UDC 004.4:004.62

AN OVERVIEW OF A PETABYTE-SCALE INFRASTRUCTURE FOR OPEN-SOURCE SOFTWARE ECOSYSTEM ANALYSIS

Polishchuk V.L., postgraduate student
Kyiv National University of Technologies and Design
Skidan V.V., Candidate of Technical Sciences, Associate Professor
Kyiv National University of Technologies and Design

Keywords: World of Code, mining software repositories, open-source software, research infrastructure, Git, empirical software engineering.

Empirical studies of open-source software (OSS) increasingly require data that spans multiple projects, platforms, and developer communities. Supply chain analysis, vulnerability propagation, and developer migration are examples of research questions that demand ecosystem-scale data, millions of repositories and billions of Git objects. However, platform-specific APIs (e.g., GitHub REST API) impose rate limits, cover only a single forge, and provide no mechanism to track code or developers across repository boundaries. Researchers who need cross-platform coverage must independently discover, clone, and index repositories, a task that demands substantial resources and months of effort.

World of Code (WoC) [1] addresses this problem by providing a pre-indexed research infrastructure that aggregates the complete version control history of virtually all publicly accessible Git repositories across all major hosting platforms. As of version V4 (August 2024), WoC contains 4.9 billion commits and 20.4 billion file blobs, connected through 47 relational maps that link Git entities to one another.

WoC models the open-source ecosystem as a graph of interconnected Git entities. The data model operates on seven entity types, each identified by a single-letter code: commits (c), blobs (b), files (f), authors (a), and projects (p), along with their enhanced counterparts – aliased authors (A) and deforked projects (P). Capital-letter entities represent corrected data produced by WoC's processing pipeline. Author aliasing (a → A) applies ML-based identity resolution [2] to merge the many name–email variants a single developer may use across repositories.

These entities are connected through 47 relational maps named by their source–target pair, e.g., c2f (commits → modified files), b2P (blob → canonical projects), a2A (raw → canonical author), c2dat (commit metadata: timestamp, author, tree, parents).

Figure 1 summarises the WoC pipeline: public Git forges are periodically re-crawled, raw Git objects are passed through identity resolution and deforking, and the resulting 47 relational maps are exposed to researchers through either a local Python library or a remote HTTP API.

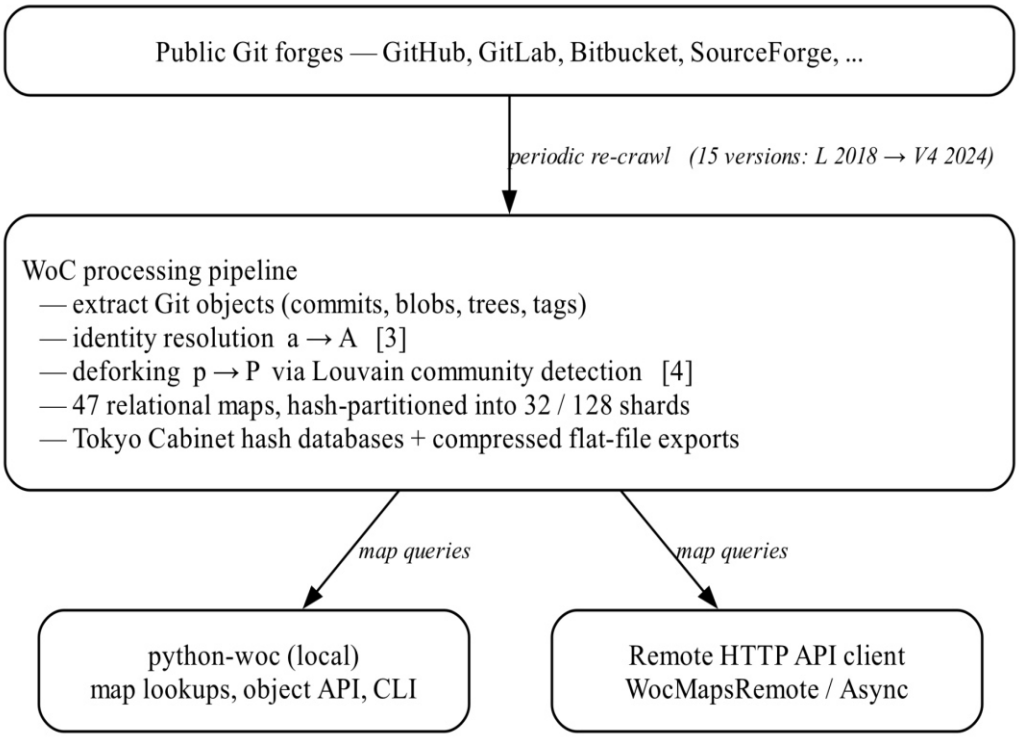

Figure 1 — WoC infrastructure overview

WoC is updated periodically by re-crawling all known forges and recomputing the relational maps. Over six years and 15 data versions, from version L (December 2018) to V4 (August 2024), the number of indexed commits grew 3.5-fold (1.4B to 4.9B) and blobs grew 3.8-fold (5.3B to 20.4B). Each version represents a complete snapshot; earlier versions remain available for longitudinal reproducibility. Table 1 summarises the growth trajectory across selected releases. Dashes denote figures not reported in the corresponding release notes.

The primary programmatic interface to WoC is the python-woc library [4], available on PyPI. It exposes three abstraction levels – raw map lookups via get_values, an object-oriented API with entity classes (Commit, Blob, Author, Project, Tree) supporting cached access and graph traversal, and a CLI that drop-in-replaces the legacy Perl tools – and achieves a 100–1000× speedup over them. Listing 1 shows a direct lookup and the multi-hop chain introduced in the Data Model subsection.

```
from woc.local import WocMapsLocal
woc = WocMapsLocal()
# Single lookup: projects containing a file blob
projects = woc.get_values("b2P", blob_sha)
# Multi-hop chain a2A → A2c → c2P
A = woc.get_values("a2A", raw_author)[0]
hits = [p for c in woc.get_values("A2c", A)
          for p in woc.get_values("c2P", c)]
```

Table 1

WoC data scale across selected versions

Version	**Date**	**Commits**	**Blobs**	**Repositories**
V4	Aug 2024	4,942 M	20,434 M	-
V3	May 2024	4,736 M	19,620 M	234.9 M
V	May 2023	3,928 M	16,252 M	209.0 M
U	Nov 2021	3,113 M	12,490 M	-
T	Mar 2021	2,596 M	10,461 M	146.1 M
S	Sep 2020	2,326 M	9,192 M	135.2 M
R	Mar 2020	2,034 M	7,918 M	123.8 M
L	Dec 2018	1,419 M	5,313 M	-

For researchers without direct server access, python-woc also provides a remote API client. The HTTP endpoint requires no account on the WoC cluster, the same map queries can be issued over the network using synchronous or asynchronous clients with built-in rate limiting and exponential backoff. This substantially lowers the entry barrier, as researchers can explore the dataset from any machine with an internet connection before deciding whether to request full server access.

WoC's pre-indexed maps collapse queries that would otherwise take months of data collection into simple lookups and multi-hop traversals, and have already been used across a range of published studies. Mauerer et al. [5] applied the identity resolution chain (a2A → A2c → c2P) to model developer expertise across the open-source ecosystem, reconstructing complete cross-platform contribution histories regardless of which forges or email addresses developers used. Other published work has used blob-level maps (b2P, b2c) to study how dependency metrics predict popularity changes in package ecosystems, and commit-to-file / commit-to-metadata maps (c2f, c2dat) to measure programming-language adoption at ecosystem scale.

The same composition pattern extends to many other questions: vulnerability propagation (a single b2P lookup returns every project containing a flawed file), license compliance auditing, ecosystem-scale clone detection, and longitudinal studies of developer practices. Cross-platform coverage, identity resolution, and fork detection together make WoC particularly valuable where single-platform datasets would introduce selection bias or miss cross-repository phenomena.

WoC's strengths come with trade-offs worth noting. First, data recency: as of April 2026, the latest public snapshot remains V4 (August 2024), leaving roughly a twenty-month gap between the frontier of public Git activity and what the infrastructure reflects; studies of recent vulnerabilities, newly released languages, or ongoing community shifts must therefore supplement WoC with forge-specific APIs. Second, the identity-resolution (a → A) and deforking (p → P) pipelines are heuristic – ML-based aliasing [2] and Louvain community detection [3] both report non-trivial error rates, so downstream studies should sample-validate canonical groupings. Third, WoC covers only publicly accessible Git repositories, so enterprise, academic, and smaller-forge codebases remain out of scope. Finally, full programmatic access requires an account on the WoC cluster; the remote HTTP API lowers but does not remove this gate.

World of Code provides a unique research infrastructure for ecosystem-scale software mining, no other system offers pre-indexed cross-repository relational maps covering 4.9 billion commits and 20.4 billion blobs across 234 million repositories. By handling data collection, identity resolution, and fork detection, WoC lowers the barrier for researchers who lack the resources to build their own mining infrastructure. Server access can be requested through worldofcode.org, and the remote API allows exploration without an account. The infrastructure continues to evolve with periodic data updates and community-driven development through regular hackathons.

References

1. Y. Ma, T. Dey, C. Bogart, S. Amreen, M. Valiev, A. Tutko, D. Kennard, R. Zaretzki, A. Mockus, "World of Code: Enabling a Research Workflow for Mining and Analyzing the Universe of Open-Source VCS Data," *Empirical Software Engineering*, vol. 26, no. 2, art. 22, 2021. DOI: 10.1007/s10664-020-09905-9

2. T. Dey, A. Karnauch, A. Mockus, "A Dataset and an Approach for Identity Resolution of 38 million Author IDs extracted from 2B Git Commits," in *Proc. 17th Int. Conf. Mining Software Repositories (MSR)*, Seoul, 2020. DOI: 10.1145/3379597.3387500

3. A. Mockus, D. Spinellis, Z. Kotti, G. J. Dusing, "A Complete Set of Related Git Repositories Identified via Community Detection Approaches Based on Shared Commits," in *Proc. 17th Int. Conf. Mining Software Repositories (MSR)*, Seoul, 2020, pp. 513–517. DOI: 10.1145/3379597.3387499

4. python-woc: Python interface to World of Code. Available: https://github.com/ssc-oscar/python-woc

5. T. Dey, A. Karnauch, A. Mockus, "Representation of Developer Expertise in Open-Source Software," in *Proc. 43rd Int. Conf. Software Engineering (ICSE)*, Madrid, Spain, 2021, pp. 995–1007. DOI: 10.1109/ICSE43902.2021.00094