Explicit Modelling of Resources for Multi-Agent MicroServices using the CArtAgO framework

Extended Abstract

Eoin O'Neill
University College Dublin
eoin.o-neill.3@ucdconnect.ie

David Lillis
University College Dublin
david.lillis@ucd.ie

Gregory M. P. O'Hare
University College Dublin
gregory.ohare@ucd.ie

Rem W Collier
University College Dublin
rem.collier@ucd.ie

ABSTRACT

This paper describes the first agent programming language agnostic implementation of the Multi-Agent MicroServices (MAMS) model - an approach to integrating agents within microservices-based architectures where agents expose aspects of their state as virtual resources, realised as CArtAgO artifacts, that are externally accessible through REpresentational State Transfer (REST).

KEYWORDS

agent programming; BDI agents

ACM Reference Format:
Eoin O'Neill, David Lillis, Gregory M. P. O'Hare, and Rem W Collier. 2020. Explicit Modelling of Resources for Multi-Agent MicroServices using the CArtAgO framework. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 3 pages.

1 INTRODUCTION

Microservices are increasingly seen as an important innovator in software design. They champion the decomposition of monolithic systems into loosely-coupled networks of services [17] that are necessary to deliver internet-scale applications [7]. This has the effect of reducing the complexity of many of the components, but comes at the cost of increasing the complexity of deployment [16]. However, this challenge has been met through the rise of DevOps [2] and Continuous Software Engineering methods [11].

The rise of microservices presents an opportunity for Multi-Agent Systems (MAS) research. As is illustrated in [18], there is a strong affinity between the principles of microservices and MAS that can be exploited to deliver innovations, both in terms of the use of MAS technologies with microservices and the use of microservices technologies with MAS. This affinity is reinforced in [14], which argues that microservices can be used to facilitate agility of development for agent-based Internet of Things (IoT) systems. This view is further reinforced in [10], which argues that microservices-based IoT systems can be modelled as agents, and in [9], which presents a multi-agent trust model for IoT.

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand. © 2020 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

This paper builds on previous work that introduced the *Multi-Agent MicroServices (MAMS)* model [18]; a model that promotes a view of agents as hypermedia entities whose body includes a set of virtual resources that can be interacted with using REpresentational State Transfer (REST) [8] and can be deployed as microservices. Overall, the work has three main objectives: to facilitate the seamless deployment of Multi-Agent Systems (MAS) within microservices ecosystems; to exploit modern industry tools to enhance the deployment of MAS; and ultimately, to enable the development of an emerging class of systems known as *Hypermedia MAS* [3][4].

2 MULTI-AGENT MICRO-SERVICES

As a concept, MAMS stems from the view that organisations combine outward-facing customer service agents, while the internal worker agents are hidden. The main impact of this technology is that it is now possible to develop packaged components with agent-level reasoning as standard practice. This would allow each component to be packaged as a Docker image and downloaded for use whenever necessary. Potentially, multiple agent-based containers hosted on a shared Docker machine could automatically become aware of each other through agent communication.

The goal of MAMS is to replicate the successes gained through applying the web model to services by applying them to MAS. For example, applying the principles of REST [8] to MAS can simplify interactions between agents and reduce the role of agent platforms as services become an integral part of the agent. This facilitates the move to simpler models, with URIs providing a simple and unambiguous global naming system for agents. As is illustrated in Figure 1, agents are associated with a set of virtual resources that are exposed as RESTful entities allowing seamless interaction with other agent (A) and non-agent (S) services.

Linking agents to virtual resources can have multiple applications. For example, they could represent the information that an agent needs to complete its internal or shared goal. or aspects of an agent's state that it wishes to share with other agents in order to influence community behaviour or to attract other agents in order to engage in collaborative activities.

A key benefit accruing from MAMS is the ease with which agent reasoning can be embedded into real-world scenarios. This lends itself the ability to monitor interactions between given entities and that service. This can imbue the service with introspective capabilities, making it aware of any interactions that may cause issues. In such scenarios, an agent can reason about the failure and select a relevant course of action to resolve the problem.

A benefit of this for the MS community is the fact that the idea of social interaction between small entities has been extensively studied by the MAS community. This allows many of these models to be implemented by the MS community in order to manage interactions between individual services in the achievement of a common goal. As described in [1] and [15], social norms become a key part of implementing a system that is composed of individual components. They provide a baseline of interaction between entities so that the system moves along a path that will eventually lead to the completion of the common system goal.

3 THE PROPOSAL

A core goal of the work presnted in this paper is the explicit modelling of resources that can be manipulated by agents. CArtAgO [13] introduces the concept of an artifact as environment-level entities that can be manipulated directly by agents. In this paper, we model resources as artifacts. Specifically, each agent is associated with an agent body, consisting of a set of CArtAgO artifacts that model the virtual resources associated with each agent. As is illustrated in Figure 2, a `base` artifact is provided as a shared base to which each `resource` artifact is linked and this in turn is linked to a shared `webserver` artifact that exposes the resources over HTTP.

Modelling resources explicitly as artifacts allows for clearly-defined semantics that includes both a description of how each HTTP verb will affect the state of the artifact modelling the associated resource and a specification of the interface between the agent and the resource, which is defined in terms of the operation, observable property, and signal concepts of CArtAgO. This engenders two models of resource management from the agents perspective:

- **Passive Resource Management**: As the artifact receives each request, depending on the HTTP verb used, the agent receives a CArtAgO signal indicating the nature of the update that was applied. This allows the agent to act in response to expected changes in the resources, but does not affect the speed by which the response returned to the system making the request. Additionally, the agent is also able to make changes to the state of the resources through a suite of internal operations. This allows rapid interaction between the resource and the entity making the request, whilst ensuring that the agent is still informed about the state of each resource. A key factor of this method is the fact that although

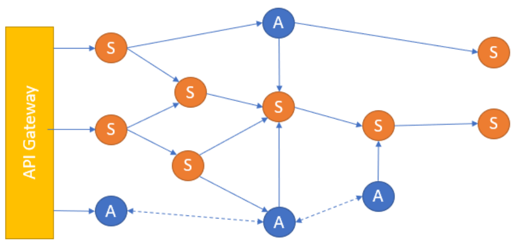

Figure 1: Agent/Service Integration

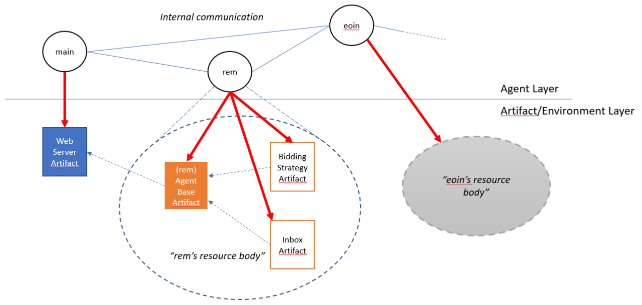

Figure 2: Modelling a RESTful agent body as artifacts

  the agent may have control over the resource, the resource is open to the world as an endpoint. This permits any service (or agent) to make a request and receive a timely response from this entity, something that may not be possible when you introduce the mentalistic aspect of deliberation that is associated with agents.

- **Active Resource Management**: Once a valid HTTP request is made of an artifact, a CArtAgO signal is generated based on the type of HTTP verb passed to the agent. For GET and DELETE requests, the request body is ignored. Conversely, the body is included for POST and PUT requests. This event is passed to the agent which then deliberates to decide on the correct response. If deemed acceptable, the agent executes the "accept" operation on the artifact. The request is then removed from the event queue and processed. A response detailing that the request made was handled correctly is issued. In the case where the request was rejected by the agent, the "refuse" operation is invoked, issuing a response to the requester that the request was denied.

For validation, a prototype has been developed that has been integrated with the ASTRA agent programming language [6][5][12]. The prototype and some examples are available at Gitlab.com[1].

4 CONCLUSIONS

This paper presents a novel approach to the implementation of MAMS that embraces current industry best practice and technology stacks and introduces the idea of virtual resources as a mechanism for facilitating the seamless integration of agents into microservices-based architectures. Through this, we gain access to a wealth of technologies and experience in how to deploy systems at scale while at the same time situating those agents in a larger web-enabled ecosystem. From a MS perspective, the benefits lie in the fact that they can now employ a level of reasoning within their systems, but also take advantage of the amount of research generated by the community since its inception.

ACKNOWLEDGMENTS

This research is funded under the SFI Strategic Partnerships Programme (16/SPP/3296) and is co-funded by Origin Enterprises Plc.

[1]https://gitlab.com/mams-ucd

REFERENCES

[1] Estefania Argente, Vicente Julian, and Vicente Botti. 2006. Multi-agent system development based on organizations. *Electronic Notes in Theoretical Computer Science* 150, 3 (2006), 55–71.

[2] Armin Balalaie, Abbas Heydarnoori, and Pooyan Jamshidi. 2016. Microservices architecture enables devops: Migration to a cloud-native architecture. *Ieee Software* 33, 3 (2016), 42–52.

[3] Andrei Ciortea, Olivier Boissier, and Alessandro Ricci. 2018. Engineering World-Wide Multi-Agent Systems with Hypermedia. In *6th International Workshop on Engineering Multi-Agent Systems (EMAS 2018).*

[4] Andrei Ciortea, Simon Mayer, Fabien Gandon, Olivier Boissier, Alessandro Ricci, and Antoine Zimmermann. 2019. A Decade in Hindsight: The Missing Bridge Between Multi-Agent Systems and the World Wide Web. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1659–1663.

[5] Rem W Collier, Seán Russell, and David Lillis. 2015. Reflecting on agent programming with AgentSpeak (L). In *International Conference on Principles and Practice of Multi-Agent Systems*. Springer, 351–366.

[6] Akshat Dhaon and Rem W Collier. 2014. Multiple inheritance in AgentSpeak (L)-style programming languages. In *Proceedings of the 4th International Workshop on Programming based on Actors Agents & Decentralized Control*. 109–120.

[7] Nicola Dragoni, Ivan Lanese, Stephan Thordal Larsen, Manuel Mazzara, Ruslan Mustafin, and Larisa Safina. 2017. Microservices: How to make your application scale. In *International Andrei Ershov Memorial Conference on Perspectives of System Informatics*. Springer, 95–104.

[8] Roy Thomas Fielding. 2000. *REST: Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[9] Kalliopi Kravari and Nick Bassiliades. 2019. StoRM: A social agent-based trust model for the internet of things adopting microservice architecture. *Simulation Modelling Practice and Theory* 94 (2019), 286–302.

[10] Petar Krivic, Pavle Skocir, Mario Kusek, and Gordan Jezic. 2017. Microservices as agents in IoT systems. In *KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*. Springer, 22–31.

[11] Rory V O'Connor, Peter Elger, and Paul M Clarke. 2017. Continuous software engineering-A microservices architecture perspective. *Journal of Software: Evolution and Process* 29, 11 (2017), e1866.

[12] Alessandro Ricci, Rafael Hector Bordini, Jomi F Hubner, and Rem Collier. 2018. Agentspeak (er): An extension of agentspeak (l) improving encapsulation and reasoning about goals. In *The 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018)*. International Foundation for Autonomous Agents and MultiAgent Systems (IFAAMAS).

[13] Alessandro Ricci, Mirko Viroli, and Andrea Omicini. 2006. CArtAgO: A framework for prototyping artifact-based environments in MAS. In *International Workshop on Environments for Multi-Agent Systems*. Springer, 67–86.

[14] Claudio Savaglio, Maria Ganzha, Marcin Paprzycki, Costin Bădică, Mirjana Ivanović, and Giancarlo Fortino. 2020. Agent-based Internet of Things: State-of-the-art and research challenges. *Future Generation Computer Systems* 102 (2020), 1038–1053.

[15] Munindar P Singh and Amit K Chopra. 2017. The internet of things and multiagent systems: Decentralized intelligence in distributed computing. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 1738–1747.

[16] Johannes Thönes. 2015. Microservices. *IEEE software* 32, 1 (2015), 116–116.

[17] Mario Villamizar, Oscar Garcés, Harold Castro, Mauricio Verano, Lorena Salamanca, Rubby Casallas, and Santiago Gil. 2015. Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. In *2015 10th Computing Colombian Conference (10CCC)*. IEEE, 583–590.

[18] Rem W Collier, Eoin O'Neill, David Lillis, and Gregory O'Hare. 2019. MAMS: Multi-Agent MicroServices. In *Companion Proceedings of The 2019 World Wide Web Conference*. ACM, 655–662.