Server Interface Descriptions for Automated Testing of JavaScript Web Applications

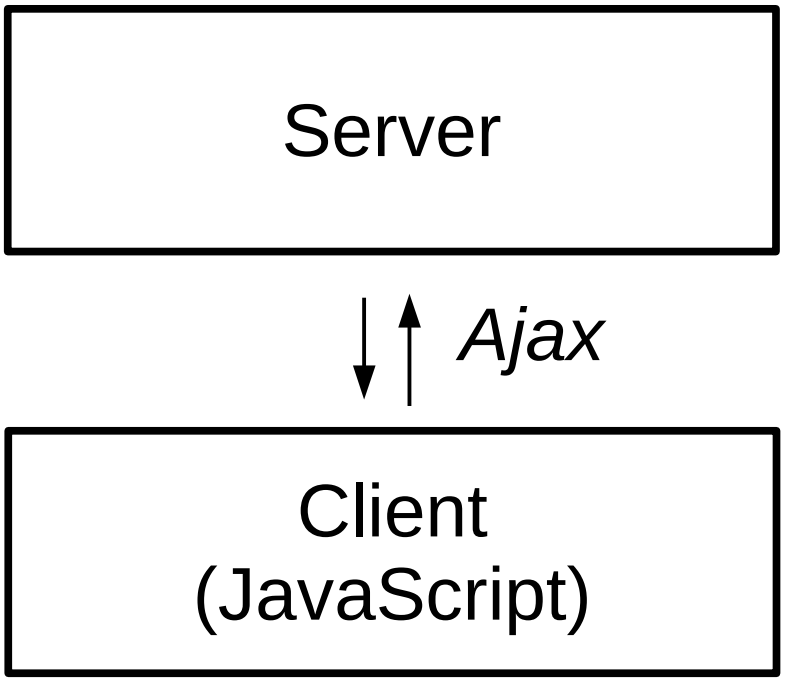
Casper S. Jensen
Aarhus University

Anders Møller
Aarhus University

Zhendong Su
University of California, Davis

St. Petersburg, ESEC/FSE 2013

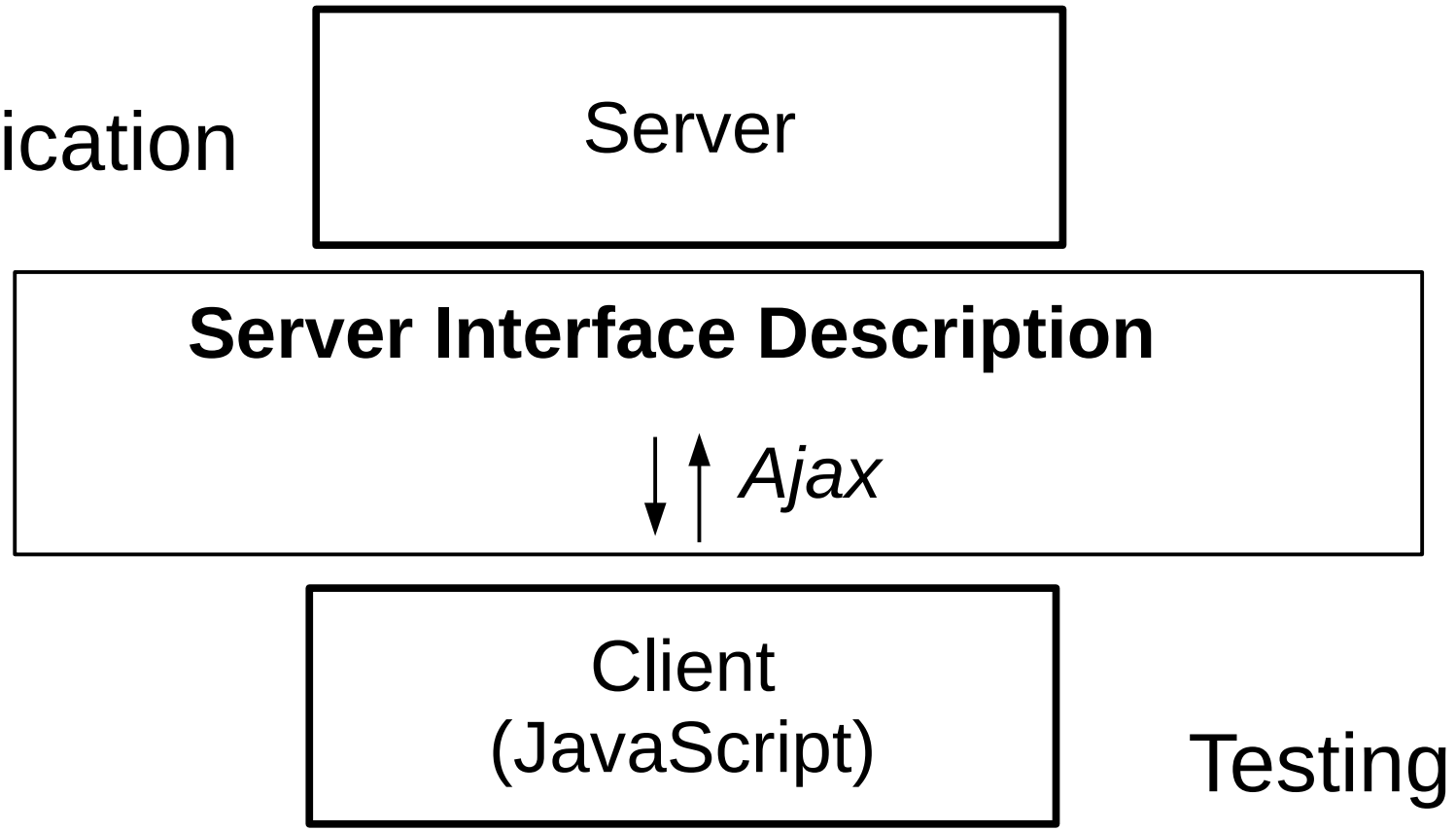
Web 2.0 Application Development
Today, the monolithic approach



Web 2.0 Application Development
Separation of Concerns

Program Comprehension

Verification



Testing

Partial Implementations

Outline

1) Advocate separation of concerns for web application communication
2) Illustrate how separation of concerns improves automated testing of JavaScript
3) Provide a learning algorithm for server interface descriptions

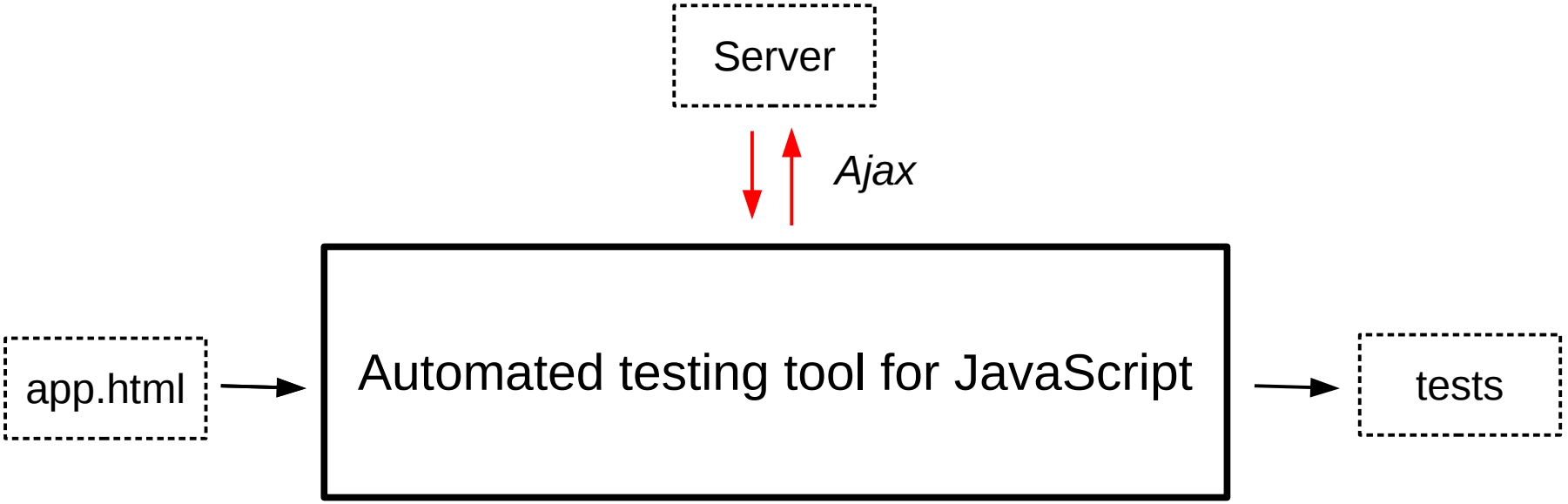
Outline

1) Advocate separation of concerns for web application communication
2) Illustrate how separation of concerns improves automated testing of JavaScript
3) Provide a learning algorithm for server interface descriptions

Automate Testing for JavaScript Applications

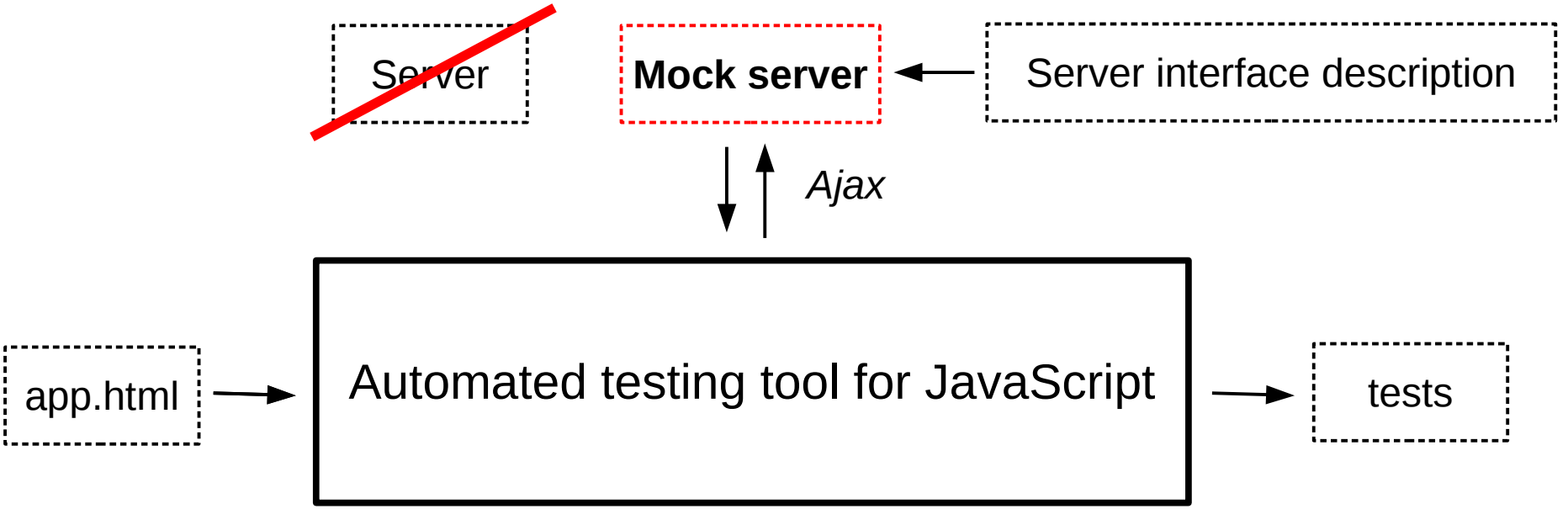
- Feedback-directed testing
  - Feedback from concrete execution generates tests
- Artemis: *A Framework for Automated Testing of JavaScript Web Applications. S. Artzi, J. Dolby, S. H. Jensen, A. Møller, and F. Tip. ICSE'11*

We Have a Problem...



- Requires the server to be available
- Time consuming to prepare server for testing
  - Software installation, database population, …

Solution



Add a mock server

- Uses server interface descriptions
- Responds with well structured responses

The AIL Language

Ajax server Interface description Language

(Simplified version of WADL)

```
GET news/read/id/*() : @items.json
GET author(name:*) : @author.json
GET users/login(user:*, pwd:*) : @token.json
POST news/update(item:@item.json) : void
```

Evaluation

Automated testing using a mock server

	Live server # lines covered	Mock server # lines covered
simpleajax	62	62
resume	108	113
globetrotter	180	205
buggenie	1322	1308
elfinder	1337	1366

Mock- and live server results in similar coverage

Outline

1) Advocate separation of concerns for web application communication
2) Illustrate how separation of concerns improves automated testing of JavaScript
3) Provide a learning algorithm for server interface descriptions

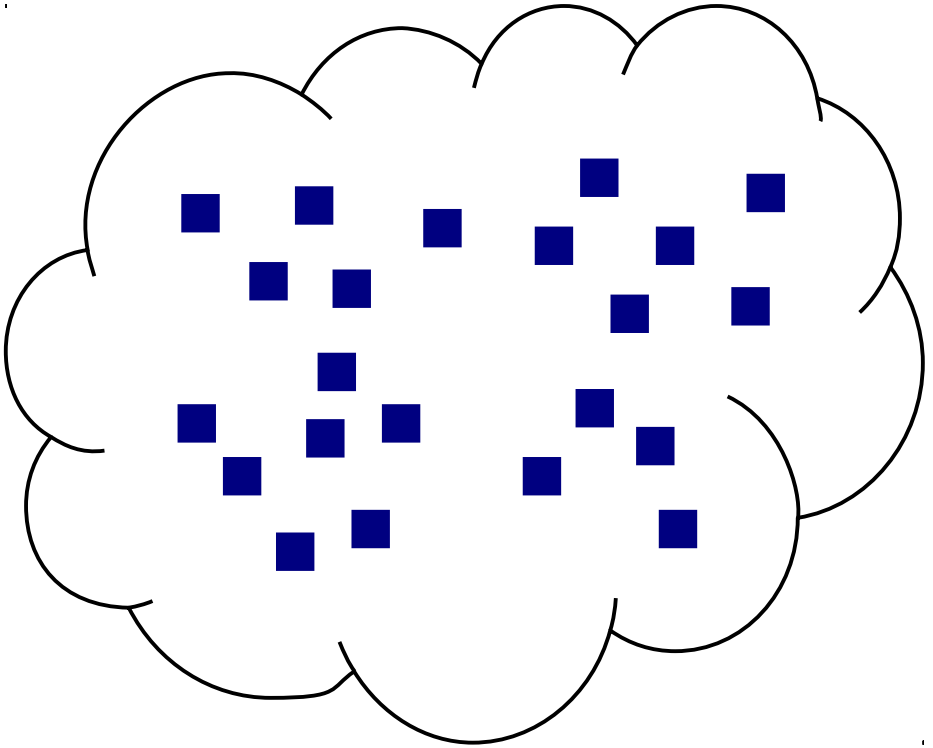
Why Learning?

- Writing server interface descriptions is time consuming
- A lot of existing applications could benefit from server interface descriptions
- We want to facilitate the construction of server interface descriptions

The Learning Process

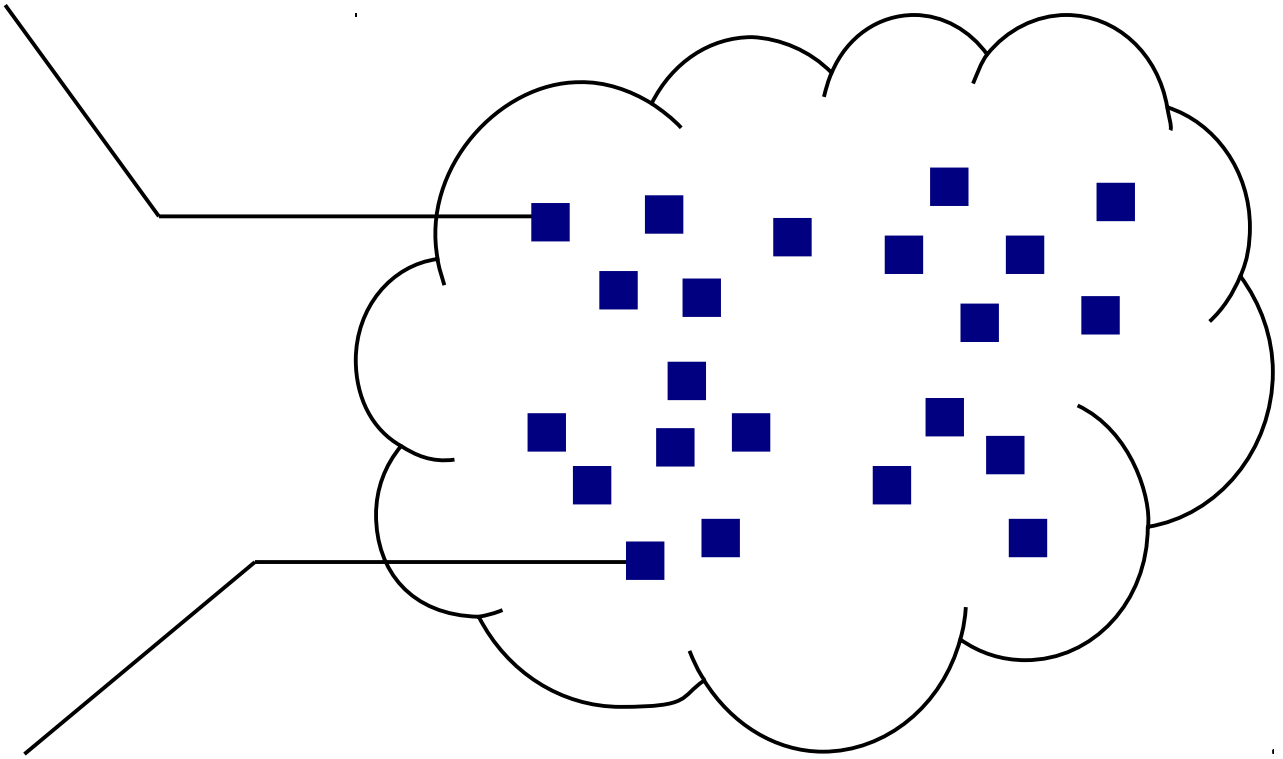
- Learning through samples
  - Monitor traffic in production
  - Black-box
  - A sample is a request and response pair
- Learning algorithm
- Subsequent manual inspection/adjustment
- Continuous maintenance as application evolves

Partitioning Samples



Partitioning Samples

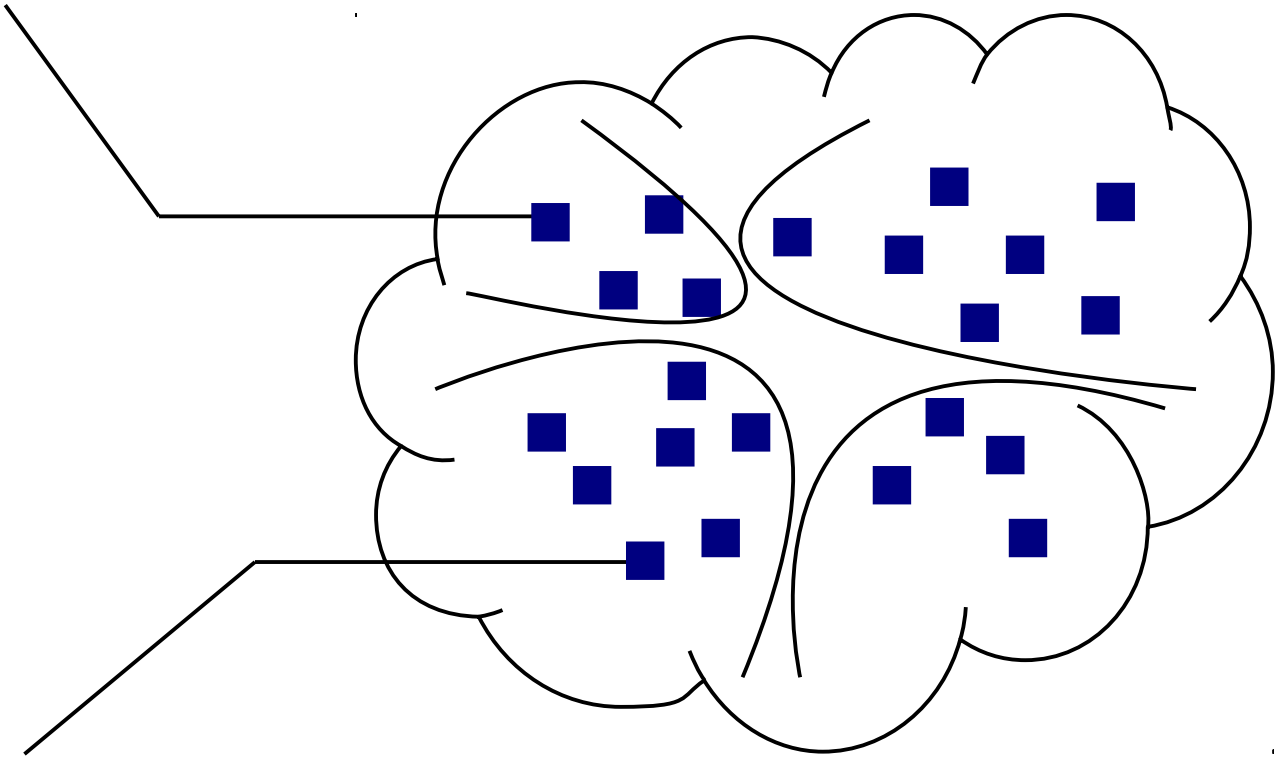
```
GET news/read/id/23
{“title”: “News item #23”}
```



```
GET news/delete/id/23
{“success”: “ok”}
```

Partitioning Samples

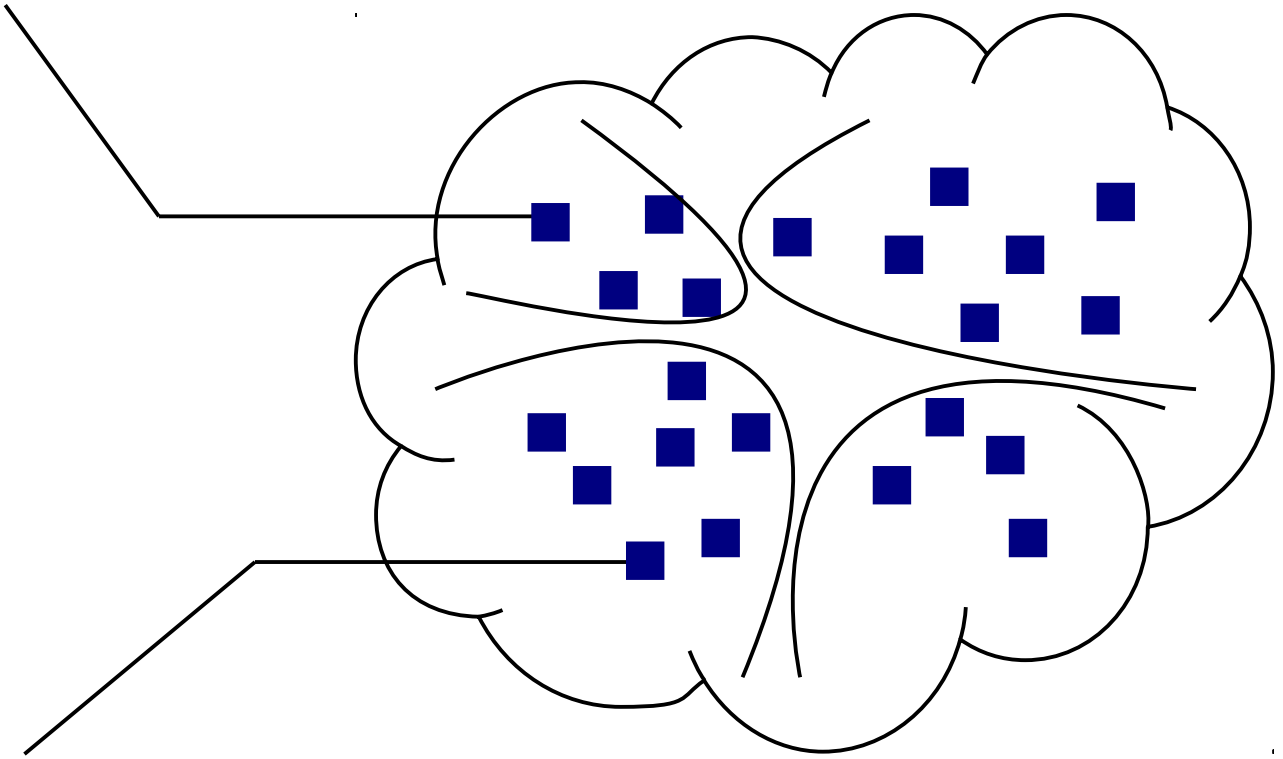
```
GET news/read/id/23
{“title”: “News item #23”}
```



```
GET news/delete/id/23
{“success”: “ok”}
```

Partitioning Samples

```
GET news/read/id/* : items.json
```



```
GET news/delete/id/* : result.json
```

Requirements to Partitioning
(Partitioning = Interface Description)

- Complete
  - The input is covered by the learned result
- Disjoint
  - Request patterns are disjoint
- Precise
  - Learned result close to input
- Concise
  - Learned result is small
    (measured in number of operations)

Algorithm (high-level)

1) Collect concrete request response pairs
2) Response data clustering
3) Request data clustering
4) Translation into AIL descriptions

Response Data Clustering

- Observation: For each operation, the possible responses are often structurally similar
- We map all responses to their *type abstraction*
  - We focus on JSON (XML is similar)
  - {“a”: 4, “b”: 5} → OBJ(“a”: INT, “b”: INT)
- Cluster according to structural equivalence of response type abstractions

Request Data Clustering

- The *response data clustering* does not satisfy the disjointness requirement
- Adjust the clustering considering request data
  - Combine clusters (generalize)
  - Split clusters (specialize)

Evaluation

Quality of learned specifications

	Samples	Sampling	Matches	1 → N	N → 1
simpleajax	70	3m	5	0	0
resume	128	9m	12	3	0
globetrotter	97	8m	4	1	0
impresspages	179	6m	5	1	0
buggenie	210	6m	4	3	0
elfinder	181	6m	11	3	0
tomatocart	370	8m	22	6	1
eyeos	611	6m	22	0	0
Total			**85**	**17**	**1**

85 out of 103 learned operations match the implementation

Summary

- Advocate for server interface descriptions
- Practical usage of server interface descriptions in automated testing of JavaScript
- Learning algorithm for server interface descriptions

Thanks