Delayed Open Source Publication: A Survey of Historical and Current Practices

Seth Schoen, James Vasile, Karl Fogel

Foreword

This research offers a number of important findings, including that delaying the release of freedoms to software users dates back to the GNU project and continues today.

Companies have experimented with various business models to keep exclusive rights for a limited duration. This research suggests that this strategy has been employed as a way to control and monetize software before eventually transitioning to an OSI Approved License®.

Richard Stallman's acceptance of a compromise, where he used an older version of Aladdin Ghostscript for the GNU system to ensure full freedoms, underscores the complexities and compromises since the early days of the Open Source movement.

Interestingly, the pioneers of the delayed Open Source approach have since shifted tactics, offering their software to users with two licensing options at the same time: Strong copyleft or a different contractual agreement. The delayed Open Source practice by companies in the cloud and software-as-a-service (SaaS) space has rekindled this debate, ignoring the historic examples.

This survey is a starting point, raising new questions starting from why is the AGPLv3 the only license trying to apply copyleft to the SaaS model. The OSI is grateful for Sentry's donation that allowed us to shed new light on this business practice. We look forward to dedicating more research in support of companies doing business with Open Source.

Stefano Maffulli
Executive Director - Open Source Initiative

Table of Contents

Version 1.0 published January 2, 2024

@ 2023 Open Source Initiative
This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International license.
(CC-BY-SA, https://creativecommons.org/licenses/by-sa/4.0/)

Authors: Seth Schoen, James Vasile, Karl Fogel
Designer: Caroline Henriksen

SENTRY
This research was made possible thanks to a donation from Sentry and OSI's individual members.

1 Executive Summary

Delayed Open Source Publication (DOSP) is the practice of distributing or publicly deploying software under a proprietary license at first, then subsequently and in a planned fashion publishing that software's source code under an Open Source[1] license.[2]

Software producers have practiced DOSP throughout the history of Open Source software. This document is a selective survey of that history. It collects and categorizes some examples and tries to identify patterns and trends.

Based on the samples we know of, we categorize DOSP into three high-level types:

- **Unconditional scheduled relicensing.**
  Planned OSS releases with just a pre-defined time delay. See Section 3.
- **Event-driven relicensing.**
  OSS publication happens regularly, but is driven each time by some expected event, e.g., the publication of the latest proprietary version, which prompts the previous version to now be open sourced. Forms of this seem to have been used — albeit loosely in some cases — in the early history of DOSP (see Section 2) but it appears to be much less common now, with time-based scheduled relicensing being favored instead.
- **Conditional relicensing.**
  "We'll publish this as Open Source as soon as we get funding" or "as soon as we find the right non-profit home for it", etc. This can include bounty mechanisms, but only if they were planned — that is, it does not include "buy-outs".

  This type is probably the weakest match for our working definition of DOSP, though it is technically a match. Unsurprisingly, stated intentions to release under Open Source license do not always result in that actually happening. Still, when it does happen, it is an instance of DOSP.

We saw two trends that seem significant:

- **The rise of the Business Source License (BUSL).**
  Use of BUSL is growing rapidly. See Section 3.3.
- **Anti-competition terms are becoming more common.**
  Traditional DOSP was typically about monopolizing direct commercial revenue: the license would grant most of the permissions necessary for Open Source but, crucially, withold permission to use the software commercially[3] — a restriction that would apply to all downstream licensees, i.e., to users, not merely to developers.

  More recently, however, some DOSP licenses are about preventing any licensee from using the software in a product or service that competes with certain specific types of software that are strategically important to the licensor, independently of direct revenue. See Section 3.3.1.

EXECUTIVE SUMMARY

We emphasize that this document is at best an initial survey and a first-pass analysis. It uncovers various interesting questions that we must leave for future research. We list some of these in Section 7; among the most important are:

- Why do organizations so often choose a non-Open Source license (such as the BUSL) and a DOSP release arrangement when simply publishing under the AGPL[4] from the start might, in many cases, meet their goals well enough?
- When do BUSL-licensed projects have different contribution dynamics than truly Open Source projects, and when (if ever) do they have similar dynamics?
- When a previously Open Source project is converted to DOSP by its licensor, under what circumstances does this tend to cause a viable fork to occur?

Just as Open Source gradually shook out into a handful of licenses that are used by the vast majority of projects, we might now be seeing a convergence toward a recognizable and relatively small set of DOSP licenses. It is too soon to know for sure if the current options will settle in as the standard. The list of most-used Open Source licenses has been quite stable for over a decade now, and there is little reason to think it will change any time soon. With DOSP licenses, though, it is possible we are still in a period of experimentation. Today's handful of commonly-used licenses may just be a precursor to tomorrow's recognized standard.

NOTES

[1]We use the term "Open Source" throughout for compatibility with the Open Source Initiative's style guide, as the OSI supported the production of this report. We mean by that term the same thing that people also use the terms "free software" or "free and open source software" to refer to. While we could use "free software" interchangeably with "Open Source" — that too would be compatible with OSI's style guide — for the sake of consistency we have chosen to just use "Open Source" throughout.

[2]Note that this definition of DOSP deliberately does not include ad hoc or improvisatory Open Source releases of formerly proprietary code. For example, the 1998 release of the Netscape Navigator source code, which through further development eventually became Mozilla Firefox, is not an example of DOSP. This report examines the history and effects of DOSP practiced as a conscious strategy; the effect of unplanned or unpredicted Open Source publication is also an interesting topic, but a separate one.

[3]This causes the license to fail clause 6, "No Discrimination Against Fields of Endeavor", in the Open Source Definition (see https://opensource.org/definition-annotated/).

[4]The Affero General Public License — an Open Source license specifically designed to ensure freedom from monopoly in network-based application service provision as well as in traditional file-based distribution. See https://en.wikipedia.org/wiki/Affero_General_Public_License.

2 Early History

The earliest notable use of DOSP we found is Aladdin GhostScript, which was a relicensing (by its original author) around 1998 of the originally GPL-licensed GhostScript project under the "Aladdin Free Public License". Aladdin's practice was to publish all new versions of the software under this license, which did not permit commercial redistribution. Aladdin also published versions of its software under GPL once they were older than about a year, initially as "GNU Ghostscript" and later as "GPL Ghostscript".[5]

GhostScript's author, L. Peter Deutsch, described this practice as providing commercial exclusivity that would help fund continued development of the project.[6] This is a commonly cited motivation for adopting DOSP.

Interestingly, GhostScript's makers eventually dropped the delay in favor of straight-up proprietary-relicensing.[7] With this approach, they simultaneously release GhostScript under both a proprietary license and GPL.[8] They continue to use this model today, though they have since replaced GPL with AGPL.[9] They determined that their market of commercial, embedded developers were paying to avoid the GPL and AGPL, and that the time-delay did not significantly change these companies' incentives to pay for licenses.[10]

Another early example of DOSP is KDE's Qt library, which committed to a form of DOSP as a minimum guarantee. KDE is a desktop environment built using the Qt GUI library.

Over the years, the company that produces Qt, Trolltech, has experimented with a variety of public collaboration approaches that includes a mix of Open Source and non-Open Source approaches.[11]

When KDE adopted Qt as its GUI toolkit, "lock-in" concerns about reliance on a codebase owned by a commercial company led to a series of agreements between a KDE nonprofit and Trolltech. The original license allowed the KDE Free Qt Foundation to release a version of Qt under BSD license if Trolltech substantially stopped Qt development for more than a year.[12] Moreover, a series of contracts between KDE's nonprofit and successive Qt copyright holders include commitments to release Qt versions under specific Open Source license terms "within a timeframe of not more than 12 months" relative to any proprietary release.[13]

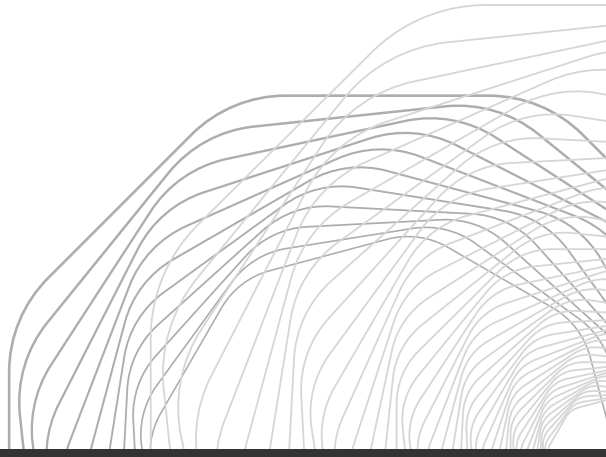
The Qt licensors did maintain separate "Qt Commercial Edition" and "Qt Open Source Edition" releases for some time; the latter complied with the licensors' commitments under the agreements. We haven't identified evidence of a significant gap in time or functionality between these releases, although such gaps may have existed. The agreements established minimal standards for the protection of KDE, but Qt's various copyright holders appear to have generally exceeded those standards in any case. DOSP ended up being a fall-back scenario for two different conditions that didn't arise in practice (unreasonably delayed Open Source releases, or complete discontinuation of upstream development). It appears that Qt licensors usually understood their commercial strategy as akin to more conventional proprietary relicensing, where proprietary adopters would pay for the Commercial Edition in order not to incur copyleft obligations. Making generalizations about this strategy is complicated, as several different commercial entities acquired Qt over time and may have had somewhat different understandings.

EARLY HISTORY

Today, all of Qt is released simultaneously under LGPL/GPL and proprietary licenses.[14]

GhostScript and Qt are the two earliest projects we found making documented use of DOSP. They used them in different ways, but both related in a broad sense to attempts to protect a licensor's commercial interests. As we will see from later projects, this is the most common use of DOSP. However, neither of these projects actively practices DOSP today, although both are still under active development.

A more recent example is the Onivim 2 project, which had a proprietary license and also maintained an 18-month-delayed MIT-licensed version.[15]



NOTES

[5]See, for example, https://web.archive.org/web/20070816214332/http://pages.cs.wisc.edu/~ghost/doc/AFPL/6.01/New-user.htm#Overview.

[6]Specifically, he wrote in the AFPL that this mechanism

> attempts to ensure that those who receive, redistribute, and contribute to the licensed Program according to the Open Source and Free Software philosophies have the right to do so, while retaining for the developer(s) of the Program the power to make those who use the Program to enhance the value of commercial products pay for the privilege of doing so.

Larry Rosen also told us, based on his communications with Deutsch, that

> Deutsch's expressed preference for [initial publication under] the AFPL over the GPL arose from what he saw as a serious "free rider" issue for commercial distribution, initially motivated by fax software vendors distributing Ghostscript executables with their products and invoking them as black boxes through the equivalent of 'exec', which the GPL allows without bringing the entire product under the GPL.

NOTES

[7]This practice is sometimes also called "dual-licensing". That term can be ambiguous, however, having historically also referred to releasing Open Source software under two or more Open Source licenses simultaneously. Bradley Kuhn pointed this out long ago to one of the authors (Karl Fogel) and suggested the more accurate term "proprietary relicensing"; we thank him again for it.

[8]This change was made in 2006. See https://web.archive.org/web/20161003082642/http:// ghostscript.com/News.html.

[9]See https://ghostscript.com/licensing/index.html.

[10]Although these events took place after Deutsch had sold Artifex, Deutsch told Rosen that

> Artifex eventually abandoned the AFPL / GPL division, I believe because they found that it was a bit of complexity that didn't affect their revenue from commercial licensing. Instead, they simply offered the choice of either GPL or a straight commercial license. In addition, I believe they offered performance-enhancing replacements for certain modules that were only available to commercial licensees. (The ones I remember hearing about were things like halftoning or shading code that used processor-specific SIMD capabilities.) At the same time, they put quite a bit of energy into identifying and taking legal action against commercial users who were violating the GPL, of which there were an astoundingly large number. For the last several years this actually resulted in substantial revenue, from retroactive commercial license payments and from new commercial license agreements: some offenders started complying with the GPL, some obtained commercial licenses, and some stopped using the code altogether.

[11]Some of them might be called "visible source" or "source available": the source code was, as far as we can tell, always available, just not always with all the freedoms guaranteed by the Open Source Definition.

[12]See https://kde.org/community/whatiskde/kdefreeqtfoundation.

[13]See id., which includes the exact language of the licensors' contractual commitments; a portion of the historical context is also described in https://tinf2.vub.ac.be/~dvermeir/manual/KDE20Development-html/ch19lev1sec4.html.

[14]The Qt Group states that there is currently one exception where it doesn't have the right to grant a proprietary license for a specific module, the Qt WebEngine, which is only available under LGPL v2.1. See https://www.qt.io/download-open-source.

[15]See https://github.com/onivim/oni2/issues/3771, and see also https://web.archive.org/web/20210929101337/https://github.com/onivim/oni2-mit. All development on the project ceased in 2021, and it was relicensed under the MIT license at that time.

3 Scheduled Relicensing

3.1 Proprietary Ramp-up, Eventually Open Source (Pre-Open Source)

DOSP is usually adopted as an ongoing commercial strategy. It reserves a window of time for a company to sell the latest features under proprietary license before they become available to all under open license.

In addition to this common form of DOSP, we find delayed publication occurs in another notable form. In this form, projects plan to eventually be fully open but initially operate in a less open manner. The plan for such projects is to become full-fledged Open Source efforts once the project has matured or stabilized. This one-time delay at the start of a new project is, to us, different enough from other DOSP that maybe it should be placed in a whole other category. Still, it is a common form of time-delay in the Open Source world.

These projects begin development in a proprietary mode. During this pre-open-source period, their practices might reflect just about any variation of non-Open Source software. They might not publish any code. Or release their code under non-Open Source license, including by not explicitly specifying a license. They might only release binaries or release nothing. In short, these projects range from wholly, traditionally proprietary in nature to public collaborations that stop just short of an Open Source license.

Usually, these projects explain that they plan to become open, explain why it hasn't happened yet, and describe (sometimes vaguely) the conditions that will trigger a relicensing toward Open Source.

There are many possible reasons why a project might start out with some public visibility, whether of source or binaries, but not initially ship Open Source code. The ones we have observed:

- shame about poor code quality
- concern about security issues that may be apparent in unaudited source code
- initial uncertainty about which license to choose
- a need to procure permissions from other copyright holders
- a desire to establish a community, governance, or a legal entity

Although these scenarios involve an intent to publish something as Open Source in the future, they are also rather different from the DOSP cases we focus on in the rest of this document. They differ, for example with regard to whether the delay is desired by the authors, whether it's predictable to users, and whether it's expected to recur. Projects that start out proprietary with a stated plan to go open eventually are not practicing DOSP as a business model. While one might usefully consider the question of when to deviate from the principle of "be open from day one",[16] the commercially interesting tradeoffs are mostly found in projects that opt for an ongoing DOSP strategy.

3.2 Bounty and Sponsorship Delays

Another model is making individual software features or enhancements available to sponsors first, with a fixed time delay before making them available to the general public. An example of this is the North Road SLYR GIS software[17], which has a published feature roadmap and releases (and licenses) its implementation of each feature to sponsors first:

> While we fully intend to make the full SLYR plugin open source and freely publish the style/LYR/MXD conversion tools, we also require financial backing in order to support the significant time required to completely reverse engineer these file formats and develop quality tools supporting their use outside of the ESRI software ecosystem. Accordingly, the specifications and file parsing library will initially be closed source and available to SLYR license holders only. Exactly six months after we hit the pledged sponsorship levels for each stage of the project (check the progress below for each stage), we will open-source that component of the code and update the community version of the plugin.

This strategy was also used by the OPSI project, which created a bounty-like "co-funding" mechanism, which is still alluded to on the associated company's website[18]. Under this model, customers could sponsor the development of particular features, which would initially be available to sponsors and later to the public. However, this mechanism appears to have fallen out of use, as there are no recent co-funding opportunities, and the project currently appears to follow an open core model with paid subscriptions for proprietary extensions.

3.3 The Business Source License (BUSL)

The Business Source License (BUSL; sometimes "BSL"[19]) was originally written in 2016 by MariaDB for its MaxScale project. The current version of BUSL, 1.1, was released in 2017 and first used for MaxScale 2.1.0.[20]

BUSL requires a licensor to specify a "Change Date" and a "Change License". On the Change Date, which is some time in the future, the license of the covered artifact will change to the Change License, which is an Open Source license.[21]

MariaDB's Change Date for MaxScale is four years after the release of a specific version, and its Change License is GPLv2.

The Linux Foundation noted that several prominent projects switched away from open-source licenses from 2018 to 2023[22]. Not all of these adopted DOSP licenses[23], but those that did so adopted BUSL. These included CockroachDB, Couchbase, Terraform, and ArangoDB. The most prominent of these BUSL adopters was HashiCorp, which wrote

SCHEDULED RELICENSING

> BSL 1.1 is a source-available license that allows copying, modification, redistribution, non-commercial use, and commercial use under specific conditions. With this change we are following a path similar to other companies in recent years. These companies include Couchbase, Cockroach Labs, Sentry, and MariaDB, which developed this license in 2013. Companies including Confluent, MongoDB, Elastic, Redis Labs, and others have also adopted alternative licenses that include restrictions on commercial usage. In all these cases, the license enables the commercial sponsor to have more control around commercialization.

This change applied to almost all of the company's software, including popular software like HashiCorp Terraform, Vagrant, and Vault.

3.3.1 Anti-competition as a Motivation

Although HashiCorp's license change attracted the most attention and commentary, the BUSL was originally written by a database company. Some of the project developers wrote that they wanted to discourage other companies from competing directly with the developers' hosted database services, and that they doubted whether an Open Source license would manage to accomplish this.[24]

By default, BUSL prohibits uses in "production" before the Change Date. Licensors using the bare BUSL would thus expect commercial adopters to pay for a separate license permitting commercial use. However, several licensors add an Additional Use Grant (AUG) under the BUSL to allow for "production" uses other than those that are considered to compete with the developer's own commercial services. For example, ArcticDB provides the following Additional Use Grant[25]:

> You may make use of the Licensed Work under the terms of this License, provided that you may not use the Licensed Work for a Database Service.
>
> A "Database Service" is a commercial offering that allows third parties (other than your employees and contractors) to access the functionality of the Licensed Work by creating tables whose schemas are controlled by such third parties.

It appears that the project thus intends to immediately allow commercial uses, including for public services, as long as these don't entail charging money for hosting databases in particular. Several other BUSL adopters have analogous grants.

The AUG mechanism — including optional free-form text that exempts certain uses from BUSL's "production use" restrictions — complicates direct comparison of uses of the BUSL; we have not yet devised a taxonomy for making these comparisons.

SCHEDULED RELICENSING

Below is a table of sixteen well-known projects that now use BUSL, showing their Change Date and Change Licenses.

Project	BUSL Date	Change Date	Change License
MaxScale https://mariadb.com/resources/blog/releasing-bsl-1-1	2017-02-14	release date +4 years	GPL v2
CockroachDB https://www.cockroachlabs.com/docs/stable/licensing-faqs#bsl	2019-06-04	release date +3 years	Apache v2
ZeroTier https://www.zerotier.com/blog/on-the-gpl-to-bsl-transition	2019-08-28	5th cal. year	Apache v2
Sentry[26] https://blog.sentry.io/relicensing-sentry	2019-11-06	release date +3 years	Apache v2
Materialize[27] https://materialize.com/docs/license	2020-02-07 ?	daily +4 years[28]	Apache v2
CouchBase https://www.couchbase.com/blog/couchbase-adopts-bsl-license	2021-03-26	release date +4 years	Apache v2
Memgraph https://memgraph.com/blog/memgraph-2-0-release	2021-10-03 ?	release date +4 years	Apache v2
SurrealDB https://github.com/surrealdb/surrealdb/blob/main/LICENSE	2021-12-14 ?	release date +4 years	Apache v2
DragonflyDB https://github.com/dragonflydb/dragonfly/blob/main/LICENSE.md	2022-05-29	release date +5 years	Apache v2
ReadySet https://github.com/readysettech/readyset/blob/main/LICENSE	2022-08-03	release date +4 years	Apache v2
Akka https://www.lightbend.com/blog/why-we-are-changing-the-license-for-akka	2022-09-07	release date +3 years	Apache v2
Codecov https://about.codecov.io/blog/codecov-is-now-open-source	2023-08-02	release date +3 years	Apache v2
Terraform (etc.)[29] https://www.hashicorp.com/blog/hashicorp-adopts-business-source-license	2023-08-10	release date +4 years	MPL 2.0
ArangoDB https://arangodb.com/2023/10/evolving-arangodbs-licensing-model-for-a-sustainable-future	2023-10-11	release date +4 years	Apache v2
ArcticDB https://github.com/man-group/ArcticDB/blob/master/LICENSE.txt	always	release date +2 years	Apache v2
evitaDB https://github.com/FgForrest/evitaDB/blob/dev/LICENSE	always	4th cal.year	Apache v2

SCHEDULED RELICENSING

BUSL is notionally designed to apply to specific software *releases*, so that a Change Date applies to a particular version of a code base. That means that, for a project with an ongoing DOSP practice, BUSL is meant to be re-applied periodically with updated details. The majority of projects we've seen have not yet demonstrated how they'll handle this process on an ongoing basis. Most don't have a clearly-visible and systematic way to apply BUSL updates to ongoing development, although one project (Materialize) automatically updates its BUSL grant every day in order to keep the Change Date at a fixed point in the future. For some projects, it is unclear at first glance exactly which version or versions of the code base the BUSL grant is meant to apply to. The Change Date concept may be complicated by the fact that not all contemporary software projects have a reliable schedule of discrete "release" events.

3.3.2 Differences From Other Licensing Strategies

MariaDB describes some of the differences between BUSL and other commonly-used licensing strategies as follows:[30]

> **Q: How is the BSL different from Open Core?**
>
> **A:** Open core offers some code under Open Source terms, but non-core code is not under Open Source terms, is not available in source form, cannot be modified and compiled, cannot be contributed to, and will never be Open Source. By using Open Core software, like with closed source code, you are locked to one vendor. With BSL, as compared to Open Core, the source code is available from the start, can be modified and compiled, contributions are encouraged, the product will become fully Open Source after a period of time and remains free for non-production use. The importance of the eventual Open Source is that users are free from vendor lock-in. If the vendor decides to stop contributing to the code, users have open access and can modify, update and extend as needed.
>
> **Q: How is the BSL different from dual GPL/commercial licensing?**
>
> **A:** When using dual licensing with GPL, companies must pay for a commercial license to use the software with their closed source code. With BSL, the companies are only paying for the software if they want to make production use of the software. From a vendor perspective, GPL dual licensing only works for infrastructure products that other companies want to deeply embed in their product. BSL works for any kind of software product.

This is echoed in statements by several BUSL adopters that they sought a way to make downstream commercial users who did not redistribute derived works pay for the use of their software (typically in cloud environments), or wanted to prevent downstream commercial users from directly competing with the initial developer's own service offerings.

We do not know why MySQL's FAQ item mentions only GPL and not AGPL, nor whether those other BUSL adopters considered AGPL.

3.4 Consequences

Projects that change from an open-source license to a delayed open-source license have attracted criticism, with some people pledging to switch to other projects or even to maintain competitive forks of the prior open-source versions.

SCHEDULED RELICENSING

The most consequential such effort appears to be OpenTofu, a fork of HashiCorp's Terraform announced soon after Terraform was relicensed under BUSL.[31] OpenTofu has announced several corporate sponsorships, apparently plans to hire multiple full-time developers, and has organized itself as a project of the Linux Foundation. The fork's creators complained that the prior Open Source license of Terraform had encouraged people to develop professional expertise with the software and to use it as a part of their infrastructure — in essence, that HashiCorp performed a bait-and-switch by moving from Open Source licensing to BUSL. They also noted concerns about whether Terraform users could be confident about whether their particular uses would be considered commercially competitive with HashiCorp.

As far as we can tell, most other forks of recently-reproprietized software have not attracted the same levels of attention, participation, or adoption. However, we have not done an extensive survey on this question and welcome further research.

It could be harder for projects under non-Open Source terms to receive or accept outside contributions, both because people may be less motivated to make them and because the licensing status of the resulting contributions is more complicated. However, some projects that have switched to BUSL (or other licenses) continue to accept outside contributions subject to a contributor license agreement ("CLA"), which grants certain rights to the original developer. HashiCorp, for example, has a CLA for its projects[32], and a bot that checks whether the authors of pull requests have signed it, so that their contributions will not be incorporated into the codebase until and unless they do so. The company does continue to receive some outside code contributions to its BUSL-licensed projects, including Terraform. HashiCorp's CLA is "non-exclusive"; an outside contributor could conceivably continue to contribute the same patches to a HashiCorp BUSL project and a non-HashiCorp fork of the same project, assuming that the codebases haven't diverged too far over time to make this practical.

3.5 Other Examples

The Child Mind Institute created its own Delayed Open Source Attribution License (DOSA), which has a three-year period during which only noncommercial uses are permitted, for its MindLogger software.[33] However, as of 2023, MindLogger and other projects from the Child Mind Institute are licensed under the CPAL Open Source license, with no associated delay.

The Poké Classic Framework has a conditional license which limits uses of the code but which converts to AGPL if the original developer ceases to operate a service based on the code.[34]

Roughly contemporaneously with MariaDB's development of BUSL, Ben Boyter proposed a "GPL time bomb" (later renamed to simply "eventually open") that is conceptually similar to BUSL with an AUG specifying a limited number of users within an organization.[35] This approach was used for Boyter's "searchcode-server" project[36], but no new development has taken place on this codebase since 2020, so the whole project is apparently now licensed under GPL v3.

In November 2023, Sentry published its own "Functional Source License" (FSL)[37] and relicensed its own previously BUSL-licensed software under it.[38] The FSL prohibits, during a period of two years, uses of covered software to provide services that "compete" with the original developer's commercial service offerings. Other uses are generally permitted. Following this two-year period, the software is licensed under MIT or Apache terms, without the competition restriction.[39]

SCHEDULED RELICENSING

BUSL expressly permits certain parameters to be set by each individual adopter (including arbitrary free-form license text in AUGs, so long as that text grants additional permissions rather than removing permissions). Sentry disapproved of the resulting proliferation of variant terms and differently-phrased AUGs; it stated that, from the licensee's point of view, each BUSL instance is actually a substantively different license. Accordingly, the FSL roughly follows the BUSL's approach, while freezing a particular set of terms.[40]

Several cloud-oriented software projects that switched away from Open Source licensing in the past few years also adopted license terms with non-competition clauses (but permanently, without any time limits). Conversely, several projects that adopted BUSL included AUGs that allow commercial uses so long as these aren't charging third parties for the service of hosting instances of the software, or so long as they don't otherwise compete with the original developer's own service offerings. The FSL codifies a version of this policy in the main license itself, rather than adding it as an optional additional permission.

NOTES

[16]See http://archive.civiccommons.org/2011/01/be-open-from-day-one/index.html for more about this principle.

[17]See https://north-road.com/slyr/.

[18]See https://web.archive.org/web/20220216132657/https://www.uib.de/en/opsi-cofunding/cofunding/.

[19]Most adopters of this license refer to it as "BSL", but this acronym was previously used for the Boost Software License. The SPDX license identifier for the Business Source License is "BUSL" (see https://spdx.org/licenses/ for the full SPDX list).

[20]See https://mariadb.com/resources/blog/releasing-bsl-1-1/.

[21]Specifically, the Change License must be either GPL 2.0 or else a license that is compatible with GPL 2.0 or a later version.

[22]See https://www.linuxfoundation.org/blog/how-open-source-foundations-protect-the-licensing-integrity-of-open-source-projects.

NOTES

[23]The trend identified by the Linux Foundation began in late 2018, with two major database projects, Redis and MongoDB, changing their licenses. Both eventually ended up adopting the Server-Side Public License (SSPL). SSPL was proposed as an Open Source license, but was not ultimately accepted as Open Source by OSI's license review process. Some proponents of this license continue to argue that it meets criteria to be considered a form of free and open source licensing.

[24]It's interesting to note that the majority of the projects we've identified that relicensed under BUSL are database systems. It's possible that there was a degree of "social contagion" as database developers observed several of their peers relicensing away from Open Source at roughly the same time, either to BUSL or to other licenses that restrict licensees from operating commercial services. As noted above, database developers were also responsible for several other relicensing decisions starting in 2018.

[25]This same text is also used by several other projects, and we have not determined which project originated it. There are also other variants with similar effect.

[26]Sentry subsequently relicensed under its own "Functional Source License"; see below for further discussion.

[27]Not to be confused with the Materialize CSS project, which is released under the MIT license.

[28]Differently from other BUSL-licensed projects, Materialize uses a bot to update the Change Date every day (not just on the occasion of release events), so that it always reflects a date exactly four years after the present date.

[29]"HashiCorp Terraform, Packer, Vault, Boundary, Consul, Nomad, Waypoint, and Vagrant" are identified as relicensed by https://www.hashicorp.com/license-faq.

[30]The quotation is from https://mariadb.com/bsl-faq-mariadb/.

[31]See https://opentofu.org/.

[32]See, for example, https://cla.hashicorp.com/hashicorp/terraform. Note that HashiCorp did previously have a CLA in place for outside contributions, since at least 2019.

[33]The developers even announced this in a journal article announcing the development of the software. See Arno Klein et al., "Remote Digital Psychiatry for Mobile Mental Health Assessment and Therapy: MindLogger Platform Development Study" (2021), available at https://www.ncbi.nlm.nih.gov/pmc/articles/ PMC8663601/; for the license text, see https://github.com/ChildMindInstitute/DOSA-license.

[34]See https://github.com/mm201/pkmn-classic-framework.

[35]See https://boyter.org/2016/08/gpl-time-bomb-interesting-approach-foss-licensing/.

[36]See https://www.searchcode.com/.

[37]See https://fsl.software/.

[38]See Sentry's announcement and discussion at https://blog.sentry.io/introducing-the-functional-source-license-freedom-without-free-riding/.

[39]FSL exists in exactly two variants, one which converts to the MIT license after two years, and one which converts to the Apache 2.0 license after two years.

[40]A similar problem of license proliferation was identified years ago among Open Source licenses; see https://opensource.org/proliferation/ and https://en.wikipedia.org/wiki/License_proliferation for more context.

4 “Grace Period” Reciprocal Licensing

One licensing practice often described as related to DOSP is implemented in the Bootstrap Open Source License (BOSL), previously called the Transitive Grace Period Public License (TGPPL). This license was mainly devised by Zooko Wilcox-O’Hearn.[41]

It is worth pointing out that the BOSL has no connection to the Bootstrap web framework project, which is under the MIT license. Both projects independently use the term “bootstrap” to refer to the concept of bootstrapping.[42]

Instead of making an initially proprietary license grant that later transforms into an open-source license, the BOSL makes an initially non-reciprocal (BSD-style) license grant that later transforms into a reciprocal (GPL-style) license. This is intended to allow downstream code reuse in proprietary software projects, but only for a limited time, something Wilcox-O’Hearn characterized as a compromise between non-copyleft and copyleft Open Source licensing models.[43]

Since both the start and end-state licenses of the BOSL are themselves Open Source, we do not regard the BOSL as a form of delayed open-source publication as defined by this report. Rather, it seems to be an unconventional form of Open Source publication with time-varying Open Source terms. While the BOSL has not been approved by the Open Source Initiative, it appears to us to be compatible with the Open Source Definition, and — unlike BUSL, for instance — is claimed by its authors to be a form of Open Source licensing.

One way to view the distinction between delayed open-source licensing and grace period reciprocal licensing is that the former aims to compromise between proprietary and Open Source licensing, where the latter aims to compromise between non-reciprocal and reciprocal licensing — in both cases by modifying the license terms after a delay.

NOTES

[41] It implements a strategy previously proposed by Ted Ts’o, at https://thunk.org/tytso/TPL.html.

[42] Furthermore, neither has any connection to the “Boost” project nor to the Boost Software License, though when reading quickly it is easy to make a transposition mistake. Not that this ever happened to any of this report’s authors.

[43] See, for example, the presentation at https://tahoe-lafs.org/~zooko/tgppl.pdf.

5 Other Terminology and Practices

We've encountered a number of other terms that can describe DOSP or the licensing mechanisms used to implement it.

- **Eventual (open) source; scheduled licensing.**
  Lawrence Rosen's book *"Open Source Licensing: Software Freedom and Intellectual Property Law"* refers to "eventual source" or "eventually open source" software, giving the example of Aladdin GhostScript. He also calls this "scheduled licensing".
- **Springing licenses.**
  A research report from Creative Commons refers to "springing licenses" (licenses that grant additional permissions after a period of time, or when some other condition has been met). Creative Commons was mainly interested in the possibility of developing licenses that would grant additional permissions over time, after a period of greater exclusivity.[44]
- **Scheduled relicensing.**
  Kyle E. Mitchell refers to "scheduled relicensing" in *"A Short, Simple Template for Scheduled Relicensing".*[45]

One can also distinguish between a public pledge to relicense on a schedule (as GhostScript did) and a license document whose text includes date or other restrictions. In the former case, the delayed release is implemented by human beings (actively making a new software release including new license text); in the latter case, it is automatic.

We do not consider "unplanned" Open Source releases to be examples of DOSP. There are a number of high-profile cases of proprietary projects that were retroactively relicensed as Open Source as a result of a one-off decision. Where developers originally had no announced plan or intention to do this, we think this is best considered a separate phenomenon, not a "delayed" release.

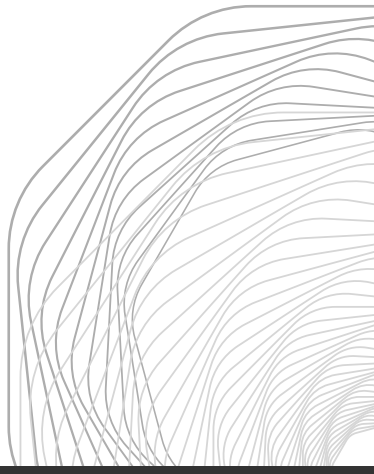
Many people also mentioned the custom among some video game developers of releasing code (though usually not assets such as graphics and sound) from proprietary video games that are no longer commercially important. This is a relatively widespread practice, with Wikipedia identifying dozens of instances.[46] The released game code is most often licensed under an established Open Source license.

Some companies like id Software have made such releases for multiple video game generations. While many of these developers apparently had a general intention to make their games Open Source, in whole or in part, at some point in the future, there was usually no public commitment to do so on any particular schedule or under any particular circumstances. This practice is thus not a core example of DOSP.

A "delayed open access" model, applied to research articles, has become popular for academic journals as a compromise between more restrictive journal licensing and open-access publishing.[47]

OTHER TERMINOLOGY AND PRACTICES

As of November 2023, Wikipedia identifies by name 108 journals that currently follow some form of this model, but cites a 2013 study that reportedly reviewed 492 journals with such a policy. In this context, journals may apply an "embargo period" to create an incentive for some journal users to pay for subscriptions or article access in order to read recent research. The license terms applied at the expiration of these embargo periods permit the public to read articles at no charge, but may or may not be equivalent to Open Source licensing.



NOTES

[44]See "Creative Commons Final Report: On the Viability and Development of Springing Licenses" https://creativecommons.org/wp-content/uploads/2018/07/Springing-licenses-FINAL.pdf.

[45]See https://writing.kemitchell.com/2023/10/24/Scheduled-Relicensing.

[46]Wikipedia lists these examples at https://en.wikipedia.org/wiki/List_of_commercial_video_games_with_later_released_source

[47]See https://en.wikipedia.org/wiki/Delayed_open-access_journal.

6 Conclusions

DOSP has been in use since the early days of Open Source.[48] Companies (it's always companies) tend to use it to preserve commercial advantage while attempting to keep as many of the advantages of Open Source as they can. To what extent and in what ways they succeed in this attempt is not yet entirely clear to us, and some of the questions in Section 7 are meant to elucidate this. We suspect that as the delay increases, the benefits of Open Source decrease. Exploring the tradeoff between those benefits and the period of exclusive exploitation might merit future research.

From our research for this report, and from our conversations with Open Source projects and with our clients over the years, we see some evidence that DOSP is most likely to achieve the licensor's goals for fast-moving, cutting-edge software, where access to the latest features is commercially significant. For software whose year-old versions are, for the typical user, suitable replacements for the latest proprietary versions, the market segmentation offered by DOSP weakens or disappears. Those cases essentially collapse down to a proprietary-licensing scheme with proprietary and A/GPL options.

By far the most important conclusion we draw from our research is that there has been a lot more experimentation and variety in DOSP than we realized — more projects have tried it than we knew, and have tried it in far more varying ways than we knew. Although there seems to be a slight trend towards convergence recently, at least in terms of DOSP license texts, there is no guarantee that this trend will continue, and in any case the same licensing terms will often lead to different outcomes for different projects.

This report should be viewed as a starting point. We strongly hope that qualified researchers will find opportunities to pursue some of the questions suggested in Section 7, and that in doing so they will discern patterns that lead them to even more important questions that we haven't thought of.

NOTES

[48]Some of that period occurred before the term "Open Source" was coined in the context of software licensing; the term "free software" was the most commonly used term for this kind of licensing then.

7 Future Research Questions

- **AGPL versus DOSP licensing.**
  The GNU Affero General Public License (AGPL)[49] arguably aims to address some of the same concerns as the BUSL or the FSL — particularly concerns about some forms of free-riding by downstream adopters. Instead of forbidding commercial (or competitive) uses, the AGPL imposes a copyleft-like requirement to publicly disclose and publicly license the source code of modifications, whenever those modifications are used to operate publicly-accessible services.

  The authors of this report have observed debates about the relative merits of the AGPL and BUSL. Interestingly, proponents of each license often agreed that both licenses might, in principle, address similar concerns about companies adopting an Open Source code base — sometimes in direct competition with the original developer's services — without rewarding its original developer either with money or with code contributions.[50] However, the proponents didn't agree about which license better responds to this scenario; in at least some cases, that may imply a more basic disagreement about values and whether Open Source licensing is intrinsically preferable.

  Did any BUSL adopters seriously consider adopting AGPL? If not, why not? If so, why did they end up preferring BUSL's approach?

- **Effects on outside contributions.**
  How much are outside contributions affected by using (or switching to) a DOSP model rather than an Open Source license? Can any contribution trends be clearly and confidently attributed to relicensing?

- **Taxonomy of BUSL Additional Use Grants.**
  The BUSL default is to prohibit production use, but most adopters of BUSL have used AUG clauses that grant permission for production use that doesn't compete commercially with the software developers' own business, or concretely that doesn't involve charging for access to a hosted instance of the software.

  How similar are the different formulations of this notion? Are there any other permissions that appear in practice in BUSL AUG clauses beyond the notion of not competing with services run by or licensed by the upstream developer? Is there a significant minority of BUSL adopters that aim to restrict (and sell licenses for) "production" use more generally?

- **Relicensing after initial Open Source publication.**
  Is it a conscious strategy — or at least a conspicuous option — to start out a new project under a purely Open Source license in order to garner interest and mindshare, and then subsequently relicense under a DOSP license? Some of HashiCorp's critics noted that the company had given adopters incentives to become expert in, or otherwise reliant upon, the company's software while it was under an Open Source license, and then tried to benefit from that familiarity and adoption by changing the license terms in the future.

FUTURE RESEARCH QUESTIONS

If some of the most popular DOSP-relicensed projects had started out under DOSP rather than Open Source terms from the outset, would they have attracted the same level of interest and adoption?

- **Why has a fork of Terraform attracted so many contributions and so much interest compared to forks of other projects?**
  It's too early to say whether the OpenTofu project will broadly outcompete Terraform among various audiences, but it's clear that this fork started off with a bang, immediately garnering substantial interest, sponsorships and financial commitments, and endorsements from various companies and developers. However, Open Source forks of other HashiCorp projects are comparatively stagnant and underpublicized. Similarly, other BUSL relicensing events did not seem to result in highly active forks (although some may have increased interest in existing Open Source competitors to the relicensed projects).

  What's special about the OpenTofu effort, or about Terraform or its community, that could account for these differences? Did Terraform's market share in its niche play a large role? Was Terraform particularly indispensable for its users in comparison to some other relicensed projects? And can answers to these questions about OpenTofu and Terraform be applied to other projects that have undergone similar relicensing schisms?

NOTES

[49]See https://www.gnu.org/licenses/agpl-3.0.html.

[50]Critics of both licenses have similarly argued that it might be hard to tell exactly which activities related to online service provision are meant to be "caught", in comparison to non-copyleft licenses.

8 Acknowledgments

This research was made possible thanks to a donation from Sentry and OSI's individual members.

The authors are grateful to the Open Source Initiative for giving us the opportunity to explore this topic and to make, we hope, a small contribution to the future health of Open Source by analyzing industry trends likely to affect it.

Many people responded to our call for examples. They always accompanied their submissions with historical context, and often with thoughtful analysis as well. We thank them all sincerely; this report would not have been possible without their help. We list them here in no particular order (in fact, in mechanically randomized order): Matija Šuklje, AntiCompositeNumber [sic], Simon Phipps, Damiano Verzulli, Josh Berkus, Marcin Koziej, Alex Scammon, Thomas Sandmann, Royce Williams, Ross Mounce, Nick Vidal, Stuart D. Gathman, Mark Chapman, Samuel Tardieu, Chad Whitacre, Johann Schöpfer, Abby Kearns, André Wolski, Heather Meeker, Neil Carpenter, Sam Ramji, Anthony Nowocien, Stefano Maffulli, and Jesse Bickel.

The authors are solely responsible for the contents of this report, including but not limited to any errors.

Disclaimer
The authors have not been influenced by Sentry nor by the Open Source Initiative in our choice of examples, our choice of questions, our analysis, or our conclusions.

Author Biographies

Seth Schoen is an independent consultant in San Francisco. He was the first Staff Technologist at the Electronic Frontier Foundation and was a part of the team that created the Let's Encrypt certificate authority. He has a long-time interest in copyright and licensing issues and has published research in computer security, as well as testifying before several courts and government agencies.

James Vasile is a founding partner at Open Tech Strategies. Previously, he was founding director of the Open Internet Tools Project, a Senior Fellow at the Software Freedom Law Center, and Director of the Freedom Box Foundation. James serves on the governing or advisory boards of Horizons, Brave New Software, The Electronic Frontier Foundation, and the New York Civil Liberties Union.

Karl Fogel is a founding partner at Open Tech Strategies. He is the author of the popular "Producing Open Source Software" and a founding developer of the Subversion project. Throughout his career, Karl has advised a wide range of projects and companies, including Google, Canonical, O'Reilly Media, and Code for America / Civic Commons. Karl has also been a board member at the Open Source Initiative, an Open Internet Tools Project Fellow at the New America Foundation, and is a member of the Apache Software Foundation.