UCov: Lightweight Syntactic API Usage Analysis

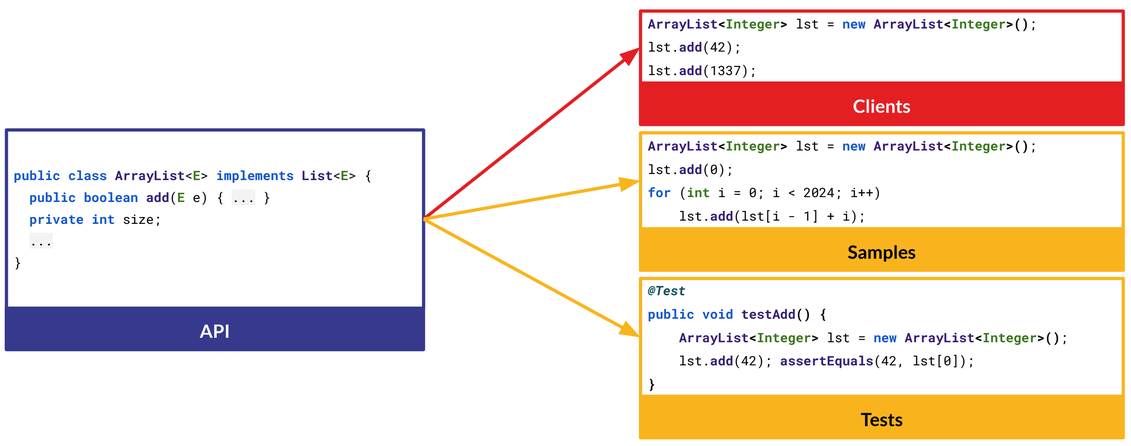
Gustave Monce - Thomas Degueule, Jean-Rémy Falleri

PROGRESS-Univ. Bordeaux, LaBRI, France

LaBRI

Software Libraries

- Today the majority of software is built on existing work provided by libraries
- These libraries export an API to multiple applications
- These are Clients, but also Samples and Tests of the library
- The API is the primary link between the library and the consumers

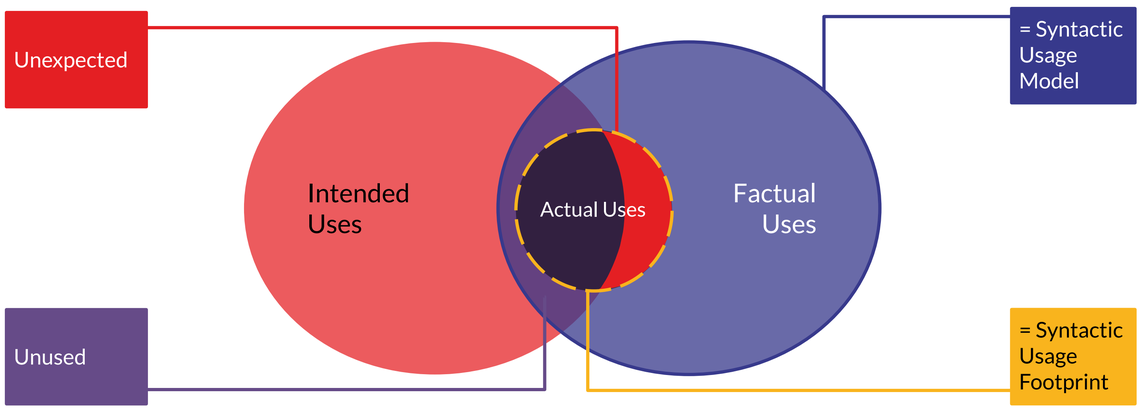


The problems at play

- Breaking changes from one library version to the next, are easily caught due to programs not compiling successfully
- Coverage tooling and metrics exist today to understand usage
- Limitation: Symbol based usage coverage (names of the API elements)
- Falls short of detecting the varied types of interactions done by consumers
- Asserting the usage; making maintainability decisions (i.e. refactoring, deprecation); prioritizing test / documentation; of a piece of software is not possible.
- Necessity: Understanding the types of interactions
- Library developers may be unaware of interactions done in the real world; unexpected or unintended. Documentation may be incomplete / lacking.

API Coverage

- Differences exist between the view of an API designer (Intended Uses) and the implementation of the developer (Factual Uses).
- Directly impacts the Actual Uses by Clients, Samples, and Tests
- We define the Syntactic Usage Model (SUM) and the Syntactic Usage Footprint (SUF) in order to help the library developers and designers have a better understanding of the uses; highlight issues; bring into focus specific work areas; as a basis for conversation on library development.
- Detection of unexpected uses, lacking/incorrect documentation/samples is possible to observe and discuss thanks to these models



SUM & SUFs

- SUMs allow determining all current possible interactions of the API symbols
- SUFs allow determining all current interactions performed by Clients, Tests, Samples
- It is possible to differentiate two widely different uses

1 3 3
```
ArrayList<Integer> lst = new ArrayList<Integer>(2)
lst.add(42);
lst.add(1337);
```

```
public class MyArrayList<E> extends ArrayList<E> 1
    @Override
    public boolean add(E e) { ... } 2
}
```

API

```
public class ArrayList<E> implements List<E>
{
  public boolean add(E e) { ... }
  private int size;
  ...
}
```

SUM

Symbol	Interaction
`public class ArrayList<E>`	*Referenced*
	Instantiated
	Extended
`public ArrayList<E>()`	*Invoked*
`public boolean add(E e)`	*Invoked*
	Overridden

SUF 1

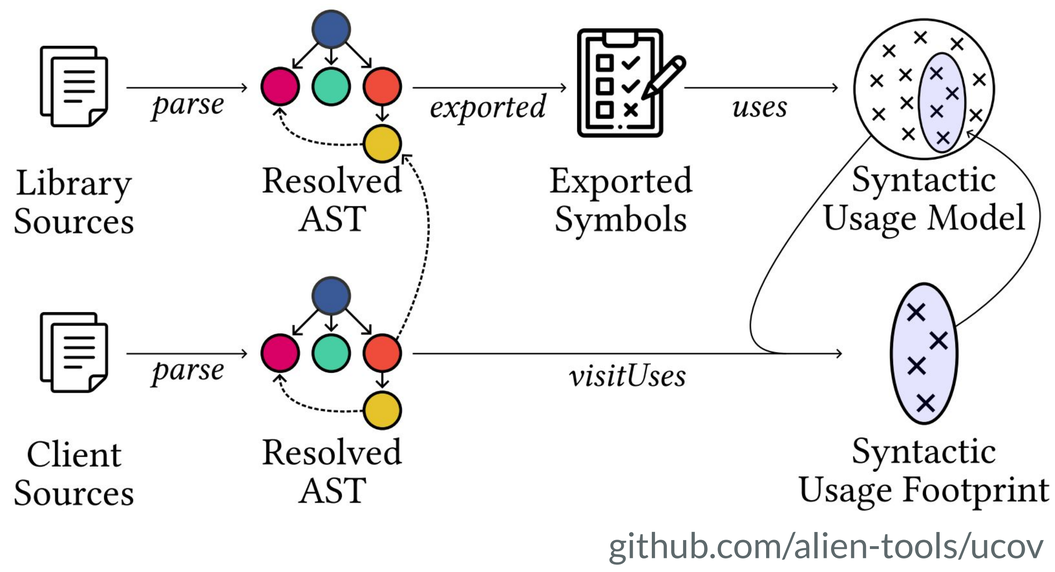
Symbol	Interaction
`public class ArrayList<E>`	*Referenced* 1
	Instantiated 2
`public ArrayList<E>()`	*Invoked* 2
`public boolean add(E e)`	*Invoked* 3

SUF 2

Symbol	Interaction
`public class ArrayList<E>`	*Extended* 1
`public boolean add(E e)`	*Overridden* 2

UCov

- UCov is static analysis program to gather the SUMs and SUFs for Java libraries
- UCov takes the library source code, extracts its API, and builds a SUM using a purpose built usage model
- UCov then builds the SUF from the original SUM and the client / sample / test source code



github.com/alien-tools/ucov

Results

- We want to explore UCov on libraries with clients, samples and tests available

Clients + **Samples** + **Tests**

- We want to study the results on three different library programming styles
- We think different styles results in different overall usage interactions

Classical

```
//commons.apache.org
//proper/commons-cli/usage.html

CommandLineParser parser = new
DefaultParser();
Options options = new Options();
options.addOption("a", "all",
  false, "do not hide entries");
options.addOption("C", false,
  "list entries by columns");
```

Fluent

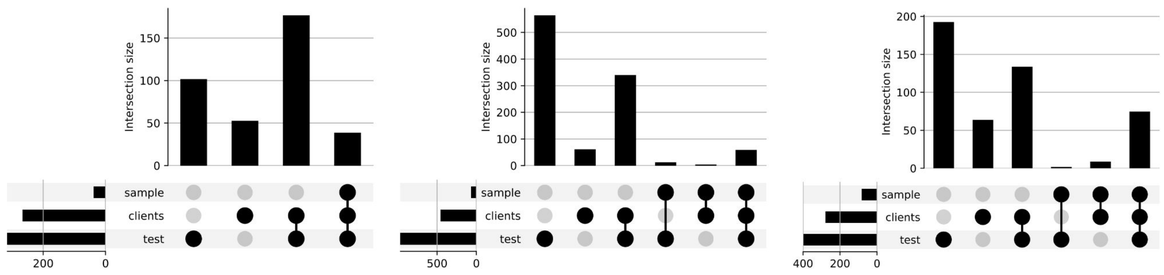
```
//jsoup.org/cookbook
//input/load-document-from-url

Document doc = Jsoup
  .connect("http://example.com")
  .data("query", "Java")
  .userAgent("Mozilla")
  .cookie("auth", "token")
  .timeout(3000)
  .post();
```

Framework

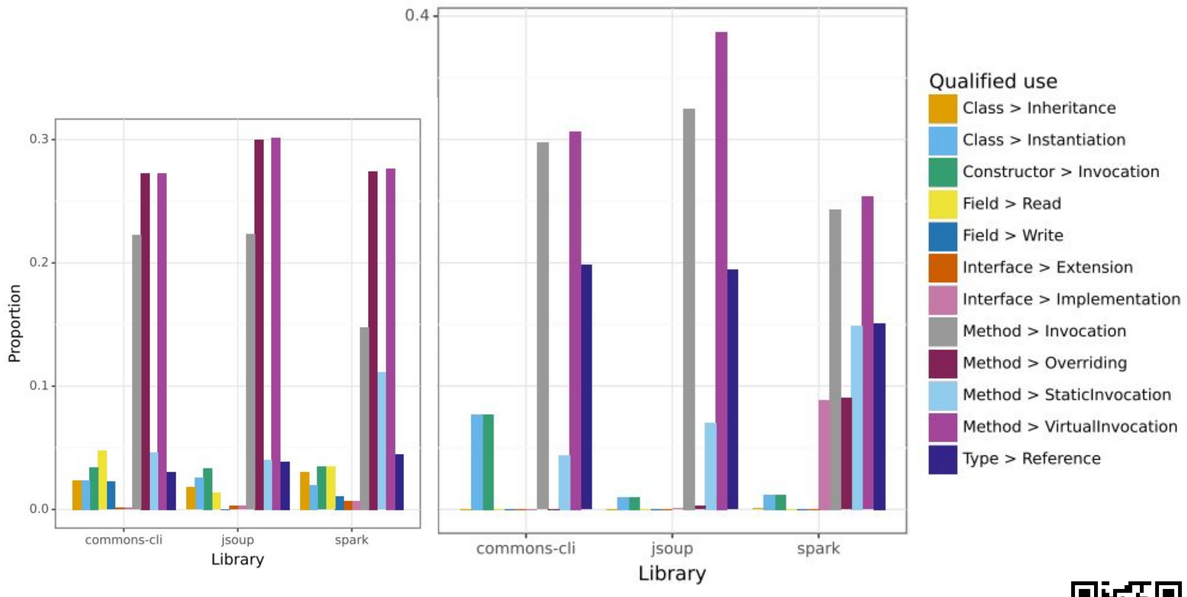
```
//sparkjava.com
//documentation#routes

get("/", (rq,rs) → { ... });
put("/", (rq,rs) → { ... });
post("/", (rq,rs) → { ... });
```

commons-cli **jsoup** **sparkjava**

The three libraries chosen for our exploratory case study



UpSet plots depicting the common and unique uses between third-party clients, tests, and samples for commons-cli, jsoup, and spark



As a distribution of the legal uses in their APIs (SUM)

As a distribution of the actual uses in client code (SUF)

zenodo.org/records/10571867

References

Gustave Monce, Thomas Couturou, Yasmine Hamdaoui, Thomas Degueule, Jean-Rémy Falleri. Lightweight Syntactic API Usage Analysis with UCov. 32nd IEEE/ACM International Conference on Program Comprehension (ICPC 2024), Apr 2024, Lisboa, Portugal. ⟨10.1145/3643916.3644415⟩. ⟨hal-04463475⟩