5th Asian Workshop of Advanced Software Engineering (AWASE2016)

Nara, Japan, 19-20 March, 2016

Modeling and Visualizing Library Dependency Updates

Raula Gaikovina Kula

Joint work:
Osaka University
University of Victoria, Canada
Vrije Universiteit Brussel, Belgium

http://sel.ist.osaka-u.ac.jp/SARF/index.html.en

1. Introduction - A story on Software Reuse, APIs and Library Dependencies
2. Problem Statement – The need to model and visualize Library Dependency and Updates
3. Results - Two published works of our model
4. Future Works – The story continues…

Software Reuse , APIs and Library Dependencies

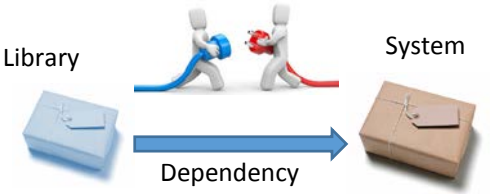
there was once a set of useful functions…

How can I access all these useful functions without reinventing each time

A library was born

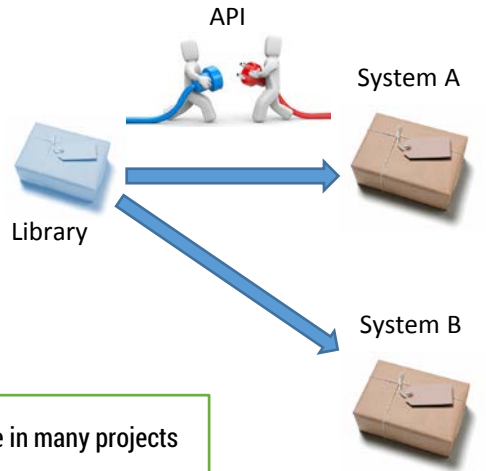
Software Reuse , APIs and Library Dependencies

Application Programming Interface (API)

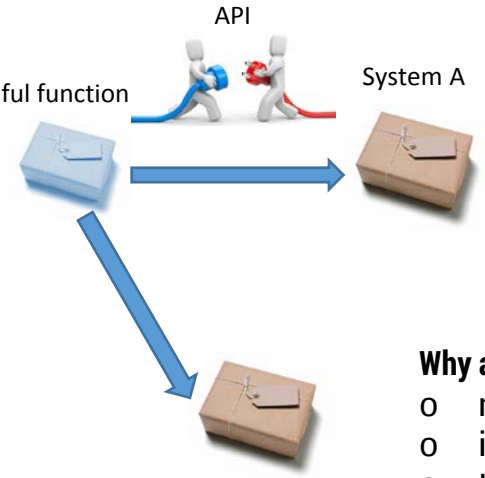


Software Reuse , APIs and Library Dependencies

Yes, I am able to reuse in many projects



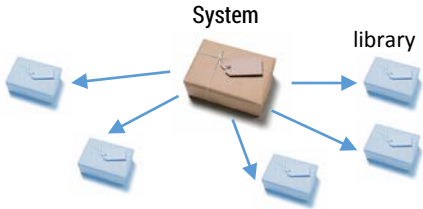
Software Reuse , APIs and Library Dependencies



Why adopt libraries?

- needed features
- inherited quality
- time/effort cost efficient
- avoid reinvent wheel

Software Systems and Library Dependencies

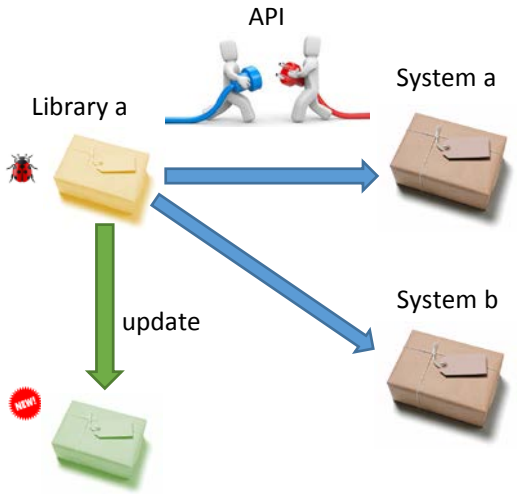


I can make more efficient systems with inherited quality

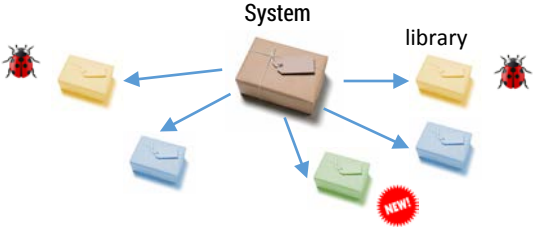
Software Reuse , APIs and Library Dependencies

Bug
Vulnerability

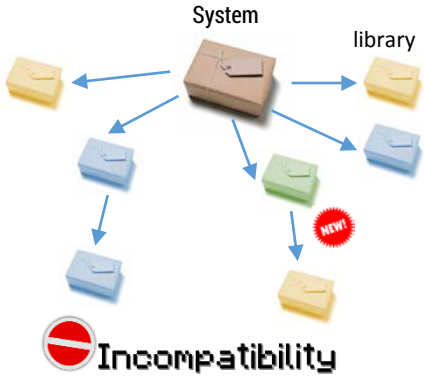
New Features

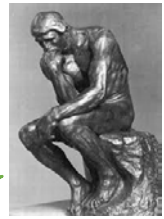


Evolving Software Systems and their Library Dependencies



An example of Dependency Hell[1] within the Web of Library Dependencies





System Maintainer needs to decide `if', `when' and `what to update?'

[1] M. Jang, "Linux annoyances for geeks," 2009.

Rise of online library super-repositories

Maven Central now has over

With the rise of massive online super-repositories

The Central Repository

npm

CRAN

The Comprehensive R Archive Network

RubyGems.org
your community gem host

452,645,894 downloads
of 33,415 gems cut since July 2009

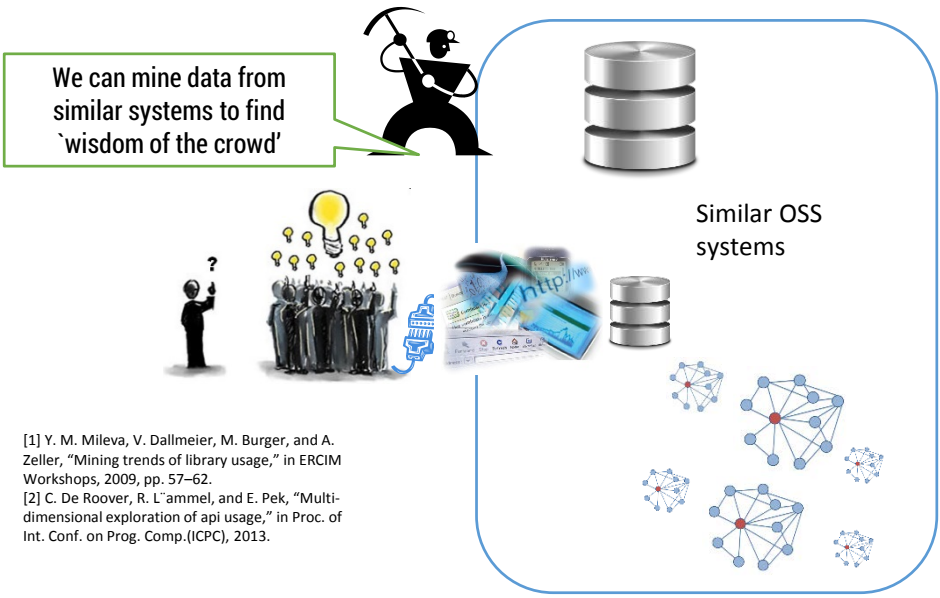
CPAN
Perl Modules

Motivation

- How can we leverage the wisdom of the crowd, to systematically mine and extract the evolution of both systems and library dependencies in these super-repositories?

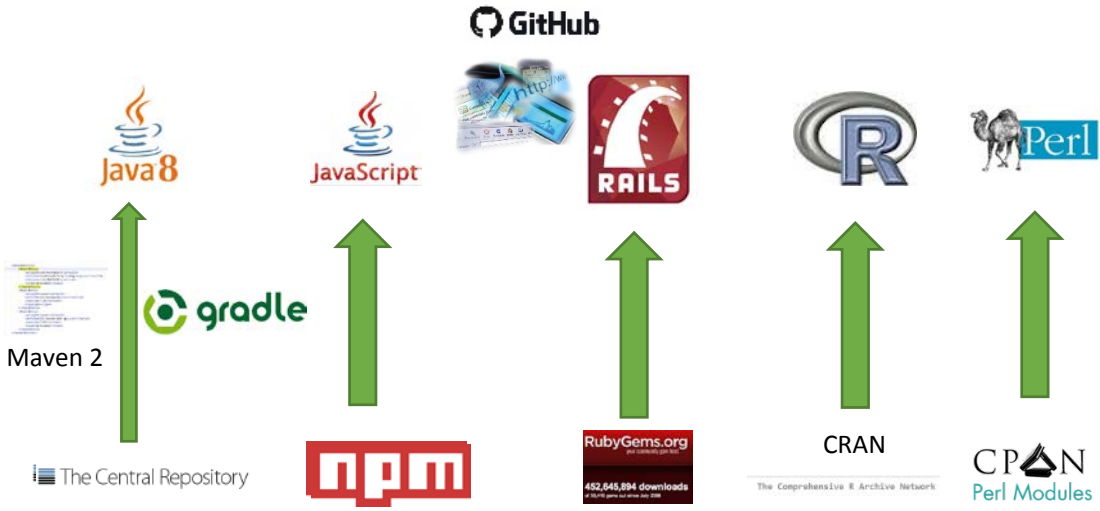
System Maintainer needs to decide `if', `when' and `what to update?'

Propose: Mining Library Usage Trends[1,2]

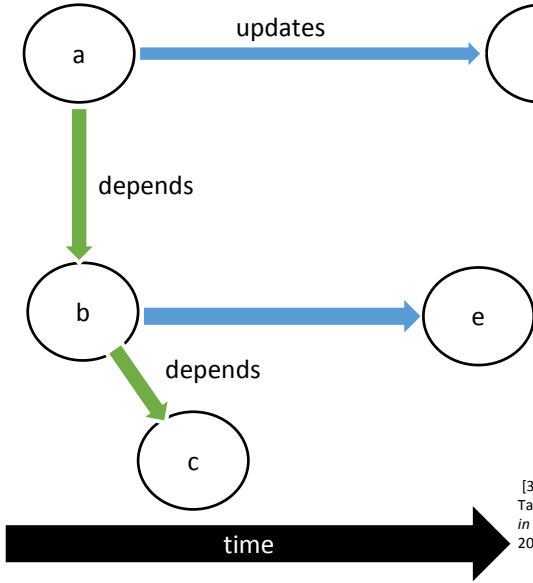


[1] Y. M. Mileva, V. Dallmeier, M. Burger, and A. Zeller, "Mining trends of library usage," in ERCIM Workshops, 2009, pp. 57–62.
[2] C. De Roover, R. L¨ammel, and E. Pek, "Multi-dimensional exploration of api usage," in Proc. of Int. Conf. on Prog. Comp.(ICPC), 2013.

Different Universes, similar super-repositories



Model of Dependency and Update Relations
SUG (Software Universe Graph)[3]



a depends on b
b depends on c
d is a updated version of a
e is an updated version of b

[3] Raula Gaikovina Kula and Coen De Roover and Daniel M. German and Takashi Ishio and Katsuro Inoue, "*Modeling Library Dependencies and Updates in Large Super Repository Universes*", Osaka University Technical Report, 2015/10/29

Two different works based on the SUG

Case Studies of Github java projects that depend on maven libraries

The Central Repository

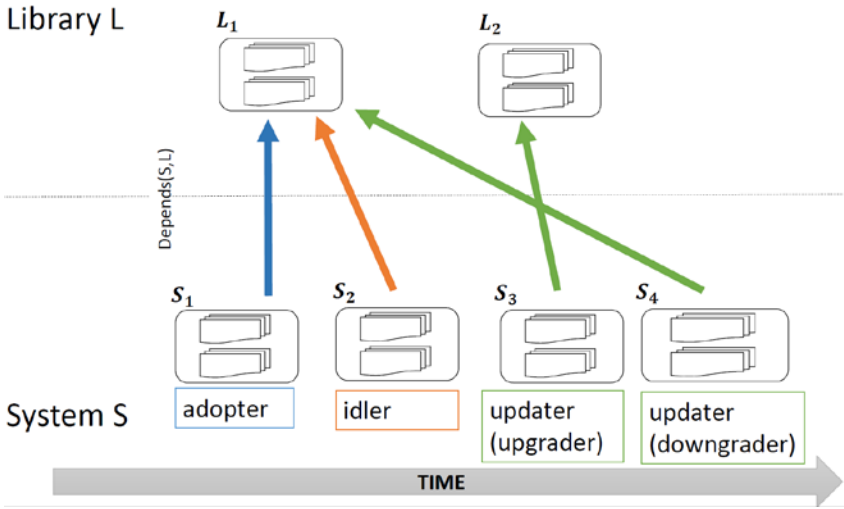
GitHub

VISSOFT 2014

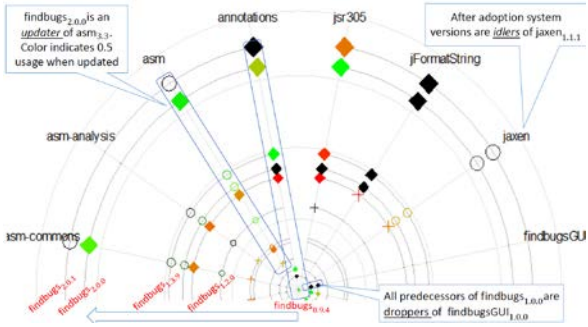
Visualizing the Evolution of Systems and their Library Dependencies

Raula Gaikovina Kula*, Coen De Roover*†, Daniel German*‡, Takashi Ishio*, Katsuro Inoue*
* Osaka University, Osaka, Japan † Vrije Universiteit Brussel, Brussels, Belgium
‡ University of Victoria, Canada
Email: {raula, coen, cderoove, ishio, inoue}@ist.osaka-u.ac.jp
dmg@uvic.ca

SUG (Software Universe Graph)

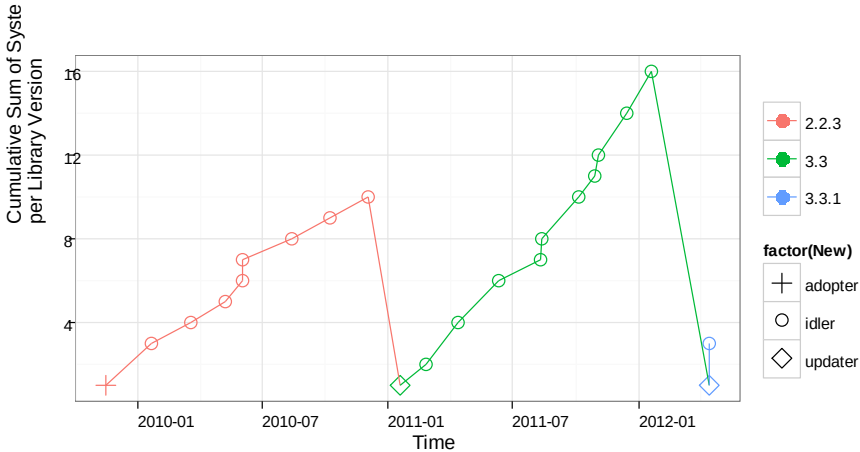


System Viewpoint

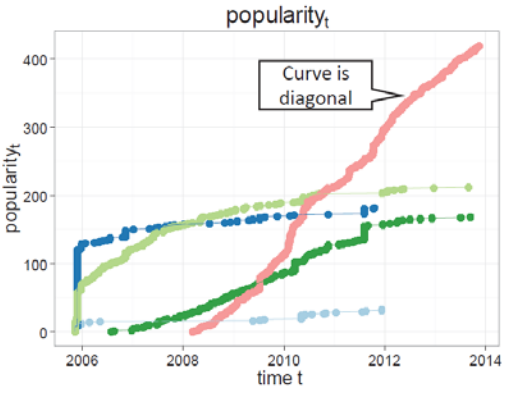


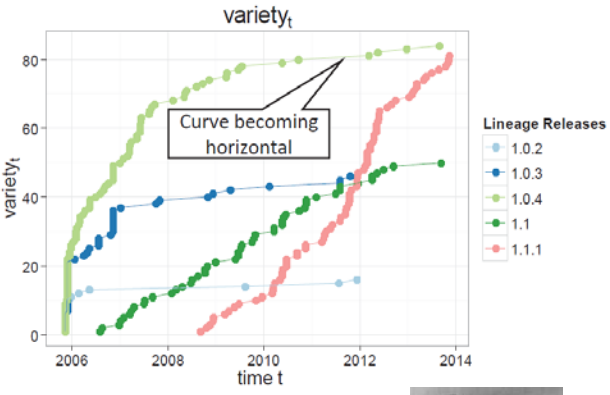
Library Viewpoint

Dependents Diffusion Plot (DDP)



Library Viewpoint





May indicate time to upgrade

Use-case Scenarios

- Towards the effective reuse of software libraries.
- System and Library Centric Views.

- 4 case scenarios with real world examples.
  - Regularity of Updates
  - Structural Dependency changes
  - Attractiveness of different Library Versions
  - Update Opportunities Current State

Check out the use-case scenario in the paper

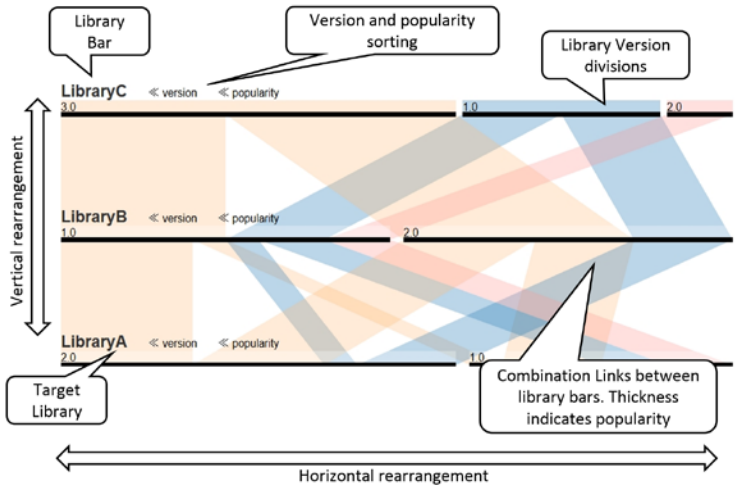
ICPC 2015 **VerXCombo**

VerXCombo: An interactive data visualization of popular library version combinations

Yuki Yano, Raula Gaikovina Kula, Takashi Ishio, Katsuro Inoue
Osaka University, Japan
{y-yano, raula-k, ishio, inoue}@ist.osaka-u.ac.jp

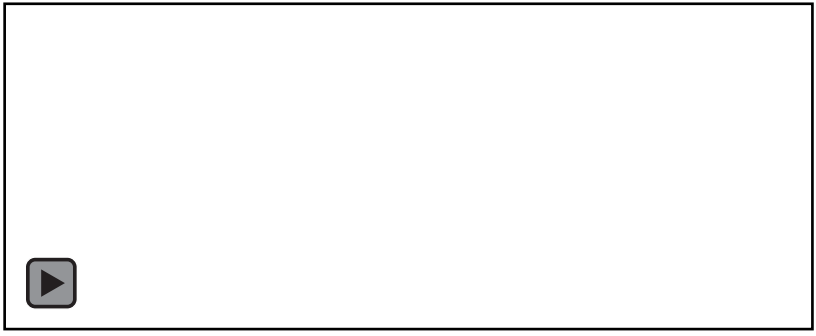
ICPC Tool Demonstration: Best Tool Award

Finding the best combination of dependencies to update



Interactive Features

VerXCombo



- ✓ Library Selection
  - o Autofill lookup interested libraries
- ✓ Interactive Manipulation
  - o Mouse over highlighting
  - o a combination link.
- ✓ Vertical Rearrangement
  - o Reorder Libraries for direct comparison
- ✓ Horizontal Rearrangement
  - o Reorder Library Version to isolate interested combinations
- ✓ Sorting by Popular Usage
  - o Thickness indicates popular versions. Most popular on left hand side
- ✓ Sorting by Version
  - o Latest Release will appear on most right hand side

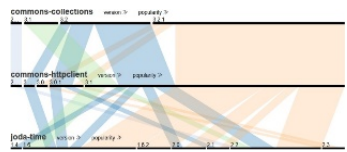
Use-case Scenarios

- Based on popularity and the latest releases, VerXCombo allows users to find the 'best fit' combination of libraries

    1. Introducing a new library
    2. Updating existing libraries

Check out the use-case scenario in the paper

VerXCombo



Back to the Motivation

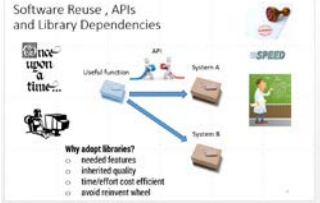
- How can we leverage the wisdom of the crowd, to systematically mine and extract the evolution of both systems and library dependencies in these super-repositories?
- We model an SUG as a graph of depends and update edges.
- We show various visualizations of how the SUG can be used to show:
  - Opportunities to update
  - Combinations of different libraries used by similar systems

The story continues…

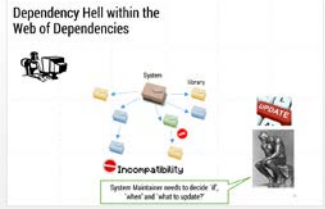
- API Usage and Library Updates
- Library Recommendations
- Disruptive Factors [4]:
  - Vulnerabilities
  - Breakages and API Incompatibilities
  - Competitors within the same domain
  - Cost benefit analysis of migrations

[3] Raula Gaikovina Kula and Daniel M. German and Takashi Ishio and Katsuro Inoue. Trusting a Library: A Study of the Latency to Adopt the Latest Maven Release. 22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering, SANER 2015, Montreal, Canada, March 2-6, 2015,

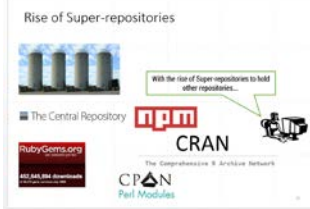
Software Reuse , APIs and Library Dependencies



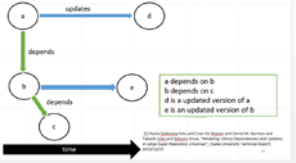
Dependency Hell within the Web of Dependencies



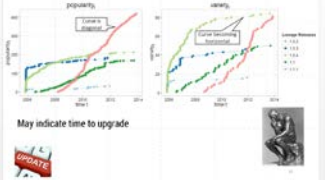
Rise of Super-repositories



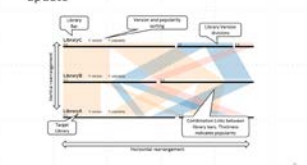
Model of Dependency and Update Relations SUG (Software Universe Graph)



Library Viewpoint



Finding the best combination of libraries to update



Future Work

- API Usage and Library Updates
- Library Recommendations
- Disruptive Factors:
  - Vulnerabilities
  - Breakages and API Incompatibilities
  - Competitors within the same domain
  - Cost benefit analysis of migrations