Rapid Prototyping of Language Servers with **Rascal**, **VScode**, and **Gitpod**

Jurgen J. Vinju
Centrum Wiskunde & Informatica
TU Eindhoven
Swat.engineering

There is way too much code...

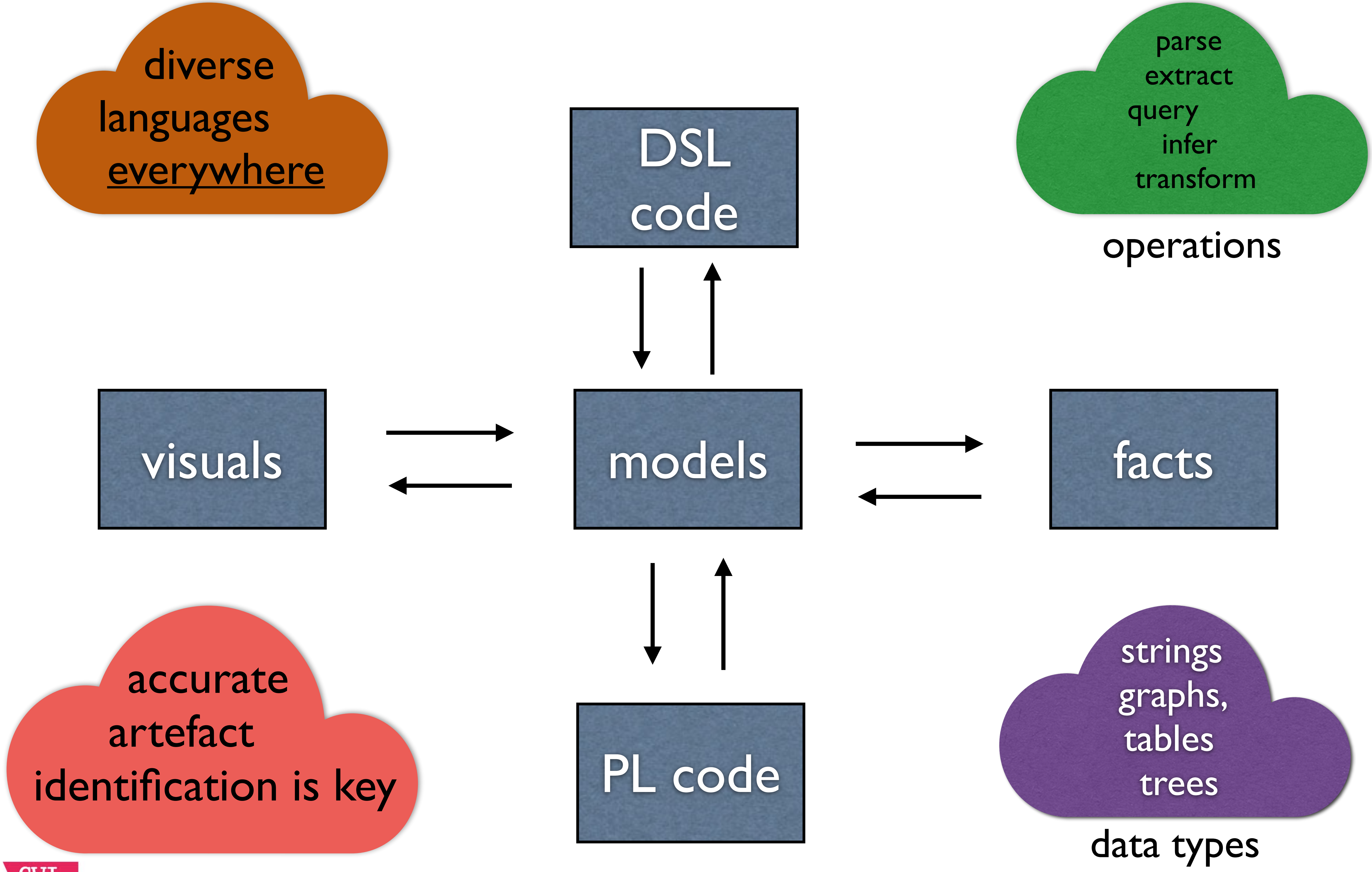
- Code can be **generated** from DSLs
  - better exploration, testing, verification
  - better maintenance
  - less boring = more fun!
- Code can be **analyzed** (using “open” compilers)
  - evidence-based maintenance
  - debugging
- Code can be **transformed**, source-to-source
  - mass maintenance activities
  - code-to-model, reverse engineering

Really <3 source code but there is too much of it!

M. Hamilton

Automating software construction and maintenance is the [only] humane solution [T. Girba]

Rascal is for *everything* meta



diverse languages everywhere

DSL code

parse
extract
query
infer
transform

operations

visuals

models

facts

PL code

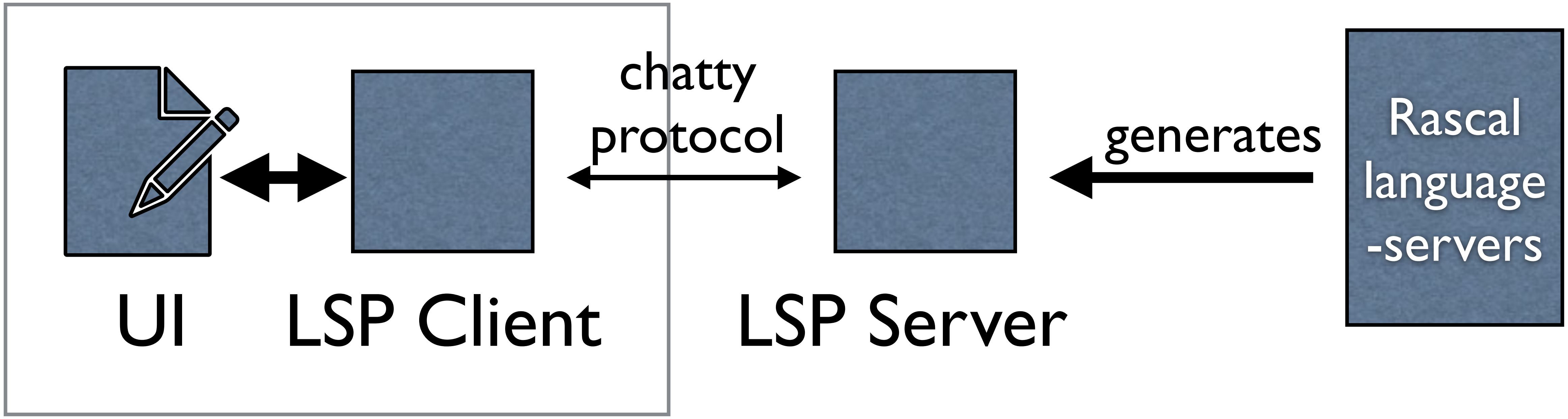
accurate artefact identification is key

strings
graphs,
tables
trees

data types

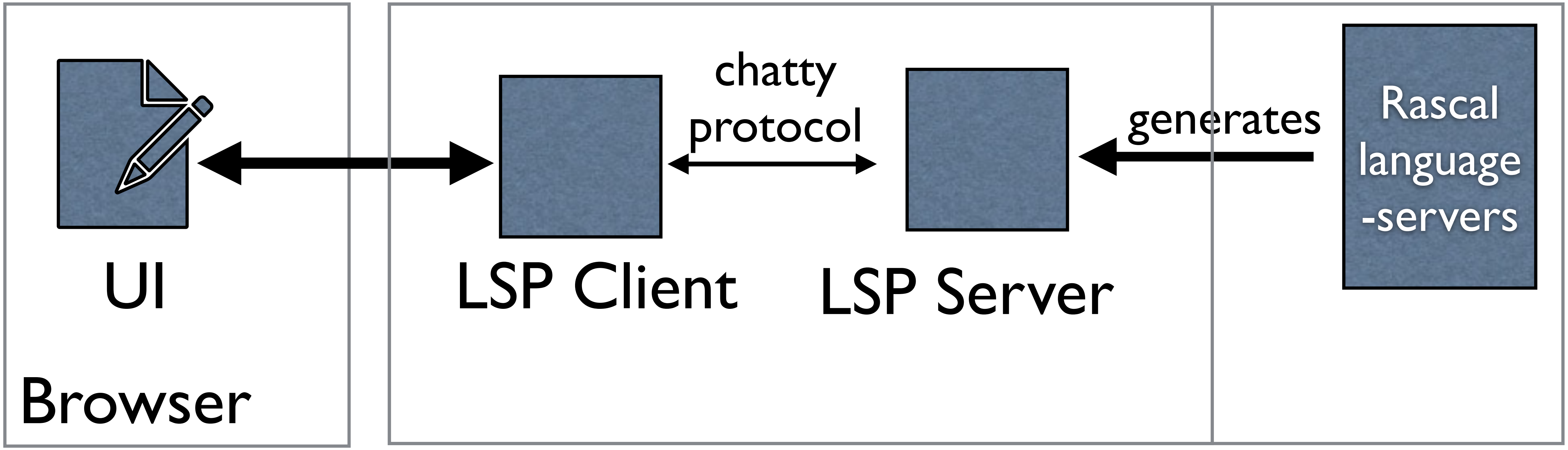
Meta in the IDE

- focus on Domain Specific Languages (DSLs)
- want editor support with all the fanciness
- Microsoft's Language Server Protocol



Meta in the IDE

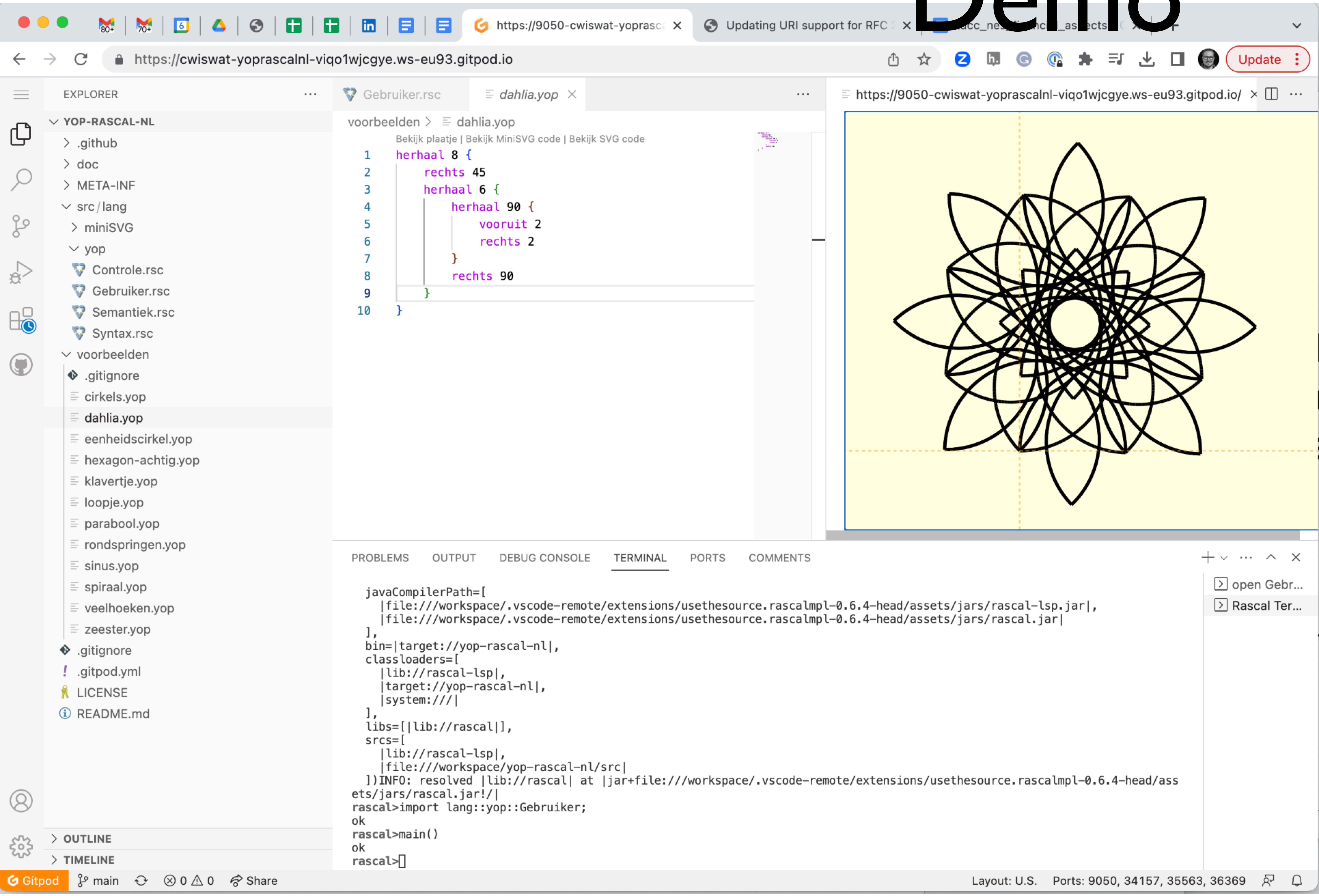
- want editor support with all the fanciness
- Microsoft's Language Server Protocol
- GitPod frees the generated IDEs
- Rascal provides <u>immediate</u> extensions



YOP

- YOP is a toy language made for highschool students
- Make Your Own Programming Language
- LOGO-like
- Features: parsing, intermediate miniSVG language, code generation to SVG, static checking, jump-to-definition
- ±350 LOC
- Funny variations:
    - colors
    - procedures with recursion
    - smoothening optimizer

Demo



```
herhaal 8 {
    rechts 45
    herhaal 6 {
        herhaal 90 {
            vooruit 2
            rechts 2
        }
        rechts 90
    }
}
```

```
  javaCompilerPath=[
    |file:///workspace/.vscode-remote/extensions/usethesource.rascalmpl-0.6.4-head/assets/jars/rascal-lsp.jar|,
    |file:///workspace/.vscode-remote/extensions/usethesource.rascalmpl-0.6.4-head/assets/jars/rascal.jar|
  ],
  bin=|target://yop-rascal-nl|,
  classloaders=[
    |lib://rascal-lsp|,
    |target://yop-rascal-nl|,
    |system:///|
  ],
  libs=[|lib://rascal|],
  srcs=[
    |lib://rascal-lsp|,
    |file:///workspace/yop-rascal-nl/src|
  ])INFO: resolved |lib://rascal| at |jar+file:///workspace/.vscode-remote/extensions/usethesource.rascalmpl-0.6.4-head/ass
ets/jars/rascal.jar!/|
rascal>import lang::yop::Gebruiker;
ok
rascal>main()
ok
rascal>
```

SWAT - Software Analysis And Transformation

An open source community;
script all your coding and research
activities with → Rascal ←
http://www.rascal-mpl.org

Amazing immediate full-blown
development environments

Co-design and co-maintenance
of software portfolios through
(domain-specific) language engineering

SWAT: the next steps in programming
and software engineering in Amsterdam

https://gitpod.io/#https://github.com/cwi-swat/yop-rascal-nl

Other examples

- DSL with ING: “Rebel” DSL for distributed transaction systems with availability and isolation guarantees
- DSL with NFI: carving fast and slow, generating “undelete” algorithms for recovering destroyed data
- Example research (by Lina Ochoa): measuring impact of breaking changes in APIs and the effective use of semantic versioning on the ecosystem level