# RepoScope: Bridging Physical Structure and Logical Flow for Intent-Driven Repository Documentation

Zezhou Yang[1] · Chaozheng Wang[2] · Hailiang Huang[3] · Peihua Zhang[3] · Ting Peng[3] · Yuetang Deng[3] · Chenxiong Qian[1]

[1] The University of Hong Kong [2] The Chinese University of Hong Kong [3] Tencent

CODE →

NeurIPS 2026

## 1 Background and motivation

Documentation connects implementation to architecture for developers and coding agents. Rapid code evolution makes manual upkeep difficult.

Bottom-up summaries describe local behavior but can miss system responsibilities and cross-component coordination.

**Decompose** → **Summarize** → **Aggregate**

### Challenge 1: Physical–logical misalignment

**One workflow, multiple directories**

- **core/handlers/ → handle request**
- **urls/ → resolve the view**
- **application/ → execute the view**

**One role can cross directory boundaries.**

**One directory, different responsibilities**

**utils/ (schematic)**

- **cache_utils.py** — **Caching**
- **mail_utils.py** — **Email**

**Nearby files need not share a system role.**

### Challenge 2: Contextual isolation of intent

**Local behavior: run response callbacks** → **Missing explanation: what system role does it serve?**

**Context reveals the role: response finalization**

**Build response** → **Run callbacks** → **Save session***

* Session saving when applicable. Examples adapted from the talk.

## 2 RepoScope framework

### PLP: define the scope

**1 Logical entry grouping**

Trace code reachable from each entry point; group entries with similar downstream implementations.

**Output: grouped entry points**

**2 Path-guided initialization**

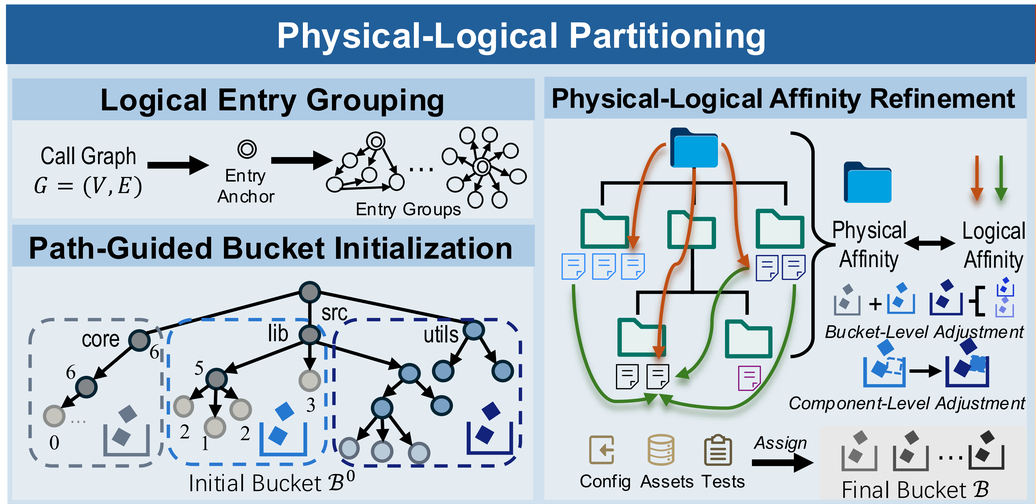
Map these groups to directories; split large regions and merge small coherent neighbors.

**Output: initial code buckets**

**3 Affinity refinement**

Use call relations and path proximity to refine bucket boundaries and component assignments.

**Output: buckets + artifacts + tests**




PLP groups code into buckets; SAIL revises each bucket's role against code evidence and uses it to guide documentation.

### SAIL: seed–revise–lift

**1 Seed intent**

Use code from a bucket and related buckets to infer its provisional architectural role.

**Output: provisional intent seed**

**2 Revise and replay**

Check the role against smaller code units (cells); accept supported changes and regenerate affected text.

**Output: revised intent and cell docs**

**3 Hierarchical lift**

Combine cell documents under revised roles to write bucket, module, and repository documentation.

**Output: documentation hierarchy**

## 3 Quality across public repositories

| | |
|---|---|
| OpenDeepWiki | 36.77 |
| CodeWiki | 55.08 |
| DeepWiki | 60.76 |
| **RepoScope** | **73.88** |

7 core repos · mean Quality Score (%) · +13.12 pp vs. DeepWiki

**All 21 repositories**

| Method | Score | Covered requirements |
|---|---|---|
| CodeWiki | 59.57 | 891 / 1,471 |
| DeepWiki | 67.69 | 992 / 1,471 |
| **RepoScope** | **81.41** | **1,194 / 1,471** |

## 4 Ablation and length controls

| | |
|---|---|
| Bottom-up | 36.21 |
| PLP only | 57.89 |
| Fixed intent | 58.92 |
| **Full SAIL** | **73.88** |

Fixed intent disables revision and replay together.

**Final evaluated text: length vs. quality**

| Method | Length (× full) | Score |
|---|---|---|
| PLP only | 0.99× | 57.89 |
| DocAgent | 1.37× | 56.37 |
| **RepoScope** | **1.00×** | **73.88** |

## 5 Agent and developer utility

| | |
|---|---|
| Raw Code | 79 / 60 |
| OpenDeepWiki | 140 / 100 |
| CodeWiki | 190 / 149 |
| **RepoScope** | **227 / 195** |

Found / pass · five-run unions · 282 expert targets

**Developer median time (min; descriptive)**

| | |
|---|---|
| Raw Code | 148 min |
| CodeWiki | 126 min |
| **RepoScope** | **103 min** |