Karlsruhe Institute of Technology

Future Challenges for Linked APIs

Steffen Stadtmüller, Sebastian Speiser, Andreas Harth

SALAD Workshop, Montpellier, May 26th 2013

INSTITUTE OF APPLIED INFORMATICS AND FORMAL DESCRIPTION METHODS (AIFB)

Agenda

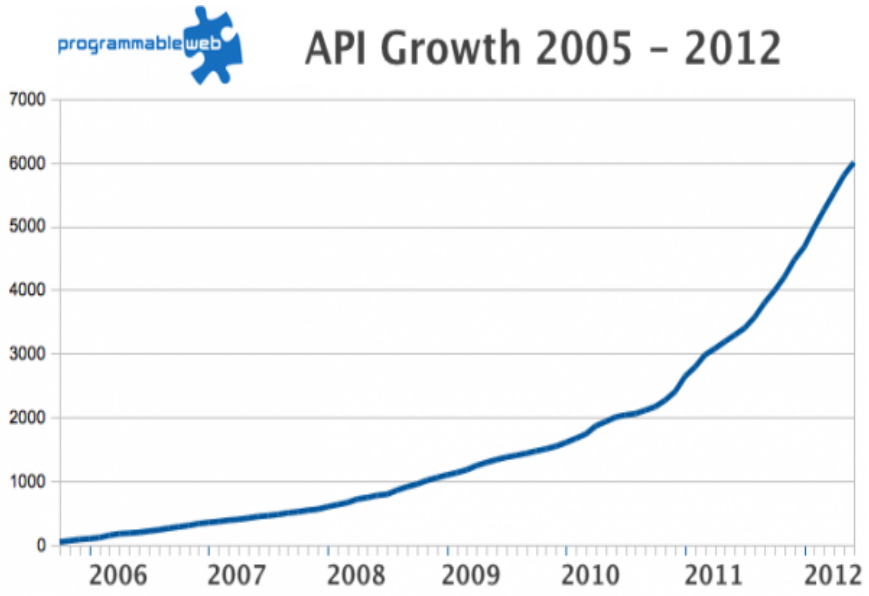
- Motivation
- Linked APIs
  - Descriptions
  - Interaction
  - Wrapping
- Identifying Challenges
  - Survey
- LAPIS Catalogue
- Conclusion

MOTIVATION

Motivation

- Data is often dynamically created as a result of some calculation carried out over input data (e.g., weather information)
- Data can change frequently (e.g., moving objects)
- Service endpoints, forms and APIs are used to trigger functionalities in the Web and the real world and provide access to dynamic and static data sources

- An important role plays **Representational State Transfer (REST)**
  - Focused on the Web architecture



[1]http://programmableweb.com

Resource-driven Programming with REST

- A resource is anything with which a client is able to interact
  - Data object (e.g., a blog post)
  - Real world object (e.g., car, movie, person…) projected onto the Web by making the information associated with it (i.e., its state) accessible
  - HTTP Verbs as methods to interact with resources

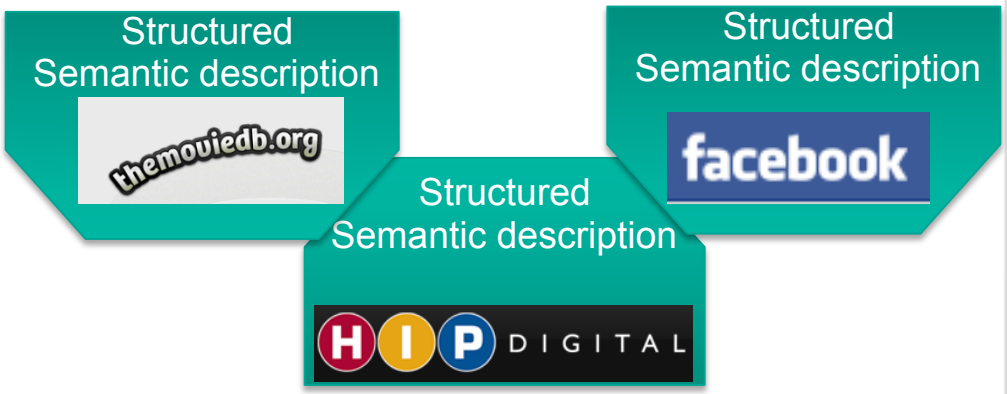
- Representations of the resources include links to other relevant resources
  - E.g. the representation of a person on a social network contains links to it's friends
  - Used by clients for the interaction
  - Clients discover the links during run time (late binding), which enables flexible evolution of services and data sources

Challenges to Address

- REST allows service providers to use arbitrary formalisms to represent resources and links
  - ➡ Developers have to gain a deep understanding of every API by reading textual descriptions
- Applications (clients) are supposed to follow links as found during runtime of the application. However, developers have to define their desired interaction at design time
  - ➡ Developers have to write individually tailored code to consume services in applications

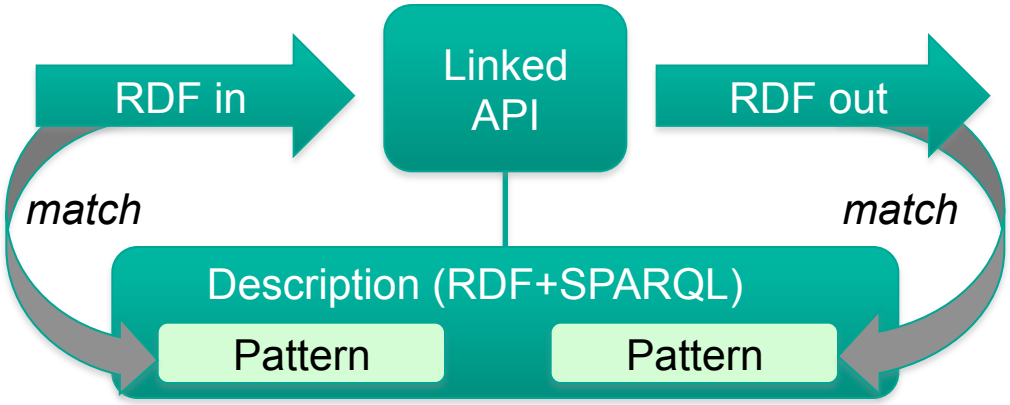
Benefits of Structured Semantic Descriptions

- Increased value comes from combinations of services and APIs
  - Structured service/API descriptions ease the composition process considerably and allow to execute several tasks automatically (e.g., data matching, discovery, repair)



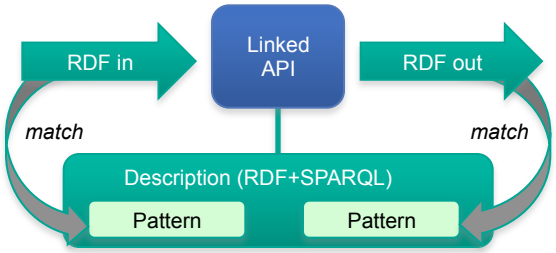
LINKED APIS

Linked API Architecture



- LAPIS bring together REST and Linked Data
- Resource representation in RDF
- LAPIS consume and produce RDF data
- LAPIS are described with graph pattern
  - Representing the structure of input and output data
  - Accessible in the Web

Linked API URI

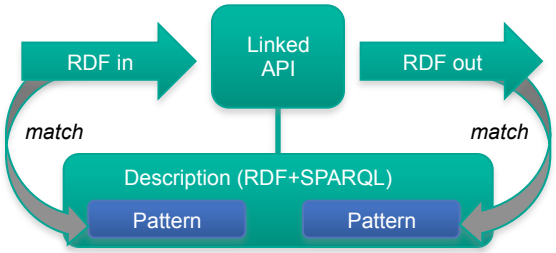


- As an example consider a RESTful movie service:
  - Ordering a movie is possible at an entry URI:

http://service.org/Movie/order

(identifies the set of all movie orders)

Linked API Description



- As an example consider a RESTful movie service:
  - Ordering a movie is possible at an entry URI:

    http://service.org/Movie/order

  - **Input** and **output description** for ordering a movie:

In:

```
?x a dbp:movie.
?x dc:name ?name.
```

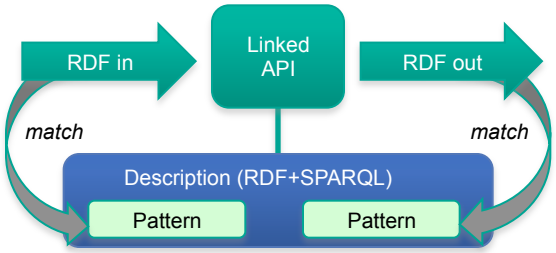
„A movie and its name“

Out:

```
?y a mov:Order.
?y db:content ?x.
?y ex:price ?p.
```

„An order, its content and its price“

Linked API Description



- As an example consider a RESTful movie service:
  - Ordering a movie is possible at an entry URI:

http://service.org/Movie/order

  - Input and output description:

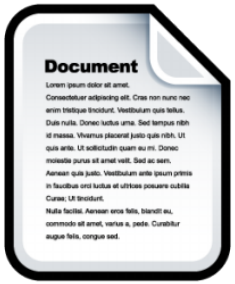
In:

```
?x a dbp:movie.
?x dc:name ?name.
```

Out:

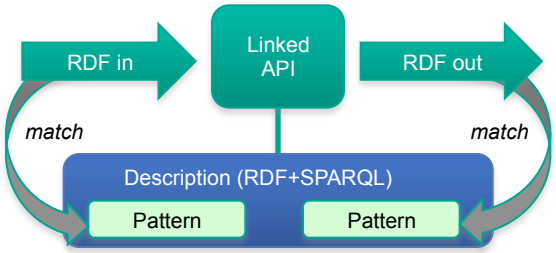
```
?y a mov:Order.
?y db:content ?x.
?y ex:price ?p.
```

Embedded description document



Linked API Description

- As an example consider a RESTful movie service:
  - This service is identified with the URI:

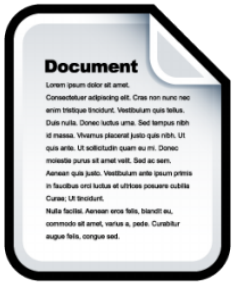


http://service.org/Movie/order

HTTP OPTIONS
Accept: text/N3

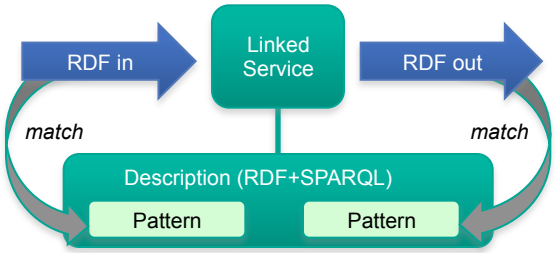
200 (OK)
Allow: OPTIONS, POST

The description can be retrieved via OPTIONS, where also the HTTP verb is given.



Linked API Invocation (POST)

- Service Execution via HTTP POST:
  - POST RDF data that matches the input pattern to the service resource
  - The service response adheres to the output pattern



http://service.org/Movie/order

HTTP POST

```
dbp:Blade_Runner a dbp:movie;
dc:name “Blade Runner”.
```

match

In:

```
?x a dbp:movie.
?x dc:name ?name.
```

Response

201 created
Location: mov:001

```
mov:001 a mov:Order.
mov:001 db:content dbp:Blade_Runner.
mov:001 ex:price “10€”.
```

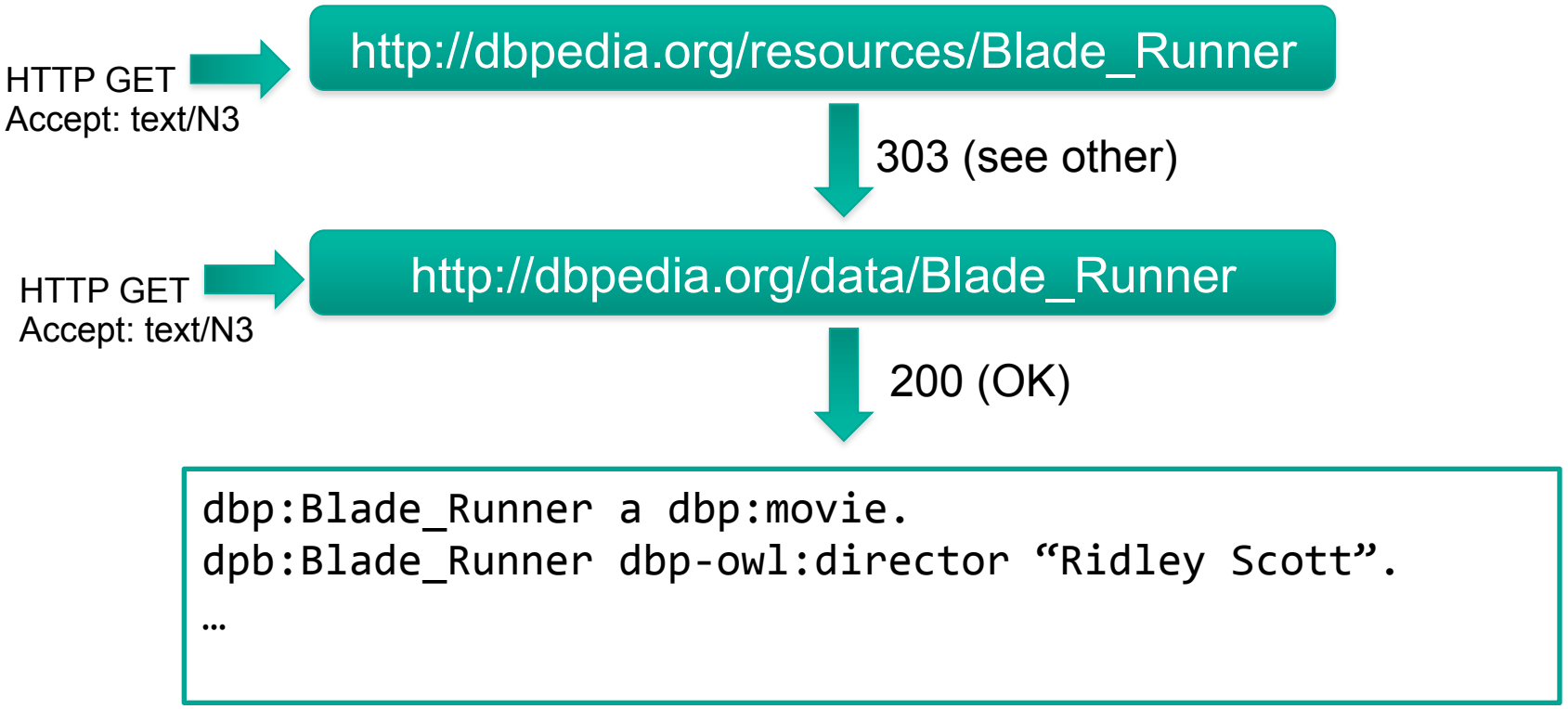
match

Out:

```
?y a mov:Order.
?y db:content ?x.
?y ex:price ?p.
```

Linked API Invocation (GET)

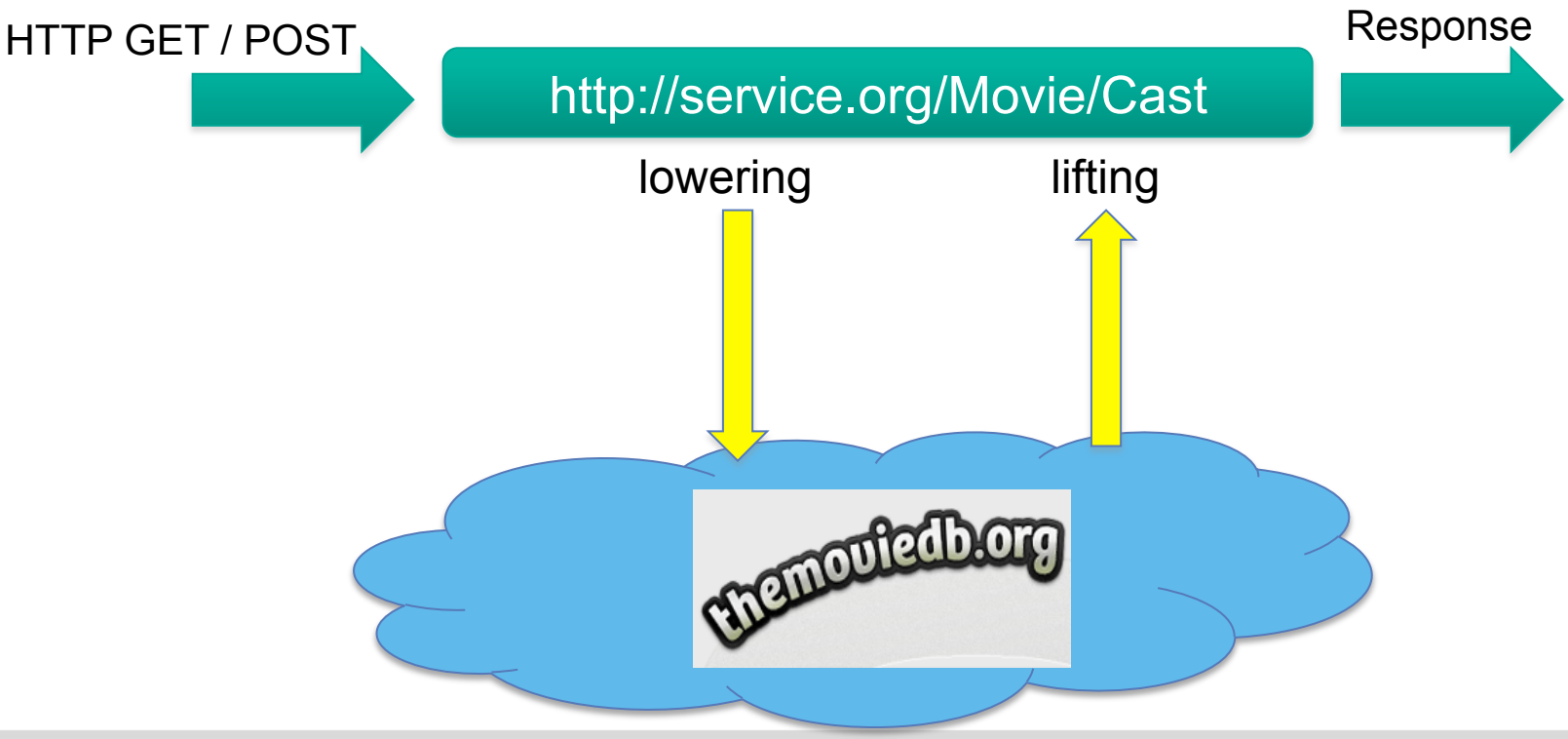
- Linked Data implements GET on resources by design:



Leveraging Existing Services

- Existing Web APIs can be wrapped to consume and produce Linked data

  ➡ valid Linked API



Benefits at a Glance

- Easy data integration due to Linked Data
- Capability to evolve dynamically due to REST
- High degree of automation possible with semantic descriptions

IDENTIFYING CHALLENGES

Survey Overview

- 20 undergraduate students
  - some programming experience
  - new to programming with Web APIs (REST)
- Task:
  - Develop at least one Linked API
  - Develop an application that makes use of at least two Linked APIs
  - Time: 4 months
  - Students are allowed to leverage existing not Linked Data-based APIs to create wrapper
  - Report about their experience
- Not a representative survey, but empirical indicators

Identified Problems

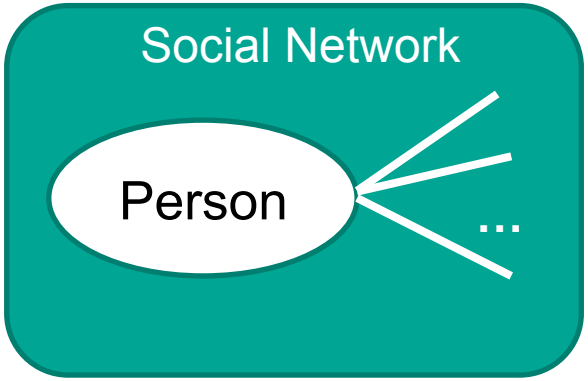
- Clustered and summarised reports:

Problem	# Students (n=20)
Response time of composed API	14
API limitations	14
Missing directories	12
No standard formalism for API descriptions	8
Complexity of RDF	3

- Similar problems supports the claim of recurring issues

Response Time and General Limitations

- Reasons for *slow response time (14 students):*
  - Response time of underlying wrapped API
  - Necessary time for the interaction between APIs and multiple API calls
  - E.g., GET all information about the friends of a person on a social network



- *API limitations (14 students)* refers to insufficient functionality and constraints of the underlying APIs
  - E.g., a maximum number of API calls per day
  - Usually external circumstances (e.g., business aspects)
  - *10 of the 14 students tried to replace the initially considered API*

Missing Directories and Description Formalism

- *Missing directories (12 Students)*
  - Identification of suitable APIs for use in an application
  - Replacement of not functional APIs
  - API development (to create links to other relevant resources)

- *No standard formalism for API descriptions (8 Students)*
  - How to serialise the graph pattern (embedded in RDF vs. direct N3)
  - A vocabulary for the description
  - Minimal set of properties to describe (e.g., input and output data)
  - A way of attaching description to an API resource (HTTP OPTIONS vs. HTTP Header vs. link in resource representation).

Next Challenges and Rewards

- A common standard minimal description mechanism for Linked APIs based on graph patterns
- Methods for an automated identification and comparison of APIs that leverage the descriptions
- The development of methods and systems to enable a scalable interaction and composition of Linked APIs

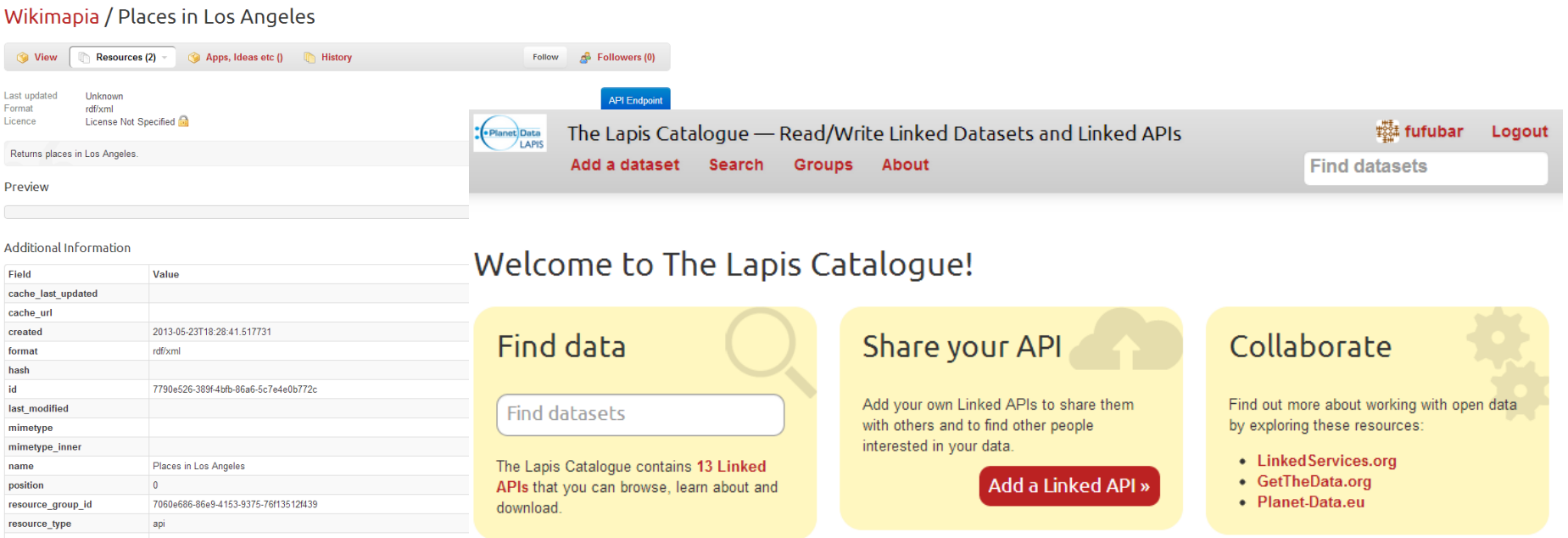
- Benefits of using Linked APIs:

Benefit	# Students (n=20)
Easy data integration	17
High modularity of composed APIs	14
Simplicity of interaction with API	9

LAPIS CATALOGUE

LAPIS Catalogue[1] Overview

- Open directory, where providers can register Linked APIs
  - Based on CKAN
  - Information like name, URI, author, license, maintainer, existing links and example calls



[1]http://lapis.planet-data.eu/
[2]http://ckan.org/

LAPIS Catalogue Purpose

- The LAPIS catalogue serves as hub for Linked APIs to support researchers, developers and providers in their tasks:
  - Evaluation of approaches related with Linked APIs
  - Survey the current developments and the adoption of approaches
  - Search Linked APIs for application development
  - Promote APIs for public use
  - Complementing the offered functionality of APIs (interlinking)

- Function between Datahub[1] and ProgrammableWeb[2]

[1]http://datahub.io/
[2]http://www.programmableweb.com/

CONCLUSION

Summary

- Challenges to address:
  - A common standard minimal description mechanism
  - Methods for an automated identification and comparison
  - Methods and systems to enable a scalable interaction and composition of Linked APIs

- Benefits to gain:
  - Easy Data Integration
  - High modularity
  - Simplicity of use

Thank You

Summary

- Challenges to address:

Problem	# Students (n=20)
Response time of composed API	14
API limitations	14
Missing directories	12
No standard formalism for API descriptions	8
Complexity of RDF	3

- Benefits to gain

Benefit	# Students (n=20)
Easy data integration	17
High modularity of composed APIs	14
Simplicity of interaction with API	9