Technical lag for software deployments

Jesus M. Gonzalez-Barahona

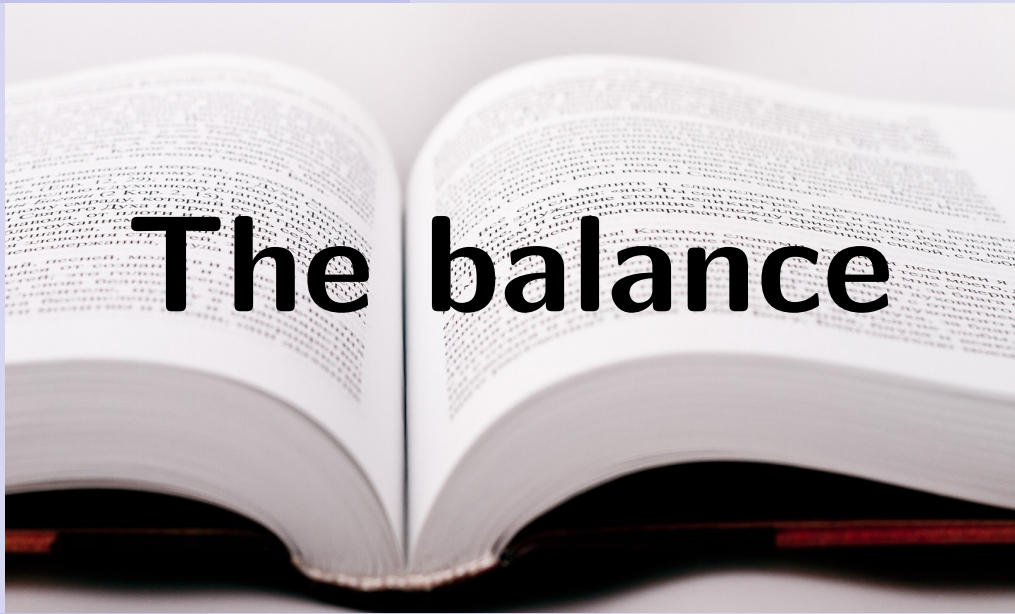
Universidad Rey Juan Carlos
@jgbarah http://github.com/jgbarah/presentations

Seminar at IMDEA Software
Madrid (Spain), October 2nd 2018

"If I go there will be trouble
And if I stay it will be double
So come on and let me know"

Should I Stay Or Should I Go?
The Clash

`https://www.youtube.com/watch?v=BN1WwmEDWAM`



Deployments

Any deployment
is the real world instance
of an “ideal” target

Deployments: the balance

"If it works, don't touch it"

vs.

"The quest for the ideal"

Deployments: example

You want the latest functionality
so you deploy it
but the day after
it is no longer the latest

Should you update?

Living the risky life

```
$ sudo apt-get dist-upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
1249 upgraded, 206 newly installed, 8 to remove and 3 not upgraded.
Need to get 2,856 MB of archives.
After this operation, 340 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Upgrading in Debian/testing

Dependencies

You want the latest functionality
so you deploy it
but dependencies may prevent you
from having the latest

Should dependencies be updated?

Living in the past

```
"dependencies": {
  "coffeescript": "~1.10.0",
  "dateformat": "~1.0.12",
  "eventemitter2": "~0.4.13",
  "exit": "~0.1.1",
  "findup-sync": "~0.3.0",
  ...
},
```

Oct. 2018: Grunt master / coffescript

Releases

Technical lag

For a release:

"difference between the deployed release and the ideal release"

- What is "ideal release"?
- How we measure difference between releases?

Ideal release (examples)

Most recent

Most recent in the stable line

Less open bugs

Less unfixed vulnerabilities

Difference (examples)

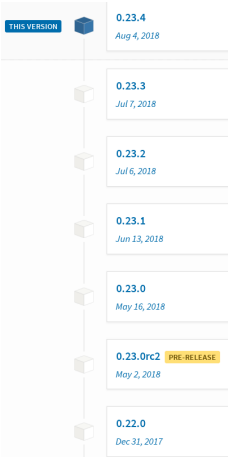
Difference in release time
Difference in version number
Number of commits
Difference in number of open bugs
Estimated effort

- ideal: $P \times Repos \rightarrow R$
  Given $p \in P$, $repo \in Repos$, $ideal(p, repo)$
- diff: $R \times R \times Repos \rightarrow L$
  Given $repo \in Repos$ and $r, s \in repo$,
  $diff(r, s, repo)$, if $package(r) = package(s)$
- techlag: $R \times Repos \rightarrow L$
  $\forall repo \in Repos, \forall r \in repo$:
  $techlag(r, repo) = diff(r, ideal(r, repo), repo)$

Example

Package: Pandas
Deployed: 0.22.0
Ideal: 0.23.4

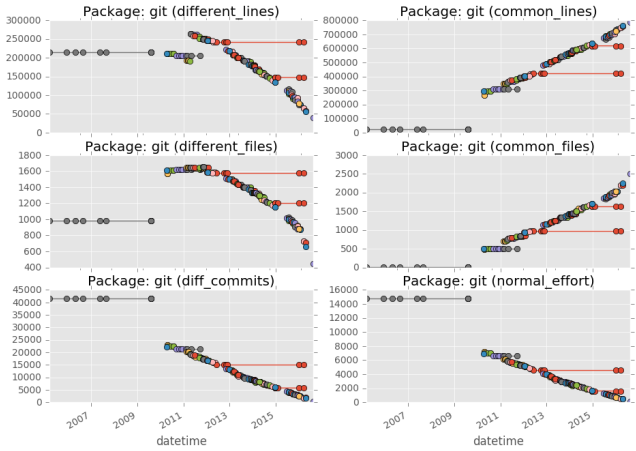
Lag (releases): 6 releases
Lag (reltime): 8 months, 4 days



Example

Debian releases for git

(source code & commits diffs)



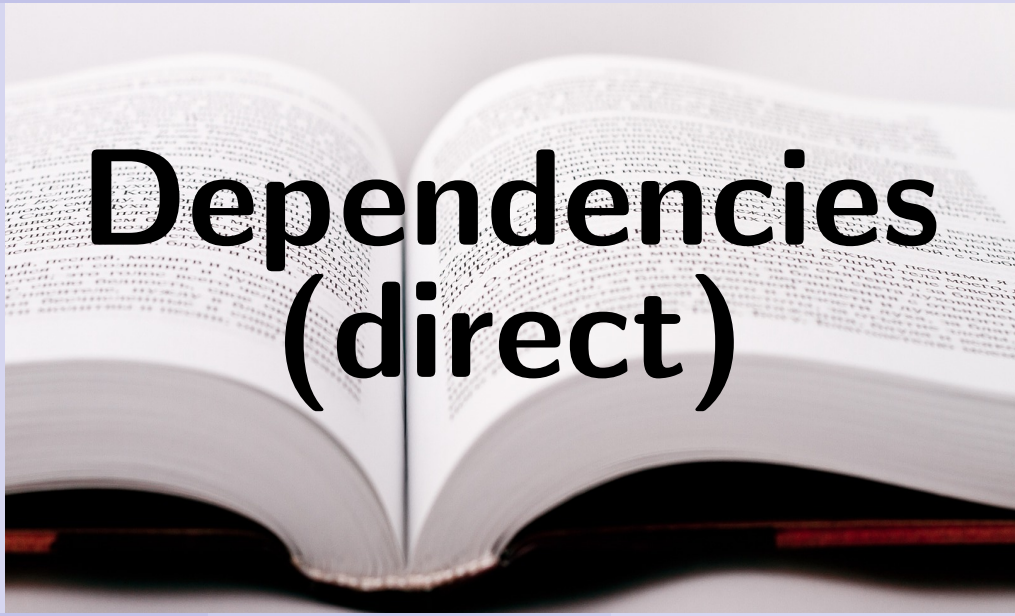
Technical lag

For a collection of releases:

"aggregation of the lag
for each release in the collection"

- How do we aggregate?
- Examples: maximum, summation, mean

- techlag: $\mathcal{P}(R) \times Repos \rightarrow L$
- Given $rcoll \in \mathcal{P}(R), repo \in Repos$, $techlag_{max}(rcoll, repo) = max_{r \in rcoll}(techlag(r, repo)$
- Given $rcoll \in \mathcal{P}(R), repo \in Repos$, $techlag_{add}(rcoll, repo) = \sum_{r \in rcoll} techlag(r, repo)$



Dependencies (direct)

Technical lag

For direct dependencies of a release:

"technical lag
for the collection formed by
direct dependencies of the release"

- Having constraints into account
- Selecting as the package manager does

- $dep : R \rightarrow \mathcal{P}(P)$
- $allowed : R \times P \times Repos \rightarrow \mathcal{P}(R)$
  $allowed(r, p, repo) = rcol$, where $rcol \subset repo$.
- $selectver : \mathcal{P}(R) \rightarrow R$
- $deploy : R \times Repos \rightarrow \mathcal{P}(R)$
  Given $repo \in Repos, r \in repo$,
  $deploy(r, repo) =$
  $\{selectver(allowed(r, p_i, repo)), \forall p_i \in dep(r)\}$
- $deplag : R \times Repos \rightarrow L$:
  $deplag(r, repo) = techlag(deploy(r, repo))$

Dependencies (all)

Technical lag

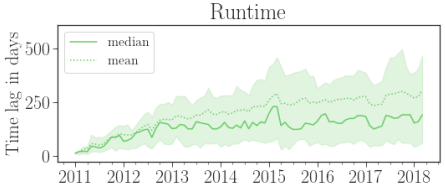
For all dependencies of a release:

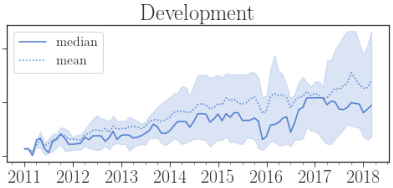
"technical lag
for the collection formed by
all (transitive) dependencies of the release"

- Having constraints into account
- Selecting as the package manager does

- $deploy^+ : R \times Repos \rightarrow \mathcal{P}(R)$
- Given $repo \in Repos, r \in repo$,
  $deploy^+(r, repo)$ as the minimal fix point such that:
  $deploy^+(r, repo) \supseteq deploy(r, repo)$
  $deploy^+(r, repo) \supseteq deploy(r', repo) \forall r' \in deploy^+(r, repo)$
- $deplag^+ : R \times Repos \rightarrow L$:
  $deplag^+(r, repo) = techlag(deploy^+(r, repo))$

Example





npm releases

release time lag, direct dependencies

Discussion

Uses

Technical lag of:

- deployed distributions
- container images
- deployed applications
- embedded systems

Uses

Who can control technical lag:

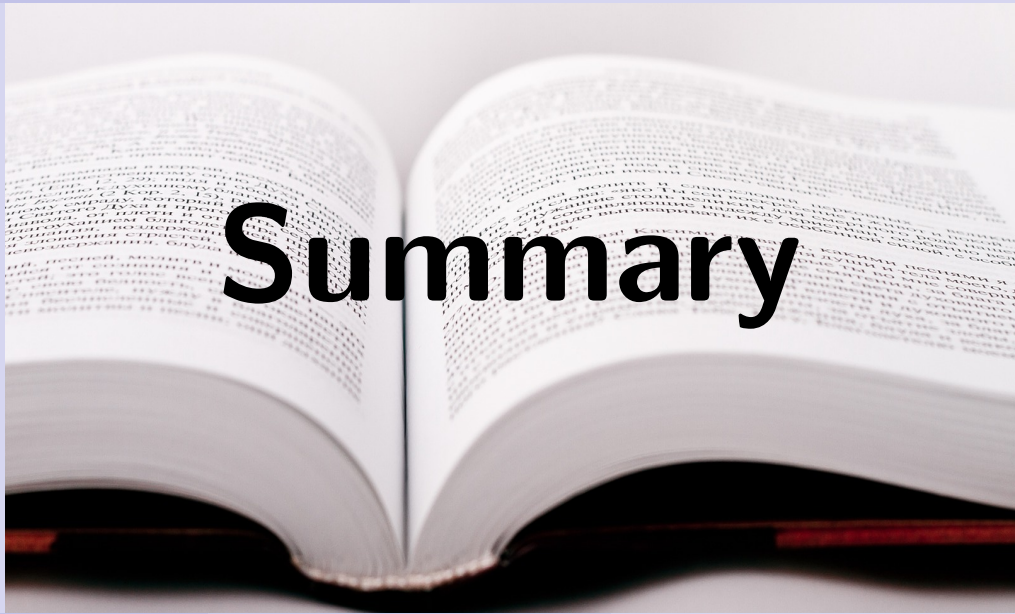
- deployers: “top level” releases
- developers: direct dependencies
- ecosystems: typical dependencies

Types

Ideal: latest, most stable, more secure, less buggy...
Difference:

- Release metadata: versions, release time...
- Source code: diff lines, diff files
- SCM: commits, normalized effort
- ITS: bugs fixed, vulnerabilities fixed, feature requests closed

Aggregations: maximum, summation, mean, median



Summary

Difference between real and ideal

What am I missing if I upgrade?

Dependencies impact on lag

More info...

Ahmed Zerouali, Eleni Constantinou, Tom Mens, Gregorio Robles, Jesús M. González-Barahona:
"An Empirical Analysis of Technical Lag in npm Package Dependencies"
ICSR 2018: 95-110

Jesús M. González-Barahona, Paul Sherwood, Gregorio Robles, Daniel Izquierdo-Cortazar:
"Technical Lag in Software Compilations: Measuring How Outdated a Software Deployment Is"
OSS 2017: 182-192