Contemporary source control for Pharo

This is the talk about Git.

ESUG, 2014-08-10

Max Leske

Overview

background

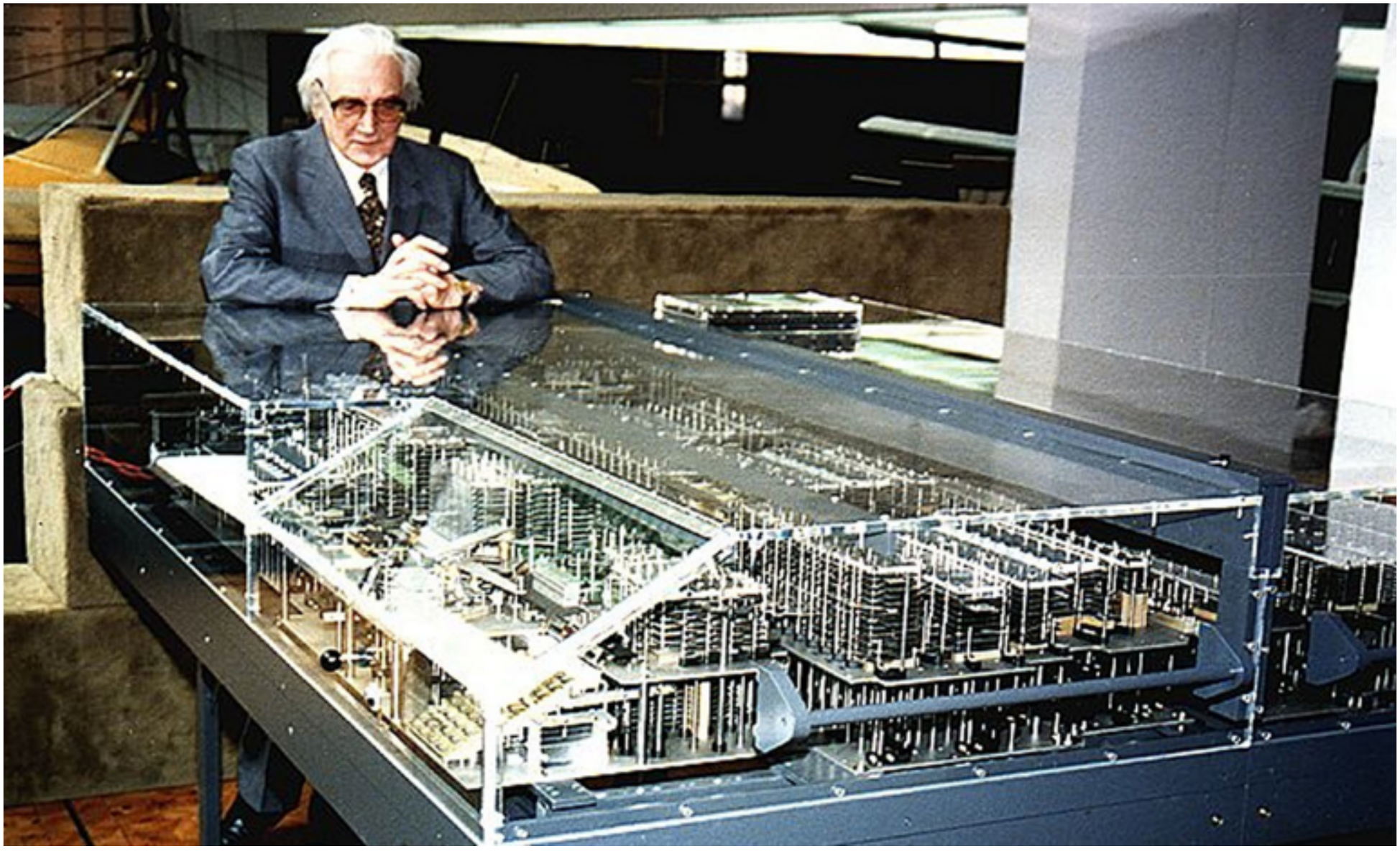
(Git)

current state

(Git)

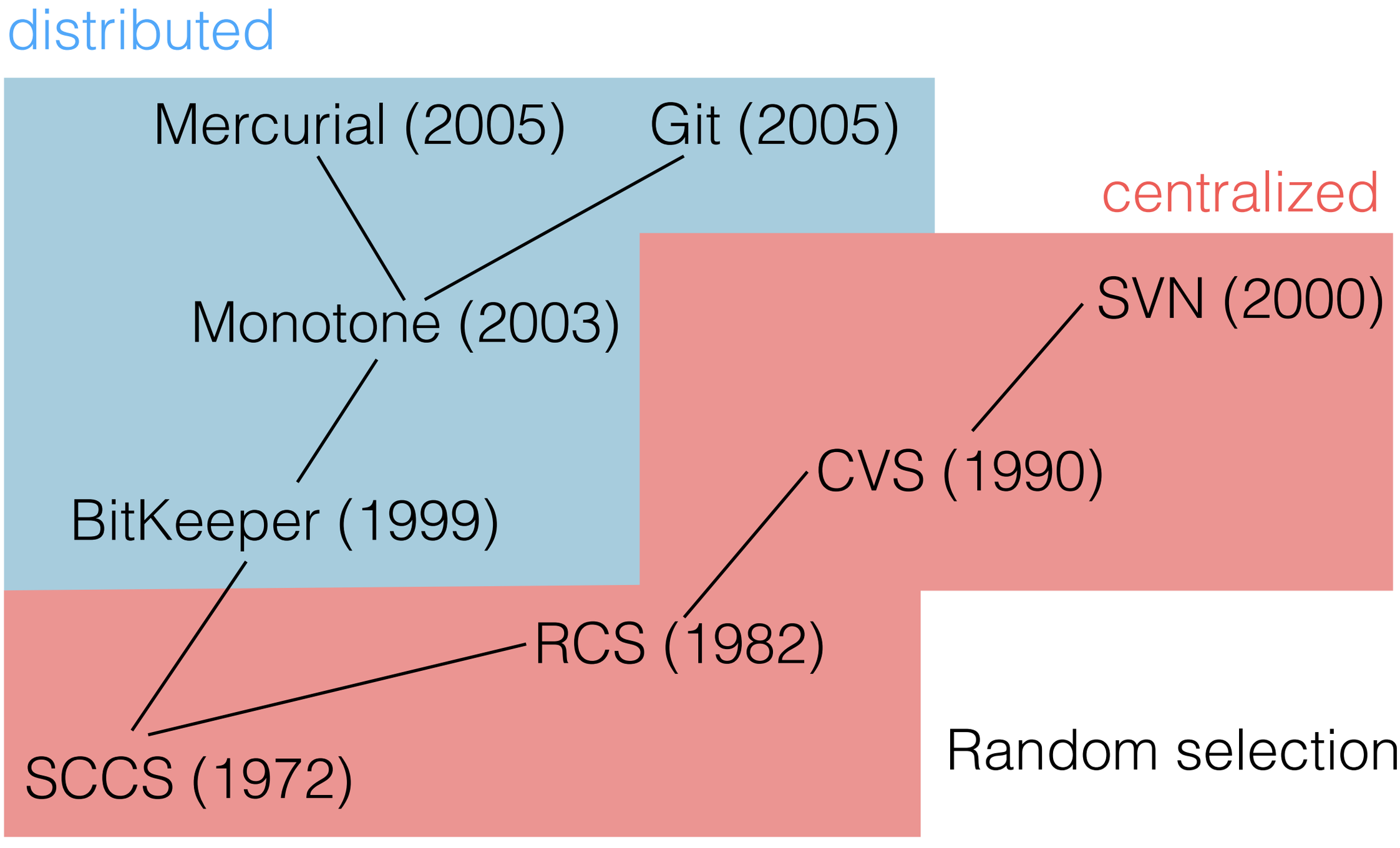
the future

(Git)

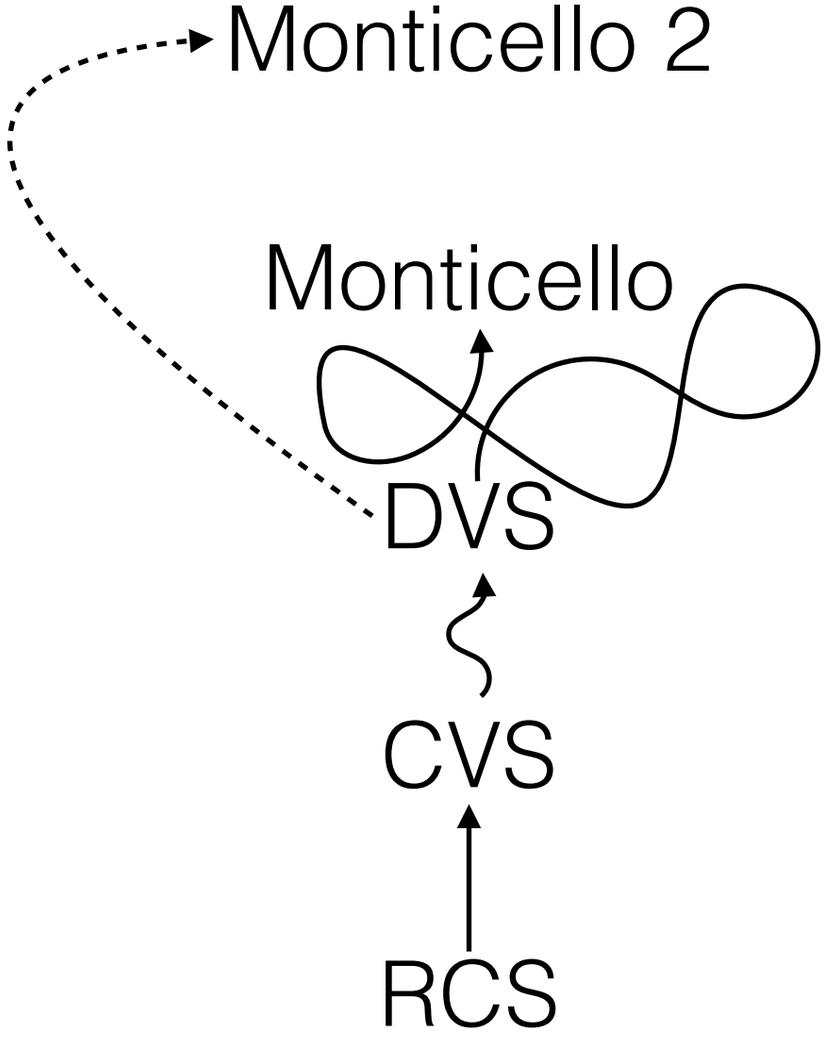


A bit of history

Revision control systems



Source control in Squeak and Pharo

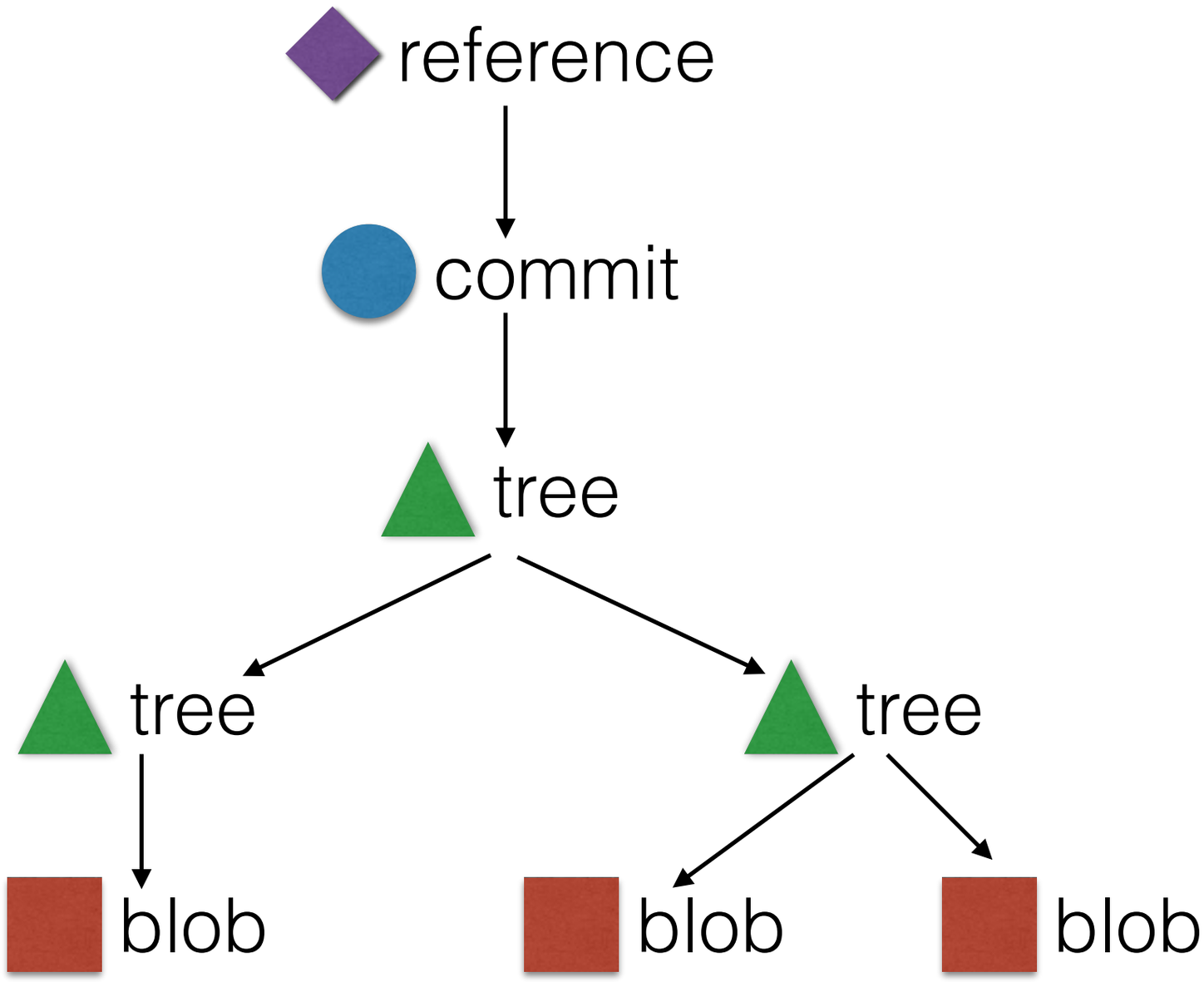


Greatly exaggerated

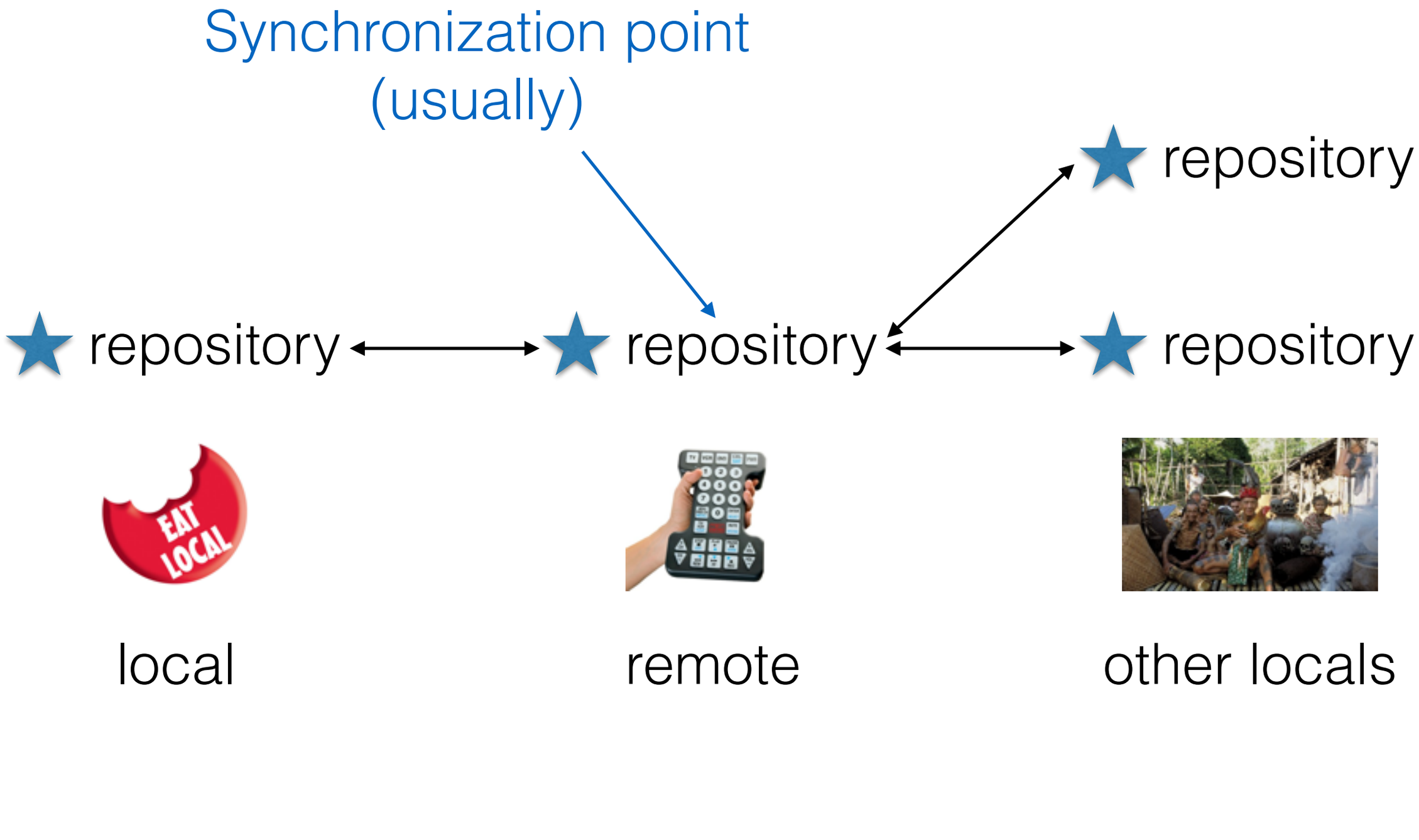
“The stupid content tracker from hell”

A bit about Git

Git objects



Git repositories



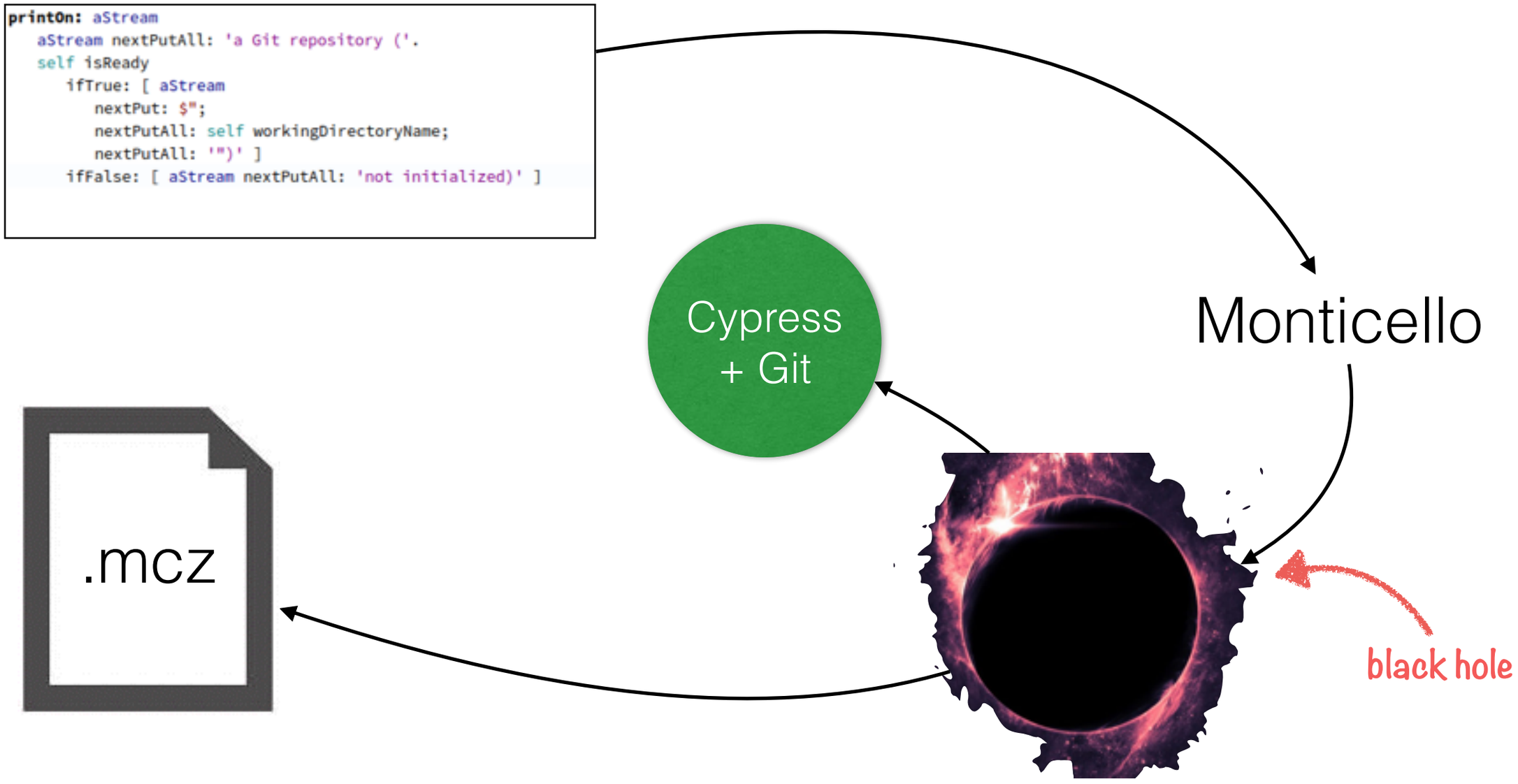
Git protocols

file protocol	file:///path/to/repo.git /path/to/repo.gi
Git protocol	git://server.com/path/to/repo.git
SSH protocol	ssh://user@server/path/to/repo.git user@server.com:path/to/repo.git
HTTP(S) protocol	http://server.com/path/to/repo.git https://server.com/path/to/repo.git

A bit about the now

How Monticello works (in my mind)

```
printOn: aStream
   aStream nextPutAll: 'a Git repository ('.
   self isReady
      ifTrue: [ aStream
         nextPut: $";
         nextPutAll: self workingDirectoryName;
         nextPutAll: '")' ]
      ifFalse: [ aStream nextPutAll: 'not initialized)' ]
```



Monticello problems

- file instead of objects
- chunk file format
- version tracking / ancestry by filename
- hard to understand

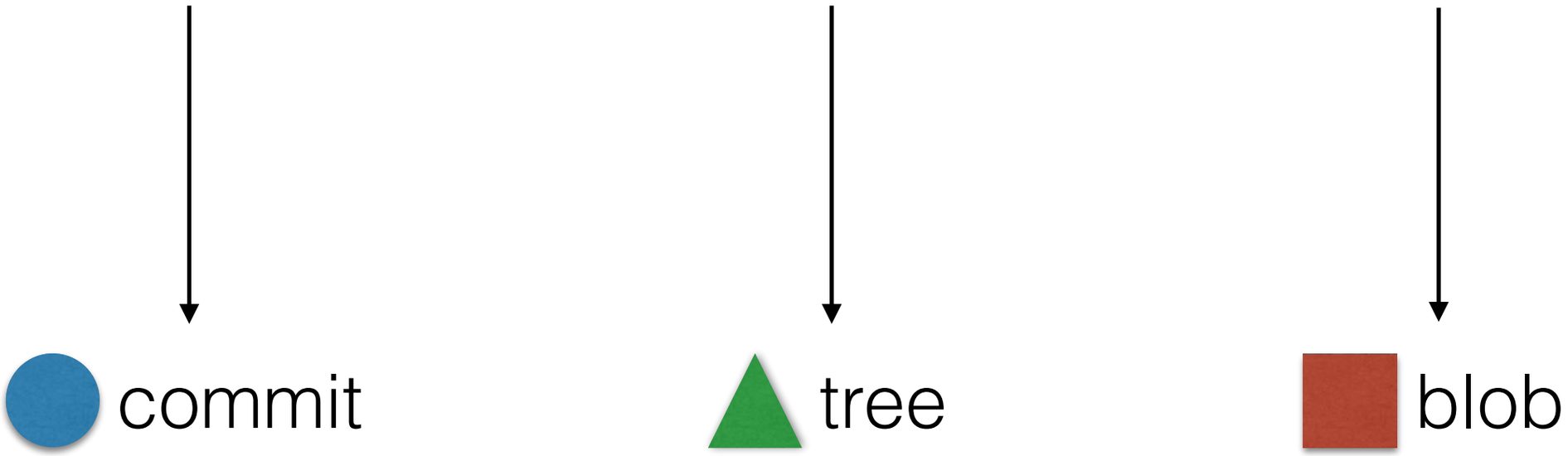
Cypress (FileTree)

meta data

package version

categorization

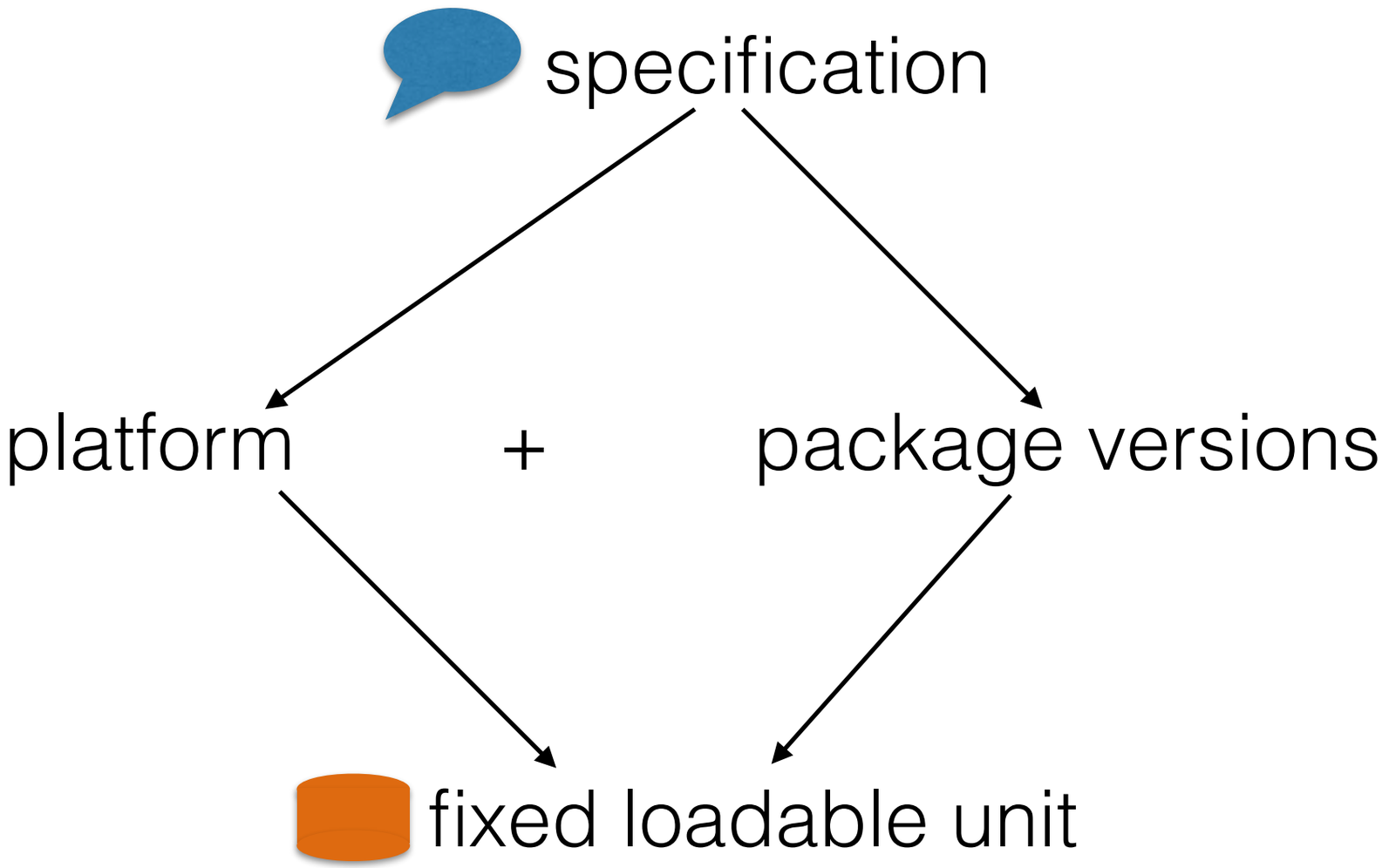
source



Cypress problems

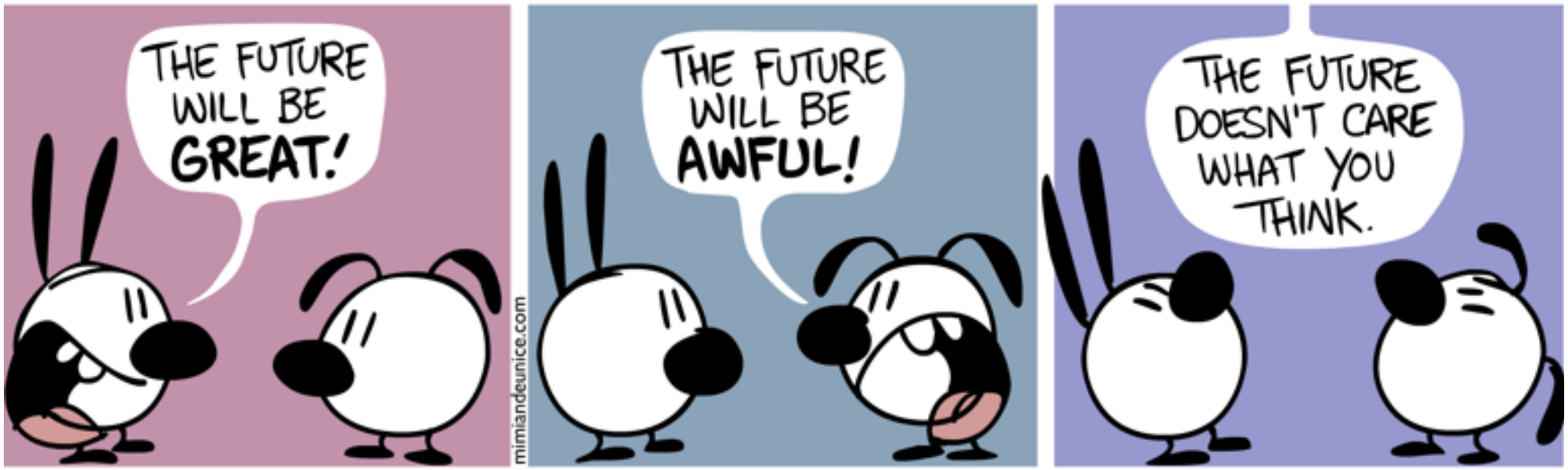
- moves / renames not tracked
- very deep directory structures
- files instead of objects
- checked out version = only visible version
- hard to merge

Metacello


specification
platform
+
package versions
fixed loadable unit

Metacello problems

- complex implementation
- hard to maintain for big projects
- very steep learning curve



A bit on the future

How Monticello might work

```
printOn: aStream
   aStream nextPutAll: 'a Git repository ('.
   self isReady
      ifTrue: [ aStream
         nextPut: $";
         nextPutAll: self workingDirectoryName;
         nextPutAll: '")' ]
      ifFalse: [ aStream nextPutAll: 'not initialized)' ]
```

Monticello

arbitrary backend

Support for arbitrary backends

Matthias Kleine,
Robert Hirschfeld, :
Gilad Bracha

An abstraction for version control systems

https://www.hpi.uni-potsdam.de/hirschfeld/publications/media/KleineHirschfeldBracha_2012_AnAbstractionForVersionControlSystems_HPI54.pdf

Pharo 2 architecture

Metacello

Monticello

Monticello meta model

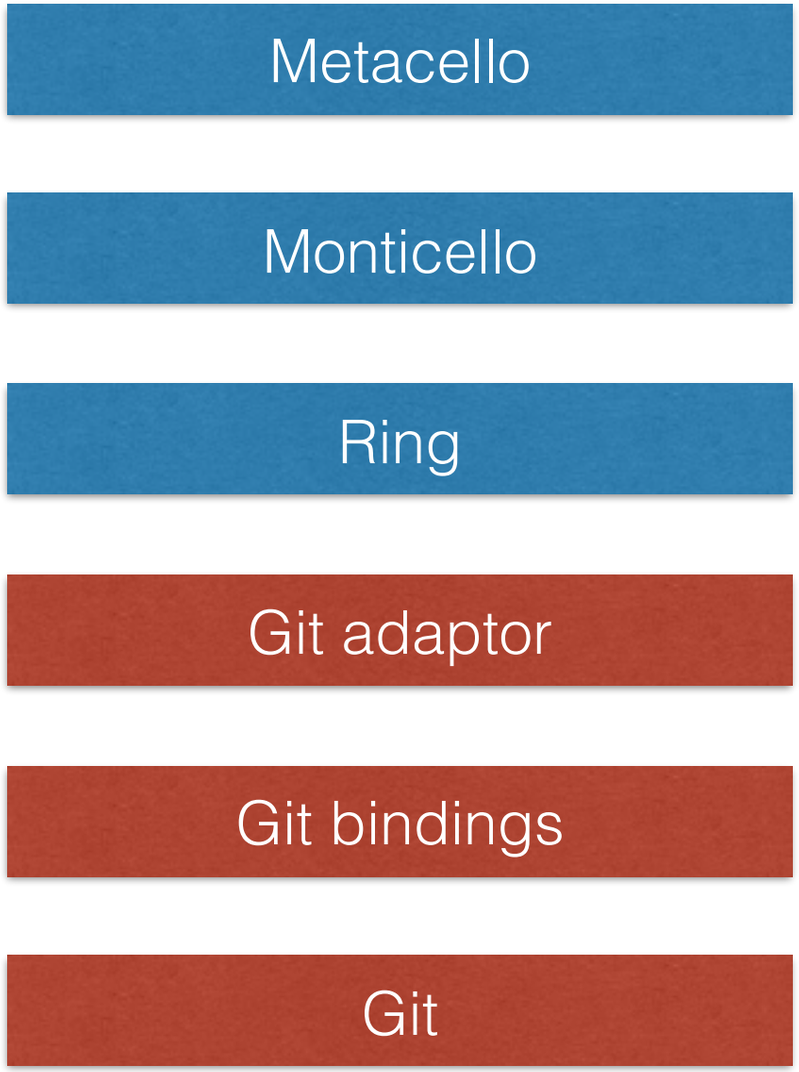
Current architecture

Metacello

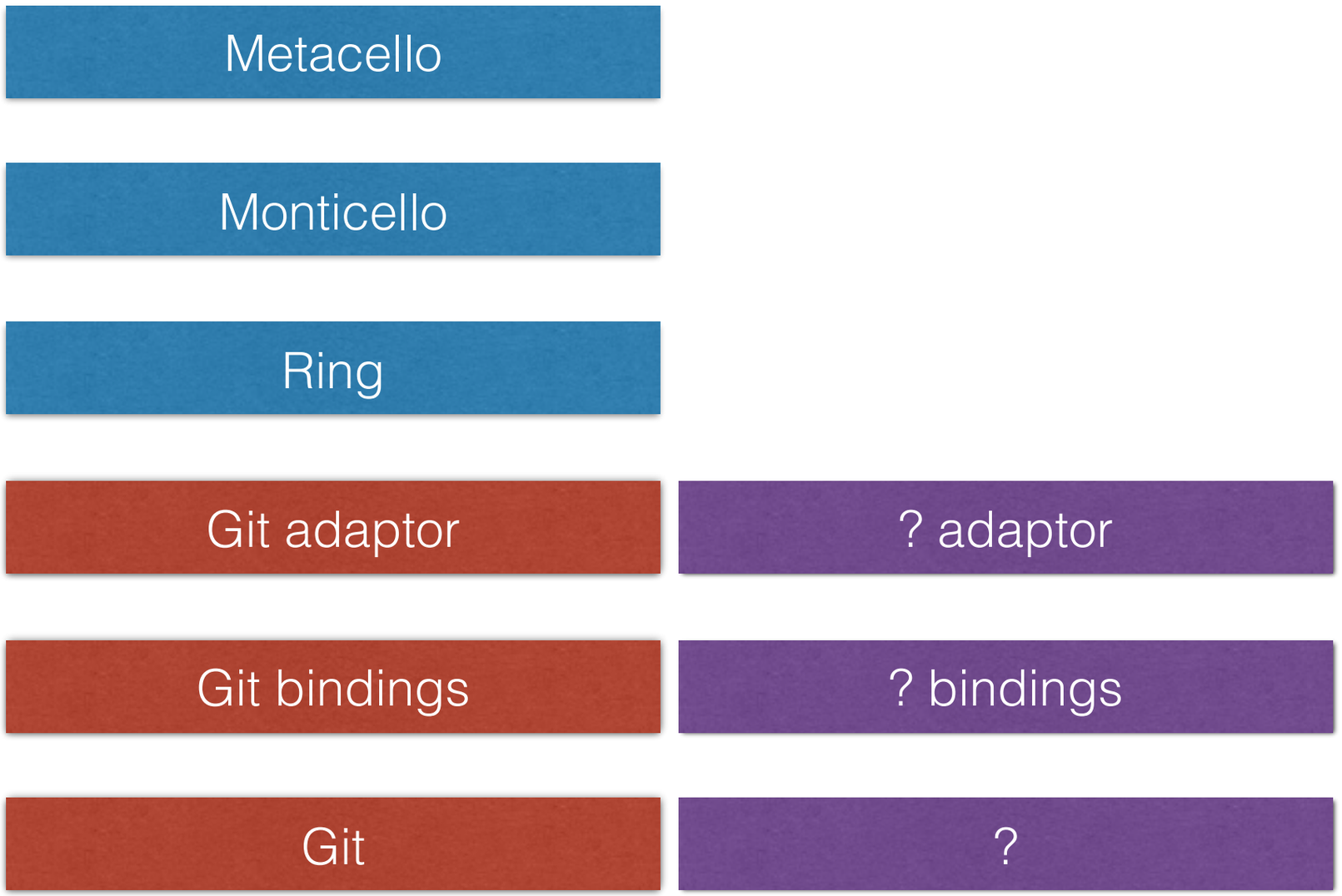
Monticello

Monticello meta model

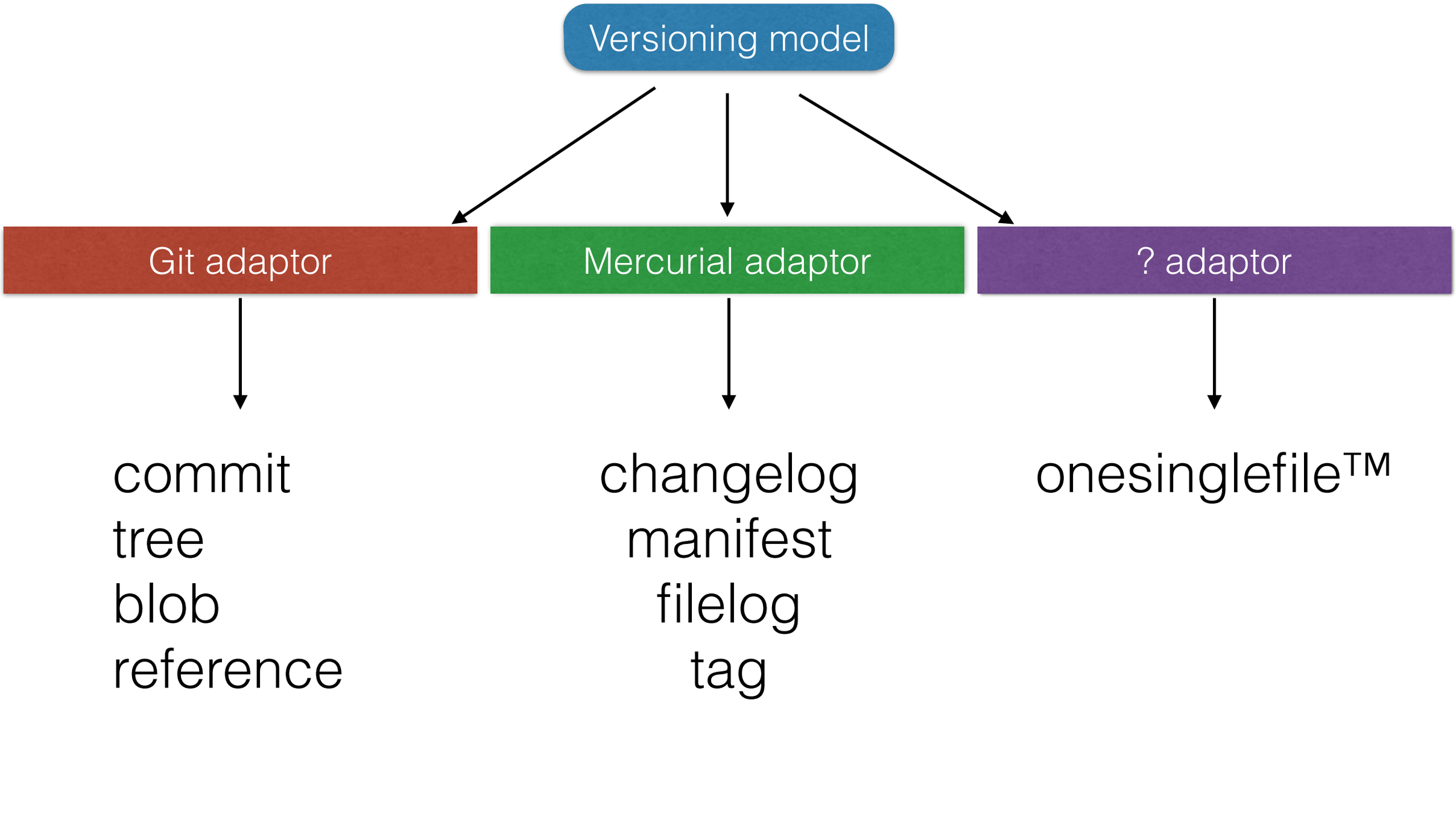
Possible future architecture



Possible future architecture



Model over matter (mostly)

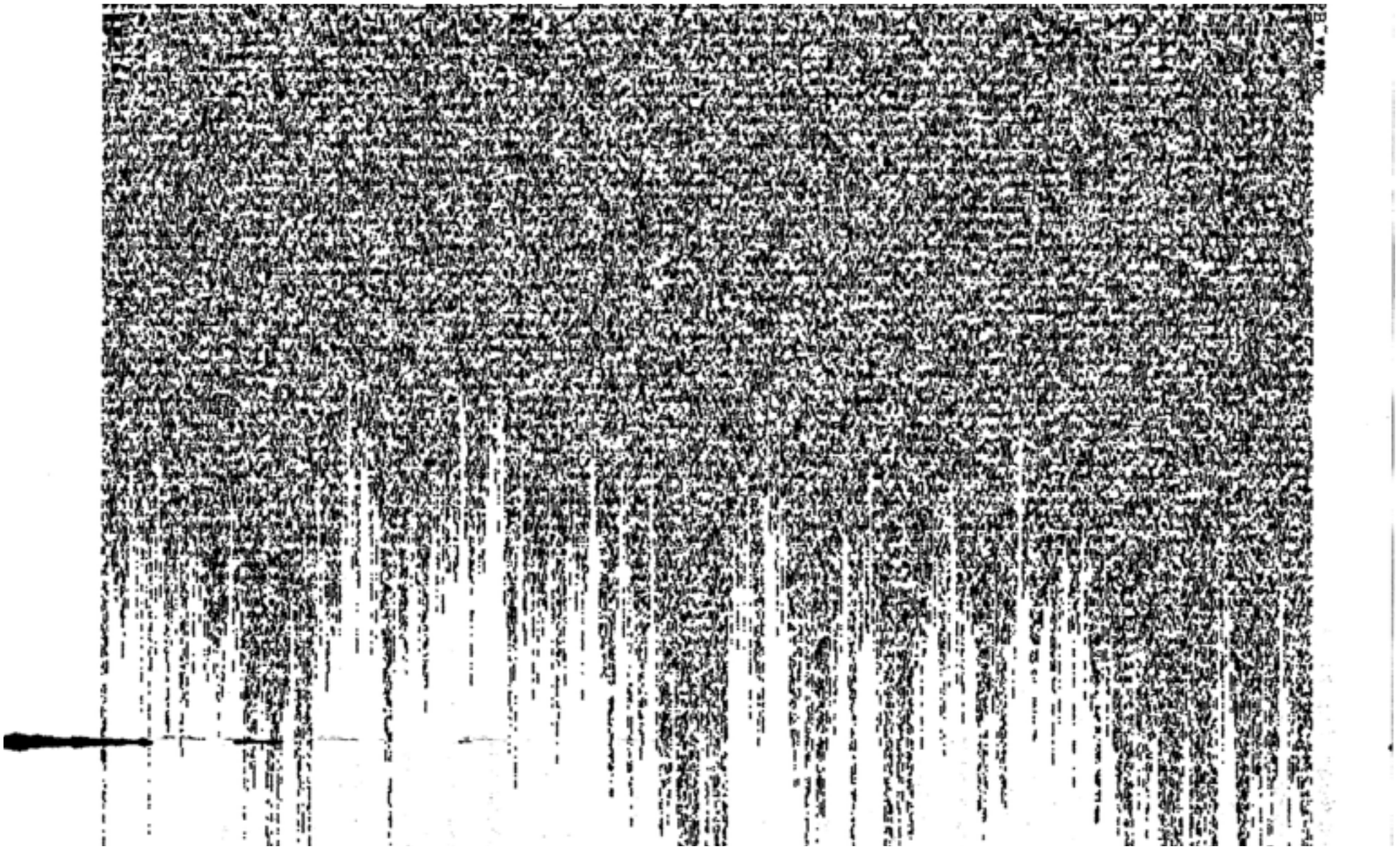


Why libgit2?

- well documented
- easy to understand
- fast response from community
- Git not required on host
- not maintained by us
- compiled to target platform
- no shell

Showing-off a bit

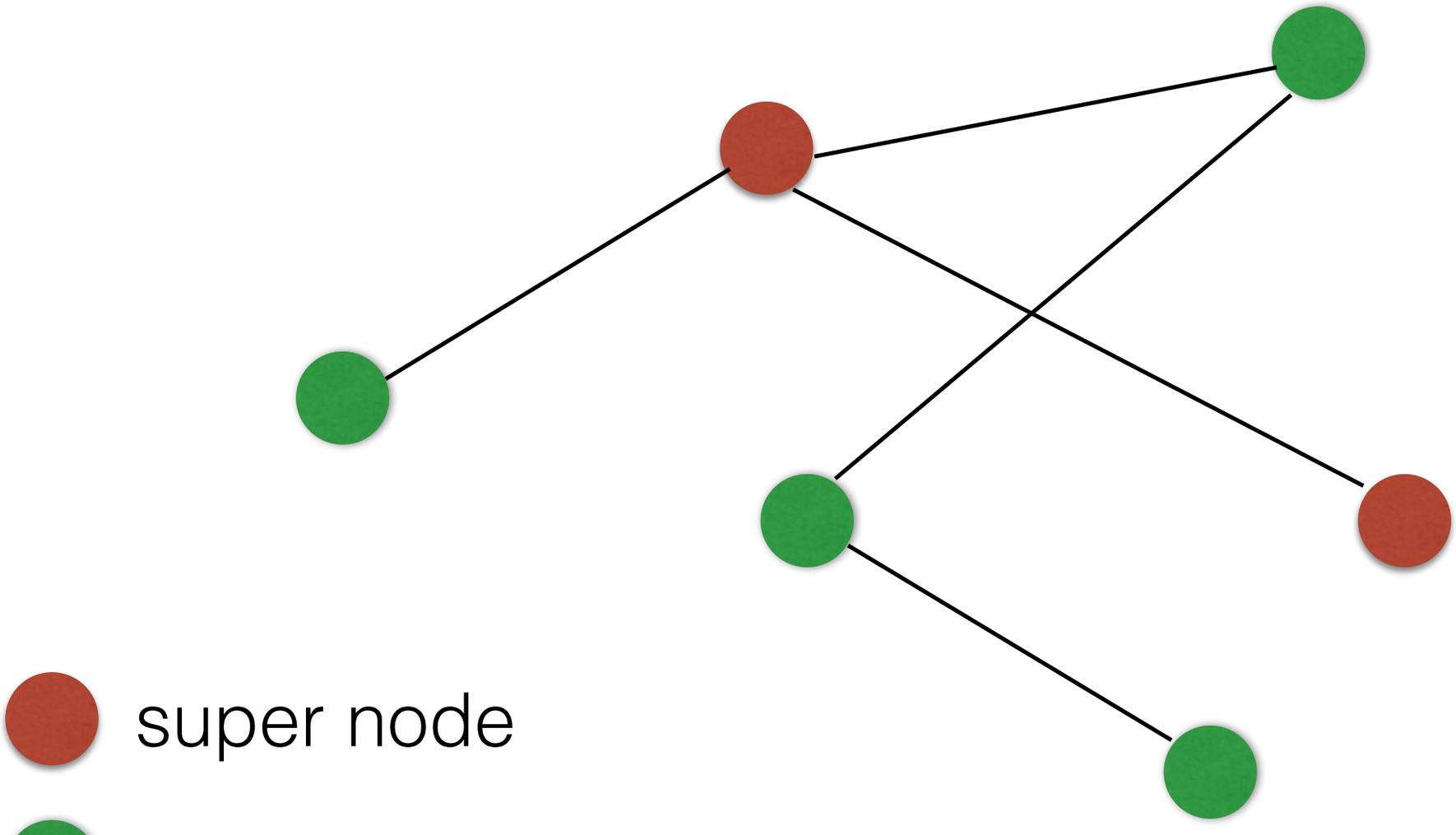
(Demo)



Some more bits

(random thoughts)

P2P backend



super node

peer

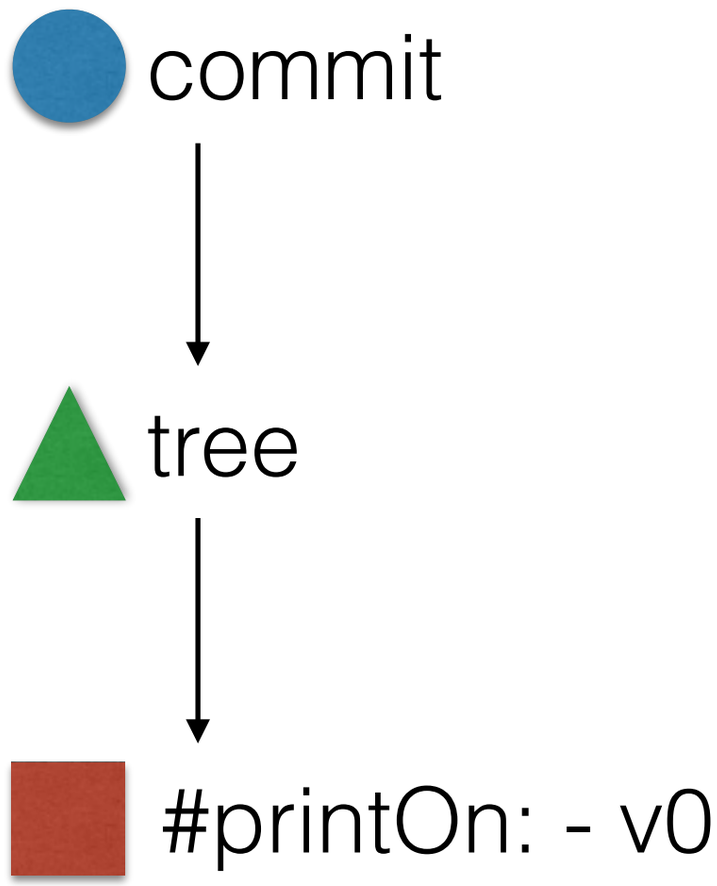
P2P backend: the good

- resilient
- independent
- don’t care about locality

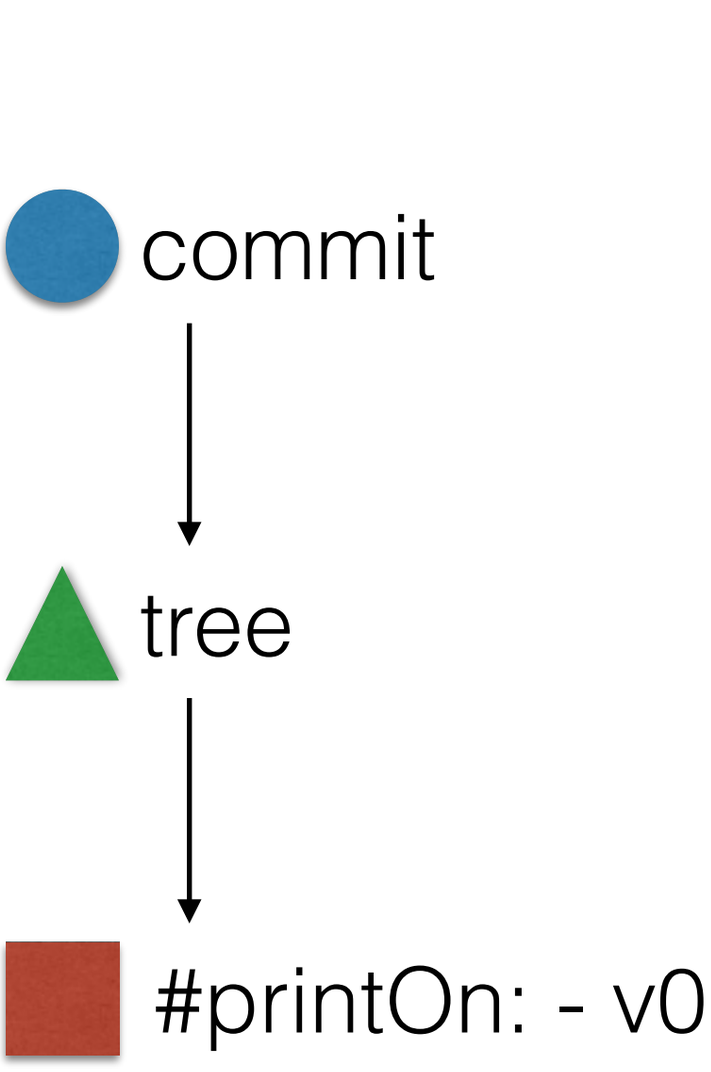
P2P backend: the bad

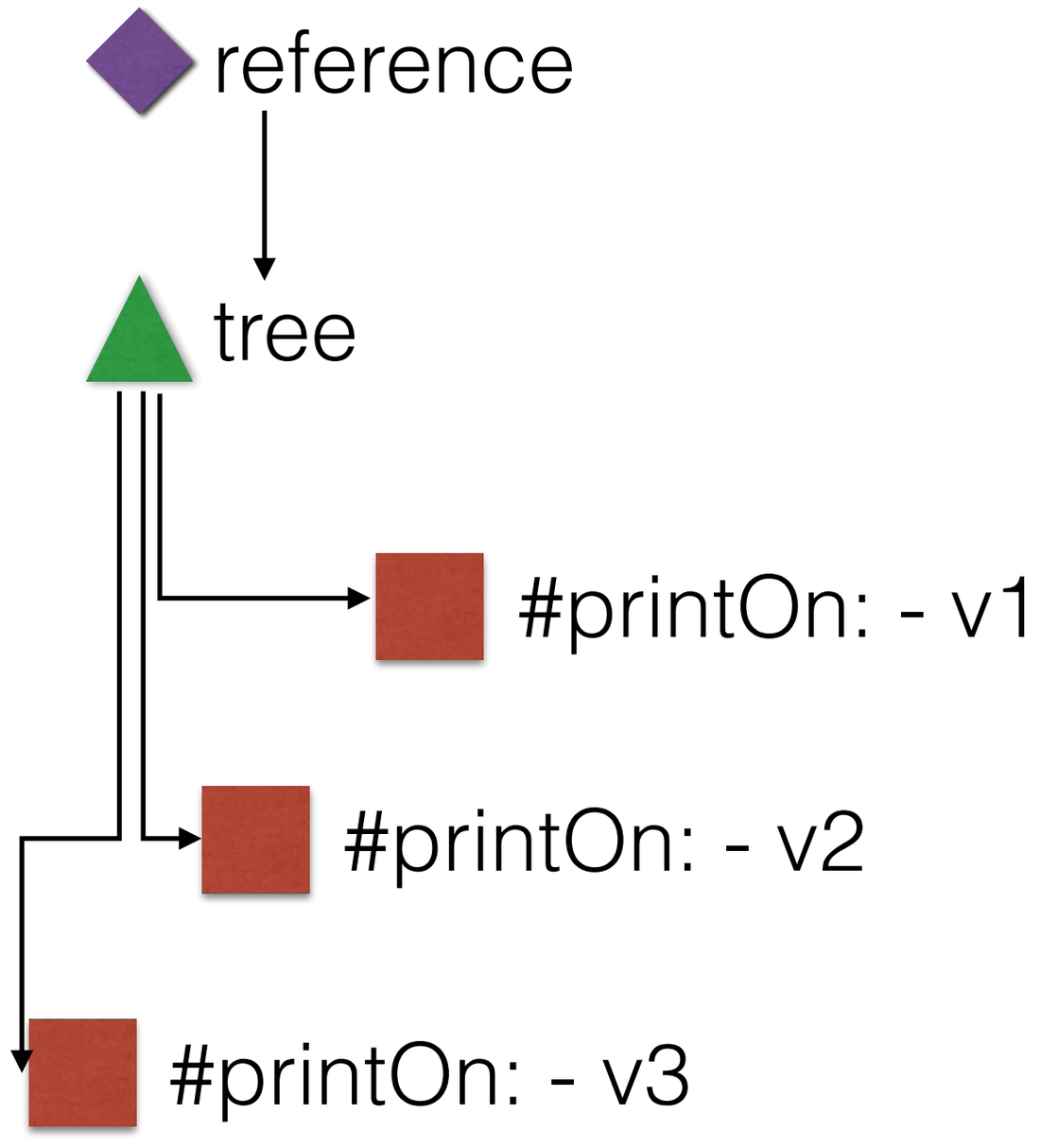
- not visible to outsiders
- not searchable from outside
- maintained by the community (super nodes)

Revision meta data in Git



Revision meta data in Git





Revision meta data in Git: the bad

- not manageable through Git
- need extra tools
- information only available to select few
- overhead (who will look at it)

Revision meta data in Git: the good

- collect data for research
- fine grained history
- don’t have to look at it if you don’t want to

A bit of a wrap up

A bit of a wrap up

- come a long way
- still a long way to go

A bit of a wrap up

- choice of storage backend
- better modularity (changing components)
- use existing solutions
- take load off community

A bit of a wrap up

- Git + abstractions: first step

- access to Git important (not just for revision control)

Acknowledgements

- Esteban Lorenzano
- Martin Dias
- Stéphane Ducasse
- Dale Henrichs
- Camillo Bruni

No more bits.

Thanks for your attention