

A Science of Programming Language Design?

William Cook, UT Austin
PLATEAU 2012

A person's hands are shown holding a small, clear vial with a white cap. The background consists of vertical blinds, which are slightly out of focus. The overall lighting is dim, creating a moody atmosphere. The text is overlaid on the image in a clean, white, sans-serif font.


Science

create and evaluate
testable models



Design

create artifact
satisfying need or
desire

A photograph of a laboratory or classroom setting. In the foreground, a metal wire cart holds three glass bottles with orange caps, a potted plant, and a stack of white trays. In the background, a person in a dark lab coat is visible. The image is dimly lit and has a dark overlay.

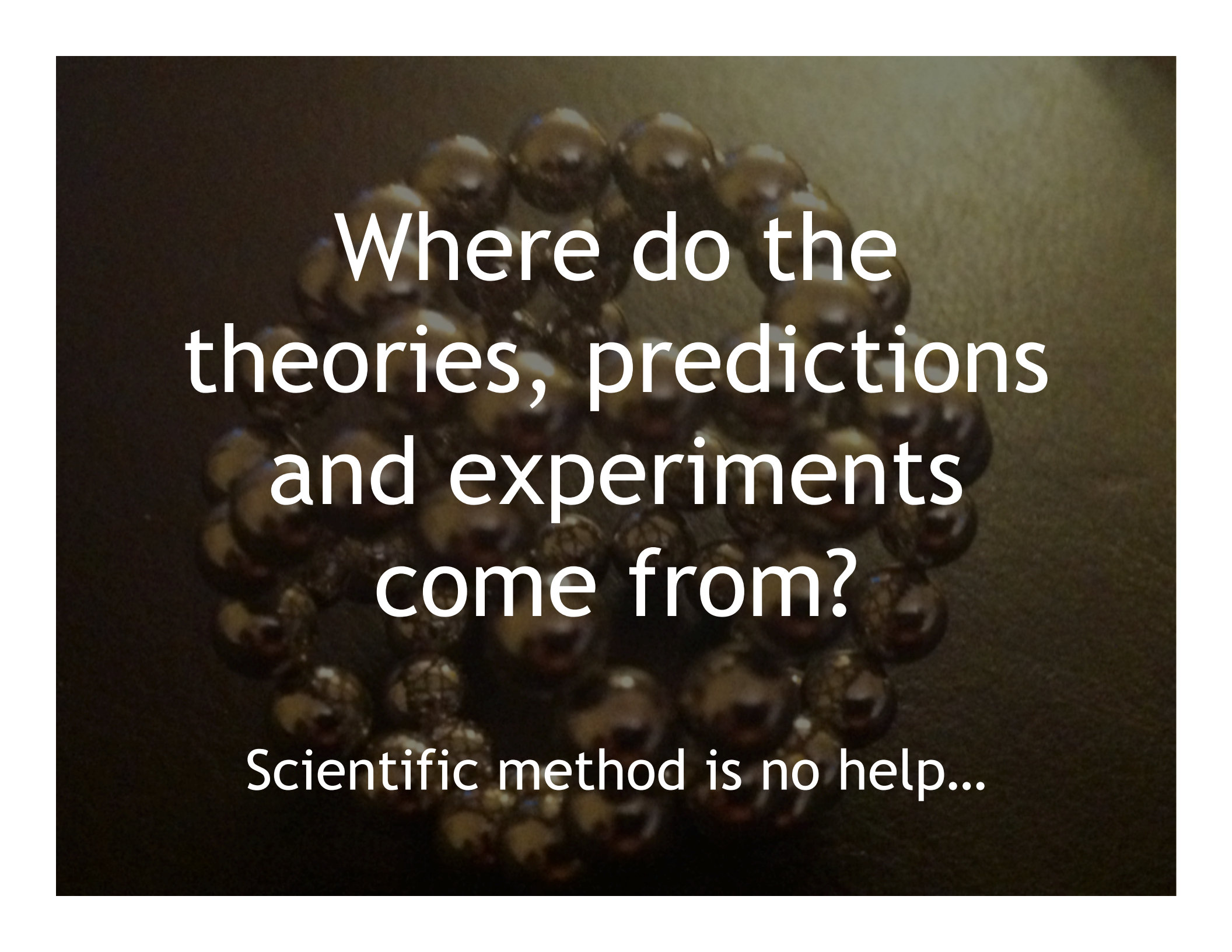
Science
testable model

Design
satisfy desire



Scientific method
is a test plan:

1. predict
2. observe
3. evaluate



Where do the
theories, predictions
and experiments
come from?

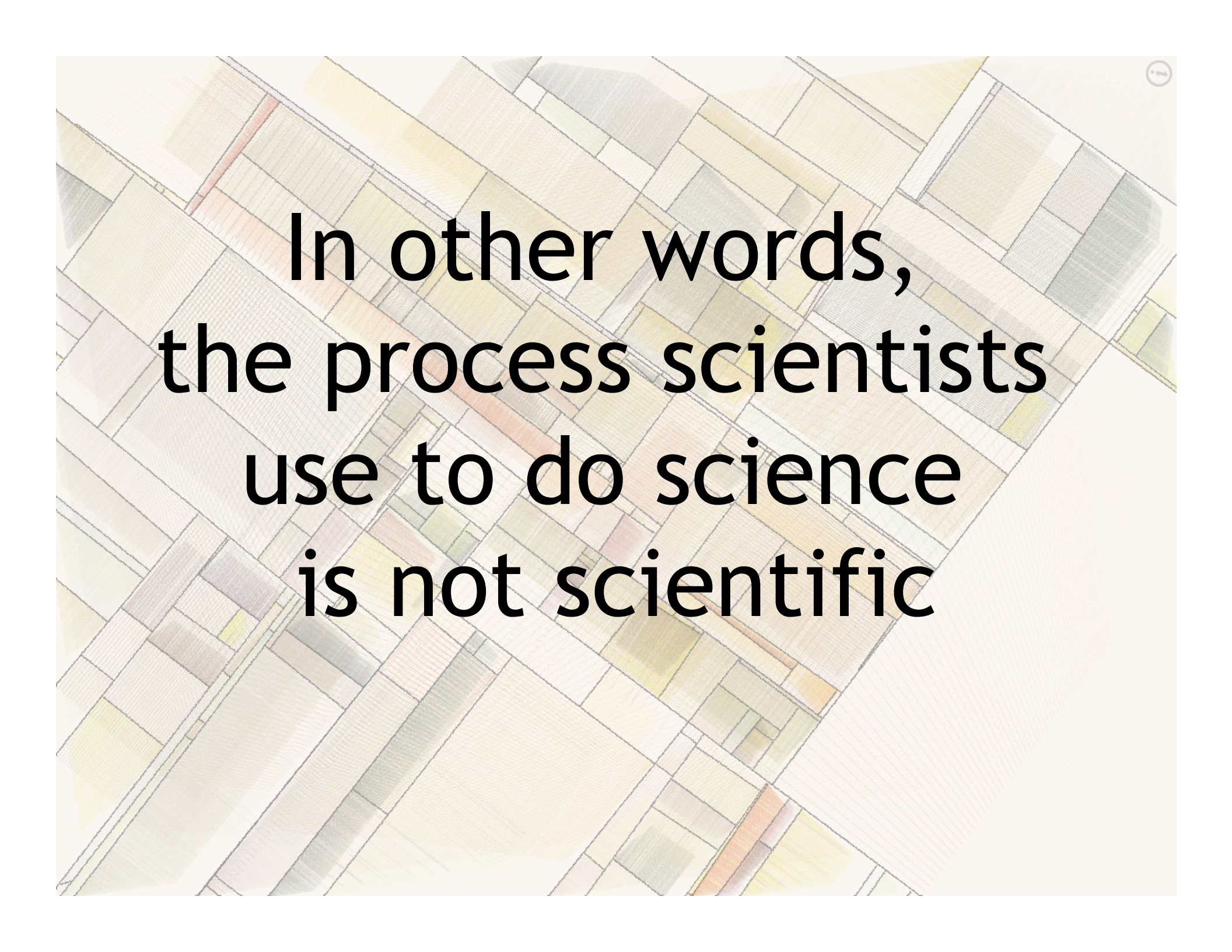
Scientific method is no help...

A close-up photograph of a petri dish containing a bacterial culture. The agar surface is covered with a dense, white, fuzzy growth of bacteria. The lighting is dramatic, with a dark background and a bright, circular light source illuminating the center of the dish, creating a strong contrast and highlighting the texture of the bacterial colonies.

Scientists
are
designers



Scientists
design
theories and
experiments

An aerial photograph of a city grid, tilted at an angle. The streets and buildings are visible, with various shades of gray, brown, and green. A small circular icon with the number '1' is in the top right corner.

**In other words,
the process scientists
use to do science
is not scientific**

SCIENCES OF THE ARTIFICIAL

H. Simon, MIT Press 1969



Optimization Satisficing Search

Artifact and Process

Does this apply to PL?
(I don't think so)



Design is not welcome in academia

survives in *professional* schools:
medicine, law, architecture,
fine arts... elsewhere on fringe

graphs: nodes/edges
labelled
ER
Entities, relationships, attributes
two sides cardinality
"roles"
one
many
required optional
child
man2
Austh
contains
Students location
"coms"
formation Model"
Diagram
SS Diagram

How many algorithms courses are about designing algorithms?

(versus analyzing them)

ID	Person	name	age	haircolor
1	Sue	45	Blonde	
2	Will	20	Brown	
3	Bob	23	Red	
4	Alice	21	Blonde	
5	Reth	-	-	
6	John	-	-	

Status Update
text: string
time: Datetime
author *

motor *

Person
age: Integer
haircolor: String

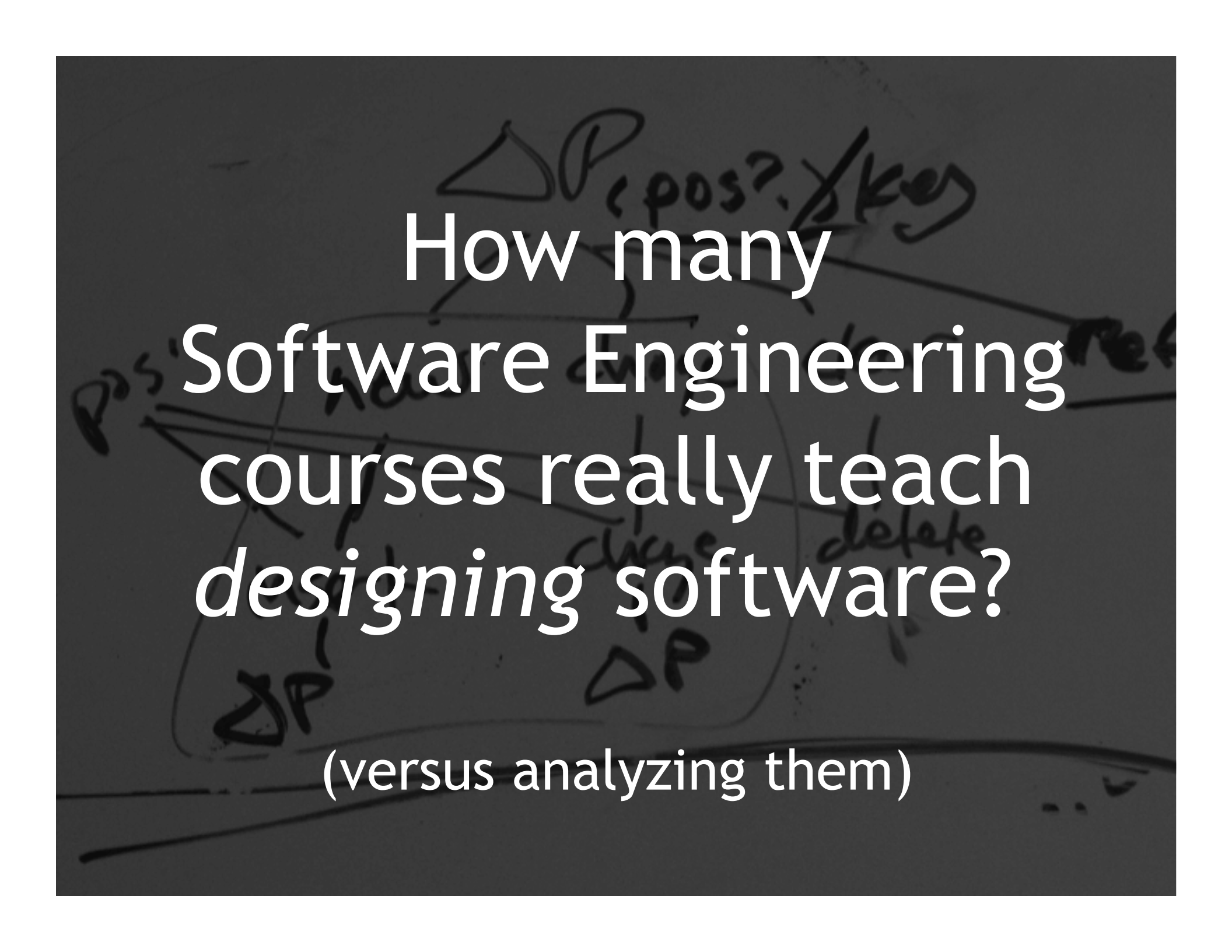
Friend
from to

3 5
3 5
6 5
4 3
7 5 3
7 5 6



How many
PL courses are about
designing PLs?

(versus analyzing them)

The background of the slide is a dark grey or black surface covered with faint, handwritten notes in white or light grey ink. The notes include various terms and symbols such as 'Propos?', 'key', 'Ref', 'delete', 'DP', and 'DA'. There are also some arrows and lines drawn, suggesting a diagram or flowchart. The overall appearance is that of a whiteboard or a piece of paper with technical or programming-related scribbles.

How many
Software Engineering
courses really teach
designing software?

(versus analyzing them)



Not Repeatable

Many design
problems are unique

A man with glasses and a light-colored button-down shirt stands in front of a chalkboard. The chalkboard is filled with faint, hand-drawn diagrams consisting of squares, circles, and arrows, suggesting a design or engineering process. The text is overlaid on the image in a large, white, sans-serif font.

Not always objective

Design cannot be
defined in a textbook
and taught in a
lecture class



Often
Human Centered
Evaluation
involves humans
(are they satisfied?)



but...

Generalize over values

- Add a new parameter to a function

We do teach design:

```
one x = f x  
two x = f (f x)  
three x = f (f (f x))
```

PhD supervision!



Apprenticeship

Practice
Critique
Reflect



How do we know
good design?



Good Design

Satisfies the human
desire or need

A man with grey hair, wearing a dark suit jacket over a white shirt, stands in the foreground. He is smiling and has his arms slightly outstretched. The background is a scenic view of a town with a prominent church featuring a red-tiled roof and a tall spire. In the distance, there are mountains under a hazy sky. The overall lighting is soft, suggesting dusk or dawn.

easy to use

high-performance

maintainable

elegant

internally consistent



Objective

high-performance

internally consistent

Intermediate

maintainable

easy to use

Subjective

elegant

Wicked Problems

No test for solutions

Cannot enumerate possible solutions

Every problem is unique, no learning

Defining "wicked problem" is a
wicked problem



My Take

Many things we
really care about...

are not
easy/possible
to measure

Industrial
experimentation
is our current
evaluation
mechanism

Academia should
embrace design

Spectrum of Criteria

Objective



Subjective

Allow...
discussion
of entire spectrum

User Studies
Repository Mining

are great
but not only options

Need to expand the
range of acceptable
"tests" for validity

Acceptable Evidence

- Controlled User Study
- Case study
- Historical data mining
- Reasoned argument
- Benchmark design problem
- Structured critique
- Detailed comparisons

Call to Action:

Formalize
PL design paper
review criteria

Other terms besides
"scientific"

Academically rigorous

Scholarly

IFIP
Working Group 2.16
on
Language Design

approved last year

A photograph of a modern living and dining area. In the foreground, a dark brown leather sofa sits on a patterned rug, accompanied by a matching ottoman. A red and yellow cushion is on the sofa. To the left, a dining table with chairs is partially visible. In the background, a kitchen with dark wood cabinets and a stainless steel refrigerator is shown. A person is standing at the kitchen counter. The room features light-colored wood flooring and large windows with decorative panels. The text "design case studies" is overlaid in white on the image.

design
case
studies

AppleScript

- We did do user studies
- Weren't sure how to do it!
- They didn't influence the language much
- We still ended up with "partial success"

Understanding

Objects

First-class behaviors (dual of ADTs)

Inheritance

Open recursion (not just for objects)

See R. Gabriel "The Structure of a Programming Language Revolution", Onward! 2012

Semantics

Denotational over operational

Operational wins

typing proofs

concurrency

Featherweight Java tells you what
inheritance does, not what it means

The PL Wars

No sub-discipline of CS is so
fundamentally at war with itself
(FP, OO, MDD)

Laughingstock?

Motivation?

(see understanding)

Choose Good Examples

Remote Method Call

```
local.print( remote.proc(inputs) )
```

Conclusions:

- marshall data

- create remote proxies

- serialize objects

Choose Good Examples

Multiple Remote Method Calls

```
local.print( remote.proc1(inputs1) )
```

```
local.print( remote.proc2(inputs2) )
```

Conclusions:

- send multiple calls to server at once

- bulk transfer of inputs and results

- no serialization, no proxies

- "batches" include conditionals and loops




Hybridize

Object Algebra

Unify Factories and Visitors

ECOOP 2012 w/Bruno Oliveira



Ensō
(motivation)

with Tijs van der Storm
Alex Loh (see Onward! 2012)

Spectrum of Programming



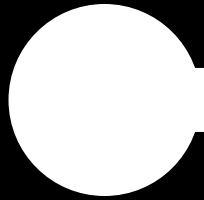
How

implementation

What

Specification

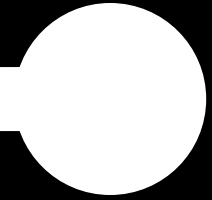
asm, C



How

implementation

Z
CASL



What

Specification

Programming
Languages

Z
CASL



How

What



Verification

Synthesis

Programming
Languages

Z
CASL



How

What

Verification

[Note]

Programming Languages
Grand Challenges panel didn't
even mention synthesis

Synthesis

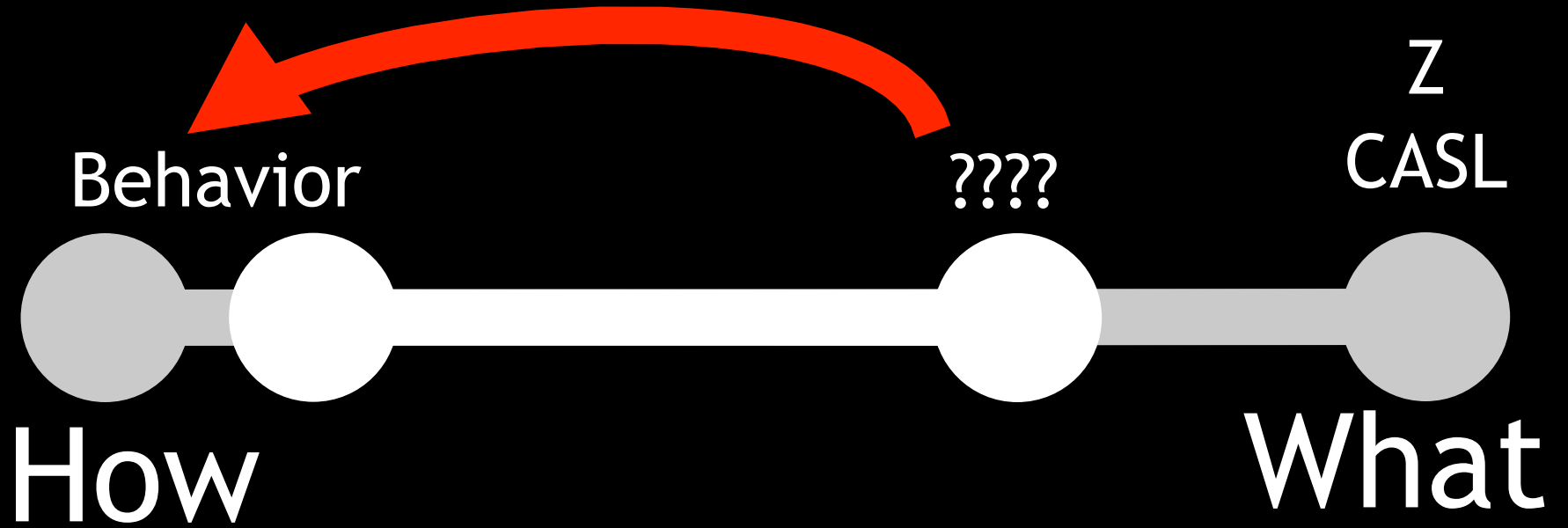
Programming
Languages

Z
CASL

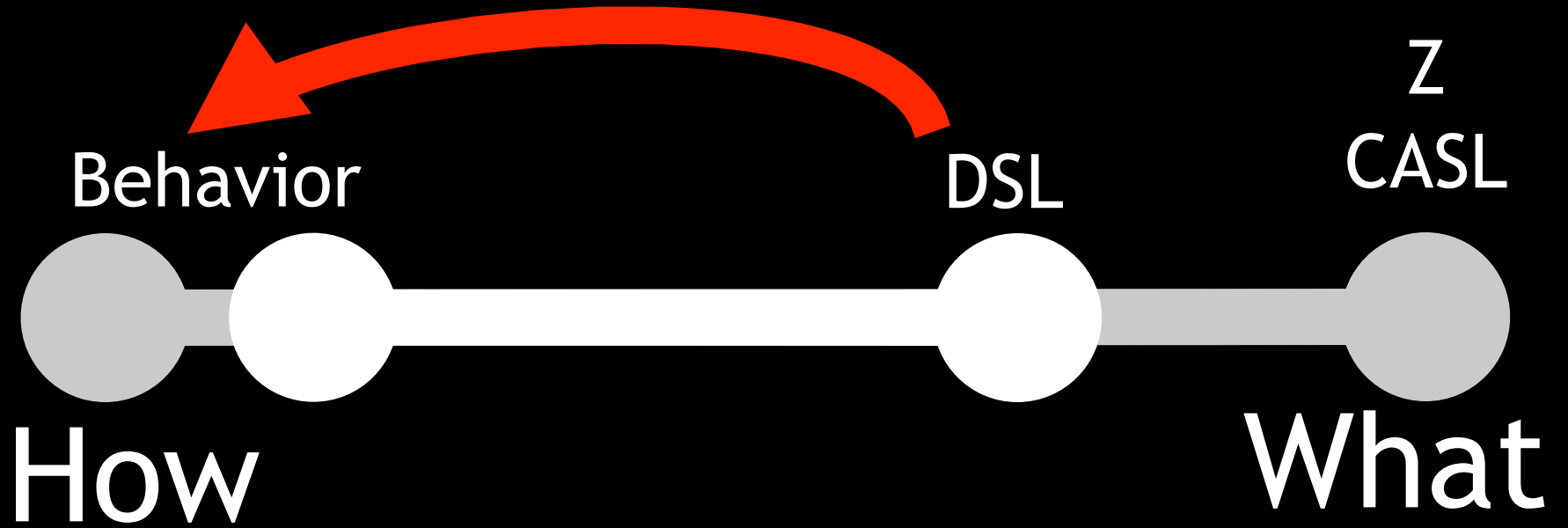


Verification
Lite

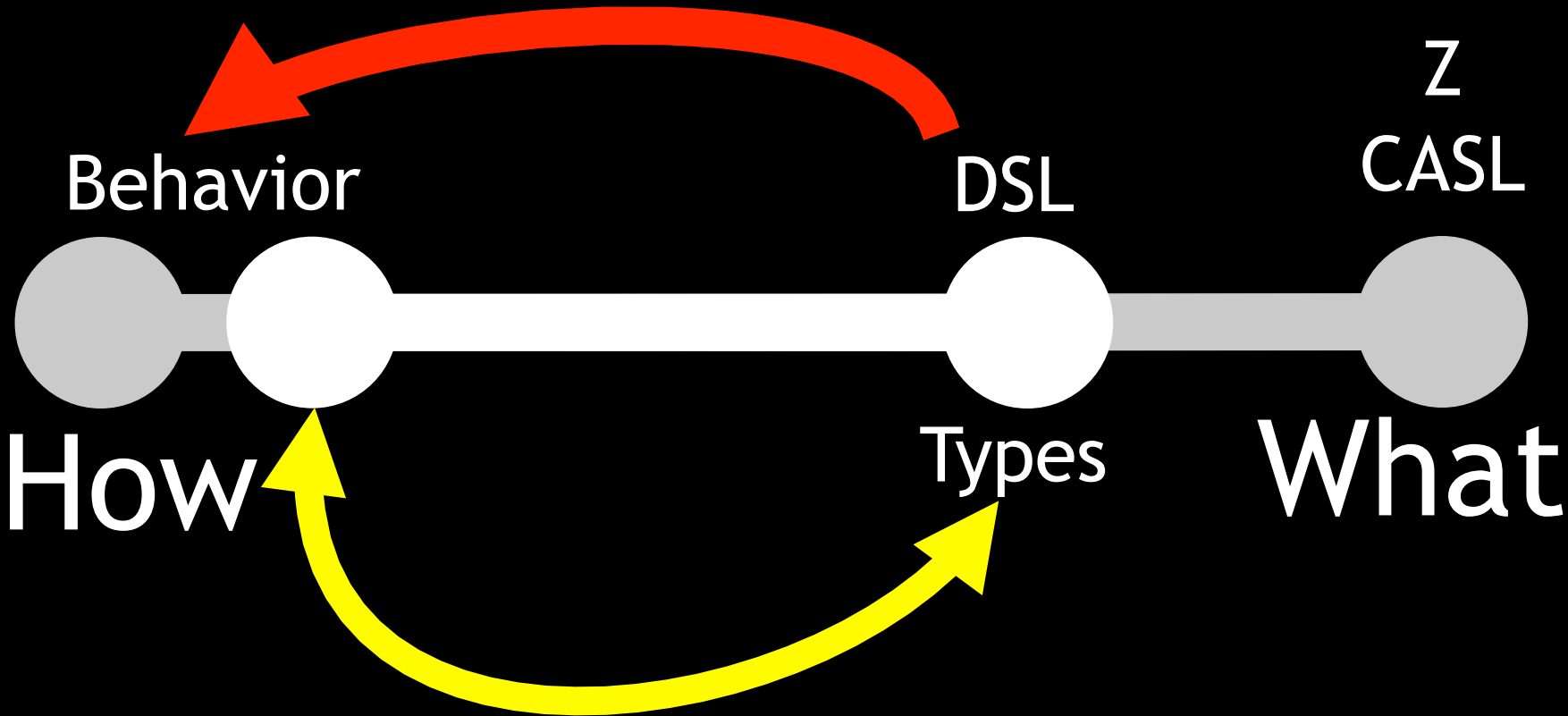
Synthesis Lite



Synthesis Lite



Synthesis Lite



Verification Lite

Ensō Plan

Integrate and Extend DSLs

Standalone, not embedded

Interpret, not compile/transform

Graphical + Textual

Partial evaluation for speed

**Data, Grammars, Security, Workflow,
Diagrams, GUIs, WebUI, Synchronization**

Prevent Bad

Enable Good

Bug Finding
Race Detection
Type Checking
etc.

Prevent Bad

Enable Good

Bug Finding
Race Detection
Type Checking
etc.

Prevent Bad

Enable Good

New languages?
New features?
For what?

Bug Finding
Race Detection
Type Checking
etc.

Prevent Bad

Advantages:
Measurable
Domain-free

Enable Good

New languages?
New features?
For what?

If somebody comes up
with the next big
thing after objects...
all bets are off

Lets try to do this!

simplicity
is the
result
of hard work

Embrace Design

Don't fall prey to
"science envy"

academic rigor
not rigor mortis

Don't Design
Your Programs

Program Your Designs