The Cost of Going it Alone

Dave Neary
dneary@gnome.org

Also:

The Cost of Collaboration

Act I: Softway

1996-97

OpenNT/Interix: POSIX for NT

GCC suite work:

- 6-8 man-months
- Touched gcc, assembler, linker, gdb, ...
- Experienced compiler engineer

- Wanted to avoid maintaining a fork
- Approached upstream(s)
- Reactions ranged from “Cool!” to “NT? Not touching it.”

Cygnus SOLUTIONS

$120,000

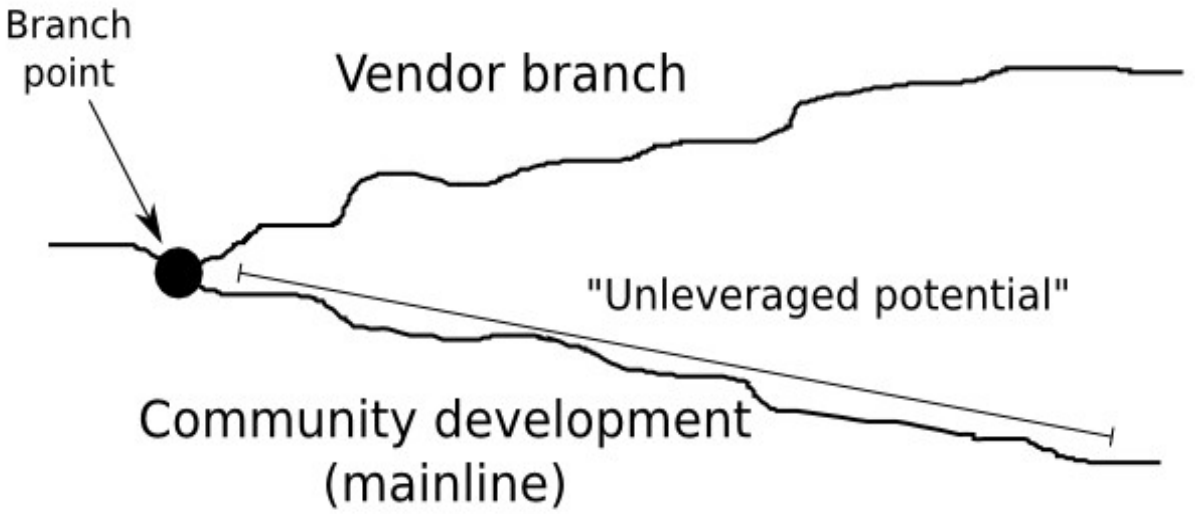
14 months lead time

$40,000

~6-8 man-months

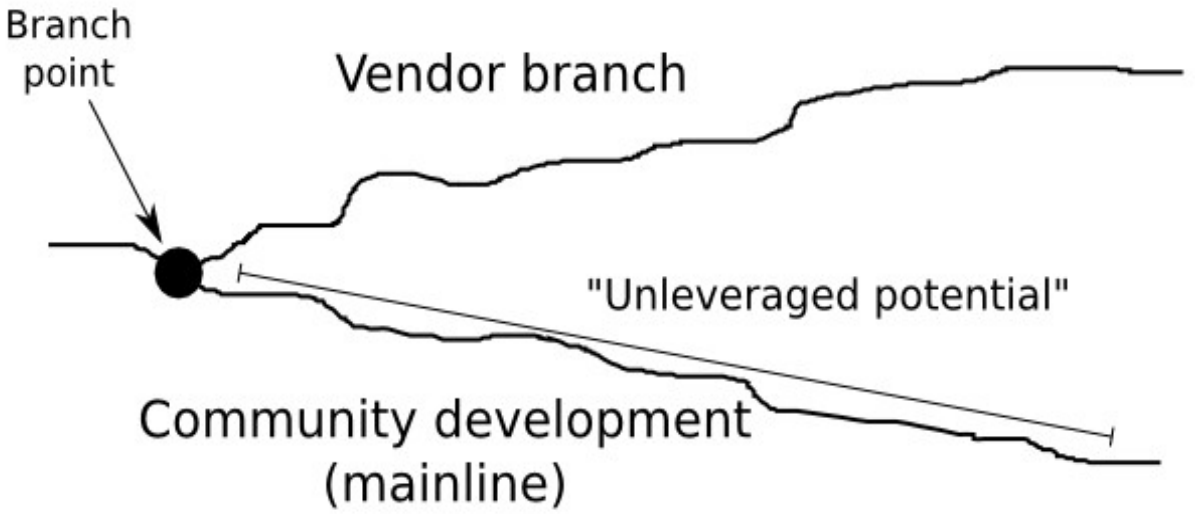
Getting Things Done

Branching strategy



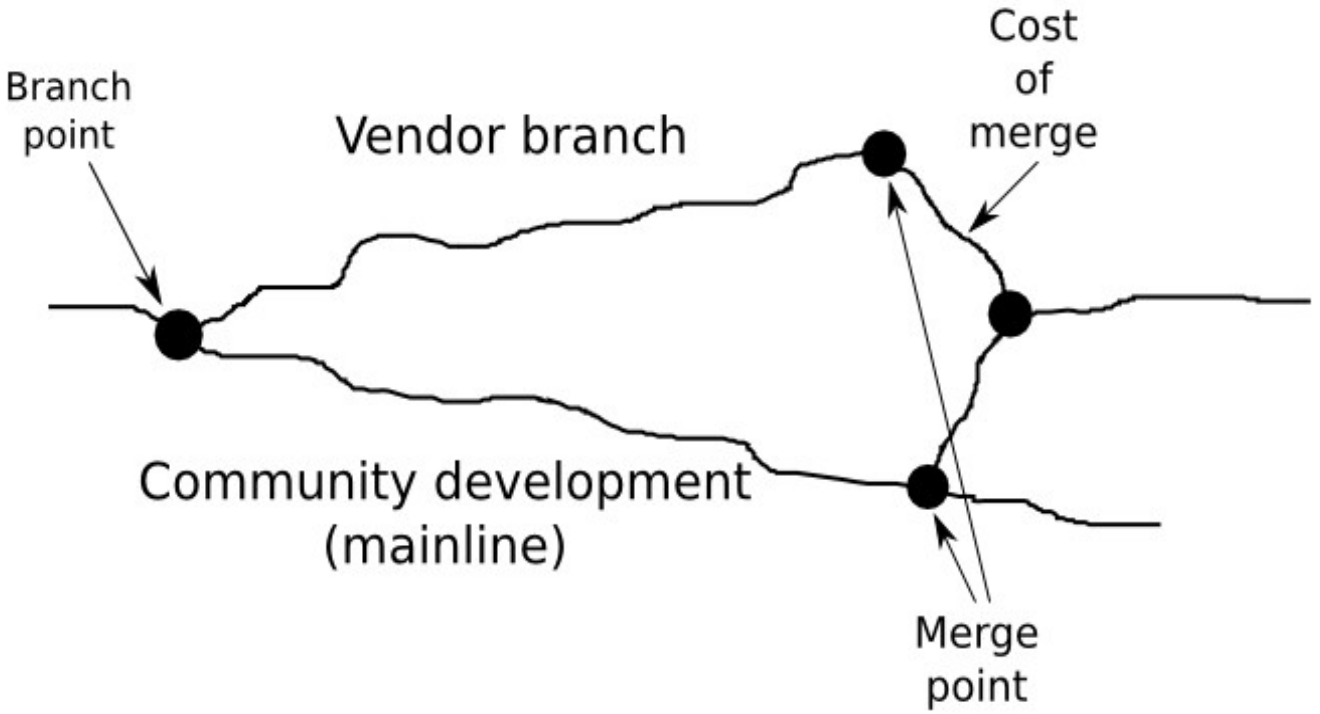
1. I branch, and do what I want

Branching strategy



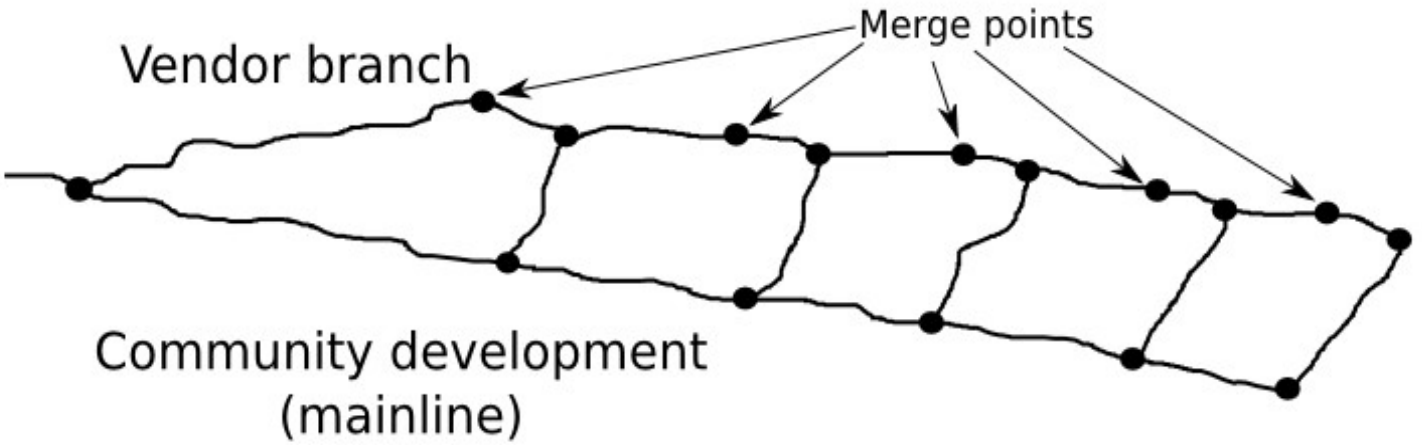
Cost: Work on vendor branch
Opportunity cost of upstream work

Branching strategy



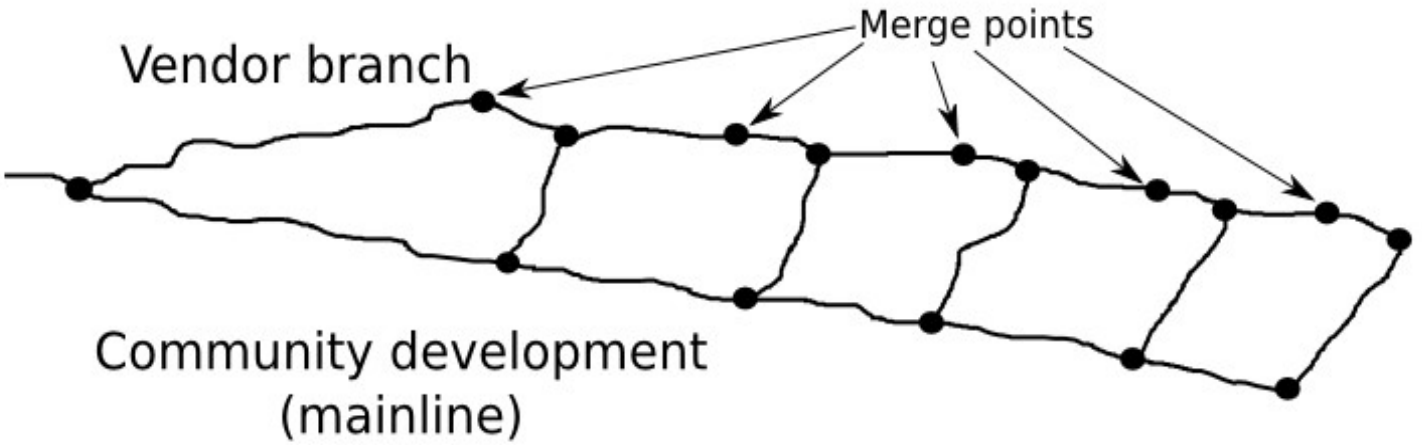
2. Branch and rebase

Branching strategy



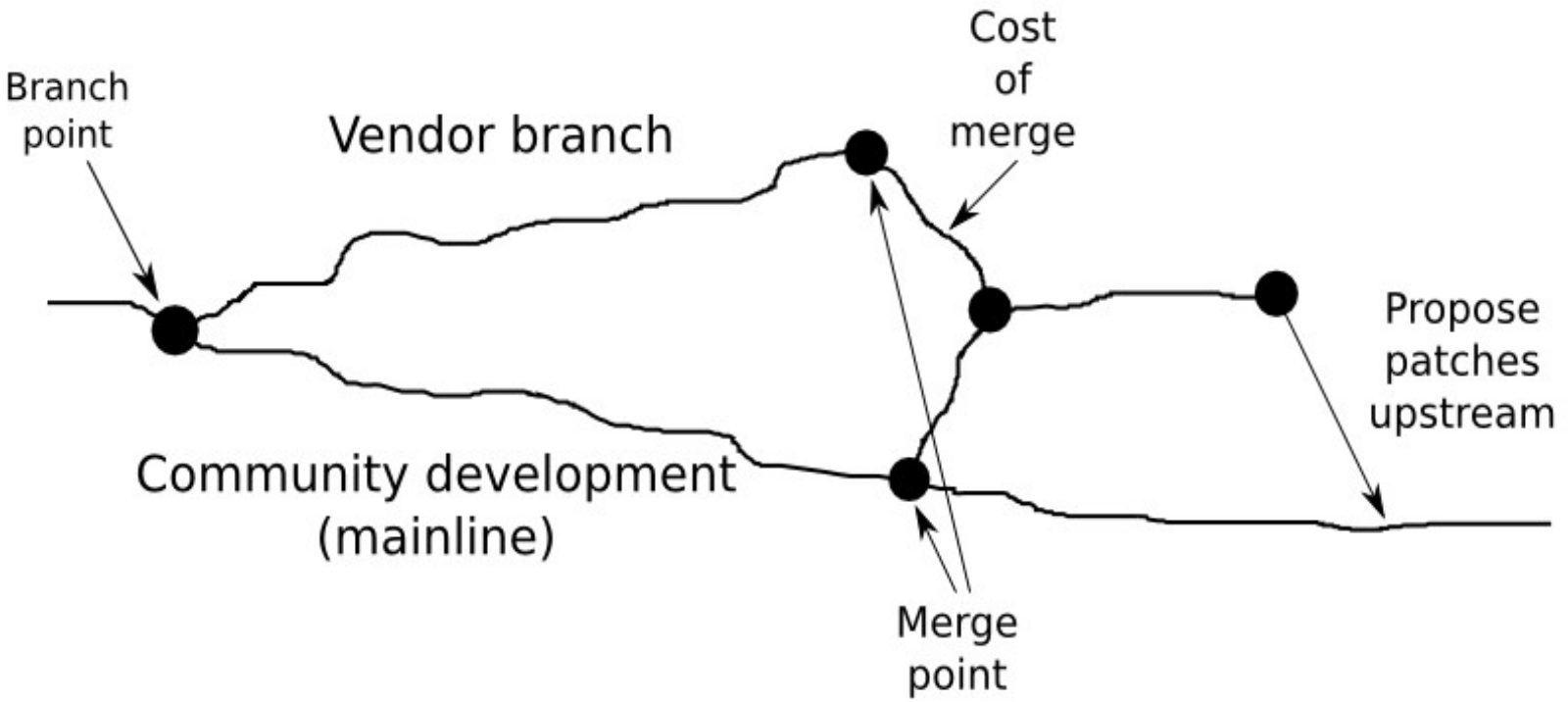
...and rebase and rebase and...

Branching strategy



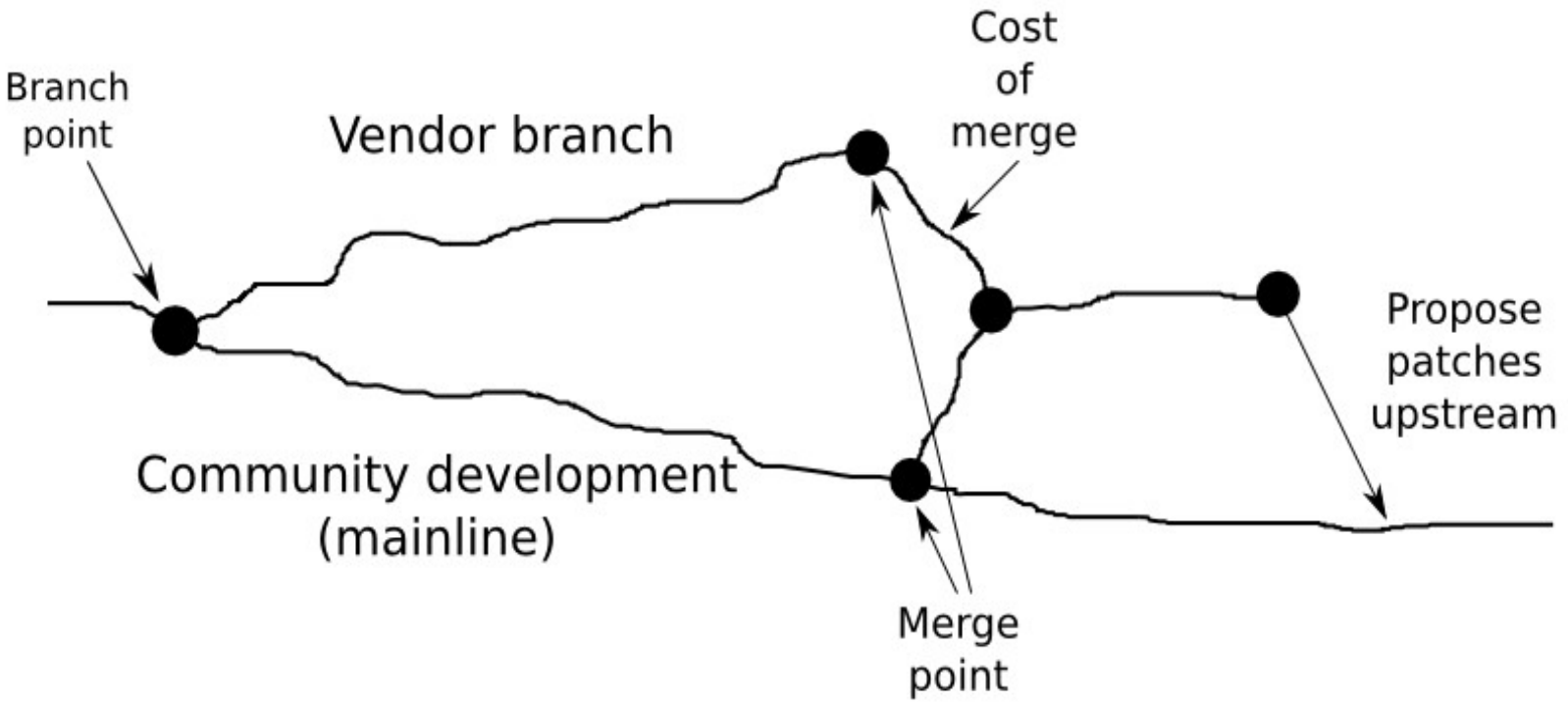
Cost: Vendor work + cost of repeated merging (maintenance)

Branching strategy



3. Branch, rebase, and upstream

Branching strategy

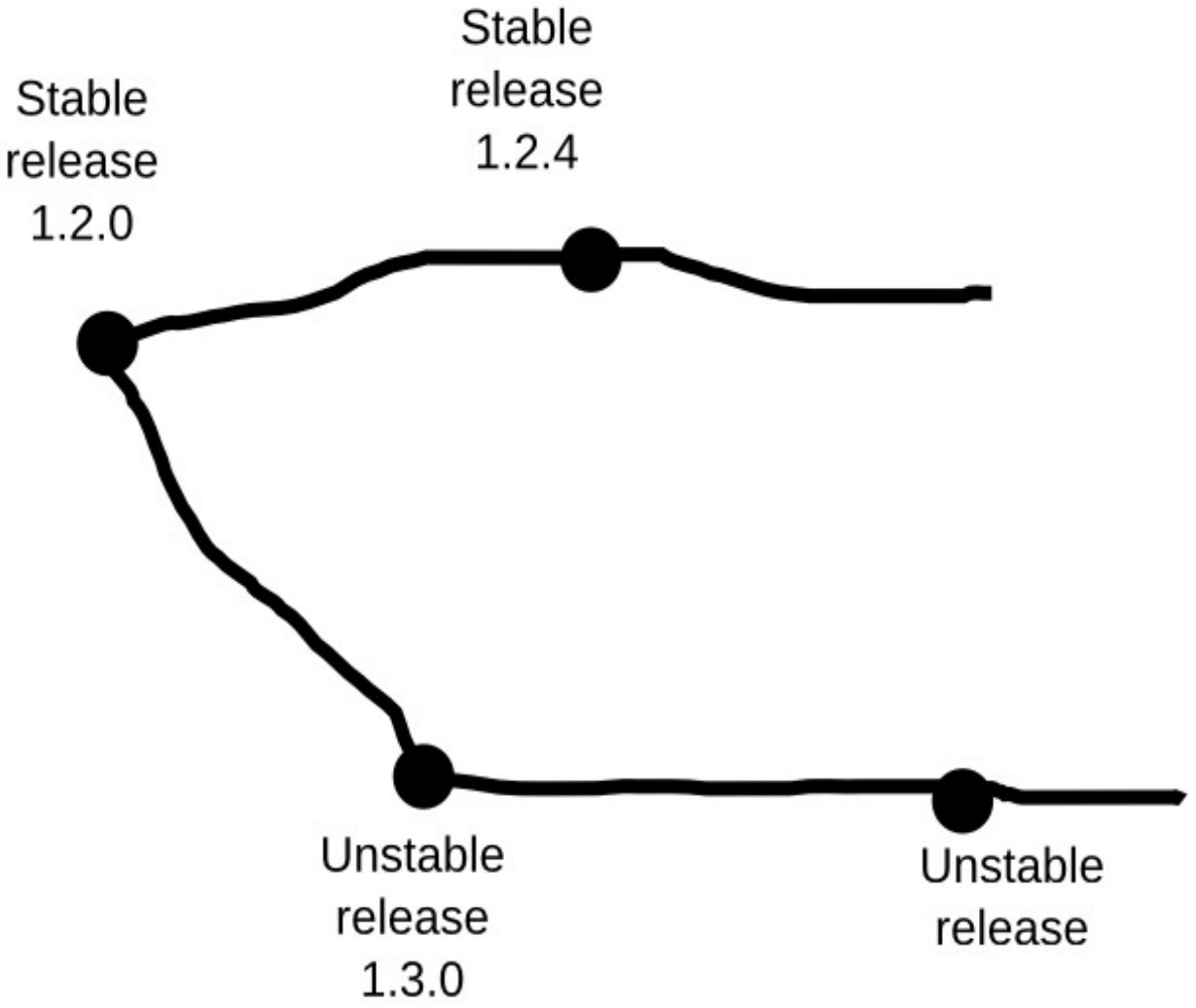


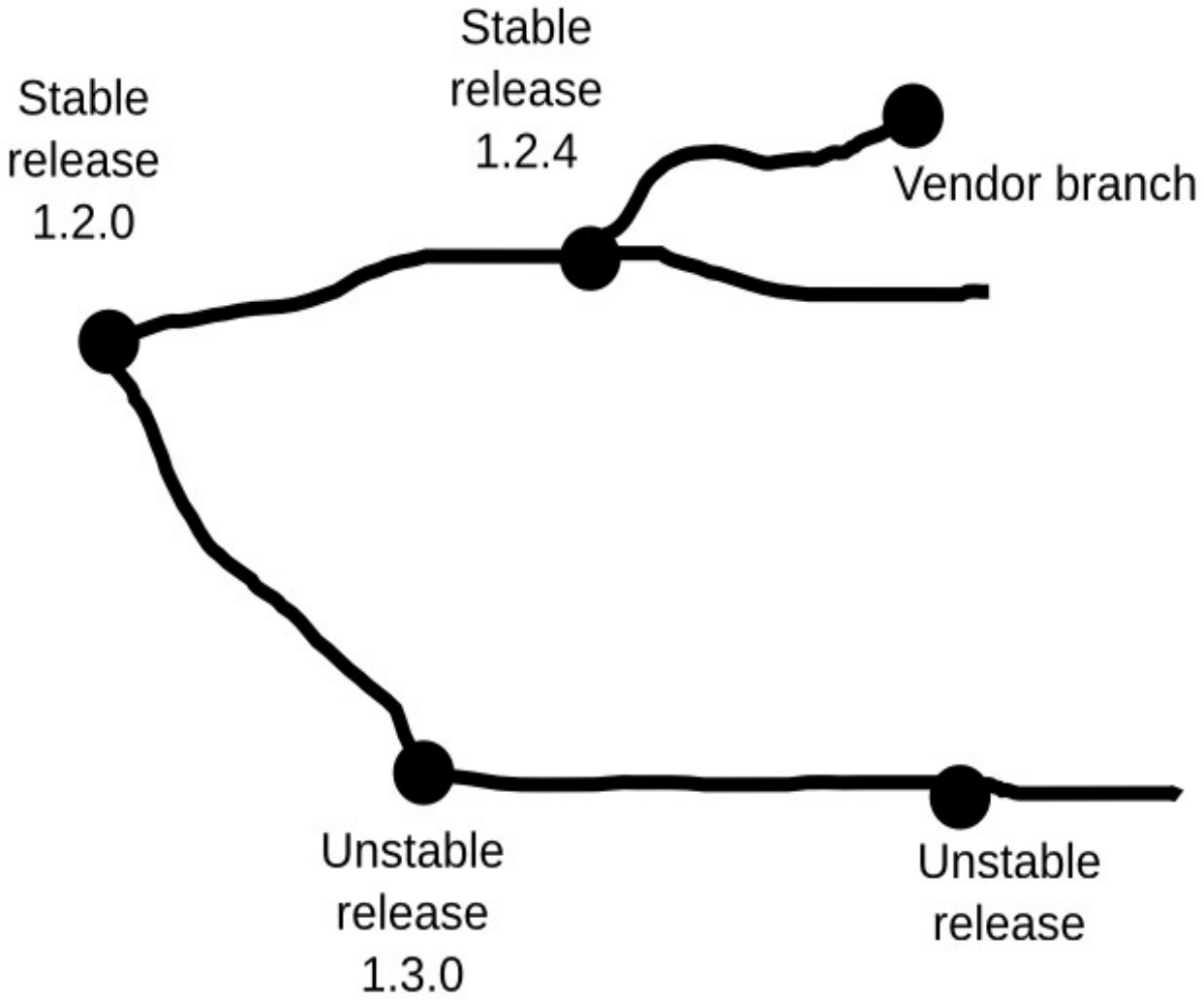
Cost: Vendor work + cost of merge + “community overhead”

“Community overhead”?

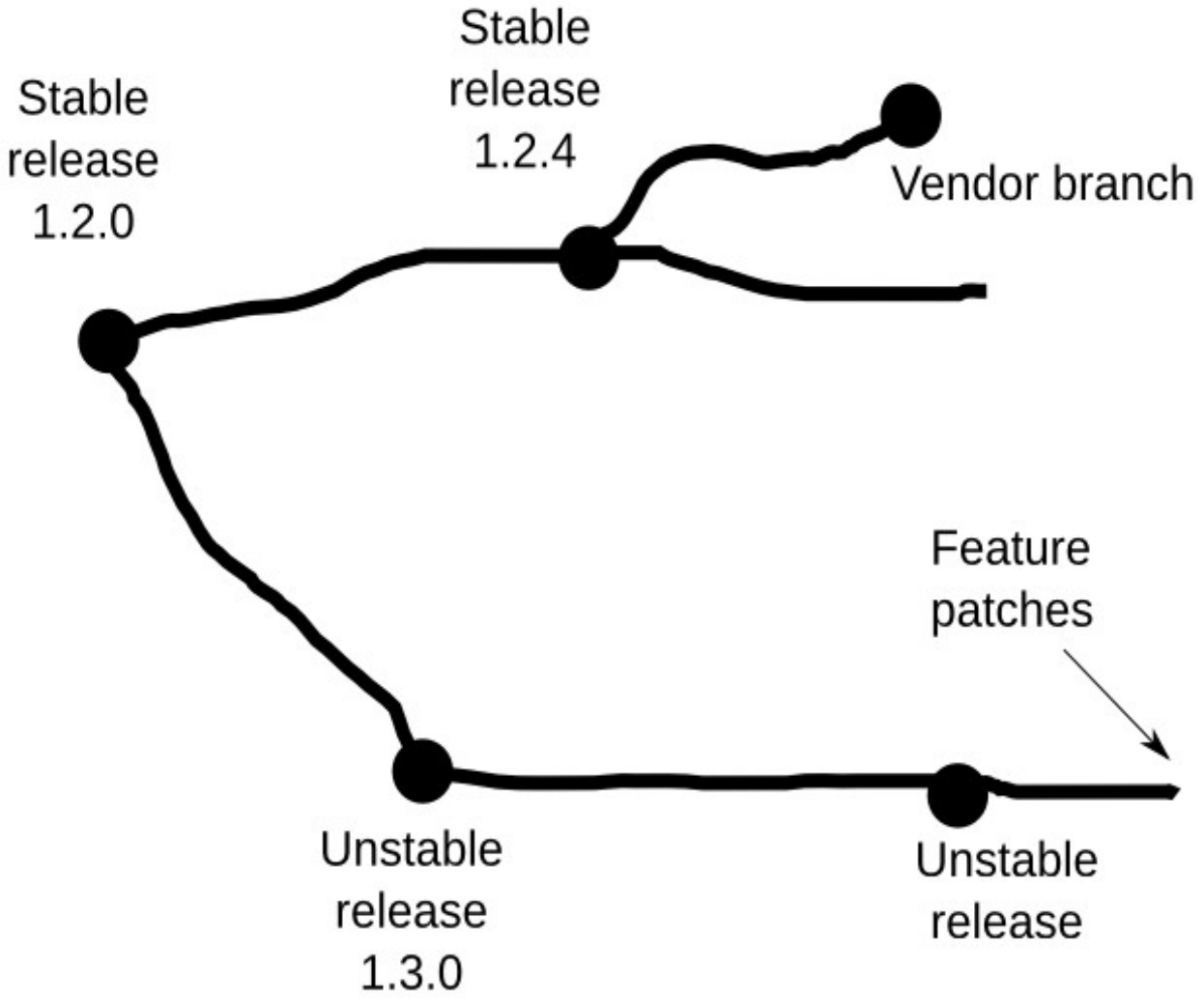
Our communities expect lots of things which companies would not otherwise do

Standard branching





Vendors want to build on a stable base



Community wants patches to HEAD

Choices, choices

Choice 0:

- Ignore upstream
- Provide a black box to your client
- Charge them for support and maintenance

“Scratch your own itch”

Choices, choices

Choice 0 results:

- Waste developer time at upgrade
- No benefit upstream

90% of software works this way

Choices, choices

Choice 1:

- Fork off stable
- Finish project
- Port to unstable & propose patches “when we have time”
- Rebase when new major release comes out

Choices, choices

Choice 1 results:

- Big merges! Pain all round
- “When I have time” never happens
- Typically becomes important at next big version change (too late!)

Act II: Nokia & Google

2003-04: Start working on GTK+
2005: Nokia 770 released

2006: Work with Imendio/Lanedo

- 50K LOC diff
- 4 years work on merge

2011: Igalia wins $50K contract to “make Hildon work on GTK+”

“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

2005: Google acquires Android

2009: Arve Hjønnevåg proposes Wakelocks for inclusion in kernel

Initial proposal rejected, with comments

2009: Several updates also rejected

Debate dies down for a year

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

* >1500 email thread

Suspend blockers 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.

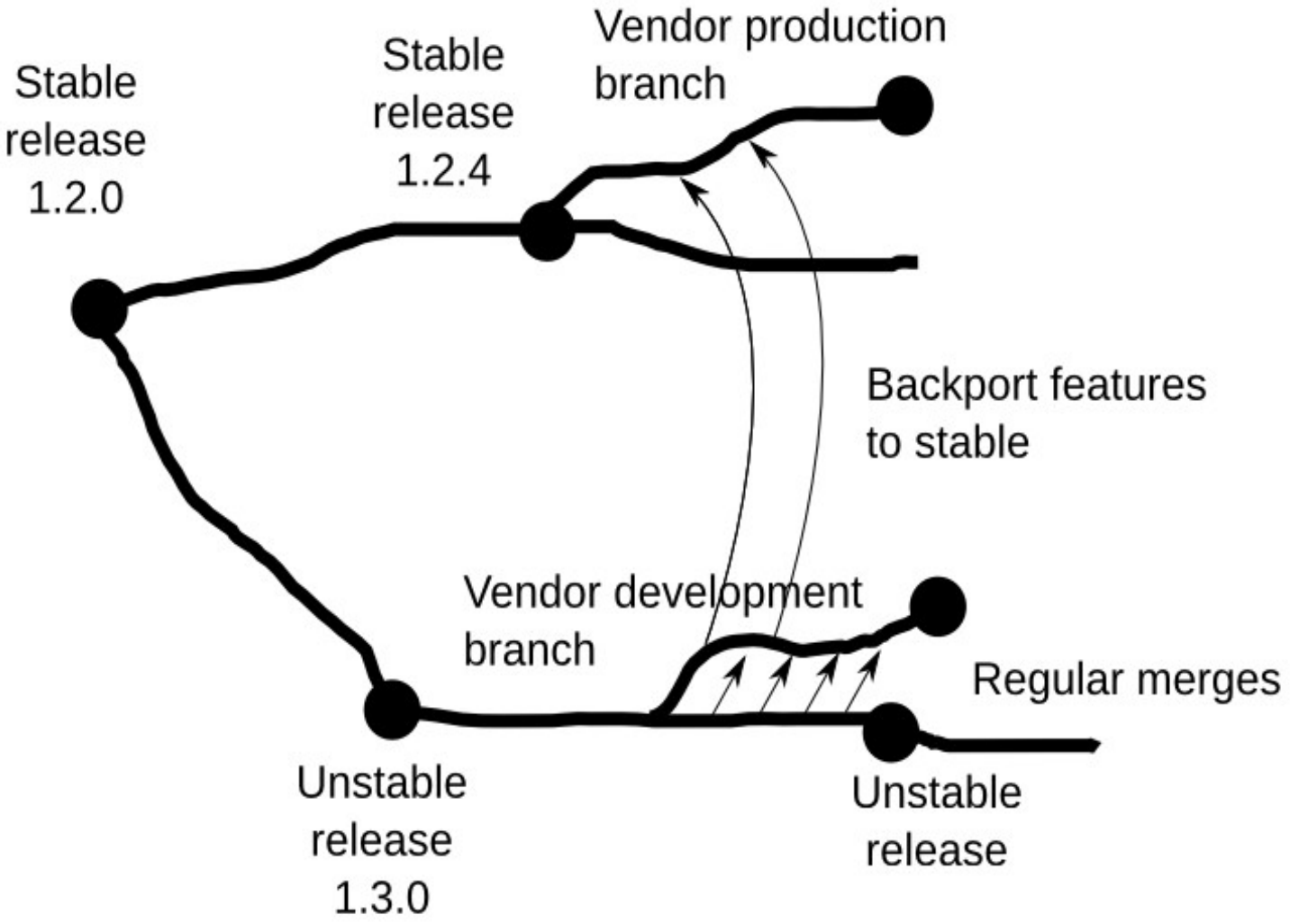
Apr 2010, Paula Rooney, ZDNet

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

Matthew Garrett, LinuxCon 2010

Act III: The Right Way

Ideal situation



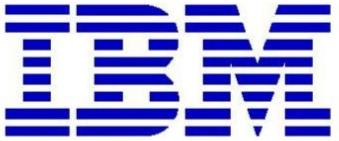
This is what **we** do

Difficulties

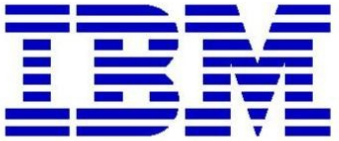
- Hard to sync product & project release dates and features
- Building a castle on quicksand
- Basically 1.5x to 2x the work up-front

Difficulties

Act IV: IBM



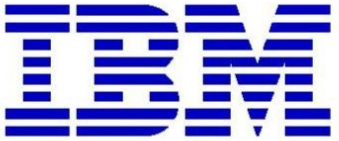
2000-01: IBM starts work on bringing EVMS to Linux, working directly upstream.
May 2001: Public project created on SourceForge, EVMS 0.1 released



Oct 2, 2002: Kevin Corry proposes EVMS for inclusion in 2.5 kernel

“To make this as simple as possible for you, there is a Bitkeeper tree available with the latest EVMS source code, located at: http://evms.bkbits.net/linux-2.5
This tree is sync'd with the linux-2.5 tree on linux.bkbits.net as of about noon today (Oct 2).”

Kevin Corry



Oct 29, 2002: Linus merges LVM2 from Sistina, says nothing about EVMS

“As many of you may know by now, the 2.5 kernel feature freeze has come and gone, and it seems clear that the EVMS kernel driver is not going to be included. With this in mind, we have decided to rework the EVMS user-space administration tools (the Engine) to work with existing drivers currently in the kernel, including (but not necessarily limited to) device mapper and MD.”

Kevin Corry, Nov 5, 2002

Cost to IBM: 6 months, ~18 man-months (est)

When to engage?

Short-term project
Small delta:
Go it alone

Learning community norms and reworking patch outweigh maintenance costs

When to engage?

Long-term project
Small to medium delta:
Think about it

Evaluate rebase cost and lifetime of software
Maybe hire a maintainer/developer to rework and upstream

When to engage?

Long-term project
Large delta:
Engage

Rebasing a large patch sucks. Reduce your delta by upstreaming work
Consider hiring a maintainer on contract to rework and upstream, rather than asking one of your engineers to do it

When to engage?

Strategic project:
Maintain

If your company's future depends on what happens in a piece of software, then hire a maintainer or figure out how to get one of your engineers to be a maintainer.
Expect a maintainer overhead of 20% - 30%

Thank you!
Questions?

Dave Neary
Neary Consulting
dneary@gnome.org

Formula for cost

- Cost of merge ~= (log(dF).log(dP))W + P.W + (1 - De).W

  Where:

  - dF = delta of fork
  - dP = delta of upstream project
  - log(dF).log(dP) normalised to scale of 0 - 1
  - W = cost of initial work to write feature
  - P = difficulty of working with project on a scale of 0 to 1
  - De = developer experience on a scale of 0 to 1

Example

- Developer new to open source spends 6 months writing a major feature for a project with a bad reputation, and proposes it in one go
  - Project needs major re-writing: De=0
  - Difficult project: P=1
  - Major delta makes patch difficult to review: log(dF).log(dP) = 1
- Cost to get code upstream = 3W
  - Reasoning: 1W to try to push existing code, 1W to re-write the code, 1W to re-push re-written code

Example 2

- Community developer with 1y experience in average project, engages community before work for ideas on how to design it, resulting 3 man-months of work needs some reworking
  - P = 0.2
  - log(dF).log(dP) = 0.3 (pretty big merge)
  - De = 0.7 (some chops, but not core team member)
  - Cost of merge = 0.3W + 0.2W + 0.3W = 0.8W

Example 3

- Maintainer develops a new feature – takes 1.3W (due to maintainer overhead)
  - Small deltas, because he's committing regularly to trunk, but participates in peer review: log(dF).log(dP) = 0.1
  - Maintainer sets the culture: P=0
  - Maintainer commits to git: De=1
- Cost to upstream patch: 0.1W