



# Inspect4py

A Knowledge Extraction Framework for Python Code Repositories

**Dr. Rosa Filgueira**, University of St Andrews, Scotland, UK

✉ [rf208@st-andrews.ac.uk](mailto:rf208@st-andrews.ac.uk)

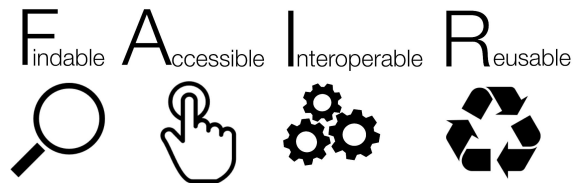
🐦 [@Rosa\\_Filgueira](https://twitter.com/Rosa_Filgueira)

**Dr. Daniel Garijo**, University Politecnica of Madrid, Madrid, Spain

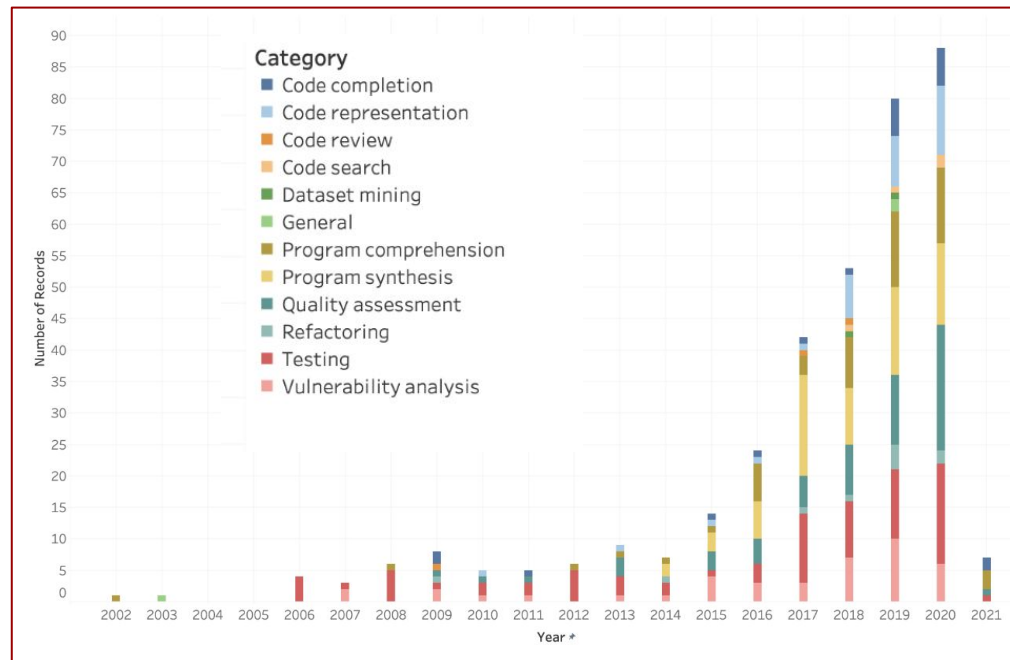
✉ [daniel.garijo@upm.es](mailto:daniel.garijo@upm.es)

🐦 [@dgarijov](https://twitter.com/dgarijov)

# Motivation



- **Best practices** for Open Science are spreading to Research Software
  - MSR now embraces FAIR! [1]
- Research Software can still be **difficult** to:
  - Understand
  - Adopt
  - Compare
  - Execute, reproduce or scale
- Research Software has become a **research topic**:
  - Each approach introduces their own tokenizer/ feature extractor/etc !!!



A Survey on Machine Learning Techniques for Source Code Analysis, 2021

[1] <https://conf.researchr.org/track/msr-2022/msr-2022-data-showcase>



Framework for extracting features from **Python** code repositories in order to:

- Ease **repository comprehension**
  - How to run a target repository (software invocation)
  - Determine the type of repository (library, package, service)
- Ease **machine readability** (extract features)
  - Extract available documentation
    - Functions
    - Methods
  - Call graph, Control Flow
  - File hierarchy, tests files
  - Dependencies and requirements

In a single, **unified** framework

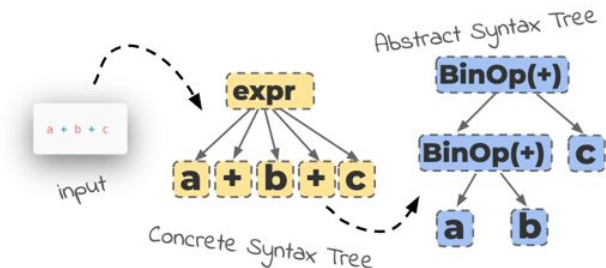
**Without executing** code repositories

**Reusing** existing tools

# Inspect4py: Main features

- Parse repositories to Abstract Syntax Tree (AST)
  - Extract details of classes, methods, functions, documentation, etc
- Employ additional tools for obtaining:
  - Requirements (*pigar*), Control Flow Graph (*cdmcfparser*)
- Analyse the previous information (*new set of heuristics*) to generate:
  - Dependencies, Call graph, Files hierarchy, Test Detection, Software Type and Software invocation
- Evaluation Summary
  - Manually annotated corpora (95 python repositories)
    - 24 packages, 27 libraries 13 services and 31 scripts
    - Overview of results for main software type classification for each category

## Intermediate parse tree



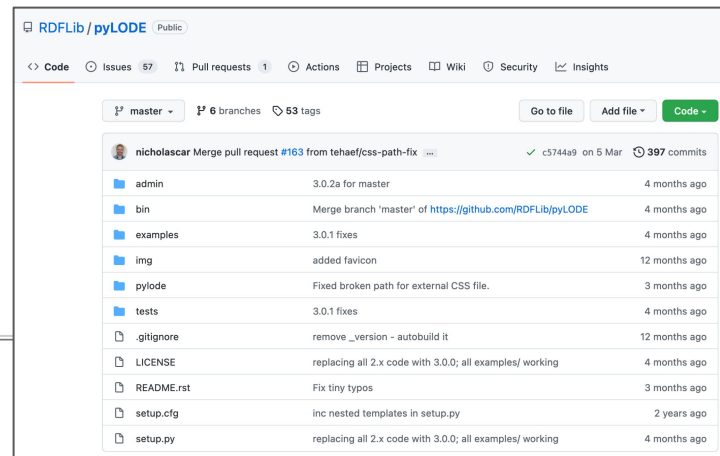
Software type	Precision	Recall	F1-score
Package	1	0.916	0.956
Library	0.93	1	0.9637
Service	1	1	1
Script	0.967	0.967	0.967

**Table 1: Results for software type classification.**



# Inspect4py: Usage example (2)

```
>> inspect4py -i pyLODE -o output_Pylode -r -html -si
```



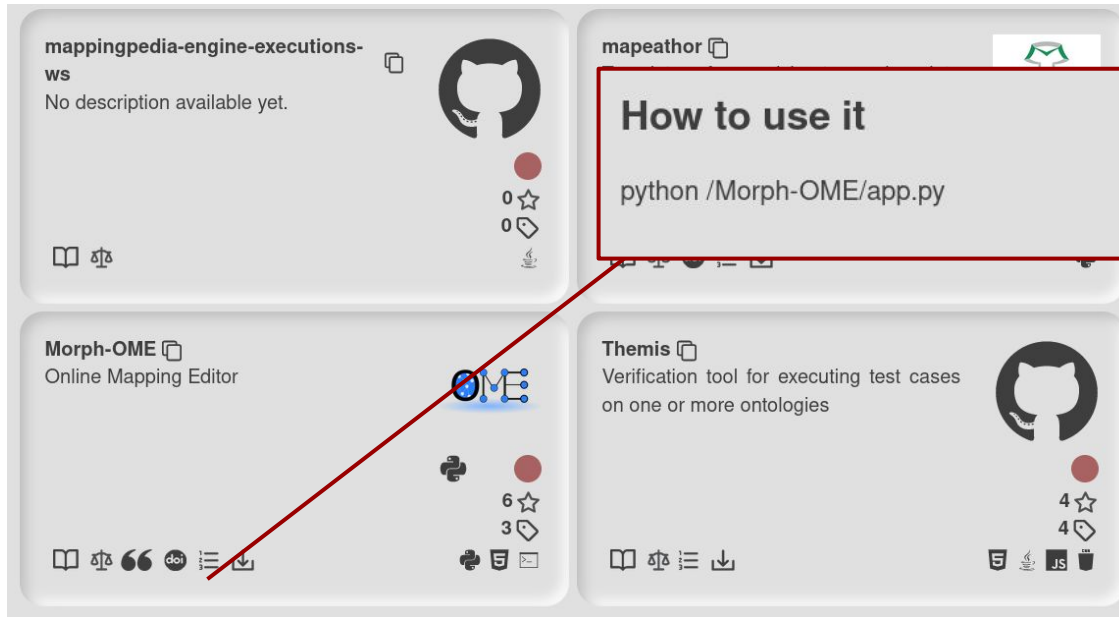
requirements	Markdown	3.3.6																				
	dominate	2.6.0																				
	pytest	7.1.2																				
	rdflib	6.1.1																				
	setuptools	54.2.0																				
tests	type	run	has_structure	mentioned_in_readme																		
	test	python /Users/rf208/pyLODE/tests/test_errors.py	main	False																		
	test	python /Users/rf208/pyLODE/tests/test_utils.py	body	False																		
	test	python /Users/rf208/pyLODE/tests/test_doc_html.py	body	False																		
software_invocation	<ul style="list-style-type: none"> <li> <table border="1"> <tr> <td>run</td> <td>pylode</td> </tr> <tr> <td>type</td> <td>package</td> </tr> <tr> <td>installation</td> <td>pip install pyLODE</td> </tr> <tr> <td>ranking</td> <td>1</td> </tr> </table> </li> <li> <table border="1"> <tr> <td>type</td> <td>script</td> </tr> <tr> <td>run</td> <td>python /Users/rf208/pyLODE/pylode/cli.py</td> </tr> <tr> <td>has_structure</td> <td>main</td> </tr> <tr> <td>mentioned_in_readme</td> <td>True</td> </tr> <tr> <td>ranking</td> <td>2</td> </tr> </table> </li> </ul>	run	pylode	type	package	installation	pip install pyLODE	ranking	1	type	script	run	python /Users/rf208/pyLODE/pylode/cli.py	has_structure	main	mentioned_in_readme	True	ranking	2			
	run	pylode																				
type	package																					
installation	pip install pyLODE																					
ranking	1																					
type	script																					
run	python /Users/rf208/pyLODE/pylode/cli.py																					
has_structure	main																					
mentioned_in_readme	True																					
ranking	2																					
software_type	package																					

Ranking software invocation methods

Software type  
based on our ranking score

# Inspect4py: Use cases

- Automated **software catalogs** (in combination with other tools for metadata extraction)
- Autocomplete readme files



The screenshot displays a grid of software project cards. The top-left card is for 'mappingpedia-engine-executions-ws' with no description and 0 stars. The top-right card is for 'mapeathor' with a 'How to use it' section highlighted in a red box, containing the command `python /Morph-OME/app.py`. The bottom-left card is for 'Morph-OME' (Online Mapping Editor) with 6 stars and 3 forks. The bottom-right card is for 'Themis' (Verification tool) with 4 stars and 4 forks. A red line connects the 'How to use it' section of the 'mapeathor' card to the 'Morph-OME' card.

## Work in Progress

- Software similarity (feature extraction)

# Inspect4py is **Open Source** and freely available



<https://github.com/SoftwareUnderstanding/inspect4py>



<https://inspect4py.readthedocs.io/en/latest/>



<https://pypi.org/project/inspect4py/>



**Inspect4py** 

Problems? Questions? Open an issue :)