Week:
12 of 12

COMP 2120 / COMP 6120

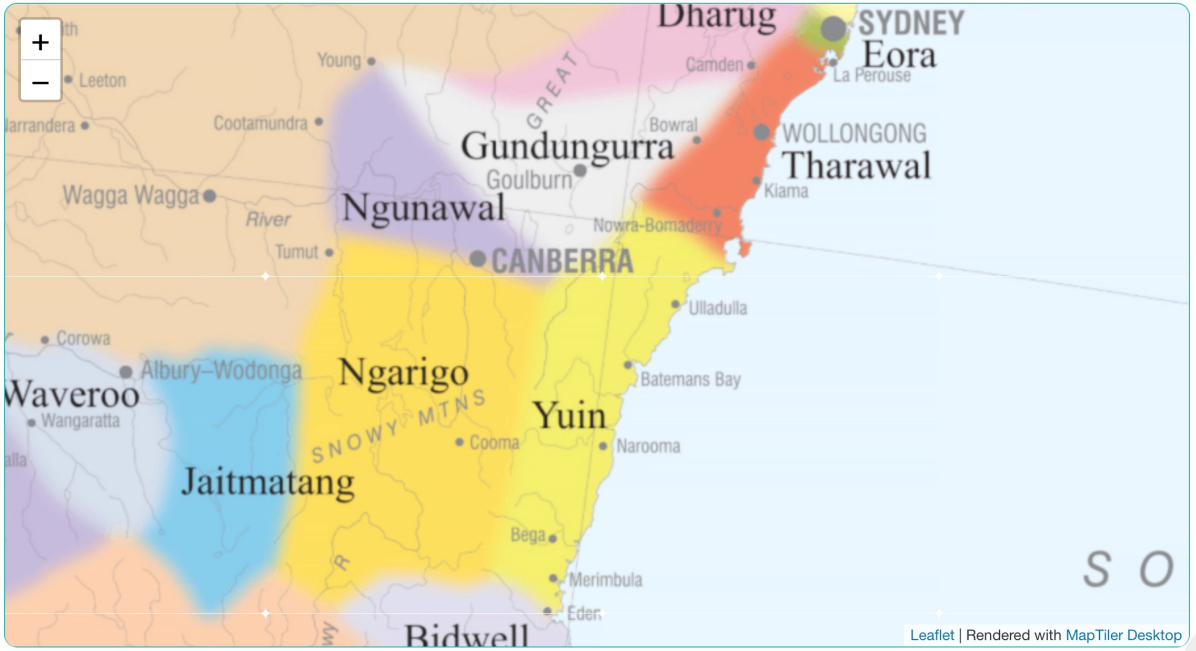
OPEN SOURCE

A/Prof Alex Potanin and Dr Melina Vidoni

ANU Acknowledgment of Country

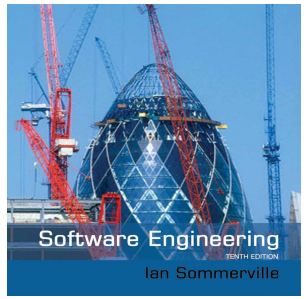
AIATSIS Explore Family history Collection Research Education What's new About Shop Search

"We acknowledge and celebrate the First Australians on whose traditional lands we meet, and pay our respect to the elders past and present."



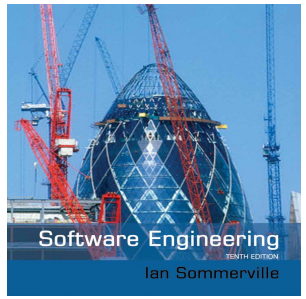
https://aiatsis.gov.au/explore/map-indigenous-australia

Open source development



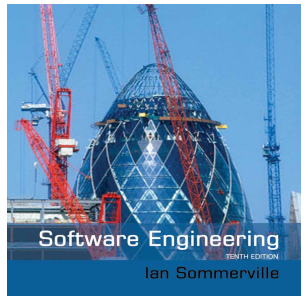
- Open source development is an approach to software development in which the source code of a software system is published and volunteers are invited to participate in the development process
- Its roots are in the Free Software Foundation (www.fsf.org), which advocates that source code should not be proprietary but rather should always be available for users to examine and modify as they wish.
- Open source software extended this idea by using the Internet to recruit a much larger population of volunteer developers. Many of them are also users of the code.

Open source systems



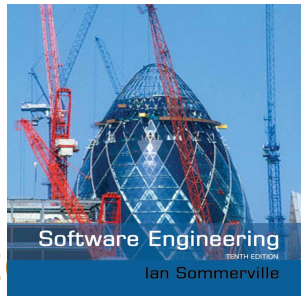
- The best-known open source product is, of course, the Linux operating system which is widely used as a server system and, increasingly, as a desktop environment.
- Other important open source products are Java, the Apache web server and the mySQL database management system.

Open source issues



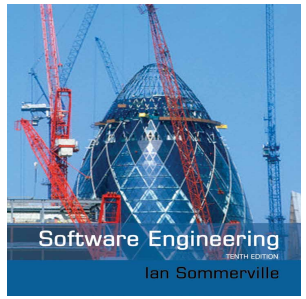
- Should the product that is being developed make use of open source components?
- Should an open source approach be used for the software's development?

Open source business



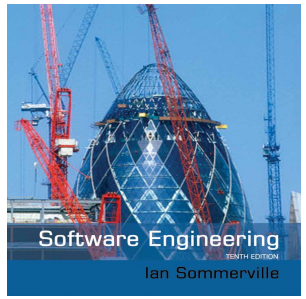
- More and more product companies are using an open source approach to development.
- Their business model is not reliant on selling a software product but on selling support for that product.
- They believe that involving the open source community will allow software to be developed more cheaply, more quickly and will create a community of users for the software.

Open source licensing



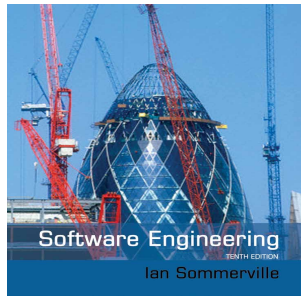
- A fundamental principle of open-source development is that source code should be freely available, this does not mean that anyone can do as they wish with that code.
  - Legally, the developer of the code (either a company or an individual) still owns the code. They can place restrictions on how it is used by including legally binding conditions in an open source software license.
  - Some open source developers believe that if an open source component is used to develop a new system, then that system should also be open source.
  - Others are willing to allow their code to be used without this restriction. The developed systems may be proprietary and sold as closed source systems.

License models



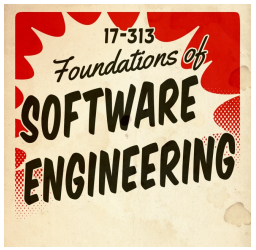
- The GNU General Public License (GPL). This is a so-called 'reciprocal' license that means that if you use open source software that is licensed under the GPL license, then you must make that software open source.
- The GNU Lesser General Public License (LGPL) is a variant of the GPL license where you can write components that link to open source code without having to publish the source of these components.
- The Berkley Standard Distribution (BSD) License. This is a non-reciprocal license, which means you are not obliged to re-publish any changes or modifications made to open source code. You can include the code in proprietary systems that are sold.

License management

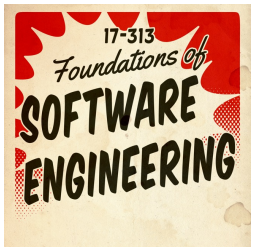


- Establish a system for maintaining information about open-source components that are downloaded and used.
- Be aware of the different types of licenses and understand how a component is licensed before it is used.
- Be aware of evolution pathways for components.
- Educate people about open source.
- Have auditing systems in place.
- Participate in the open source community.

“Free as in free speech.”



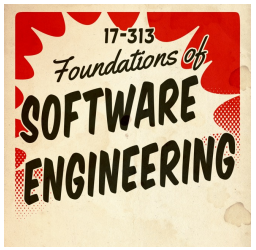
Open Source

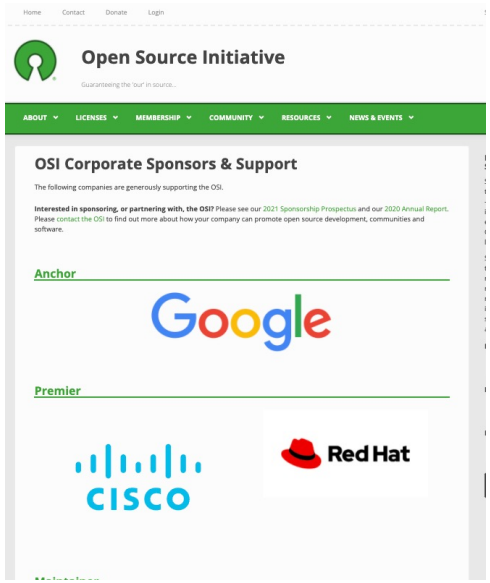


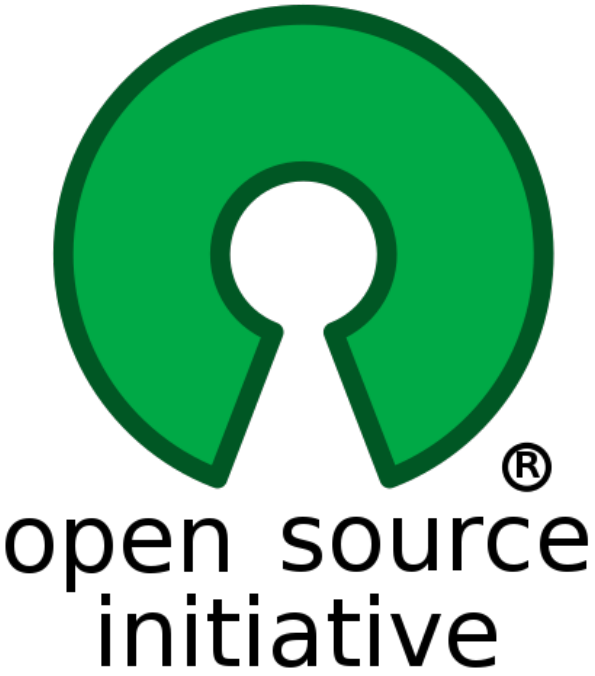
aka Free Software

aka Free and Open Source Software

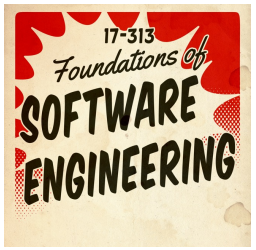
Open Source



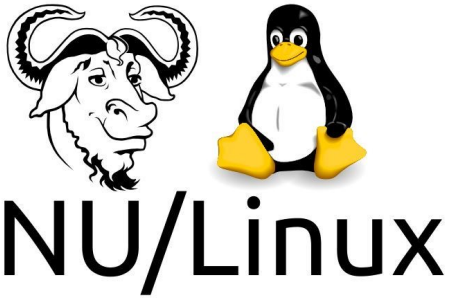




Free Software vs Open Source



- Free software origins (70-80s ~Stallman)
  - ~~Cultish~~ Political goal
  - Software part of free speech
    - free exchange, free modification
    - proprietary software is unethical
    - security, trust
  - GNU project, Linux, GPL license



- Open source (1998 ~ O'Reilly)
  - Rebranding without political legacy
  - Emphasis on internet and large dev./user involvement
  - Openness toward proprietary software/coexist
  - (Think: Netscape becoming Mozilla)

Free Software vs Open Source

- The freedom to run the program as you wish, for any purpose **(freedom 0).**
- The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

The Open Source Definition

free-redistribution

Introduction

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost, preferably downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

The license may restrict source-code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product

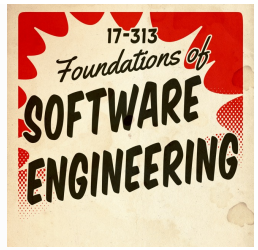
The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software

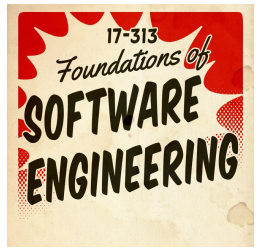
The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

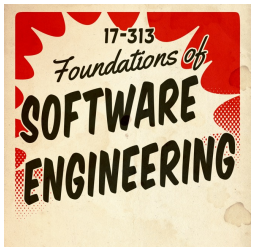
10. License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.



The Cathedral and the Bazaar





https://www.tesla.com/blog/all-our-patent-are-belong-you

All Our Patent Are Belong To You

Elon Musk, CEO • June 12, 2014

Yesterday, there was a wall of Tesla patents in the lobby of our Palo Alto headquarters. That is no longer the case. They have been removed, in the spirit of the open source movement, for the advancement of electric vehicle technology.

Tesla Motors was created to accelerate the advent of sustainable transport. If we clear a path to the creation of compelling electric vehicles, but then lay intellectual property landmines behind us to inhibit others, we are acting in a manner contrary to that goal. Tesla will not initiate patent lawsuits against anyone who, in good faith, wants to use our technology.

When I started out with my first company, Zip2, I thought patents were a good thing and worked hard to obtain them. And maybe they were good long ago, but too often these days they serve merely to stifle progress, entrench the positions of giant corporations and enrich those in the legal profession, rather than the actual inventors. After Zip2, when I realized that receiving a patent really just meant that you bought a lottery ticket to a lawsuit, I avoided them whenever possible.

At Tesla, however, we felt compelled to create patents out of concern that the big car companies would copy our technology and then use their massive manufacturing, sales and marketing power to overwhelm Tesla. We couldn't have been more wrong. The unfortunate reality is the opposite: electric car programs (or programs for any vehicle that doesn't burn hydrocarbons) at the major manufacturers are small to non-existent, constituting an average of far less than 1% of their total vehicle sales.

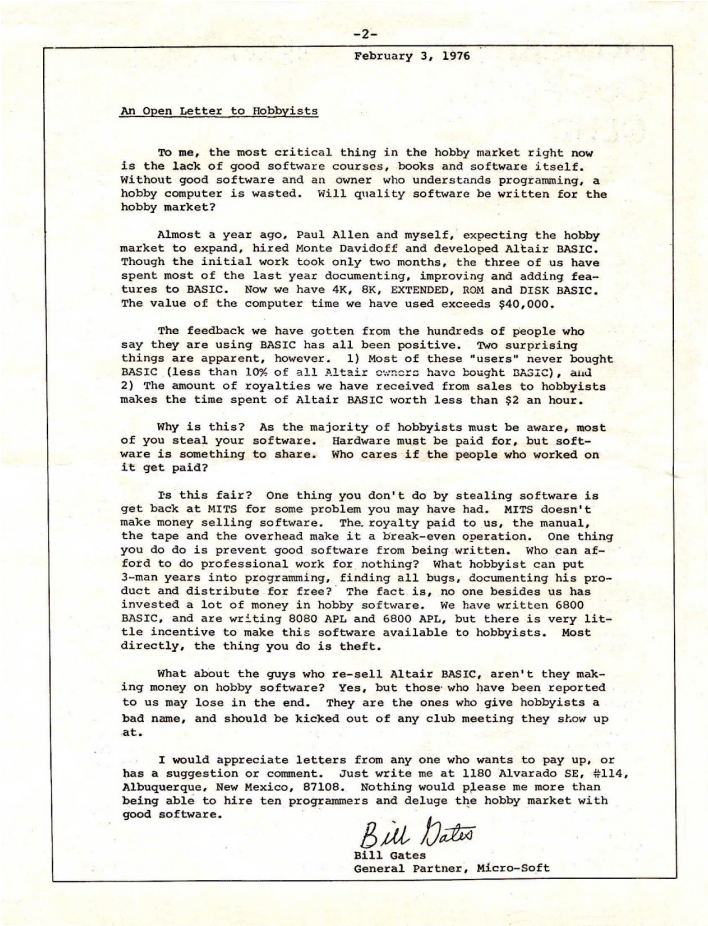
At best, the large automakers are producing electric cars with limited range in limited volume. Some produce no zero emission cars at all.

Given that annual new vehicle production is approaching 100 million per year and the global fleet is approximately 2 billion cars, it is impossible for Tesla to build electric cars fast enough to address the carbon crisis. By the same token, it means the market is enormous. Our true competition is not the small trickle of non-Tesla electric cars being produced, but rather the enormous flood of gasoline cars pouring out of the world's factories every day.

We believe that Tesla, other companies making electric cars, and the world would all benefit from a common, rapidly-evolving technology platform.

Technology leadership is not defined by patents, which history has repeatedly shown to be small protection indeed against a determined competitor, but rather by the ability of a company to attract and motivate the world's most talented engineers. We believe that applying the open source philosophy to our patents will strengthen rather than diminish Tesla's position in this regard.

Tables have turned



-2-

February 3, 1976

An Open Letter to Hobbyists

To me, the most critical thing in the hobby market right now is the lack of good software courses, books and software itself. Without good software and an owner who understands programming, a hobby computer is wasted. Will quality software be written for the hobby market?

Almost a year ago, Paul Allen and myself, expecting the hobby market to expand, hired Monte Davidoff and developed Altair BASIC. Though the initial work took only two months, the three of us have spent most of the last year documenting, improving and adding features to BASIC. Now we have 4K, 8K, EXTENDED, ROM and DISK BASIC. The value of the computer time we have used exceeds $40,000.

The feedback we have gotten from the hundreds of people who say they are using BASIC has all been positive. Two surprising things are apparent, however. 1) Most of these "users" never bought BASIC (less than 10% of all Altair owners have bought BASIC), and 2) The amount of royalties we have received from sales to hobbyists makes the time spent of Altair BASIC worth less than $2 an hour.

Why is this? As the majority of hobbyists must be aware, most of you steal your software. Hardware must be paid for, but software is something to share. Who cares if the people who worked on it get paid?

Is this fair? One thing you don't do by stealing software is get back at MITS for some problem you may have had. MITS doesn't make money selling software. The royalty paid to us, the manual, the tape and the overhead make it a break-even operation. One thing you do do is prevent good software from being written. Who can afford to do professional work for nothing? What hobbyist can put 3-man years into programming, finding all bugs, documenting his product and distribute for free? The fact is, no one besides us has invested a lot of money in hobby software. We have written 6800 BASIC, and are writing 8080 APL and 6800 APL, but there is very little incentive to make this software available to hobbyists. Most directly, the thing you do is theft.

What about the guys who re-sell Altair BASIC, aren't they making money on hobby software? Yes, but those who have been reported to us may lose in the end. They are the ones who give hobbyists a bad name, and should be kicked out of any club meeting they show up at.

I would appreciate letters from any one who wants to pay up, or has a suggestion or comment. Just write me at 1180 Alvarado SE, #114, Albuquerque, New Mexico, 87108. Nothing would please me more than being able to hire ten programmers and deluge the hobby market with good software.

Bill Gates

Bill Gates
General Partner, Micro-Soft

UNDERSTANDING LICENSES
NOTE: IANAL (I AM NOT A LAWYER)

Why learn about licenses?

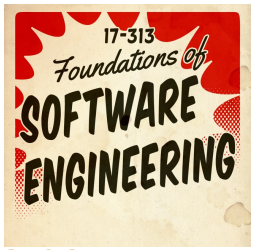
- Companies will avoid certain licenses – commonly the copyleft licenses
- Specific licenses may provide competitive advantages
- You may eventually want to release open source software or become more involved in an open source project

Open Source Licenses

Software	Percentage
MIT License	24%
GNU General Public License (GPL) 2.0	23%
Apache License 2.0	16%
GNU General Public License (GPL) 3.0	9%
BSD License 2.0 (3-clause, New or Revised) License	6%
GNU Lessor General Public License (LGPL) 2.1	5%
Artistic License (Perl)	4%
GNU Lesser General Public License (LGPL) 3.0	2%
Microsoft Public License	2%
Eclipse Public License	2%

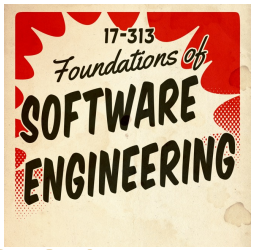
List from: https://www.blackducksoftware.com/resources/data/top-20-open-source-licenses

GNU General Public License: The Copyleft License



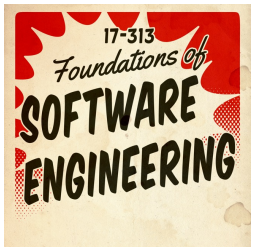
- Nobody should be restricted by the software they use. There are four freedoms that every user should have:
  - the freedom to use the software for any purpose,
  - the freedom to change the software to suit your needs,
  - the freedom to share the software with your friends and neighbors, and
  - the freedom to share the changes you make.
- Code must be made available
- Any modifications must be relicensed under the same license (copyleft)

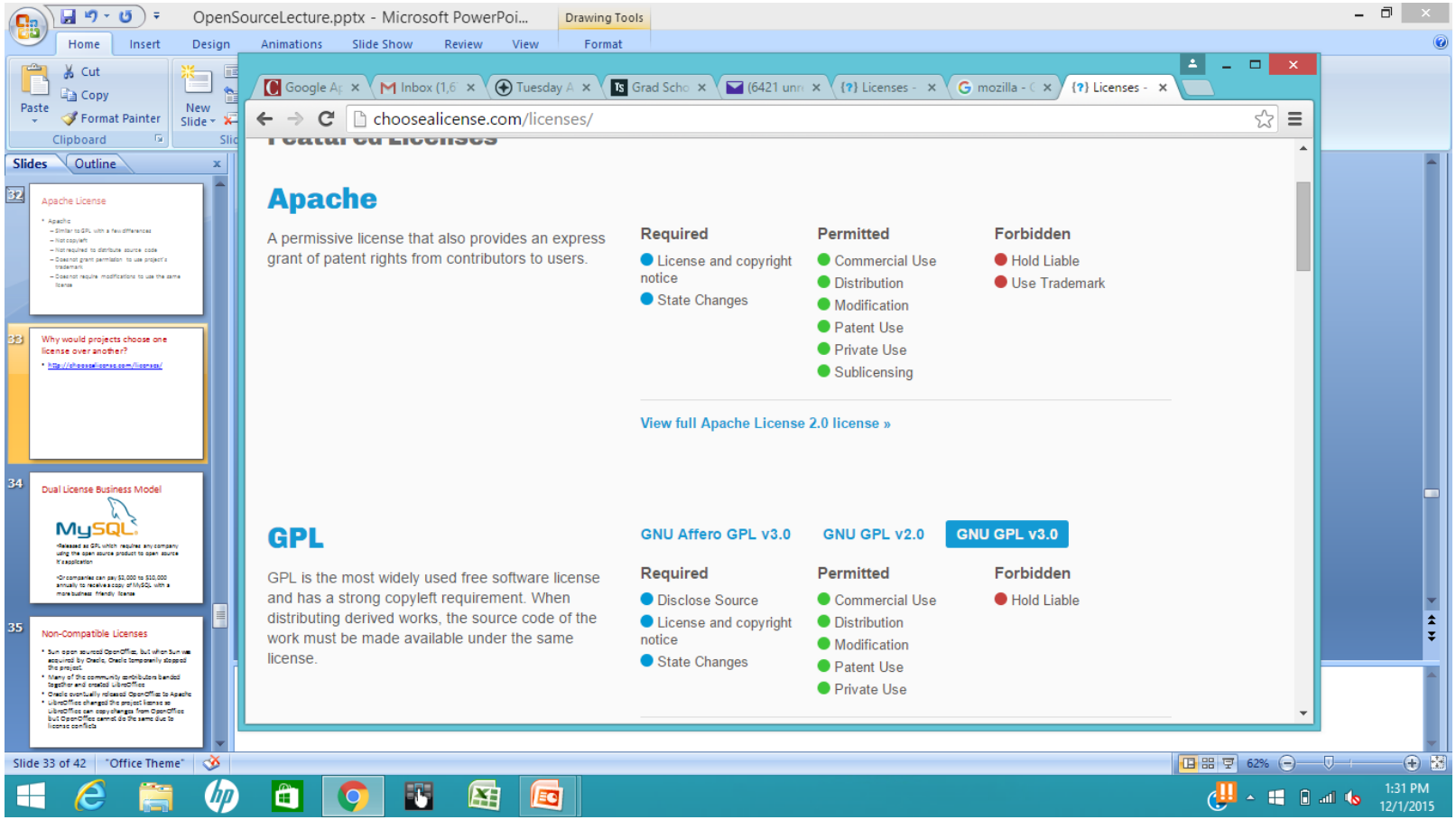
GPL 2.0 and 3.0 – Addresses free software problems



- 2.0 - Court ruling cannot nullify the license and if a court decision and this license contradict in distribution requirements, then the software cannot be distributed
- 3.0 – patent grant and prevent Tivoization
- Not compatible with each other; Can't copyleft both at the same time – phrase: "GLP Version 3 or any later version"

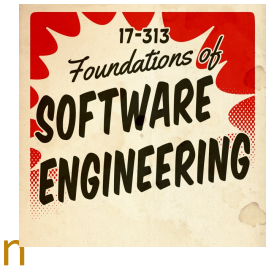
Why would projects choose one license over another?





[From http://choosealicense.com/licenses/]

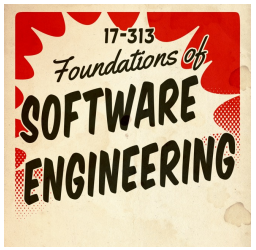
Dual License Business Model



- Released as GPL which requires a company using the open source product to open source it's application

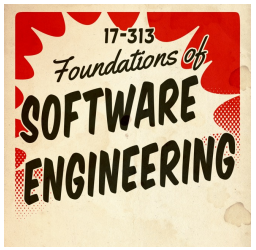
- Or companies can pay $2,000 to $10,000 annually to receive a copy of MySQL with a more business friendly license

Risk: Incompatible Licenses



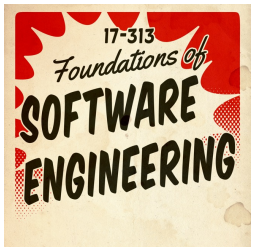
- Sun open sourced OpenOffice, but when Sun was acquired by Oracle, Oracle temporarily stopped the project.
- Many of the community contributors banded together and created LibreOffice
- Oracle eventually released OpenOffice to Apache
- LibreOffice changed the project license so LibreOffice can copy changes from OpenOffice but OpenOffice cannot do the same due to license conflicts

MIT License



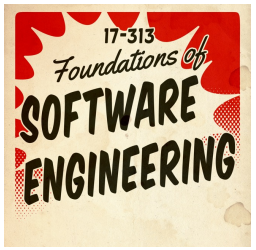
- Must retain copyright credit
- Software is provided as is
- Authors are not liable for software
- No other restrictions

LGPL



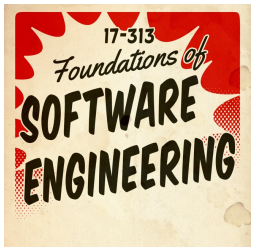
- Software must be a library
- Similar to GPL but no copyleft requirement

BSD License



- No liability and provided as is.
- Copyright statement must be included in source and binary
- The copyright holder does not endorse any extensions without explicit written consent

Apache License



- Apache
  - Similar to GPL with a few differences
  - Not copyleft
  - Not required to distribute source code
  - Does not grant permission to use project's trademark
  - Does not require modifications to use the same license

Perception:

- Anarchy
- Demagoguery
- Ideology
- Altruism
- Many eyes

DEPENDENCY MANAGEMENT & VERSIONING

Left-pad (March 22, 2016)

OBSESSIONS QUARTZ

THE VERG

NPM ERR!

How one programmer broke the internet by deleting a tiny piece of

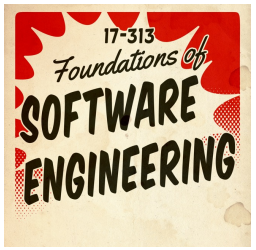
SIGN IN The Register

{* SOFTWARE *}

How one developer just broke Node, Babel and thousands of projects in 11 lines of JavaScript

Code pulled from NPM – which everyone was using

Left-pad (March 22, 2016)



npmjs.org tells me that left-pad is not available (404 page) #4

Closed **silkentrance** opened this issue on Mar 22, 2016 · 193 comments

silkentrance commented on Mar 22, 2016

When building projects on travis, or when searching for left-pad on npmjs.com, both will report that the package cannot be found.

Here is an excerpt from the travis build log

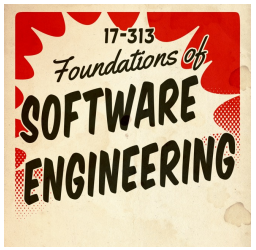
```
npm ERR! Linux 3.13.0-40-generic
npm ERR! argv "/home/travis/.nvm/versions/node/v4.2.2/bin/node" "/home/travis/.nvm/versions/node/v4.2.2/bin/npm
npm ERR! node v4.2.2
npm ERR! npm  v2.14.7
npm ERR! code E404
npm ERR! 404 Registry returned 404 for GET on https://registry.npmjs.org/left-pad
npm ERR! 404
npm ERR! 404 'left-pad' is not in the npm registry.
npm ERR! 404 You should bug the author to publish it (or use the name yourself!)
npm ERR! 404 It was specified as a dependency of 'line-numbers'
npm ERR! 404
npm ERR! 404 Note that you can also install from a
npm ERR! 404 tarball, folder, http url, or git url.
npm ERR! Please include the following file with any support request:
npm ERR!     /home/travis/build/coldrye-es/pingo/npm-debug.log
make: *** [deps] Error 1
```

And here is the standard npmjs.com error page https://www.npmjs.com/package/left-pad

However, if I remove left-pad from my local npm cache and then reinstall it using npm it will happily install left-pad@0.0.4.

👍 88 😕 3

Left-pad (Docs)



left-pad

String left pad

build unknown

Install

```
$ npm install left-pad
```

Usage

```
const leftPad = require('left-pad')

leftPad('foo', 5)
// => "  foo"

leftPad('foobar', 6)
// => "foobar"

leftPad(1, 2, '0')
// => "01"

leftPad(17, 5, 0)
// => "00017"
```

Install

```
> npm i left-pad
```

Repository
github.com/stevemao/left-pad

Homepage
github.com/stevemao/left-pad#readme

Weekly Downloads
2,962,641

Version	License
1.3.0	WTFPL

Unpacked Size	Total Files
9.75 kB	10

Issues	Pull Requests
3	7

Last publish
4 years ago

Left-pad (Source Code)

17 lines (11 sloc) | 222 Bytes

```
1   module.exports = leftpad;
2
3   function leftpad (str, len, ch) {
4     str = String(str);
5
6     var i = -1;
7
8     if (!ch && ch !== 0) ch = ' ';
9
10    len = len - str.length;
11
12    while (++i < len) {
13      str = ch + str;
14    }
15
16    return str;
17  }
```


See also: isArray

isarray

`Array#isArray` for older browsers and deprecated Node.js versions.

build passing downloads 227M/month

Just use Array.isArray directly, unless you need to support those older versions.

Usage

```
var isArray = require('isarray');

console.log(isArray([])); // => true
console.log(isArray({})); // => false
```

```
5 lines (4 sloc) | 133 Bytes

1  var toString = {}.toString;
2
3  module.exports = Array.isArray || function (arr) {
4    return toString.call(arr) === '[object Array]';
5  };
```

```
> npm i isarray
```

Repository
github.com/juliangruber/isarray

Homepage
github.com/juliangruber/isarray

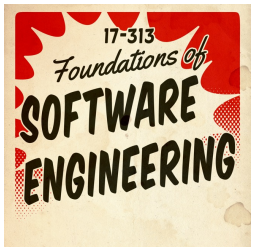
Weekly Downloads
50,913,317

Version	License
2.0.5	MIT

Unpacked Size	Total Files
3.43 kB	4

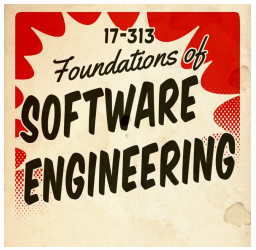
Issues	Pull Requests
4	3

Dependency Management



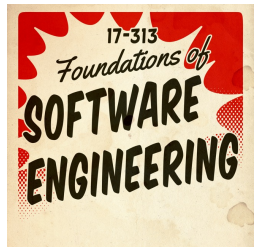
- It’s hard
- It’s mostly a mess (everywhere)
- But it’s critical to modern software development

What is a Dependency?



- Core of what most build systems do
  - "Compile" and "Run Tests" is just a fraction of their job
- Examples: Maven, Gradle, NPM, Bazel, …
- **Foo->Bar**: To build Foo, you may need to have a built version of Bar
- Dependency Scopes:
  - **Compile**: Foo uses classes, functions, etc. defined by Bar
  - **Runtime**: Foo uses an abstract API whose implementation is provided by Bar (e.g. logging, database, network or other I/O)
  - **Test**: Foo needs Bar only for tests (e.g. JUnit, mocks)
- Internal vs. External Dependencies
  - Is Bar also built/maintained by your org or is it pulled from elsewhere using a package manager?

Dependencies: Example



master ▾ M

Roberto Rosa

1 contributor

45 lines (45 sloc

```
1  Pillow==8.3
2  PyPDF2==1.2
3  PyYAML==5.4
4  Whoosh==2.7
5  bleach==4.0
6  celery==5.1
7  django-acti
8  django-cele
9  django-colo
10 django-cors
11 django-form
12 django-math
13 django-mode
14 django-mptt
15 django-pure
16 django-qsst
17 django-solo
18 django-stro
19 django-widg
20 djangorestf
21 djangorestf
22 drf-yasg==1
23 extract-msg
24 flanker==0.
25 flex==6.14.
26 furl==2.1.2
27 fusepy==3.0
28 gevent==21.
29 graphviz==0
30 gunicorn==2
31 jsonschema=
32 mock==4.0.3
33 node-semver==0.8.0
34 packaging==21.0
```

Package: git (1:2.17.1-1ubuntu0.9 and others) [**security**]

fast, scalable, distributed revision control system

Other Packages Related to git

● depends ◆ recommends ■ suggests • enhances

- ● git-man (<< 1:2.17.0-.) [not amd64, i386]
  fast, scalable, distributed revision control system (manual pages)
- git-man (<< 1:2.17.1-.) [amd64, i386]
- git-man (>> 1:2.17.0) [not amd64, i386]
- git-man (>> 1:2.17.1) [amd64, i386]
- ● libc6 (>= 2.16) [not arm64, ppc64el]
  GNU C Library: Shared libraries
  also a virtual package provided by libc6-udeb
- libc6 (>= 2.17) [arm64, ppc64el]
- ● libcurl3-gnutls (>= 7.16.2)
  easy-to-use client-side URL transfer library (GnuTLS flavour)
- ● liberror-perl
  Perl module for error/exception handling in an OO-ish way
- ● libexpat1 (>= 2.0.1)
  XML parsing C library - runtime library
- ● libpcre3
  Old Perl 5 Compatible Regular Expression Library - runtime files
- ● perl
  Larry Wall's Practical Extraction and Report Language
- ● zlib1g (>= 1:1.2.0)
  compression library - runtime
- ◆ less
  pager program similar to more
- ◆ patch
  Apply a diff file to an original
- ◆ ssh-client
  virtual package provided by openssh-client

Links for git

Ubuntu Resources:

- Bug Reports
- Ubuntu Changelog
- Copyright File

Download Source Package git:

- [git_2.17.1-1ubuntu0.9.dsc]
- [git_2.17.1.orig.tar.xz]
- [git_2.17.1-1ubuntu0.9.debian.tar.xz]

Maintainer:

- Ubuntu Developers (Mail Archive)

Please consider filing a bug or asking a question v
Launchpad before contacting the maintainer directly.

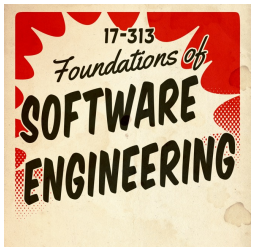
Original Maintainers (usually fror Debian):

- Gerrit Pape
- Jonathan Nieder
- Anders Kaseorg

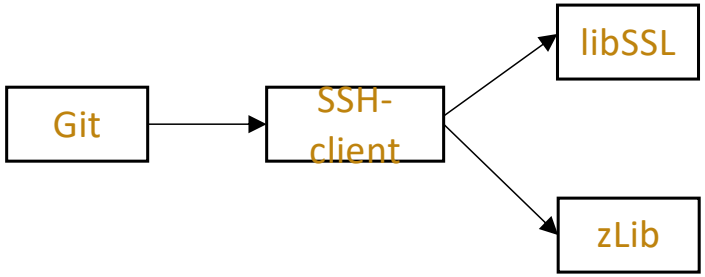
It should generally not be necessary for users to contact the original maintainer.

External Resources:

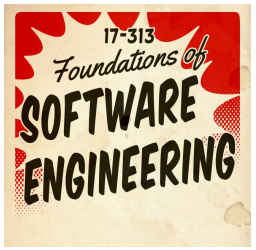
Transitive Dependencies



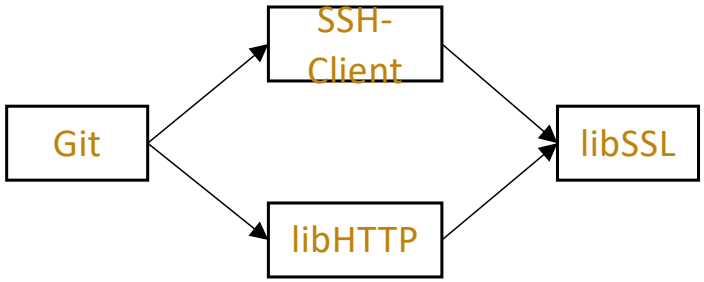
- Should Git be able to use exports of libSSL (e.g. certificate management) or zLib (e.g. gzip compression)?



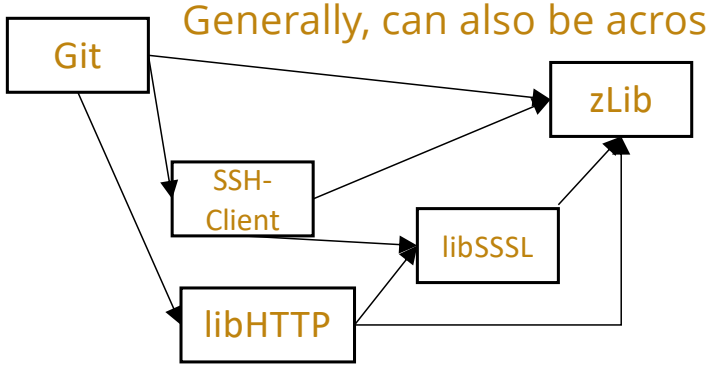
Diamond Dependencies



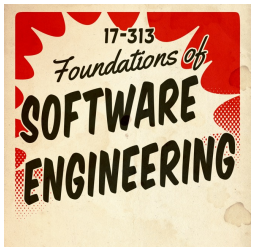
- What are some problems when multiple intermediate dependencies have the same transitive dependency?



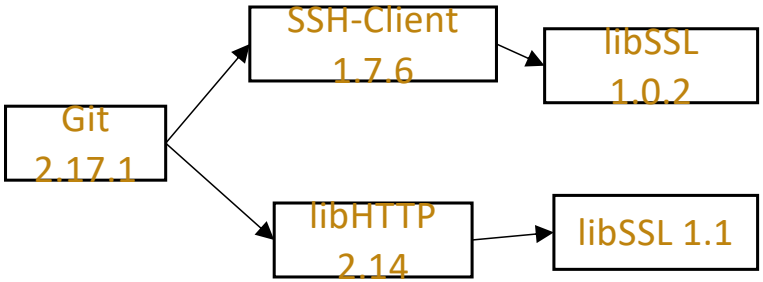
Generally, can also be across levels



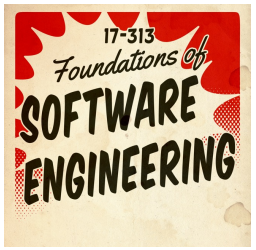
Diamond Dependencies



- What are some problems when multiple intermediate dependencies have the same transitive dependency?

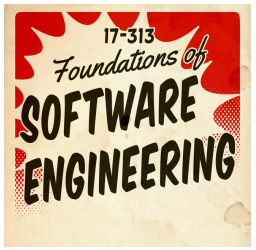


Resolutions to the Diamond Problem



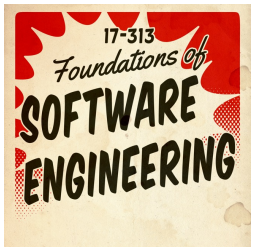
1. Duplicate it!
   - Doesn't work with static linking (e.g. C/C++), but may be doable with Java (e.g. using ClassLoader hacking or package renaming)
   - Values of types defined by duplicated libraries cannot be exchanged across
2. Ban transitive dependencies; just use a global list with one version for each
   - Challenge: Keeping things in sync with latest
   - Challenge: Deciding which version of transitive deps to keep
3. Newest version (keep everything at latest)
   - Requires ordering semantics
   - Intermediate dependency may break with update to transitive
4. Oldest version (lowest denominator)
   - Also requires ordering semantics
   - Sacrifices new functionality
5. Oldest non-breaking version / Newest non-breaking version
   - Requires faith in tests or semantic versioning contract

Semantic Versioning



- Widely used convention for versioning releases
  - E.g. 1.2.1, 3.1.0-alpha-1, 3.1.0-alpha-2, 3.1.0-beta-1, 3.1.0-rc1
- Format: {MAJOR} . {MINOR} . {PATCH}
- Each component is ordered (numerically, then lexicographically; release-aware)
  - 1.2.1 < 1.10.1
  - 3.1.0-alpha-1 < 3.1.0-alpha-2 < 3.1.0-beta-1 < 3.1.0-rc1 < 3.1.0
- Contracts:
  - MAJOR updated to indicate breaking changes
    - Same MAJOR version => backward compatibility
  - MINOR updated for additive changes
    - Same MINOR version => API compatibility (important for linking)
  - PATCH updates functionality without new API
    - Ninja edit; usually for bug fixes

https://semver.org/



2.0.0 2.0.0-rc.2 2.0.0-rc.1 1.0.0 1.0.0-beta

Semantic Versioning 2.0.0

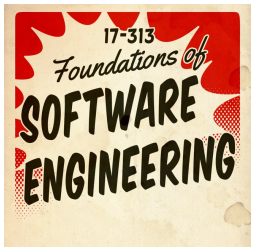
Summary

Given a version number MAJOR.MINOR.PATCH, increment the:

1. MAJOR version when you make incompatible API changes,
2. MINOR version when you add functionality in a backwards compatible manner, and
3. PATCH version when you make backwards compatible bug fixes.

Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

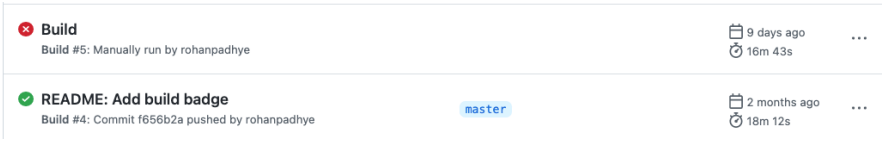
Dependency Constraints

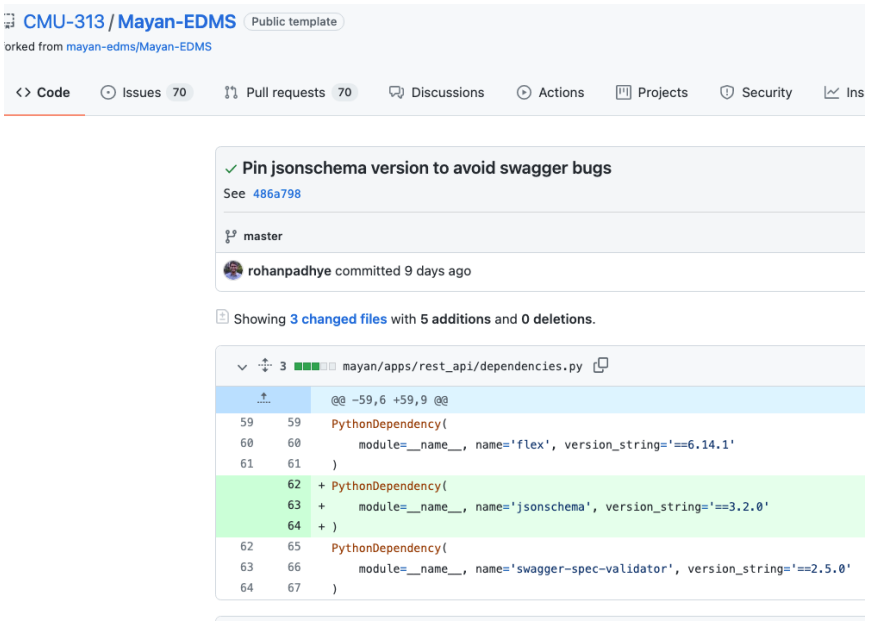


- E.g. Declare dependency on ”Bar > 2.1”
  - Bar 2.1.0, 2.1.1, 2.2.0, 2.9.0, etc. all match
  - 2.0.x does NOT match
  - 3.0.x does NOT match
- Diamond dependency problem can be resolved using SAT solvers
  - E.g. Foo 1.0.0 depends on “Bar >= 2.1” and “Baz 1.8.x”
    - Bar 2.1.0 depends on “Qux [1.6, 1.7]”
    - Bar 2.1.1 depends on “Qux 1.7.0”
    - Baz 1.8.0 depends on “Qux 1.5.x”
    - Baz 1.8.1 depends on “Qux 1.6.x”
  - Find an assignment such that all dependencies are satisfied
    - Solution: Use Bar 2.1.0, Baz 1.8.1, and Qux 1.6.{latest}

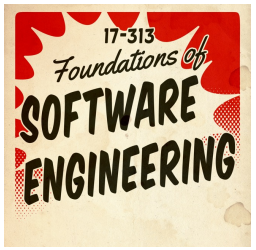
Semantic Versioning Contracts

- Largely trusting developers to maintain them
- Constrained/range dependencies can cause unexpected build failures
- Automatic validation of SemVer is hard

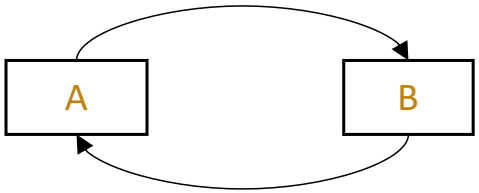




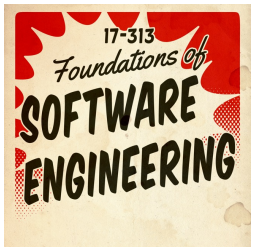
Cyclic Dependencies



- A very bad thing
- Avoid at all costs
- Sometimes unavoidable or intentional
  - E.g. GCC is written in C (needs a C compiler)
  - E.g. Apache Maven uses the Maven build system
  - E.g. JDK tested using JUnit, which requires the JDK to compile

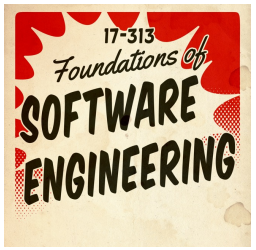


Cyclic Dependencies



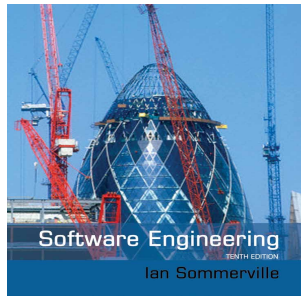
- Bootstrapping: Break cycles over time
- Assume older version exists in binary (pre-built form)
- Step 1: Build A using an older version of B
- Step 2: Build B using new (just built) version of A
- Step 3: Rebuild A using new (just built) version of B
- Now, both A and B have been built with new versions of their dependencies
- Doesn't work if both A and B need new features of each other at the same time (otherwise Step 1 won't work)
  - Assumes incremental dependence on new features
- How was the old version built in the first place? (it's turtles all the way down)
  - Assumption: cycles did not exist in the past
  - Successfully applied in compilers (e.g. GCC is written in C)

Dependency Reliability



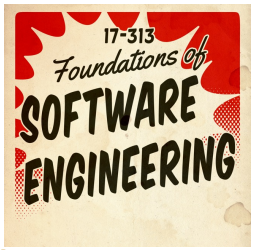
- Availability
  - Remember left-pad?
  - Many orgs will mirror package repositories
- Security
  - Will you let strangers execute arbitrary code on your laptop?
  - Think about this every time you do “pip install” or “npm install” or “apt-get updgrade” or “brew updgrade” or whatever (esp. with sudo)
    - Scary, right? Who are you trusting? Why?
  - Typo squatting
    - “pip install numpi”

Key points



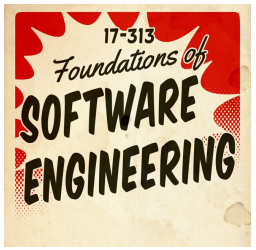
- When developing software, you should always consider the possibility of reusing existing software, either as components, services or complete systems.
- Configuration management is the process of managing changes to an evolving software system. It is essential when a team of people are cooperating to develop software.
- Most software development is host-target development. You use an IDE on a host machine to develop the software, which is transferred to a target machine for execution.
- Open source development involves making the source code of a system publicly available. This means that many people can propose changes and improvements to the software.

Key Points



- Understand the terminology “free software” and explain open source culture and principles.
- Express an educated opinion on the philosophical/political debate between open source and proprietary principles.

Key Points



- Dependency management is hard.