

Zulip: lower-mid market maintainability

Ranked against a benchmark of several thousand real-world systems, Zulip sits in the **lower-mid market** (5th to 35th percentile). Its strongest property is test ratio (3.34), its weakest file size (1.94).

CLIENT	—
SYSTEM	Zulip
SIZE	368,362 lines of code · 3,250 files
BUILT WITH	Python, TypeScript, CSS, HTML, Handlebars, Puppet, JavaScript and 12 more
HISTORY	since 2012 · 1,454 contributors
CODE AS OF	21 September 2026 (last commit)
MEASURED	23 September 2026

OVERALL · MAINTAINABILITY

1.96

 ★★☆☆☆

Lower-mid market

5th to 35th percentile. The overall score ranks the whole system against the market; it is not the average of the properties.

Market position

★★★★★	above 4.5	Top 5% of the market
★★★★☆	3.5 – 4.5	65th to 95th percentile
★★★☆☆	2.5 – 3.5	35th to 65th percentile
★★☆☆☆	1.5 – 2.5	5th to 35th percentile
★☆☆☆☆	below 1.5	Bottom 5% of the market

PROPERTY

SCORE

WHAT WAS MEASURED

Volume

How much code there is. The larger the system, the more it costs to keep in your head and to change.

3.28 ★★☆☆☆ Average

py 46.7% ts 31.5% css 9.8% other (16) 12.0%

368,362 lines of main code, sized at about 52 person-years by the model

Test ratio

How much test code there is beside the main code. Tests are what make a change safe to make.

3.34 ★★☆☆☆ Average

main 368,362 · 63.8% test 208,905 · 36.2%

Test code is 56.7% the size of the main code

Duplication

How much code is copied. A fix in one copy is easily missed in the others.

3.08 ★★☆☆☆ Average

10.6% of the code is duplicated (35,139 lines)

File size

How large the files are. Large files gather unrelated changes and are harder to navigate.

1.94 ★☆☆☆☆ Below average

1001+ 29.0% 501-1000 20.1% 201-500 20.4% 101-200 10.7% 1-100 19.8%

49.1% of the main code sits in files of 501+ lines

Unit size

How long the functions and methods are. Short units are easier to understand, test and reuse.

2.41 ★☆☆☆☆ Below average

101+ 9.7% 51-100 11.5% 21-50 25.3% 11-20 22.4% 1-10 31.1%

21.2% of the code in units sits in units of 51+ lines

Unit complexity

How many decision paths a unit has. Every path is one more case to understand and to test.

2.08 ★☆☆☆☆ Below average

51+ 2.2% 26-50 3.5% 11-25 11.1% 6-10 16.0% 1-5 67.1%

5.7% of the code in units sits in units with 26+ decision points

How to read the score

Every score runs from **0.5 to 5.5**, higher is better, and compares this system with a benchmark of several thousand real-world systems measured the same way. **3.0 is the market median**.

A property score ranks one trait. The overall score ranks the **whole system**, so it can sit below or above every property score: being average at everything places a system lower than its average suggests.

Basis

Scope One repository with its full git history

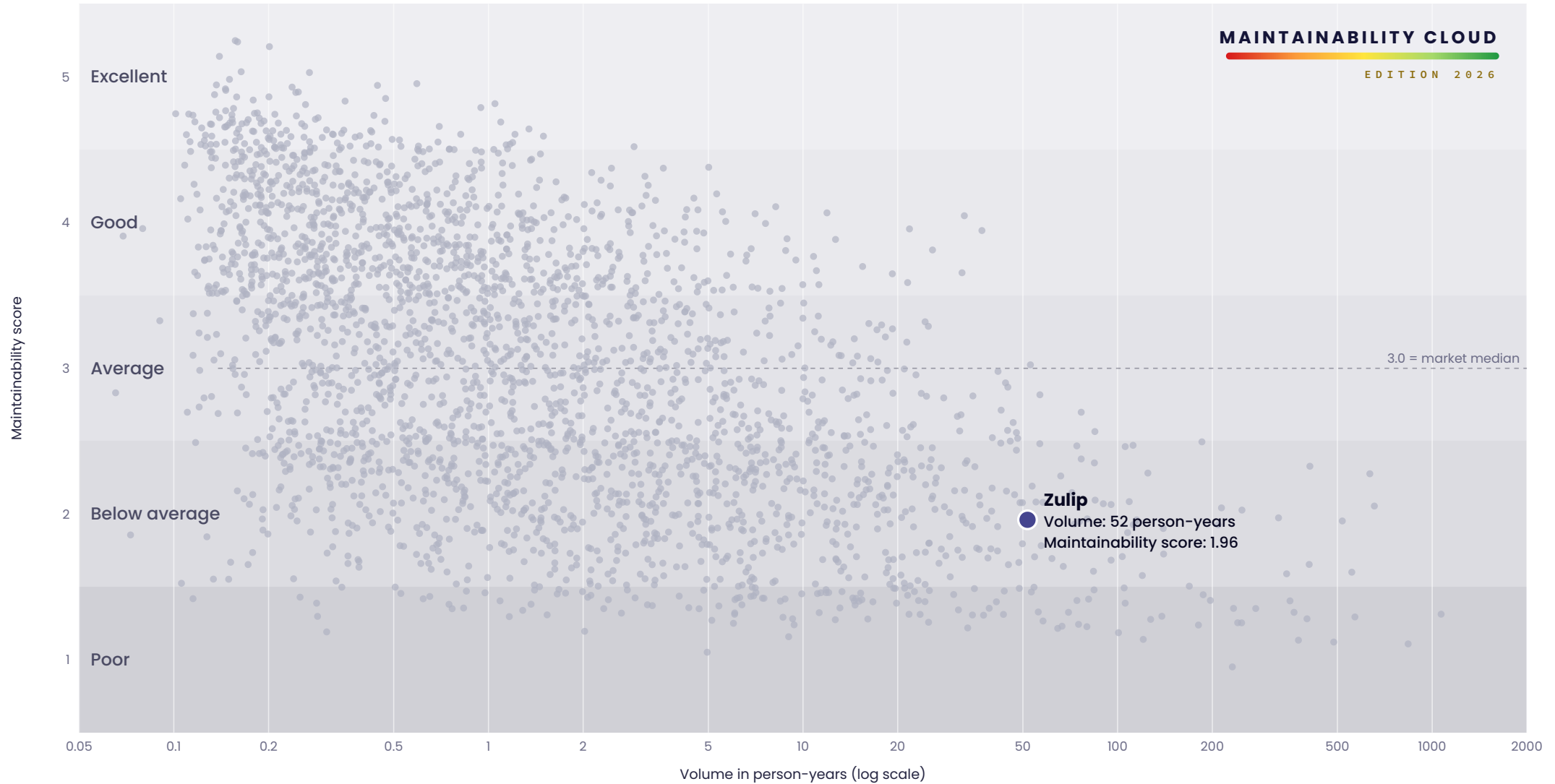
Assessor Claroview · claroview.io

Model Gem:2026-Q3B · mc-algorithms-20260718 · mc-quality-engine-20260728

The measurements behind these scores were produced with **Socrates** (socrates.dev) and scored by Maintainability Cloud. The score describes the code as of its last commit; it is not a certification. The score holds for the scope measured and the model named above: refining the measurement scope, for example by excluding code that is not maintained by the team, or rescored with another model version, can change it.

Zulip in the Maintainability Cloud

Each grey point is one open-source system from the benchmark, scored with the same model as Zulip. Zulip is the navy point: **1.96** at about 52 person-years.



Volume is the model's estimate of the effort the code represents, so both axes belong to the scoring model named on page 1: the cloud is only valid for scores produced with it. Higher is better; further right is larger.