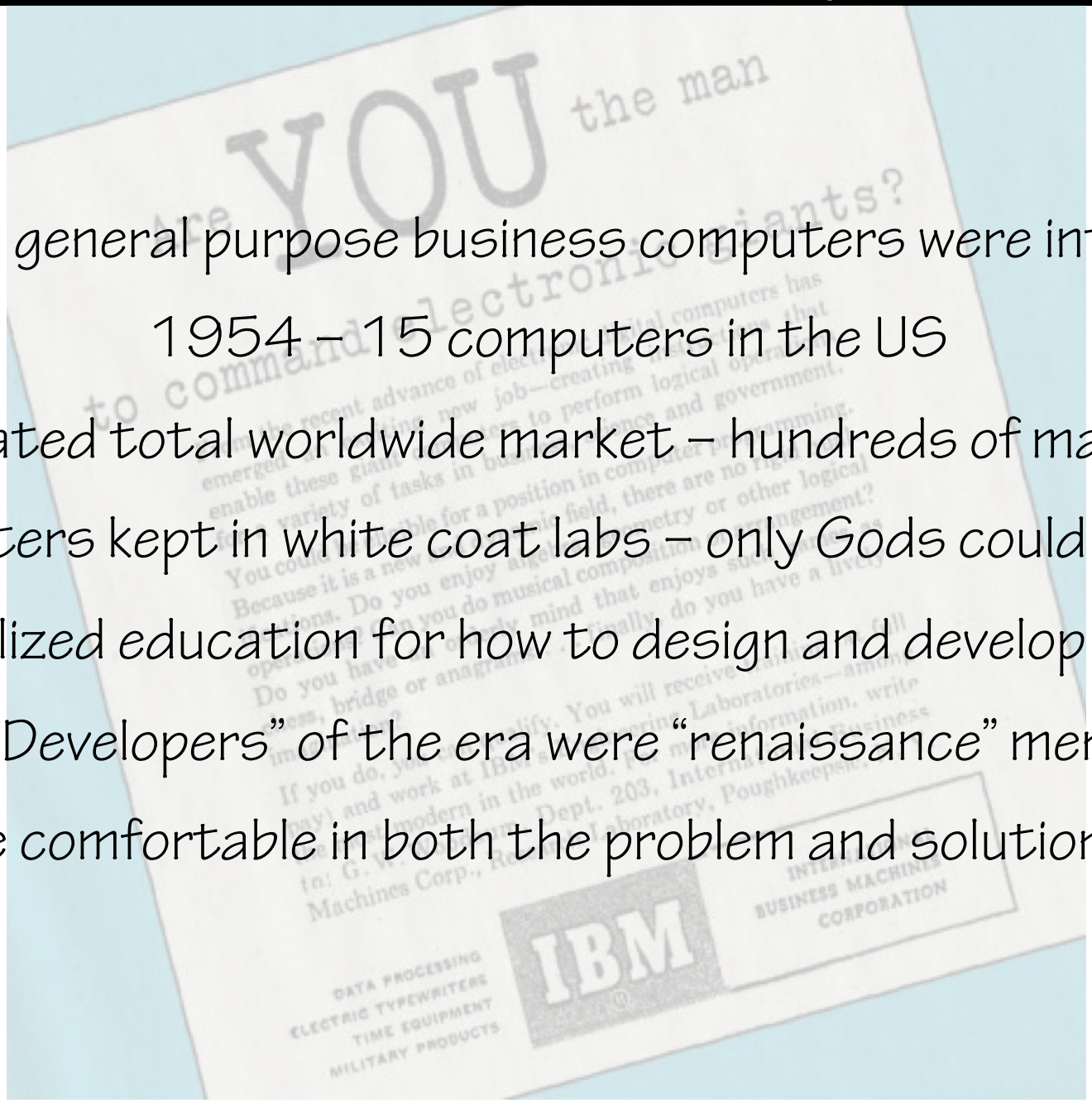
If you do, you can qualify. You will receive training (at full pay) and work at IBM's Engineering Laboratories—among the most modern in the world. For more information, write to: G. W. Woodsum, Dept. 203, International Business Machines Corp., Research Laboratory, Poughkeepsie, N. Y.

DATA PROCESSING
ELECTRIC TYPEWRITERS
TIME EQUIPMENT
MILITARY PRODUCTS

IBM

INTERNATIONAL
BUSINESS MACHINES
CORPORATION

1956 - Are You Man Enough?



1950s - general purpose business computers were introduced

1954 – 15 computers in the US

Estimated total worldwide market – hundreds of machines

Computers kept in white coat labs – only Gods could access

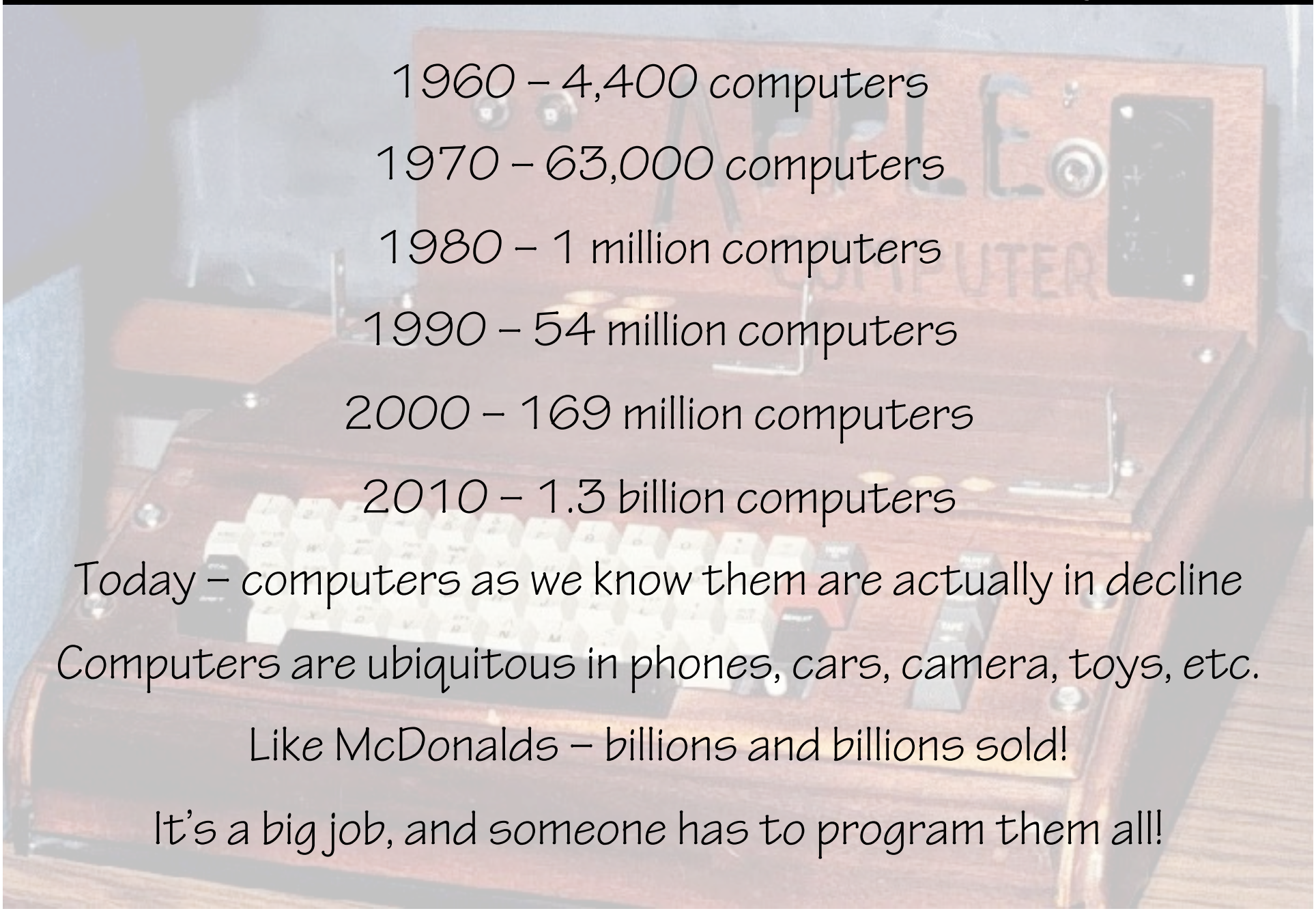
No formalized education for how to design and develop software

"Developers" of the era were "renaissance" men

Had to be comfortable in both the problem and solution domains

1976 - Stand Back - This Could Get Real Big!

1976 - Stand Back - This Could Get Real Big!



1960 – 4,400 computers

1970 – 63,000 computers

1980 – 1 million computers

1990 – 54 million computers

2000 – 169 million computers

2010 – 1.3 billion computers

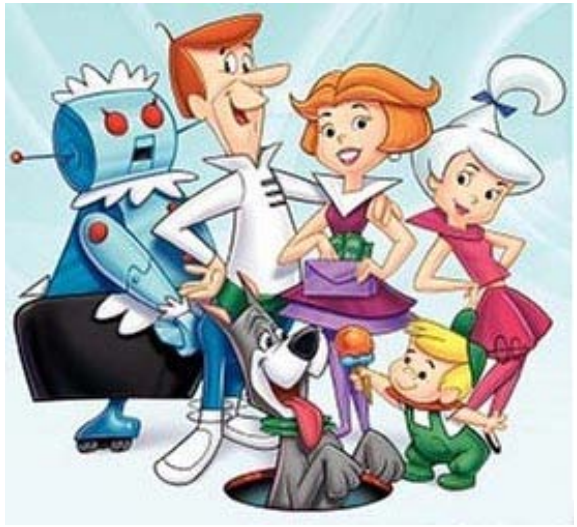
Today – computers as we know them are actually in decline

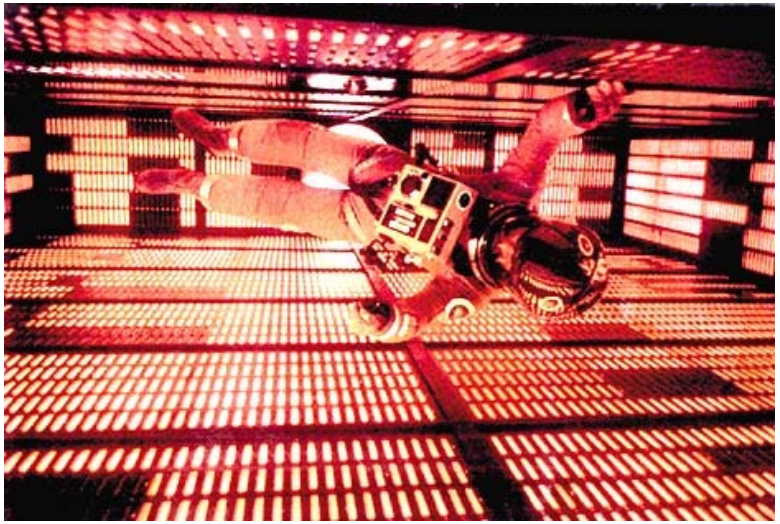
Computers are ubiquitous in phones, cars, camera, toys, etc.

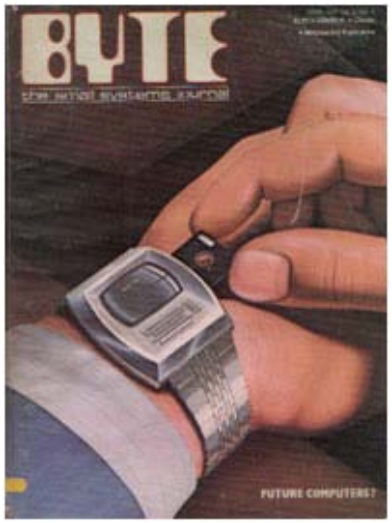
Like McDonalds – billions and billions sold!

It's a big job, and someone has to program them all!

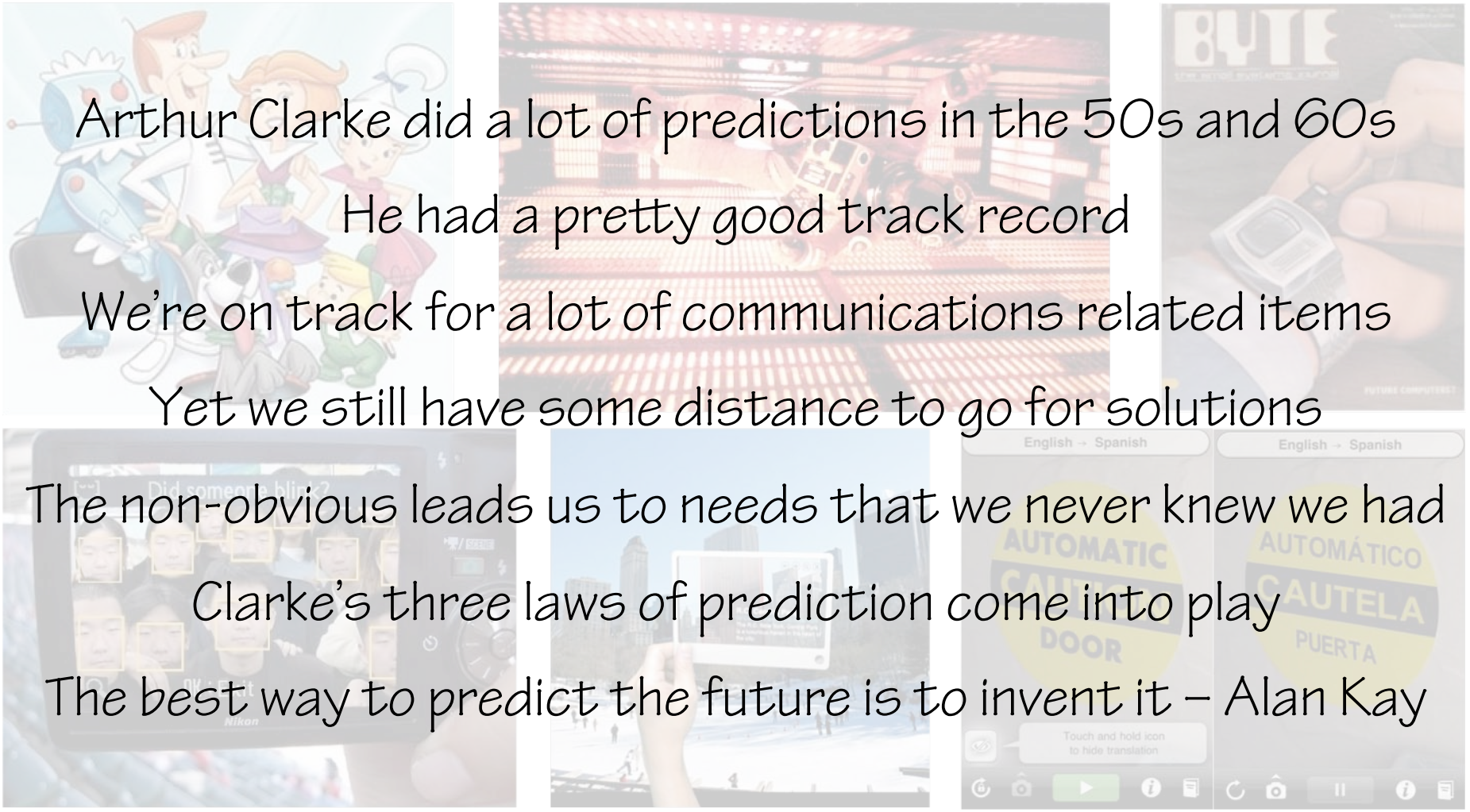
Expectations For Software Are Ever Increasing







Expectations For Software Are Ever Increasing



Arthur Clarke did a lot of predictions in the 50s and 60s

He had a pretty good track record

We're on track for a lot of communications related items

Yet we still have some distance to go for solutions

The non-obvious leads us to needs that we never knew we had

Clarke's three laws of prediction come into play

The best way to predict the future is to invent it – Alan Kay

The Odd Thing Is That the Stuff Gets Easier to Do All the Time...

```
public bool Accepts(IType type)
{
  foreach(IExpectedTypeConstraint constraint in myConstraints)
  Loop can be converted into LINQ-expression
    if (!constraint.Accepts(type)) return false;
  }
  return true;
}
```

```
if(FRead == null)
    throw new ArgumentNullException("FRead");
new TextReader(stream).WithDispose(FRead);
Cannot create an instance of the abstract class 'System.IO.TextReader'
```

```
else
{
  ctor = info.Type.GetConstructor(Type.EmptyTypes);
  externalizable = (IXmlExternalizable) ctor.Invoke(Type.EmptyTypes);
}
  Code is heuristically unreachable
```

```
case MembershipCreateStatus.InvalidQuestion:
    return "The password retrieval question provided
```

- Move to resource
- Annotate method 'ErrorCodeToString' with 'Localizable(false)'
- Annotate class 'AccountController' with 'Localizable(false)'
- Convert to verbatim string
- Split string
- Disable localization for this project
- Suppress inspection "Element is localizable" with comment
- Inspection options for "Element is localizable"

```
    action = delegate
    {
      Assertion.Assert(action != null,
```

- Copy to local variable
- Wrap local variable in array
- Suppress inspection "Access to modified closure" with comment
- Inspection options for "Access to modified closure"

```
    IDeclaredType declaredType = type as IDeclaredType;

    if (declaredType != null)
    {
      ITypeElement element = declaredType.GetTypeElement();
```

- Use 'var'
- Inspection options for "Use 'var' keyword when possible"

```
if (elementType == null) return default(TResult);
Convert to 'return' statement
```

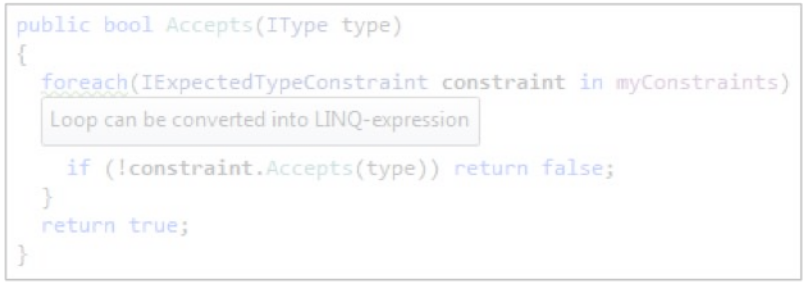
Before

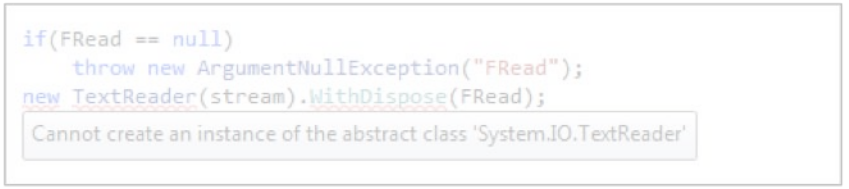
```
public static void AssertKnownType(Type type)
{
    IList<Type> typesKnown =
        new Type[] {TypeOf.Boolean, TypeOf.Int32, TypeOf.String};

    if(!typesKnown.Contains(type))
```

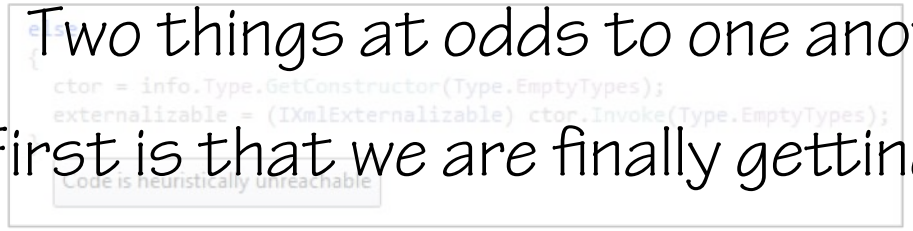
After

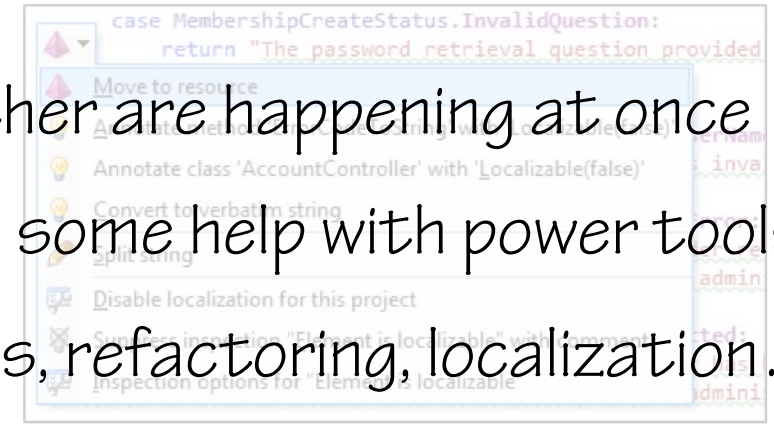
```
public static void AssertKnownType
    (Type type, Func<IList<Type>, bool> contains)
{
    IList<Type> typesKnown =
        new Type[] {TypeOf.Boolean, TypeOf.Int32, TypeOf.String};

    if(!contains(typesKnown))
```

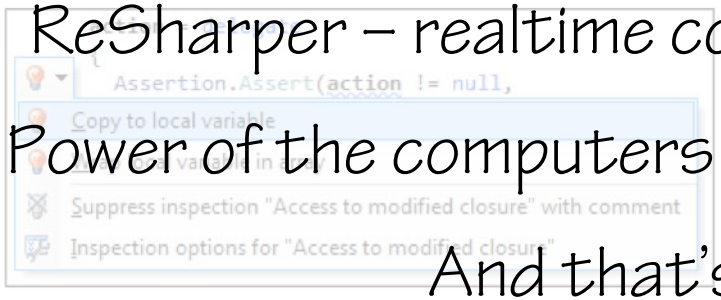
The Odd Thing Is That the Stuff Gets Easier to Do All the Time…

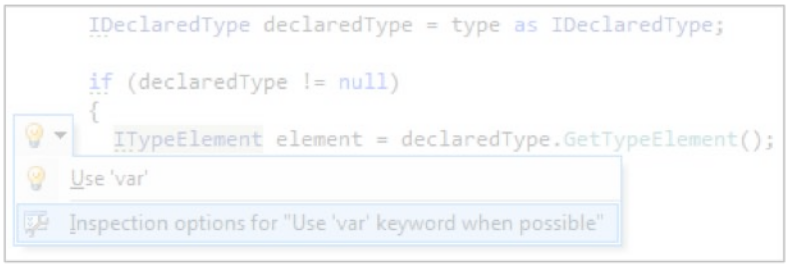


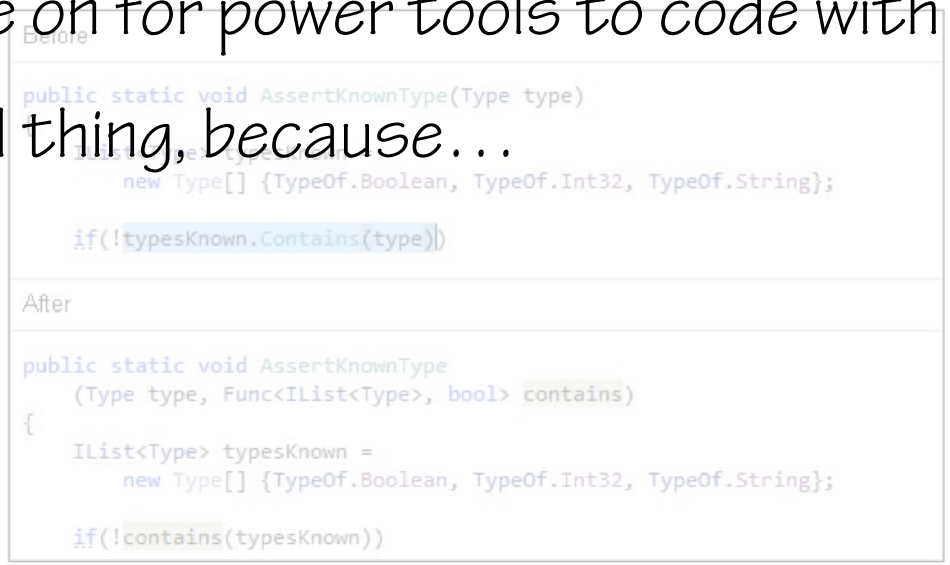












Two things at odds to one another are happening at once

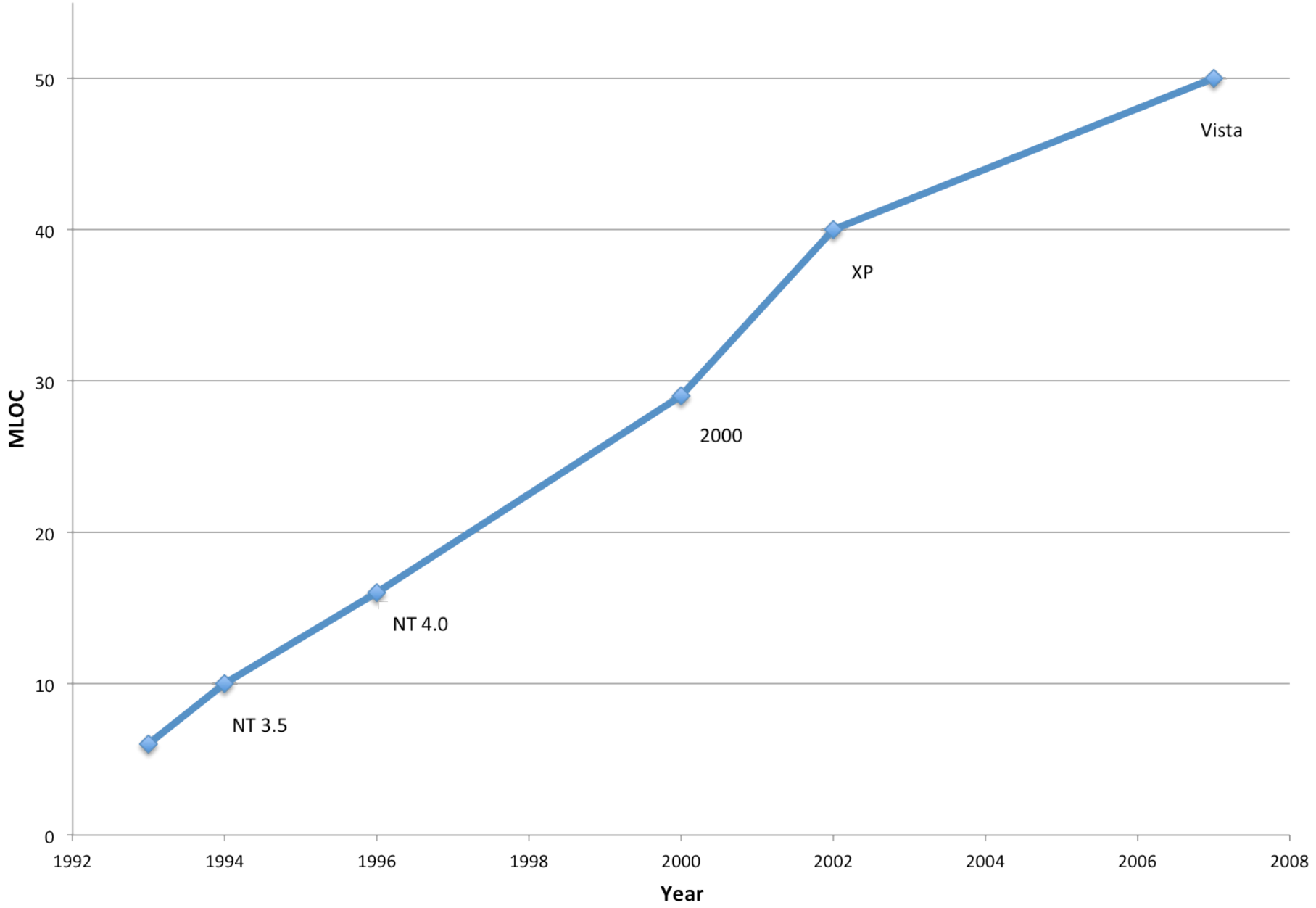
First is that we are finally getting some help with power tools

ReSharper – realtime code analysis, refactoring, localization…

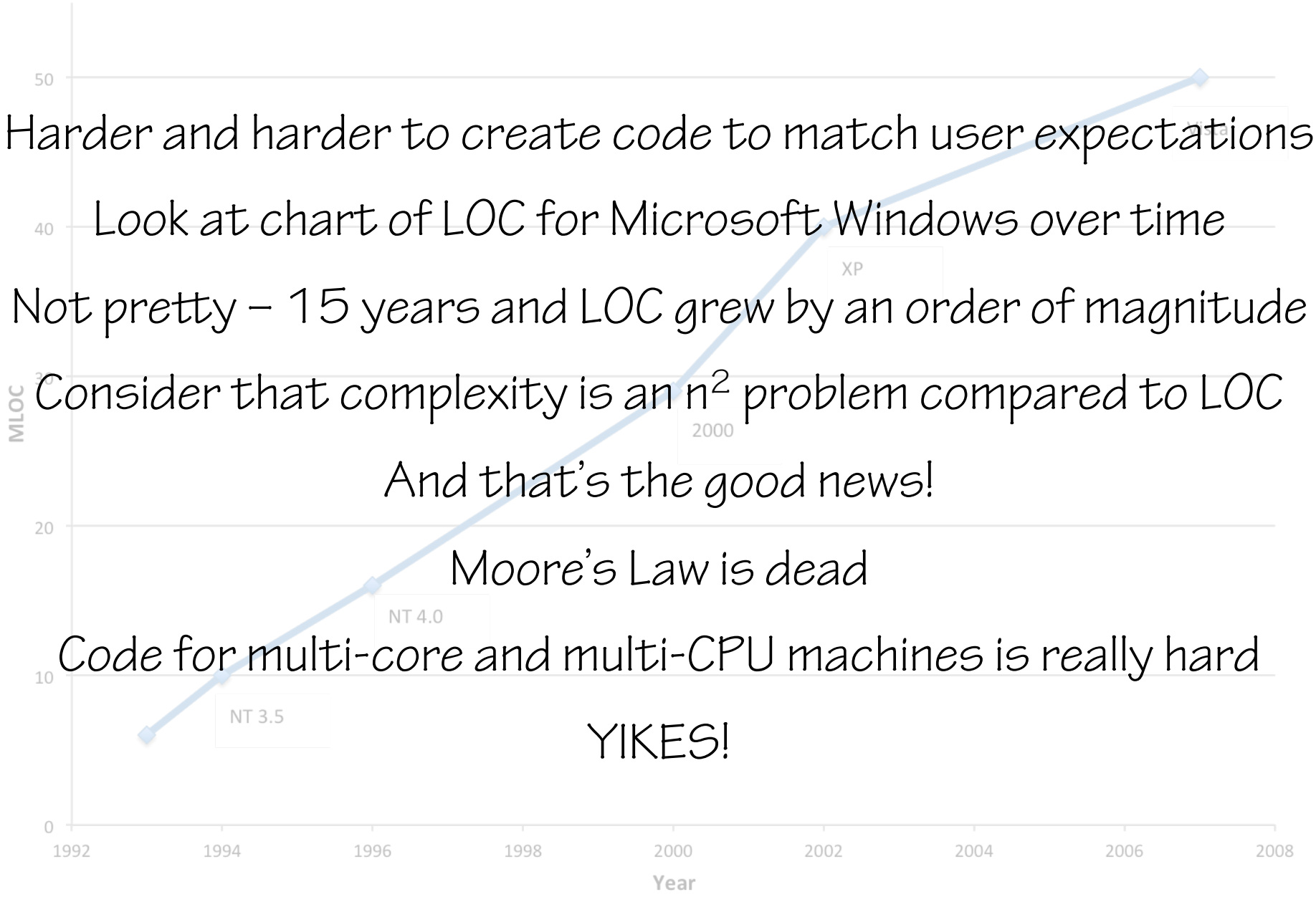
Power of the computers we code on for power tools to code with

And that's a good thing, because…

The Odd Thing Is That the Stuff Gets Harder to Do All the Time...



The Odd Thing Is That the Stuff Gets Harder to Do All the Time…



Harder and harder to create code to match user expectations

Look at chart of LOC for Microsoft Windows over time

Not pretty – 15 years and LOC grew by an order of magnitude

Consider that complexity is an n^2 problem compared to LOC

And that's the good news!

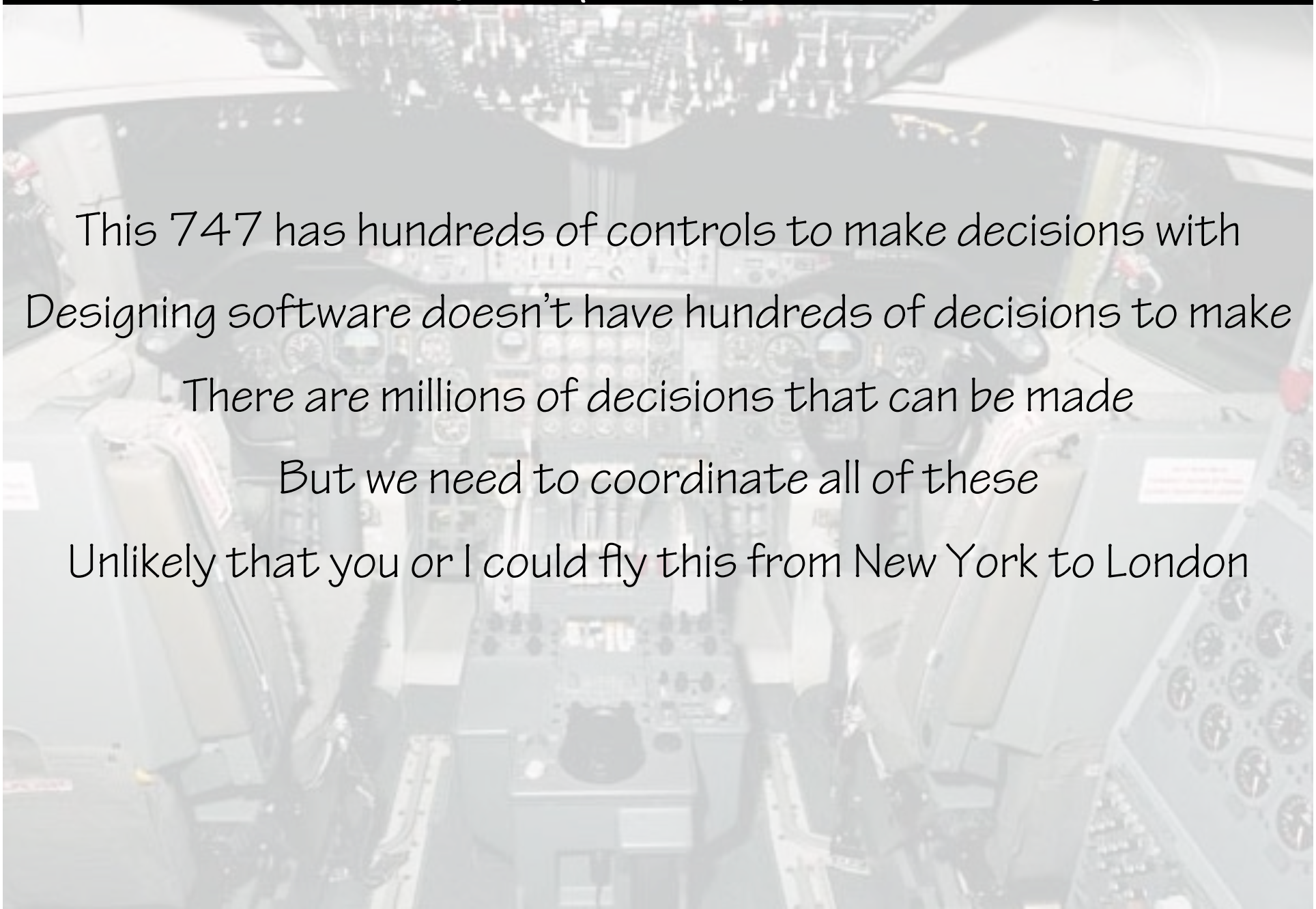
Moore's Law is dead

Code for multi-core and multi-CPU machines is really hard

YIKES!

Think Handling Multiple Things At Once is Easy?

Think Handling Multiple Things At Once is Easy?



This 747 has hundreds of controls to make decisions with

Designing software doesn't have hundreds of decisions to make

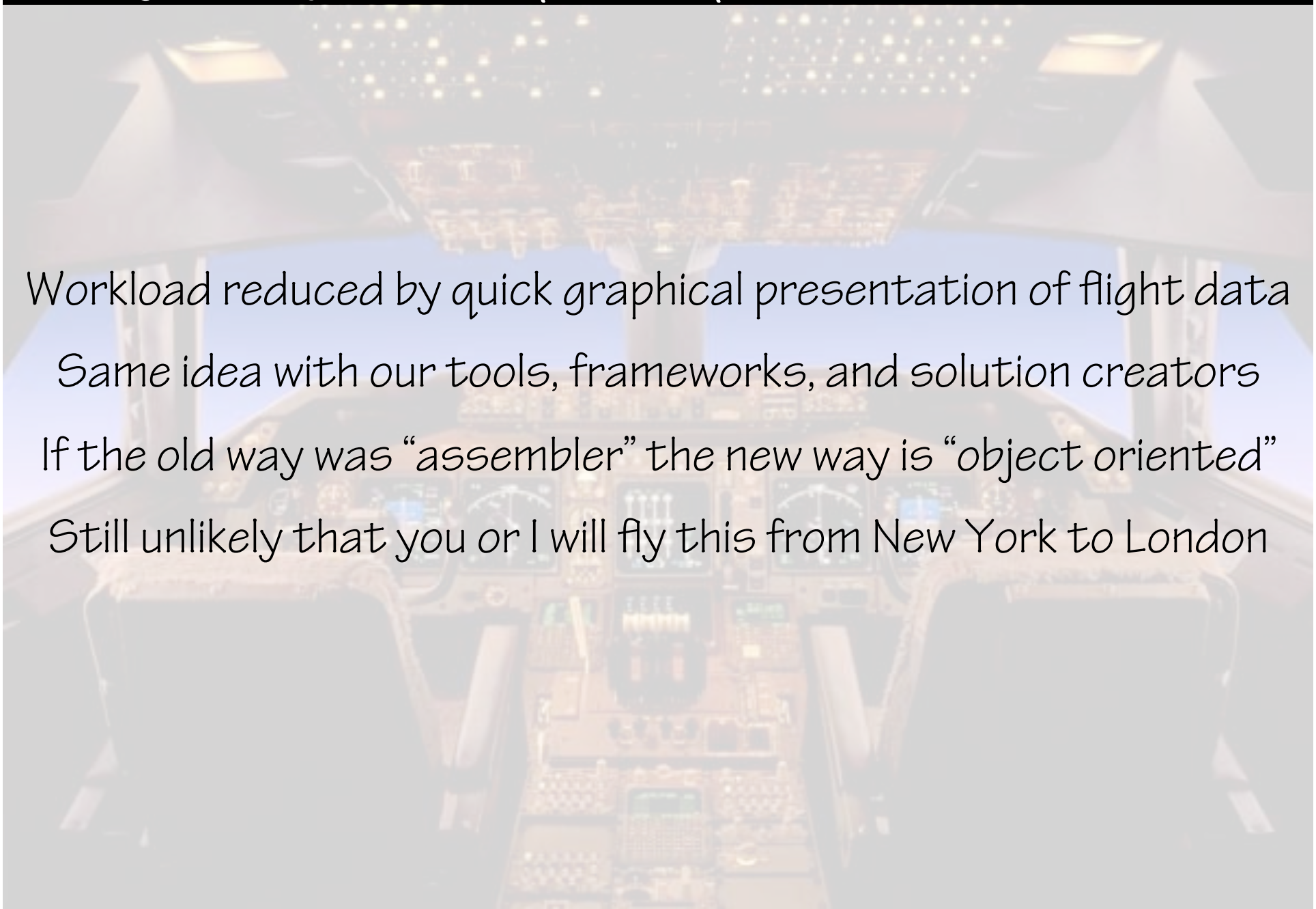
There are millions of decisions that can be made

But we need to coordinate all of these

Unlikely that you or I could fly this from New York to London

Maybe Using More Computer Help Will Make Your Life Better

Maybe Using More Computer Help Will Make Your Life Better



Workload reduced by quick graphical presentation of flight data

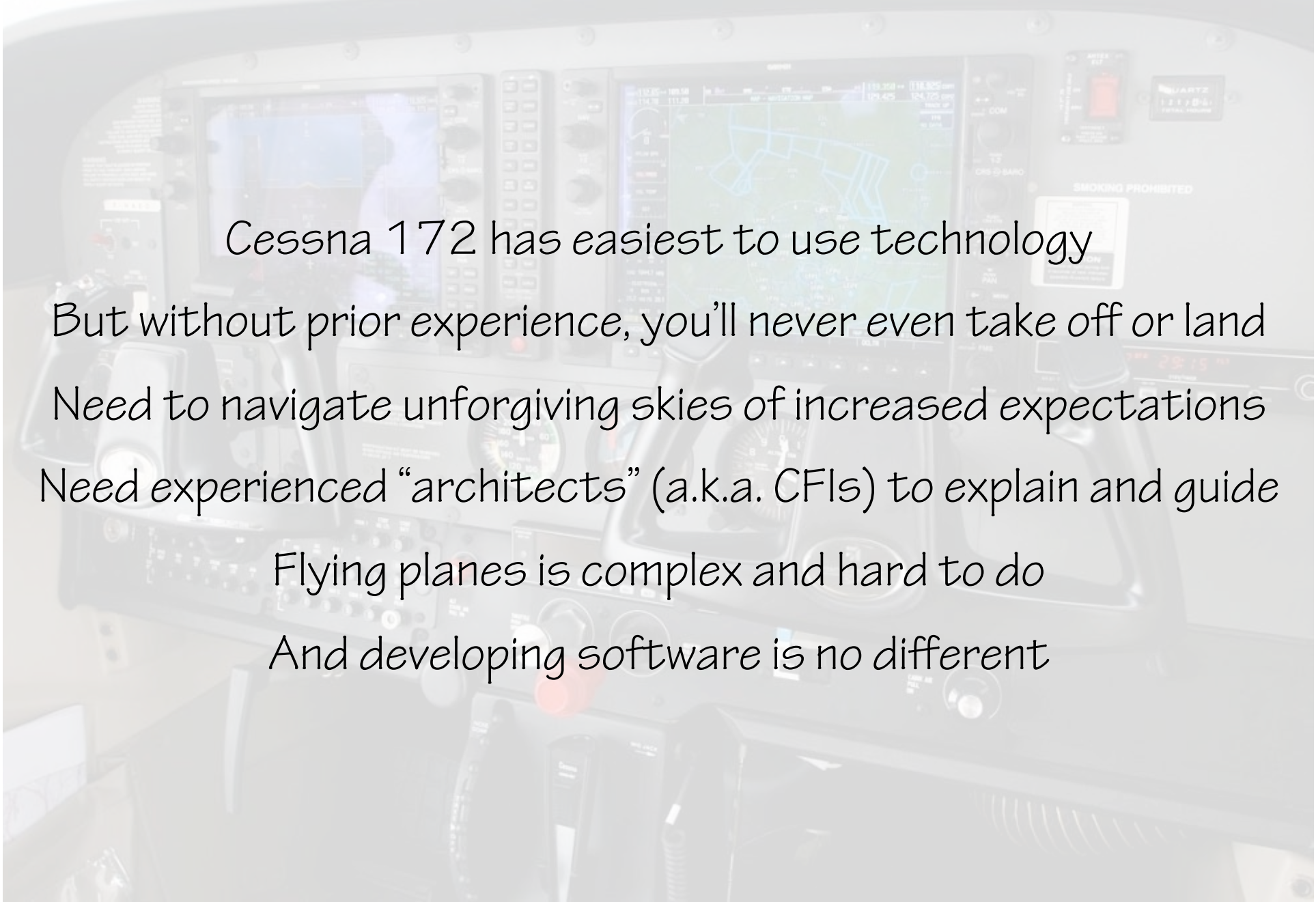
Same idea with our tools, frameworks, and solution creators

If the old way was "assembler" the new way is "object oriented"

Still unlikely that you or I will fly this from New York to London

How About Just the Visual Basic Approach to Planes?

How About Just the Visual Basic Approach to Planes?



Cessna 172 has easiest to use technology

But without prior experience, you'll never even take off or land

Need to navigate unforgiving skies of increased expectations

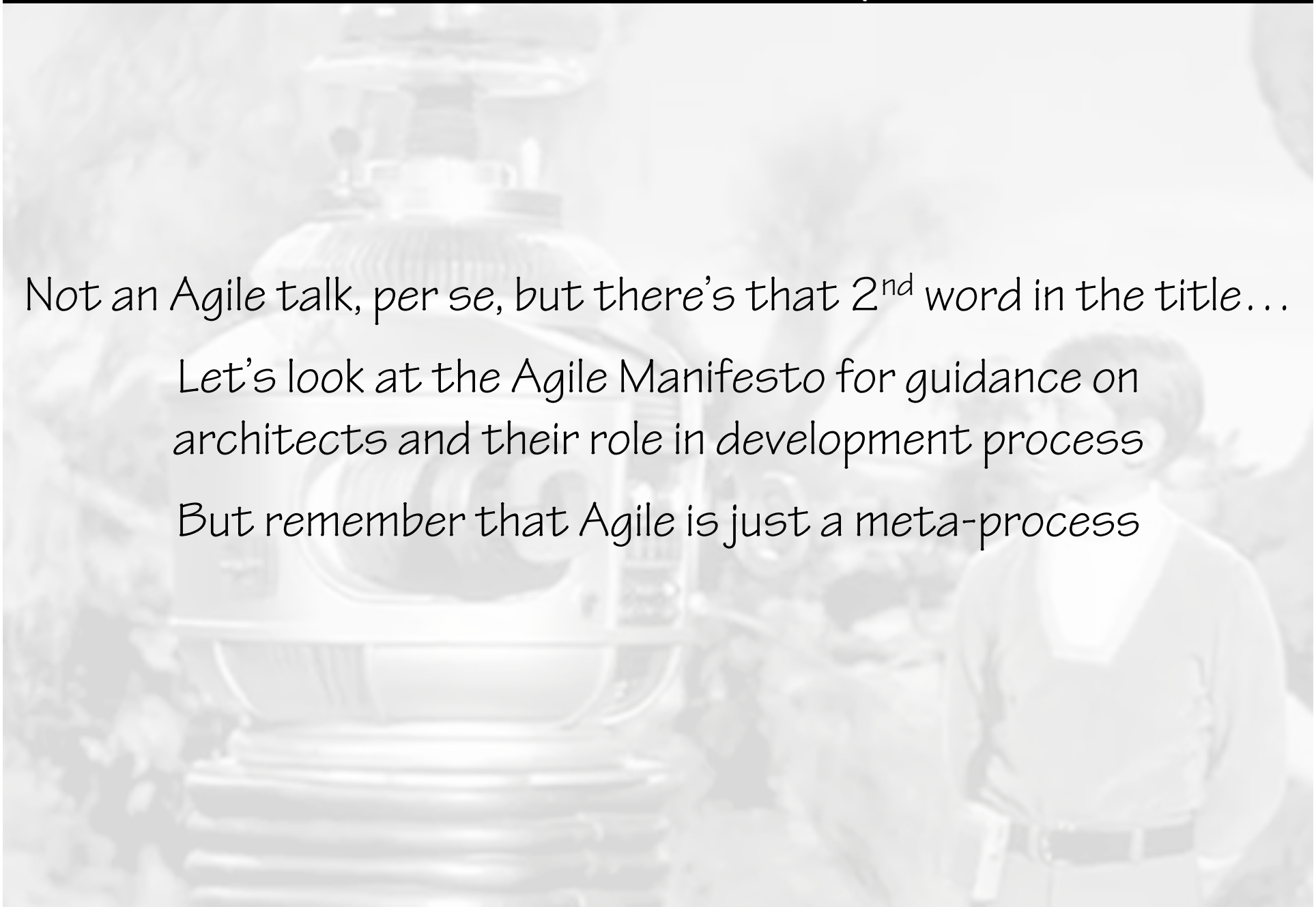
Need experienced "architects" (a.k.a. CFIs) to explain and guide

Flying planes is complex and hard to do

And developing software is no different

Danger, Will Robinson – We Have Agile Impedance Mismatches!

Danger, Will Robinson – We Have Agile Impedance Mismatches!



Not an Agile talk, per se, but there's that 2nd word in the title…

Let's look at the Agile Manifesto for guidance on architects and their role in development process

But remember that Agile is just a meta-process

Danger, Will Robinson – We Have Agile Impedance Mismatches!

All four values from the Manifesto are guidance

Individuals and interactions over process and tools

Working software over comprehensive documentation

Customer interaction over contract negotiation

Responding to change over following a plan

Danger, Will Robinson – We Have Agile Impedance Mismatches!

Four of twelve principles from the Manifesto are also of interest

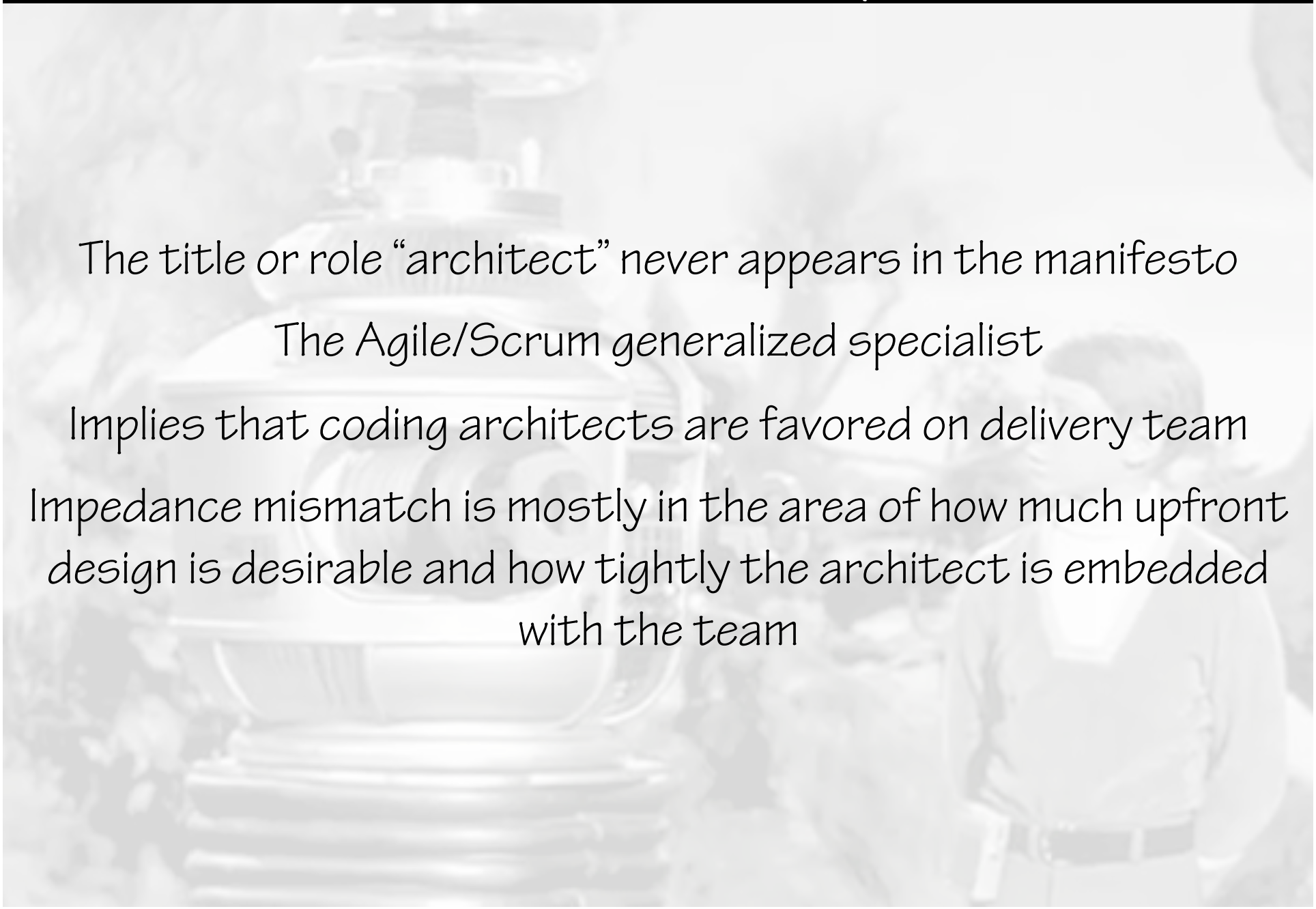
Working software is the primary measure of progress

Continuous attention to technical excellence
and good design enhances agility

Simplicity – the art of maximizing the amount
of work **not** done – is essential

The best architectures, requirements, and designs
emerge from self-organizing teams

Danger, Will Robinson – We Have Agile Impedance Mismatches!



The title or role "architect" never appears in the manifesto

The Agile/Scrum generalized specialist

Implies that coding architects are favored on delivery team

Impedance mismatch is mostly in the area of how much upfront design is desirable and how tightly the architect is embedded with the team

So, isn't the Whole Concept of an Agile Architect an Oxymoron?

So, isn't the Whole Concept of an Agile Architect an Oxymoron?

Learning as you go is a good (and very Agile) thing to do

But you need an idea of direction and basics before you start

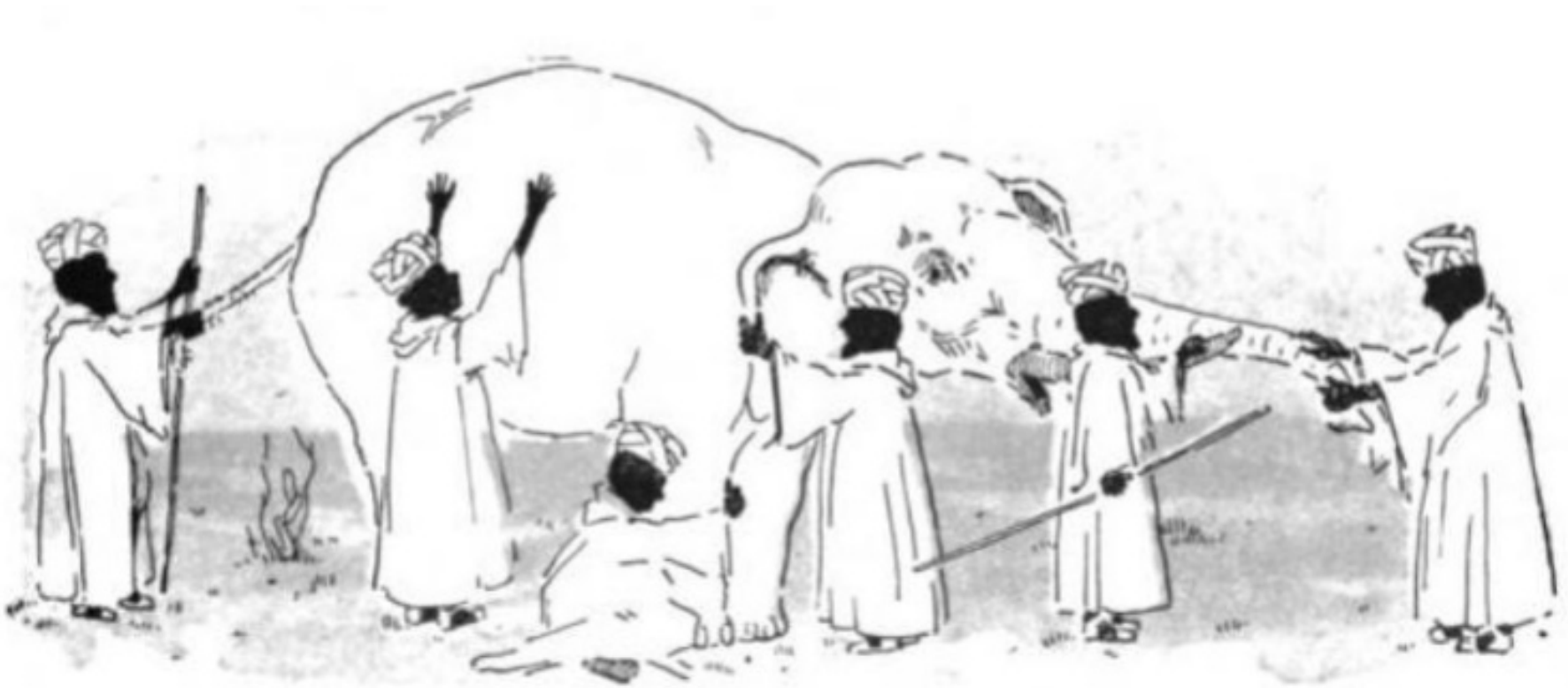
That's what architects do

And that's the premise here, so no on the oxymoron issue

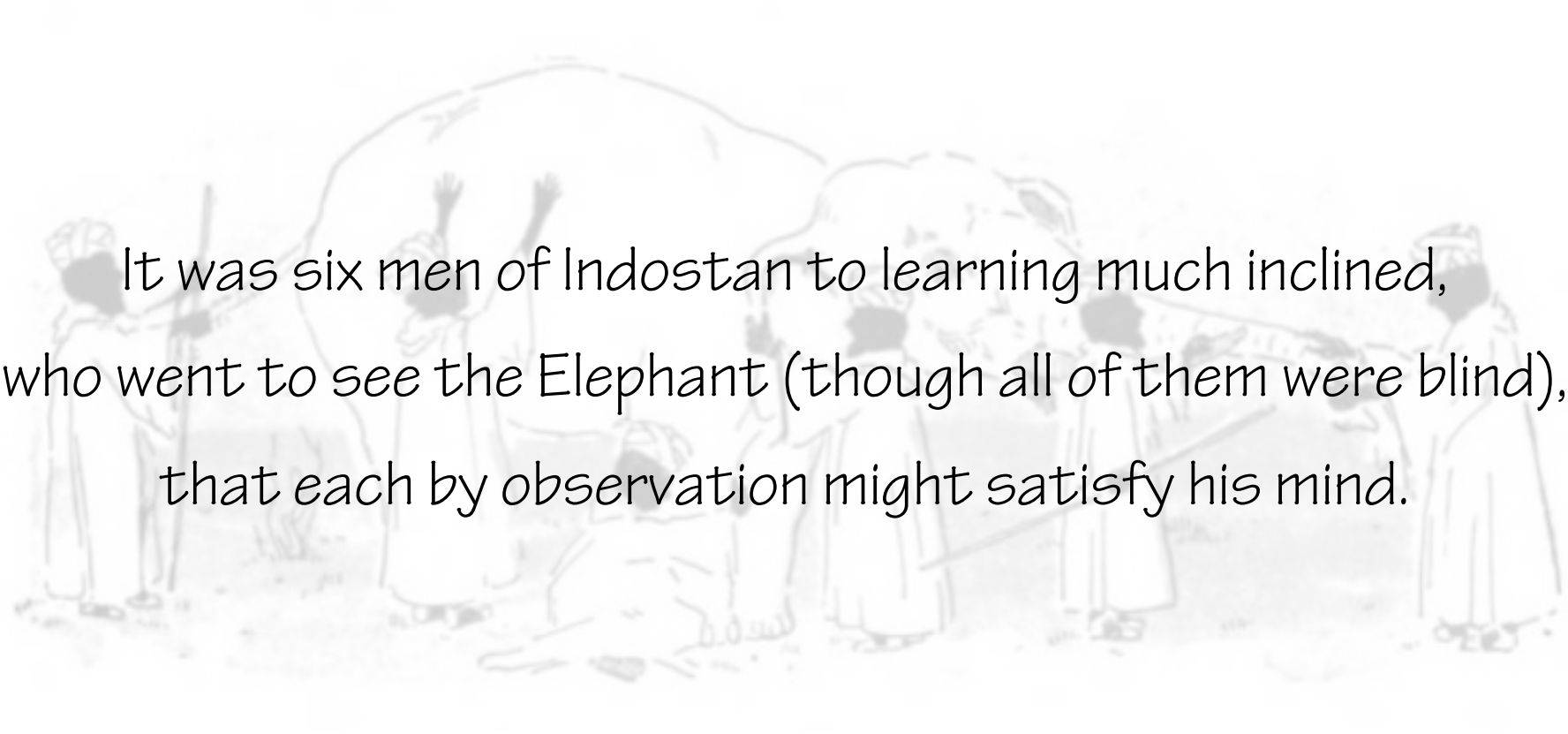
But just saying "Hey, we're Agile", and using power tools doesn't mean that there isn't a need for architects in Agile

However, the role changes somewhat in Agile

Blind Men and Elephants

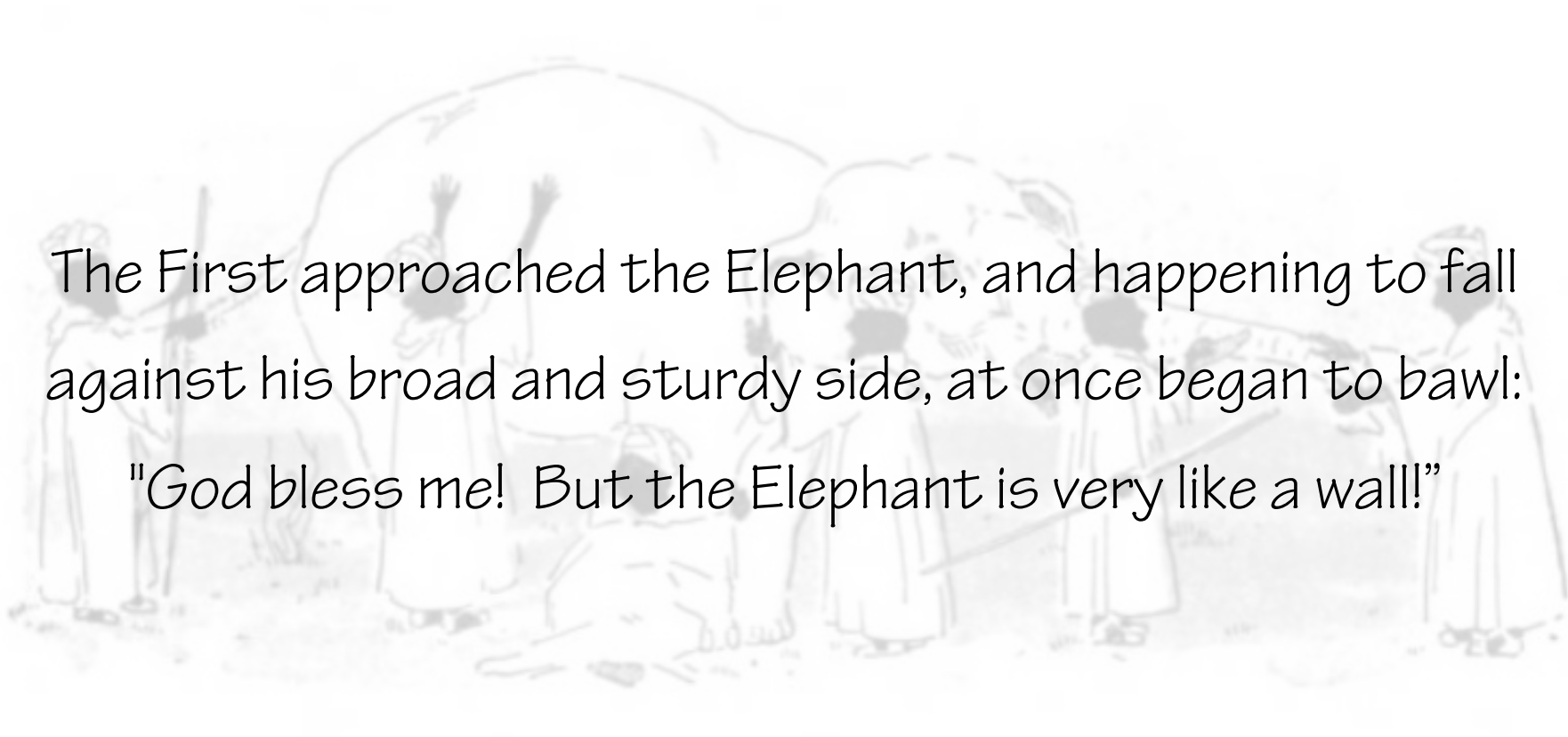


Blind Men and Elephants



It was six men of Indostan to learning much inclined,
who went to see the Elephant (though all of them were blind),
that each by observation might satisfy his mind.

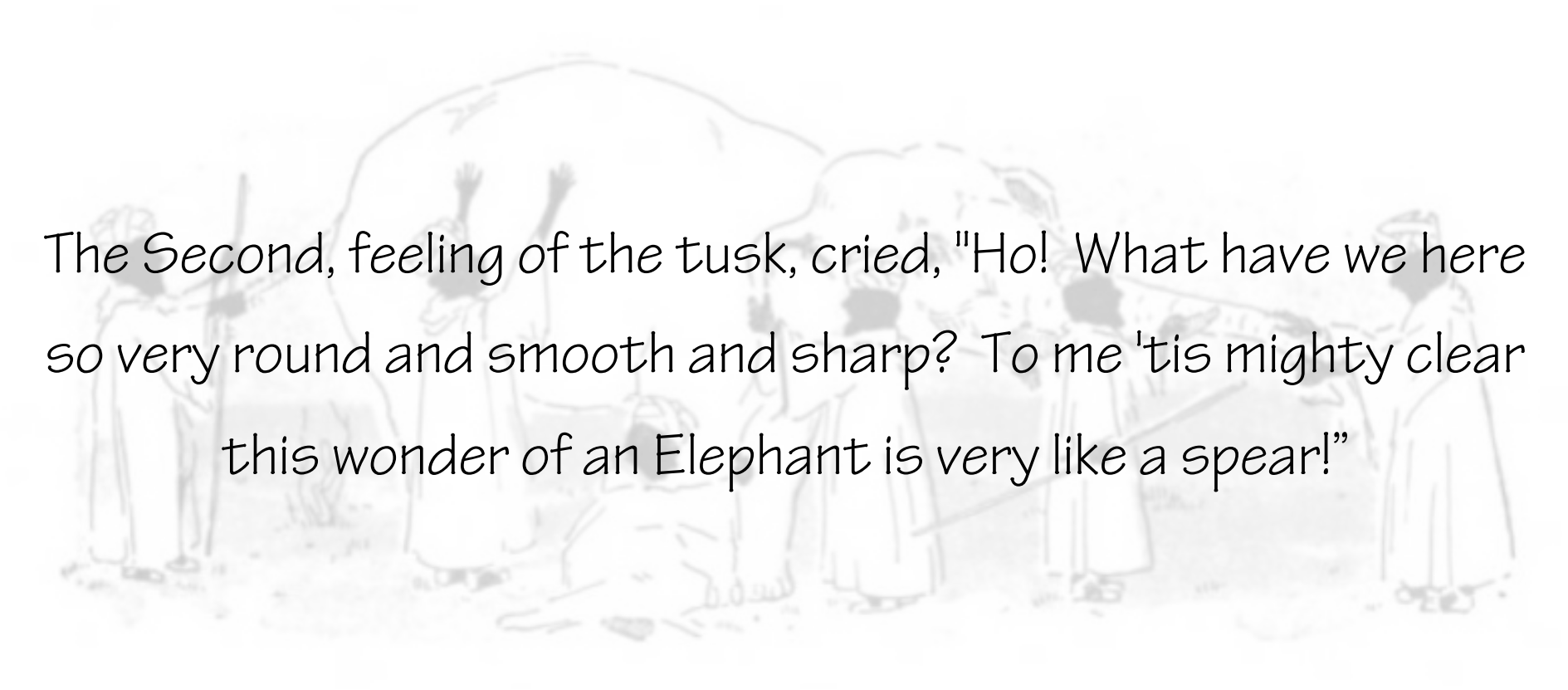
Blind Men and Elephants



The First approached the Elephant, and happening to fall against his broad and sturdy side, at once began to bawl:

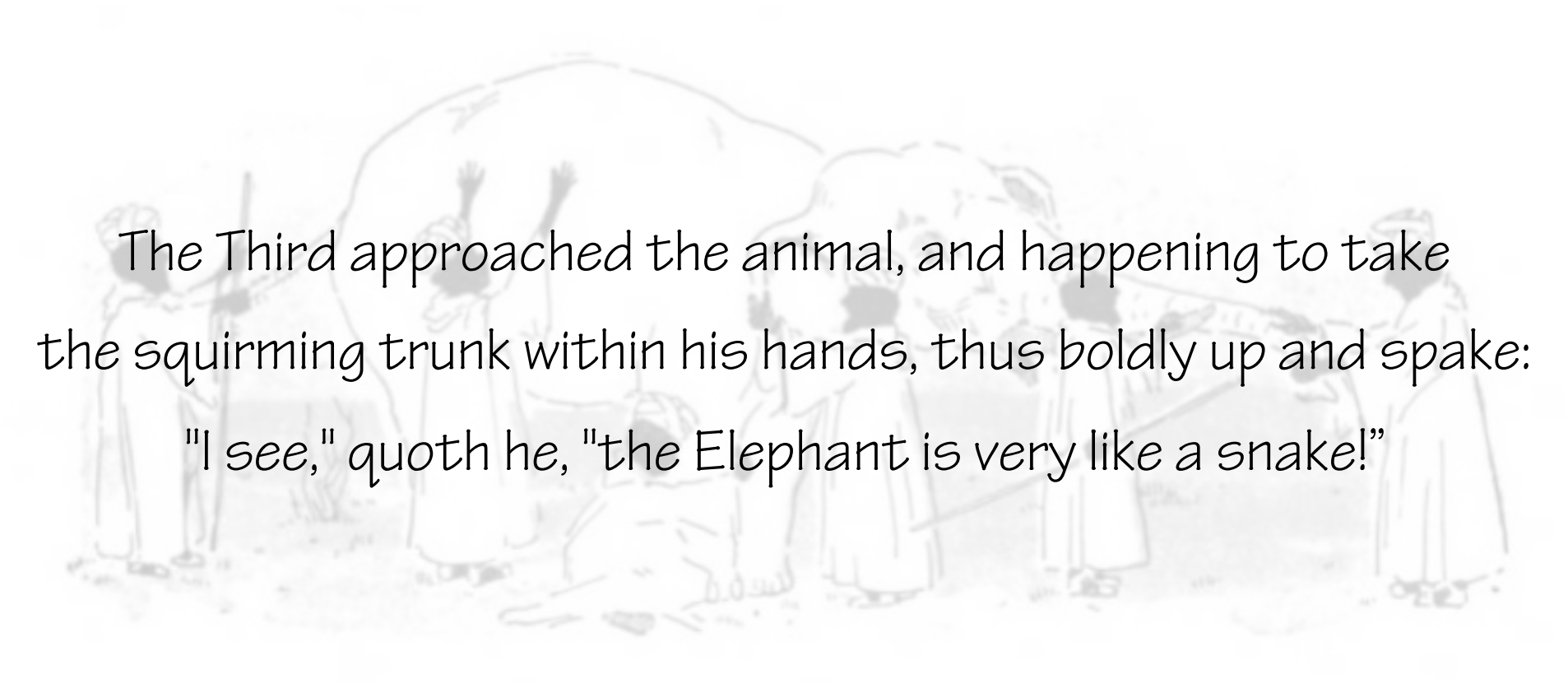
"God bless me! But the Elephant is very like a wall!"

Blind Men and Elephants



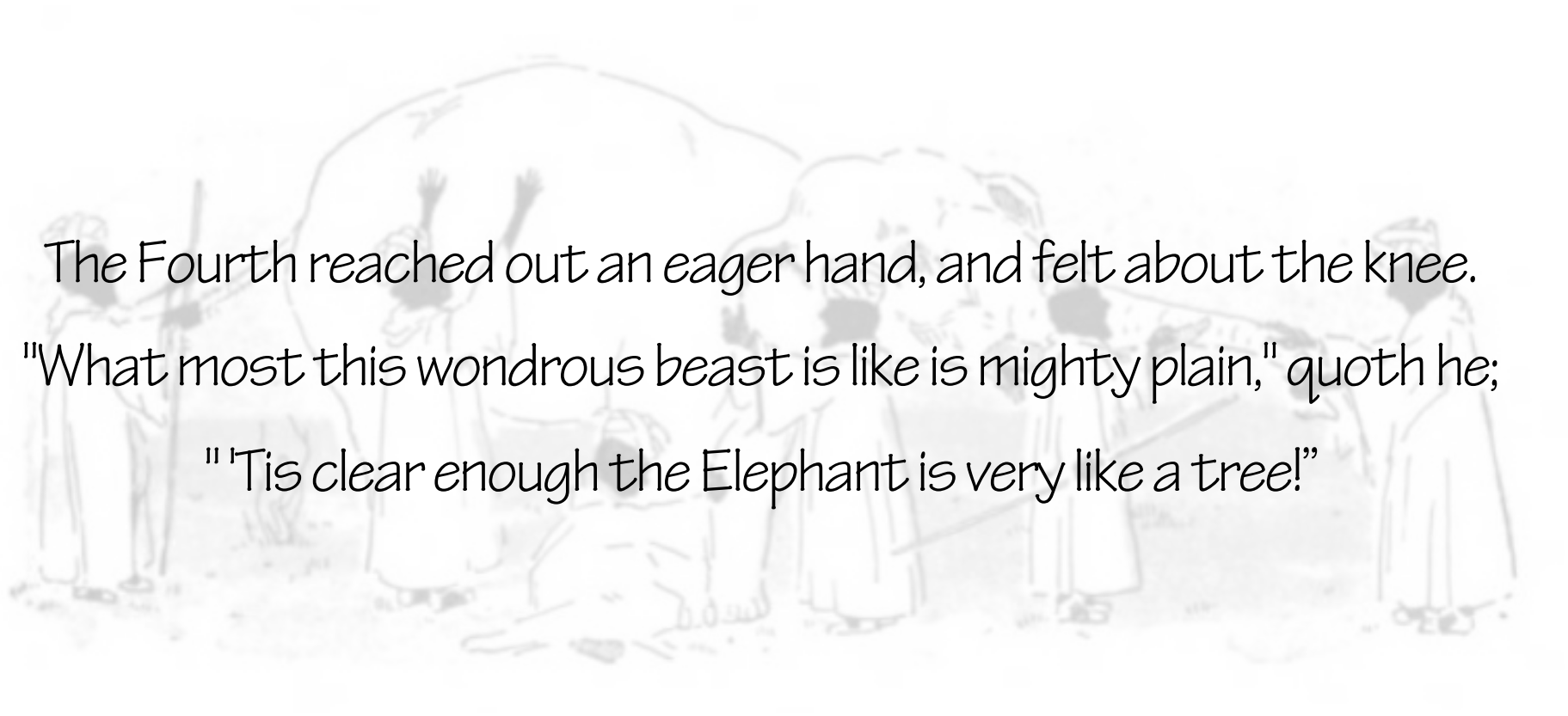
The Second, feeling of the tusk, cried, "Ho! What have we here so very round and smooth and sharp? To me 'tis mighty clear this wonder of an Elephant is very like a spear!"

Blind Men and Elephants



The Third approached the animal, and happening to take
the squirming trunk within his hands, thus boldly up and spake:
"I see," quoth he, "the Elephant is very like a snake!"

Blind Men and Elephants

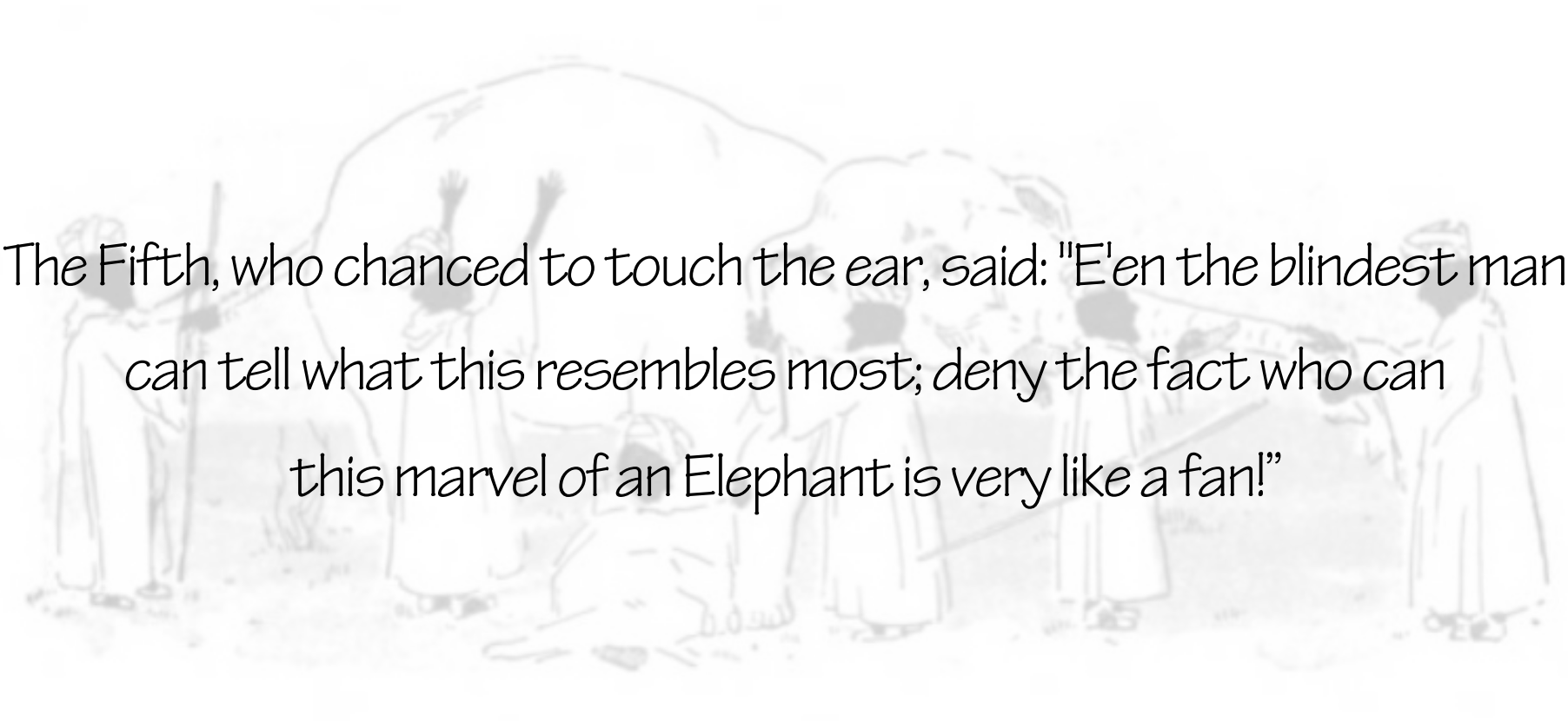


The Fourth reached out an eager hand, and felt about the knee.

"What most this wondrous beast is like is mighty plain," quoth he;

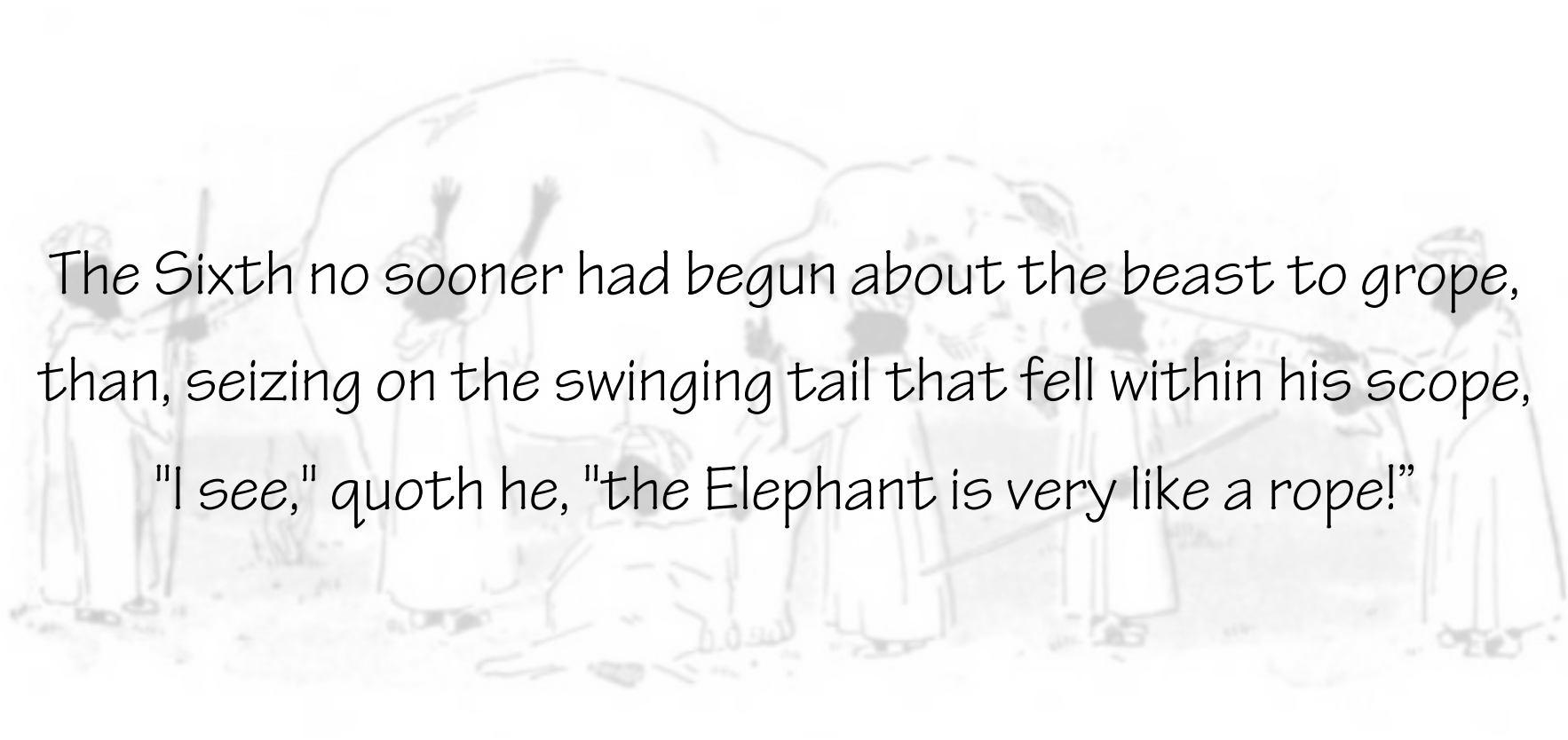
" 'Tis clear enough the Elephant is very like a tree!"

Blind Men and Elephants



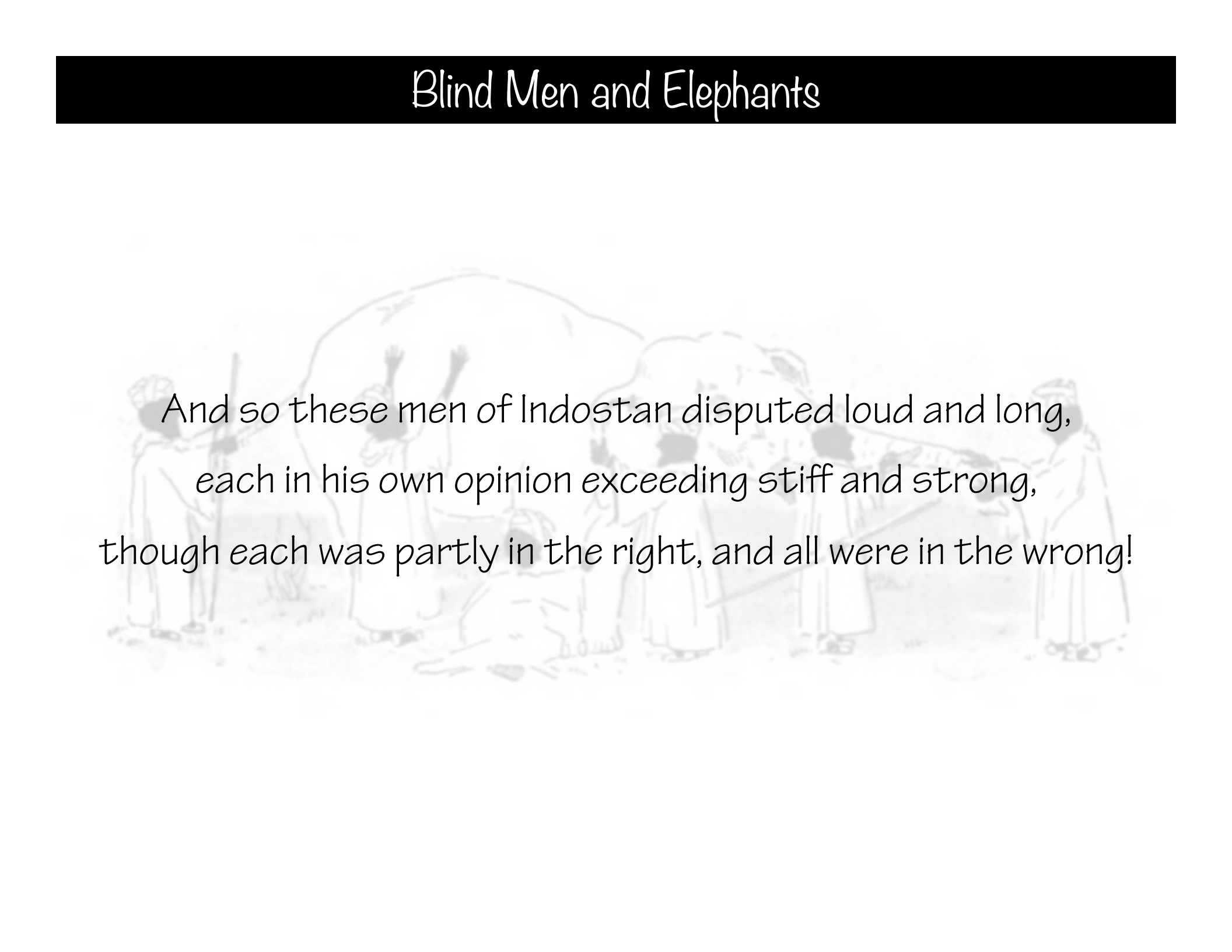
The Fifth, who chanced to touch the ear, said: "E'en the blindest man
can tell what this resembles most; deny the fact who can
this marvel of an Elephant is very like a fan!"

Blind Men and Elephants



The Sixth no sooner had begun about the beast to grope,
than, seizing on the swinging tail that fell within his scope,
"I see," quoth he, "the Elephant is very like a rope!"

Blind Men and Elephants

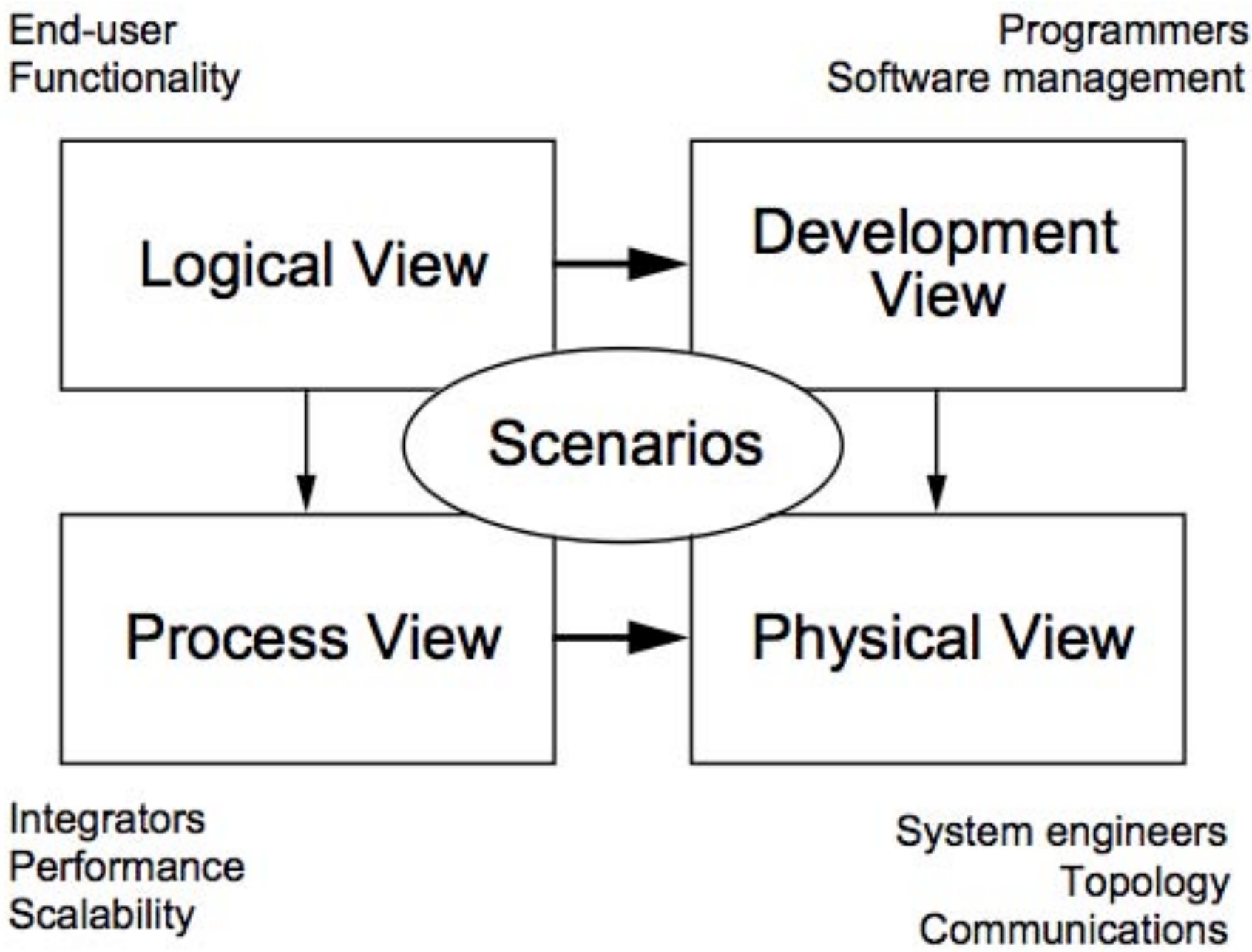


And so these men of Indostan disputed loud and long,

each in his own opinion exceeding stiff and strong,

though each was partly in the right, and all were in the wrong!

Traditionally, An Architect Saw 4+1 Views



Traditionally, An Architect Saw 4+1 Views

From Kruchten's seminal work

Architecture = set of rationales for making design decisions

Functional as well as non-functional considerations

Logical view is the object model of the design

Process view for concurrency and synchronization aspects

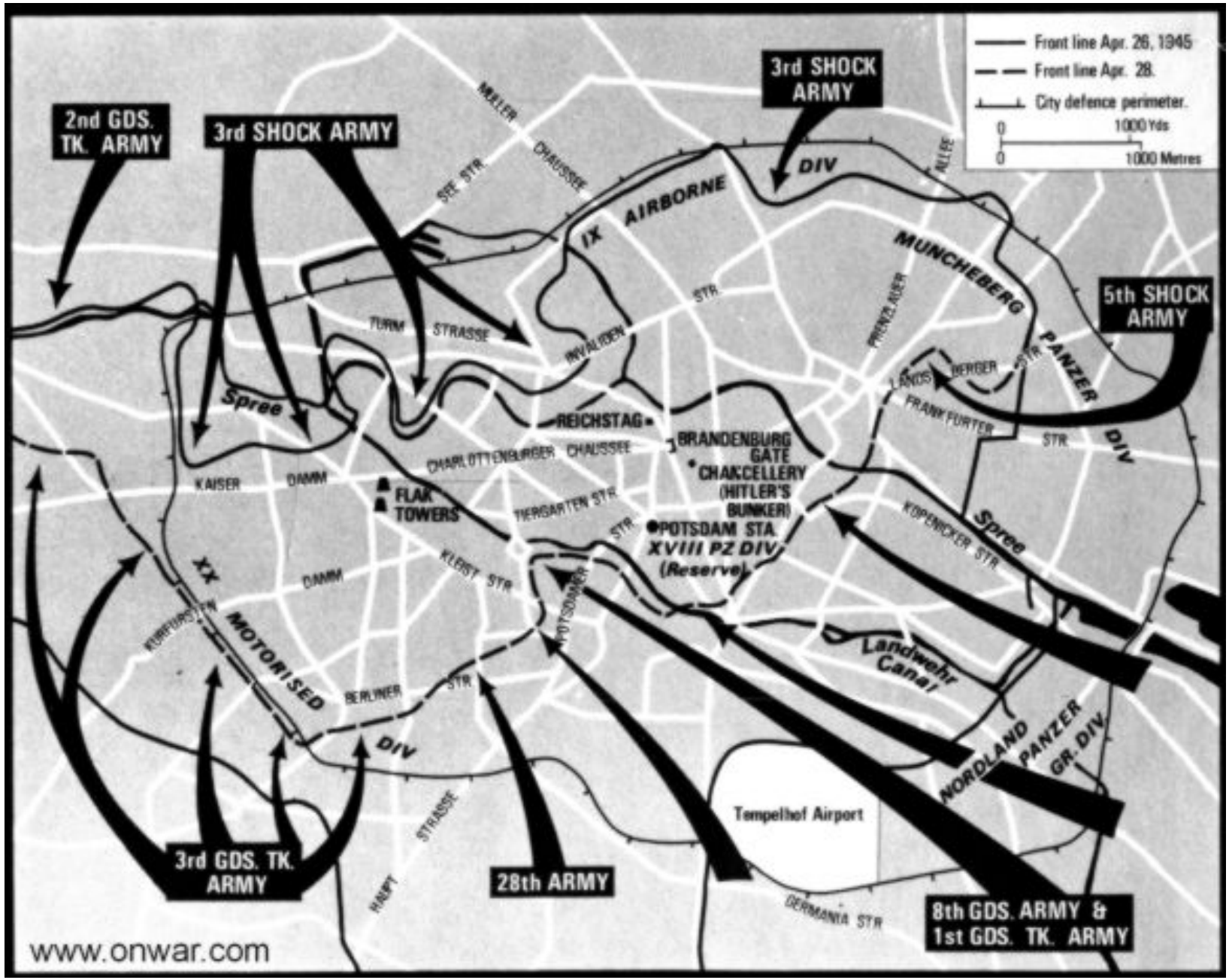
Physical view describes hardware and distributed aspects

Development view describes software module organization

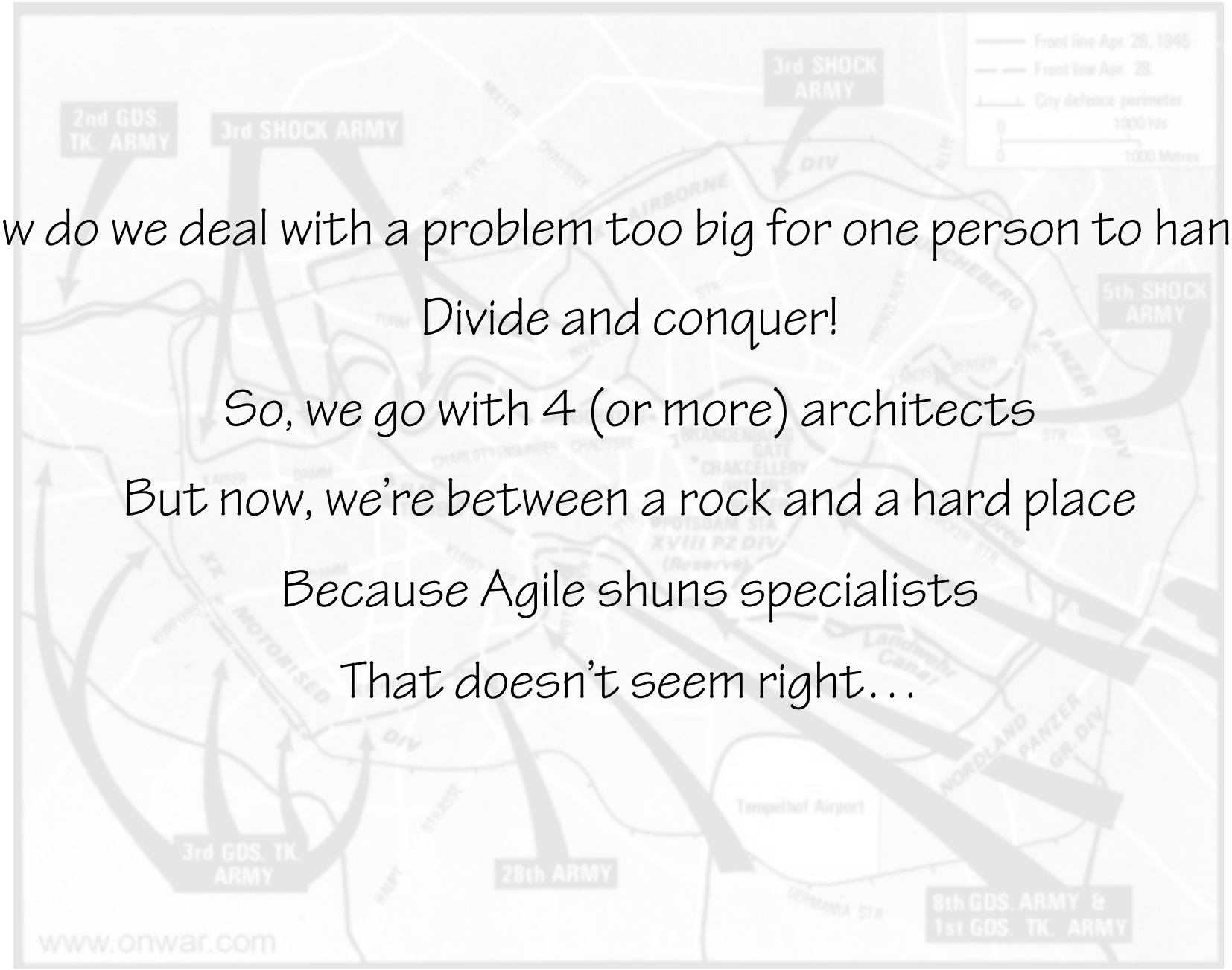
Scenarios ties together all inter-related views

With today's complexities, who can be this superhero architect?

Too Much to Do Alone? Get Agile - Divide and Conquer!



Too Much to Do Alone? Get Agile - Divide and Conquer!



How do we deal with a problem too big for one person to handle?

Divide and conquer!

So, we go with 4 (or more) architects

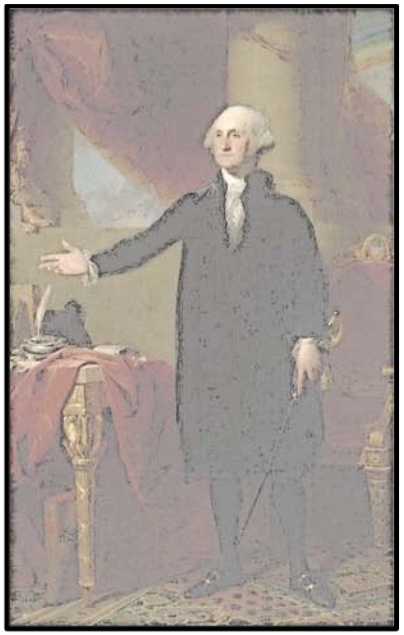
But now, we're between a rock and a hard place

Because Agile shuns specialists

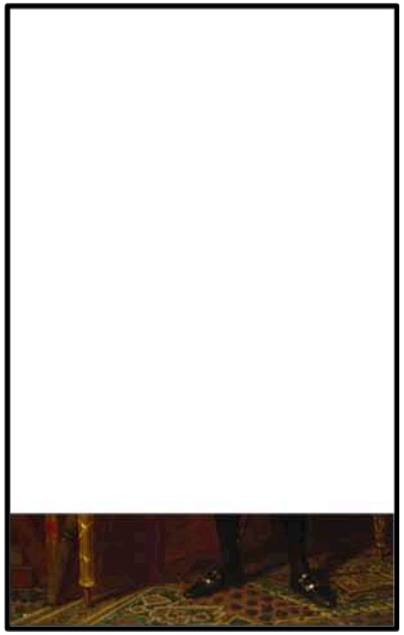
That doesn't seem right…

Agile Encourages Emergent Design

Iterative



Incremental



Agile Encourages Emergent Design

Basic difference between Agile and Waterfall approaches

The code is developed in a different fashion

Traditional Waterfall wants a BDUF before implementation

Then implement from the bottom up

But if assumptions or requirements change - ton of rework

For example, Gilbert Stuart's portrait of George Washington

Incremental gets messy if George wants to sit after we start

In Agile we decide at the last responsible moment

We develop iteratively, and constantly verify requirements

Rough design needed to start – detailed design emerges

Agile Needs Architectural Roles Based on Delivered Business Value

Agile Needs Architectural Roles Based on Delivered Business Value

Prime Directive – bring the business closer to the delivery team

We'll need different architect roles to do that

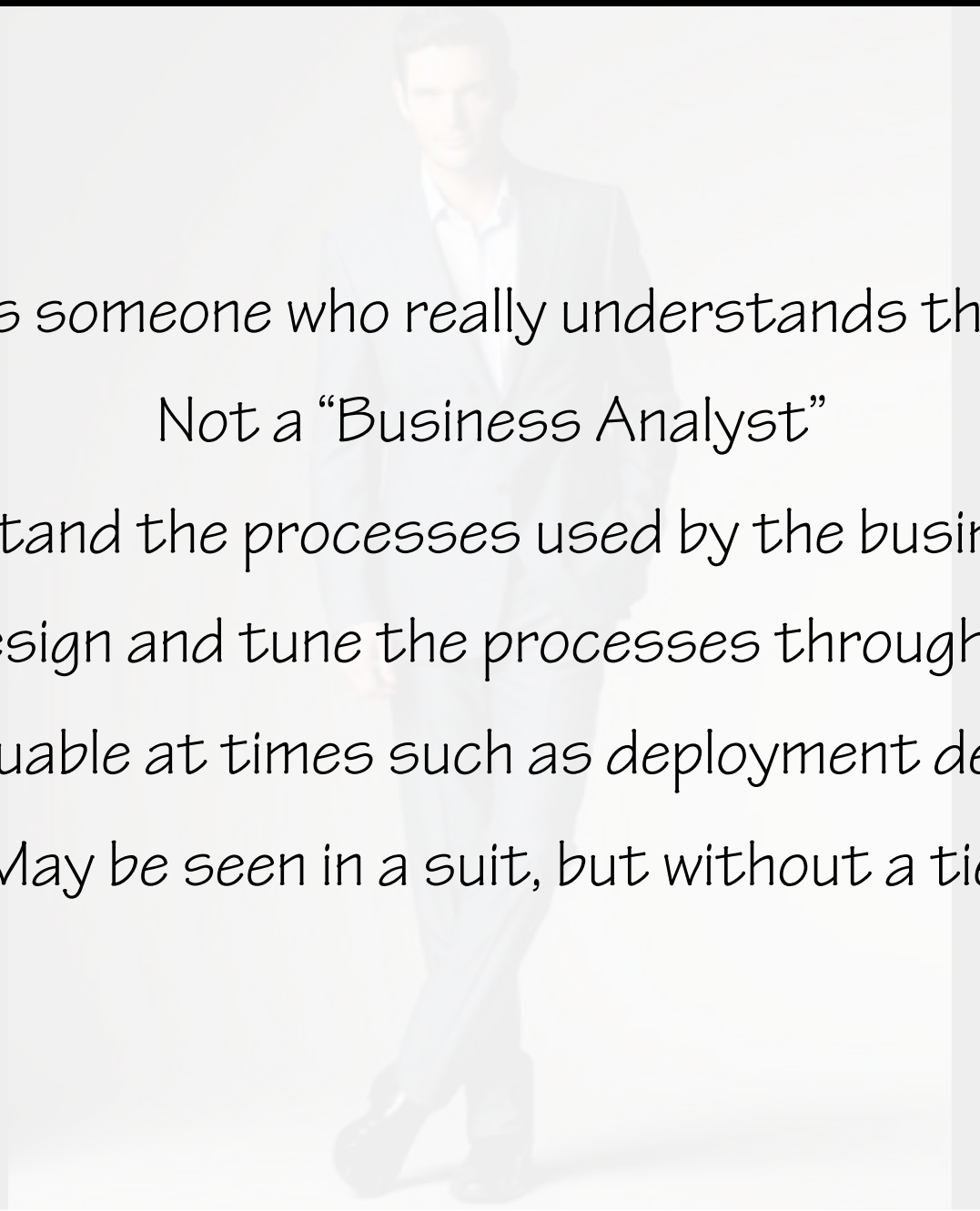
Roles are not titles

Roles are not one-to-one with physical people

And now, on to the fashion show of Agile Architects!

The Business Architect

The Business Architect



Agile needs someone who really understands the business

Not a "Business Analyst"

Must understand the processes used by the business for value

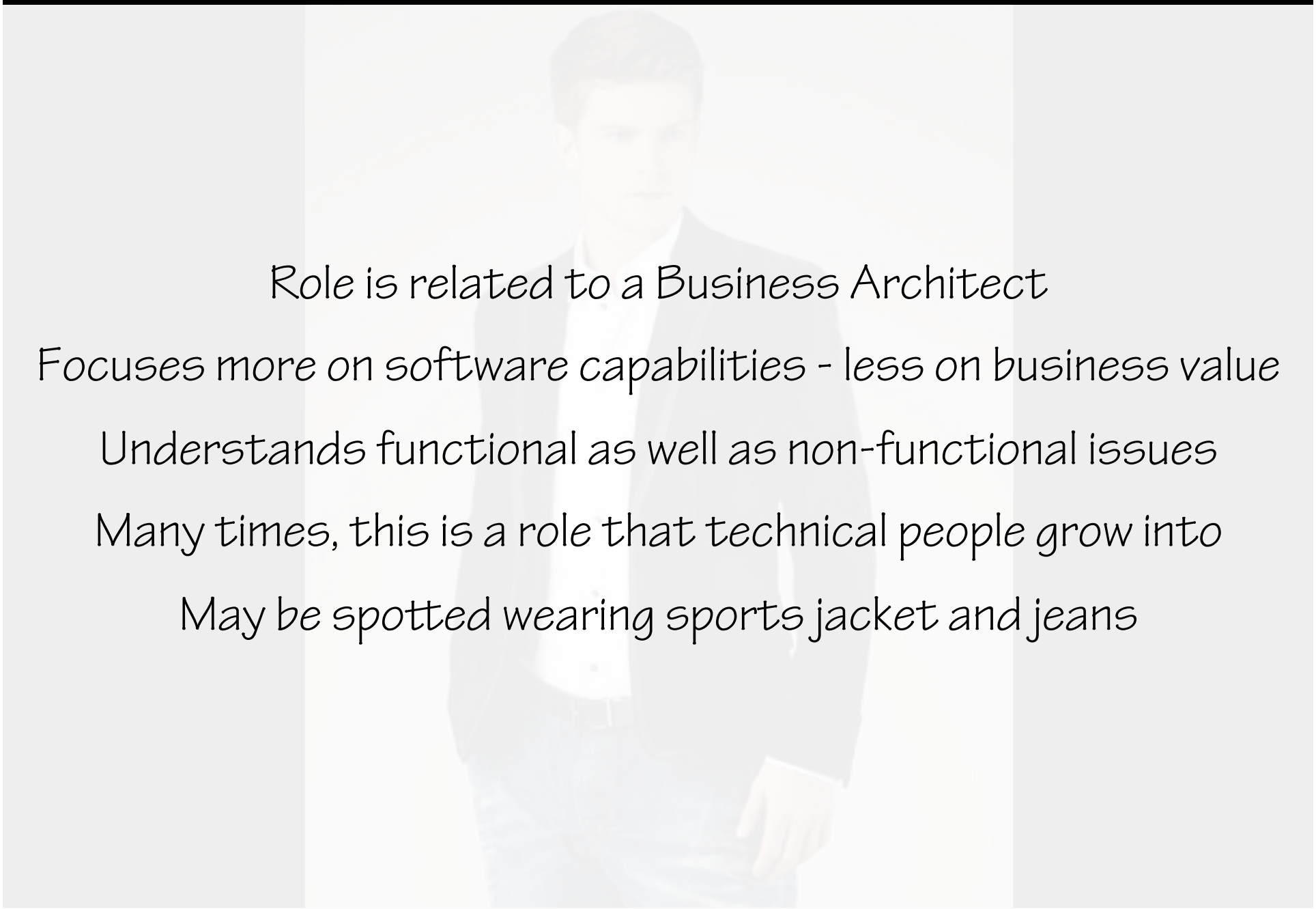
Help to design and tune the processes through software

Invaluable at times such as deployment delays

May be seen in a suit, but without a tie

The Solution Architect

The Solution Architect



Role is related to a Business Architect

Focuses more on software capabilities - less on business value

Understands functional as well as non-functional issues

Many times, this is a role that technical people grow into

May be spotted wearing sports jacket and jeans

The Enterprise Architect

The Enterprise Architect

Concerned with software, frameworks, solutions, and systems as well as business need to not support every commercial or open source system, component, or framework in existence

Differs from solution architect

Solution architect is concerned with understanding common business functionality between components

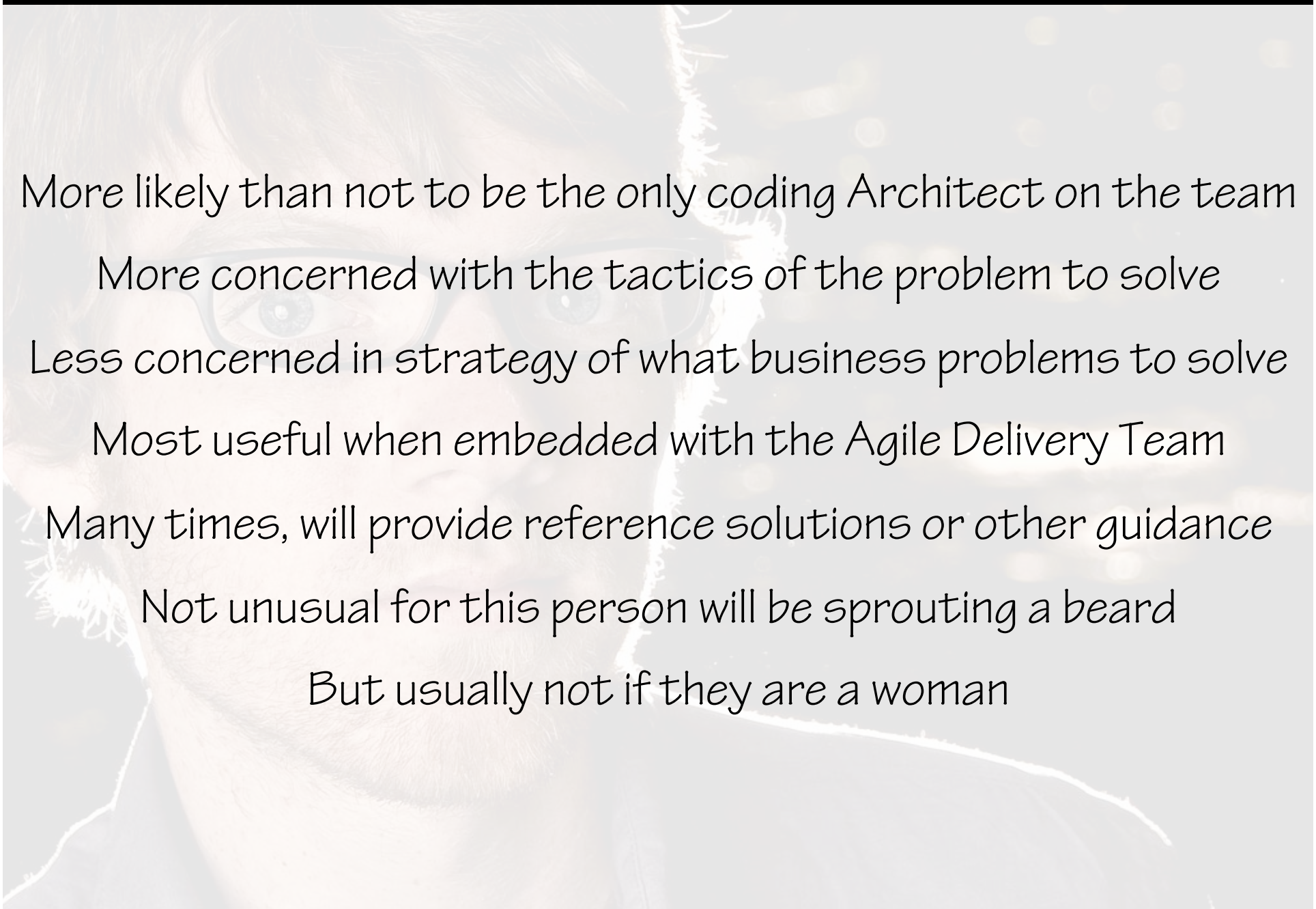
Enterprise architect is concerned with common software functionality between components

Provides strategic guidance to organization

Probably spotted wearing Dockers to work

The Software Architect

The Software Architect



More likely than not to be the only coding Architect on the team

More concerned with the tactics of the problem to solve

Less concerned in strategy of what business problems to solve

Most useful when embedded with the Agile Delivery Team

Many times, will provide reference solutions or other guidance

Not unusual for this person will be sprouting a beard

But usually not if they are a woman

Why Software Architects Are So Necessary

Why Software Architects Are So Necessary

Let's start here, since most of us are probably in that role

Architects help make decisions that we won't regret later

Would like to make decisions once, but that just doesn't work

Delay decisions until the last responsible moment

Develop iteratively and in an Agile/XP fashion

Stay current with components, frameworks, and tools

XP practices help us better reverse effects of a poor decision

Less rework = less waste = faster delivery = lower cost

And Why The Other Three Agile Architects Are Important As Well

And Why The Other Three Agile Architects Are Important As Well

Decisions by Enterprise Architects may take years for effects

Work to undo a poor choice also measured in years, not days

Solution Architects also have long lasting business implications

Need big picture to avoid local optimization

Business Architects are definite big picture people

But don't forget to use them in some thorny tactical uses

Power of manual processes to tide things over for final solution

Architectural Decisions Are Made at Many Levels

Architectural Decisions Are Made at Many Levels

Remember, the 4 Agile Architect roles all work together

But those are not the only decision makers

Architectural decisions are also made right in the code!

Anything with a long lasting effect is an architectural decision

But avoid analysis paralysis – think YAGNI

That's why, in Agile we…

Code a little…

Test a lot…

Validate working software…

Wash, rinse, and repeat!

So, What Makes an Architect Agile?

So, What Makes an Architect Agile?

Respond to change, but with a plan

Shun the BDUF and think Emergent Design instead

Ensure technical excellence

Think concept to cash and join hands with the business

And above everything, aid the team to make their commitments

So, lead your team, follow the money, and get out of the way of progress!

And, Why the Word "Savior" in the Title?

And, Why the Word "Savior" in the Title?

There's a difference between building the right stuff

and building the stuff right

Agile needs both

and Agile Architects help us do both

They save us from ourselves thinking too locally,

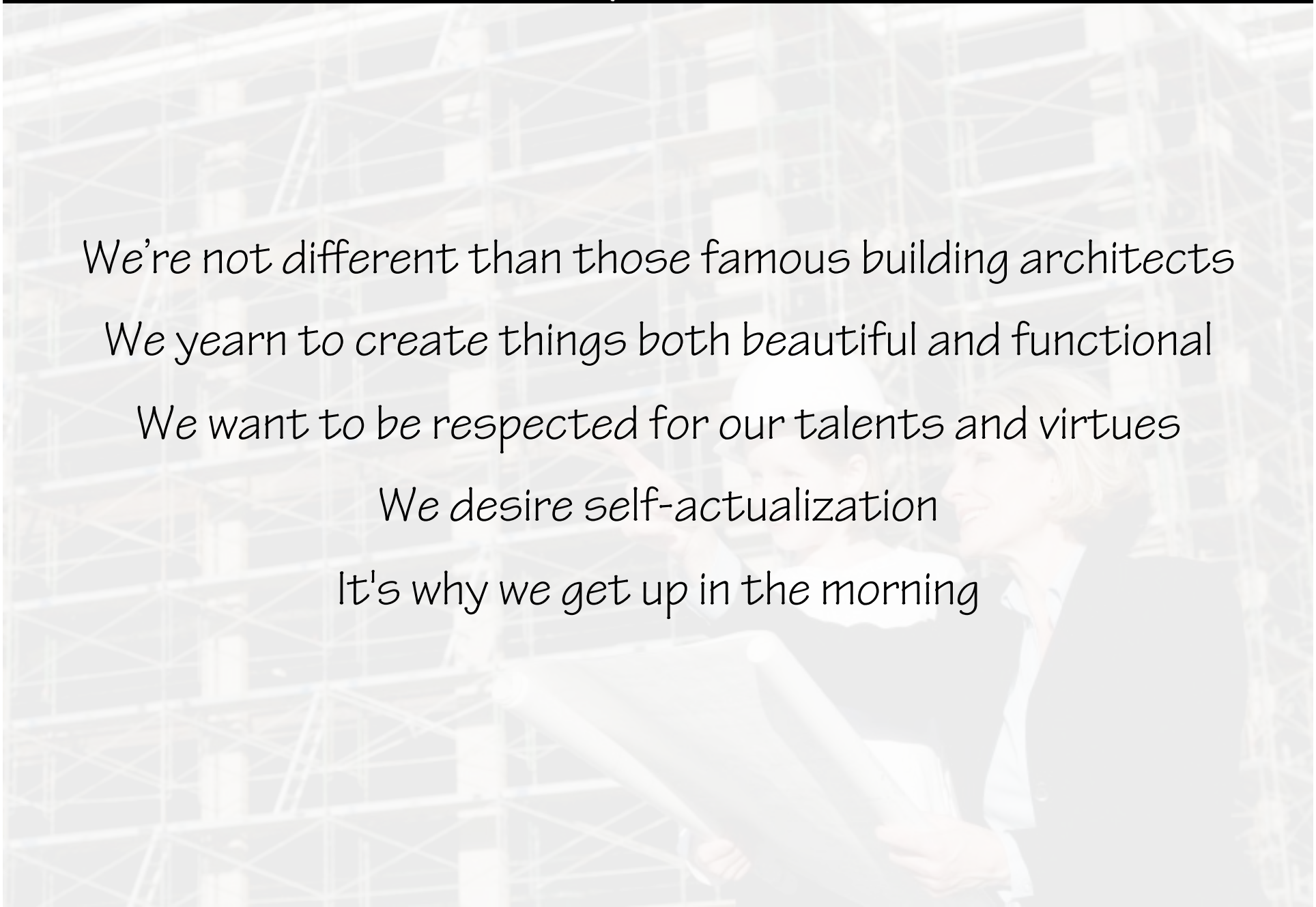
and lead Agile teams into holistic value based solutions

Get some for your Agile teams

Collect them all!

Mommy, When I Grow Up, I Want To Be An Architect!

Mommy, When I Grow Up, I Want To Be An Architect!



We're not different than those famous building architects

We yearn to create things both beautiful and functional

We want to be respected for our talents and virtues

We desire self-actualization

It's why we get up in the morning

Fin

The Agile Architect

Oxymoron or Savior?

Howard Deiner

Agilist. Pragmatist. Craftsman.
Scrum. eXtreme Programming.
Consultant. Coach. Instructor.

+1 (203) 243-1515

howard.deiner@deinersoft.com
http://www.deinersoft.com

Software Architect 2011