Building Management using the Semantic Web and Hypermedia Agents

Eoin O'Neill[1], Katharine Beaumont[1], Nestor Velasco Bermeo[1] and Rem Collier[1]

[1]*UCD School of Computer Science, University College Dublin, Belfield, Dublin 4, Ireland*

Abstract

The following document describes the submission to the All The Agents Challenge of a system that interacts with the Built on Linked Data Server (BOLD) [1]. Our solution provides a Building Management Agent Oriented Micro Service (AOMS) built using Multi-Agent MicroServices (MAMS) technology in order to manage luminance levels on a per room basis based on the occupancy levels of each room. This solution provides showcases a collaboration between a multitude of different agents that work together in order to continuously monitor and interact with the server using REST, in accordance with updates applied to the system with each time tick.

Video: https://youtu.be/uriJT-kAVMg

Source Code: https://gitlab.com/mams-ucd/atac-bold

Keywords

Semantic Web, Semantic Agents, Intelligent Agents, Multi-Agent Microservices

1. Introduction

When Ciortea et. al [1] discussed the cross over between autonomous agents and the semantic web, they described agents being deployed in a semantically defined web environment that could be interacted with and reasoned about. This submission is a step towards achieving the link between autonomous agents and the web: a step towards hMAS[2]. It is concerned with the design of agents that are able to interact with the Building on Linked Data (BOLD) environment, store semantic information from this environment within Knowledge Stores that are particular to each agent, and reason about this data in order to enact changes on the system. Our solution is implemented as a Multi-Agent MicroServices (MAMS)[3] system that routinely polls the BOLD Server in order to check the status of each room.

Through the use of ontologies, agents have the ability to create an internal model of the building. By defining multiple ontologies we allowed the agents to adopt beliefs about different entities within the room, both static and dynamic values that change with each iteration of the simulation.

[1]https://all-agents-challenge.github.io/atac2021

All the Agents Challenge (ATAC 2021)

eoin.o-neill.3@ucdconnect.ie (E. O'Neill); katharine.beaumont@ucdconnect.ie (K. Beaumont); nestorvb@ucd.ie (N. V. Bermeo); rem.collier@ucd.ie (R. Collier)

1.1. Technologies

This system combines semantic web, multi-agent systems and web technologies. The overall system is build using the Multi-Agent MicroServices (MAMS) [3, 4, 5] architectural style. MAMS promotes the creation of systems that are comprised of Agent-Oriented MicroServices (AOMS) and Plain-Old MicroServices (POMS). In this approach, AOMS are comprised of one or more agents whose state is partially exposed through a REpresentational StateTransfer (REST) interface. In our system, the AOMS is the Building Management Service presented in Figure 1. This service interacts with the Built on Linked Data (BOLD) Server, a semantically defined building environment which models the IBM building in Dublin, Ireland. The agents in this system interact with the environment using REST to retrieve updates about the current state of the environment. This information is then incorporated into their logical reasoning apparatus.

The ASTRA programming language is used because BDI agents are both reactive and responsive to their environment. They continually receive events from the environment and update their beliefs, which are incorporated into flexible plans [6]. They offer significant advantages in developing autonomous systems and support the incorporation of a range of AI techniques [7].

Using ASTRA allows agents to take advantage of ASTRA's modularity [8]. ASTRA modules represent internal libraries, which allow for five kinds of annotation: terms, formulae, sensors, actions and events [8]. A single agent can create several copies of the same module, with different names and states [8]. The key module in the current system is the Jena Module, which acts both as a Knowledge Store for the agents and as an interface that connects agents to semantic web technology. The RDF Schema module allows the agent to identify the relationships between the different entities within the system in order to classify different parts of the building. This allows the agent to adopt beliefs about the environment which leads to action based on these entities. This is further discussed in 2.2.

2. System architecture

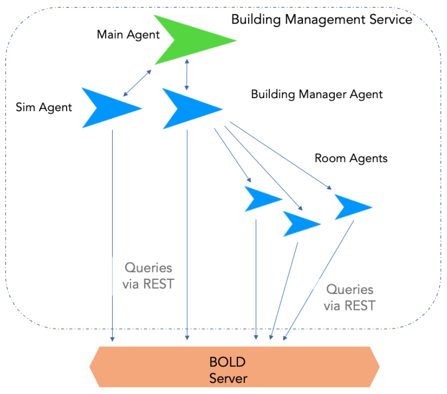

Figure 1: System Diagram

2.1. Building Management Service

The Building Management Service comprises of a number of BDI agents. A Main agent has initial beliefs concerning the sim url of the building and the building url: these are the BOLD Server endpoints. The Main agent starts the Sim agent and injects it with the sim url. The Sim agent iteratively queries the url, replacing the contents of the Knowledge Store with the results of each request.

During the first iteration, the Sim agent sends a message to the Main agent to inform it that the BOLD Server is running. The Main agent then creates the Building Manager which has the dual responsibility of mapping the building and creating Room agents for each room it finds. The building to be explored is passed by the Main agent.

To map the building, the Building Manager recursively iterates through the building using the **brickf:hasPart** predicate. The Building Manager uses it's Knowledge Store, referencing schema and maintaining a set of internal beliefs about the places that have been explored. Once complete, it queries the Knowledge Store returning the urls of all the rooms identified by the **rdf:type** predicate, where the object is the **brick:Room** concept. Once complete, the Building Manager creates a Room agent for each room, and initialises them with their individual room url.

Each Room agent then queries all of the resources that exist within it's room, searching for things using the **brickf:isLocatedIn** predicate. It maintains a set of beliefs about the properties associated with various things that it finds.

On successive iterations, the Room agents use these beliefs to capture the updated state of the building after each iteration. The Room agent is interesting because it maintains two RDF triple stores: a *static* one for triples that do not change; and a *dynamic* one for triples that do change (i.e. the properties of the things in the room). This allows us to minimise the number of GET requests required on each iteration.

The overall behaviour of the system is driven by the Sim agent which is responsible for detecting the start of each iteration of the simulator. Upon detection, the Sim agent informs all the Room agents via internal message passing which is routed via the Main agent and the Building agent. Upon receipt of this message, each Room agent updates their dynamic belief store by querying all the things that it knows of in the room. If the room is unoccupied but the lights are left on, the agent turns the light off by issuing a PUT request to the resource with the appropriate TURTLE content.

2.2. Jena Module (RDF Knowledge Store) and RDF Schema Module

A core component of the agents is the use of Apache Jena and modules. Apache Jena is an open source framework that provides tools to interact with the Semantic Web and Linked Data applications. ASTRA uses modules to represent internal libraries and agents access the library via the module API [9, 8].

The Knowledge Store is the module used by the Building Manager, Room and Sim to interact with the building via REST and add the information returned (in the form of turtle) to the Jena model contained within it. Each agent has an individual Knowledge Store. See Figure 2. In addition, an RDF Schema module allows agents to interact with RDF Schema. Given Room agents must cater for both static and dynamic knowledge, they are given two Knowledge Stores: one for the static knowledge and one for the dynamic knowledge. In this way, the Room agents only need to update the dynamic knowledge on each iteration.

The Jena integration itself takes advantage of two ASTRA features: custom events and custom formulae. The custom event model allows agents to be notified when a url has successfully been read. The `TripleFormula` class is created to provide an internal representation of Triples. ASTRA includes a mechanism for expressing custom formulae in the agent code which is implemented using the RDFSchema module. For example, in ASTRA, to query room triples, we use `rdf.type(string url, brick.qualifiedName("Room"))`. A `NodeFactory` class is then created to extend the basic reasoner to support the new triple formulae. The actual support is implemented in a `TripleNode` class. Finally, the Knowledge Store modules are registered with the agent using the `Queryable` interface. Implementors of this interface are sources of potential formulae to be matched against a query during the reasoning process. These potential formulae are then processed in the `TripleNode`[1].

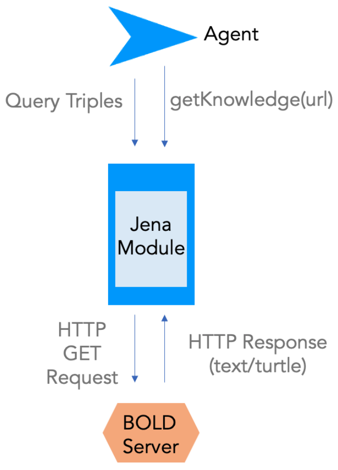

Figure 2: Agent-BOLD Server Interaction

3. Dynamicity and Coordination

The Building Management system supports dynamicity and coordination. The agents in this system are dynamic in the sense that they react to changes in the environment. Based on their updated set of beliefs, the Room agents enact changes on the environment individually by making RESTful calls to the BOLD server in order to turn light switches on and off based on the values of both luminance and occupancy levels.

The Room agents also update their state each time there is a time tick in the environment. Using the dynamic Knowledge Store, each Room agent maintains an up to date description of

[1]Implementations of all these classes can be found in https://gitlab.com/mams-ucd/mams-cartago/-/tree/master/mams-astra-jena and are now part of the MAMS codebase.

its room in accordance with the information provided by the server. To do this, Room agents clear their dynamic Knowledge Store and update their beliefs. This effects the actions they perform on the environment.

This is a fully coordinated system with multiple agents communicating using FIPA-ACL. When created, the Sim agent issues a message to the Main agent. On receipt of this first message, the Main agent creates the Building Agent.

For every subsequent time tick, the Main agent forwards the message to the Building Manager agent, who in turn forwards the message to each Room agent, prompting them to act.

4. Existing work and challenges

The combination of BDI agents with semantic web technology has been broached with JASDL [10]. This uses an annotation system in a way that is similar to the way our approach uses ASTRA modules. However this can be seen as a tighter integration, with a strong coupling with an agent's beliefs. In the system presented in this paper, the use of modules is more akin to adding a skill to the agent, that informs but does not integrate with, nor enforce the consistency of beliefs. It is a more loosely coupled integration.

By allowing this, this system can integrate with multiple ontologies simultaneously. A disadvantage is that there is no enforced consistency, but an advantage is that it paves the way for future work involving the dynamic insertion of modules. This would allow agents to discover which ontologies are required at runtime, making them more flexible, and facilitating reuse across different environments.

In [11], the vision for the coming web is described in it's entirety. It describes how the semantic web will lend a hand in creating a web of resources, each of which could be it's own network in turn, such as an encapsulated neural network. This allows for resources such as computational entities to exist within the network, and for those resources to also make use of other resources within the network. This was one of the core tenets of the MAMS vision, with agents themselves exposing different aspects of themselves as virtual resources on the web, while also utilising the web in order to achieve system and individual agent goals.

It is the hope of the authors that future work will build on ideas in this submission, moving towards agent systems that are situated in a web environment, with a modular use of ontologies, in order to learn relationships that exist across the network. By observation and reasoning about the network, agents can start to autonomously, at runtime utilise semantically defined, previously unknown resources on the network.

Acknowledgments

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

References

[1] A. Ciortea, S. Mayer, F. Gandon, O. Boissier, A. Ricci, A. Zimmermann, A decade in hindsight: the missing bridge between multi-agent systems and the world wide web, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, 2019.

[2] F. Gandon, 4.4 merry hmas and happy new web: A wish for standardizing an ai-friendly web architecture for hypermedia multi-agent systems, Autonomous Agents on the Web 11 (2021) 42.

[3] R. Collier, E. O'Neill, D. Lillis, G. O'Hare, Mams: Multi-agent microservices, in: Companion Proceedings of The 2019 World Wide Web Conference, 2019, pp. 655–662.

[4] E. O'Neill, D. Lillis, G. M. O'Hare, R. W. Collier, Delivering multi-agent microservices using cartago, in: International Workshop on Engineering Multi-Agent Systems, Springer, 2020, pp. 1–20.

[5] E. O'Neill, D. Lillis, G. M. O'Hare, R. W. Collier, Explicit modelling of resources for multi-agent microservices using the cartago framework, in: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, 2020, pp. 1957–1959.

[6] S. Airiau, L. Padgham, S. Sardina, S. Sen, Incorporating learning in bdi agents, in: Workshop AAMAS: adaptive and learning agents and MAS (ALAMAS+ ALAg), ACM Estoril, 2008, pp. 49–56.

[7] R. H. Bordini, A. El Fallah Seghrouchni, K. Hindriks, B. Logan, A. Ricci, Agent programming in the cognitive era, Autonomous Agents and Multi-Agent Systems 34 (2020) 1–31.

[8] R. W. Collier, S. Russell, D. Lillis, Reflecting on agent programming with agentspeak (l), in: International Conference on Principles and Practice of Multi-Agent Systems, Springer, 2015, pp. 351–366.

[9] T. A. S. Foundation, Apache jena, 2021. URL: https://jena.apache.org/.

[10] T. Klapiscak, R. H. Bordini, Jasdl: A practical programming approach combining agent and semantic web technologies, in: International Workshop on Declarative Agent Languages and Technologies, Springer, 2008, pp. 91–110.

[11] F. Gandon, From linked data knowledge graphs to linked intelligence intelligence graphs or the potential of the semantic web to break the walls between semantic networks and computational networks, 2020. URL: https://www.youtube.com/watch?v=b9GPOOu2PTM, iSWC Vision Track.