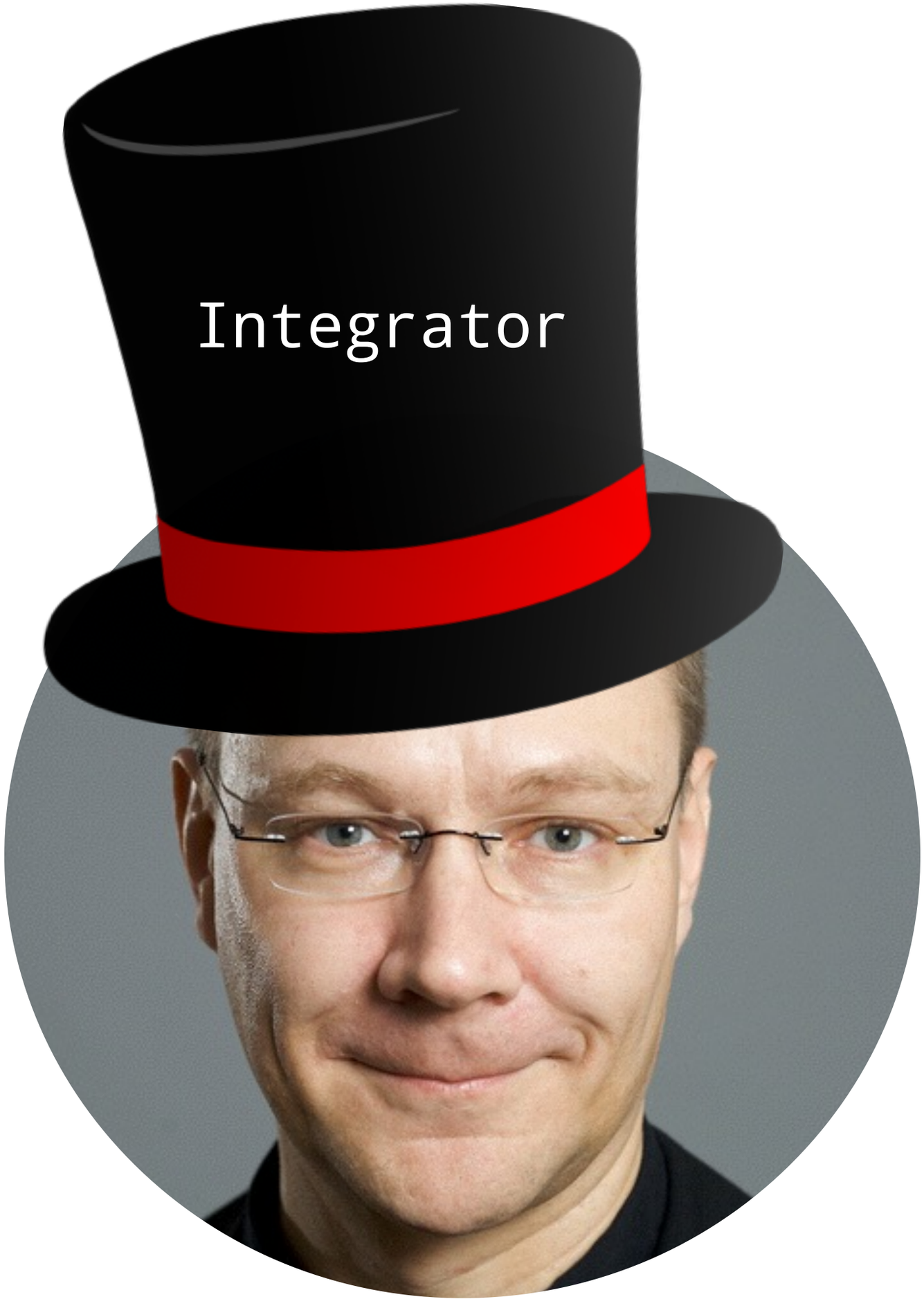
Why Plone Will Die

(or turn into a CMS zombie)

Andreas Jung
Plone Conference 2014, Bristol

Disclaimer

- perhaps tendentious
- perhaps provoking
- perhaps unbalanced
- always my personal opinion
- not representing any official Plone position


Integrator


Consultant

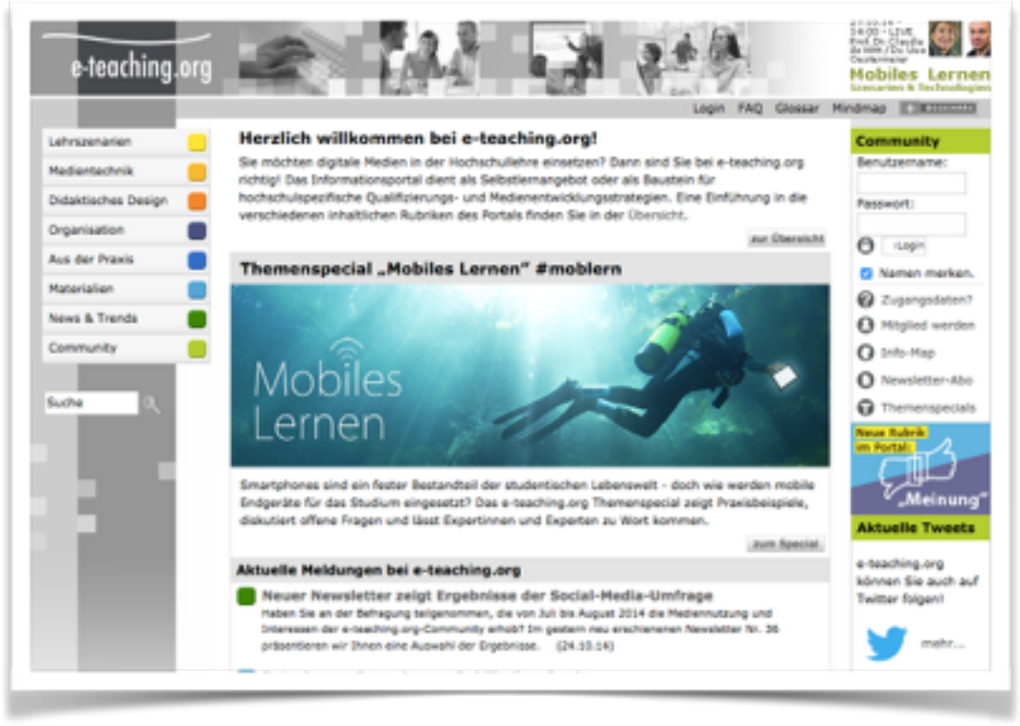

Plone coredev

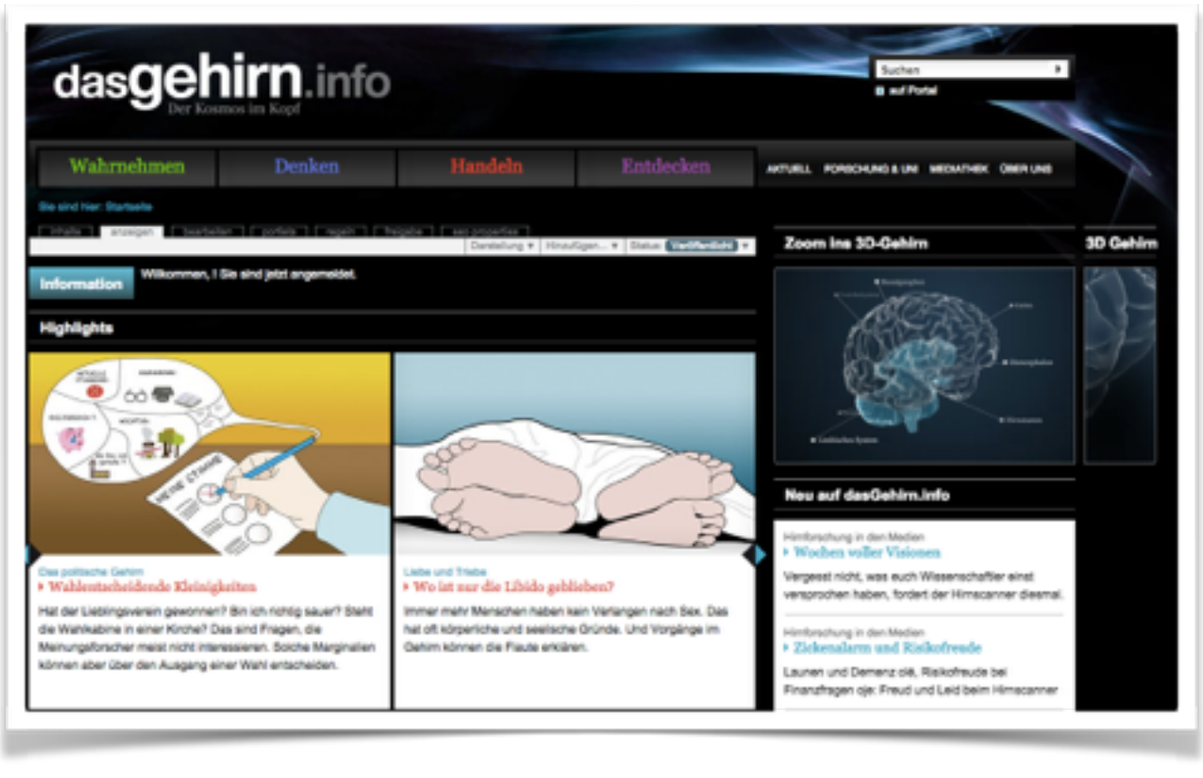

Loudmouth

Why am I giving this talk?

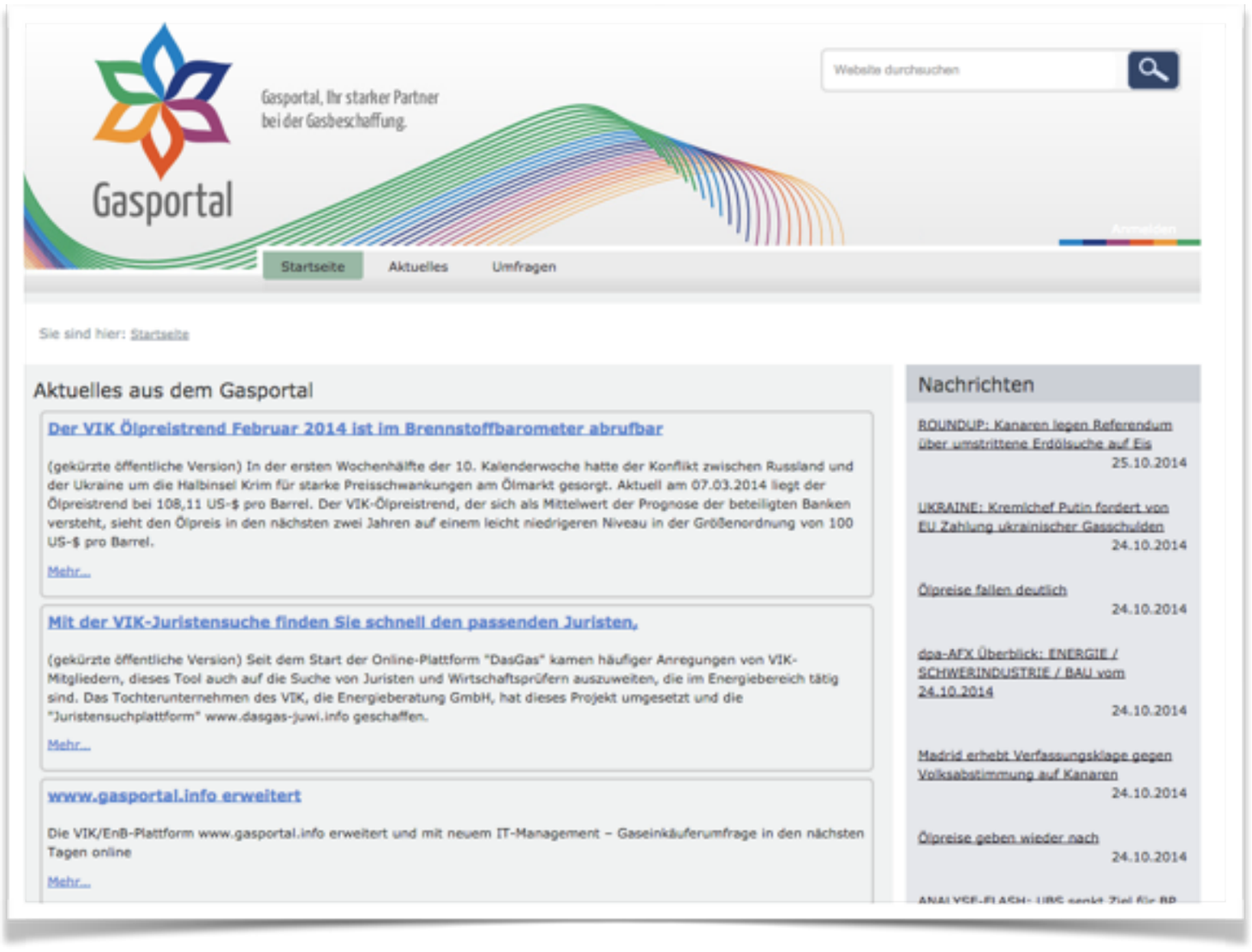
- growing developer pain
- growing integrator pain
- rising and unpredictable project costs
- developer frustration at every corner

2014 - the year of Plone legacy



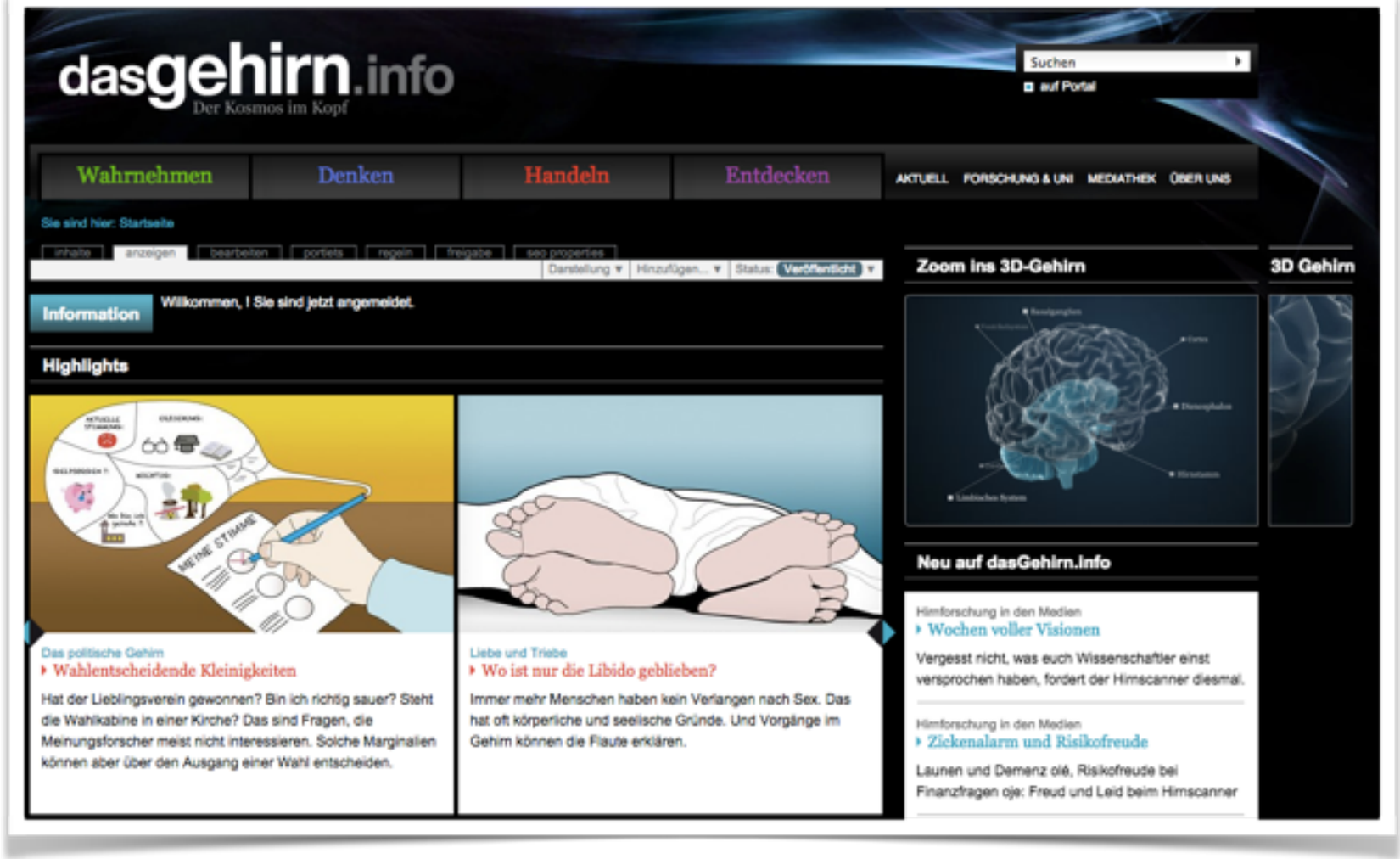


2014 - the year of Plone legacy



- Plone 4.2→4.3
- 3rd party code
- TinyMCE issues
- schema tabs no longer working
- Javascript issues
- JQueryUI incompatibility

2014 - the year of Plone legacy



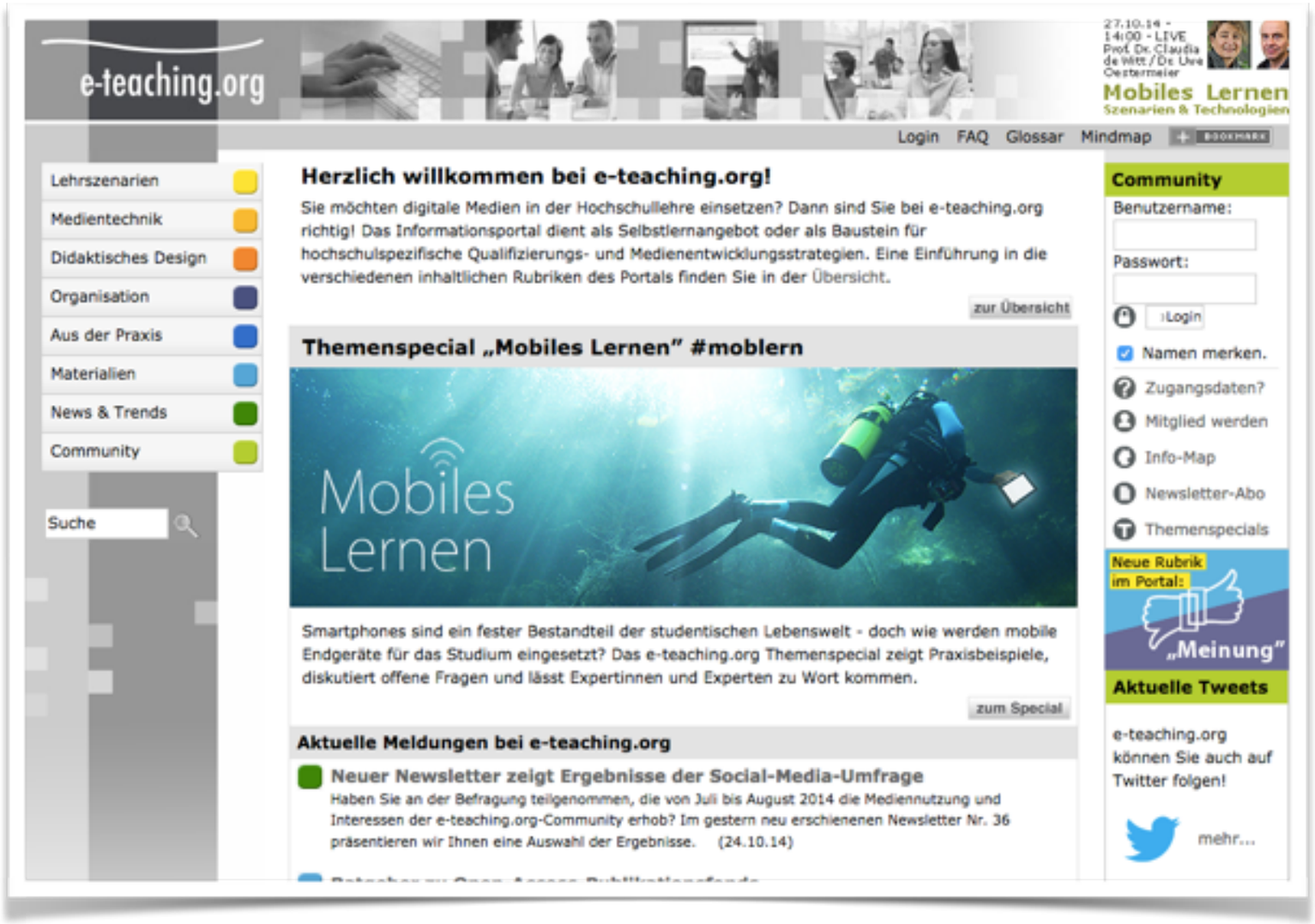
- Plone 4.0→4.2
- 3rd party garbage code
- Migration to Plone 4.3 not easily doable
- Migration costs not predictable

2014 - the year of Plone legacy

- Plone 4.0→4.2
- own project
- Migration to Plone 4.3 not easily doable
- Migration costs not predictable
- Problems with fully qualified links in images and links

2014 - the year of Plone legacy

- Plone 2.X→4.3
- customer grown project
- AT→Dexterity
- p.a.event pain
- p.a.widgets pain
- z3c.form fun
- much more pain…



2014 - the year of Plone legacy

- Plone 4.0→4.1→4.2→4.3
- own project
- started in 2010
- every migration upgrade had more or less severe issues and caused significant work and costs

I am not talking about UX with Plone

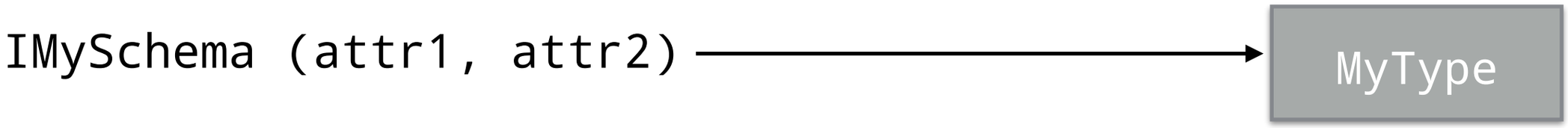
Let me talk of _my_

(backend) developer experience

```
>>> reasons[„programming“]
```

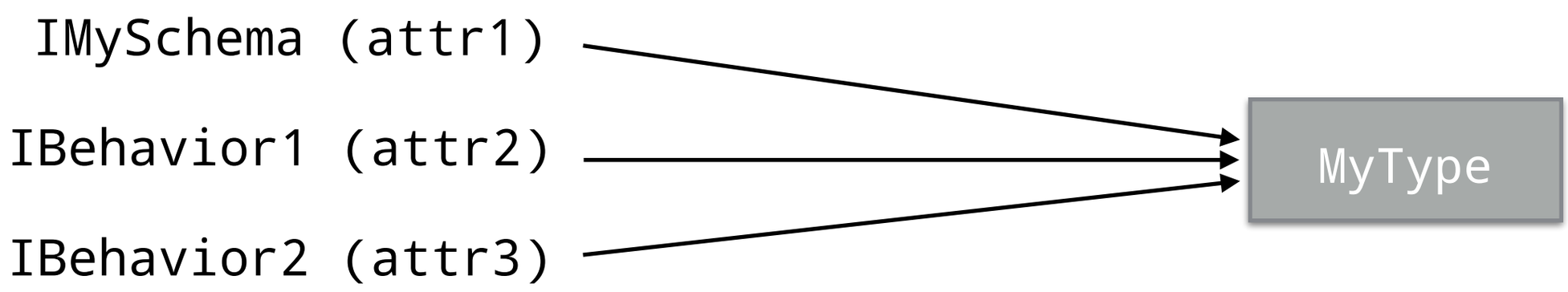
- Dexterity programming experience

Dexterity wanted to be pythonic




```
obj = plone.api.create(„my.type“, container, id)
obj.attr1 = „foo“
obj.attr2 = „bar“
```

…but it ended like this




```
obj = plone.api.create(„my.type“, container, id)
obj.attr1 = „foo“

IBehavior1(obj).attr2 = „bar“
IBehavior2(obj).attr3 = „hello world“

call_some_subscriber(obj) # plone.app.event
```

Q: Is this pythonic?

```
>>> reasons[„programming“]
```

- Insufficient parameter checks
- Insufficient error handling
- Stupid error messages

My personal enemy: z3c.form

- no compatibility checks between schema fields and widgets, pdb needed
- pointless, unhelpful error messages saying all and nothing
- vocabulary issues always require to use pdb (NOVALUESET error with lots of fields and vocabularies)

…and then this madness

```
'/Users/ajung/.buildout/eggs/z3c.form-3.1.1-py2.7.egg',
'/Users/ajung/.buildout/eggs/plone.z3cform-0.8.0-py2.7.egg',
'/Users/ajung/.buildout/eggs/plone.autoform-1.6-py2.7.egg',
'/Users/ajung/.buildout/eggs/plone.app.z3cform-0.7.6-py2.7.egg',
'/Users/ajung/.buildout/eggs/plone.formwidget.namedfile-1.0.9-py2.7.egg',
'/Users/ajung/.buildout/eggs/collective.z3cform.datetimewidget-1.2.6-py2.7.egg',
'/Users/ajung/.buildout/eggs/plone.app.form-2.2.4-py2.7.egg',
'/Users/ajung/.buildout/eggs/zope.formlib-4.0.6-py2.7.egg',
'/Users/ajung/.buildout/eggs/five.formlib-1.0.4-py2.7.egg',
'/Users/ajung/.buildout/eggs/z3c.formwidget.query-0.10-py2.7.egg',
```

```
>>> reasons[„programming“]
```

- complexity
- growing complexity
- wrapper modules…

250 modules (4.3 install)

Up to 400 modules (custom install)

Q: Who understands this? (completely)?

http://www.wired.com/wp-content/uploads/images_blogs/wiredscience/2011/08/iceberg-towing-illustration-dassault-systemes.jpg

```
>>> reasons[„programming“]
```

- Zope Component Architecture
  - A framework is not an API

Zope Component Architecture ZCA

- ZCA is the foundation of the Plone core (not of Zope2 itself)
- ZCA has good parts and bad parts
- brain***k features like multi-adapters neither lead to better code nor better readability
- ZCA is a reasonable implementation framework
- ZCA based code can be complex and complicated
- ZCA often misused as public API in lack of an API at all
- _I_ do not want to see much ZCA stuff on the public API level (except schemas, interfaces)

```
>>> reasons[„programming“]
```

- Lack of explicit and documented APIs

Documented APIs are essential

- APIs define an entry-point to a particular functionality
- APIs encapsulate and hide complexity
- APIs define a well-defined behavior
- APIs can validate and enforce constraints on incoming and outgoing data
- plone.api
  - „Facade“ design pattern
  - covers only the most elementary functionality of Plone
- APIs must be humane

(Kenneth Reitz: https://speakerdeck.com/kennethreitz/python-for-humans)

```
>>> reasons[„programming“]
```

- Migration pain

Migration pain

- Plone portal_migration usually works but
  - often TinyMCE issues
  - often Javascript issues
  - incompatible add-ons
  - (subtle) behavioral changes
- Not a single smooth Plone 4.3 migration this year
- Migration costs
  - often no longer predictable
  - higher risks & higher costs with every Plone major release

```
>>> reasons[„programming“]
```

- Legacy all around us

The pain of dealing with legacy

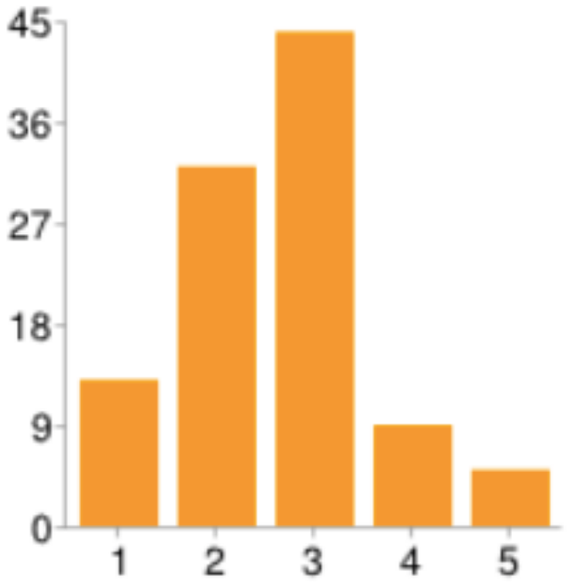
- e.g.Zope 2, CMF
- new rewritten functionality added without removing all old culprit
- more radical cuts are needed in order to get rid of old garbage before adding hopefully better code

>>> reasons[„others“]

- Stagnation!?
- Shrinking community!?
- Shrinking adaptation!?

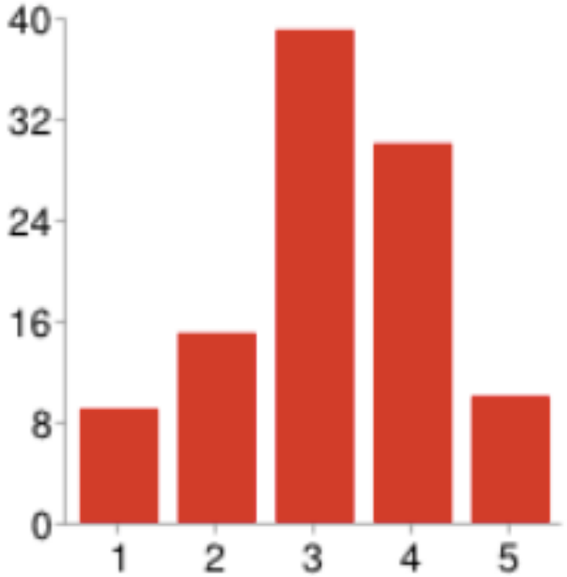
Stagnation

What is your observation about the Plone community growth?



1	**13**	13 %
2	**32**	31 %
3	**44**	43 %
4	**9**	9 %
5	**5**	5 %

How do your customers consider Plone for further projects

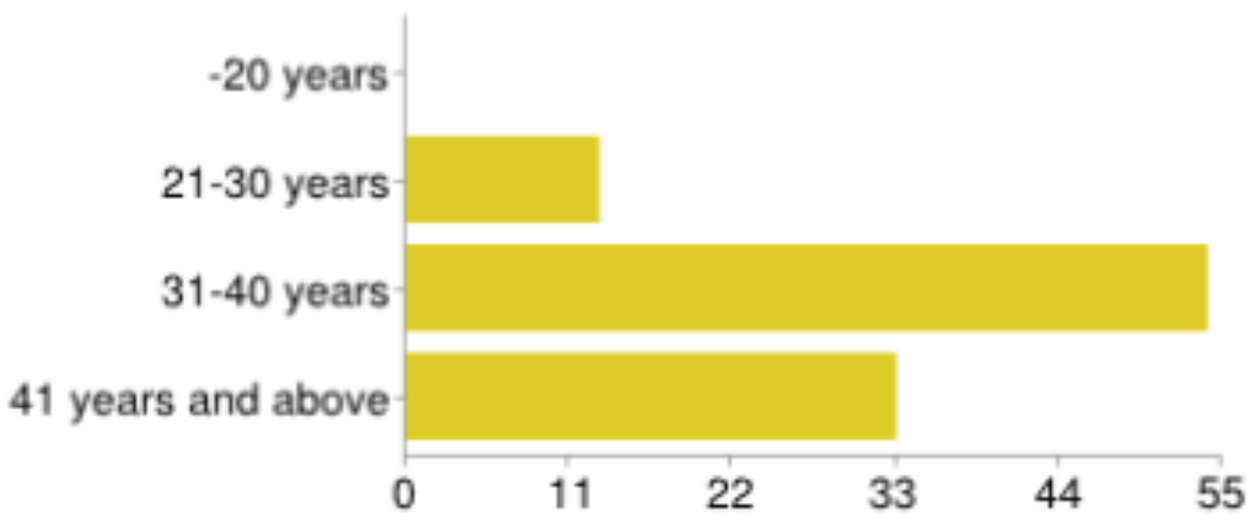


1	**9**	9 %
2	**15**	15 %
3	**39**	38 %
4	**30**	29 %
5	**10**	10 %

Source: Plone Developer Survey 2014

Stagnation

Your age?



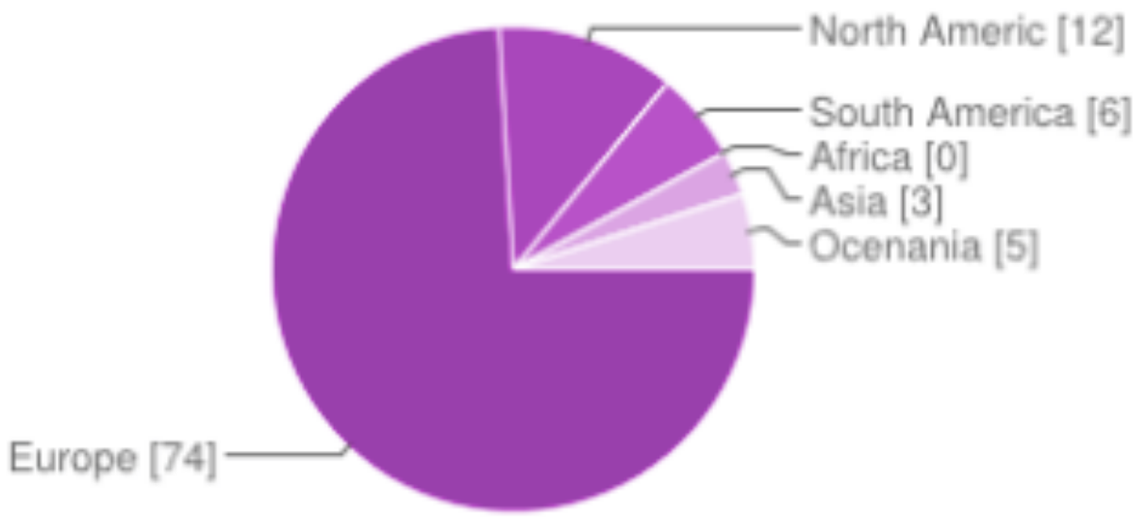
-20 years	**0**	0 %
21-30 years	**13**	13 %
31-40 years	**54**	52 %
41 years and above	**33**	32 %

Gender

male	**95**	92 %
female	**4**	4 %
other	**1**	1 %

Source: Plone Developer Survey 2014

Geographical stagnation?



Europe	**74**	72 %
North America	**12**	12 %
South America	**6**	6 %
Africa	**0**	0 %
Asia	**3**	3 %
Ocenania	**5**	5 %

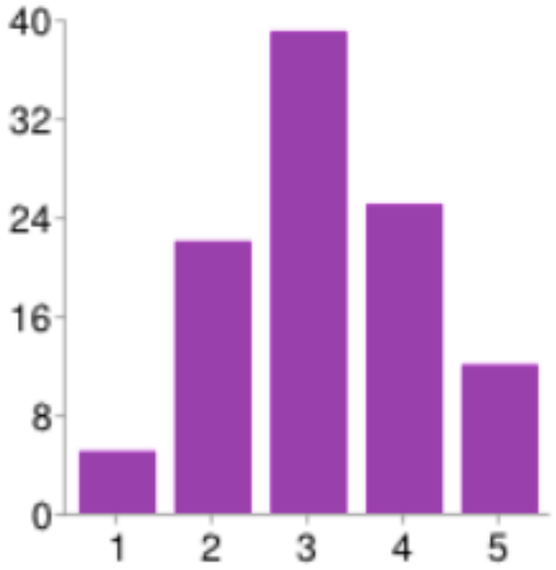
Source: Plone Developer Survey 2014

>>> reasons[„others“]

- Shrinking CMS market?
- More legacy Plone projects than new Plone projects?

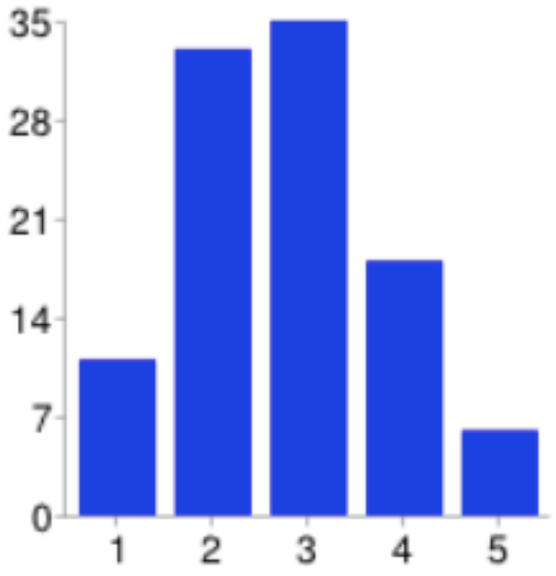
Shrinking CMS market?

How do you consider the CMS market in the future?



1	**5**	5 %
2	**22**	21 %
3	**39**	38 %
4	**25**	24 %
5	**12**	12 %

How do you consider the role of Plone in the CMS market in the future?



1	**11**	11 %
2	**33**	32 %
3	**35**	34 %
4	**18**	17 %
5	**6**	6 %

Source: Plone Developer Survey 2014

#WhatMakesPloneAnOptionFor_ME_Today

- Enterprise CMS
- fine-grained security model
- outstanding security record over the last 12 years
- flexible workflows
- ZODB was great but time to move on…

#WhatMakesPloneAnOptionFor_ME_Tomorrow

- As a developer
  - getting rid of old code cruft
  - getting rid of over-designed and over-engineered code cruft (e.g. portlets, z3c.form)
- explicit and consistent APIs everywhere
- explicit type checking
- much more explicit error messages

#WhatMakesPloneAnOptionFor_ME_Tomorrow

- As a consultant/integrator for „enterprise“ projects:
  - better search-engine (e.g. ElasticSearch)
  - better scalable database backend
  - Python 3.x
  - a coherent and consistent architecture

How can this be achieved?

- Define what Plone currently is
- Define what Plone should be in the future
  - Define the scope of Plone
  - Know your market segment
- Be realistic about the goals that can be achieved with the given resources

How can this be achieved?

- Zope 2 development is dead
  - → Forget Zope 2
- CMF development is more than dead
  - → Forget CMF
- ZCA…well…
  - Take the good parts and forget the rest
- ZODB
  - It was nice with you but we need and we do have better options…no more pickle graves please.

How can this be achieved?

- Throw everything away and start from scratch
  - Pyramid
  - Python 3
  - a new persistence API
  - a new pluggable persistence layer
  - evaluate the database market for the best database option with respect to scalability, ease-of-use etc.
  - see how existing functionality like Dexterity fits into a new architecture

The time of the monster application servers is over! Take the best components on the market and build it yourself.

And there’s one more thing…

- Don’t call it Plone!
- Develop it under the umbrella of the brand Plone
- But give it a different name (similar case with Typo 3 and its reincarnation NEOS)

Plone Developer Survey (103 answers)

- 103 answers
- most interesting: a lot of interesting comments about the pros and cons of Plone

- http://goo.gl/pBK3DM

Questions?


Source: Monty Python
Thank you for your patience 😊