

# What Zope did wrong

(and what to do instead)

**Lennart Regebro**

EuroPython 2007, Vilnius

Zope is zuper!

First!

Object oriented!

Python!

Open source!

Batteries included!

Secure!

Easy!

And many other exclamation marks!!!

What Zope2 did right

Used Python

ZODB

DTML/ZPT

Batteries included

Easy entry into development

# Zope2: The dead-ends

The ZODB pile of scripts

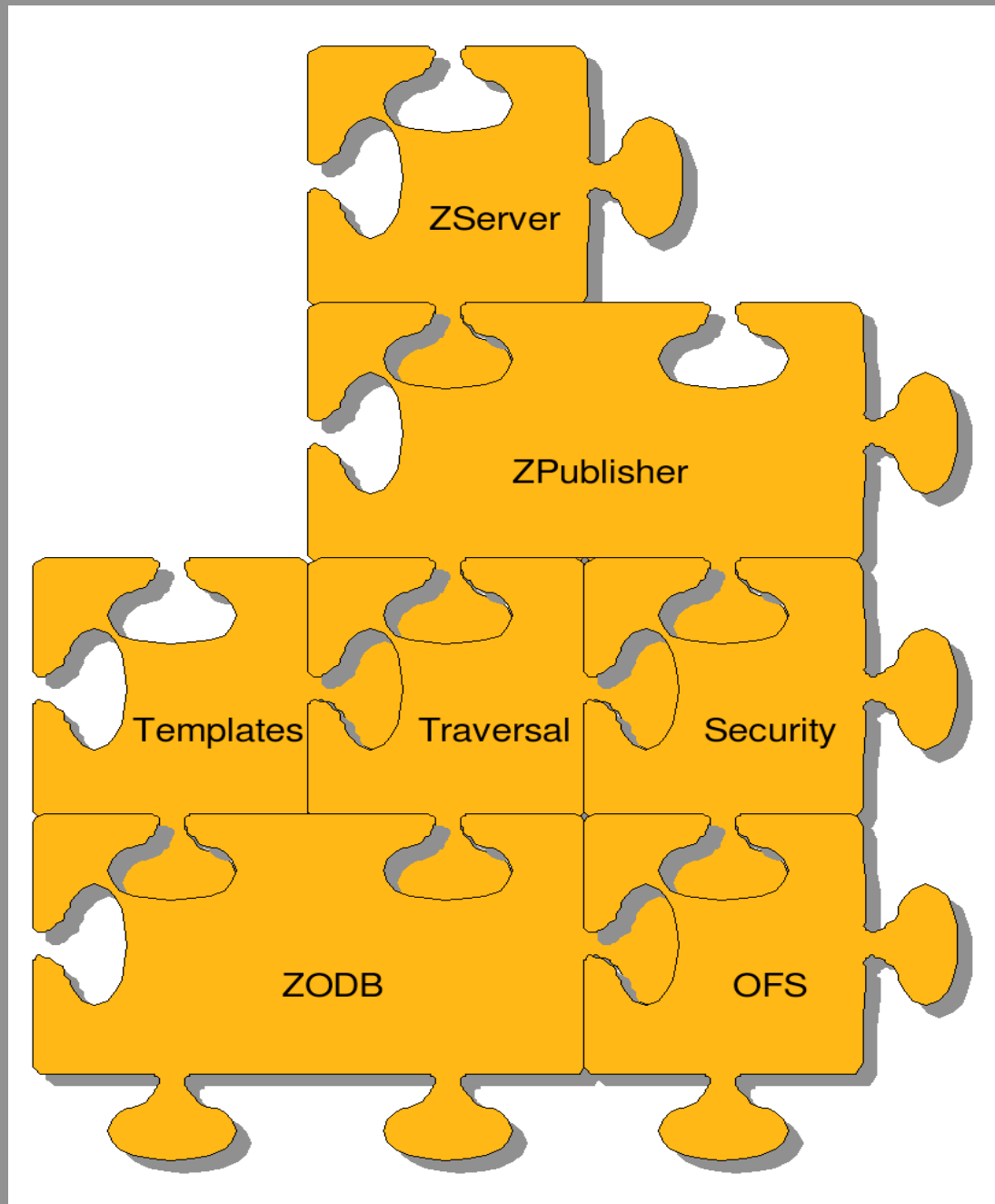
ZClasses

Disk-based products



Starts off easy

Never ends



It's unpythonic!

Products instead of modules

Way to much magick!

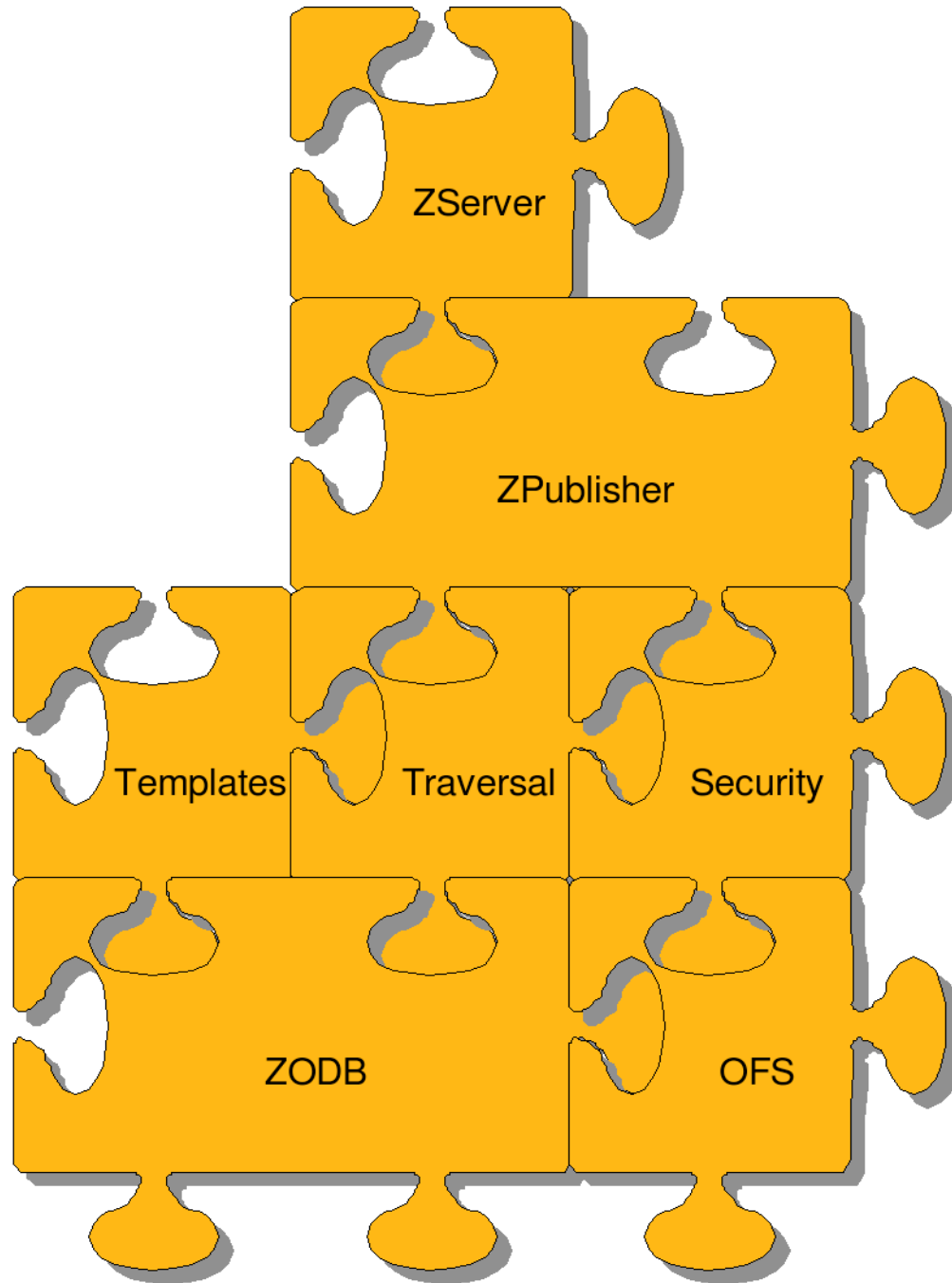
Zope is the Application  
(not the library)

Maybe more?

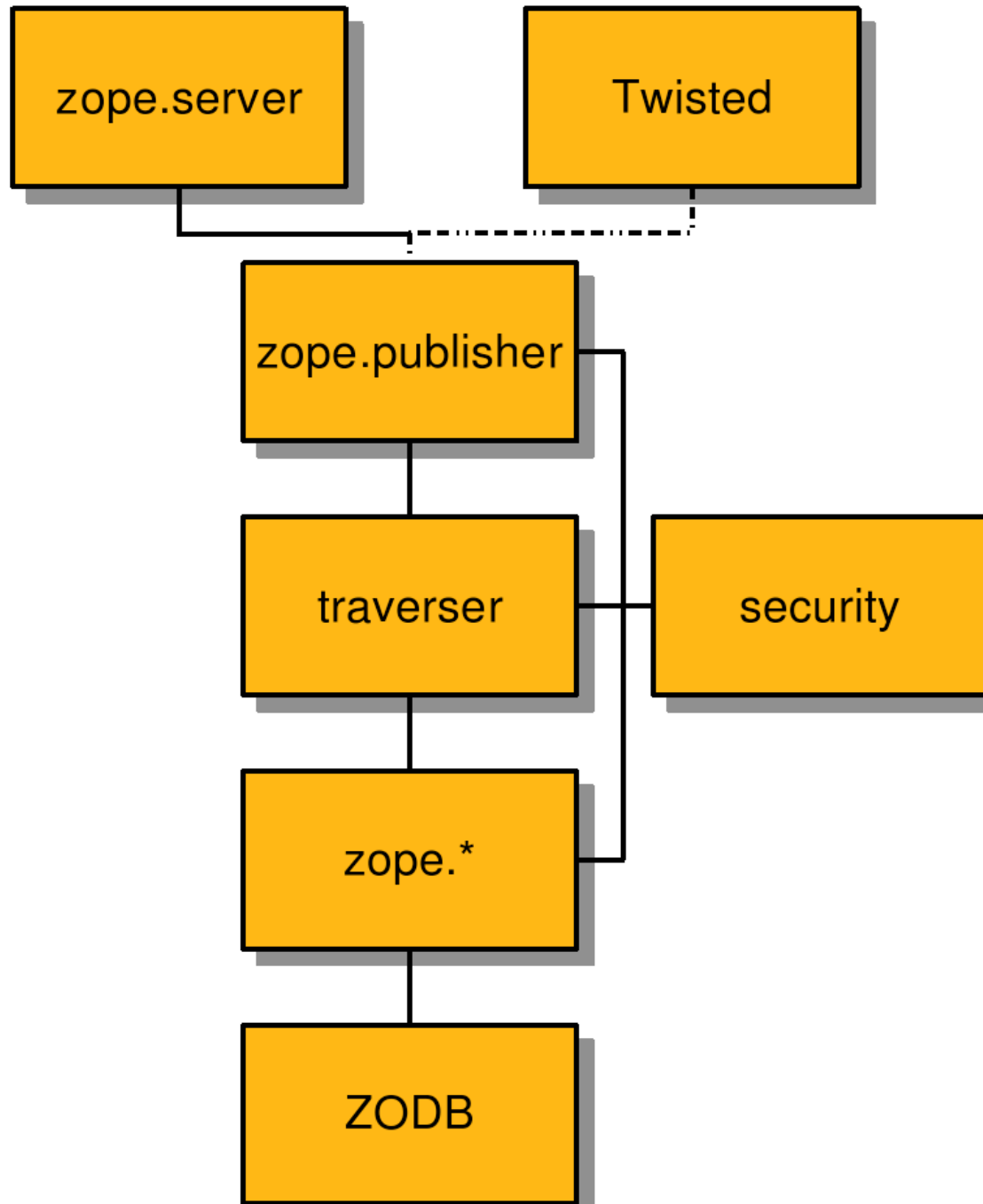
Zope 3: Knight in shining armour!



# Zope 3?



# Zope 3!



To complicated

```
<configure
  xmlns="http://namespaces.zope.org/zope"
  xmlns:browser="http://namespaces.zope.org/browser"
  xmlns:hello="http://namespaces.zope.org/hello">

  <content class=".hello.Hello">
    <require
      permission="zope.Public"
      interface=".interfaces.IHello"
      set_schema=".interfaces.IHello"
    />
  </content>

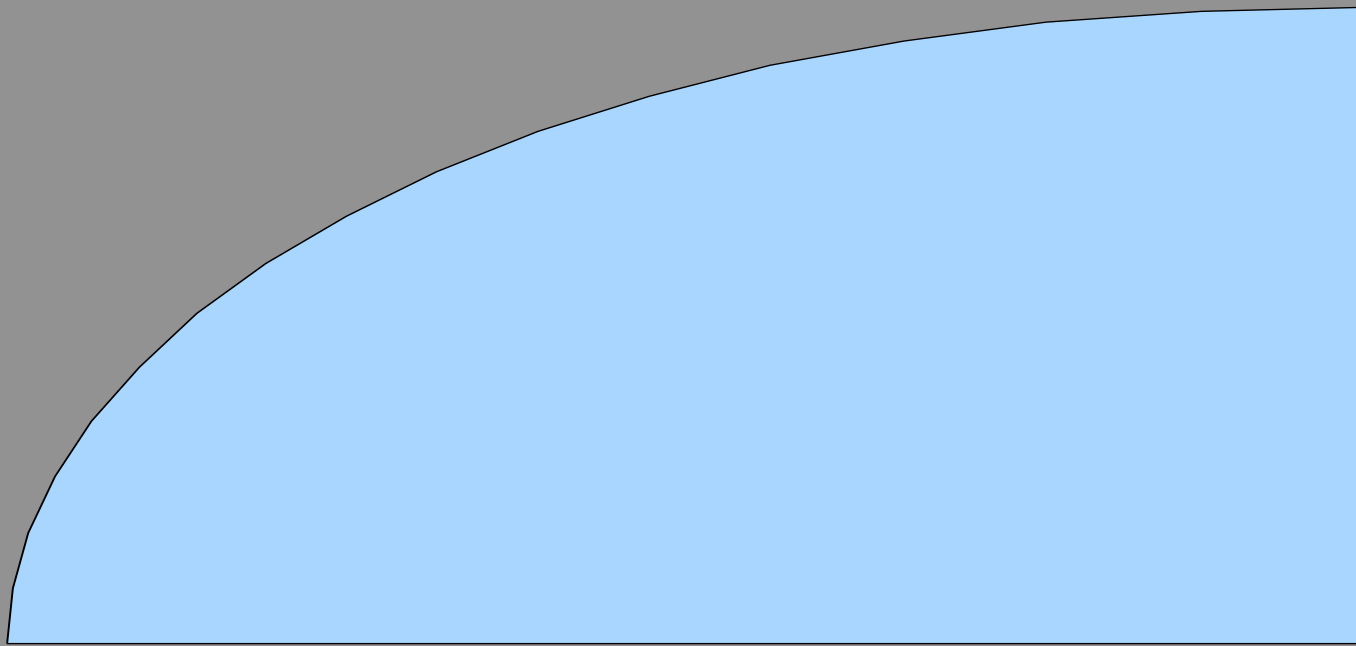
  <browser:defaultView
    for=".interfaces.IHello"
    name="edit.html"
  />

  <browser:addMenuItem
    class=".hello.Hello"
    title="Add Hello"
    permission="zope.Public"
    for="*"
  />
```

Ugh..

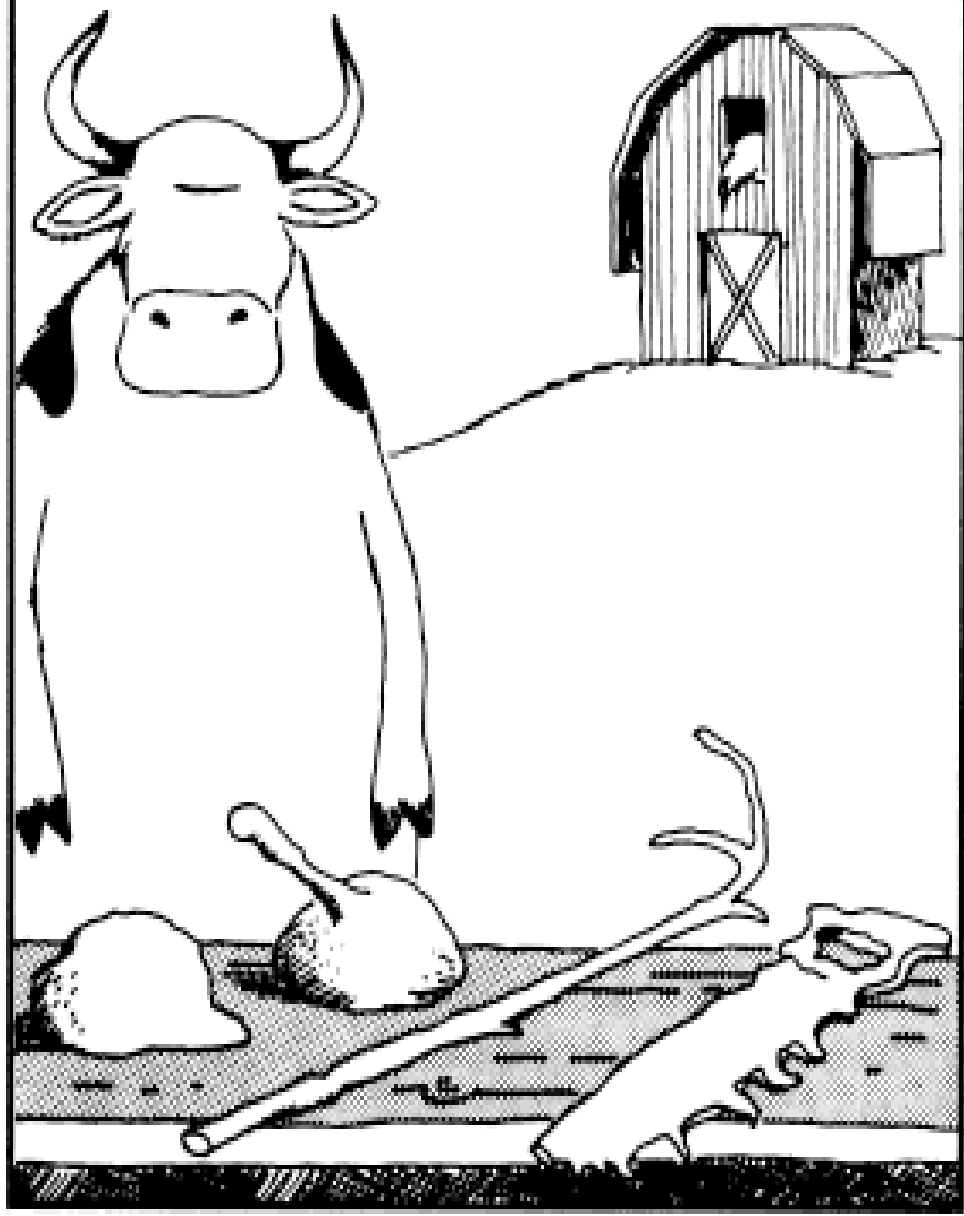
.

Phew...



Zope3: Death by abstraction

Larson



It's unpythonic!

XML

(So, not so unpythonic as people think)

In J2EE:

- A mail service API
- Implementation neutral
- Req: J2EE
- Not web-only

The JavaMail API!

## In Zope 3:

- A mail service API
- Implementation neutral
- Req: components
- Not web-only

zope.app.mail

## Zope 3: The long march

Development of Zope 2 slowed down

Documentation no longer updated

A general waiting for Godot

Backwards compatibility

Didn't happen

Forwards compatibility

Not Yet



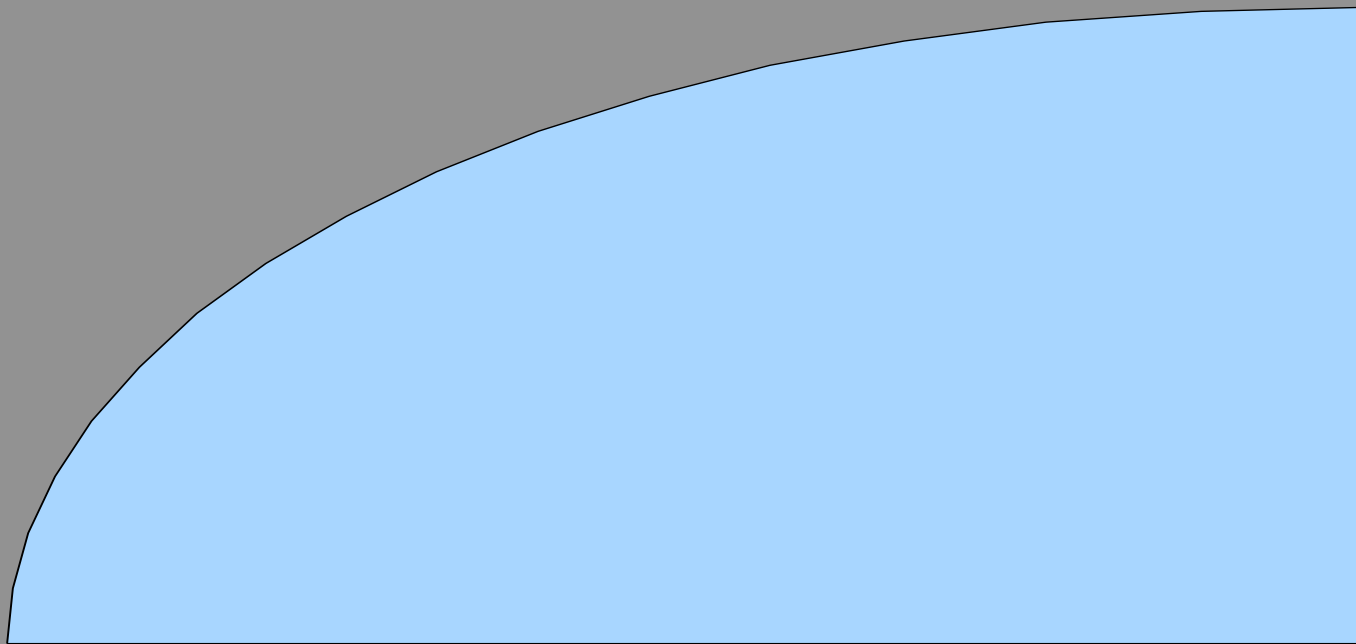
Starts off easy

Never ends

Ugh..

.

Phew...

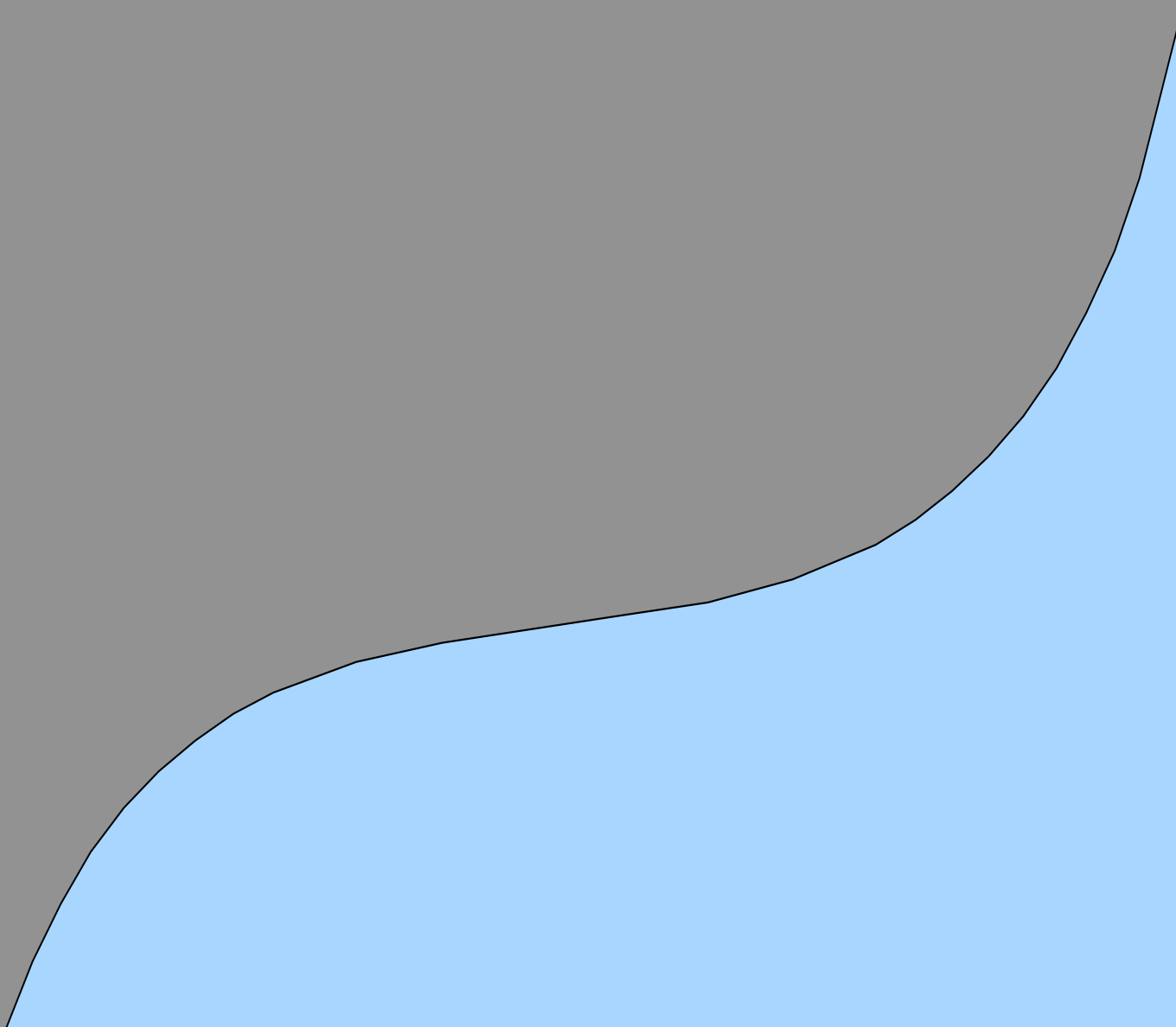


2 + 3 = Five

Argh!

Ugh..

.



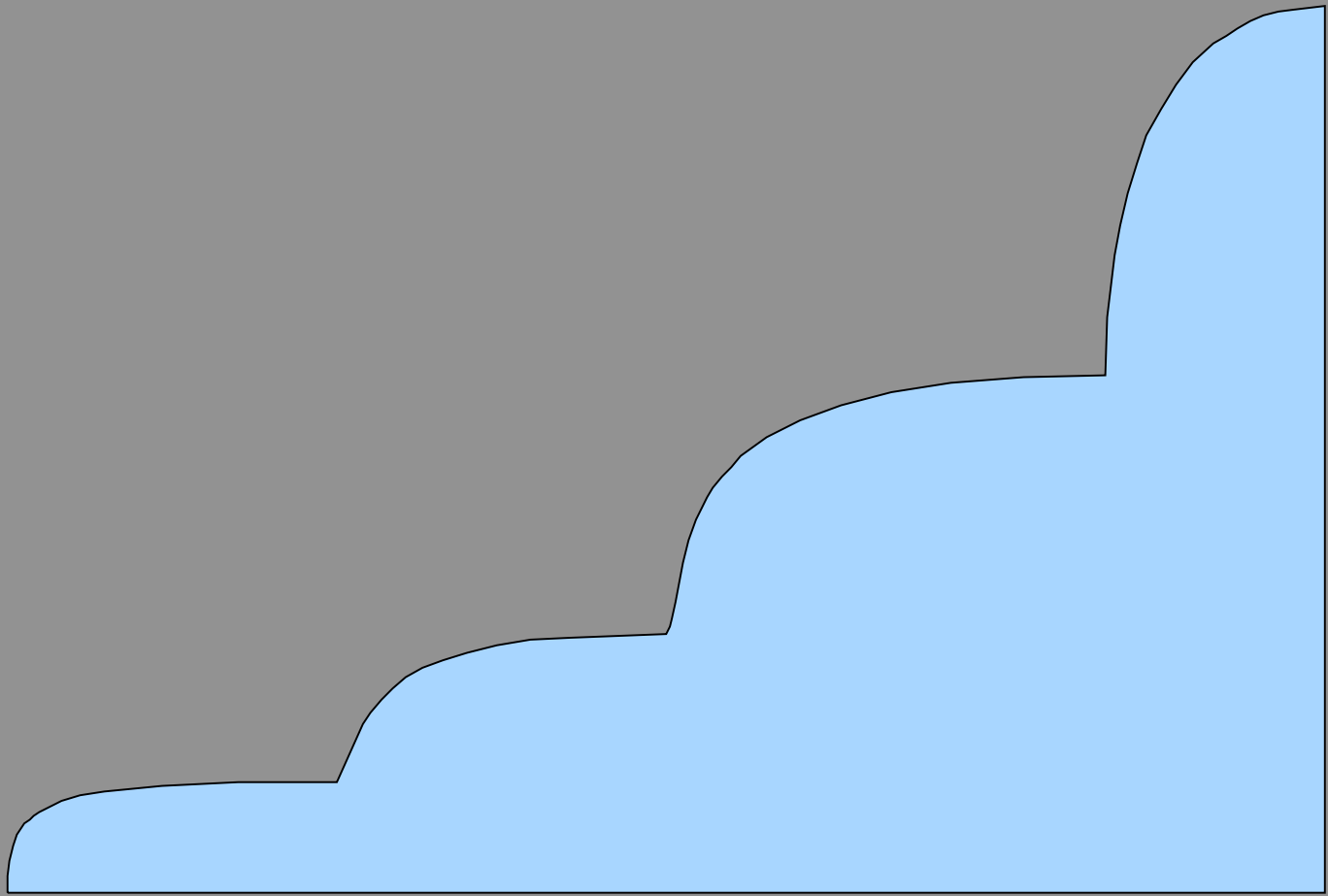
Low entry threshold

and

all the power of a big  
framework

without

dead ends?





Information Manager  
IT department

TTW Layer

GUI schema/form editor  
GUI workflow editor

Site developer

Site Layer

A CMS, ERP or similar

Application developer

Application Layer

A Web application server  
Application Layer API

Core developer

Core Layer

Language  
Component Architecture  
Loads of components

## ▣ Pythonic

- Use existing modules and APIs if possible
- The whole application framework is a library

## ▣ Highly modular

- Pick and choose from the modules
- Modify and replace internal components

Too much freedom?

Start with a finished application!

How can we accomplish this?

Python

A setup framework

# Zope Component Architecture

Zope 3 the libraries

Grok

Information Manager  
IT department

GUI Tools

Site developer

The CMS/ERP

Application developer

Grok

Core developer

Component  
Architecture