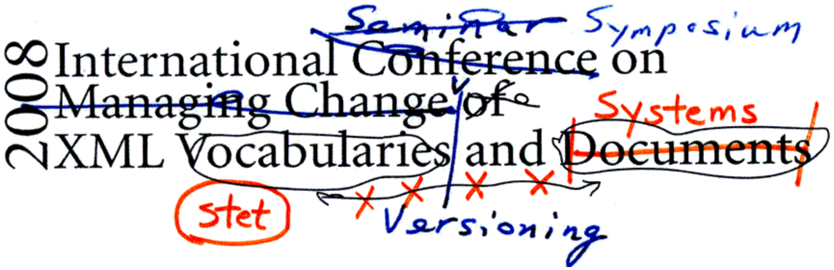
This presentation has no version

Versioning as a social construct

Presentation to the “Versioning Symposium”

Montréal, 11 August 2008


~~Seminar~~ Symposium
2008 International ~~Conference~~ on
~~Managing Change of~~
XML Vocabularies and ~~Documents~~
Systems
stet
Versioning

Peter F Brown

Pensive S.A.

Overview

- A bit of philosophy: Plato, Socrates and Aristotle
- Wittgenstein to the rescue
- Functional approaches
- A new (and not so new) model
- Some conclusions

Plato’s Phaedrus Dialogue

- Socrates talks about reality having “natural joints”:
  - “according to the natural formation...not breaking any part as a bad carver might”
  - Implies that there are clear carving points between ***x*** and ***not-x***
- But, dear Socrates, what is a “good carver”?

Aristotle

- "Everything is defined by clear and knowable traits"
  - Aristotle's principle of *essentialism* – the set of characteristics that define something as belonging to a category
  - Assumes that there are "natural joints"
- However...
  - "essentialism makes the world seem more manageable, but it can lead us to miss what's really going on" (David Weinberger)
    - We think we've found the perfect "carving points", but we realise that in certain contexts, it "doesn't work"
    - "accidentalism" – "versions happen"

What’s the point of a version?

- “What’s the point”,
  - as in: “why bother”,
  - as well as: “where *is* the specific point”
- If we accept, at some point in the future, that: $\boldsymbol{x}$ becomes $\boldsymbol{x}_{\textit{(next version)}}$
  - what is particular, special, about that point?
  - What decisions are implied or explicitly stated?
- More to the point:
  - Do we actually have a clear model of what it means to be a “version”?

What *is* a version?

- A Wittgensteinian model:
  - Would require there to be some “prototype” concept of “version” that carries the characteristics of what it is to “be” a version
  - If so, what are the prototypical characteristics? In other words
    - what are the characteristics of a prototypical *version* that any version should contain?
    - Which characteristics constitute an essential characteristic of the prototype are culturally determined.
  - But such characteristics are also functionally (culturally? Socially?) determined

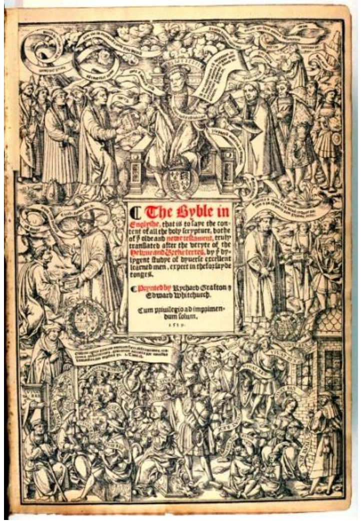
Functional differences

- We use the same concept “version” to play too many roles concomitantly:
  - A stage in the evolution of an artefact (a process)
  - A state (“incomplete”, “finished”)
  - A status (“draft”, “approved”)

Let’s look at a few...

“Authentic” versions

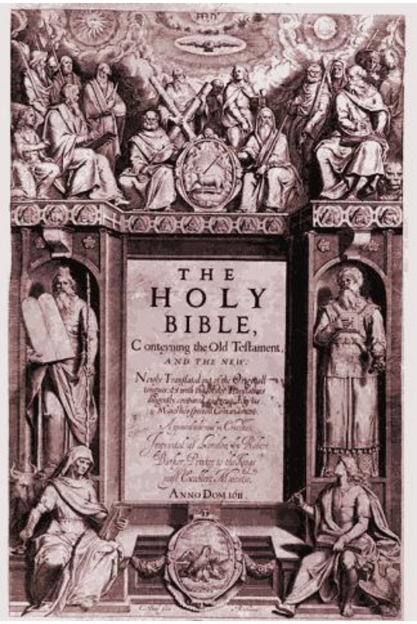
- The Bible
  - King James Authorised Version
  - ...and the Great Bible?
  - ...and the Bishops Bible?



v1.0?

v2.0?

They are considered “authentic” because an authority (a king) deemed them so. The Bishops Bible was never officially “authenticated” but nonetheless is the basis of “sanctioned” texts such as Psalms and the Book of Common Prayer



v3.0?

If they are “versions”, versions of what, exactly? What is the referent?

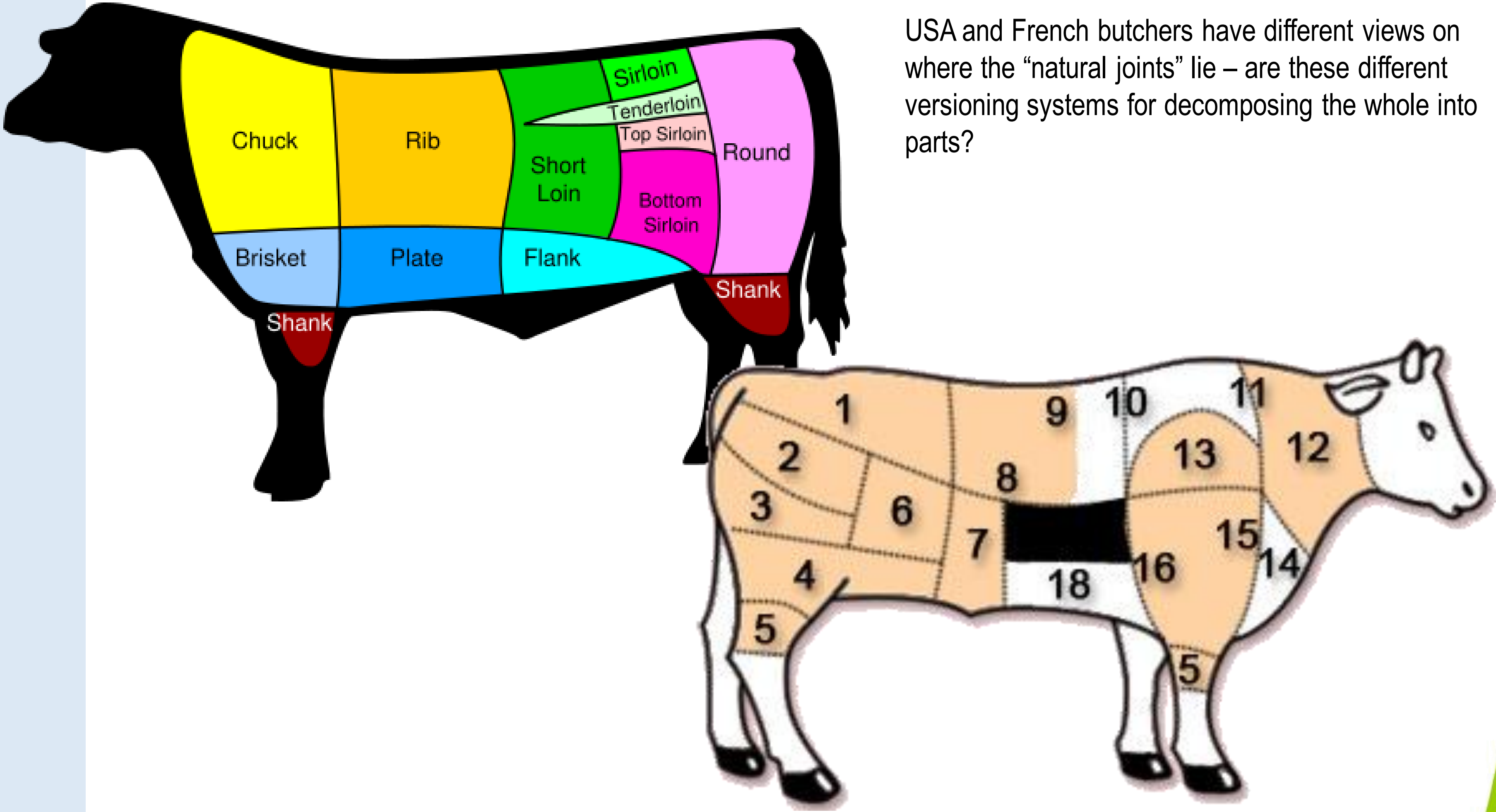
“Point of View” as a version

- 2008 film “Vantage Point”
  - 8 versions of a story.
  - “versions” of the same events?
  - “8 Points of View. 1 Truth”

Where's the Beef?

- What about versions of different truths?

USA and French butchers have different views on where the “natural joints” lie – are these different versioning systems for decomposing the whole into parts?



What are versions for?

- Identifying content?
- Tracking evolution?
- Tracking differences? What sort of differences?
  - "Significant progress"?
  - Capturing and notifying cock-ups and errors?
- and canonical forms?
  - Are they versions?
  - If so, what of? Or are they the elusive "referent"?
  - If not, then what are the other non-canonical forms?

Version numbers

- All numbering systems have an implied hierarchy
  - "v2" implies "v1", and suggests that a "v3" could be coming along later...
  - Would be odd to suddenly have a "v2.1" slipped in...
  - Nonsense of the "Anything 2.0" trend
    - "SOA 2.0": SOA is a reference model not a software build.
    - "Web 2.0" is what exactly? Can someone define Web 1.0? And was there a Web 1.1 somewhere along the line? Did I miss it? Says who?
    - and when our shiny, new, perfectly formed "v1.0.0.0" turns out not to be fit for major release after all?
- The main problem is that we implicitly attribute too much semantics to the numbers

When is a thing *not* a version?

- "...you can't make any progress by being afraid of your own shadow during design." (Greg Hudson)
  - But a "version" ***is*** often a shadow – expectation that your next "version" must be "better", "newer", than the current one
  - Does the "version" *n+1* always have to appear as following version *n*?
- If there is a v1 and v2,
  - who says there cannot be a v1.1 or v2.3, or v1.9.7.3.6.1....?
  - Are there any "holes" (Pythagoras, Christianity and Milton would say not)
  - Can there ever be a "final version"?
- Is the process just a steady continuum?
  - If so, where are the "natural joints"? And how do you measure a point on the continuum?

Parkinson’s Law 2.0

- “Versions expand to fill the the most recent versioning system available”
- Compare the 1951 Treaty of Paris and the 2003 Treaty of Nice:
  - Did Monnet even think of the idea that his Treaty had versions? The “work” simply evolved until approved
  - Not many CMS systems around then
  - probably a better written text too....
  - The 2003 Treaty had thousands of iterations, versions
  - It didn’t improve the outcome (talk to the Irish – they should know)

Case study

- Hierarchical naming convention to capture several orthogonal properties:

  `1234-v2.3-en.doc`

  - A referent: a single “document” reference;
  - A version
  - A language version
  - A media type version

- The ordering of the values became semantically and computationally significant:

  Implies that

  `1234-v2.3-ro.doc`

  can be considered as being the Romanian translation of:

  `1234-v2.3-en.doc`

  Or that

  `1234-v4.1-fi.doc`

  Implies that there is an equivalent version in any/all other languages

Versions of version

- Release
- Revision
- Build
- Patch
- Status
- State
- Translation
- Media type

- Feature
- PoV
- Variant
- Form

Most common use – as a “snapshot” on the timeline of some evolving artefact

What do we actually need?

- All of these and possibly more!
  - We’ve often been locked in sterile conflict because of misunderstandings over what we mean by “version”
- So we need a different model...
  - That breaks the idea that versions are simply linear/scalar progressions
  - That allows different orthogonal properties to be expressed unambiguously
  - That nonetheless keeps some principles intact
  - That disambiguates our understanding of what a particular artefact actually “is” – this is the crucial point

Every thing *is*; It has no version

- If every thing has an identity, then versions of “it” are also things, and therefore also have identity.
- Every “thing” ***is***; but every thing can also be “something” ***of*** other things
  - it has relationships or ***associations*** with other things
    - one/some of those associations should be “is version of”
  - Version is not an **inherent** (or intrinsic) property...
  - ...but an associative property
    - Don’t talk about “version 5 of fooML”
    - Instead that the work “fooML” is currently represented by “bar.xsd” and that “bar.xsd is considered as version 5 of fooML”

Back to Wittgenstein

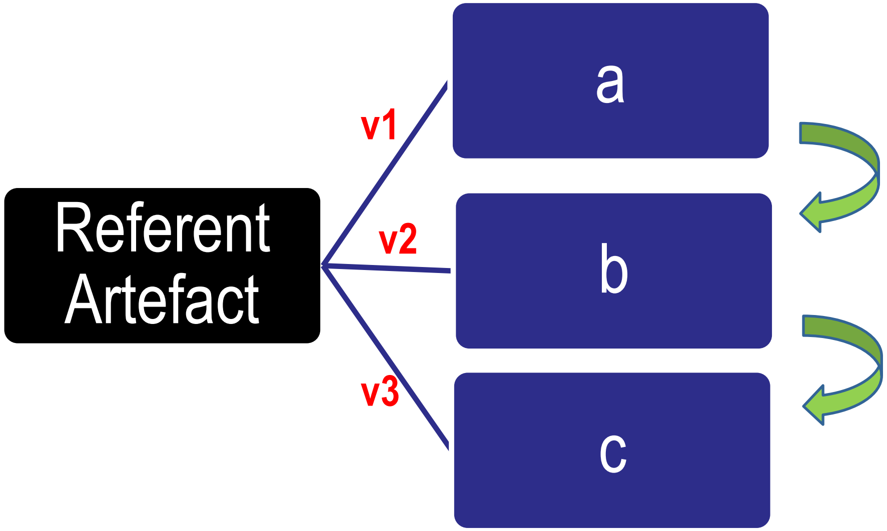
- Are there ***any*** prototypical characteristics of “version”?
- We only attribute “versions” to artefacts
  - So it ***is*** a creative process
  - And we ***attribute***, therefore it is not an inherent property
- We attribute versions according to some agreed or implied process:
  - “approved”; “checked-in”; “published”; “supersedes”....
- But who or what drives that process?
  - A piece of software (CVS, SVN, SharePoint, a CMS)?
  - An authority (a consortium, a standards body)

A version of...?

- “And you are a ‘version’ of....***what*** exactly”?
- Implies a referent:
  - Not so much:

a is v1 → b is v2 → c is v3

  - as:



Referent Artefact

v1

v2

v3

a

b

c

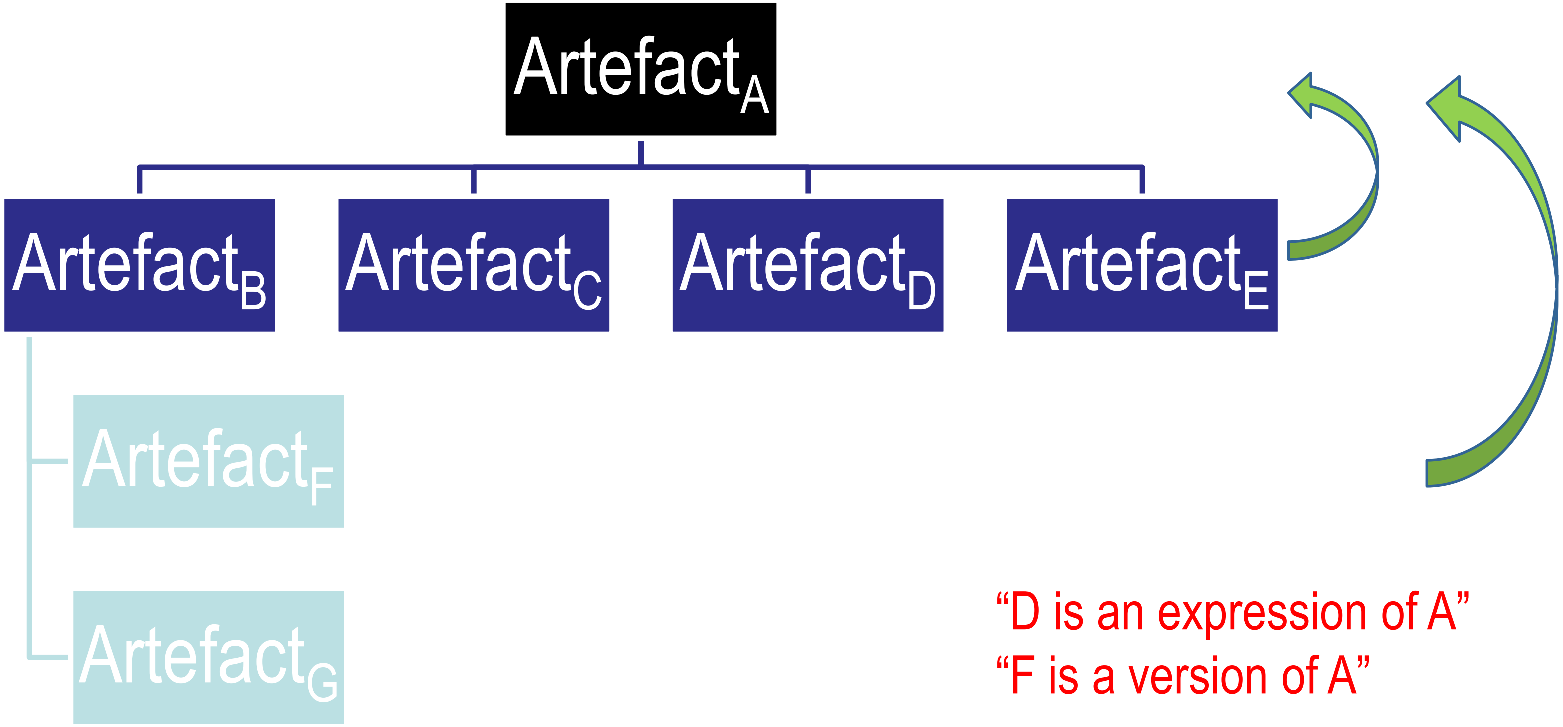
A more complete model (1)

- Inspired by the “Functional Requirements for Bibliographical Records” of IFLA
- 4-layer model for artefacts:

“Work”	“The fooML language”	“Abstract”
“Expression”	“The fooML schema, v1”	
“Manifestation”	“http://www.fooML.org/Schema...”	“real-world”
“Item”	“c:\Data\fooMLv6.xsd”	

- Allows us to address requirements to the most appropriate level:
  - Sometimes we need to relate to the “work”, sometimes as far down the stack as the “item”. Often why so much software is cracked – no explicit agreement as to which level is or needs to be addressed

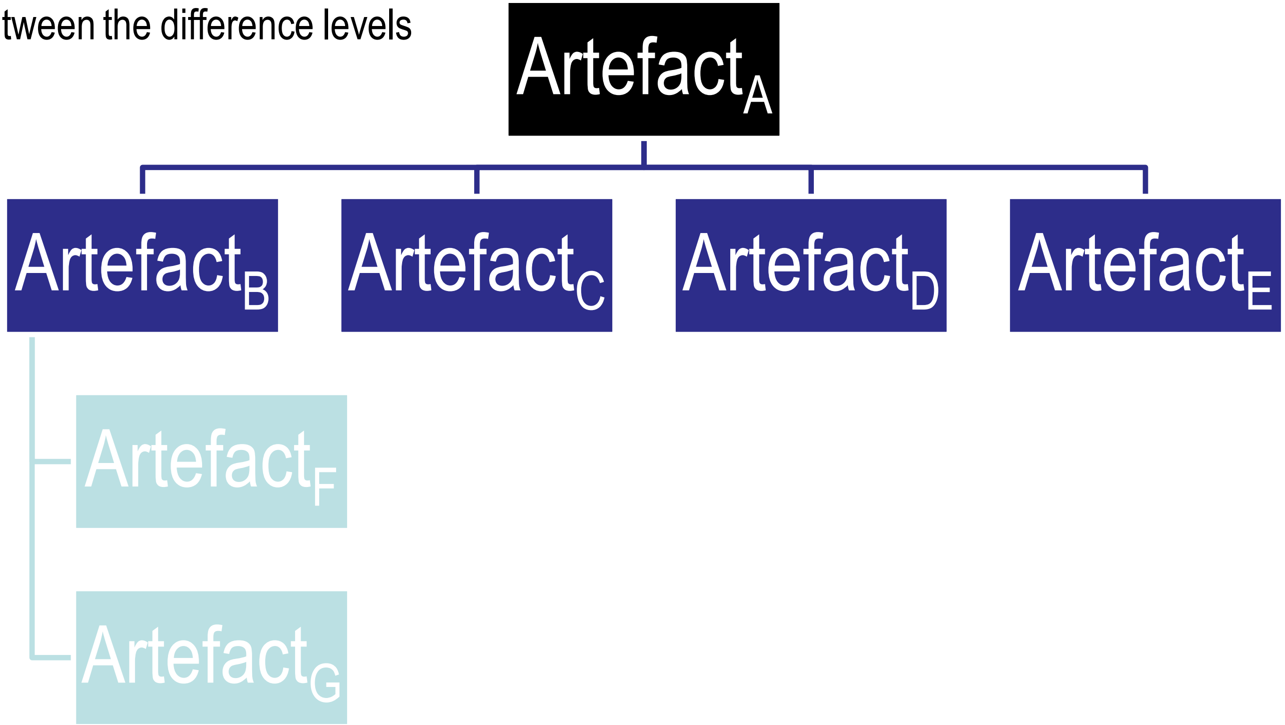
A more complete model (2)



“D is an expression of A”
“F is a version of A”
....

A more complete model (2)

Allows us also to be much more expressive about the relations between the difference levels



Artefact_A

Artefact_B

Artefact_C

Artefact_D

Artefact_E

Artefact_F

Artefact_G

A new vision of version

- No, not “Version 2.0”
  - Rather an understanding that versions are not pure scalar values
  - Even if ***only*** understood in the chrono-evolutionary sense, think of a version as a relationship between two distinct things rather than one “thing” with an inherent property, the value of which changes according to its evolution.
  - If we learnt anything about XML, it is that no label is truly “self-describing” – least of all the concept of “version”

Bad signs

- The tyranny of the software build:
  - Ever more frequent
- “Granularity creep”:
  - From “alpha, beta, go” to “Beta1, Beta2..., RC1...”, etc
- Towards the “permanent beta”
  - One-time-and-for-always stable releases of software are a thing of the past:
    - Living with a Heraclitusian nightmare of never stepping into the same runtime environment twice.
    - Are we ever going to be able to keep track of which “version” of what works together? And who/what really needs to know?

Better signs

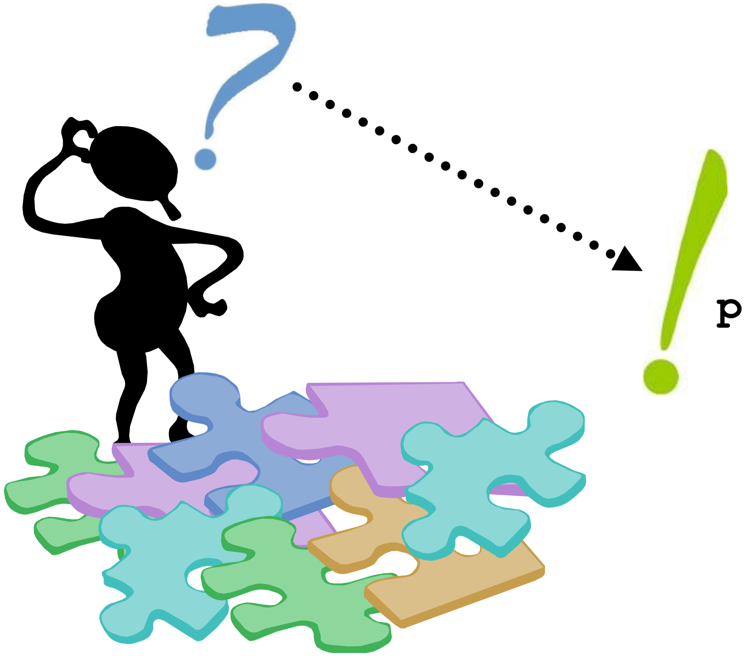
- We’ve reached the “Houston, we have a problem” stage
  - Recognition that there is a problem
  - Attempt to separate out the different functional concerns (backward compatibility, “must understand”, “ignore”, “compatible with this build”, “licensed for this release”, etc.)

Key issues

- When anyone mentions versions or versioning – assume nothing! Be explicit
- Identify the properties that are important, for which stakeholders, and for which purposes
- Accept that these properties are all orthogonal
- Have a clearer understanding of different layers of abstraction of any artefact
- Come to terms with the ensuing nightmare of connections
- Recognize that that is something that computers are good at handling ☺
- Don’t succumb to Parkinson’s Law 2.0

This presentation has no version*

Versioning as a social construct



peter@pensive.eu

*This "work" does however have different "manifestations", which will be available after the conference via www.pensive.eu/uid/0217

My thanks to David Weinberger, and his book "Everything is Miscellaneous", for triggering some of the philosophical reflections in this work