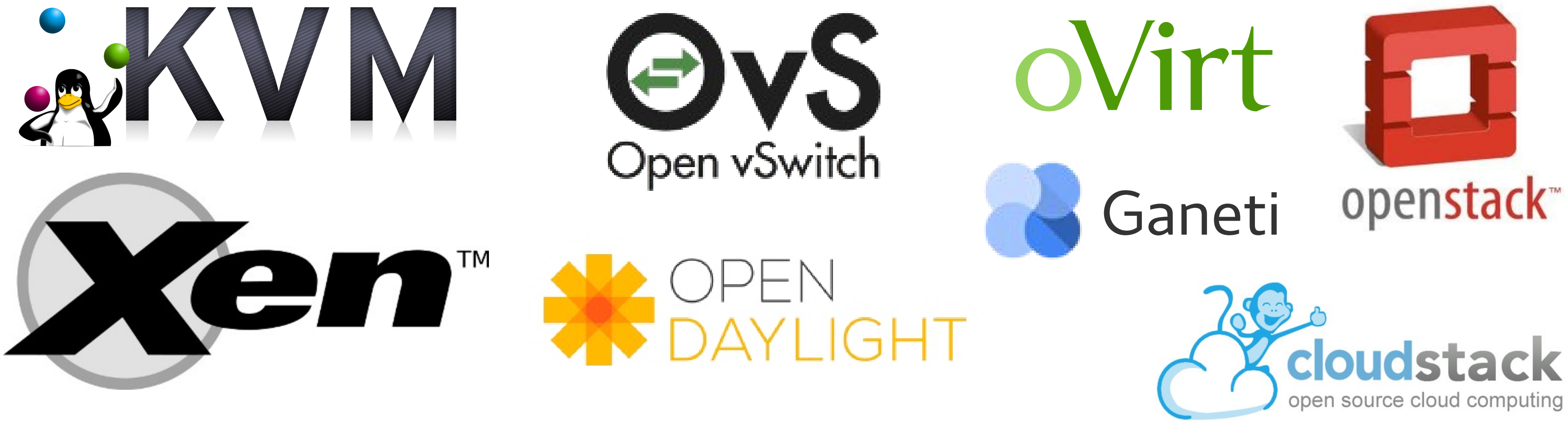
Swimming Upstream

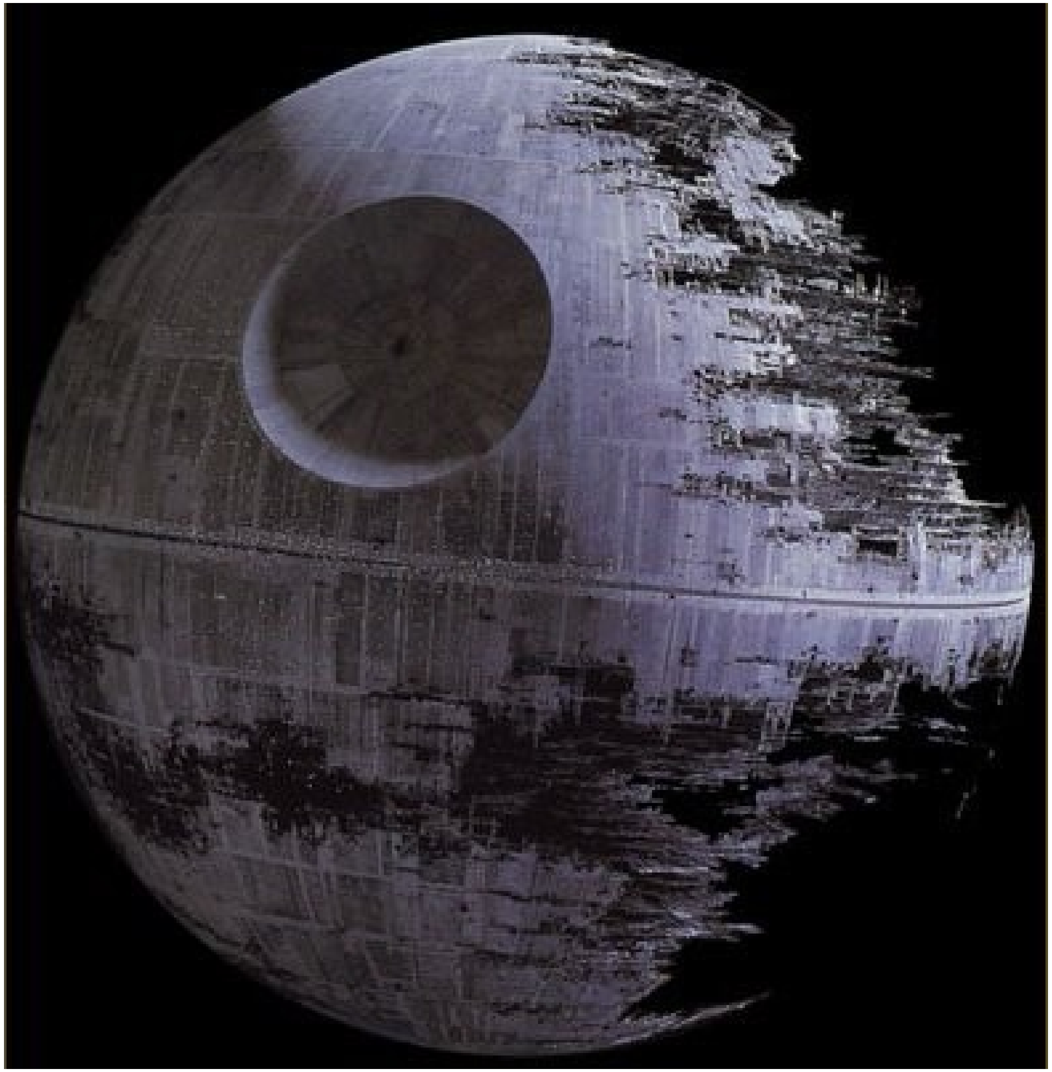
Dave Neary, Red Hat
dneary@redhat.com
@nearyd

Why bother?



Open source is everywhere!

puppet
labs
OPENSHIFT
origin
ceph
Chef
FOREMAN
CLOUD
FOUNDRY™

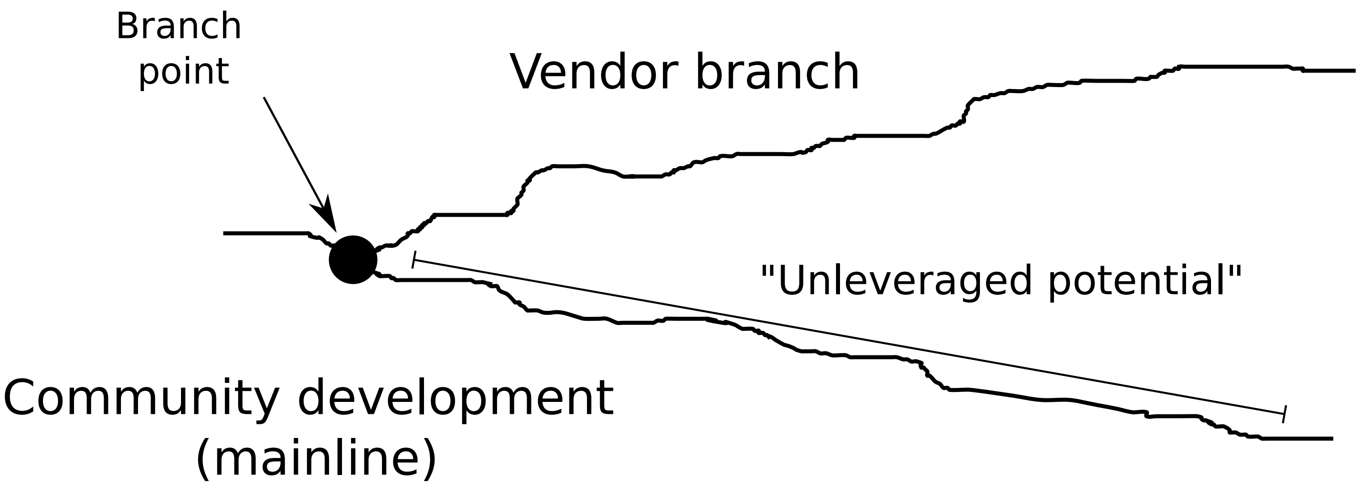


80% as good
as the last guy needed it to be

David Schlessinger

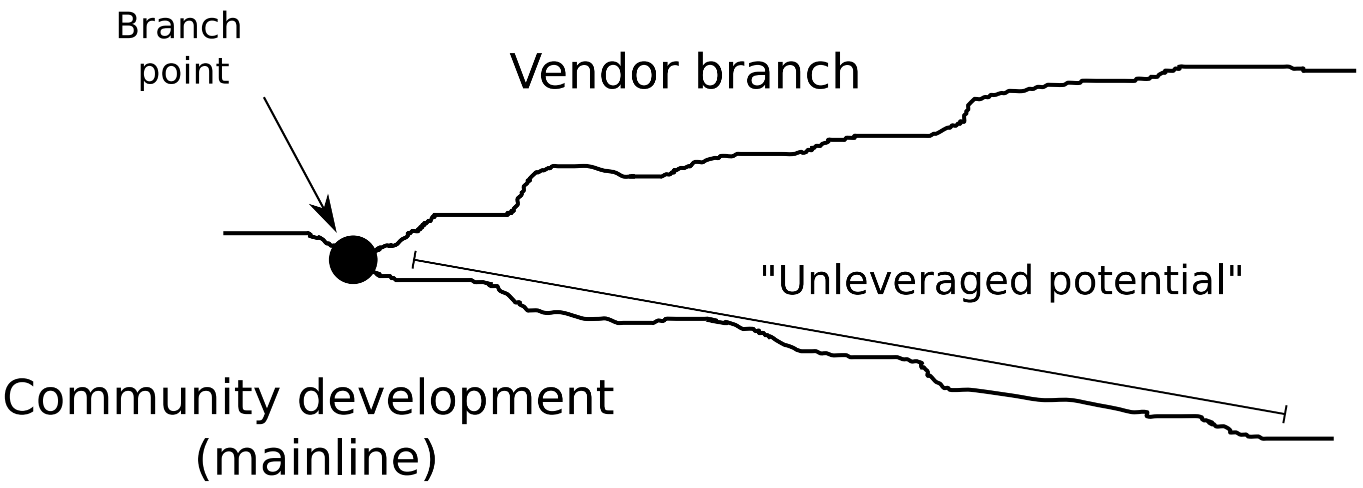
Building on open source projects

Branching strategy



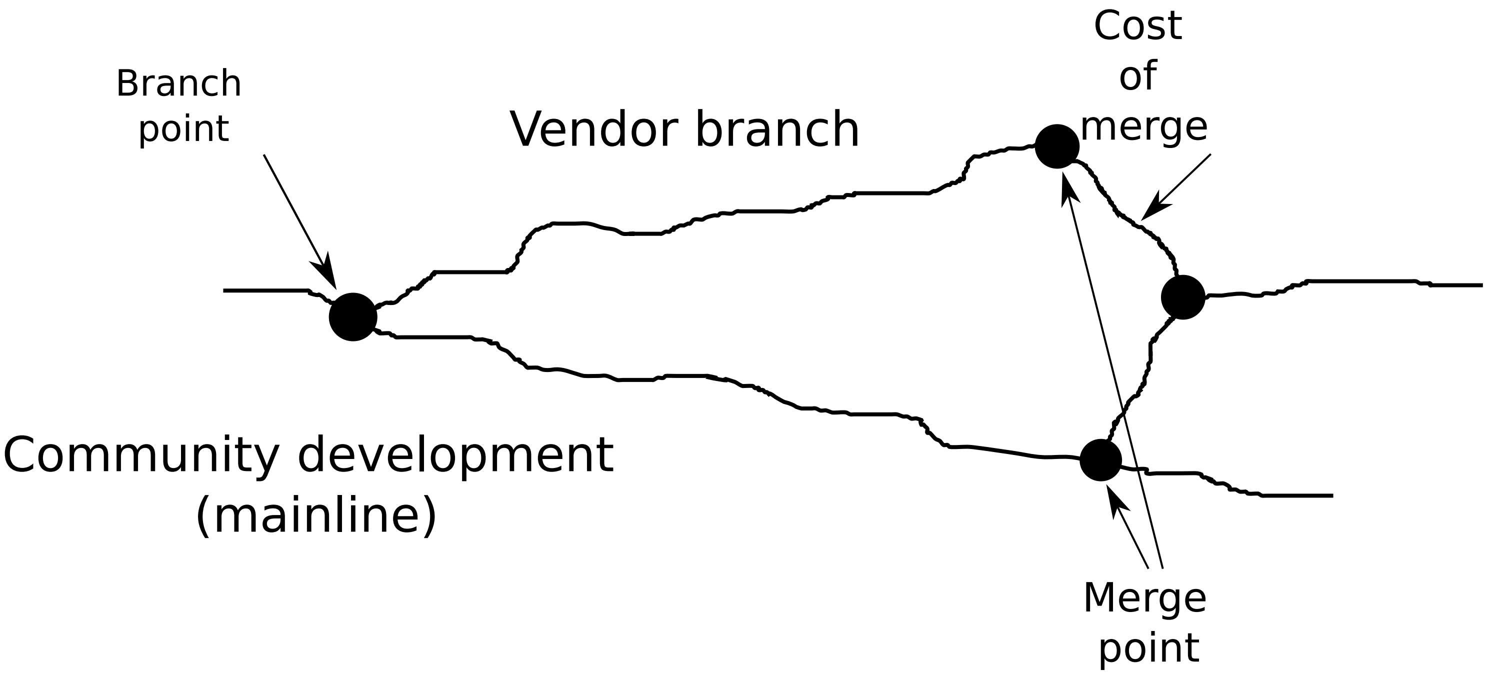
1. I branch, and do what I want

Branching strategy



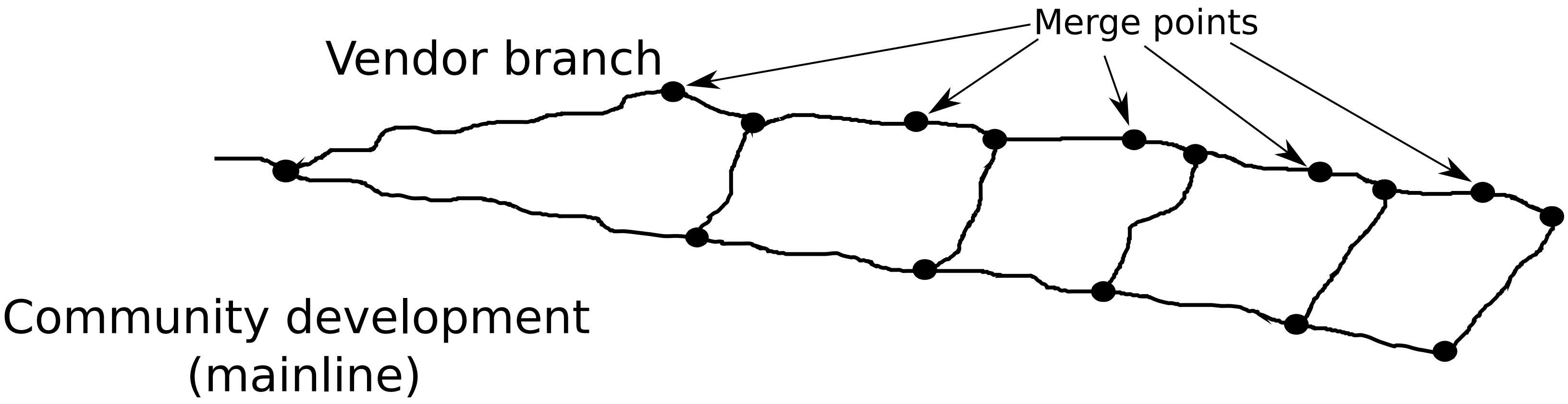
Cost: Work on vendor branch
Opportunity cost of upstream work

Branching strategy



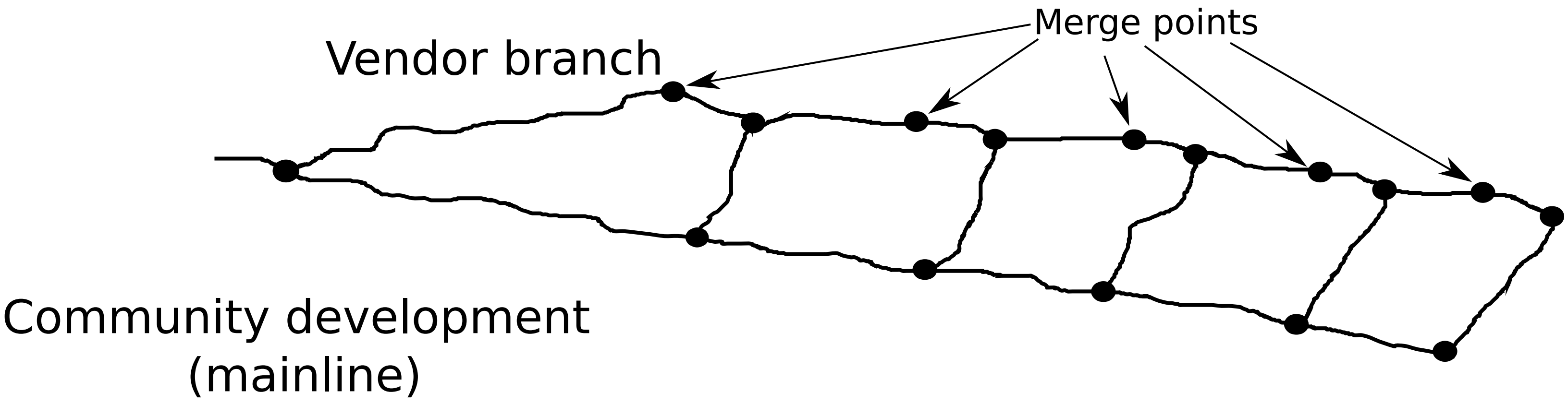
2. Branch and rebase

Branching strategy



...and rebase and rebase and...

Branching strategy



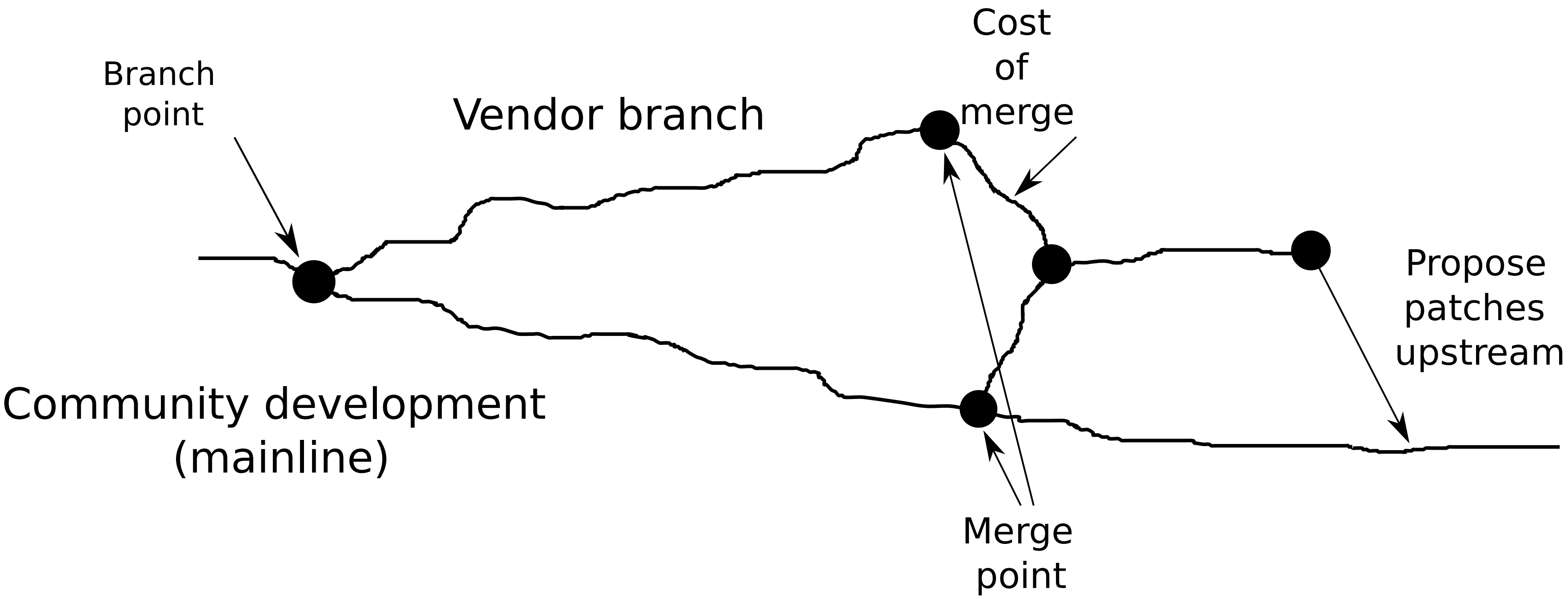
Merge points

Vendor branch

Community development
(mainline)

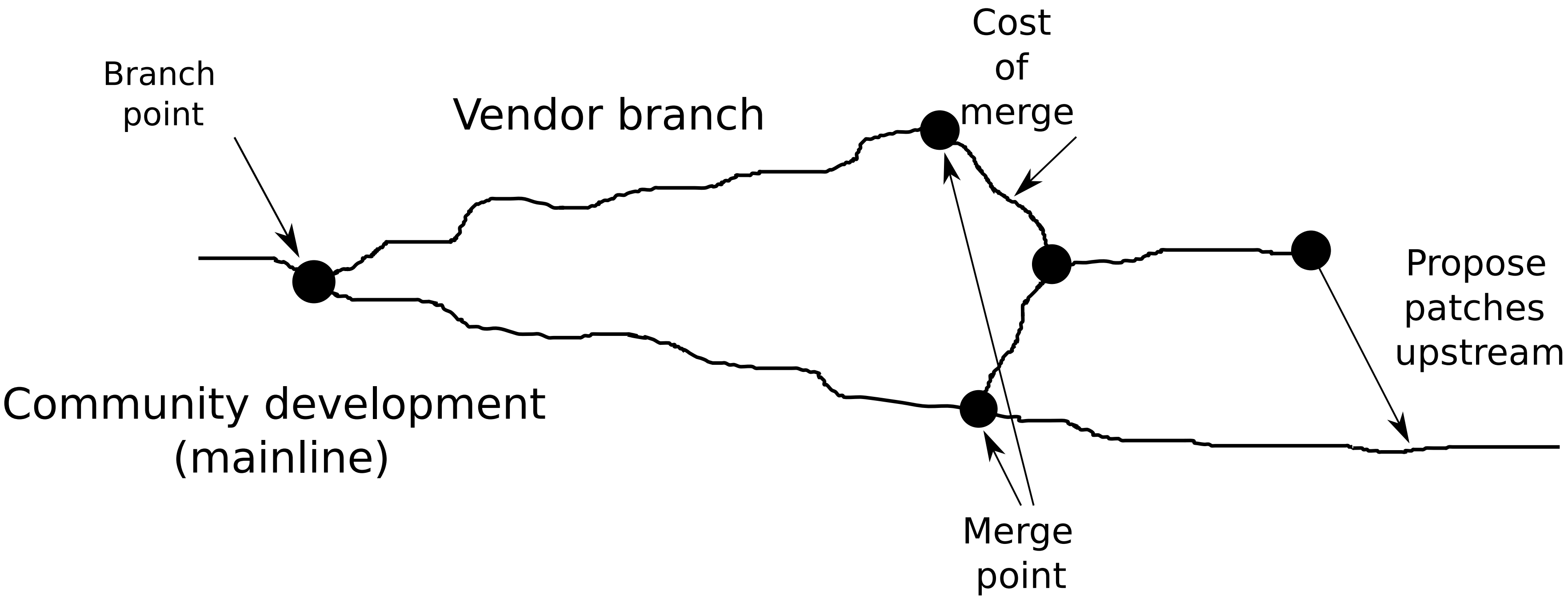
Cost: Vendor work + cost of repeated merging (maintenance)

Branching strategy



3. Branch, rebase, and upstream

Branching strategy

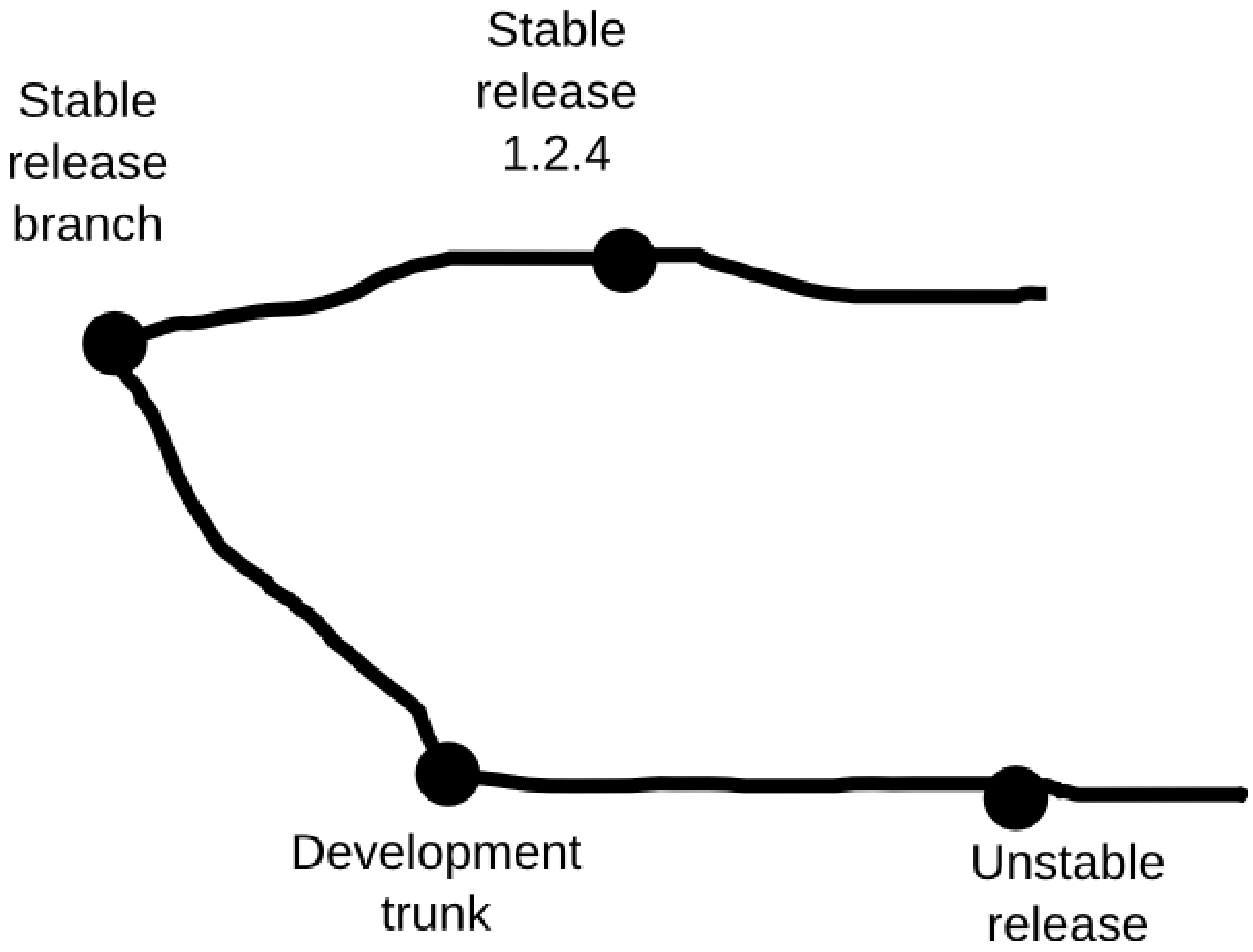


Cost: Vendor work + cost of merge + “community overhead”

“Community overhead”?

Our communities expect lots of things which companies would not otherwise do

Standard branching

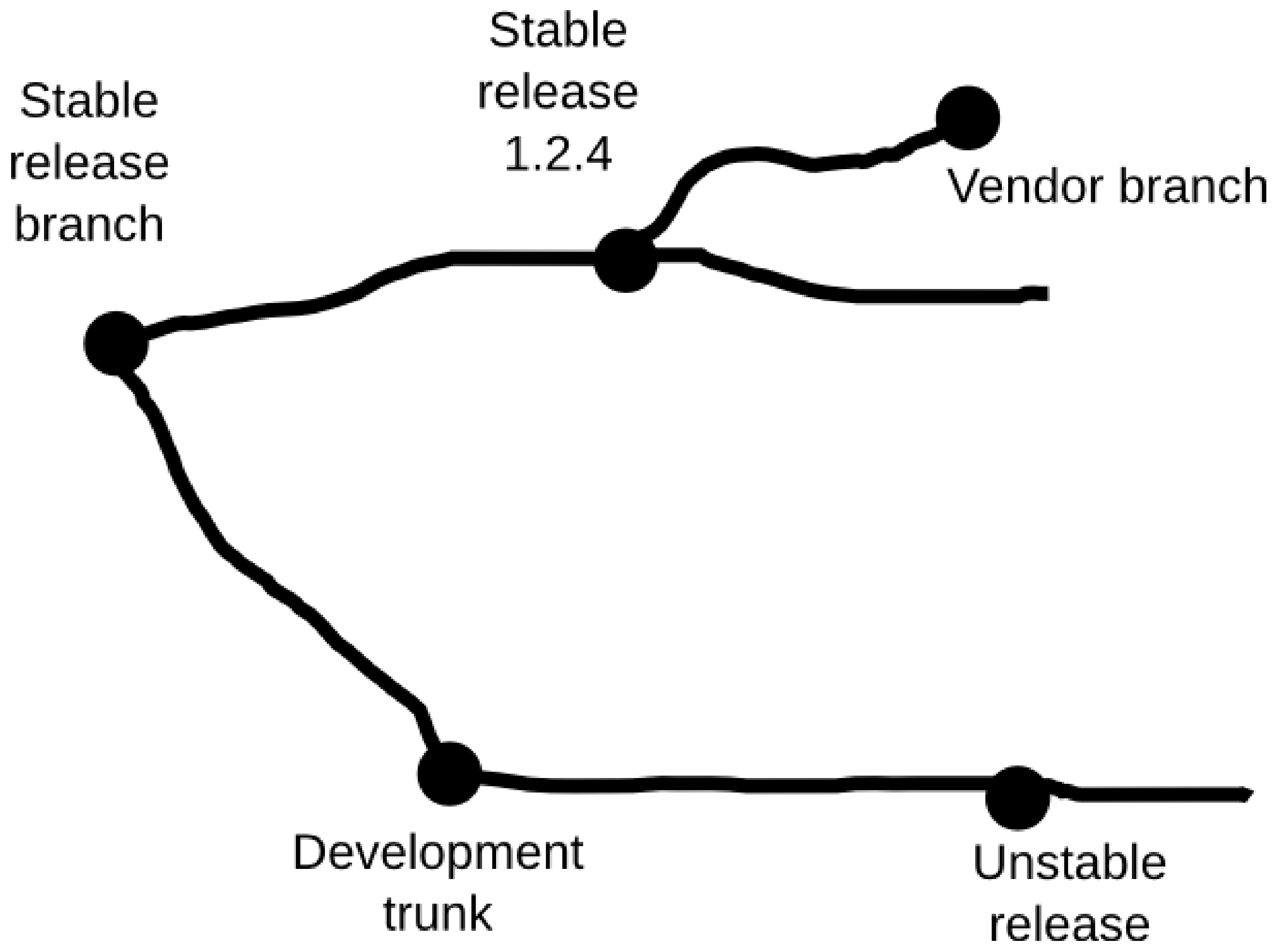


Stable release branch

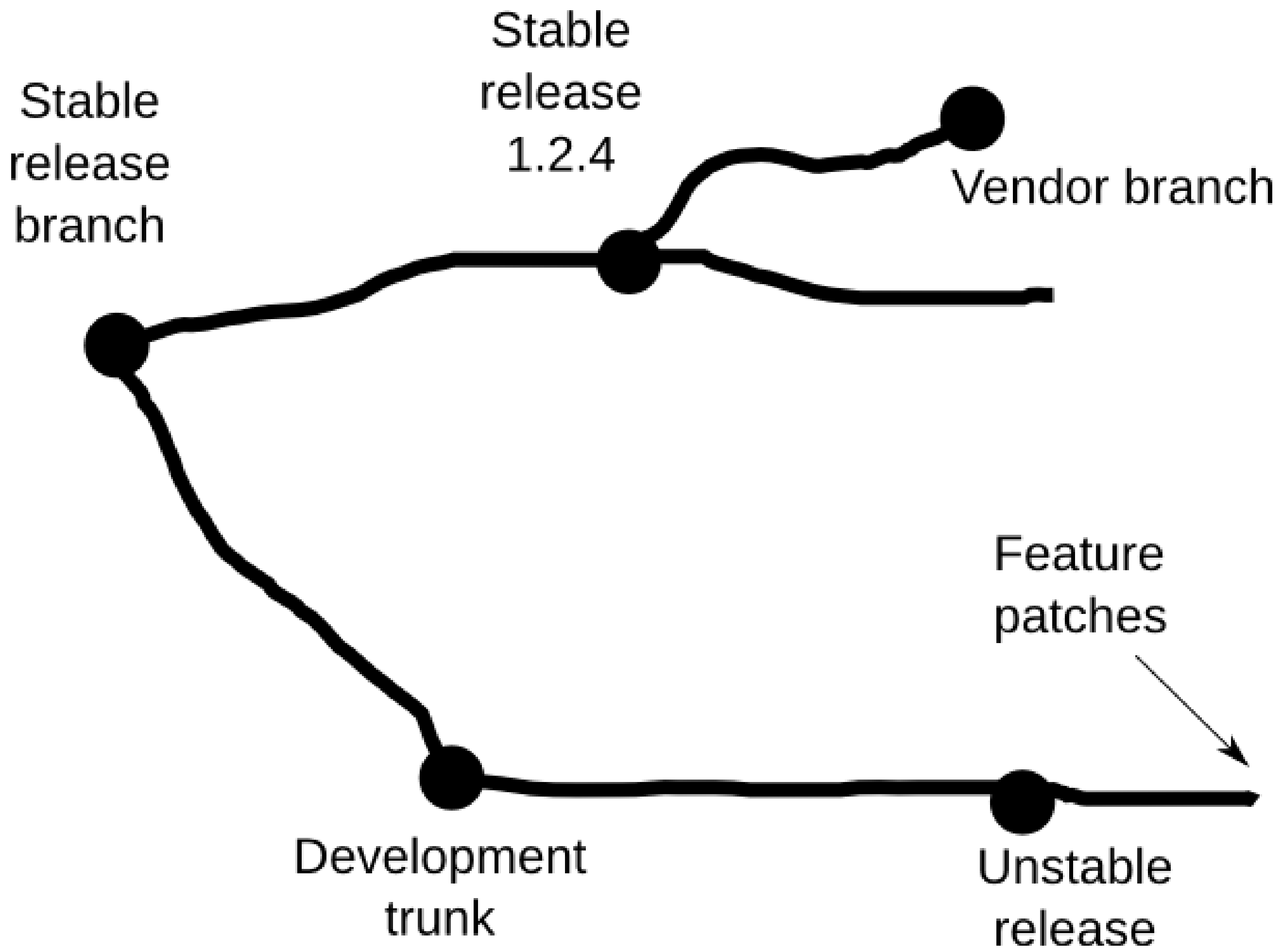
Stable release 1.2.4

Development trunk

Unstable release



Vendors want to build on a stable base



Community wants patches to trunk

Case study: Wakelocks

2005: Google acquires Android

2009: Arve Hjønnevåg proposes Wakelocks for inclusion in kernel

Initial proposal rejected, with comments

Google™

2009: Several updates also rejected

Debate dies down for a year

Google

2010: Wakelocks and alternative “autosleep” from Rafael Wyszocki “debated”* on LKML

* >1500 email thread

Autosleep patches merged,
wakelocks still outside kernel

Android team members have spent literally hundreds of man hours (my mail folder on the suspend blocker thread has over 1500 mail messages, and is nearly 10MB), and have tried rewriting the patches several times, in an attempt to make them be main-line acceptable.

Aug 2010, Theodore T'so

DiBona said there were some developers at Google working on it who “feel burned” by the decision but he acknowledged that the “staffing, attitute and culture” at Google isn’t sufficient to support the kernel crew.

Apr 2010, Paula Rooney, ZDNet

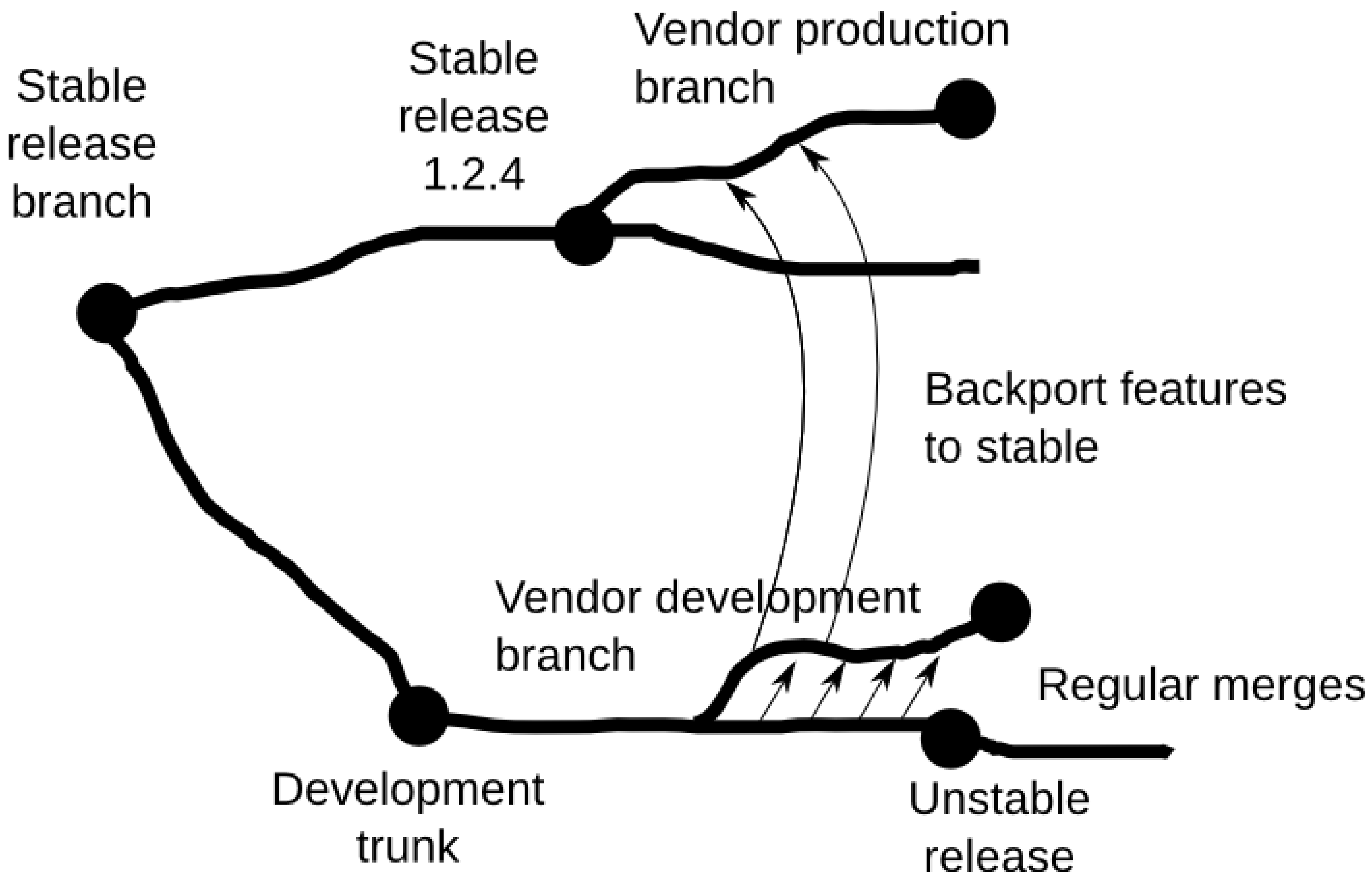
“Getting code into the kernel is always easier if you have a recognised name associated with it”

Matthew Garrett, LinuxCon 2010

“Do NOT fall into the trap of adding more and more stuff to an out-of-tree project. It just makes it harder and harder to get it merged. There are many examples of this.”

Andrew Morton

Ideal situation



This is what **upstream** does

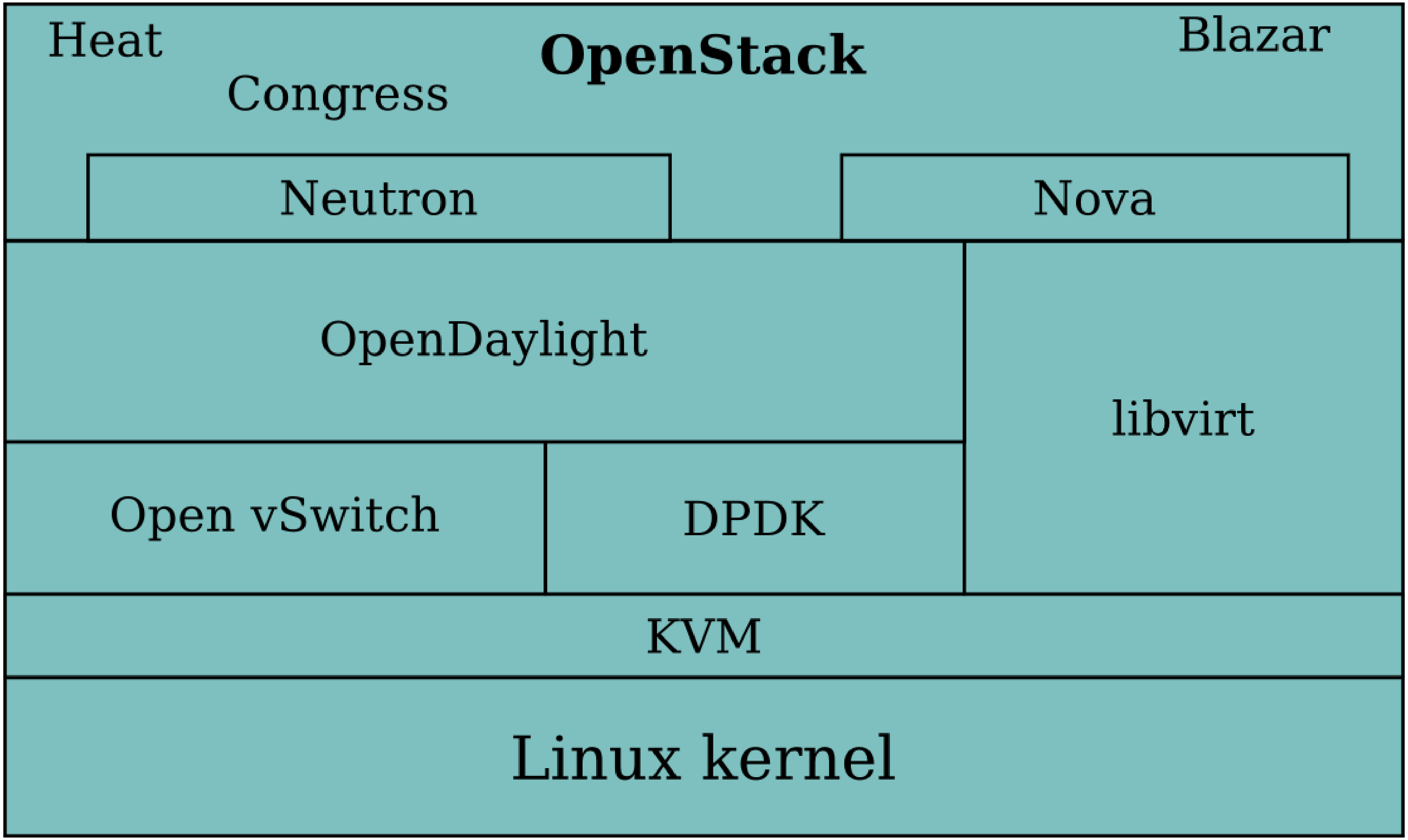
Difficulties

- Hard to sync product & project release dates and features
- Building a castle on quicksand
- Basically 1.5x to 2x the work up-front

Difficulties

Why is this relevant for OPNFV?

NFV is not just OpenStack



OpenStack

Heat

Congress

Blazar

Neutron

Nova

OpenDaylight

libvirt

Open vSwitch

DPDK

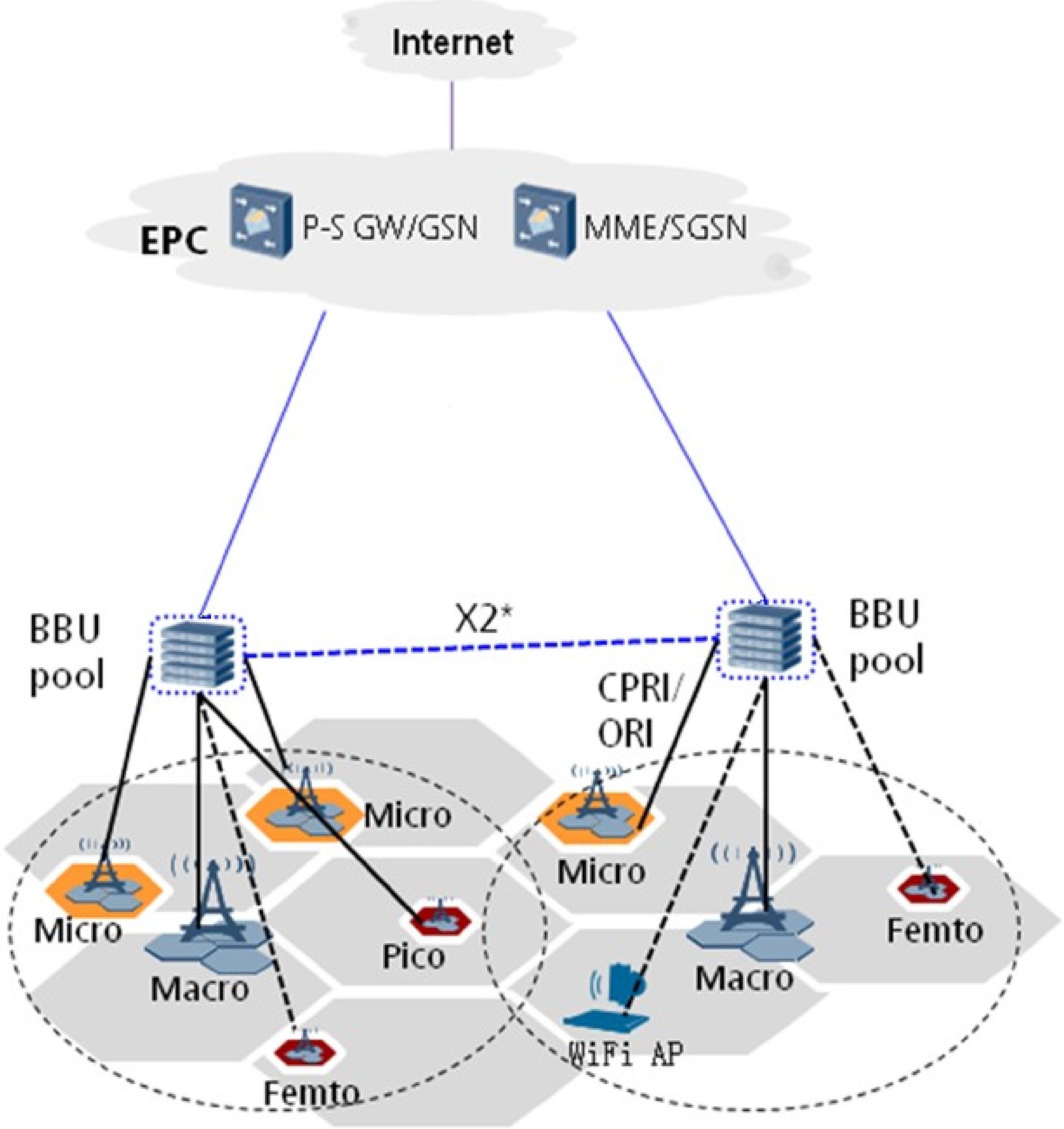
KVM

Linux kernel

Example: SR-IOV support

- Performance: VMs to be allocated an SR-IOV Virtual Function on capable hosts
- Requires NUMA awareness in Nova, libvirt, QEMU to ensure VM is on the same NUMA segment as the VF
- Improved SR-IOV support in libvirt (VM placed in same NUMA node as VF)
- PCI pass-through support for Nova
- User interface and policy enforcement

Virtualized Mobile Base Station



- Multiple RAN technologies from multiple vendors to be consolidated on a single BS to improve utilization
- BS virtualization can share resources among multiple logical RAN nodes from different systems.
- C-RAN for efficient resource utilization among different physical BSs.
- Baseband Unit (BBU) pool with high performance servers and real-time processing for signaling capacity

(ETSI #4)

Example: C-RAN/vRAN

- Determinism
  - Real-time patches to KVM
- Performance
  - DPDK accelerated Open vSwitch
- Reliability at scale
  - HA instances in OpenStack Nova
  - HA OpenStack
  - Fault management and reporting
  - IPv6 support – kernel through Nova, Neutron

Upstream first

Upstream first

- Maintaining significant out-of-tree patches across multiple projects is unsustainable
- Features should be designed, written and proposed with upstream developers
- Cascading changes across multiple projects require explanation and advocacy

Upstream First

- Requires discipline (small changes, discussed first)
- Relationships (your priority might not be their priority)
- Many communities – not just OpenStack!

Not always easy!

Steps to success

1. Empathy

Bug: assert(int+100 > int) optimized away

c2: “signed type overflow is undefined by the C standard”

c4: “You have GOT to be kidding? [...] You are missing the point here. [...] PLEASE REVERT THIS CHANGE. [...] THIS IS NOT A JOKE. FIX THIS! NOW!”

c5: “I am not joking, the C standard explictly says signed integer overflow is undefined behavior.”

c9: “Hey Andrew, do you really think this issue goes away if you keep closing the bugs fast enough?”

...

c48: “Oh wow, another wise cracking newbie who comments without actually understanding the issue. I AM NOT RELYING ON UNDEFINED BEHAVIOR.”

...

https://gcc.gnu.org/bugzilla/show_bug.cgi?id=30475

2. Framing

Moving in

New job

Start small

- Listen for a while
- Introduce yourself
- Ask questions with an open mind

Make friends

- Relationships are key
- Learn cultural norms

3. Mapping the lay of the land

Identify the leader(s)

Case study: OpenDaylight

OpenDaylight community structure

- http://www.opendaylight.org/community/how-participate - “Getting started guide, “Developer wiki”
- Technical Steering Committee: https://wiki.opendaylight.org/view/TSC:Main
- Project meetings: Attend TSC meetings
- Project list: https://wiki.opendaylight.org/view/Project_list
- Active IRC channel: #opendaylight on Freenode

Tools to help identify leaders

- Source code analysis: https://spectrometer.opendaylight.org/
- Mailing list analysis: MLStats: http://metricsgrimoire.github.io/MailingListStats/

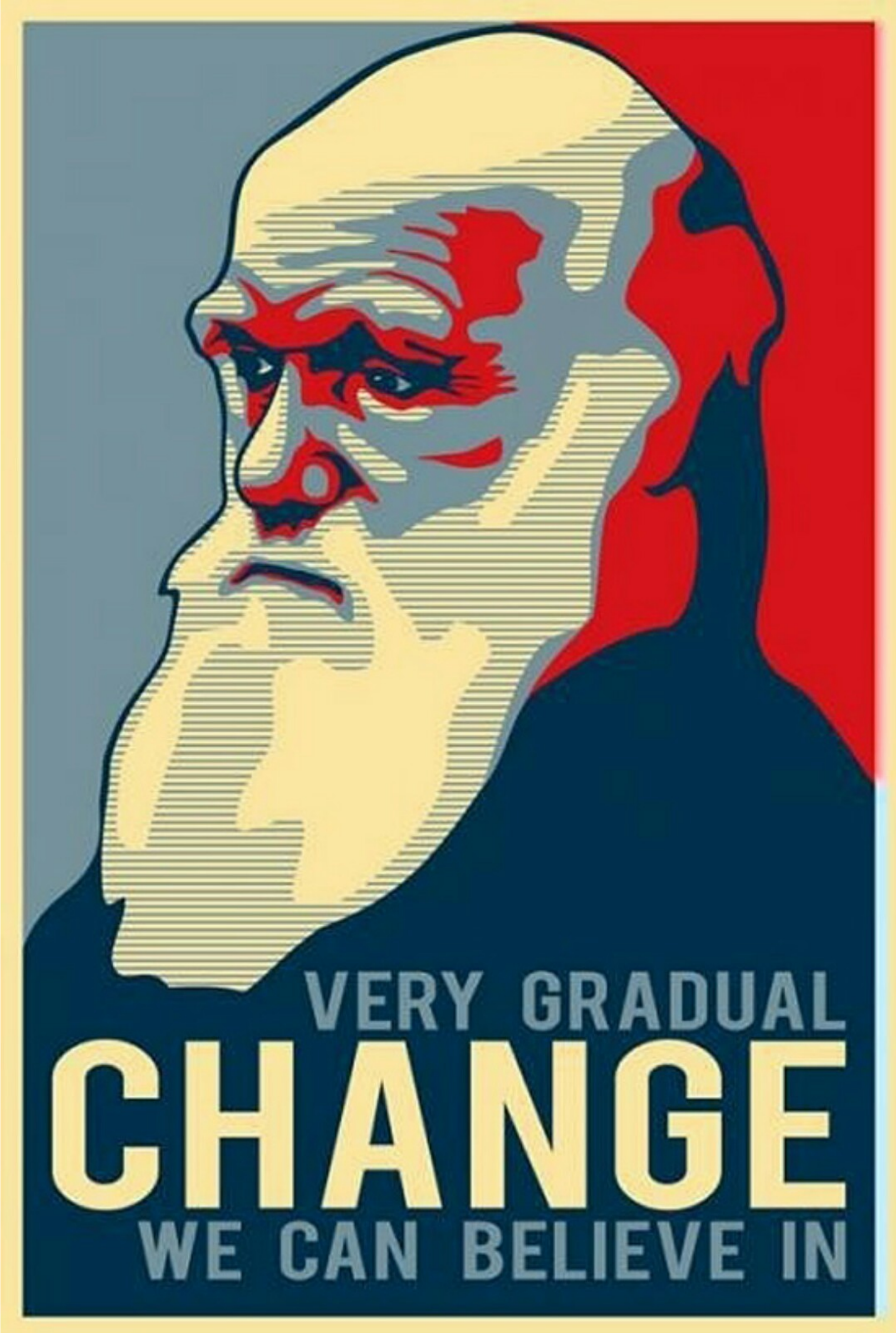
Infrastructure and Channels

Common communication infrastructure

- Mailing lists
- IRC
- Wiki
- Code review
- Source control

Community processes:
When in Rome...

4. Effecting change



Avoid the Water Cooler

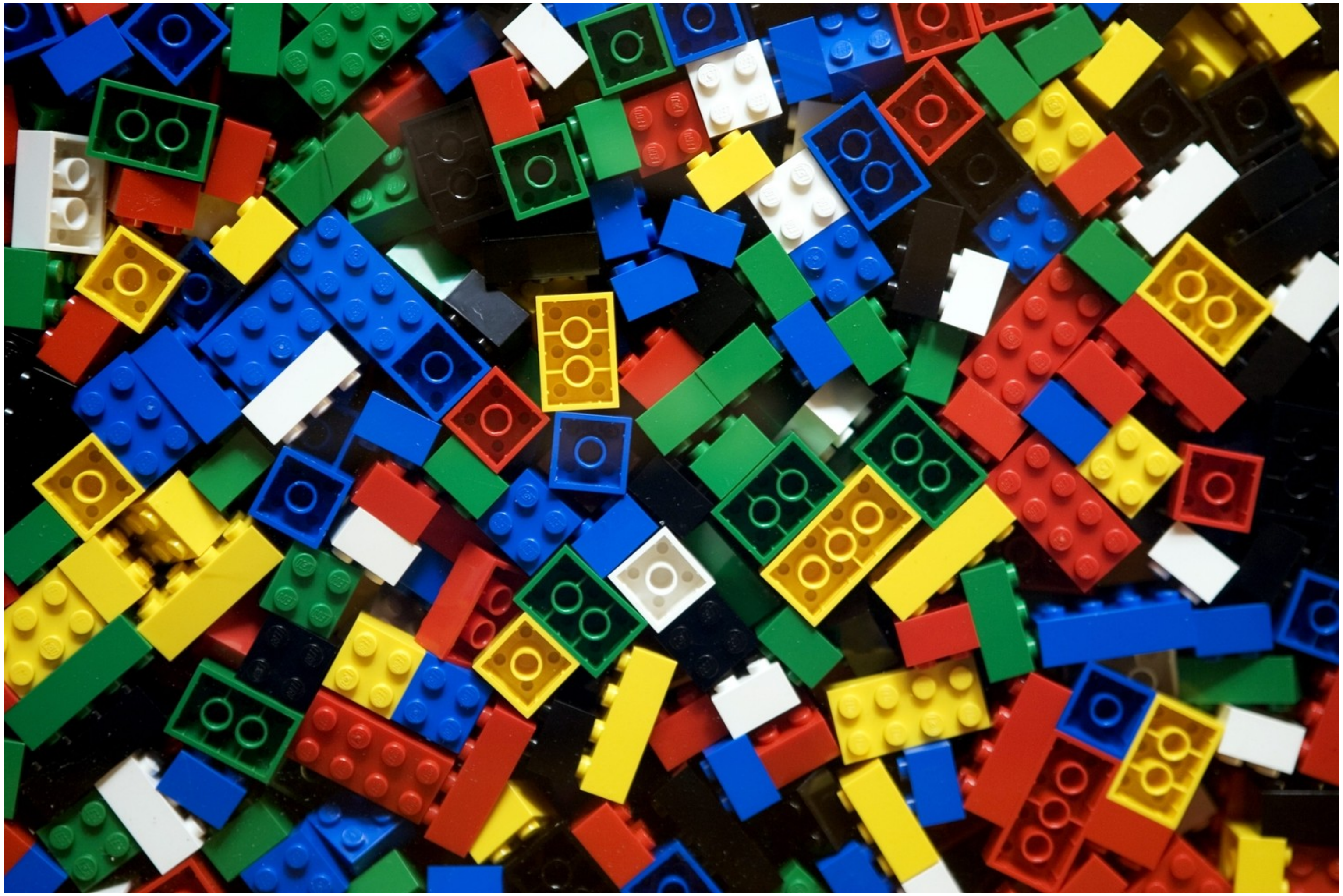
Open and transparent

Do not fear mistakes.
There are none.

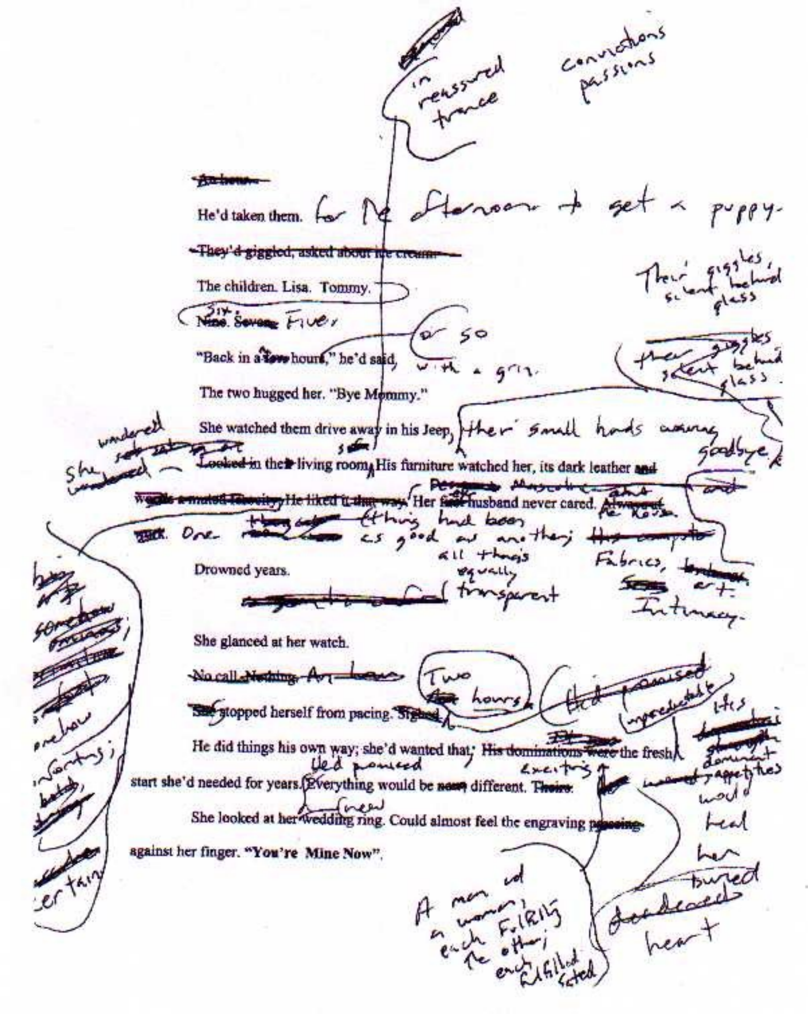
Miles Davis

Communicate the vision

Break things down



Showing your work



in reassured trance

convictions passions

~~An hour.~~

He'd taken them. for the afternoon to get a puppy.

~~They'd giggled, asked about ice cream.~~

Their giggles, silent behind glass

The children. Lisa. Tommy.

~~Nine. Seven.~~ Six. Five.

"Back in a ~~few~~ hours," he'd said, with a grin. or so

The two hugged her. "Bye Mommy."

She watched them drive away in his Jeep, their small hands waving goodbye.

She wandered ~~Looked in~~ their living room. His furniture watched her, its dark leather ~~and~~ Masculine and ~~[illegible] He liked it that way.~~ Her ~~first~~ husband never cared. ~~Always~~ ~~[illegible]~~. One thing had been as good as another; all things equally transparent ~~His~~ Fabrics, art. Intimacy.

Drowned years.

She glanced at her watch.

~~No call. Nothing. An hour.~~ Two hours.

~~She~~ stopped herself from pacing. ~~Sighed.~~ He'd promised unpredictable

He did things his own way; she'd wanted that; ~~His dominations were~~ the fresh exciting start she'd needed for years. He'd promised Everything would be ~~now~~ different. ~~Theirs.~~

His dominant appetites would heal her buried heart

She looked at her new ~~wedding~~ ring. Could almost feel the engraving ~~pressing~~ against her finger. "You're Mine Now".

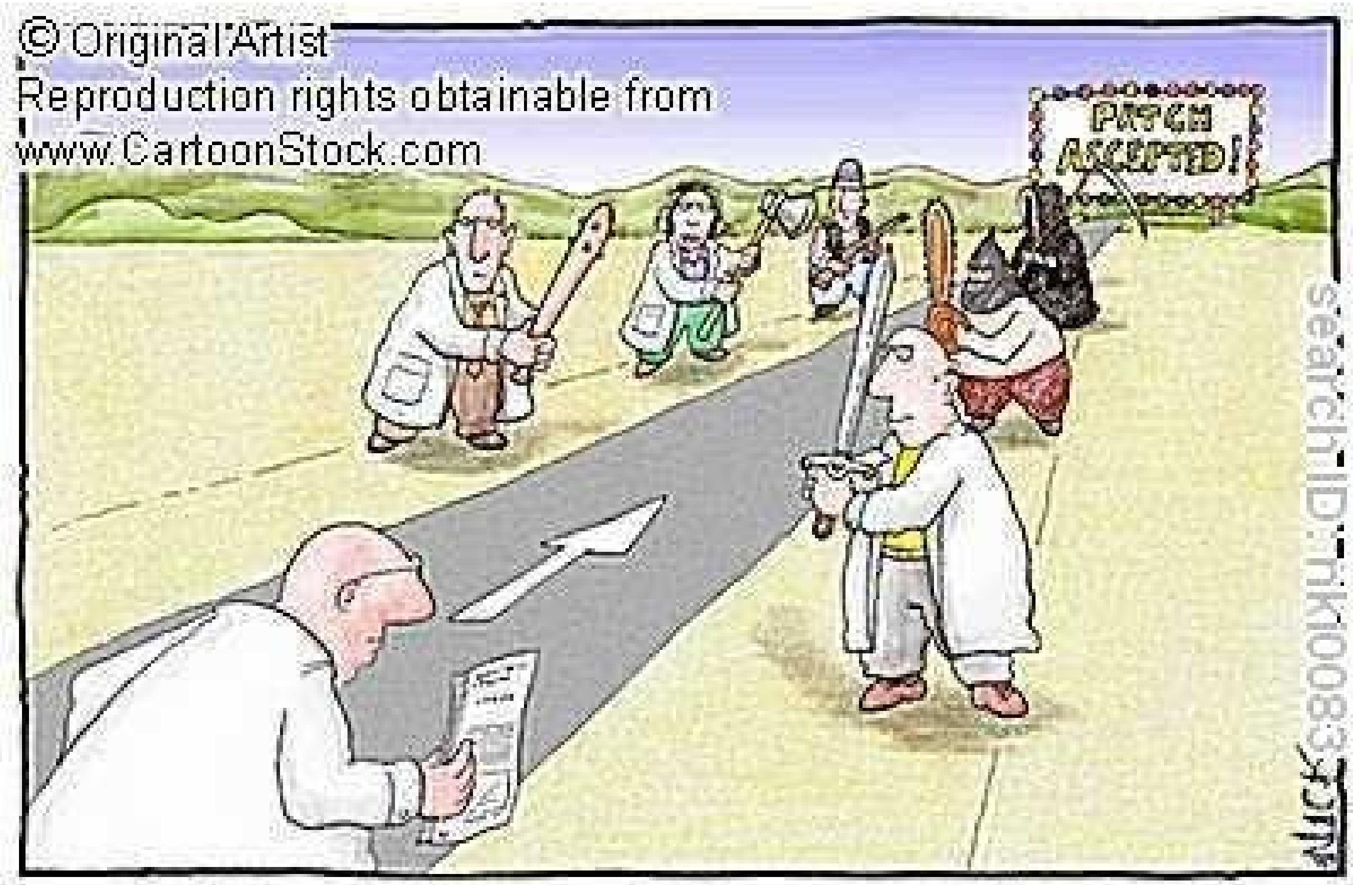
A man and a woman, each fulfilling the other; each fulfilled, sated

somehow certain

Optimise for reading

Rewrite history with git rebase --interactive

Rinse, repeat



Most scientists regarded the new streamlined peer-review process as 'quite an improvement.'

Keep your eyes
on the prize

Thank you

Dave Neary, Red Hat
dneary@redhat.com
@nearyd