

What Zope did wrong

(and what to do instead)

Lennart Regebro
EuroPython 2007, Vilnius

Zope is zuper!

- First!
- Object oriented!
- Open source!
- Python!
- Batteries included!
- Secure!
- Easy!
- And many other exclamation marks!!!

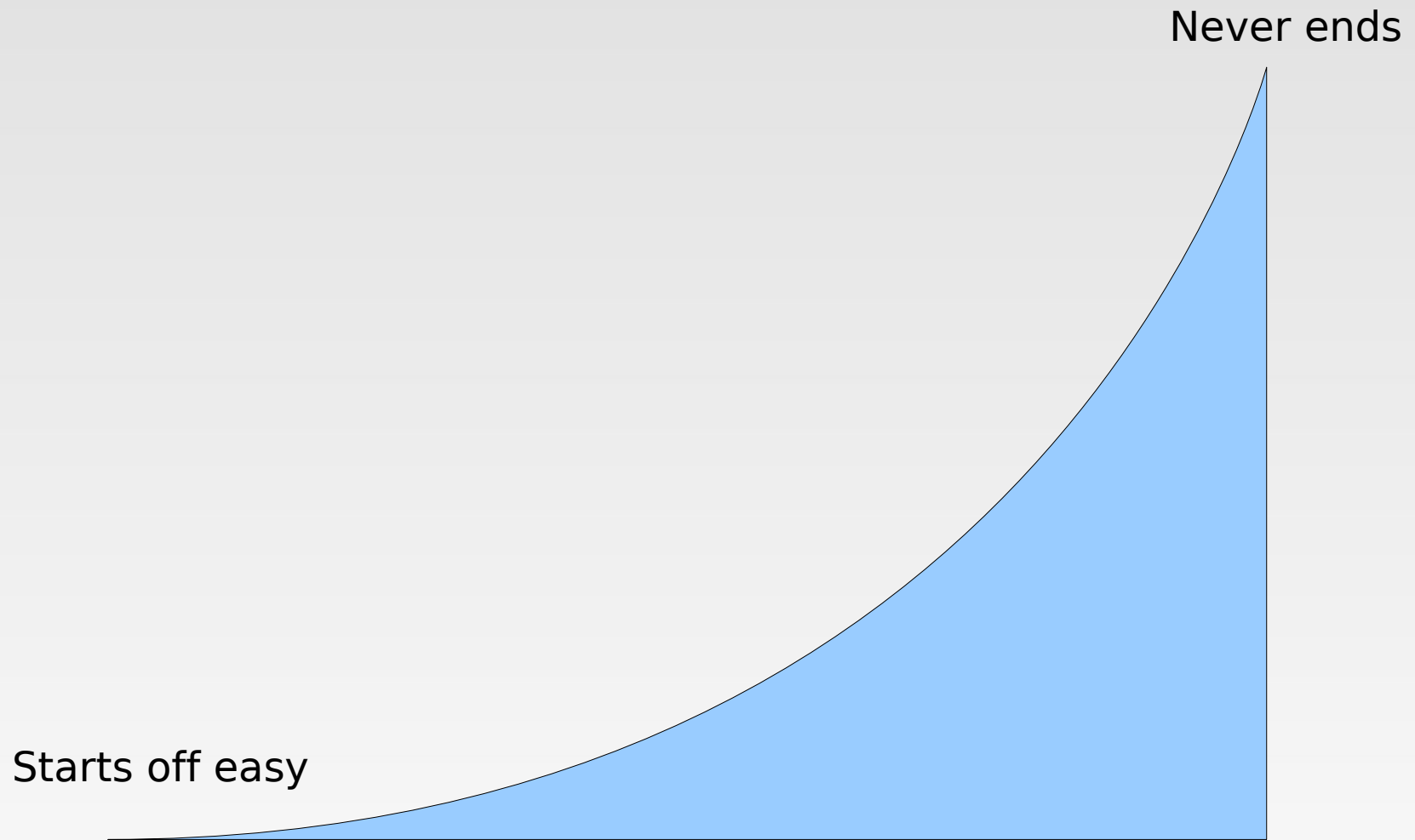
What Zope2 did right

- Used Python
- ZODB
- DTML/ZPT
- TTW development
- Easy entry into development

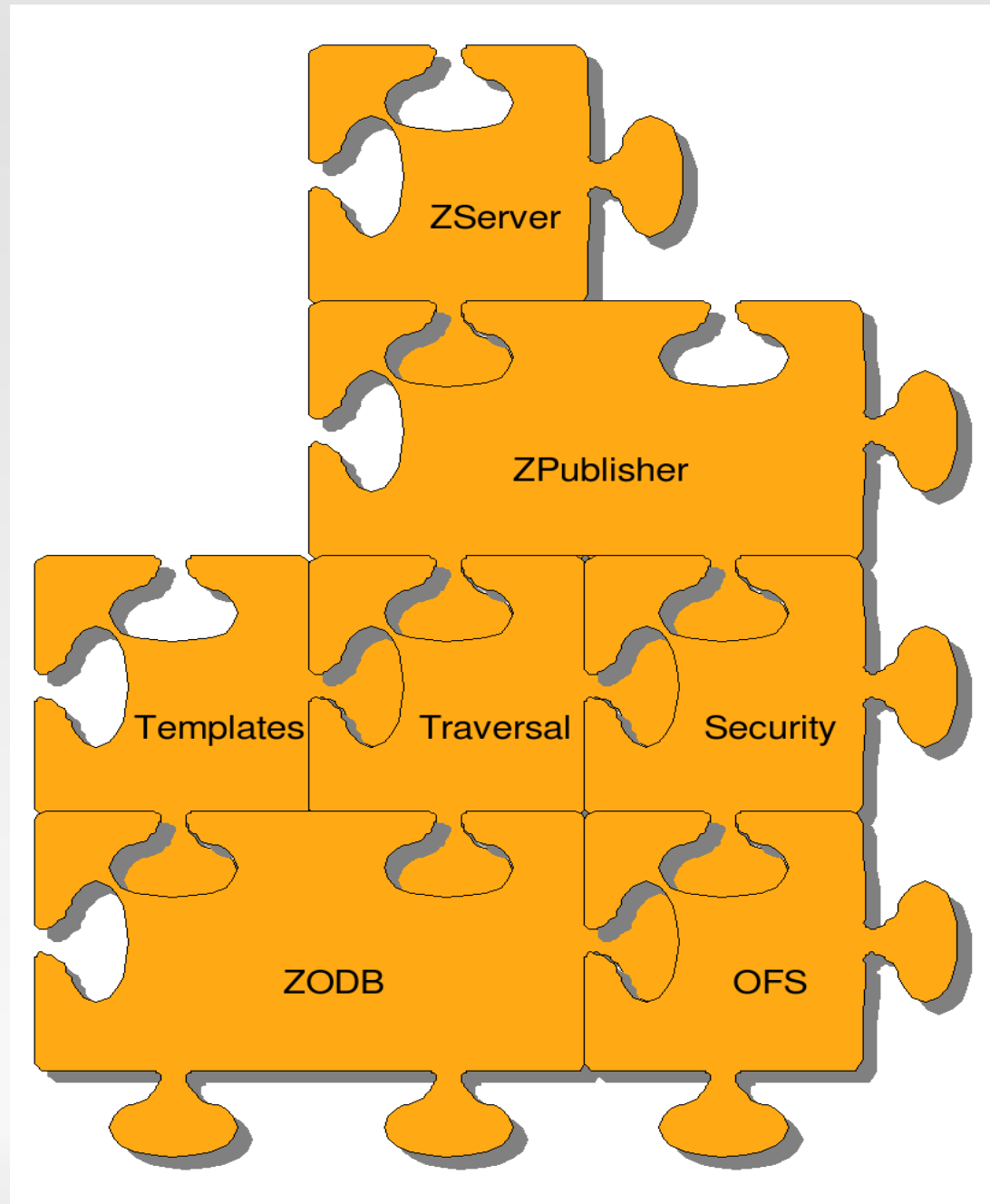
Zope2: The dead-ends

- The ZODB pile of scripts
- ZClasses
- Disk-based products

Zope2: The learning curve



Zope2: The monolithic lock-in



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!



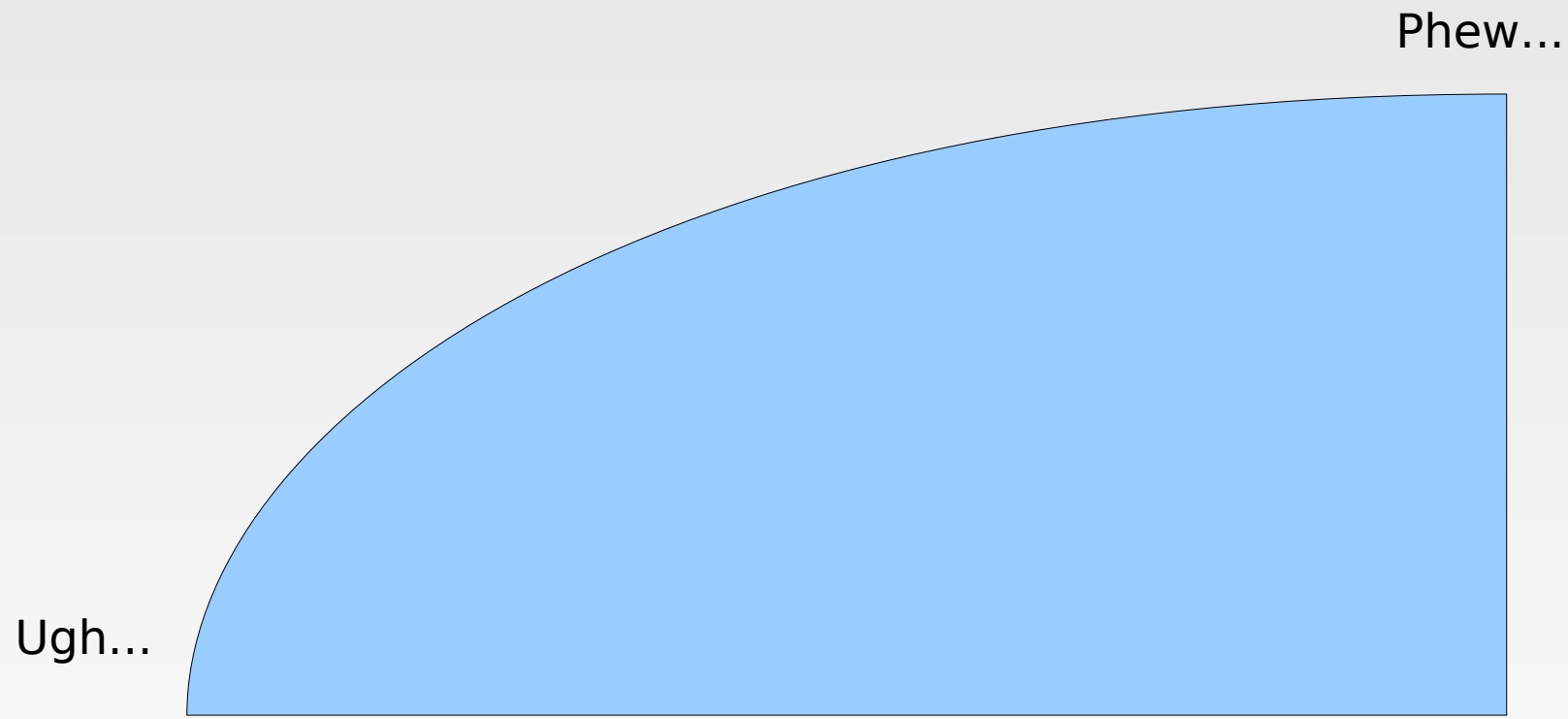
Zope 3: The long march

- Development of Zope 2 slowed down
- Documentation no longer updated
- A general waiting for Godot

Backwards, forwards or not at all

- Backwards compatibility
 - Didn't happen
- Forwards compatibility
 - Didn't happen
 - (Well, yet, at least. Could still happen)

Zope3: The learning curve



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="*"
  />

</configure>
```

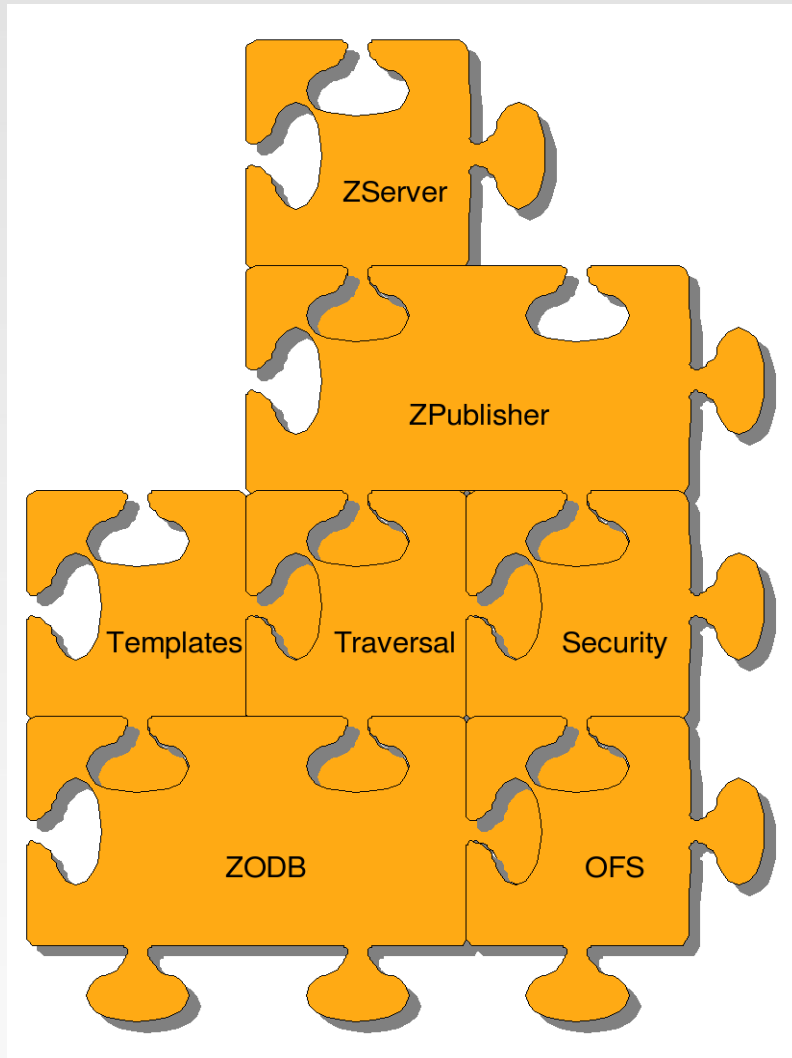
It's unpythonic!

XML

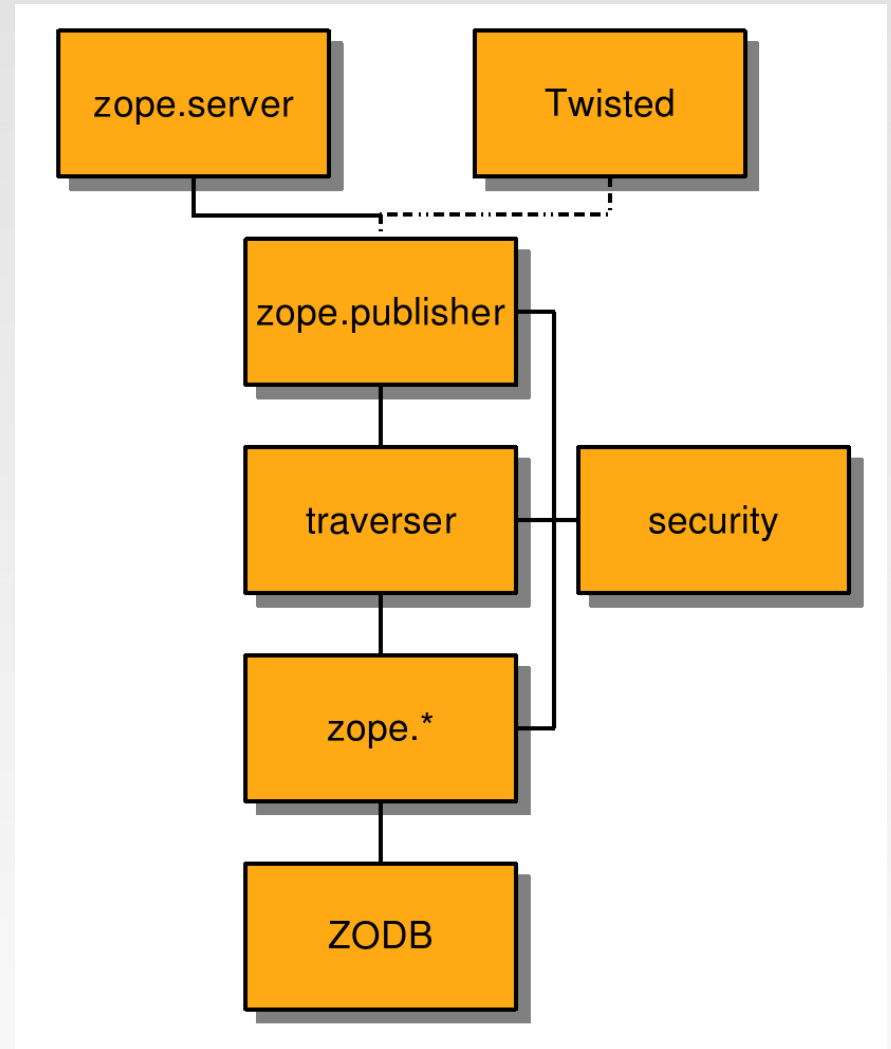
(So, not so unpythonic as people think)

Zope 3: Appearances deceive

Zope 3?



Zope 3!



Two attitudes

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

Zope3: Component Architecture

Everything is extensible and pluggable!

Surprise!

zope.component is not Zope only!

Surprise again!

zope.component requires no XML!

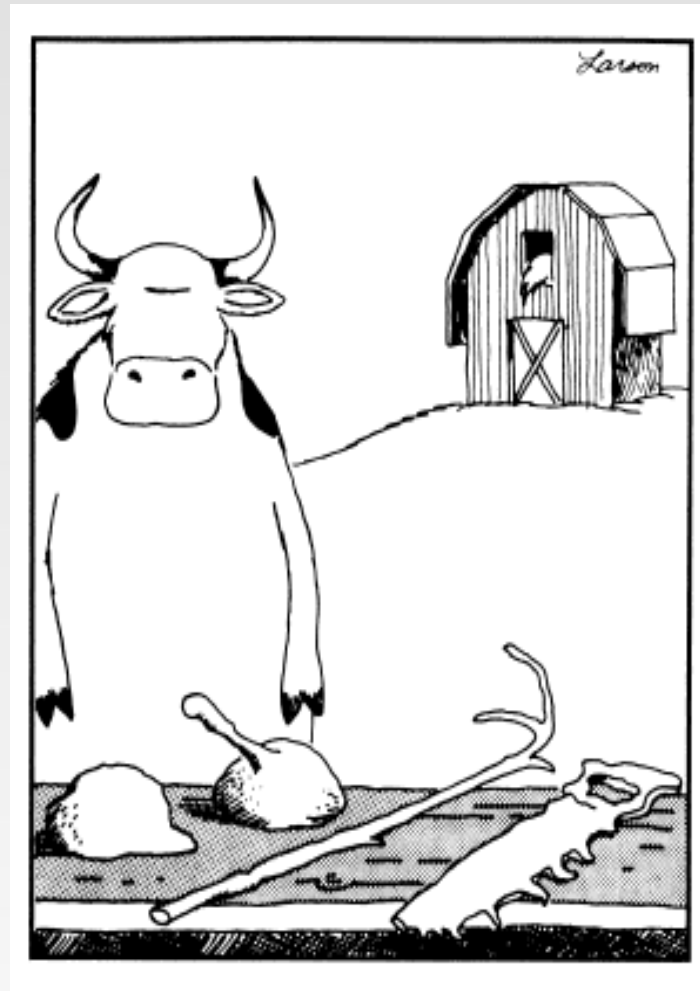
Zope3: Adapters everywhere!

A view of an object is really an adapter between the object and the request to the `IBrowserView` interface.

Zope3: Death by abstraction

- Add a field to a schema
- Schemas are interfaces
- You need to write Python-code!

Tools need to make sense



The blank sheet

- 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

The developer dilemma

Low entry threshold
and
all the power of a big
framework
without
dead ends?

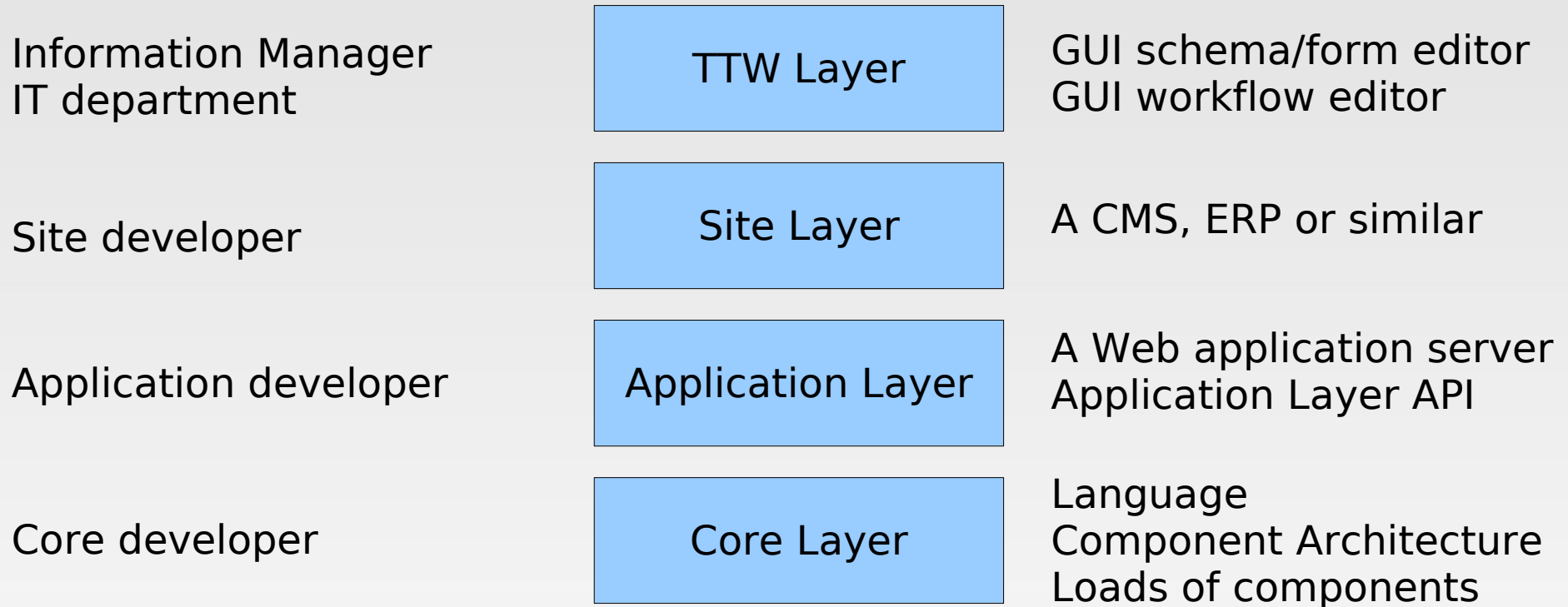
Zope3: The learning curve



The layered cake!



The layered cake



Too much freedom!

How can the unexperienced choose
between Genshi, Zpt and Kid

They don't

They will start by using a finished application, like a CMS

How can we accomplish this?

The basic things exists:

Python

Zope Component Architecture

Zope 3 the libraries (well, soon)

Grok

Replace this:

```
<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="*"
  />

</configure>
```

With this:

```
import grok
```

```
class HelloWorld(grok.Application, grok.Model):  
    pass
```

```
class Index(grok.View):  
    pass
```

The hierarchy

Information Manager
IT department

GUI Tools

Site developer

The CMS/ERP

Application developer

Grok

Core developer

Component
Architecture