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

(Extended Abstract)

Andrei Ciortea, Olivier Boissier, Antoine Zimmermann
Univ. Lyon, MINES Saint-Étienne
CNRS Lab Hubert Curien UMR 5516
F-42023 Saint-Étienne, France
{andrei.ciortea, olivier.boissier, antoine.zimmermann}@emse.fr

Adina Magda Florea
Department of Computer Science
University "Politehnica" of Bucharest
313 Splaiul Independentei
060042 Bucharest, Romania
adina.florea@cs.pub.ro

ABSTRACT

In this paper, we claim that the World Wide Web provides a suitable middleware for Internet-scale, open, and human-centric multi-agent systems. The novelty of our approach is to design the *agent environment* as a *hypermedia application.* We apply REST, the architectural style of the Web, to introduce a resource-oriented abstraction layer that decouples the *application environment* from its *deployment context.* Higher-level environment abstractions can then be implemented on top of this lower-level abstraction layer. To demonstrate our approach, we implemented a multi-agent application for the Internet of Things in which software agents can seamlessly navigate, use, and cooperate in an environment deployed over multiple Web services (e.g., Facebook, Twitter) and constrained devices.

Keywords

Agent environments; service-oriented computing; REST; Web architecture; Web of Things; Internet of Things

1. MOTIVATION

Over the last decade, the agent environment has gained broad recognition as a *first-class abstraction* in multi-agent systems (MAS) [15]: it is a key component designed and programmed with clear-cut responsibilities, such as mediating interaction among agents and access to the deployment context (e.g., physical devices, digital services). Recently, three research topics were identified as major challenges to future research on agent environments [14]: to design agent environments that (i) support *large scale systems*, (ii) can cope with *open systems* in which components are deployed and evolve at runtime, and (iii) *support humans in the loop.*

If we are to examine existing software systems based on the above criteria, the World Wide Web is arguably the most scalable, versatile, and human-centric software system deployed on the Internet. In fact, the Web was specifically designed to be an Internet-scale and long-lived system in which components can be deployed and can evolve independently from one another at runtime [9]. Our claim is that we can apply the design rationale behind the Web architecture to engineer agent environments that address the three above-mentioned challenges in an integrated manner.

Even though there has been extensive research on integrating multi-agent systems (MAS) with the Web, to the best knowledge of the authors, existing approaches to engineer Web-based MAS do not conform to the architectural style of the Web, known as *Representational State Transfer (REST)* [8]. Most approaches make limited use of the Web as a *transport layer*, which includes the work influenced by WS-* Web services [13, 12, 11] or the FIPA specification for using HTTP as a message transport protocol [10]. More recent approaches adhere to some of the REST principles and use the Web as an *application layer*, but they do not address one of the core tenets of REST and the Web architecture: using *hypermedia as the engine of application state* (a.k.a. the HATEOAS constraint)[1] (see [9] for further details).

We propose a novel approach that uses HATEOAS to design agent environments as *hypermedia applications.* To the best knowledge of the authors, this proposal is the first approach to engineer MAS that are completely aligned with the Web architecture.

2. RESOURCE-ORIENTED AGENT ENVIRONMENTS

Our approach is to apply REST to design a *resource-oriented*[2] abstraction layer for agent environments that decouples the *application environment*[3] from its *deployment*

[1]A *hypermedia application* can be modeled as a finite state machine. HATEOAS implies that, in any given application state, software clients have to retrieve *at runtime* from origin servers what are the next reachable application states *and* how to transition to those states (e.g., via HTTP requests). Hypermedia provides the *transmission medium* for carrying the communication between software clients and origin servers.

[2]A *resource* is the key abstraction of information in REST, where "any information that can be named is a resource" [8].

[3]That is to say, the part of the environment designed and programmed for the application at hand [15].

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

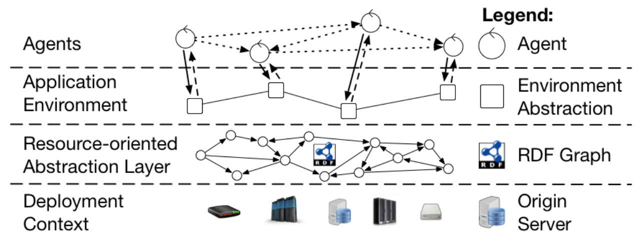

Figure 1: Layers of abstraction in hypermedia-driven agent environments.

context (e.g., physical devices, digital services), as illustrated in Figure 1. Higher-level *environment abstractions* – which agents can use, for instance, to interact with other agents or to search for resources in their environment – are then implemented on top of this lower-level abstraction layer. Environment abstractions would typically be programmed and run using MAS platforms (e.g., JaCaMo [6]). For MAS platforms that do not support environment abstractions (e.g., JADE [3]), the proposed abstraction layer can remain hidden behind the agents.

Similar to how the Web works, the resource-oriented abstraction layer uses *hypermedia* and HATEOAS to enable software clients (e.g., software agents, MAS platforms, Web browsers) to *discover* and *interface* with components in the deployment context *at runtime*. To address the HATEOAS constraint, our approach relies on *socio-technical networks (STNs)*, that is dynamic networks of humans, software agents, and artifacts interrelated in a meaningful manner via *typed relations* (e.g., friendship, ownership, provenance, colocation) [1]. Having typed relations represented explicitly in the agent environment supports *hypermedia as the engine of agent environment state* and enables dynamic discoverability in open and distributed MAS. Human and software agents can "navigate" the distributed agent environment in an informed manner, and they can "rewire" their networks in pursuit of their goals.

All entities in an STN are mapped to Web resources identified through URIs [4] and described in RDF [7] using the *STN ontology*[4], a Web ontology for describing STNs and their *hosting platforms* (called hereafter *STN platforms*). By using hypermedia, Web standards, and a shared ontology, this approach achieves a *uniform interface* between software clients and STN platforms in the deployment context. The uniform interface then allows components to be deployed and to evolve independently from one another.

3. IMPLEMENTATION AND EVALUATION

To demonstrate our approach, we implemented an STN-based agent environment for the Internet of Things (IoT) that is deployed over multiple constrained devices and Web services (i.e., Facebook, Twitter, and multiple instances of an STN platform[5]). The experiment setup is depicted in Figure 2. Software agents and their application environments were implemented with JaCaMo [6], and were deployed independently from one another.

In this application scenario, each person owns a *digital*

[4]`http://w3id.org/stn/`

[5]See [2] for technical details on the STN platform.

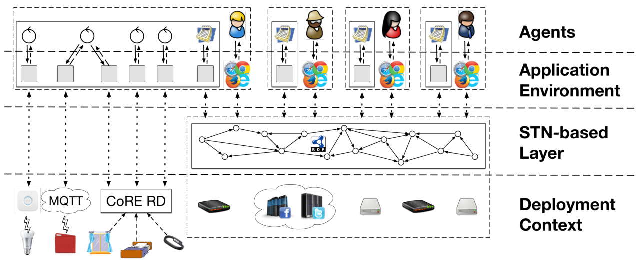

Figure 2: Experiment setup: An STN-based agent environment deployed over Facebook, Twitter, four STN platforms, and multiple constrained devices (i.e., Philips Hue, TI SensorTag, CoAP devices).

calendar agent that can use various IoT devices (e.g., wristbands, window curtains, light bulbs) to keep track of their owner's sleeping schedule. If the owner is asleep and there is an important upcoming event scheduled, the calendar agent attempts to wake him up (via sound alarms, opening the curtains etc.). If unsuccessful, the calendar agent then crawls its owner's distributed social graph (e.g., on Facebook and Twitter) to search for friends that can wake him up. In doing so, the agent has to discover and interact with other calendar agents to find out which friends are not asleep, and then *delegate* the task to those friends via Twitter. The human agents in our scenario can thus use any Web browser or Twitter's mobile application to interact with the MAS.

The application environments are decoupled from Facebook, Twitter, and the STN platforms via semantic descriptions created using the STN ontology: they are agnostic to the underlying infrastructure, which is central to engineering *Internet-scale MAS*. Software agents "living" in these environments are able to discover and interact with other agents at runtime, which is essential in *open MAS*. Software and human agents work together towards a common goal, and the human agents are *brought into the loop* at runtime. Therefore, the proposed approach addresses all three research challenges that motivate our work in an integrated manner.

4. CONCLUSIONS

Most software systems today revolve around the Web, and this huge success is owed to its key architectural properties. In this paper, we claimed that the Web provides a suitable middleware for agent environments that (i) support large scale systems, (ii) can cope with open systems, and (iii) support humans in the loop, three important and current research topics in the engineering of agent environments [14]. The novelty of our approach is to apply REST to design agent environments as *hypermedia applications*. We use STNs to support humans in the loop and to address the HATEOAS constraint in REST. By analogy with how the Web enables the discovery of Web pages, STNs enable the discovery of agents and artifacts on the Web.

We consider the impact of the contribution presented in this paper to be twofold: (i) it addresses important research topics in engineering MAS, and (ii) it enables the transfer of MAS technology to the development of Web-based systems in a manner that is completely aligned with the Web architecture. We consider the latter to be an important step towards achieving the original Semantic Web vision [5].

REFERENCES

[1] Andrei Ciortea, Antoine Zimmerman, Olivier Boissier, Adina Magda Florea. Towards a social and ubiquitous Web: A model for socio-technical networks. In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 1, pages 461–468. IEEE, 2015.

[2] Andrei Ciortea, Olivier Boissier, Antoine Zimmerman, Adina Magda Florea. Responsive decentralized composition of service mashups for the Internet of Things. In *Proceedings of the 6th International Conference on the Internet of Things (IoT)*. ACM, 2016.

[3] F. Bellifemine, A. Poggi, and G. Rimassa. JADE – A FIPA-compliant agent framework. In *Proceedings of the 5th International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, volume 99, page 33, 1999.

[4] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifier (URI): Generic Syntax. RFC 3986 (INTERNET STANDARD), Jan. 2005. Updated by RFCs 6874, 7320.

[5] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):28–37, 2001.

[6] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, and A. Santi. Multi-agent oriented programming with JaCaMo. *Science of Computer Programming*, 78(6):747–761, 2013.

[7] R. Cyganiak, D. Wood, and M. Lanthaler. RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation 25 February 2014. W3C Recommendation, World Wide Web Consortium (W3C), Feb. 25 2014.

[8] R. T. Fielding. *Architectural styles and the design of network-based software architectures.* PhD thesis, University of California, Irvine, 2000.

[9] R. T. Fielding and R. N. Taylor. Principled design of the modern Web architecture. *ACM Transactions on Internet Technology (TOIT)*, 2(2):115–150, 2002.

[10] Foundation for Intelligent Physical Agents. FIPA Agent Message Transport Protocol for HTTP Specification. http://www.fipa.org/specs/fipa00084/SC00084F.html, 2002. Document number: SC00084F.

[11] N. Gibbins, S. Harris, and N. Shadbolt. Agent-based Semantic Web services. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(2):141–154, 2004.

[12] M. N. Huhns. Agents as Web services. *IEEE Internet Computing*, 6(4):93, 2002.

[13] M. N. Huhns and M. P. Singh. Service-oriented computing: Key concepts and principles. *IEEE Internet Computing*, 9(1):75–81, 2005.

[14] D. Weyns, F. Michel, et al. Agent environments for multi-agent systems – a research roadmap. In *Lecture Notes in Computer Science*, volume 9068, pages 3–21. Springer, 2015.

[15] D. Weyns, A. Omicini, and J. Odell. Environment as a first class abstraction in multiagent systems. *Autonomous Agents and Multi-agent Systems*, 14(1):5–30, 2007.