Give Agents Some REST:
A Resource-oriented Abstraction Layer for Internet-scale Agent Environments

Andrei Ciortea¹  Olivier Boissier¹  Antoine Zimmermann¹  Adina Magda Florea²

¹Univ. Lyon, MINES Saint-Étienne, CNRS Lab Huber Curien UMR 5516, France
²Computer Science Department, University “Politehnica” of Bucharest, Romania

5th International Workshop on Engineering Multi-Agent Systems
May 9th, 2017
1. The World Wide Web was designed to be an Internet-scale and long-lived system [Fielding, 00].
The World Wide Web: An Internet-scale and Long-lived Software System

Stateless communication + **uniform interface** => Intermediaries

*Representational State Transfer (REST)* [Fielding, 00]

⇒ Scalability and evolvability
1. The World Wide Web was designed to be an Internet-scale and long-lived system [Fielding, 00].

2. Most existing “RESTful APIs” are not, in fact, REST-compliant [Fielding, 08].
   - REST is an architectural style for distributed hypermedia systems [Fielding, 00].
     • No hypermedia => no REST
Hypermedia As The Engine Of Application State (HATEOAS)

HATEOAS is a **core tenet** of REST and the modern Web architecture [Fielding,00].

Hypermedia-driven interaction enables *loose coupling*:

- URIs, possible transitions, and requests to implement those transitions are never hardcoded into clients

⇒ components can be developed, deployed, and can evolve *independently*
Give Agents Some REST: A Resource-oriented Abstraction Layer for Internet-scale Agent Environments

Screenshot from: http://www.google.com
Give Agents Some REST: A Resource-oriented Abstraction Layer for Internet-scale Agent Environments

Give Agents Some REST: A Resource-oriented Abstraction Layer for Internet-scale Agent Environments

Local guidance: hypermedia (HATEOAS)
Global guidance: the buyer’s goal

Reference Documentation

These are the REST API endpoint reference docs.

GET

- GET account/settings
- GET account/verify_credentials
- GET application/rate_limit_status
- GET blocks/ids
- GET blocks/list
- GET collections/entries
- GET collections/list
- GET collections/show
- GET direct_messages
- GET direct_messages/events/list
- GET direct_messages/events/show
- GET direct_messages/sent
- GET direct_messages/show
- GET direct_messages/welcome_messages/list

1. The World Wide Web was designed to be an Internet-scale and long-lived system [Fielding, 00].

2. Most existing “RESTful APIs” are not, in fact, REST-compliant [Fielding, 08].
   - REST is an architectural style for distributed hypermedia systems [Fielding, 00].
     - No hypermedia => no REST

3. Existing approaches to engineer Web-based MAS are misaligned with the Web architecture.
   - Most approaches use the Web as a transport layer
     - E.g., FIPA Agent Message Transport Protocol for HTTP Specification [FIPA, 02], approaches based on WS-* Web services
1. The World Wide Web was designed to be an Internet-scale system.

2. Most existing "RESTful APIs" are not, in fact, REST-compliant [Fielding, 08].

   - REST is an architectural style for distributed hypermedia systems [Fielding, 00].
   - No hypermedia => no REST

3. Existing approaches to engineer Web-based MAS are misaligned with the Web architecture.
   - Most approaches use the Web as a transport layer
     - E.g., FIPA Agent Message Transport Protocol for HTTP Specification [FIPA, 02], approaches based on WS-* Web services
1. The World Wide Web was designed to be an Internet-scale and long-lived system [Fielding, 00].

2. Most existing “RESTful APIs” are not, in fact, REST-compliant [Fielding, 08].
   - REST is an architectural style for distributed hypermedia systems [Fielding, 00].
     - No hypermedia => no REST

3. Existing approaches to engineer Web-based MAS are misaligned with the Web architecture.
   - Most approaches use the Web as a transport layer
     - E.g., FIPA Agent Message Transport Protocol for HTTP Specification [FIPA, 02], approaches based on WS-* Web services
   - More recent approaches use the Web as an application Layer
     - … but no hypermedia (and no hypermedia-driven interaction)!
How can we build MAS that **inherit the architectural properties** of the World Wide Web? (e.g., Internet-scalability, evolvability)
Motivation

Major challenges for future research on engineering MAS are to design *agent environments* that [Weyns et al., 15]:

- support large-scale MAS;
- can cope with open MAS in which components are deployed and evolve independently from one another at runtime;
- support humans-in-the-loop.
In a Nutshell

**Objective:** To engineer MAS that are Internet-scale, open, and support humans-in-the-loop.

**Approach:** Reuse the design rationale behind the modern Web architecture to address the three above challenges in an integrated manner.

**Novelty:** Apply REST and use HATEOAS to design the agent environment as a distributed hypermedia application.

⇒ piggyback on the Web architecture and get all its properties for free
Outline

• Hypermedia-driven Agent Environments
• Socio-technical Networks (STNs)
• A Uniform Interface for STN-based Agent Environments
• Proof of Concept: The Wake-up Call
• Conclusions
Hypermedia-driven Agent Environments

Agent environments designed as **distributed hypermedia applications** in conformance to the REST architectural style.

- Given an **entry point** into the environment, agents should be able to *seamlessly navigate* and *interact with* the environment.

**Core idea**: Introduce a **resource-oriented** and **hypermedia-driven** abstraction layer that decouples the *application environment* from its *deployment context*. 
Hypermedia-driven Agent Environments

**Problem:** Enable *software clients* (e.g., software agents, MAS platforms, Web browsers) to **discover** and **interface** with components in the deployment context at runtime.

- Achieve a **uniform interface** between the application environment and its deployment context.

**Key ingredients:** REST + **STN model** + linked data
Outline

• Hypermedia-driven Agent Environments
• Socio-technical Networks (STNs)
• A Uniform Interface for STN-based Agent Environments
• Proof of Concept: The Wake-up Call
• Conclusions
Socio-technical Networks (STNs)

Dynamic networks of **humans**, **software agents**, and **artifacts** interrelated in a **meaningful** manner via **typed relations** (e.g., friendship, ownership, provenance, colocation) [Ciortea et al., 15].

Agents (both human and software) can “rewire” the networks in pursuit of their goals.

An STN is reflected in the digital world through **digital artifacts** that can be distributed across multiple **platforms**.
From STNs to Web Resources

All entities in an STN are mapped to Web resources.

- real-world entities (e.g., people, physical devices) \(\Rightarrow\) non-information resources
- digital artifacts \(\Rightarrow\) information resources

The state of an STN is represented using the *STN ontology*\(^1\).

\(^{1}\)http://w3id.org/stn
From STNs to Web Resources

All entities in an STN are mapped to Web resources.
- real-world entities (e.g., people, physical devices) → non-information resources
- digital artifacts → information resources

The state of an STN is represented using the *STN ontology*\(^1\).

\(^1\)http://w3id.org/stn
From STNs to Web Resources

All entities in an STN are mapped to Web resources.

- real-world entities (e.g., people, physical devices) → non-information resources
- digital artifacts → information resources

The state of an STN is represented using the STN ontology\textsuperscript{1}.

\textsuperscript{1}http://w3id.org/stn
Outline

- Hypermedia-driven Agent Environments
- Socio-technical Networks (STNs)
- A Uniform Interface for STN-based Agent Environments
- Proof of Concept: The Wake-up Call
- Conclusions
A Uniform Interface for STN-based Agent Environments

• Uniform identification of entities
• Manipulation of digital artifacts via representations
• Self-descriptive messages
• Hypermedia-driven interaction
Uniform Identification of Entities

All entities in an STN are *uniformly identified* through **URIs**

- If platform-specific identifiers are used, all context information has to be encapsulated within the digital artifact reference

```sparql
@prefix stn: <http://w3id.org/stn/core#> .

<http://home1.example.org/david#me> a stn:Person ;
  stn:holds <http://home1.example.org/david> .
  stn:holds [ a stn:UserAccount ;
    stn:hostedBy <http://facebook.example.org/#platform> ;
    stn:id "1550387481863557" ] .

<http://home1.example.org/david> a stn:UserAccount ;
  stn:name "David Doe" ;
  stn:description "An IoT enthusiast!" ;
  stn:connectedTo <http://home2.example.org/bob> .
```
Uniform Identification of Entities

All entities in an STN are uniformly identified through URIs

- If platform-specific identifiers are used, all context information has to be encapsulated within the digital artifact reference

```
@prefix stn: <http://w3id.org/stn/core#> .

<http://home1.example.org/david#me> a stn:Person ;
    stn:holds <http://home1.example.org/david> .
    stn:holds [ a stn:UserAccount ;
                stn:hostedBy <http://facebook.example.org/#platform> ;
                stn:id "1550387481863557" ] .

<http://home1.example.org/david> a stn:UserAccount ;
    stn:name "David Doe" ;
    stn:description "An IoT enthusiast!" ;
    stn:connectedTo <http://home2.example.org/bob> .
```

Dereferencing an entity’s URI should return something useful:

- if the entity is a digital artifact, an RDF serialization of its current state;
- otherwise, an RDF serialization of a digital artifact that represents or describes the entity
Manipulation of Digital Artifacts via Representations

Clients interact with STN platforms by exchanging representations of digital artifacts.

- A *representation* is an RDF serialization of the current or intended state of an artifact.
  - State transfer
  - No direct access to resources
Self-descriptive Messages

In REST, the semantics of messages are defined through standard methods and representation formats (a.k.a. media types).

The states of digital artifacts can be represented in RDF:

- using standard RDF serialization formats, or
- new serialization formats that fit domain- and application-specific needs.

... but most existing Web APIs don’t use RDF.

- A simple mapping language for extracting RDF data from heterogeneous JSON representations:

```xml
<#facebookAccountJSONMapping> a stn-ops:Representation;
   stn-ops:mediaType stn-http:JSON;
   stn-ops:entityType stn:UserAccount;
   stn-ops:contains [ a stn-http:Mapping;
       stn-http:key "id";
       stn-http:STNTerm stn:id;
   ];
```
Self-descriptive Messages

In REST, the semantics of messages are defined through standard **methods** and **representation formats** (a.k.a. **media types**).

The states of digital artifacts can be represented in RDF:

- using **standard RDF serialization formats**, or
- **new serialization formats** that fit domain- and application-specific needs.

... but most existing Web APIs don’t use RDF.

- A simple mapping language for extracting RDF data from heterogeneous JSON representations:

  ```
  <#facebookAccountJSONMapping> a stn-ops:Representation;
  stn-ops:mediaType stn-http:JSON;
  stn-ops:entityType stn:UserAccount;
  stn-ops:contains [ a stn-http:Mapping;
    stn-http:key "id";
    stn-http:STNTerm stn:id;
  ] .
  ```
Self-descriptive Messages

In REST, the semantics of messages are defined through standard **methods** and **representation formats** (a.k.a. **media types**).

The states of digital artefacts can be represented in RDF:

- using **standard RDF serialization formats**, or
- **new serialization formats** that fit domain- and application-specific needs.

... but most existing Web APIs don’t use RDF.

- A simple mapping language for extracting RDF data from heterogeneous JSON representations:

  ```xml
  <#facebookAccountJSONMapping> a stn-ops:Representation ;
  stn-ops:mediaType stn-http:JSON ;
  stn-ops:entityType stn:UserAccount ;
  stn-ops:contains [ a stn-http:Mapping ;
    stn-http:key “id” ;
    stn-http:STNTerm stn:id ;
  ] .
  ```
Hypermedia-driven Interaction

Typed relations are represented explicitly in the agent environment in a *uniform manner* in RDF using the STN ontology.

STNs enhance hypermedia-driven interaction through:

- the *social network metaphor* => discovery of agents;
- the *stn:holds* relation => discovery of user accounts held by agents in various STNs;
- the *stn:hostedBy* relation => discovery of platforms and any APIs they may expose.

```
  <#getAccount> a stn-ops:GetUserAccount ;
  stn-ops:implementedAs [ a stn-http:AuthSTNRequest ;
    http:methodName "GET" ;
    http:requestURI "/:id?fields=id,name,website" ;
  ] ;
  stn-ops:hasRequiredInput 
  [ a stn-ops:UserAccountID ;
    stn-http:key ":id" ;
    stn-http:paramIn stn-http:Path;
  ] ;
  stn-ops:hasOutput <#fbAccountJSONMapping> .
```
Outline

• Hypermedia-driven Agent Environments
• Socio-technical Networks (STNs)
• A Uniform Interface for STN-based Agent Environments
• Proof of Concept: The Wake-up Call
• Conclusions
Proof of Concept: The Wake-up Call

David switched to night mode!

Yeah, he’s asleep!
Proof of Concept: The Wake-up Call

David switched to night mode!

David has a meeting in 1h and he’s sleeping!

Yeah, he’s asleep!
Proof of Concept: The Wake-up Call

Vibration alarm!

Artificial light!

Who can wake up David?

Natural light!

Yeah, he’s asleep!
Proof of Concept: The Wake-up Call

No confirmation that David woke up, searching for friends...

World Wide Web
Proof of Concept: The Wake-up Call

- JaCaMo [Boissier et al., 13]
- STN Platform\(^1\), Twitter Public API\(^2\), Facebook Graph API\(^3\)
- Philips Hue, Texas Instruments SensorTag, Californium\(^4\), Californium Tools\(^5\)

\(^1\)https://github.com/andreiciortea/stn-platform
\(^2\)https://dev.twitter.com/rest/public
\(^3\)https://developers.facebook.com/docs/graph-api
\(^4\)https://github.com/eclipse/californium
\(^5\)https://github.com/eclipse/californium.tools
Proof of Concept: The Wake-up Call

1. David’s agent connect to one another via his home STN.

- JaCaMo [Boissier et al., 13]
- STN Platform\(^1\), Twitter Public API\(^2\), Facebook Graph API\(^3\)
- Philips Hue, Texas Instruments SensorTag, Californium\(^4\), Californium Tools\(^5\)

\(^1\)https://github.com/andreiciortea/stn-platform
\(^2\)https://dev.twitter.com/rest/public
\(^3\)https://developers.facebook.com/docs/graph-api
\(^4\)https://github.com/eclipse/californium
\(^5\)https://github.com/eclipse/californium.tools
Proof of Concept: The Wake-up Call

2. David’s agents attempt to wake him up.

- JaCaMo [Boissier et al., 13]
- STN Platform¹, Twitter Public API², Facebook Graph API³
- Philips Hue, Texas Instruments SensorTag, Californium⁴, Californium Tools⁵

¹https://github.com/andreiciortea/stn-platform
²https://dev.twitter.com/rest/public
³https://developers.facebook.com/docs/graph-api
⁴https://github.com/eclipse/californium
⁵https://github.com/eclipse/californium.tools
Proof of Concept: The Wake-up Call

3. The calendar agents crawl David’s distributed social graph to discover friends that are awake.

- JaCaMo [Boissier et al.]
- STN Platform\(^1\), Twitter Public API\(^2\), Facebook Graph API\(^3\)
- Philips Hue, Texas Instruments SensorTag, Californium\(^4\), Californium Tools\(^5\)

---

\(^1\)https://github.com/andreiciortea/stn-platform
\(^2\)https://dev.twitter.com/rest/public
\(^3\)https://developers.facebook.com/docs/graph-api
\(^4\)https://github.com/eclipse/californium
\(^5\)https://github.com/eclipse/californium.tools
Proof of Concept: The Wake-up Call

4. The calendar agents sends messages to David’s friends via Twitter.

- JaCaMo [Boissier et al., 2013]
- STN Platform\(^1\), Twitter Public API, Facebook Graph API
- Philips Hue, Texas Instruments SensorTag, Californium\(^4\), Californium Tools\(^5\)

\(^1\)https://github.com/andreiciortea/stn-platform
\(^2\)https://dev.twitter.com/rest/public
\(^3\)https://developers.facebook.com/docs/graph-api
\(^4\)https://github.com/eclipse/californium
\(^5\)https://github.com/eclipse/californium.tools
Proof of Concept: The Wake-up Call

The hypermedia-driven agent environment is deployed over multiple heterogeneous Web services (e.g., Twitter, Facebook) and constrained devices.

– **Entry point:** David’s IRI.

**Internet-scalability:** the application environment uses semantic descriptions discovered and interpreted at runtime in order to interface with Twitter, Facebook, and the STN platforms.

**Openness:** given David’s IRI as an entry point, the calendar agent is able to discover and interact with other software agents or with humans available in the system at runtime.

**Humans-in-the-loop:** the calendar agent is able to cooperate with humans via Twitter in order to prevent David from missing his event.
Conclusions

The World Wide Web provides a suitable middleware for MAS that are Internet-scale, open, and support humans-in-the-loop.

We have demonstrated an approach to engineer Web-based MAS using hypermedia and STNs.

Similar to how the Web enables the discovery of Web pages, STNs enable the discovery of agents and artifacts through the Web.
Thanks!

**Poster Session B:** Wednesday, 14:30 – 17:30  
**Poster Session C:** Thursday, 10:20 – 11:20

`andrei.ciortea@emse.fr`
References


