Long-Lived Agents on the Web: Continuous Acquisition of Behaviors in Hypermedia Environments

Danai Vachtsevanou
University of St. Gallen
St. Gallen, Switzerland
danai.vachtsevanou@unisg.ch

Philip Junker
ETH Zurich
Zurich, Switzerland
junkerp@ethz.ch

Andrei Ciortea
University of St. Gallen
St. Gallen, Switzerland
Inria, Université Côte d'Azur, CNRS
Sophia Antipolis, France
andrei.ciortea@unisg.ch

Iori Mizutani
University of St. Gallen
St. Gallen, Switzerland
iori.mizutani@unisg.ch

Simon Mayer
University of St. Gallen
and ETH Zurich
St. Gallen, Switzerland
simon.mayer@unisg.ch

ABSTRACT

We demonstrate how autonomous goal-directed agents can exploit hypermedia to acquire and execute new behaviors at run time. In addition to behaviors programmed into the agents, in our system agents can discover and reuse behaviors extracted from machine-readable resource manuals. Such manuals can be published by developers, synthesized by agents through automated planning, or even specified by human users at run time. Agents can then discover and use physical and virtual resources in flexible ways, which allows them to better cope with the rapid evolution of *open* and *dynamic* Web environments.

CCS CONCEPTS

• **Human-centered computing → Web-based interaction**; **Hypertext / hypermedia**; • **Computing methodologies → Multi-agent systems**.

KEYWORDS

Hypermedia Multi-agent Systems, Web of Things, Interoperability

ACM Reference Format:
Danai Vachtsevanou, Philip Junker, Andrei Ciortea, Iori Mizutani, and Simon Mayer. 2020. Long-Lived Agents on the Web: Continuous Acquisition of Behaviors in Hypermedia Environments. In *Companion Proceedings of the Web Conference 2020 (WWW '20 Companion), April 20–24, 2020, Taipei, Taiwan.* , 4 pages. https://doi.org/10.1145/3366424.3383537

1 INTRODUCTION

The vision of autonomous agents on the Web is almost as old as the Web itself.[1] In a *semantic* Web, in which real objects and the relationships among them are represented explicitly [2], agents hold the potential to automate and enhance real-world activities. At the same time, the need for autonomous agents on the Web is motivated by the accelerating growth in the number of Web APIs[2] and the evolution towards dynamic, event-driven mashups that integrate both virtual and physical resources (see also [6]).

These developments stress the need for *goal-directed clients*, such as those proposed in [1], that are able to cope with *open* and *dynamic* Web environments: such general-purpose clients would evolve at run time to exploit the dynamic APIs exposed by hypermedia-driven services. Other approaches investigate the use of reactive rule-based programming for specifying and executing dynamic workflows based on a multitude of environment conditions such as the states and properties of resources (e.g. in [10], [14]). To support the openness and interoperability of such environments, the *Web of Things* (WoT) initiative[3] and the W3C WoT Thing Description (TD) [9] help to decouple applications from specific device APIs through semantic descriptions of interaction affordances. These abstract interaction affordances accommodate the openness of hypermedia environments by expressing *actions*, *properties*, and *events* of devices together with the hypermedia controls required for their usage.

Approaches like these foster the development of more open and flexible Web systems by designing clients that are loosely coupled to device APIs and adapt in response to sensory inputs and changing user requirements. This dynamic behavior, however, currently remains limited to *prescripted behaviors* which are defined and implemented during development time. After deployment, clients lack the behaviors needed to address new user requirements or unforeseen changes in the environment without human intervention. To tackle such limitations, other recent approaches draw from research on *Multi-Agent Systems (MAS)* [16][4] to engineer *autonomous agents* that can leverage hypermedia to discover and use resources with

[1] https://www.w3.org/Talks/WWW94Tim/

[2] https://www.programmableweb.com/news/apis-show-faster-growth-rate-2019-previous-years/research/2019/07/17, accessed: 02.01.2020

[3] The WoT is currently being standardized through combined efforts of the W3C (https://www.w3.org/WoT/WG/), IETF (https://datatracker.ietf.org/doc/charter-ietf-core/) and IRTF (https://datatracker.ietf.org/doc/charter-irtf-t2trg/)

[4] MAS is a branch of distributed AI that can be traced back to the mid-70s.

WWW '20 Companion, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7024-0/20/04.
https://doi.org/10.1145/3366424.3383537

abstract semantic interfaces: for instance, in [7] agents use automated planning to synthesize new behaviors as action sequences, which enhances the agents' ability to achieve given goals (i.e., intended system states) using resources available at run time. Yet, the problem of *acquiring new behaviors at run time* remains insufficiently investigated: while clients can already use hypermedia to discover and interact with resources (e.g. using the W3C WoT TD), they are not yet able to access and *acquire higher-level functionalities* in an autonomous manner – to create evolvable, interoperable, Web-based systems.

In this paper, we demonstrate that agents can use hypermedia to acquire new behaviors at run time. In our system, autonomous agents can not only discover and use individual interaction affordances of resources, but also find and make use of *operating instructions* that describe entire new behaviors together with the preconditions and postconditions of adopting such behaviors. This enables them to adjust their capabilities to new goals and in light of the dynamic availability of heterogeneous resources, and thereby promotes agents' *longevity* in *open* and *flexible* hypermedia environments.

2 APPROACH

In this section, we introduce our approach towards enabling agents to continuously acquire new behaviors in hypermedia environments.

2.1 Conceptual Overview

In AI research, an *agent* is commonly defined as "a computer system that is situated in some *environment* and is capable of *autonomous* action in order to meet its design objective" [17]. An agent can perceive its environment via sensors and has an available repertoire of actions that it can execute on resources in its environment. Autonomous agents can then act flexibly towards specific goals based on their own precepts rather than the prior knowledge of the designer [13]. Autonomous agents are furthermore social: they can interact with humans and other agents to achieve tasks that *exceed* their current capabilities. In distributed AI, a MAS is then a system of interacting agents that are situated in a shared environment.

As a new class of MAS, *Hypermedia MAS* use hypermedia as the engine of application state (HATEAOS) [8] in order to integrate MAS into the hypermedia fabric of the Web [5, 6]: agents are situated in a hypermedia-driven environment in which both autonomous entities (e.g. humans, software agents) as well as non-autonomous entities (e.g. physical devices, digital services, knowledge repositories) are represented in terms of Web resources. Agents navigate the hypermedia environment to discover and use resources to achieve their goals - for instance, to cooperate with other agents or to enrich their action repertoire by exploiting non-autonomous entities and their affordances.

In Hypemedia MAS, the agents' environment is designed as a distributed hypermedia application where agents – human and software – access and use devices and other tools modeled as *artifacts* to achieve their goals. Artifacts are designed based on the Artifacts & Agents (A&A) meta-model [12], where artifacts provide agents with access to physical resources (e.g. industrial robots), computational resources (e.g. production scheduling tools), and information resources (e.g. repositories of manuals for exploiting artifacts). Autonomous agents can furthermore use hypermedia search engines [3] to discover resources, while humans can interact with agents to delegate goals and monitor their activity.

Recent work in [5] strongly benefits from the similarity between the A&A meta-model to design artifacts that encapsulate *Web Things*[5], carry *IRIs* and *TDs*, and expose *property*, *event*, and *action* affordances described in their TDs. Agents can instantiate *browser* artifacts to retrieve, parse, and map individual interaction affordances of Web Things to actions in their repertoire at run time [5]. However, these abilities are currently limited to low-level interactions with individual APIs and do not yet extend to more complex behaviors.

2.2 Acquisition of Behaviors for Agents in Hypermedia Environments

A dynamic action repertoire provides the usage interfaces for individual affordances that are then combined to provide complex behaviors for agents[6]. Each such behavior encapsulates a series of interactions for achieving a goal when a certain event is triggered and a set of preconditions holds in the environment. For example, an agent can use a robot resource to deliver a detected item by requesting from the robot to move to the source position, load the item and, finally, move to the target position as long as the item has a maximum allowed weight. Currently, such composite interactive behaviors are generally implemented as plans that are either defined manually or inferred via automated planning at the expense of added complexity and thus reduced responsiveness.

By introducing a proper abstraction layer for *operating instructions* for resources, we enable agents to discover and acquire new behaviors and thereby profit from the openness and dynamics of the hypermedia environment. We consider that agents can use various information sources[7] to enhance their repertoire of plans by exploiting useful operating instructions for using resources towards reaching their design goals. Overall, we consider agents can acquire behaviors from the following sources:

- Behaviors as plans that already exist in the agent's plan library.
- Behaviors as plans shared by other human or software agents.
- Behaviors as plans synthesized via automated planning.
- Behaviors extracted from operating instructions that are translated into plans and are meant to guide agents in exploiting artifacts.

We take a special interest in deriving behaviors for agents through operating instructions of artifacts, as this method could facilitate the flexible and efficient interaction with heterogeneous resources that are discovered dynamically in the environment. Similar to how humans use instructions found in manuals and tutorials to learn about the gainful use of new or complex tools, we design

[5] https://www.w3.org/TR/wot-architecture/#sec-web-thing

[6] We design and program the autonomous agents based on the Belief-Desire-Intention model, i.e. in terms of *beliefs* about the world, *goals* desired to be achieved and *plans* for achieving intended goals [4].

[7] Agents in MAS acquire information through three types of sources: a) internal mental notes, i.e. beliefs about known behaviors that are pre-programmed or generated by the agent itself, b) perceptual information derived from the environment, and c) communication with other agents [4].

software agents that extract meaningful behaviors from operating instructions for artifacts. Such instructions are provided as machine-readable documents that can be discovered, shared, and updated just like any other document on the Web. Such instructions could be provided for instance by device vendors or by individuals who use these artifacts directly or within mashups. Additionally, operating instructions could be reverse engineered from existing plans in case an agent owns a pre-defined or newly synthesized plan and decides to annotate it and share it in the environment for other agents to discover.

Operating instructions allow agents to acquire behaviors that may have been unknown to them so far, or that may be rarely needed and the agent decides avoid storing them permanently in its (constrained) memory. For this, it is important to exploit the full spectrum of information sources that the agent can use to discover and manage operating instructions for artifacts in hypermedia environments. In our system, operating instructions can be acquired by a number of different means:

- Parsing semantic descriptions of artifacts that expose operating instructions in a machine-readable format,
- Look-up in knowledge repositories,
- Discovery of independent resources (e.g. by means of a hypermedia search engine) that expose operating instructions in a machine-readable format and refer to relevant artifacts,
- Communication with other software or human agents.

2.3 Artifact Manuals

MAS researchers have already examined the specification of operating instructions for artifacts in an attempt to design maintainable and extensible goal-directed autonomous consumers of resources (e.g. [11]). Following the methodology in [11], we enable autonomous agents to dynamically read and interpret high-level operating instructions that encapsulate low-level operations of artifacts. Such information is explicitly described in *artifact manuals* and is meant to provide the agent with the operating instructions that are necessary to adequately exploit artifact functionalities.

Artifact manuals can be discovered, managed within manually curated repositories, or consulted by agents that look for behaviors. In this demonstrator, we define artifact manuals in terms of *usage protocols* (see Listing 1 for an example) that are defined in the language introduced in [15] and are characterized by[8]:

- A *function* which defines the goal achieved by the described behavior (e.g. "deliver an item from source position to target position"),
- A set of *preconditions* which defines the necessary conditions for exhibiting such behavior (e.g. conditions on the weight of the item to be delivered),
- A *body* which defines the sequence of actions that result in exhibiting the described behavior,
- A set of *postconditions* which defines the new conditions that will hold after the execution of the actions specified in the *body* (e.g. the new state of the robot resource),
- A *language* that defines the agent programming language of the usage protocol.

[8]The list can be extended to include the *source* of a usage protocol to enable the filtering of information based on its provenance (https://www.w3.org/TR/prov-overview/).

Listing 1 shows a Turtle[9] representation of a manual that specifies a usage protocol for loading and driving items with a robot artifact from a source position to a target position.

Listing 1: A Turtle representation of an artifact manual.

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

<http://localhost:8080/manuals/drivermanual>
  a eve:Manual ;
  eve:hasName "driverManual" ;
  eve:hasLanguage <http://jason.sourceforge.net/wp/> ;
  eve:explains <http://localhost:8080/artifacts/robot2>  ;
  eve:hasUsageProtocol [
    eve:hasName "loadAndDrive" ;
    eve:hasFunction "drive(X1,Y1,X2,Y2)" ;
    eve:hasPreconditions "true" ;
    eve:hasBody " move(X1,Y1);
                  load;
                  move(X2,Y2);
                  unload " ]] .
```

3 SYSTEM DESIGN

In this section, we first provide an overview of our demonstrator scenario, and then present the implemented system and our demonstrator deployment. The latter reflects the demonstrator to be presented at *The Web Conference 2020*. A demonstrator video is available online[10] and the code is available on GitHub[11].

3.1 Demonstrator Scenario

Our demonstrator consists of three virtual robots in a simulated warehouse environment. The environment contains two robot arms at fixed positions and one transporter robot. The robot arms can pick-and-place items and the transporter robot can move freely in the environment and transport items. In the demonstrator scenario, an autonomous agent is responsible for transporting a specific item to a predefined destination location. Depending on the location of the item and the current availability of the robot artifacts, the agent attempts to discover a suitable plan for achieving its goal and reconfigures and adapts the process of delivering the item to the destination. The user can place the item anywhere on the map and activate/deactivate robots and their manuals (Figure 1, left), which will impact the agent's available behaviors and serves to demonstrate the flexibility of our system through the agent's continuous acquisition of behaviors. If the autonomous agent is unable to discover or generate a viable plan by itself, the agent can ask a human expert for help. In our demonstrator scenario, this happens when the item is placed out of reach of the robot arms. A human can then provide a plan that the agent can use to exploit the transporter robot in pushing the item towards a reachable position. To demonstrate the approach outside of a simulation environment and with physical robots, we integrated a *PhantomX AX-12 Reactor Robot Arm*[12] (Figure 1, right) that shadows the movements of one of the simulated robot arms.

At The Web Conference, we will minimally set up the simulation environment such that conference attendees can *challenge* our system by placing the item at different locations, and *help* the

[9]https://www.w3.org/TR/turtle/
[10]https://w3id.org/interactions/www2020
[11]https://github.com/danaivach/hyperlab/tree/yggdrasil_connection
[12]https://github.com/Interactions-HSG/leubot

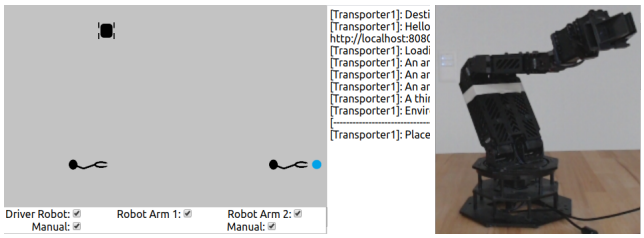

Figure 1: User Interface of the demonstrator and our robot arm.

autonomous agent by specifying capabilities for it to acquire. As part of the demonstration, the attendees will thus experience the continuous acquisition of behaviors by autonomous agents in a hypermedia environment as well as the possibility to escalate to human experts. We will also attempt to bring at least one robot arm to the conference venue that will shadow the virtual robot's movements (or otherwise create a live video stream from the laboratories at our institution).

3.2 System Overview

Our demonstrator consists of several components: The *React*-based user interface[13] (Fig. 2, left) is served by a lightweight *Express.js*[14] back-end which instantiates a *WebSocket* server. We use the *JaCaMo*[15] MAS platform and *Yggdrasil*[16], a platform for hypermedia agent environments, to program and deploy the Hypermedia MAS in our demonstrator. Both the user interface and the MAS connect to the WebSocket server to exchange messages at run-time (i.e., user inputs and agent messages to the GUI).

All resources in our system (agents, workspaces, manuals, artifacts, devices etc.) and relations among them are described in RDF[17]. Browser artifacts for devices and other artifacts are described along with their interaction affordances using the W3C WoT TD. For describing the agent environment, we use the *EVE ontology* [5] which we extended to address artifact manuals (*eve:Manual*) and their relations (*eve:explains*, *eve:hasManual*) to artifacts (*eve:Artifact*).

We further implemented repositories that are dynamically filled with any manuals advertised by artifacts available in workspace at run time. Agents can use manual repositories to find and consult desired usage protocols for the artifacts that they use. Artifact manuals can be continuously added, removed or updated as the artifacts themselves evolve so that agents can acquire new behaviors that reflect the artifacts' current affordances.

We equipped agents with essential plans for managing and acquiring behaviors. In case an agent desires to use an artifact but there is no applicable plan in its plan library, it consults available manual repositories. In case the relevant artifacts do not provide a relevant usage protocol, then the agent uses a hypermedia search engine to search specifically for manuals that explain these artifacts. Finally, the agent results in asking other software or human agents in the environment that may be aware of relevant plans or usage protocols.

4 CONCLUSIONS

The deployed demonstrator shows the various ways that goal-directed agents can use hypermedia to acquire and exhibit new behaviors. Agents implemented in different agent programming languages[18] can discover and use artifact manuals in order to acquire new plans at run time, which allows them to arrive-and-operate in dynamic and open Web environments.

REFERENCES

[1] Mike Amundsen. 2017. *RESTful Web Clients: Enabling Reuse Through Hypermedia.* O'Reilly Media, Inc.
[2] Tim Berners-Lee, James Hendler, Ora Lassila, et al. 2001. The semantic web. *Scientific American* 284, 5 (2001), 28–37.
[3] Simon Bienz, Andrei Ciortea, Simon Mayer, Fabien Gandon, and Olivier Corby. 2019. Escaping the Streetlight Effect: Semantic Hypermedia Search Enhances Autonomous Behavior in the Web of Things. In *Proc. IoT.* ACM.
[4] Rafael H Bordini, Jomi Fred Hübner, and Michael Wooldridge. 2007. *Programming Multi-Agent Systems in AgentSpeak using Jason.* Vol. 8. John Wiley & Sons.
[5] Andrei Ciortea, Olivier Boissier, and Alessandro Ricci. 2018. Engineering World-Wide Multi-Agent Systems with Hypermedia. In *Proc. EMAS.*
[6] 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 *Proc. AAMAS.* 1659–1663.
[7] Andrei Ciortea, Simon Mayer, and Florian Michahelles. 2018. Repurposing Manufacturing Lines on the Fly with Multi-agent Systems for the Web of Things. In *Proc. AAMAS.* 813–822.
[8] Roy T Fielding and Richard N Taylor. 2000. *Architectural Styles and the Design of Network-based Software Architectures.* Vol. 7. University of California, Irvine Doctoral dissertation.
[9] Sebastian Käbisch, Matthias Kovatsch, Takuki Kamiya, Victor Charpenay, and Michael McCool. 2019. *Web of Things (WoT) Thing Description.* Candidate Recommendation. W3C. https://www.w3.org/TR/2019/CR-wot-thing-description-20191106/.
[10] Tobias Käfer and Andreas Harth. 2018. Rule-based Programming of User Agents for Linked Data. In *Proc. LDOW.*
[11] Alessandro Ricci, Michele Piunti, and Mirko Viroli. 2009. Externalisation and Internalization: A New Perspective on Agent Modularisation in Multi-Agent System Programming. In *Proc. LADS.* Springer, Berlin, Heidelberg, 34–54.
[12] Alessandro Ricci, Michele Piunti, and Mirko Viroli. 2011. Environment programming in multi-agent systems: an artifact-based perspective. *Autonomous Agents and Multi-Agent Systems* 23, 2 (2011), 158–192.
[13] Stuart J Russell and Peter Norvig. 2016. *Artificial Intelligence: A Modern Approach.* Malaysia; Pearson Education Limited,.
[14] Steffen Stadtmüller, Sebastian Speiser, Andreas Harth, and Rudi Studer. 2013. Data-Fu: a language and an interpreter for interaction with read/write linked data. In *Proc. WWW.* ACM, 1225–1236.
[15] Mirko Viroli, Alessandro Ricci, and Andrea Omicini. 2006. Operating instructions for intelligent agent coordination. *The Knowledge Engineering Review* 21, 1 (2006), 49–69.
[16] Gerhard Weiss. 1999. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.* MIT press.
[17] Michael Wooldridge and Nicholas R Jennings. 1995. Intelligent agents: Theory and practice. *The Knowledge Engineering Review* 10, 2 (1995), 115–152.

[13] https://reactjs.org/
[14] https://expressjs.com/
[15] http://jacamo.sourceforge.net/
[16] https://github.com/Interactions-HSG/yggdrasil/tree/www2020
[17] https://www.w3.org/RDF/

[18] In our system, usage protocols are defined in Jason - an extention of the AgentSpeak [4] programming language, but they also can be defined in different agent programming languages and platforms such as 2APL (http://apapl.sourceforge.net/) and Jadex (https://www.activecomponents.org/).