

Mixed License FOSS Projects

Unintended Consequences, Worked Examples, Best Practice

Lars Kurth

Community Manager, Xen Project

Chairman, Xen Project Advisory Board

Director, Open Source, Citrix

  lars_kurth



About Me

Was a contributor to various projects

Worked in parallel computing, tools,
mobile and now virtualization

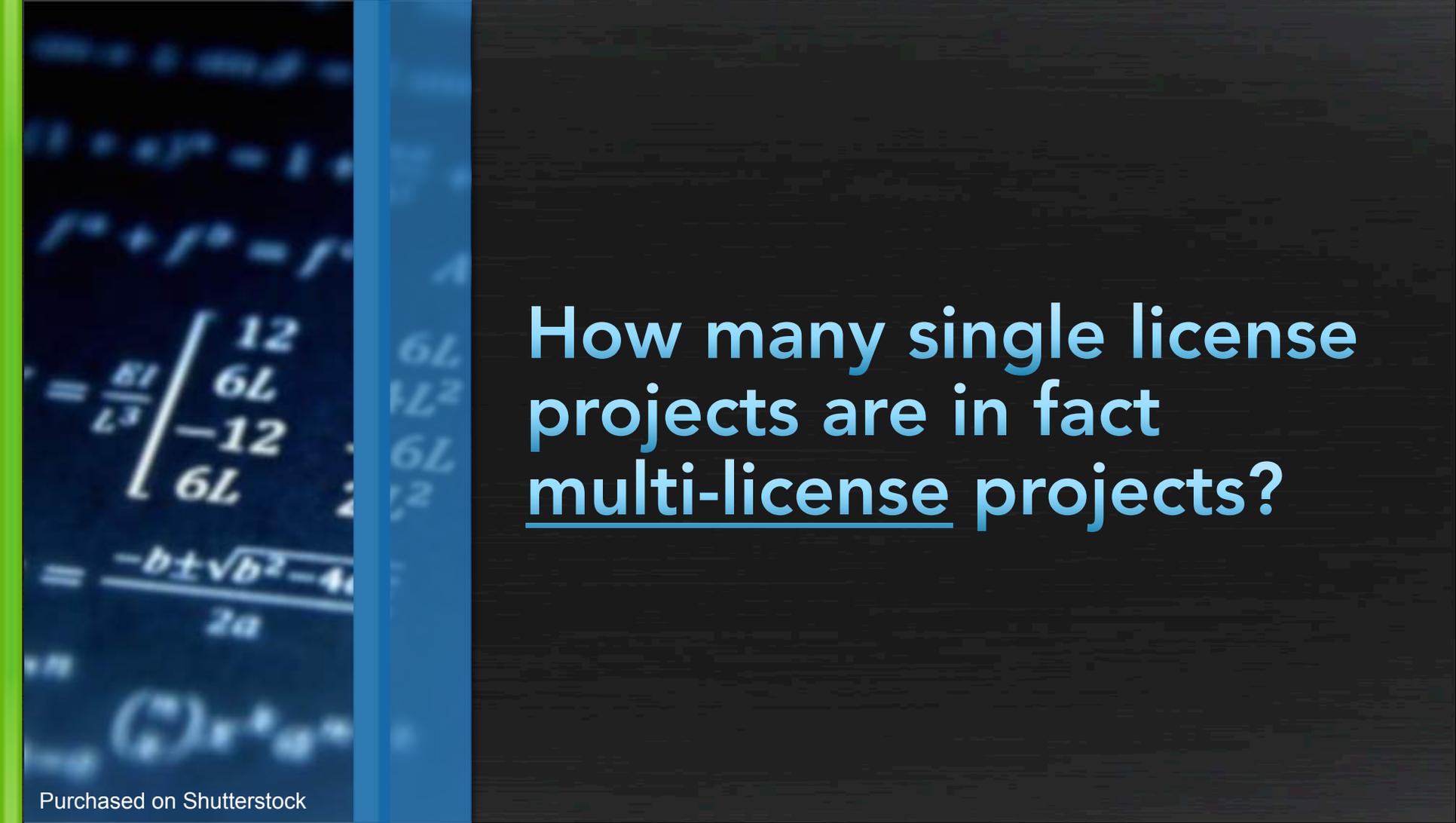
Community guy for the Xen Project

Working for Citrix

Accountable to the Xen Project Community
Chairman of Xen Project Advisory Board

Led and supported a number of licensing
related activities in 2016 (for Citrix and the
Xen Project)



The background features a dark blue gradient with a vertical light blue bar on the left. Faint mathematical formulas are visible, including a determinant calculation:
$$= \frac{\pi l}{L^3} \begin{vmatrix} 12 & 6L & -12 \\ 6L & 12L^2 & 6L \\ 6L & 6L & 12 \end{vmatrix}$$
 and the quadratic formula:
$$= \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

How many single license projects are in fact multi-license projects?



Examples of Projects that started out as single License Projects

Linux: GPLv2

QEMU: GPLv2

Xen Project: GPLv2

FreeBSD: BSD

**What is the percentage of files
in those projects that use the
original license?**



Linux: ≤ 96% GPLv2

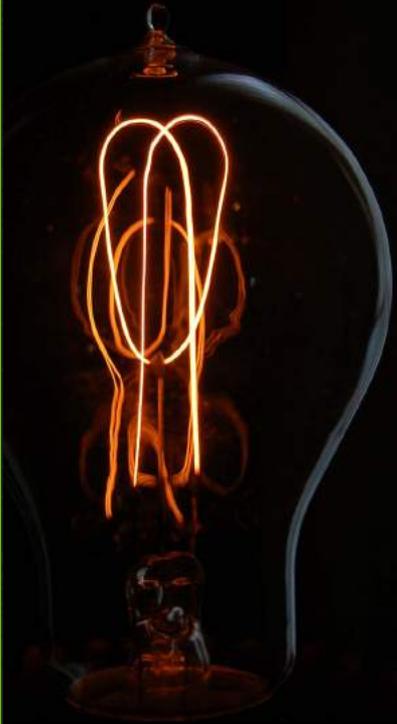
QEMU: ≤ 86% GPLv2

Xen Project: ≤ 98% GPLv2

FreeBSD: ≤ 84% BSD

Data obtained with scancode toolkit 1.6.0 as an approximation
Files with no (c) header classed as “native license”
for the purpose of this approximation

**Many (most?) projects are
not 100% single license**



**Reasons why code with
difference licenses may
end up in your codebase**

You may need to interface with projects of another license

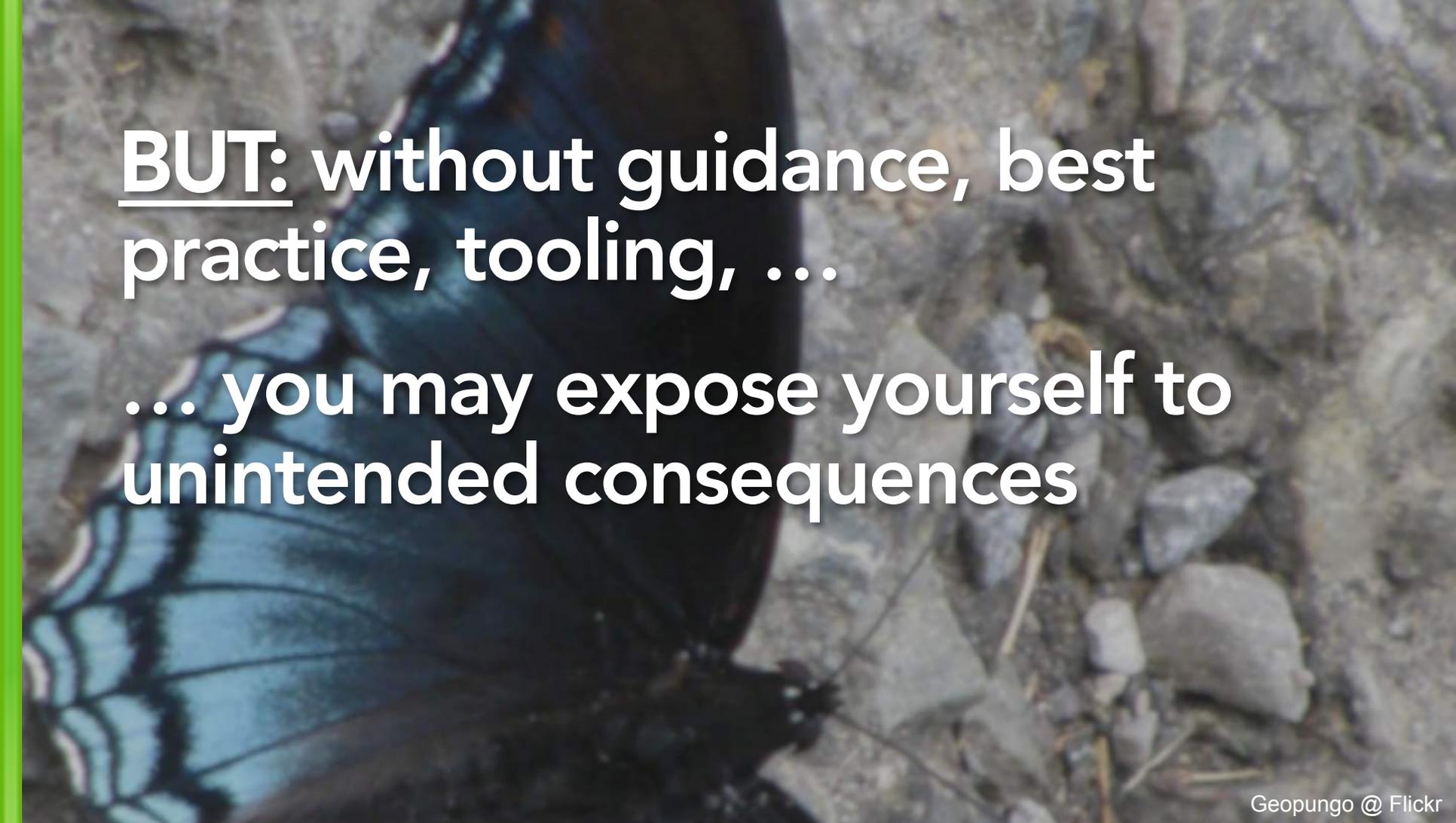
You may want to allow other projects (with another license) to interface with you

You may want to import code from other projects

Your project may not have clear rules that govern license exceptions (→ people assume it's OK to add code with other **compatible** licenses → increasing “entropy”)

...

**These are all good and valid
reasons for license exceptions**

A close-up photograph of a blue butterfly with black markings on its wings, resting on a grey, textured rock surface. The butterfly is positioned on the left side of the frame, with its wings partially spread. The background is a blurred, rocky terrain.

BUT: without guidance, best practice, tooling, ...

... you may expose yourself to unintended consequences



War Stories from the Xen Project

What is the Xen Project?

Developing Open Source
Virtualization Technologies since 2003
> 10M Users

Several sub-projects

Xen Hypervisor, XAPI management tools, Mirage OS, Windows Drivers and Embedded/Automotive Support

Linux Foundation Collaborative Project

Financially sponsored by Alibaba Cloud, Amazon Web Services, AMD, ARM, Cavium, Citrix, Huawei, Intel, Oracle, Qualcomm, Rackspace

Our reasons for GPLv2 license exceptions

Want to enable Guest Support for non-GPL OSes

Most headers are BSD style or MIT licenses

Want to make it possible for such OSes to have Xen support

Some BSD style or MIT licenses

Some code is dually licensed (enable re-use elsewhere)

Want to enable non-GPL tools to interface with Xen

Key tools libraries are LGPL 2.1+

Want to be able to import code from other projects

We had **no codified rules** about licensing exceptions

We assumed we are a single license project



War Story 1:

The perils of license related information that is not easy to consume by lawyers



Late 2015: a large vendor (codename Dragonblood) is reviewing the Xen Project with a view of allowing their staff to contribute

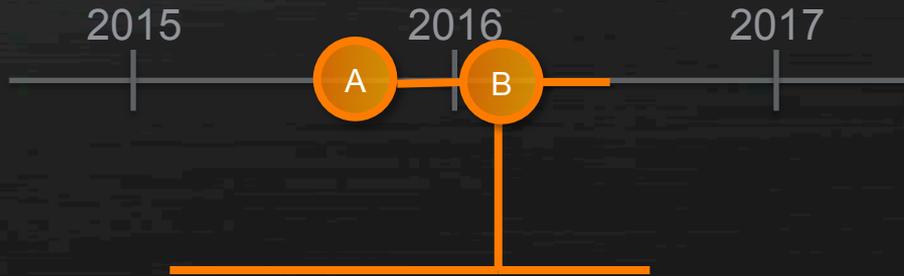


Dragonblood company starts IP and patent review

Note: the IP lawyer is very thorough

Evaluates license, COPYING files, runs FOSSology, ...

- Picks up a number of mismatches between COPYING file and reality (e.g. the COPYING file stated that headers are BSD, but some were MIT)
- Lots of questions about the rationale for licensing exceptions (unfortunately this was not always easy to find out)



Dragonblood company **won't allow staff to contribute until all questions were resolved**

Ended up doing lots of code archaeology to answer questions and secure future Dragonblood contributions

- Reason for why a license exception existed
- Rationale for why a piece of code was imported and where it came from



Dragonblood company **allows staff contributions**

Why did this take so long?

Needed information was present,
but not readily consumable

Information was in commit logs, sometimes in
source files, sometimes in COPYING files,
sometimes in mailing list conversations referred to
from elsewhere

Inconsistencies

Which confused the IP lawyer and didn't build trust

Lawyers tend to work on multiple projects

Elapsed time periods with no activity



What did we do?

In-tree information on license exceptions

Guidance on license exceptions:

- When do we use what license
- Rationale for specific and classes of exceptions

COPYING file for each non-GPLv2 component

README.source files (one per directory)

For code imports (even for GPLv2 imports) tracking:
rationale, source, and other relevant information

Fixed inconsistencies in documentation

A few things we merely documented

E.g. some imported code had inconsistent
licenses (license headline said MIT, text was BSD)





War Story 2:

Relicensing a key component :
a worked example with
complications

Patch Series:

Make ACPI builder available to components other than hvmloder

Enabling a major new piece of functionality (PVHv2)

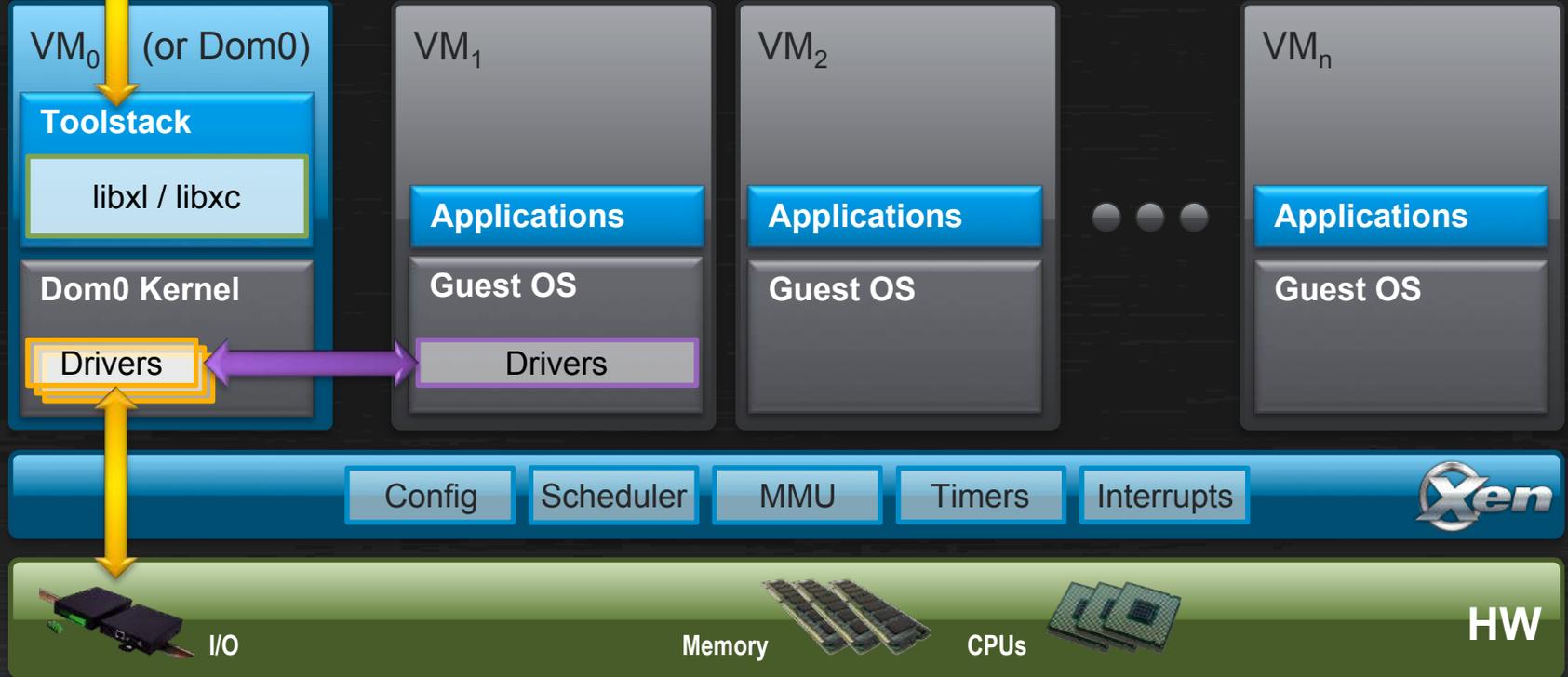
A bit of taxonomy and terminology



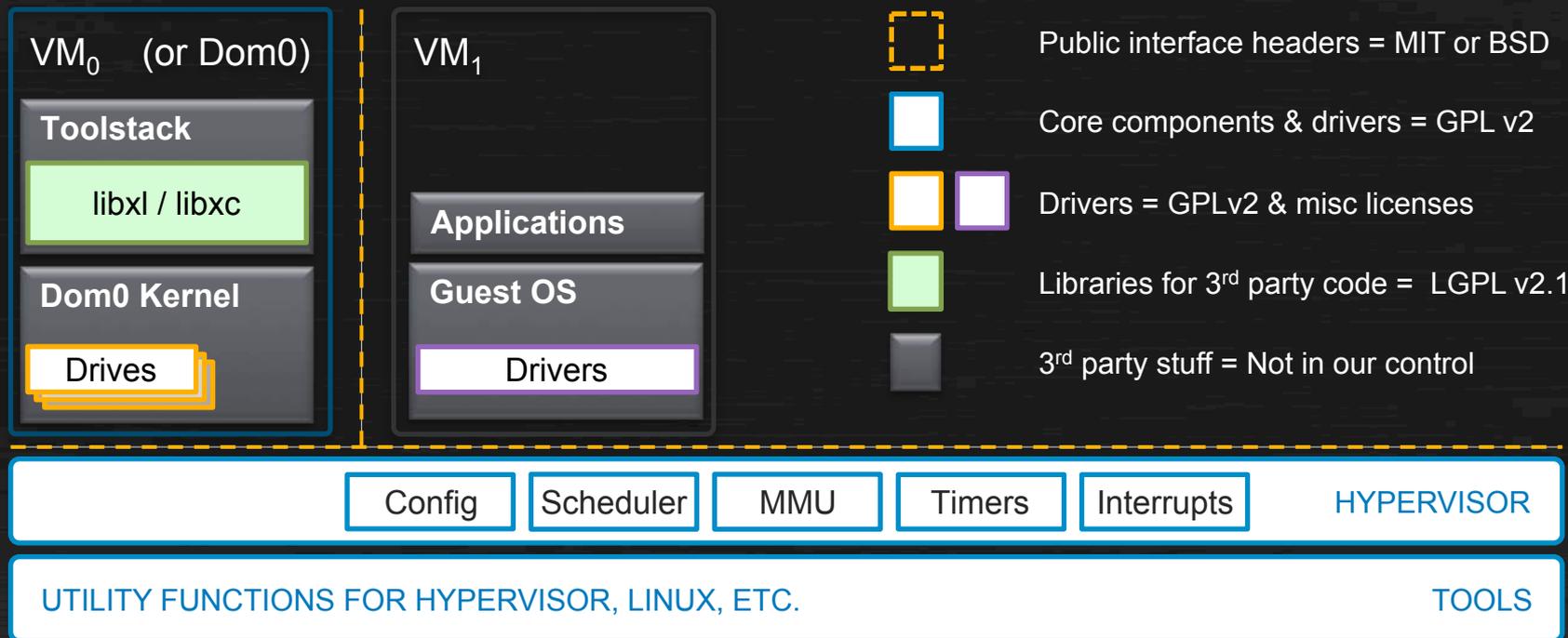
Picture by Lars Kurth
Catasetum maculatum in Costa Rica



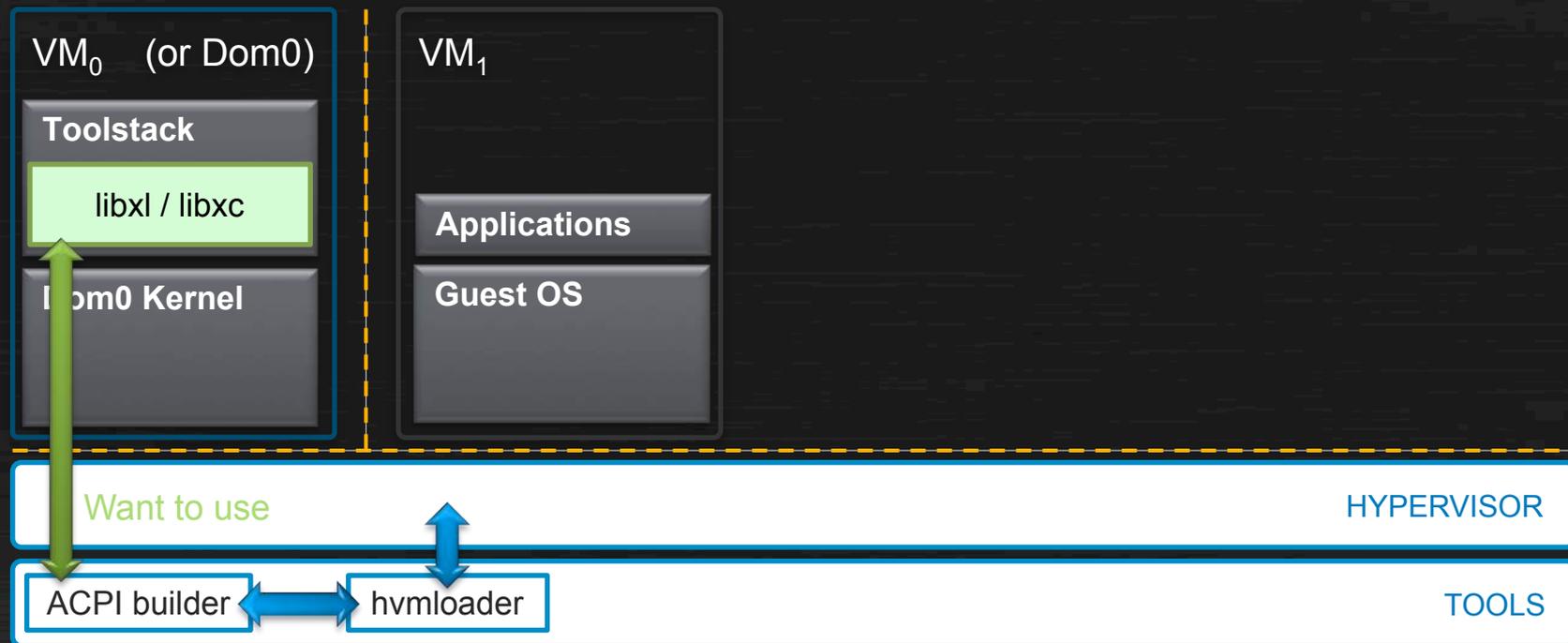
Console



The licensing view of the previous diagram



Zoom: ACPI Builder change



Our Options?

Do a clean-room re-implementation

Too hard

Allow GPLv2 encumberment of
libxl / libxc and its consumers

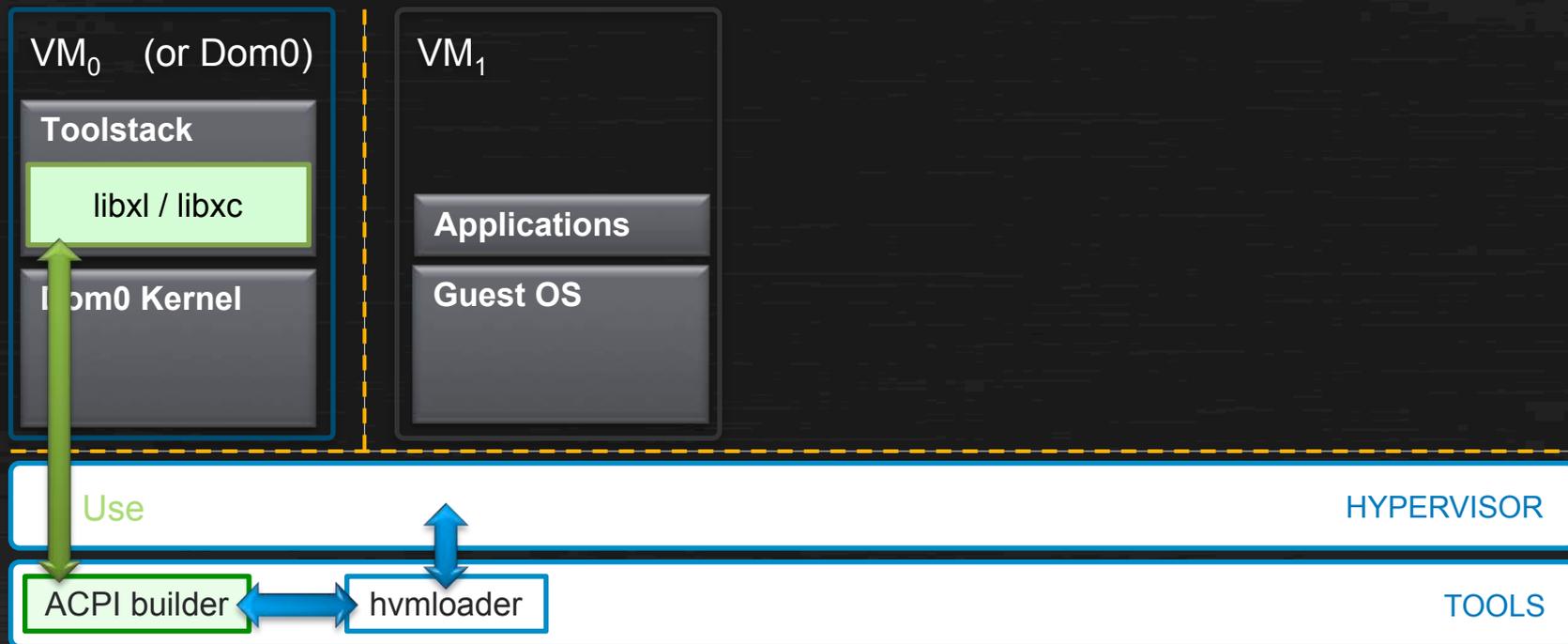
Too disruptive

Relicense

Seemed relatively straightforward



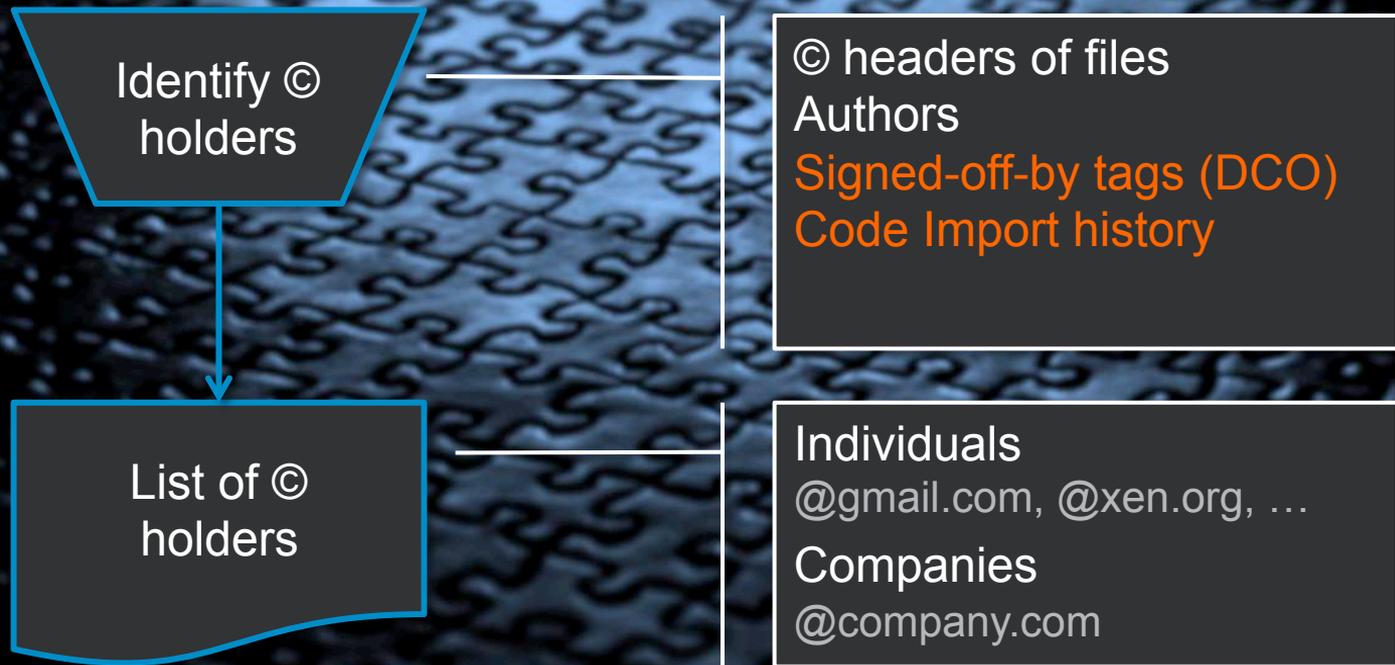
Goal: Relicense ACPI Builder to LGPL v2.1



Observation

Refactoring and new feature development may require *unanticipated* license changes

Could have been avoided with more foresight



Identifying © holders: Easy, right?

Tooling: Hg to Git conversion, code motions, ...

Can lead to an incomplete list of © holders due to tooling issues

Was the code (or some of it) imported from elsewhere?

You may want to run FOSSology or similar

If yes, there may be more © holders

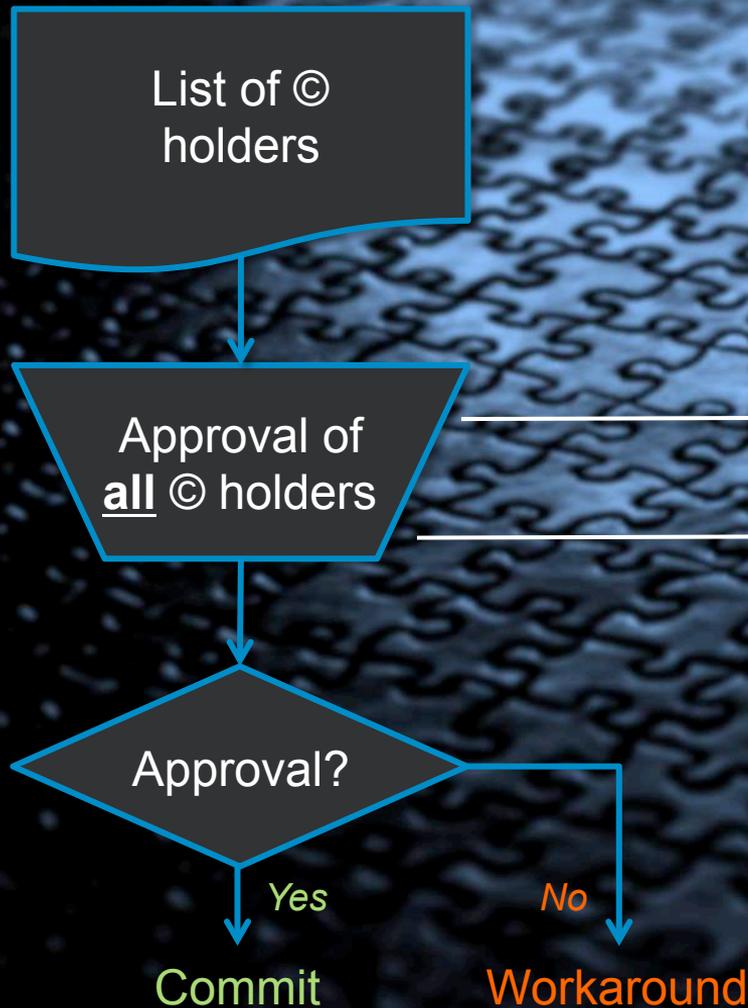
In our case, the code was imported from Linux

There could potentially issues with CLA's (if parent project has CLA's)

Use of private email addresses by company employees

If yes, you probably have to ask both

Chasing individuals can be harder than chasing companies



Individuals
Contact by e-mail

Companies
Find company stake-holders that can make a decision
In our case: most companies were also Advisory Board members

Chasing and follow-up
By LinkedIn, phone, etc.
Sometimes e-mail addresses change

Companies
If you don't have an up-to-date contact you will have a challenge

Mid 2016: contributor XYZ (working for vendor codename Dendrobium) could not be tracked down and approval could not be obtained

Find new
contact details
of XYZ

Failed (tried LinkedIn, etc.)
Because XYZ is based in China

Enlist help
from Chinese
companies

Failed
Tracked down individual, but that did not help. The individual changed company and the team in *Dendrobium* did not exist any more.

Searched for
Dendrobium FOSS
contributors

Failed
There were none; not a FOSS company

Searched for
Dendrobium FOSS
staff

Succeeded via LinkedIn
Worked on a contingency plan
Eventually got approval (took 2 months)

Contingency Plan

Made use of the fact that **binaries**, not source code, are licensed
And that not all functionality was needed in the LGPL v2.1 library

Could not remove the change by Dendrobium

- Too far in the past and a key piece of functionality*
- Too complex for “fair use” clause*

Build two variants of ACPI Builder library from the same codebase

GPLv2 and LGPL v2.1 variants

- Keep GPLv2 code clearly separated in the source tree*
- Not ideal from an engineering perspective*

BUT: ugly, not easily maintainable, hack

Pain Points

Tooling

Pre-Git code motions (delete, create)

Documentation

No README.source file for import from Linux

Nearly missed code import

Sign-Off's on Company time

In the early days of the project many people signed off DCO using private e-mail addresses or Xen alias

Approval

Getting approval from all stake-holders

Implemented a backup (**ultimately not needed**)



War Story 3:

The unintended consequences of mixing *GPL / LGPL version X only* code with *GPL / LGPL version X or later* code



Beginning 2016: vendor (codename Dragonblood) was rather sensitive towards patents and GPL v3



Code marked as GPL v2+ could be copied into a GPLv3 project.

GPLv3 projects are problematic for us from a patent protection perspective.

Thus, we may not be able to contribute to your project.



*Dragonblood company IP lawyer (paraphrased).
Does not reflect the views of the Xen Project*

Why did we have GPLv2+ code in our codebase? Was this a conscious decision?



No: Purely accidental, because some contributors copied license text from FSF (or elsewhere) without specifying the GPL version.

Is this issue specific to the
Xen Project?

% of GPLv2 or later (relative to GPLv2 code) ...

Linux: 14%

QEMU: 9%

Xen Project: 10%

FreeBSD: 32% ¹⁾

Data obtained with scancode toolkit 1.6.0 as an approximation

¹⁾Total is 7% GPL code of which 34% are GPLv2+, 84% BSD



What did we do?

Could we fix this?

Couldn't find clear guidance and a precedence

Too much work/disruption and potentially divisive

CONTRIBUTING file

Added common © header templates

In particular for GPL v2 and LGPL v2.1

Raised awareness amongst committers

Issue went away

When I pointed out that other projects Dragonblood company contributes to, have the same issue

BUT: it is possible that Dragonblood company instructed their staff not to contribute to GPLv2+ files



Picture by Lars Kurth

A bigger issue!

If you are an L/GPL vX only project
L/GPL vX or later files in your codebase

→ could scare away some contributors

If you are an L/GPL vX or later project
L/GPLvX only files in your codebase

→ diminish your capability to upgrade to vX+1
in the future

Have mechanisms in place to avoid
a mixture of L/GPL vX only and or later

→ worst of both worlds



An aerial photograph of a tropical beach. The top half shows a clear blue sky. Below it is a wide, white sandy beach. The ocean is a vibrant turquoise color near the shore, transitioning to a deeper blue further out. There are some shallow reefs or sandbars visible in the water.

Summary of Best Practices

Picture by Lars Kurth

If you want to stay single license

Need tooling to enforce

L/GPL vX only vs. L/GPL vX or later

Have some mechanisms in place to avoid a mixture

Document License Exceptions

Rules/conventions, rationale, instances

Provide © template headers

README.source files or similar

For all code imports (even for “native” imports)

Company / Personal Sign-off

Document conventions (@xenproject.org, @kernel.org)

Awareness by committers

Plan for the future

Consider licensing carefully for any code that may be re-usable





Questions