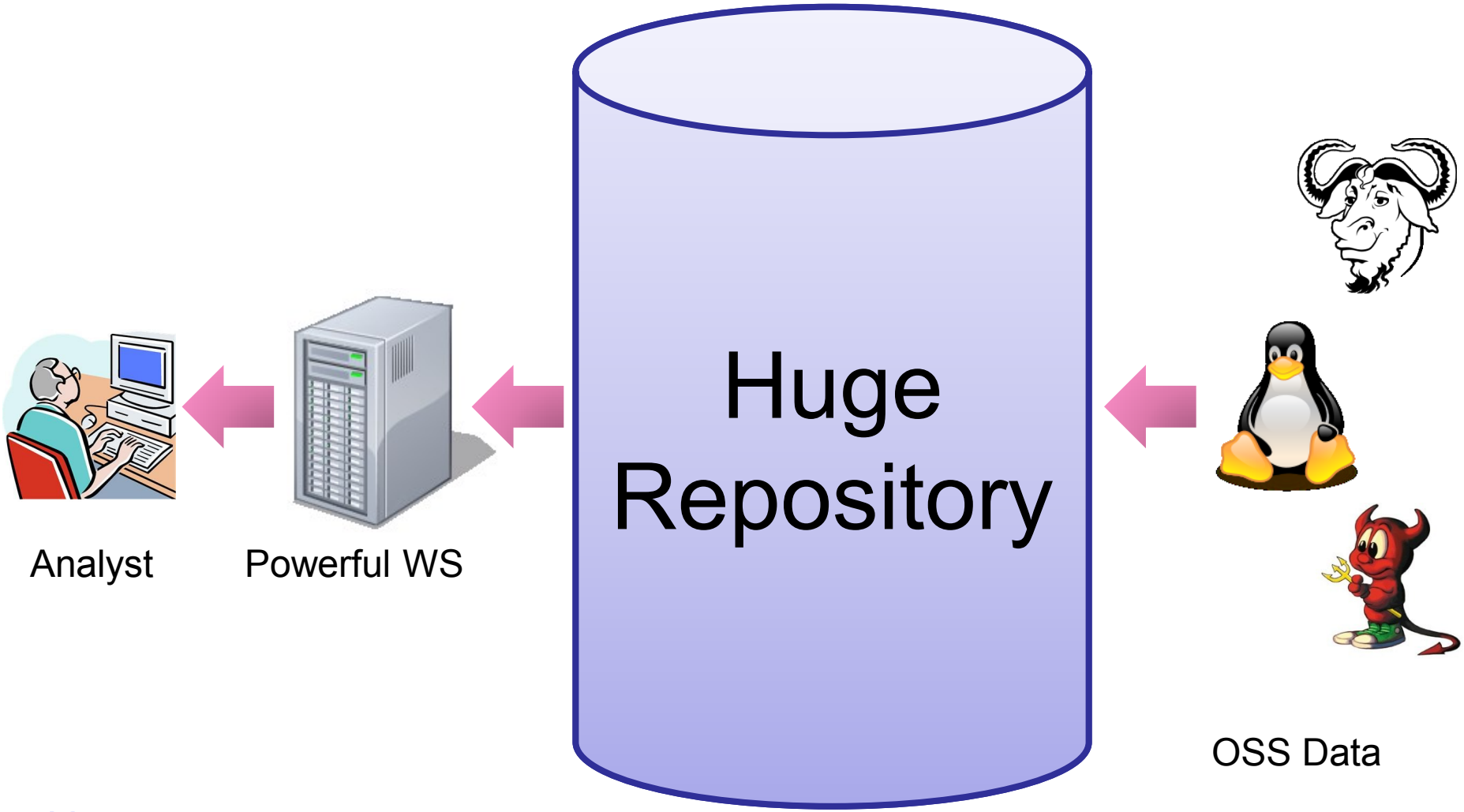
Mining Resources in the Internet Space

Katsuro Inoue

Osaka Univeristy

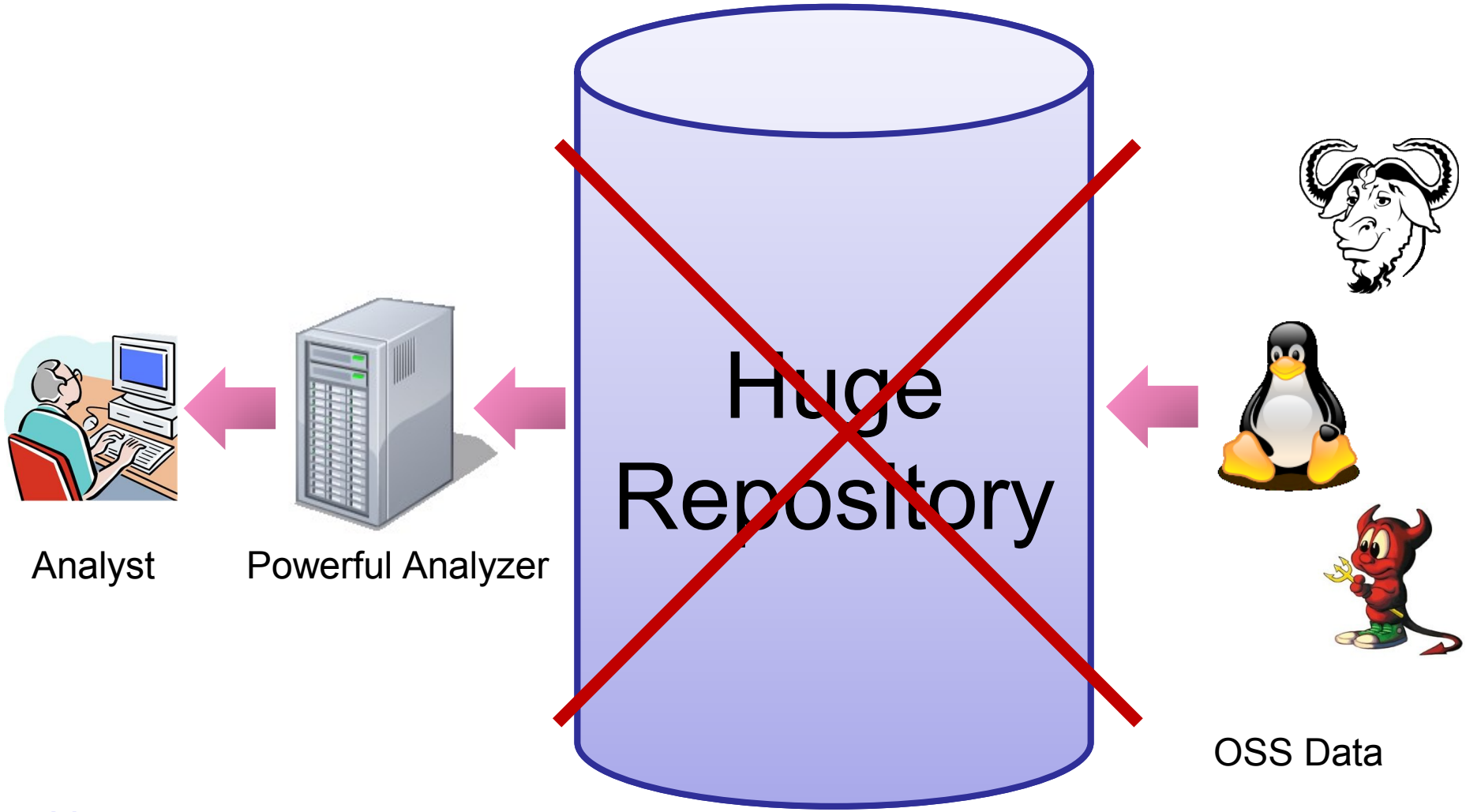
Mining OSS Data



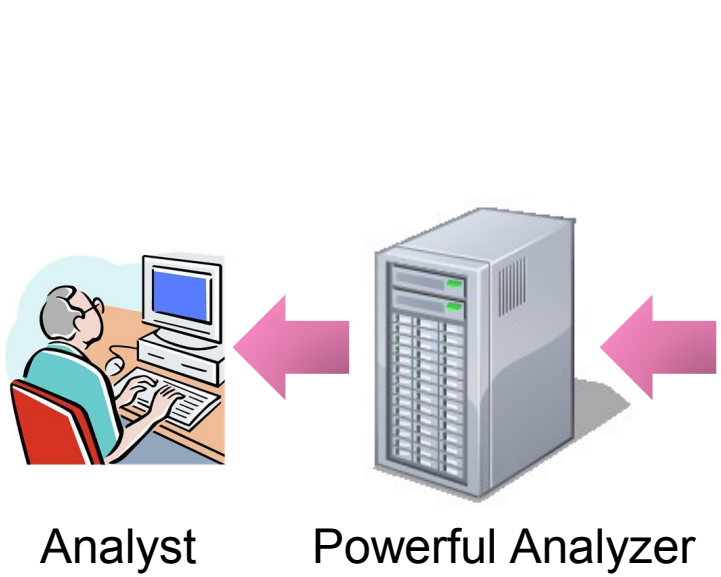
Maintaining Huge Repository

- Keeping the repository updated is important
  - Time consuming effort
  - Automatic crawling is not easy

Mining OSS Data



Mining OSS Data

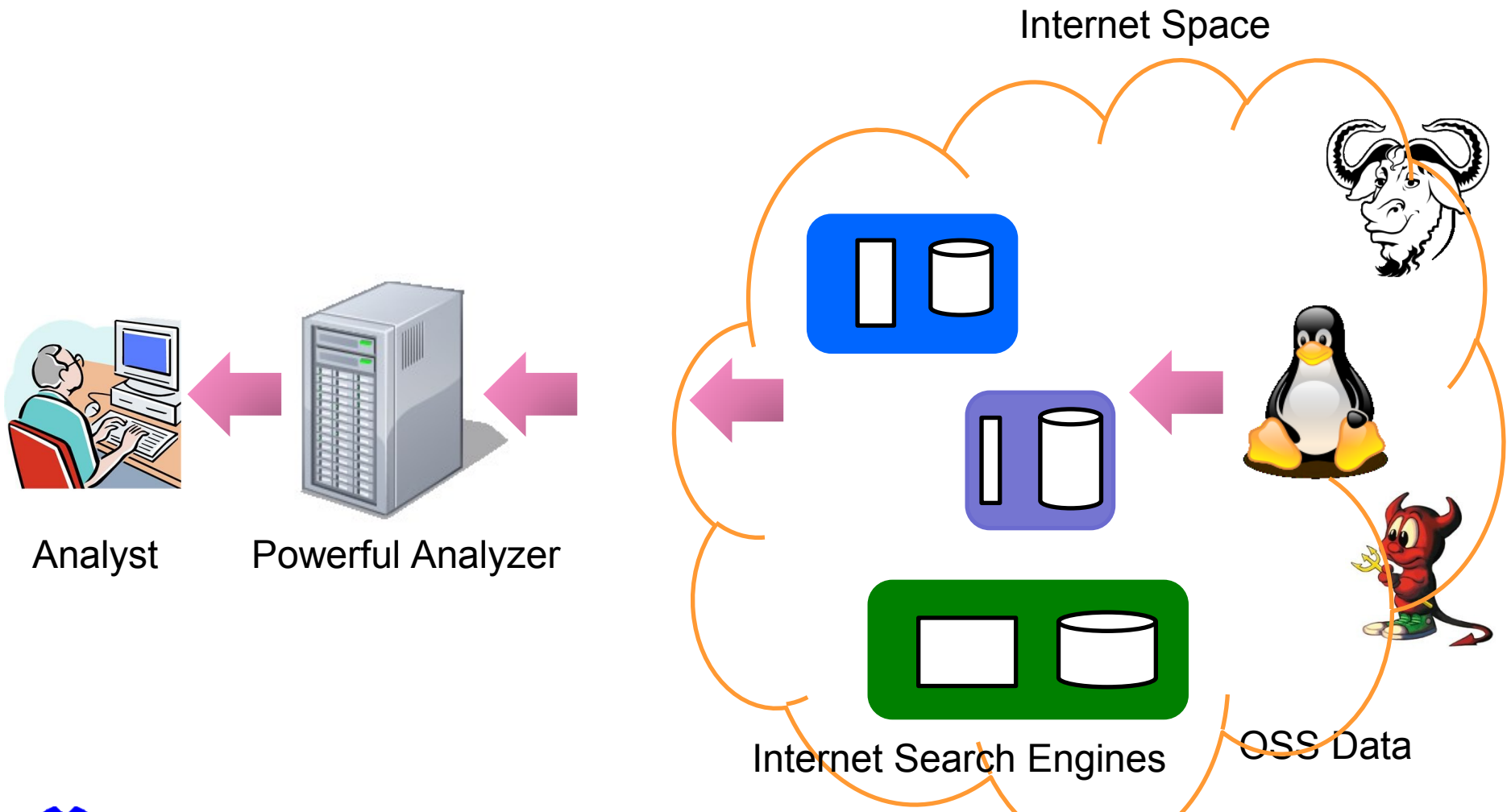


Analyst

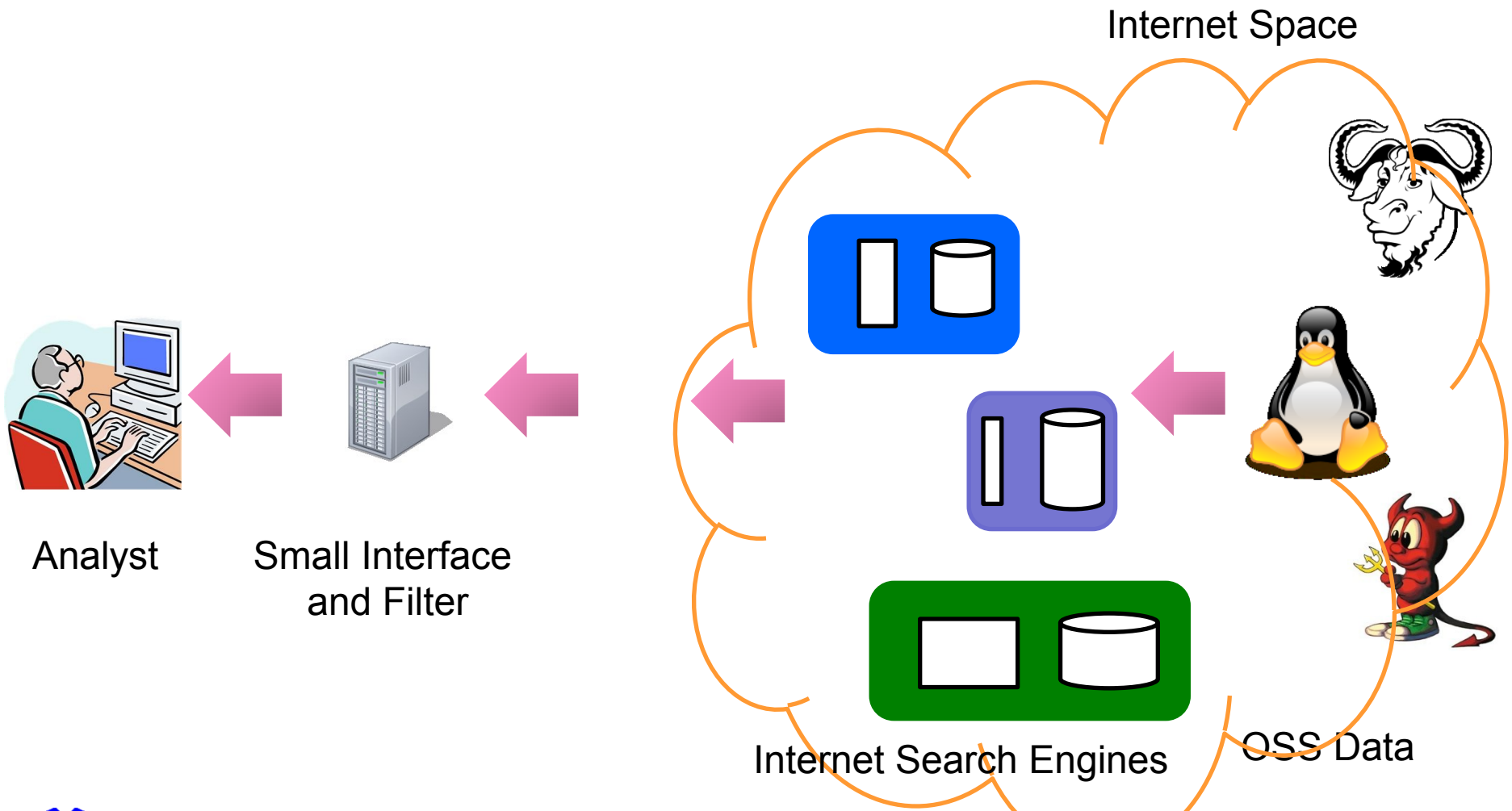
Powerful Analyzer

OSS Data

Mining OSS Data

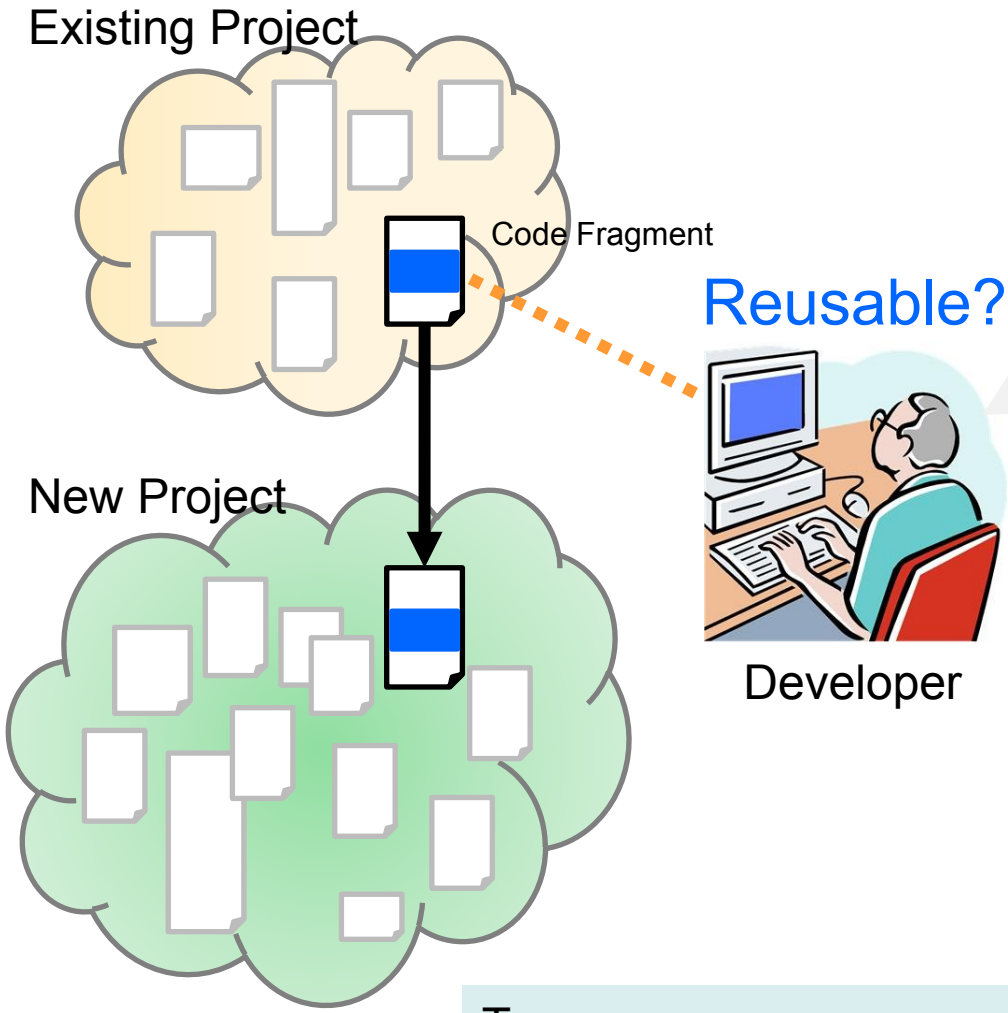


Mining OSS Data



Code History Tracking Model and Ichi Tracker

Developer's Concerns

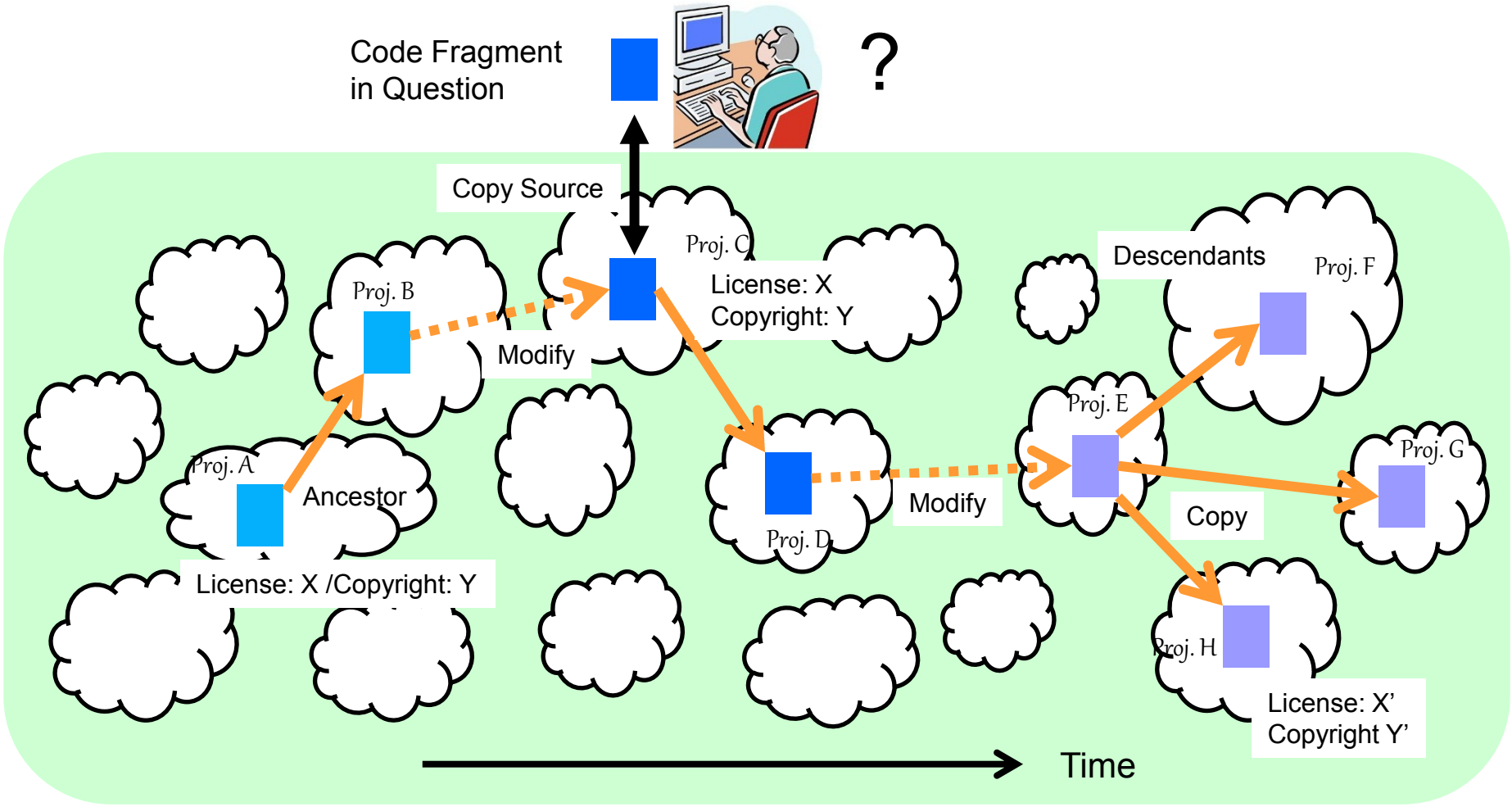


Concerns

- Origin
  - Who?
  - When?
  - License?
  - Copyright?
- Evolution
  - Maintenance?
  - Popularity?
  - Newer version?
  - …

To ease concerns, a support system is needed

Code History Tracking System



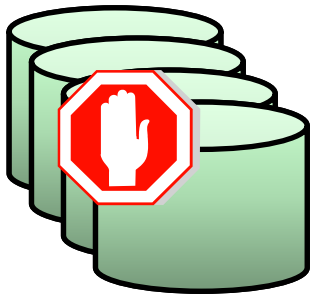
OSS Repositories

Design Policy of Ichi Tracker

OSS repository

- Target many OSS projects from old to new ones
- No crawling, no maintenance

→ Do not have local repository,
but use external code search engines

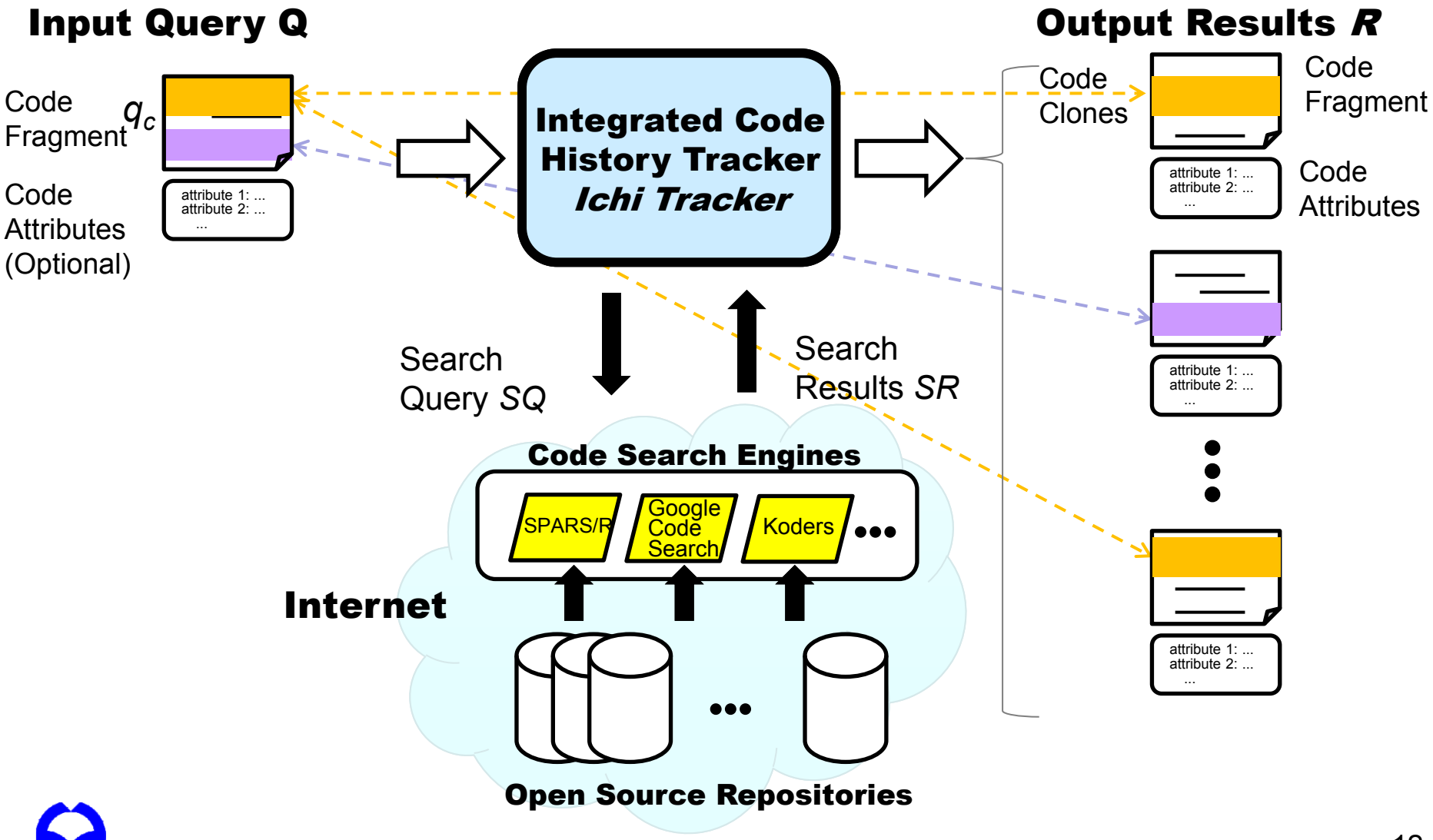


Output quality

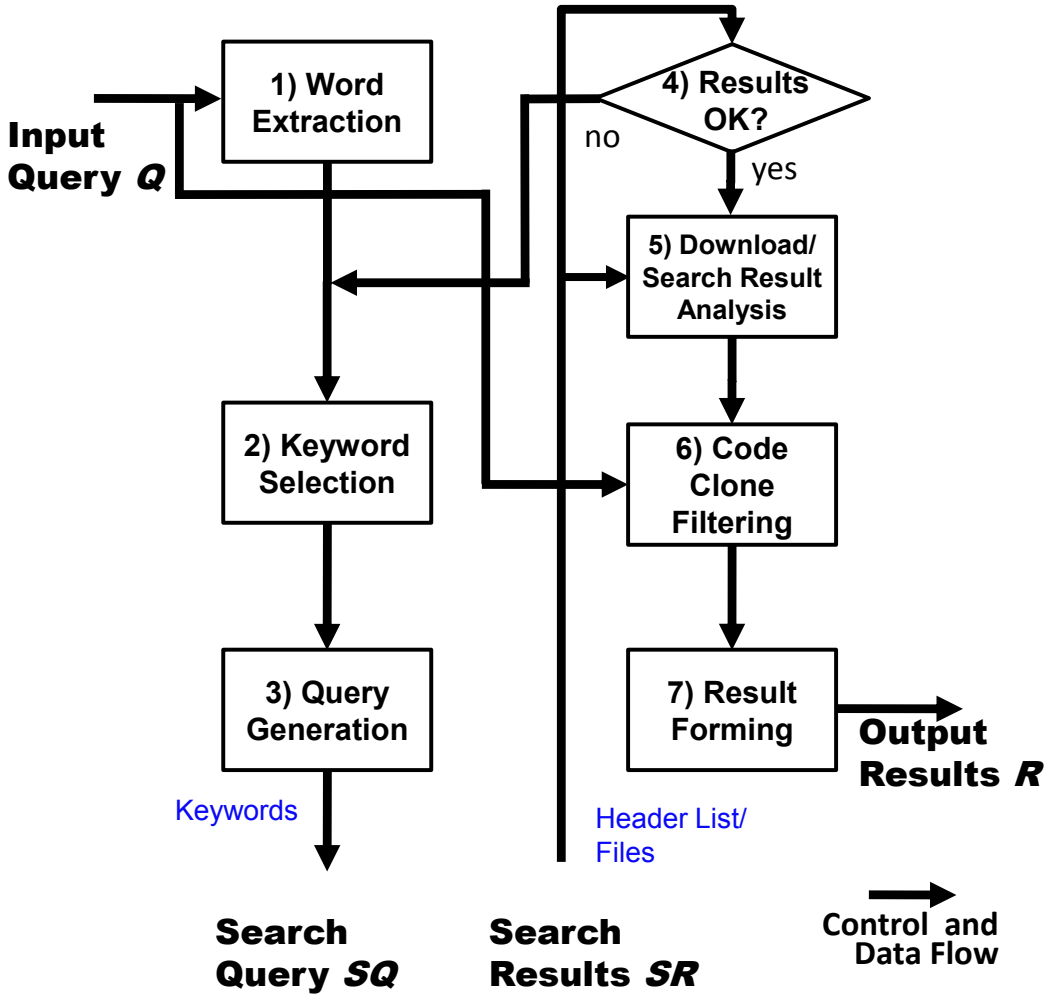
- Find not only exactly same code fragments, but also similar ones
- Lower false positive results
- No real-time response

→ Use code clone filtering to improve the output quality

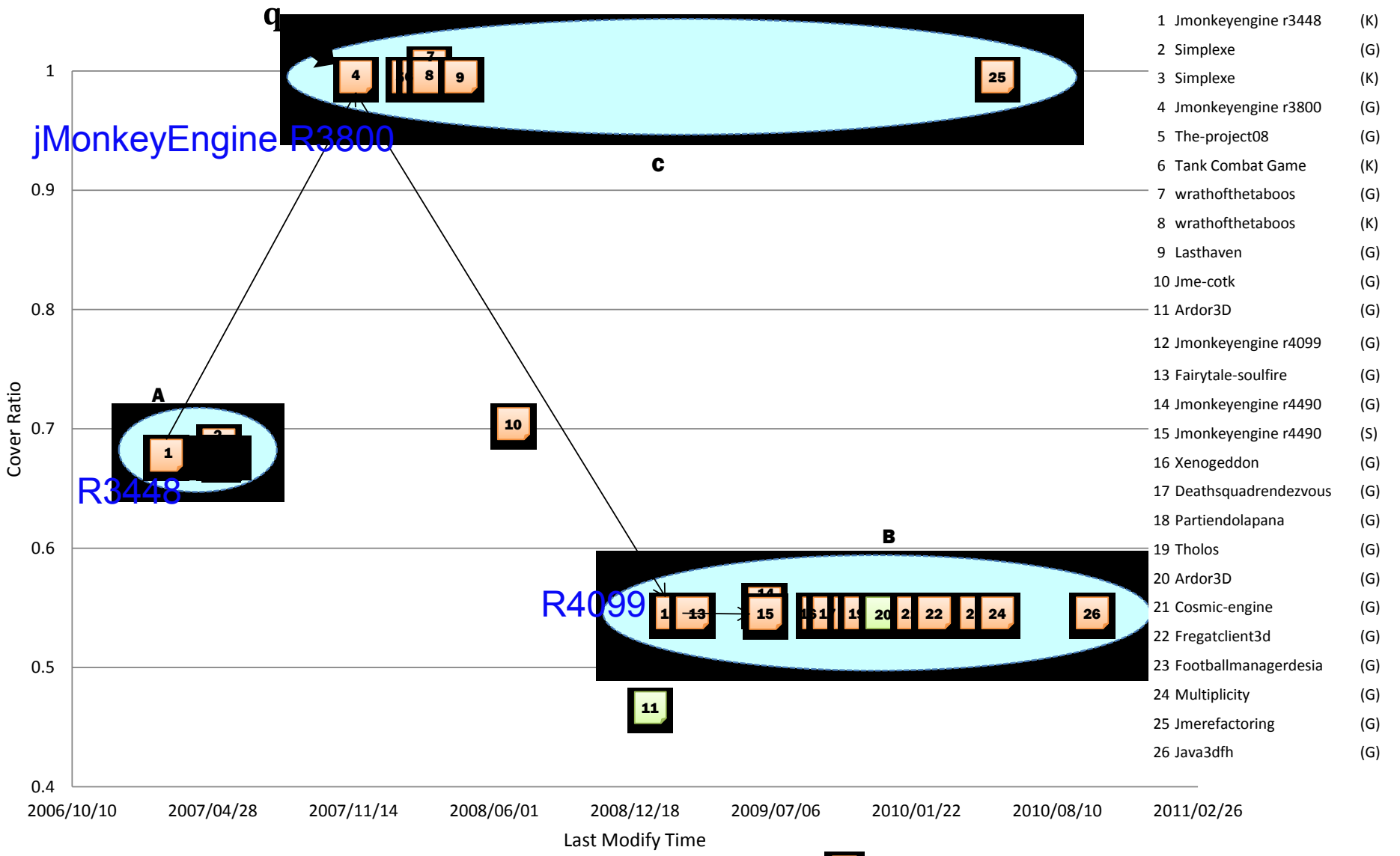
Code History Tracking Model



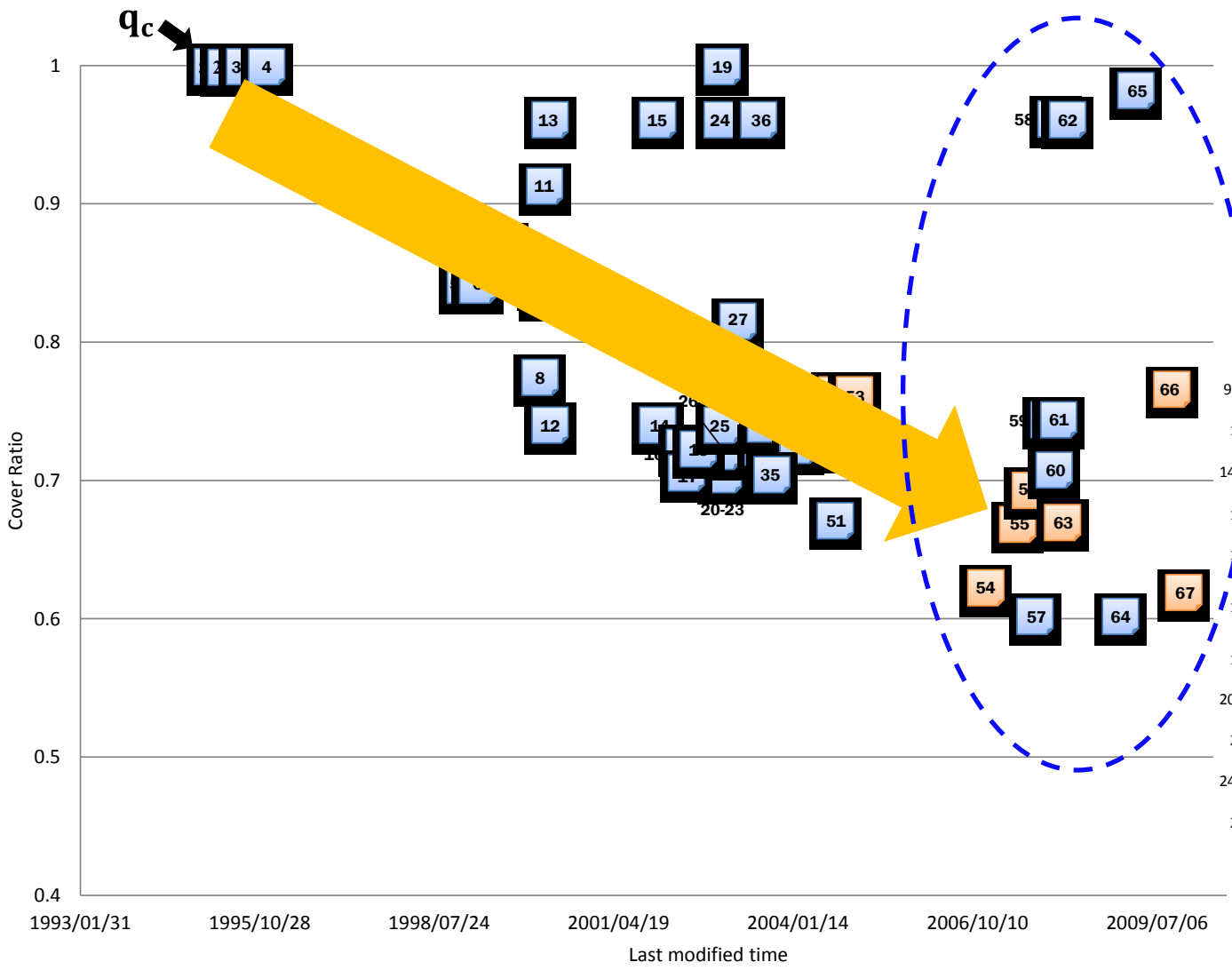
Process of Ichi Tracker



Evolution Pattern of Texture.java



Evolution Pattern of kern_malloc



1	Lites 1.0	(G)	28-33	Kame	(G)
2	Kernel Source Archive - CMU Mach 3.0	(K)	34-36	SimOS	(K)
3	Lites 1.1.u3	(G)	27-46	Kame	(G)
4	Lites 1.1-950808	(G)	47	Netnice	(G)
5	The Rio (RAM I/O) Project	(K)	48	Kame	(G)
6	ftp in The University of Edinburgh	(G)	49-50	Psumip	(G)
7	Mip-summer98	(G)	51	Netnice	(G)
8	freeBSD/SPARC	(G)	52	Reflexprotocol	(G)
9-12	ftp in Stockholm University	(G)	53	Netnice	(G)
13	freeBSD-cam2.1.5R	(G)	54	NetBSD v1.105	(K)
14-15	SonicOSX	(K)	55	OpenBSD PV Xen	(G)
16	Labyrinth BSD(labyrinthos)	(K)	56	OpenBSD v1.73	(K)
17	Oskit	(G)	57	Pmon	(G)
18	Psumip	(G)	58-62	Proyecto A.T.L.D. GNU/hurd(extremelinux)	(K)
19	Mach	(G)	63	OpenBSD v1.74	(K)
20-22	Savannah	(G)	64	Pmon	(G)
23	Unofficial OSKit source	(K)	65	774	(G)
24-26	Unofficial OSKit source(oskit)	(K)	66	Chord-ns3	(G)
27	ftp in Stockholm University	(G)	67	Openbsd-loongson-vc	(G)

Results by G(Google Code Search) and K(Koders)

File in New BSD License

File in Original BSD License

Oldest Files Found for Each Query File

Query File in SSHTools	Project Name of Found File	Cover Ratio of Found File	License	Copyright	Last Modified Time
SocketProxySocket.java	CVS client interface in Java	0.88	GPL 2	1998-99 Mindbright Technology AB	2001/11/12
Sftp.java	Apache Ant	0.62	Apache 1.1	2000-2002 Apache Software Foundation	2002/4/15
StringScanner.java	Programmer's Friend 4.1	0.81	CPL 1.0	2000-2003 Manfred Duchrow	2002/9/29
GeneralUtil.java	Gruntspud CVS Client	0.73	GPL 2	2002 Brett Smith	2003/11/12
Base64.java	Base64 notation	0.62	Public Domain	No copyright	2004/1/6
CharBuffer.java	Java Telnet daemon	0.81	GPL 2	2000 Dieter Wimberge	2004/1/16

SSHTools (all 339 files)

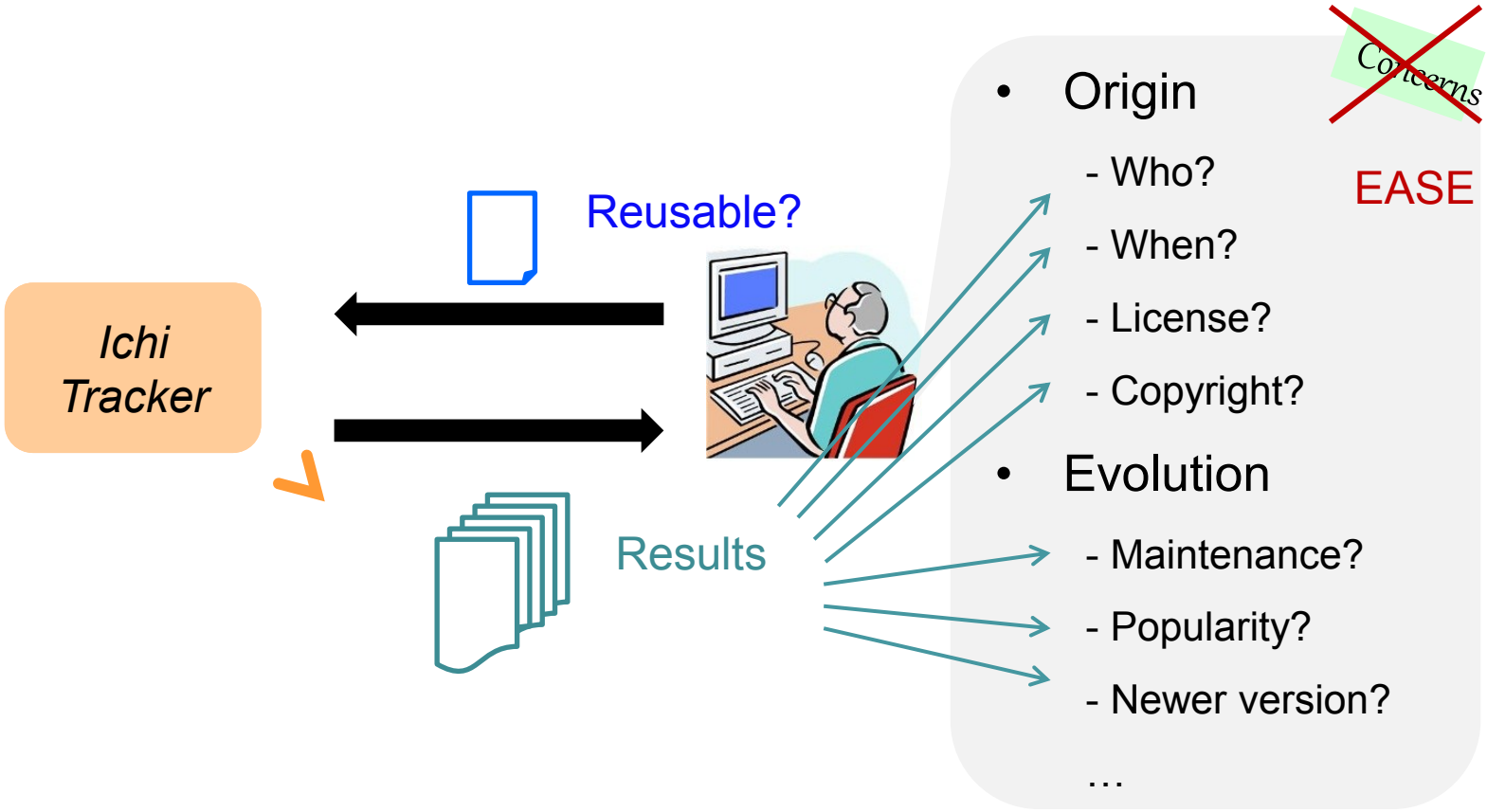
License:	GPL 2
Copyright:	2002-2003 Lee David Painter and Contributors
Last Modified Time:	2007/6/23

We have found many cases of different projects, different licenses and different copyrights

-> We need to check very carefully when we reuse SSHTools code

Usefulness

With simple check of the output of Ichi Tracker, we can get useful information for the history and evolution of code



Approach and Process

- Choosing good code search engines is a key to get high quality results

  GCS >> Koders > SPARS/R

  (GCS, Koders, SPARS/R) >> (Google, Bing)

  Need good code search engine available!

- Keyword selection strategy:

  Incremental strategy: try 1, 2, … keywords until the header list becomes less than 50

  – Decrement strategy, random, less frequently-used keywords, comment keywords, short keywords, …

  Less effective

Other Issues

- Performance
  - Case Studies A and B: 1 to 4 min.
  - Heavily depend on the code search engines and network performance

  Acceptable as non-interactive support system
- Quality of search result
  - Non-removed rate at the code clone filtering is an indicator of effectiveness of keyword search
    e.g., 0.46 (Case Study A(1) default setting)
  - The final output contains no false positives results

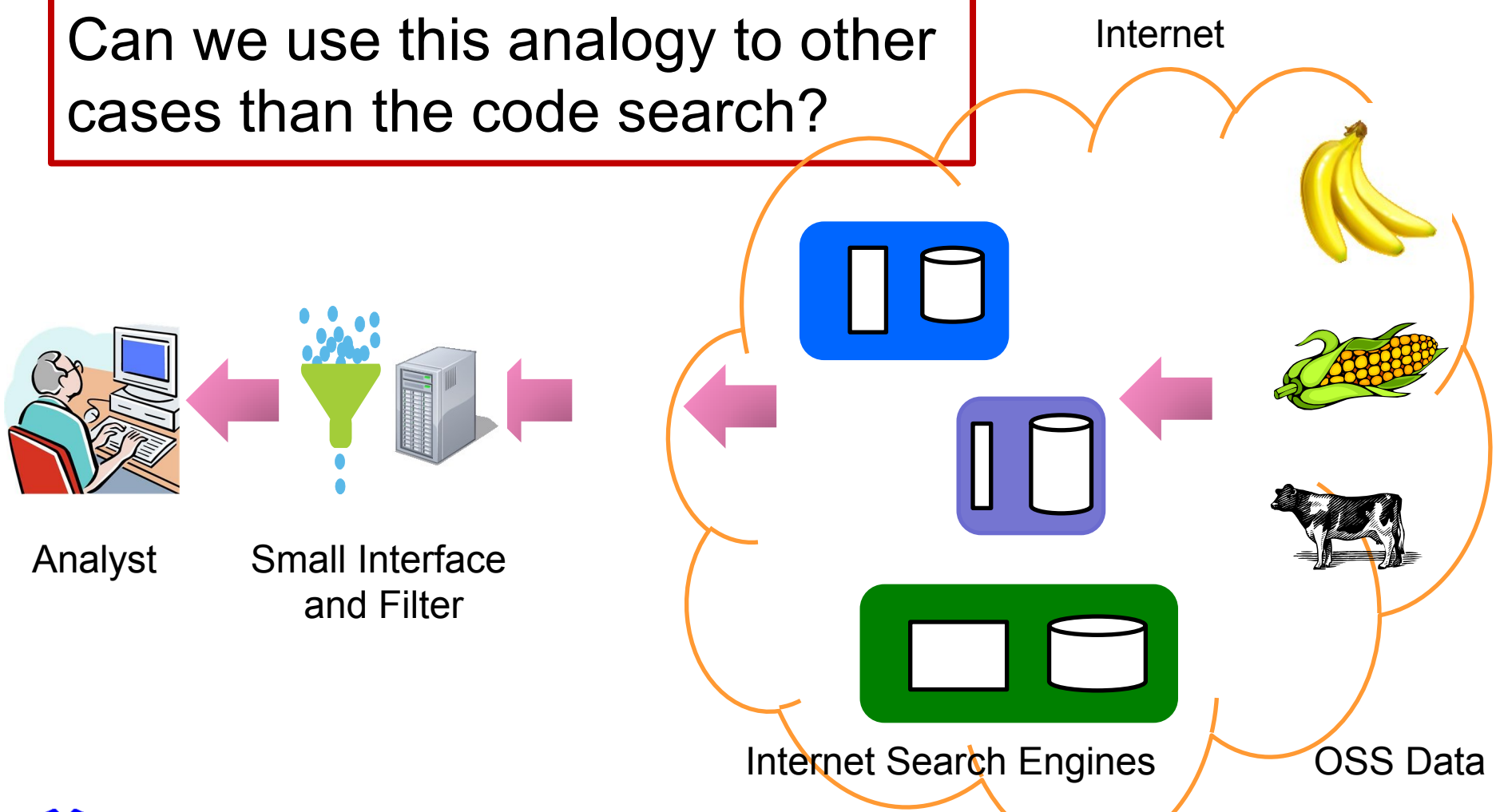
Pros and Cons

+ No huge repository
+ No maintenance cost and inconsistency
+ Easy to get the initial results

– Hard to control search results (engines) in details
– Fragile results depending on search engines

Mining Resources in the Internet Space

Can we use this analogy to other cases than the code search?



Thank you!