SN Computer Science (2022) 3:510
https://doi.org/10.1007/s42979-022-01424-2

ORIGINAL RESEARCH

Towards the Use of Hypermedia MAS and Microservices for Web Scale Agent-Based Simulation

Rem Collier[1] · **Seán Russell**[1] · **Saeedeh Ghanadbashi**[1] · **Fatemeh Golpayegani**[1]

Received: 25 March 2022 / Accepted: 16 September 2022

Abstract
This paper presents a vision for a new breed of Agent-Based Simulations that are based on the principles of the current generation web technologies. We propose a novel approach to implementing complex agent-based simulations built from loosely-coupled reusable components in a manner that ensures scalability. This vision is inspired by the emergence of the recently proposed hypermedia multi-agent systems concept, which combines hypermedia systems, semantic web and affordances.

Keywords Semantic web · Hypermedia systems · Agent-based modelling · Microservices

Introduction

Agent-Based Modelling (ABM) has been successfully applied in many domains, including: logistics [1], intelligent transportation systems [2] and Power Systems [3]. ABM is a bottom-up approach to studying the behaviour of complex systems [4]. These systems are modelled as collections of agents where each agent encapsulates private state and behaviour. The global behaviour of the system then emerges through interactions between the constituent agents.

As the complexity of these systems increases, modelling them accurately presents a challenge. Kitova et al. [5] highlighted that such systems are becoming too complex to be captured in a single model. One solution is to use Hybrid Simulation (HS) [6]. Broadly, HS combines multiple interconnected sub-simulations, potentially implemented using a diverse set of modelling techniques [7]. It is increasingly used in Operational Research [8] and Socio-Environmental Systems [9].

The development of HS tools and methodologies is still an open research problem. In [4], the authors highlight the lack of suitable tools and frameworks for integrating ABM with other techniques. Interoperability is a particular issue leading to many existing HS being built using a single tool [6].

This paper argues that HS should not be built on monolithic architectures and homogeneous technology stacks, but should instead be implemented as loosely-coupled collections of reusable components, written using diverse programming languages and frameworks, that are designed to be deployed at scale. In essence, it argues that HS should be implemented using a microservices architecture [10].

To support this view, this paper presents a vision of a new framework for implementing HS based on ABM, that are constructed as a set of microservices that inter-operate by using REpresentational State Transfer (REST) to serve semantically enriched state representations. The agent layer, which consumes this state, is implemented as a *Hypermedia Multi-Agent System (MAS)* [11], and accordingly, we name our approach *Hypermedia MAS Simulation*.

The remainder of the paper is organised as follows. "The current state of play" reviews the state of the art in three related areas. The first part focuses on simulating at scale, considering both cluster and cloud based approaches to simulation. The remaining two parts focus on how intelligence

This article is part of the topical collection "Web Information Systems and Technologies 2021" guest edited by Joaquim Filipe, Francisco Domínguez Mayo and Massimo Marchiori.

✉ Rem Collier
rem.collier@ucd.ie

Seán Russell
sean.russell@ucd.ie

Saeedeh Ghanadbashi
saeedeh.ghanadbashi@ucdconnect.ie

Fatemeh Golpayegani
fatemeh.golpayegani@ucd.ie

[1] School of Computer Science, University College Dublin, Dublin, Ireland

Published online: 06 October 2022

can be incorporated into the design of the agents and on the use of affordances in simulation. Following on from this "A new vision for simulation at scale" outlines our proposed framework and illustrates it through a traffic simulation scenario. "Building hypermedia MAS simulations" explores a possible approach to implementing the framework, focusing on the relationship between agents and their environments and the role of affordances. Finally, "Challenges and opportunities" outlines challenges and opportunities for further research, and "Concluding remarks" presents some concluding remarks.

The Current State of Play

There are a range of approaches to implementing ABMs, and [12] provides an excellent review of them. What is clear from the review is that ABM has traditionally been viewed as a desktop computer style of exercise where tools are provided to support the creation and execution of models on a single machine. Such tools typically consist of some mechanism to specify agent types and behaviours, the environment(s) that the agent inhabits, any constraints on interaction, and any rules for adaptation of behaviour. The main constraint on the use of such tools to create simulations is often the time taken for the simulation to be executed, which can lead to repeated simplifications of the model until the execution time is acceptable [13].

Simulating at Scale

Within the simulation community, two main approaches have emerged to overcoming the limits of desktop simulation. The Distributed Simulation (DS) community [14] focuses on the development of techniques that can be deployed on high-performance computing clusters. This has led to the development of tools, such as: RePAST HPC [15] and MATSim [16]. Taylor [17] reviews DS through the lens of Operational Research, highlighting the prevalence of bespoke implementations tailored to specific scenarios, the lack of model reuse and the need for well-designed frameworks.

A criticism of DS is the cost and availability of computing clusters. This led to the emergence of Cloud Based Simulation (CBS) [17] which is concerned with the deployment of simulations in the cloud. Example ABM tools include: CloudSME [18], RISE [19] and Distributed Mason [20], a port of Mason [21] for the cloud.

In [22], the authors describe MARS, a cloud-based ABM tool that uses a layered model adapted from Geographical Information Systems, where each layer represents a distinct feature of the simulation. It supports the distribution of models; each layer is deployed as a separate node and individual layers can be distributed across multiple nodes.

More recently, the CBS community has started exploring the intersection of microservices and simulation. Microservices are an architectural style that promote the decomposition of complex systems into distributed applications composed of simple services that are designed to scale [10, 23]. In [13], the authors introduce a microservices framework for the creation of deadline based simulations. Their prototype implemented an auto-scaling mechanism that runs multiple REPAST simulations in parallel to meet deadlines [24]. This is achieved by modelling each simulation as a microservice.

Finally, [25] describes the only known project that has used microservices to create a HS from heterogeneous models. However, as the authors state, the developed model was bespoke and tailored to the specific problem they were addressing.

The adoption of microservices in the design of simulators is appealing for a number of reasons:

- they promote the development of small, loosely-coupled systems that maintain their own independent state [26];
- they are deployed within an ecosystem of tools and components that facilitate rapid and agile development techniques, are easy to extend and support automated management of fault tolerance and scaling [27]; and
- they support the integration of heterogeneous systems built using diverse technology stacks [28].

This is especially true of Hybrid Simulations, where bounded context, isolation and loose coupling [23] promotes the decomposition of simulators into discrete components that can be developed and deployed independently.

The decomposition of simulators into components offers the potential for their reuse across multiple scenarios. Microservices facilitate this through the requirement that all the components adhere to a uniform interface, engendering the use of diverse technology stacks. This presents a route towards the development of hybrid simulations that combine not only different programming languages and tool sets, but also integrate different simulation techniques.

Simulating Intelligence

A recent survey on the use of the Belief-Desire-Intention (BDI) architecture [29] in social simulation [30] highlights the potential quality improvements that cognitive architectures bring to simulation. They outline an emerging trend in social simulation that argues for the application of the "Keep It Descriptive, Stupid" (KIDS) principle over the "Keep It Simple, Stupid" (KISS) principle. Their argument is that, with increased computing resources there is less need to use simplistic agent models in an effort to maximise the performance of the simulation. A better solution is to use

richer cognitive agent architectures to facilitate the creation of more nuanced models.

Agent-Oriented Programming (AOP) languages [31] offer such rich architectures through logic based programming constructs that are both verifiable and explainable [32]. Recently, there has been a resurgence of interest in using AOP for simulation [33–36]. Other work has explored how to support discrete event simulation using AOP. For example, in [37] the authors describe a simulator tool based on GAMA [38] that is applied to Hospital staff planning [39]. In [40] a Socio-Ecological Systems approach to modelling the agricultural economy of the Rancherina River Basin using the BESA [41] agent toolkit is presented.

Finally, [42] presents an emerging simulation platform based on JaCaMo [43] and [44, 45] presents work exploring the creation of *PanSim*, a pandemic simulator based on the 2APL language [46].

Affordances in Simulation

Recent research suggests that affordances may provide a more flexible approach to ABM [47]. The concept of an affordance originates in ecological psychology as a means of representing the relationship between environmental objects and the potential actions that an agent (human or otherwise) may perform with those objects [48].

Affordances are information perceived from the environment that signifies that a particular action may be performed. They present a higher level of abstraction in agent-environment interactions, allowing an agent to reason about the actions it can perform instead of having hard coded actions in plans.

The use of affordances in ABM parallels the changing perception of agent-environment interaction where the environment now viewed as an explicit part of the MAS [49]. This view is being realised through systems such as EIS [50] and CArtAgO [51], which provide an abstraction of the environment that can be used across agent platforms. CArtAgO has already been used in simulations systems, JaCaMo-sim platform [42] is based on its use, and also in combination with affordances in a Web of Things environment [52].

Affordances-effect pairs have been utilised in modelling of human behaviour in complex environments [53]. Affordance fields have been used to represent the suitability of potential actions available to an agent at a given time in path planning simulations [54]. Affordances have also been used in traffic simulation [55] and simulating the behaviour of tanks in a capture the flag exercise [56].

Each of these examples represents a bespoke implementation of the concept of affordances in ABM, each choosing how to represent affordances, how to fit them into the execution of the agent, and how to represent them in the language. The benefits of using affordances in ABM can be greatly enhanced, or the cost of implementation diminished, by the development of a standard representation. This would enable greater interoperability between agents and simulation systems, between agents and environments, and between simulation systems.

A New Vision for Simulation at Scale

We believe there is a need for Hybrid Simulations that can scale while being built from reusable components that are designed for interoperability. The combination of techniques introduced in this paper represent a potential pathway to achieving this. Our approach is inspired by the successes achieved by leading technology companies, such as Netflix and Amazon, who have met the challenge of deploying their infrastructures at Web Scale.

Underpinning their success is their adoption of the microservices architecture [10]. As is highlighted in "Simulating at scale", little research has been carried out into the use of microservices in simulation. While the approach we propose is focused on the application of microservices to ABM, we believe that it can be used to support the creation of simulations that combine diverse modelling approaches.

Microservices and Simulation

In our view, microservices are ideally suited to the implementation of Hybrid Simulations (HS). The mapping of microservices to sub-simulations is in keeping with the ethos of the approach. In fact, as is discussed in "Simulating at scale", adopting a microservices perspective brings a range of additional benefits including mature tool ecosystems and polyglot computing.

The notion of polyglot computing sits well with HS as it is expected that such systems will be composed of multiple sub-simulations implemented using heterogeneous modelling techniques and languages. However, it does not address how to engender interoperability between those sub-simulations. It is our view that this can be achieved through the adaptation of the way that Linked Data is currently used in the Web of Things (WoT) [57]. Broadly speaking, Linked Data is an approach to realising Tim Berners Lee's vision of the Semantic Web [58] through the creation of typed links. These links are embedded within documents that represent data from different Web sources. Critically, Linked Data supports machine readability through the use of ontologies, encoded and interpreted using Semantic Web technologies [59].

Within the WoT, Linked Data has been used to help develop a set of standards, known as *Thing Descriptions* [60]. These are machine readable documents that describe the "things" that have been deployed. They describe the

capabilities of the devices, allowing clients to reason about how (or whether) to use them. The availability of such descriptions is driving research into a range of "thing" composition techniques, including: thing discovery [61], Web Mashups [62], automated composition tools [63] and even agent-based approaches [64]. Such descriptions are also compatible with manual composition tools.

Our vision is inspired by this view. We believe that a *Simulation Description* that is analogous to Thing Descriptions would promote interoperability and reuse of sub-simulations. Such a description could be used both to specify the nature of the environment as well as the inputs and outputs associated with it. The application of semantic meaning to all data inputs and outputs related to each simulation would allow for their integration into data processing pipelines for tasks such as transforming data between representations or handling (time) scaling issues.

Microservices and Agent-Based Simulation

In our view, there are two main ways in which ABM can be integrated with the Hybrid Simulation framework proposed above. One approach is to view them simply as self-contained sub-simulations that are integrated into larger simulations that are not agent-based. While there are a number of challenges in achieving this, it is not the focus of the vision we present in this paper. In our view, such an approach can be realised through existing simulation frameworks.

Instead, our vision is oriented towards a second approach where ABM plays a key role in the overall simulation. Here, the ABM acts as a central component in which agents are linked with multiple sub-simulations, developed using diverse simulation techniques, whose state is consumed by the agents so that they may more accurately model the behaviour of the population they are simulating. Due to the potential complexity arising from processing multiple state data streams, we believe that the simple agent models traditionally used in ABM will be inadequate and more nuanced models are required. In our view, such an approach is congruent with the "Keep It Descriptive, Stupid" principle described in [30]. The authors use this principle to argue that the social simulation community needs to adopt BDI style models, tools and programming languages.

We agree with this sentiment and argue that it is essential for managing the complexity emerging from agents interacting with multiple sub-simulations as is proposed within our vision. In our view, the best way to deliver this is through the use of *Hypermedia MAS*.

In [65], the authors introduce the concept of *Hypermedia MAS* as an approach to building dynamic, open and long-lived MAS [66] that are designed to inter-operate seamlessly with the World Wide Web. In [11], this approach is expanded by outlining a vision in which agents are integrated into the hypermedia fabric of the web, and that by doing so, enter into a shared hypermedia environment that is based on the open standards of the Web. In such an environment, devices and physical services can be exposed as first class entities. Hyperlinks and hypermedia controls can be used to discover and interact with those entities or even other agents regardless of their location. Such controls can be published through hypermedia documents that are adapted based on the state of the underlying entities.

Through these hypermedia documents, agents are able to discover at run-time the capabilities of the entities in their environment. Presented appropriately, such a document could be linked to the concept of affordances as described in "Affordances in simulation" enabling them to understand not just what the state of the entity is, but also what they are currently able to do to that entity. It is important to understand that the concept of Thing Descriptions, as discussed in "Microservices and simulation" is an example of just such a document.

As described earlier, our vision views ABM as the integration point for multiple sub-simulations. This is achieved by viewing the sub-simulations as being part of the environment. The agents would then interact with the sub-systems through an agent-environment interface. In our approach, the environment would be decomposed into a set of microservices and the agents would interact via the APIs exposed by these microservices. There are two possible integration strategies to achieve this: the use of a single microservice that acts as the intermediary between the agents and the subsystems - in the microservices world, this would be considered an API Gateway [67]; and the direct integration of the agents with multiple linked microservices.

We are especially interested in the latter approach as we believe that it offers a more decentralised and scalable solution. The view also has many parallels with the recent work done on applying Hypermedia MAS to the Web of Things [68]. In their approach, the highlight the use of *Thing Descriptions* as a means for describing *interaction affordances* that can be used by the agent to understand its options. We aim to mirror this approach through the introduction of an *Entity Description* that describes the interface between each sub-simulation and the environment. Agents would use this to configure themselves as they connect to a sub-simulation. We term the interface defined by the entity description to be an *Affordance API*. Finally, the entity descriptions should be linked to corresponding *Simulation Descriptions* as defined in "Microservices and simulation".

Mirroring the decomposition of the environment into a set of microservices, we believe that the agents should also be deployed across a set of microservices. A possible solution for achieving this is *Multi-Agent MicroServices (MAMS)* approach [69, 70], which has been demonstrated in the ASTRA agent programming language [71, 72].

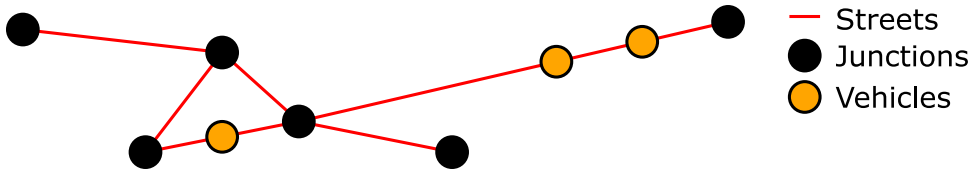

Fig. 1 Representation of road network being simulated

In summary, this section expands our vision for a Hybrid Simulation framework to require the use of Hypermedia MAS to implement the agent component of an ABM. Inspired by recent work in the area of WoT, the agents would interact with sub-simulations, modelled as environment microservices, using REST and Linked Data. As with the WoT, agent-environment interactions are governed through the use of interaction affordances.

Illustrating the Vision

To illustrate our vision, we present a sketch of a traffic simulation based on a small road network which is presented in Fig. 1. The road network is shown as a graph with nodes representing junctions and edges representing streets. Vehicles in transit are shown superimposed over the edges/nodes.

While the example is not really a hybrid simulation, it can be considered to be hybrid because it is composed of multiple sub-simulations that are linked together to form a simulation web. In our example, each street and junction is an individual simulation that is hosted on a specific simulation service. The simulation service hosts multiple simulations that are executed in parallel. Each simulation is customisable based on the overall design of the simulation service. For example, a street simulation service could be customised based on the number of lanes, speed limits and the junctions that the street is connected to. The Junction simulation services are similarly customisable.

A key feature of our approach is the use of Linked Data concepts. Specifically, the individual simulations are uniquely identified by a URL and are linked to one another based on those URLs. As a result, multiple street simulation services can be used in a single traffic simulation. This can offer benefits not only in terms of distribution of computation, but also in terms of the ability to support different kinds of simulation. For example, some street simulations could be implemented using continuous models, while others could be implemented using discrete or even statistical models. Assuming there is some agreement about the basic format of each simulation service, the linked nature of the framework allows for these diverse simulation models to be connected seamlessly.

Another benefit of the approach is that other simulations or components that have no direct link to the agents can also be integrated. These external simulations may be entirely different in nature, either using heterogeneous technologies or alternative forms of intelligence. In effect, these external simulations could be providing input in the form of agents/entities injected in to or removed from the simulation or they may be supplying data that effects the execution of the simulation. For example, a weather simulation could feed information into the sub-simulations within the system to measure the change in traffic patterns. Alternatively, these simulations could represent a source for the network being simulated, introducing agents at a rate that simulates commuters travelling into the city, or they could be a sink, providing a destination for a large number of agents (like a sports stadium) that is also rate limited. These simulations could be monolithic in nature, using diverse technologies, but as long as the output can be processed and feed into the simulation they should be compatible.

As is discussed in "Microservices and agent-based simulation", the agent components of the simulation are modelled as a Hypermedia MAS. This means that the agents view the environment as a collection of hypermedia resources and that all interaction between the agent and the environment is based on REST. One key constraint on the approach advocated here is that this interaction is stateful in that an agent becomes associated with a simulation instance and that association must me maintained as long as is necessary. To achieve this, we propose that an agent body be created in the simulation instance. Further, as the agent migrates across simulation instances (e.g. as it moves from a street to a junction or vice versa), the agent body should transfer from the old simulation service to the new one.

In the transportation example, the agents are embodied within the simulation as the vehicles traversing the network. This connection between the agent and the entity being simulated provides a context for the actions of the agent and further allows the simulation to take advantage of affordances by presenting a list of possible actions to the agent for consideration. This facilitates the programming of the agent at a higher level, or potentially, for the behaviour of the agent to be evolved rather than programmed.

Figure 2 illustrates the simulation web that emerges from the example traffic simulation. Each node on the graph is a hypermedia resource that has an associated URL. Each resource is a unique simulation instance. Querying any of these URLs returns a representation of the current state of the simulation instance. This representation includes the URLs of other connected resources. When underpinned by a Linked Data format, such as JSON-LD or a Semantic Web

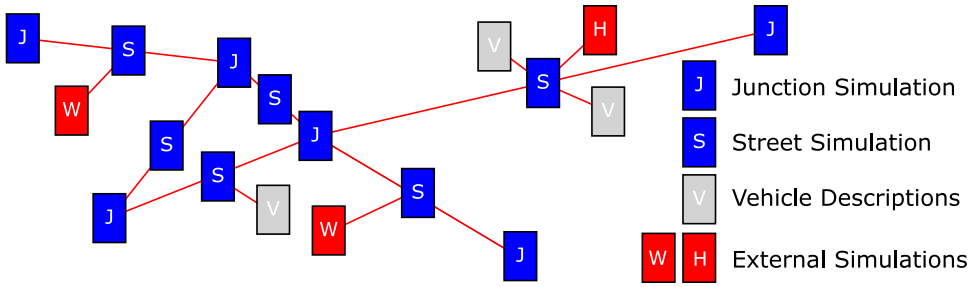

Fig. 2 Representation of hypermedia in MAS simulation

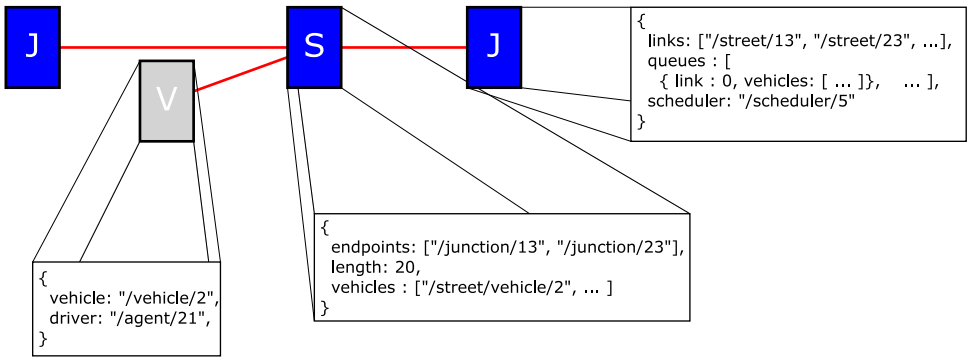

Fig. 3 Internal representation of hypermedia documents in MAS Simulation

format, such as turtle, the resulting simulation web becomes a knowledge graph that can easily be consumed, stored and queried. This has the ability to provide the agents with a wealth of knowledge that can be used to improve their decision making. For example, an agent could explore the knowledge graph of the road network to discover routes between different locations of interest, or could access a job description to understand what tasks must be completed at work.

The use of a Linked Data approach is highlighted in Fig. 3 which presents a simplified view of the contents that these hypermedia documents may contain. The vehicle description gives us a unique endpoint for this vehicle as well as the address of the agent that is coordinating the actions of the vehicle.

Together, these descriptions provide all the information necessary to enable interactions between individual microservices. It gives scope for simulations to communicate and for agents to interact with and within these simulations.

A nice feature of the approach is that, with suitable mechanisms in place, the integration of other simulations can be as simple as connecting two simulation services. In the example, home and work simulations are connected to street simulations allowing, in the correct context, for the agents to exit the street simulation and enter a home or work simulation. We believe that this flexibility is a very powerful feature of the approach as it allows the system to be easily reused or even extended.

This capability to compose simulations from multiple simulation services has implications beyond agent-based modelling simulations. For example, it is possible to enhance our traffic simulation through the integration of a weather simulation service based on numerical weather prediction models. Such a service could be linked to each of the traffic simulation services, providing weather information that can be used to customise the behaviour of the traffic simulation. For example, wet weather could result in stopping distances being increased, while fog could reduce the visibility range of drivers. In this way, the state of the weather simulation would be observed only indirectly though its impact on the connected services (e.g. increased stopping distances or reduced visibility). In our view, the traffic and weather simulators together represent a hybrid simulation as they combine agent-based and numerical simulators. While this combination is quite obvious and natural, we believe that the approach can be generalised and applied to other scenarios.

While not the focus of this paper, it is clear that the approach will also require some form of governing entity (agent or otherwise) to manage the set-up and execution of the simulations. We leave discussion of such issues to future work.

Building Hypermedia MAS Simulations

This section builds on the vision presented in "A new vision for simulation at scale" by exploring how it can be realised in practice. The scope of the vision and the emergent nature of the work makes a comprehensive discussion impossible. Instead, this section focuses primarily on the relationship between the agents and the environment. The objective is to

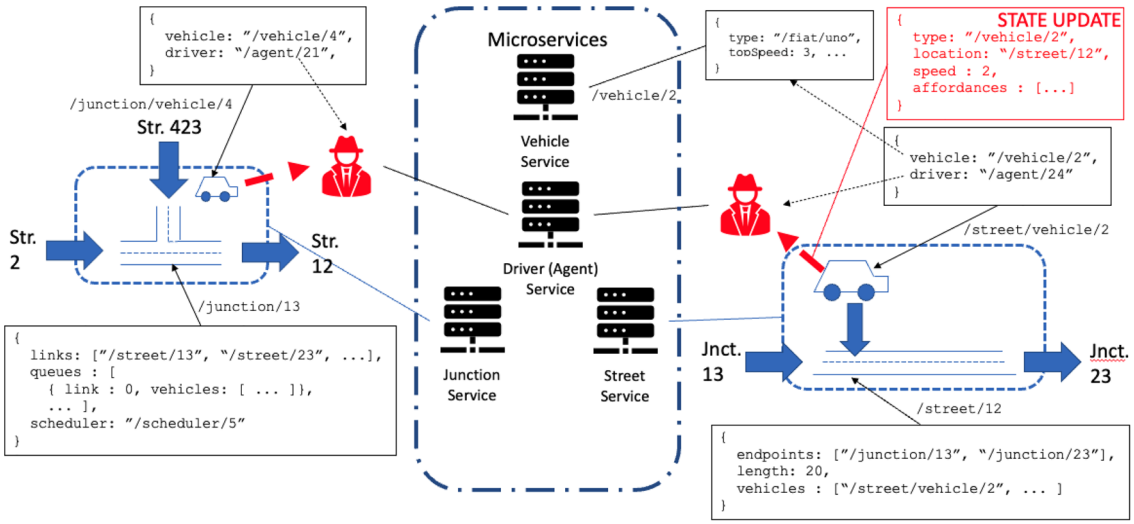

Fig. 4 Microservices view of a traffic simulation

provide some clarity around how Hypermedia MAS Simulations can be implemented.

Central to our vision is the recasting of Agent-Based Models as a collection of interacting microservices. The resultant microservices fall into one of two categories: *agent-oriented microservices*, which host the agent part of the simulation and *environment microservices*, which host pieces of the simulation environment. For the agent-oriented microservices, it is expected that the underlying platform will be largely homogeneous, in the sense that all the agents will be deployed using the same agent platform (although we do not require this) and, critically, the location of each agent has no effect on its behaviour. Conversely, it is expected that the environment microservices will be heterogeneous, in the sense that different microservices will implement different aspects of the simulation environment.

An example of this can be seen in Fig. 4 which presents a simplified view of how microservices might be used to realise the traffic simulation scenario highlighted in "Illustrating the vision". As can be seen, the example simulation consists of two main environment microservices and one agent-oriented microservice. The environment microservices provide simulations of streets and junctions respectively. The agent-oriented microservice provides the agents that drive the simulated vehicles. A fourth microservice also exists. Technically, this is also an environment microservice, but it is more informational in purpose; it provides abstract descriptions of the vehicles used in the simulation that can be used by the agent understand the capabilities of the vehicle they are driving.

The existence of the vehicle service illustrates how we anticipate Linked Data concepts to be applied. At a high level, agents receive periodic state updates from the environment microservice that they are currently connected to. This update includes the URLs of both the body of the agent (see "Agent–environment interactions") and the specific environment simulation that the agent inhabits. The body URL can be used to access general information about the body of the agent (in this case a car), and the environment URL can be used to access general information about the specific environment simulation (in this case, a specific street), which in turn can contain other environment URLs or other relevant information. As was explained in "A new vision for simulation at scale", once integrated with ontologies, this is in effect a Knowledge Graph that spans the entire simulation and which can be used as additional context by the agents.

The example presented offers a fairly homogeneous view of the traffic simulation as there is only one junction microservice and one street microservice. In practice, it is envisaged that there may be multiple variations of each type of environment microservice. For example, street microservices could be created to model different types of road, such as: one way roads, two way roads, or dual-carriageways. A single simulation could then combine these diverse approaches, for example a single lane road could be simulated using continuous models, discrete models, or even statistical models. The value of this is that different parts of the environment can be modelled at different levels of detail depending of the requirements of the simulation.

The remainder of this section focuses on the relationship between the agent-oriented microservices and the environment microservices. "Agent–environment interactions" focuses on how an agent connects to and interacts with an environment microservice. Following this, "The role of affordances" focuses on the role of affordances in supporting

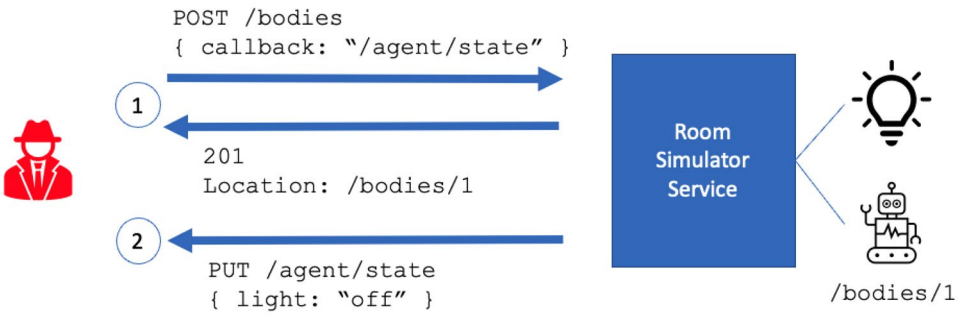

Fig. 5 Illustration of the steps involved in the creation of an agent body

that interaction. Finally, Section Transitioning Between Simulation Components describes how agents transition between environment microservices. The relationship between these techniques and the use of Semantic Web technologies is discussed separately in Folding in semantics.

Agent-Environment Interactions

The relationship between agent-oriented microservices and environment microservices is central to the realisation of Hypermedia MAS Simulations. In traditional ABM simulations, agents are viewed as embodied entities that are situated in some environment. Agent-environment interaction takes the form of a push of relevant environment state from the environment to each agent. The expectation is that the response will contain the next action of the agent which is subsequently executed within the environment.

The approach proposed in this paper mirrors this through the use of a *webhook*[1]. Webhooks are a callback mechanisms for web-based services. They allow services acting as clients in some interaction to receive data asynchronously from an associated server service. This is achieved by exposing a HTTP endpoint on the client side and passing the associated URL to the server. Once received, the server is able to pass data back to the client by submitting HTTP requests via the specified URL. This approach fits well with our vision, as described in Section Microservices and Agent-Based Simulation, because it is based on the assumption that both the agent and the environment microservices are built on the principles of REST.

In order to setup a webhook, the client service must contact the server service by submitting a HTTP request that includes the webhook URL. This URL is then stored in the server service. In some cases, webhook URLs are stored in a simple list with the server service unaware of which client it connects to. However, where relevant, the association between the URL and the client service can also be maintained.

Webhooks are a natural fit for Hypermedia MAS Simulation, where the design of environment microservices can be standardised to include the concept of an agent body. In our approach, each agent has an associated body that is maintained by the environment microservice that the agent is currently interacting with. This body contains, as a minimum, the webhook URL. However, it may also include other properties, such as the URL of the body that has been created for the agent, should it be required. The agent creates a body by registering with an environment microservice. Once registered, the agent begins to receive representations of the relevant simulation state from that microservice via the webhook. For simplicity, the discussion below assumes that the simulation state can be adequately represented by a JSON document. For the purposes of illustrating our approach, we assume that the agent is able to transform this state representation into knowledge that can be used in the decision making process. Later, Section Folding in Semantics explores how the use of Semantic Web technologies can enrich such representations and simplify the transformation of the state into actionable knowledge.

Figure 5 illustrates our approach through a simple example based on a Room Simulation that contains a light and a light switch. As can be seen, the agent creates a body in the simulation by submitting a HTTP POST request to the `/bodies` endpoint. The body of the request contains the webhook URL of the agent (here modelled by the property `callback`). The simulation microservice responds by creating a body for the agent and returning the URL of the newly created body in the HTTP Response. The 201 response code highlights that a new resource has been created, and the URL of the resource is encoded within the `Location:` header field of the response.

Once the body has been created, it is the responsibility of the environment microservice to push any new simulation states to the agent via the webhook URL. Based on the principles of REST, this can be achieved in one of three ways, via a HTTP POST, PUT or PATCH request. In the figure, we illustrate the use of a PUT request which has the effect of replacing the hypermedia entity referenced by the webhook URL through the submission of a new representation of the environment state. This means that the previous state

[1] https://webhook.net/.

representation is replaced by a new state representation. For simplicity, the example uses a JSON representation of the environment state. In practice, we expect this state to use a Linked Data format as is described in Section Microservices and Agent-Based Simulation. This would enable the state to be automatically converted into actionable knowledge that can be used by the agent. However, this requires the adoption or creation of appropriate ontologies, which is an ongoing piece of work. There is some discussion of this later in Section Folding in Semantics.

In cases where it is desirable for the agent to be able to maintain previous state representations, a POST request could be used instead. This has the effect of creating a new resource each time the environment microservice sends a new simulation state. In this second approach, it would be left as the responsibility of the agent to decide when (or whether) to destroy historical simulation states. A third approach is the use of a PATCH request, which may be appropriate for scenarios where the simulation state is large, but where only a few properties change at a time. In this scenario, the environment microservice would only need to send the properties of the simulation state that have changed since the last update. As with PUT requests, the objective is to replace the previous state with the new state, but this is achieved by only transmitting the changes between the previous and current state. Given the three approaches have advantages and disadvantages, our view is that a Hypermedia MAS Simulation should ultimately support all three options, however, in our initial prototypes, only PUT requests have been implemented.

In our view, the selection of a particular method should be done through some form of negotiation based around the requirements of the agent (is a state history required) and the characteristics of the environment (level of dynamism). It is worth noting here that the idea of partial state updates is also compatible with the use of POST requests as REST does not require that the the newly created entity be solely based on the state representation contained in the request body. However, additional context would need to be provided in advance of the use of the webhook to clarify whether the incoming POST requests contain full or partial information.

In addition to providing support for receiving simulation state the environment microservices must also use a standard approach to handling the submission of the agents next action. Given the agents body is hosted remotely in the environment microservices, any actions executed by the agent must be executed in that same microservice. This requires that the agent select the next action to be performed based on the current (and possibly previous) state of environment, and transmit a representation of that action to the environment microservice. This can be achieved in one of four ways: in the HTTP response associated with the webhook request, or via a HTTP POST, PUT or PATCH request to the URL identifying the body of the agent.

The use of the HTTP response associated with the request that is sent to the agents webhook is not a good solution for two reasons: it is not consistent with the principles of REST and synchronous interaction between the environment and the agents has the potential to limit the scalability of the simulation. The first issue relates the requirement that all REST APIs adhere to the HTTP specifications[2] which state that the expected response of a POST, PUT or PATCH should be either a ”201 Created:” response, which should have no body; or a ”200 OK” response whose body should contain the representation of the resource that has just been created or updated. This means that the response must contain the simulation state representation, not an associated action representation. The second issue relates to the realities of network programming each agent microservice maintains a single server socket for all the webhooks. This socket can only maintain a limited number of concurrent connections. When this limit is reached, the socket starts rejecting incoming connection requests (this is the basis on which Denial of Service attacks work). This severely restricts the number of agents that can be deployed in each agent microservice, restricting the ability of the simulation to scale.

The remaining three approaches all satisfy the REST requirement and either create resources or update an existing resource that is associated with the agent body hosted by the environment microservice. The HTTP specification states that POST requests create new resources. This means that the environment microservice would need to create a resource for each action request that it receives. Further, it is expected that a representation of each resource created will be accessible via an associated URL. This would apply to all actions submitted by all agents interacting with a given environment microservice, even if the leave that microservice, for the entire simulation run. Unless some sort of garbage collection strategy is introduced, this would have a significant impact on the resource utilisation of the environment microservices, potentially inhibiting its scalability. Conversely, PUT and PATCH requests do not result in the creation of additional resources, but the full or partial update of an existing resource. PATCH requests should be preferred to PUT requests when the size of the associated representation is large, but where it does not change significantly between requests. In theory, our approach could support both request types, with the environment microservice automatically switching from PUT to PATCH when there is a benefit. However, for simplicity, our current preference is to employ only PUT requests.

[2] https://datatracker.ietf.org/doc/html/rfc2616.html.

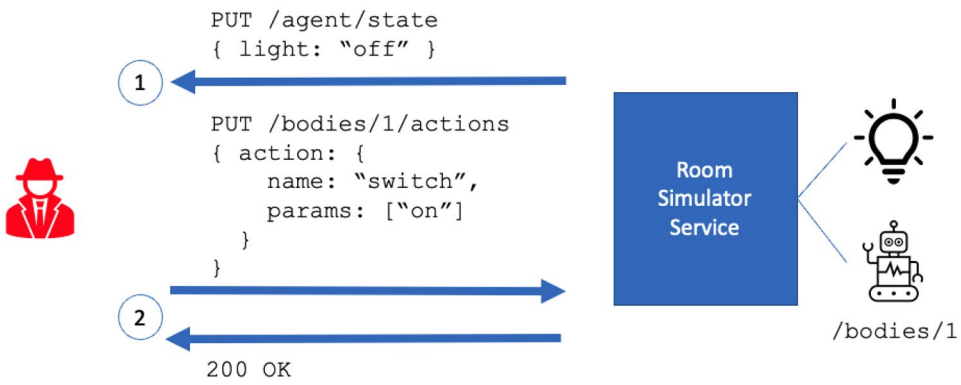

Fig. 6 Illustration of how an agent submits actions to the environment microservice

An example of how an agent submits an action is illustrated in Fig. 6. As with the environment state representations described earlier, for simplicity our illustration uses a JSON representation of the action that is submitted by the agent. This representation takes the form of a unique identifier and a list of parameters which is a widely used representation of actions in the literature.

The Role of Affordances

One of the issues that arises from the approach described in the previous section is the selection of the action to be submitted by the agent. In the example presented in Fig. 6, the agent submits a `switch("on")` action. But the question to be asked is how it made the decision to do this? Answering this question requires three issues to be resolved: (1) identification of the actions that can be performed in the environment that the agent is connected to, (2) filtering out of actions that cannot be performed based on the current environment state, and (3) selecting which of the available actions it should perform.

Traditionally, all of these issues are resolved through the hard coded implementation of specific agent behaviours that define what the agent should do in a range of predefined situations. This works in scenarios where the agent interacts with an environment that is fixed in the sense that all possible environment states and actions are known in advance but is somewhat less useful in scenarios where the agent starts with limited knowledge of the environment. Full prior knowledge of the environment is not always possible or desirable. This is especially the case when the goal is to build simulation components that can be reused across simulations. The alternative solution is to try to reduce the degree of hard coding and to enable some level of adaptation. Perhaps the easiest way to do this is by introducing some form of mechanism to expose the actions associated with the environment to the agent. The value of this is that the agent at least has an awareness of the actions that it could perform even if it ends up it a scenario where it does not know which action to perform next. In such a scenario, the agent could simply select a random action, attempt to learn a new behaviour through trial and error, or simply recognise that the action it intended to perform is invalid.

In our view, there are two ways to expose the list of possible actions to each agent. The first is to create a list of all possible actions for each environment microservice that can be accessed either when the agent connects to the environment or as part of the simulation state. Given such a list would be static (for the environment), and potentially quite long, the single download model would seem to be more efficient. The second approach is to use affordances as described in Section The Role of Affordances. While the literature describes are a number of approaches to implementing the concept of affordances, the basic idea is that the identification of the list of possible actions is a function of the agents perception of its environment rather than being part of the decision making process. As such, affordances are commonly passed as part of the simulation state. The benefit of the latter approach is that it addresses the identification and filtering issues highlighted above. The agent must still select the action it wishes to perform from this list, but it is at least guaranteed that the action it chooses will succeed. If the former approach is used, then the agent is responsible for both filtering and selecting. The filtering process typically involves analysis of the agent's simulation state. Such analysis tends to increase the complexity of processing to be done by the agent without offering any benefit to the simulation.

Selection of what action the agent should perform next is typically a function of the current state of the environment, the goals of the agent and the available actions. Depending on the design of the agent framework, the internal state of the agent may also influence the decision. How the agent makes this decision depends on the simulation requirements. However, offering both the current state and the possible actions, opens the possibility for reinforcement learning techniques to be employed. It also allows for the use of planners or other symbolic reasoning models. Where symbolic approaches are used, there may be a requirement for additional information to be provided. Possible mechanisms for delivering this additional information include the association

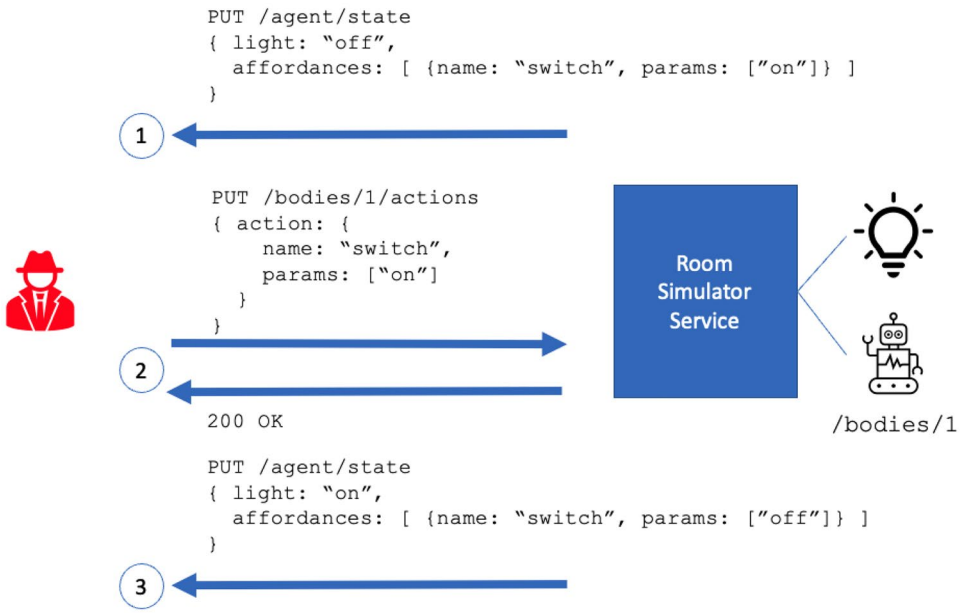

Fig. 7 Introducing affordances into the simulation state

of additional resources, such as manuals, with the environment simulations, potentially encoded using ontologies to for ease of ingestion by the agent.

Figure 7 illustrates our approach through a revision to Figure 6 that includes the affordances within the simulation state. As can be seen, the simulation state now includes the affordances of the environment and the affordances change as the state of the environment changes.

Transitioning Between Simulation Components

The final issue discussed in this paper is how to implement the transition of agents between environment microservices. As was seen in Fig. 4 each simulation in an environment microservice can be linked to other environment microservices through the use of a simulation URL. The idea here is that, given a specific exit state, an agent is able to move its body from one environment microservice to another linked service. For example, in the figure `/agent/24` is driving a vehicle along street `/street/12`. When it reaches the end of this street, it must transition from the street to junction `/junction/23`. A similar scenario arises as the vehicle departs the junction and enters another street. In these scenarios, the transition of the agent body is mandatory. From a technical perspective, the objective is to migrate the agent body from the current microservice to the new microservice. The natural solution to this is to simply copy the agent body to the new microservice and on confirmation destroy the body on the old service. The most natural way to do this is to use a POST request.

The only issue with this is that the URL of the agent body changes as the agent moves around the environment. The agent needs to know the URL of its body so that it can submit the next action to be executed. This can be solved in three ways: (1) by including the agent body URL in the simulation state, (2) by adding a special webhook to update the agent body URL, or (3) by using a "301 Moved Permanently" response to redirect the requester to the new body URL when the old body URL is queried.

The first approach is nice in that it ensures that the agent is always aware of its URL, but it places an additional burden on the design of the simulation state. It is also wasting bandwidth by transmitting redundant information because, once the agent knows its body has moved, it can cache the URL until the next time it moves. While the second approach gets around the wastage of bandwidth, it does so by requiring an additional HTTP request be performed on transition to the new environment microservice. Again, this is adding unnecessary complexity to the design of the environment services.

The third option seems less appealing at first, but actually turns out to be quite nice. This option is known as a server-side redirect. The idea is that the server tells the client when the resource it is trying to access has moved. The server also provides the new URL or the resource. Supporting a server-side redirect minimises the requirements on both the environment microservice and the agent. The agent simply continues to use the URL is has until it receives a 301 response. At that point, the agent simply updates the URL and uses it into the future. This third approach is illustrated in Fig. 8 where, in step 1, the agent sends a PUT request to the old agent body and the body responds with a 301 response. In step 2, the agent updates the agent body URL to reflect the address given in the `Location:` header of the 301 response. Finally, in step 3, the agent uses the new URL to submit its next action.

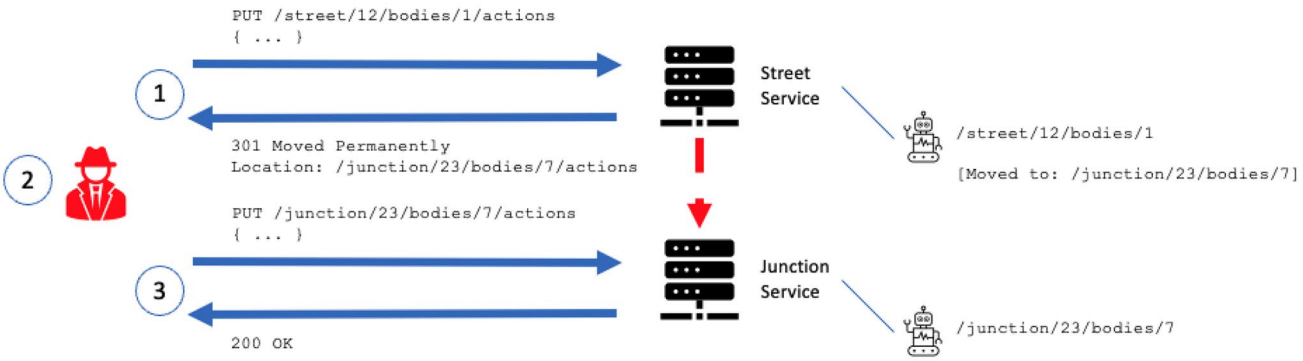

Fig. 8 Updating the agent body URL via a server side redirect

Finally, while the transition in the above example is mandatory, there will also be cases where the transition is optional. For example, an agent that is currently in a home simulation may decide to go to work by car. To achieve this, the agent must choose to leave the home simulator and enter the traffic simulator. Upon reaching work, the agent must leave the traffic simulator and enter the work simulator. Such transitions are driven by the agent rather than the simulator. To support them, it is important to introduce a specific action (e.g. `transition(url)`) that can be executed by the agent to initiate the transfer process. In the context of the discussion around affordances in Section The Role of Affordances, supporting such an action would be trivial. The parameter should be the URL of the destination simulation.

Folding in Semantics

Agent-based simulations are often ad-hoc and based on a subjective understanding of the agent-based concept [73]. Standard agent frameworks are typically not designed to address simulation-specific issues, such as the non-deterministic nature of user behaviour [74]. There are many decisions and actions to be taken by users at any given time, affected by their purpose and context simultaneously, so it is impossible to predict future user behaviour during the design process [74]. While a large amount of knowledge provided by different disciplines/sources (such as psychology, sociology, ergonomics, philosophy, and cognitive science) could be useful hypotheses for an agent-based simulation, able to predict future user behaviour, their lack of integration and formalisation into a reliable framework makes it almost impossible to use them in actual design [74]. To solve this problem, researchers have combined an agent-based simulation with a knowledge base that can formalise and manage data and information [74]. A knowledge base consists of an ontology and individual instances of classes. Although user behaviour is essentially complex and non-deterministic, it can be modelled with a set of behavioural rules, hierarchically structured on multiple levels and linked to each other [74].

Ontologies for Modelling and Simulation

An ontology defines representational primitives to model specific domain knowledge and provides a foundation for reasoning about objects, behaviour, and knowledge as a formal methodology [75]. Ontologies can be classified based on abstraction level and field of application, upper ontology formalising, ontology development concepts, domain ontology including concepts relevant to a particular domain, interface ontology to represent the concepts relevant to the intersection of two disciplines and process ontology describing inputs, outputs, constraints, and sequence of information in a particular process [76]. There are two main classes of ontologies for modelling and simulation: methodological, which describes methods, and referential, which represents real-world systems [77]. The Ontology for Simulation, Modelling, and Optimization (OSMO) is a metadata standard for modelling and simulation workflows in computational engineering [78]. In the literature, various ontology-based templates (i.e., general and domain-specific ontologies) have been defined to represent, formalise, manage, and organise the knowledge about the agent, its behaviour within the environment, and its interaction with other entities [73–75, 79]. In [80], the authors propose a new transformation method to construct a complete and executable agent-based simulation model from an ontology-based model.

Ontologies allow simulation developers to specify environments based on the semantic relationships between the objects of the environment. The representational primitives are classes or sets, properties or attributes, and relationships between concepts or class members of a particular domain [73, 74]. To represent the real-world behaviour in the simulated environments, multi-agent systems' ontology represent agent, resource, action, reaction,

and perception [74, 79], and offer the following advantages. By providing unambiguous and machine-accessible interpretations of terms ontologies act as a middle-ware between the physical layer and the information layer enabling interconnectivity, interoperability and coordination of activities between intelligent heterogeneous systems in an environment. For example, network routing might require a distributed decision making process involving several entities from different systems that can have access to data and behaviour from their local perspectives [81].

Furthermore, ontologies allow a simulation engine to gather and combine information from various sources, to be used as a set of initial hypotheses to tackle the cold start challenge, as well as making assumption about how interactions and behaviour from different systems and domains can be linked to one another [74, 82].

Any change in the environment, assumptions and running scenarios might require change in the models, which is time-consuming and can delay decision-making [80]. By formalising the knowledge base through ontologies, it is possible to perform automated reasoning on the process of modelling and simulation and reuse abstract concepts, and automatically compose agent-based models to form new and scalable simulations for complex domains. Also, it is possible to automate the design and execution of experiments, and the generation of validation tests [73, 74, 83]. Furthermore, Ontology can be used as a benchmark or as a formal framework to understand the compatibility of models expressed in different languages or disciplines, and to evaluate the similarities and differences between different models and objects [75], when more than one model exist for representing a real-world phenomenon.

A unified ontology reduces uncertainty in the decision-making process, ensures compatibility and consistency between different perspectives, and resolves semantic conflicts when operating in a domain with multiple sources of information and behaviour [84, 85], and the implied assumptions and implicit presuppositions [75] can be identified using reasoning techniques [73].

In ontology-driven simulation modelling, the ontology quality strongly impacts the overall quality of the model. The construction of a taxonomy tree and its flexibility is crucial to explaining how agent properties, their values, and rules are inherited. The accuracy and correctness of a conceptualisation can be assessed using factors including effectiveness of the ontology on the achievement of the goals, and its reliability and safety to address the changes in the domain [74, 82]. Also, the simulation tool should keep the agents' ontology up-to-date by sending messages to those agents that should react to newly created facts in the environment [81].

Ontologies for Hypermedia MAS Simulation

According to the ontologies associated with a specific domain, various environment microservices can be developed to model different aspects of the simulation environment and to collect information about them. Ontologies define the conceptual structure at which different aspects of the environment can be modelled at different levels of detail. Concept weighting [86] is a method used in data analysis to determine the importance of concepts based on their weight. Ontologies that relate actions to concepts are called action ontologies [87], which help to determine which concepts are relevant to the action. When dealing with large simulation states, the environment microservice only needs to send to the agents the properties of important concepts identified based on weighting methods or relevant concepts identified based on action ontologies. Ontologies can provide unambiguous interpretations of state representations, thereby reducing the uncertainty in agents' decision-making processes. The ontology-based model proposed in [88] can also be used to augment the representation of the state of an environment and enhance the agents' decision making, particularly when the environment is dynamic and partially observable.

Ontologies for Transportation

In Intelligent Transportation Systems (ITS), the ontology can give a semantic interpretation to information collected by the sensors (located into vehicles, bridges, roads, or traffic signal controllers), facilitating safe driving and improving traffic performance. Adaptive and situation-aware intelligent traffic signal control systems require semantic knowledge about the driver, vehicle, and current driving situation. It is highly beneficial to collaborate in such an environment [89]. Activity-based micro-simulation using agent-based models allows researchers to predict traffic demands by analysing individual trip patterns derived from human behaviours. However, in transportation simulation, data handling can be challenging since several types of data from various domains are generally required as inputs. Furthermore, because activity-based approaches consider non-deterministic and highly context-dependent human behaviour, it is difficult to build agents' behavioural rules. In [85], the authors propose using ontology as a solution to these difficulties. In multi-modal transportation, users use at least two different modes of transportation to reach their destination. It is, therefore, necessary to provide and manage a variety of real-time information on departure times, routes, and traffic conditions before and during travel. Ontologies can be used to integrate all this various information, manage their semantic conflicts, and ensure interoperability, which results in sharing of knowledge and communications between

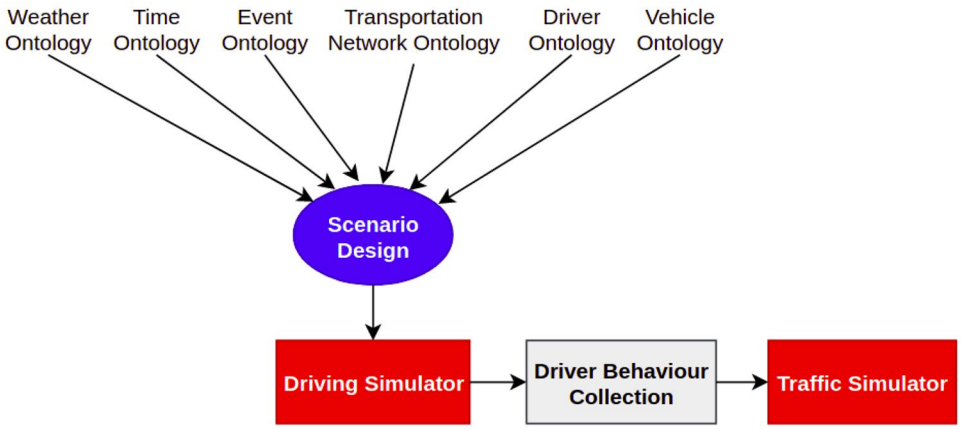

Fig. 9 Use of ontology in the agent-based simulation—an example in the traffic

the different agents involved in the multi-modal transport network [84]. In [90], the authors present a survey of current mobility-related ontologies developed across multiple domains including transportation, smart cities, goods mobility, and sensors. Also, in [91], the authors review the available transportation ontologies and a summary of their scope. The common high-level concepts include location, space, geometry, time, unit of measure, activity (i.e., emergency, entertainment, education, trip, disruptive event), agent (i.e., person, organisation), resource, plan, objective, precondition, effect, transportation network (i.e., road, railway), mode (i.e., vehicle, transit, walking, bicycle, segway), and parking. In [83] the authors introduce the multi-agent-based simulation system which provides reusable and extendable ontological models in different levels including top-level domain ontology (i.e., the ontology for physical objects and abstract services), and transport specific ontologies (e.g., ontologies for multi-modal transportation, logistics, communication, and goods). The environment is perceived by autonomous vehicles using a variety of sensors. To generate the suitable actions that can be implemented by these vehicles, it is helpful to utilise static and mobile concepts defined in the context of three ontologies, including weather, highway, and vehicle, and using the relationships between these concepts [92, 93].

An Example in Traffic Domain

The change in traffic flow condition is often due to the change in drivers' behaviour. Researchers have found that weather conditions, events, and time affect drivers' behaviour significantly [90, 94, 95]. To determine how these parameters affect traffic flow characteristics (i.e., average speed, traffic volume, and road capacity), we can use a combination of a driving simulator and a traffic simulator. The driving simulator can collect drivers' behaviour by simulating the effects of different parameter values (e.g., different weather conditions), while traffic simulation can evaluate changes in traffic flow characteristics by using collected drivers' behaviour (see Fig. 9).

To design scenarios for driving simulation experiments, one can use foundation/basic ontologies that are common across different domains, for example, the ontologies of time, event, and weather. They help in finding the combination of parameters (or concepts) and their different values that make a simulation model match as close to reality as possible. For example, based on the foundation ontologies shown in Fig. 10, fog, rain, snow, and wind should be considered as adverse weather conditions. Additionally, the scenario design should incorporate various planned special events (e.g., sports games or festivals). The relationship between concepts of time of day and visual cues of the driver makes it apparent that time of day effects need to be considered when simulating driving behaviour.

Challenges and Opportunities

Realising the vision presented in this paper raises a number of challenges that also present opportunities for the Multi-Agent Systems community. Some of the key challenges are described below:

- **Tools for building Hypermedia MAS:** Suitable tools for building Hypermedia MAS do not currently exist [11]. Such tools would need to be hypermedia friendly and provide mechanisms to allow agents to reason with and act on semantic knowledge.
- **Defining ontologies for describing simulations:** The effective description of simulation components is critical to enable their composition and reuse. Development of a standardised suite of ontologies to support this is an essential community activity.
- **Exploring the methods and protocols needed to deliver hybrid simulation:** Before effective standards can emerge, there is a need to explore how to combine

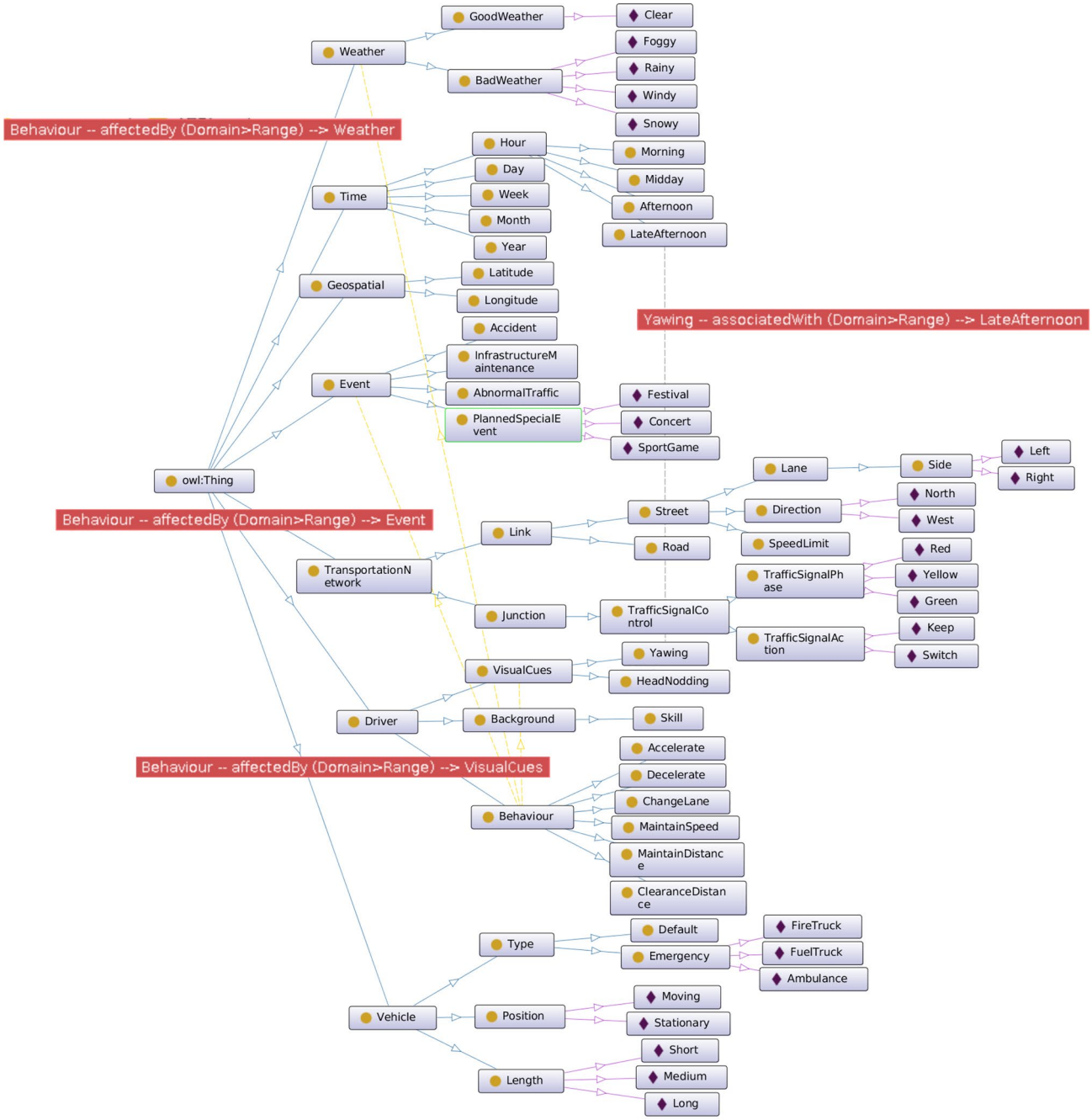

Fig. 10 Ontologies can be used in traffic domain (the OntoGraf plug-in in Protégé is used to create a graph representation of our ontologies)

simulation components: how component connections should be presented to agents and how an agent should migrate from one component to another. It is concerned with the creation of the methods and protocols needed to allow agents to operate seamlessly across them.

- **Building simulations that can scale:** Being able to decompose a simulation into constituent microservice is not enough. There is a need to understand how to do this in a way that enables scalability. This challenge is further complicated by the need to consider multiple sub-simulations and the need for mechanisms that can knit the individual sub-simulations into a larger whole.
- **Reusability across simulation domains:** Understanding how to make sub-simulations reusable across problem domains is another key challenge. This requires the development of standards for defining simulations and techniques for composing them. Some discussion on

potential approaches is presented in section Microservices and Simulation.

Concluding Remarks

This paper presents a vision of a microservices-based approach to implementing Hybrid Simulations that include an ABM component that is underpinned by the recently proposed notion of *Hypermedia MAS*. Section Illustrating the Vision motivates our approach and illustrates the vision from a simple traffic simulation example. Following this, in Section Building Hypermedia MAS Simulations, we propose a number of the basic techniques for realising the core of our vision. Specifically, we focus on the relationship between the agents and the environment. Following this, Section Folding in Semantics further explores the role of ontologies in realising the vision and explores how ontologies could be applied to the traffic simulation scenario. Finally, Section Challenges and Opportunities outlines a number of challenges and opportunities for future research.

While the approach is still largely theoretical, we believe that it is quite novel. There has been little research into the use of microservices for simulation. What they offer is an approach to constructing decentralised simulations that can leverage the infrastructure of the web to function at scale. Through the use of microservices, developers are able to decompose complex systems into small suites of services that can be designed to be reusable, and which are (potentially) implemented using a diverse range of tools, languages and frameworks. A key feature or our approach, is the proposed use of *Simulation Descriptions*, a simulation equivalent of Thing Descriptions that are used to describe sensors in the Web of Things. By providing these descriptions, we believe that it will be possible to simplify the composition and integration of different simulator services, allowing the construction of complex simulations from reusable components. We believe that this will provide a powerful tool to help in addressing the increasingly complex challenges around the modelling of large socio-economic and socio-environmental systems.

Declarations

Conflict of Interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Du, J., Jing, H., Choo, K.-K.R., Sugumaran, V., Castro-Lacouture, D.: An ontology and multi-agent based decision support framework for prefabricated component supply chain. Information Systems Frontiers, 1–19 (2019)
2. Golpayegani, F., et al.: Co-ride: Collaborative preference-based taxi-sharing and taxi-dispatch. In: 2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI), pp. 864–871 (2018). IEEE
3. Teixeira B, Santos G, Pinto T, Vale Z, Corchado JM. Application ontology for multi-agent and web-services' co-simulation in power and energy systems. IEEE Access. 2020;8:81129–41.
4. Polhill JG, Ge J, Hare MP, Matthews KB, Gimona A, Salt D, Yeluripati J. Crossing the chasm: a 'tube-map'for agent-based social simulation of policy scenarios in spatially-distributed systems. GeoInformatica. 2019;23(2):169–99.
5. Kitova OV, Kolmakov IB, Dyakonova LP, Grishina OA, Danko TP, Sekerin VD. Hybrid intelligent system of forecasting of the socio-economic development of the country. Int J Appl Bus Econ Res. 2016;14(9):5755–66.
6. Eldabi, T., Brailsford, S., Djanatliev, A., Kunc, M., Mustafee, N., Osorio, A.F.: Hybrid simulation challenges and opportunities: a life-cycle approach. In: 2018 Winter Simulation Conference (WSC), pp. 1500–1514 (2018). IEEE
7. Mustafee, N., Brailsford, S., Djanatliev, A., Eldabi, T., Kunc, M., Tolk, A.: Purpose and benefits of hybrid simulation: contributing to the convergence of its definition. In: 2017 Winter Simulation Conference (WSC), pp. 1631–1645 (2017). IEEE
8. Brailsford SC, Eldabi T, Kunc M, Mustafee N, Osorio AF: Hybrid simulation modelling in operational research: A state-of-the-art review. European Journal of Operational Research 278(3) (2019)
9. Turner B II, Esler KJ, Bridgewater P, Tewksbury J, Sitas N, Abrahams B, Chapin FS III, Chowdhury RR, Christie P, Diaz S, et al. Socio-environmental systems (ses) research: what have we learned and how can we use this information in future research programs. Current opinion in environmental sustainability. 2016;19:160–8.
10. Fowler M. MicroServices: A definition of this new architectural term (2014). https://martinfowler.com/articles/microservices.html
11. Ciortea A, Mayer S, Gandon F, Boissier O, Ricci A, Zimmermann A. 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)
12. Abar S, Theodoropoulos GK, Lemarinier P, O'Hare GM. Agent based modelling and simulation tools: A review of the state-of-art software. Computer Science Review. 2017;24:13–33.
13. Taylor S, Anagnostou A, Abubakar N, Kiss T, Deslauriers J, Terstyanszky G, Kacsuk P, Kovacs J, Kite S, Pattison G, et al. Innovations in simulation: Experiences with cloud-based simulation experimentation. In: Winter Simulation Conference 2020 (2020)
14. Rashid ZN, Zebari SR, Sharif KH, Jacksi K. Distributed cloud computing and distributed parallel computing: A review. In: 2018 International Conference on Advanced Science and Engineering (ICOASE), pp. 167–172 (2018). IEEE
15. Collier N, North M. Repast hpc: A platform for large-scale agent-based modeling. Large-Scale Computing. 2012;10:81–109.
16. Wshhsh Axhausen K, Horni A, Nagel K. The Multi-agent Transport Simulation MATSim. NY: Ubiquity Press; 2016.
17. Taylor SJ. Distributed simulation: state-of-the-art and potential for operational research. Eur J Oper Res. 2019;273(1):1–19.
18. Taylor SJE, Kiss T, Anagnostou A, Terstyanszky G, Kacsuk P, Costes J, Fantini N. The CloudSME simulation platform and its applications: A generic multi-cloud platform for developing and executing commercial cloud-based simulations. Futur Gener Comput Syst. 2018;88:524–39. https://doi.org/10.1016/j.future.2018.06.006.
19. Al-Zoubi, K., Wainer, G.: Rise: A general simulation interoperability middleware container. Journal of Parallel and Distributed Computing 73(5) (2013)

20. Cordasco G, Scarano V, Spagnuolo C. Distributed mason: A scalable distributed multi-agent simulation environment. Simul Model Pract Theory. 2018;89:15–34.
21. Luke S, Cioffi-Revilla C, Panait L, Sullivan K. Mason: A new multi-agent simulation toolkit. In: Proceedings of the 2004 Swarmfest Workshop, vol. 8, pp. 316–327 (2004). Michigan, USA
22. Hüning C, Adebahr M, Thiel-Clemen T, Dalski J, Lenfers U, Grundmann L. Modeling & simulation as a service with the massive multi-agent system mars. In: Proceedings of the Agent-Directed Simulation Symposium, pp. 1–8 (2016)
23. Zimmermann, O.: Microservices tenets. Computer Science-Research and Development 32(3-4) (2017)
24. Anagnostou, A., Taylor, S.J., Abubakar, N.T., Kiss, T., DesLauriers, J., Gesmier, G., Terstyanszky, G., Kacsuk, P., Kovacs, J.: Towards a deadline-based simulation experimentation framework using micro-services auto-scaling approach. In: 2019 Winter Simulation Conference (WSC), pp. 2749–2758 (2019)
25. Pump, R., Koschel, A., Ahlers, V.: Applying microservices principles to simulation tools. In: Service Computation, 11th International Conference on Advanced Service Computing (2019)
26. Dragoni, N., Lanese, I., Larsen, S.T., Mazzara, M., Mustafin, R., Safina, L.: Microservices: How to make your application scale. In: Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (2018). doi 10.1007/978-3-319-74313-4_8
27. Richards, M.: Microservices Vs. Service-oriented Architecture. O'Reilly Media, (2015)
28. Thönes J. Microservices. IEEE Softw. 2015;32(1):116–116.
29. Rao, A.S., Georgeff, M.P., et al.: Bdi agents: from theory to practice. In: Icmas, vol. 95 (1995)
30. Adam C, Gaudou B. Bdi agents in social simulations: a survey. The Knowledge Engineering Review. 2016;31(3):207–38.
31. Shoham Y. Agent-oriented programming. Artif Intell. 1993;60(1):51–92.
32. Kravari K, Bassiliades N. A survey of agent platforms. J Artif Soc Soc Simul. 2015;18(1):11.
33. Bădică A, Bădică C, Ivanović M, Dănciulescu D. Multi-agent modelling and simulation of graph-based predator-prey dynamic systems: A bdi approach. Expert Syst. 2018;35(5):12263.
34. Lawlor, F., Collier, R., Nallur, V.: Towards a programmable framework for agent game playing. arXiv preprint arXiv:1807.08545 (2018)
35. Bădică, A., Bădică, C., Buligiu, I., Ciora, L.: Devs modeling and simulation using bdi agents: Preliminary considerations. In: Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics, pp. 1–8 (2018)
36. Balabanov K, Cejrowski T, Logofătu D, Bădică C. Study on population dynamics for triple-linked food chain using a simulation-based approach. Evol Syst. 2020;11(2):215–26.
37. Larsen JB. Going beyond bdi for agent-based simulation. Journal of Information and Telecommunication. 2019;3(4):446–64.
38. Taillandier, P., Bourgais, M., Caillou, P., Adam, C., Gaudou, B.: A bdi agent architecture for the gama modeling and simulation platform. In: International Workshop on Multi-Agent Systems and Agent-Based Simulation, pp. 3–23 (2016). Springer
39. Larsen, J.B.: Hospital staff planning with multi-agent goals. PhD thesis, Department of Applied Mathematics and Computer Science, Technical University of Denmark (2019)
40. Muto, T.J., Bolivar, E.B., González, E.: Bdi multi-agent based simulation model for social ecological systems. In: International Conference on Practical Applications of Agents and Multi-Agent Systems, pp. 279–288 (2020). Springer
41. González, E., Avila, J., Bustacara, C.: Besa: Behavior-oriented, event-driven, social-based agent framework. In: PDPTA, vol. 3, pp. 1033–1039 (2003)
42. Ricci, A., Croatti, A., Bordini, R., Hübner, J., Boissier, O.: Exploiting simulation for mas programming and engineering-the jacamo-sim platform. In: 8th International Workshop on Engineering Multi-Agent Systems (EMAS 2020) (2020)
43. Boissier O, Bordini RH, Hübner JF, Ricci A, Santi A. Multi-agent oriented programming with jacamo. Sci Comput Program. 2013;78(6):747–61.
44. Bhattacharya, P., Mooij, A., Dell'Anna, D., Dastani, M., Logan, B., Swarup, S.: Pansim+ sim-2apl: a framework for large-scale distributed simulation with complex agents. In: International Workshop on Engineering Multi-Agent Systems, pp. 1–21 (2021). Springer
45. Mooij, J.d., Dell'Anna, D., Bhattacharya, P., Dastani, M., Logan, B., Swarup, S.: Quantifying the effects of norms on covid-19 cases using an agent-based simulation. In: International Workshop on Multi-Agent Systems and Agent-Based Simulation, pp. 99–112 (2021). Springer
46. Dastani M. 2apl: a practical agent programming language. Auton Agent Multi-Agent Syst. 2008;16(3):214–48.
47. Klügl F. Affordance-Based Interaction Design for Agent-Based Simulation Models. In: Bulling N, editor. Multi-Agent Systems. Cham: Springer; 2015. p. 51–66.
48. Gibson, J.J.: The Ecological Approach to Visual Perception. Houghton-Mifflin, (1979)
49. Weyns, D., Omicini, A., Odell, J.: Environment as a first class abstraction in multiagent systems. Autonomous Agents and Multi-Agent Systems 14(1) (2007). 10.1007/s10458-006-0012-0
50. Behrens, T., Hindriks, K.V., Bordini, R.H., Braubach, L., Dastani, M., Dix, J., Hübner, J., Pokahr, A.: An Interface for Agent-Environment Interaction. In: Collier, R., Dix, J., Novák, P. (eds.) Programming Multi-Agent Systems. Lecture Notes in Computer Science, vol. 6599, pp. 139–158. Springer, (2012). 10.1007/978-3-642-28939-2_8
51. Ricci, A., Viroli, M., Omicini, A.: CArtAgO: A Framework for Prototyping Artifact-Based Environments in MAS. In: Weyns, D., Parunak, H.V., Michel, F. (eds.) Environments for Multi-Agent Systems III. Lecture Notes in Computer Science, vol. 4389, pp. 67–86. Springer, (2007). 10.1007/978-3-540-71103-2_4
52. Ciortea, A., Mayer, S., Michahelles, F.: Repurposing Manufacturing Lines on the Fly with Multi-Agent Systems for the Web of Things. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. AAMAS '18, pp. 813–822. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2018)
53. Jooa J, Kimb N, Wyskc RA, Rothrockd L, Sone Y-J, Ohb Y-G, Leef S. Agent-based simulation of affordance-based human behaviors in emergency evacuation. Simul Model Pract Theory. 2013;32:99–115.
54. Kapadia, M., Singh, S., Hewlett, W., Faloutsos, P.: Egocentric Affordance Fields in Pedestrian Steering. In: Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games. I3D '09, pp. 215–223. Association for Computing Machinery, New York, NY, USA (2009). 10.1145/1507149.1507185.
55. Ksontini F, Mandiau R, Guessoum Z, Espié S. Affordance-based agent model for road traffic simulation. Auton Agent Multi-Agent Syst. 2015;29(5):821–49. https://doi.org/10.1007/s10458-014-9269-x.
56. Papasimeon M. Modelling agent-environment interaction in multi-agent simulations with affordances. Defence Science and Technology Organisation, Air Operations Division: Phd; 2010.
57. Guinard, D.D., Trifa, V.M.: Building the Web of Things vol. 3. Manning Publications Shelter Island, (2016)
58. Berners-Lee T, Hendler J, Lassila O. The semantic web. Sci Am. 2001;284(5):34–43.

59. Bizer, C., Heath, T., Berners-Lee, T.: Linked data: The story so far. In: Semantic Services, Interoperability and Web Applications: Emerging Concepts, pp. 205–227. IGI Global, (2011)
60. Charpenay, V., Käbisch, S.: On modeling the physical world as a collection of things: The w3c thing description ontology. In: European Semantic Web Conference, pp. 599–615 (2020). Springer
61. Zhou Y, De S, Wang W, Moessner K. Search techniques for the web of things: A taxonomy and survey. Sensors. 2016;16(5):600.
62. Guinard, D., Trifa, V.: Towards the web of things: Web mashups for embedded devices. In: Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in Proceedings of WWW (International World Wide Web Conferences), Madrid, Spain, vol. 15, p. 8 (2009)
63. Noura, M., Gaedke, M.: Wotdl: web of things description language for automatic composition. In: 2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI), pp. 413–417 (2019). IEEE
64. Savaglio C, Ganzha M, Paprzycki M, Bădică C, Ivanović M, Fortino G. Agent-based internet of things: State-of-the-art and research challenges. Futur Gener Comput Syst. 2020;102:1038–53.
65. Ciortea, A., Boissier, O., Ricci, A.: Engineering world-wide multi-agent systems with hypermedia. In: International Workshop on Engineering Multi-Agent Systems, pp. 285–301 (2018). Springer
66. Vachtsevanou, D., Junker, P., Ciortea, A., Mizutani, I., Mayer, S.: Long-lived agents on the web: Continuous acquisition of behaviors in hypermedia environments. In: Companion Proceedings of the Web Conference 2020, pp. 185–189 (2020)
67. Montesi, F., Weber, J.: Circuit breakers, discovery, and api gateways in microservices. arXiv preprint arXiv:1609.05830 (2016)
68. Ciortea, A., Mayer, S., Boissier, O., Gandon, F.: Exploiting interaction affordances: On engineering autonomous systems for the web of things. In: Second W3C Workshop on the Web of Things The Open Web to Challenge IoT Fragmentation, Munich, Germany (2019)
69. Collier, R.W., O'Neill, E., Lillis, D., O'Hare, G.: Mams: Multi-agent microservices. In: Companion Proceedings of The 2019 World Wide Web Conference, pp. 655–662 (2019)
70. O'Neill, E., Lillis, D., O'Hare, G.M., Collier, R.W.: 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)
71. Collier, R.W., Russell, S., Lillis, D.: Reflecting on agent programming with agentspeak (l). In: International Conference on Principles and Practice of Multi-Agent Systems, pp. 351–366 (2015). Springer
72. Dhaon, A., Collier, R.W.: Multiple inheritance in agentspeak (l)-style programming languages. In: Proceedings of the 4th International Workshop on Programming Based on Actors Agents & Decentralized Control, pp. 109–120 (2014)
73. Christley, S., Xiang, X., Madey, G.: An ontology for agent-based modeling and simulation. In: Proceedings of the Agent 2004 Conference (2004). Citeseer
74. Simeone D, Fioravanti A. An ontology-based system to support agent-based simulation of building use. Journal of Information Technology in Construction (ITcon). 2012;17(16):258–70.
75. Livet, P., Müller, J.P., Phan, D., Sanders, L., Auatabu, T.: Ontology, a mediator for agent-based modeling in social science (2010)
76. Petrov, V.: Process ontology in the context of applied philosophy. Ontological Landscapes: Recent Thought on Conceptual Interfaces between Science and Philosophy, 137 (2011)
77. Hofmann, M.: Ontologies in modeling and simulation: An epistemological perspective. In: Ontology, Epistemology, and Teleology for Modeling and Simulation, pp. 59–87. Springer, (2013)
78. Horsch, M.T., Toti, D., Chiacchiera, S., Seaton, M.A., Goldbeck, G., Todorov, I.T.: Osmo: Ontology for simulation, modelling, and optimization (2021)
79. Okuyama, F.Y., Vieira, R., Bordini, R.H., da Rocha Costa, A.C.: An ontology for defining environments within multi-agent simulations. In: Workshop on Ontologies and Metamodeling in Software and Data Engineering (2006). Citeseer
80. Kang, D., Bing, Z.C., Song, W., Hu, Z., Chen, S., Zhang, J., Xi, H.: Automatic construction of agent-based simulation using business process diagrams and ontology-based models. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, pp. 1793–1795 (2017)
81. Merdan, M., Vittori, L., Koppensteiner, G., Vrba, P., Favre-Bulle, B.: Simulation of an ontology-based multi-agent transport system. In: 2008 SICE Annual Conference, pp. 3339–3343 (2008). IEEE
82. Gorshkov, S.: Building ontologies for agent-based simulation. In: International Conference in Swarm Intelligence, pp. 185–193 (2015). Springer
83. Warden, T., Porzel, R., Gehrke, J.D., Herzog, O., Langer, H., Malaka, R.: Towards ontology-based multiagent simulations: The PlaSMA approach. In: ECMS, pp. 50–56 (2010)
84. Larioui, J., Byed, E.: Towards a semantic layer design for an advanced intelligent multimodal transportation system. International Journal of Advanced Trends in Computer Science and Engineering **2018** (2020)
85. Cho S, Kang J-Y, Knapen L, Bellemans T, Janssens D, Wets G, Hwang C-S, et al. An activity-based carpooling microsimulation using ontology. Procedia Computer Science. 2013;19:48–55.
86. Belohlavek, R., Macko, J.: Selecting important concepts using weights. In: International Conference on Formal Concept Analysis (ICFCA), pp. 65–80. Springer, (2011)
87. Vitkute-Adzgauskiene, D., Markievicz, I., Krilavicius, T., Tamosiunaite, M.: Learning and execution of Action Categories (ACAT). https://if.vdu.lt/en/research/projects/project-learning-and-execution-of-action-categories-acat
88. Ghanadbashi, S., Golpayegani, F.: An ontology-based intelligent traffic signal control model. (2021). International Intelligent Transportation Systems Conference (ITSC)
89. Feld, M., Müller, C.: The automotive ontology: Managing knowledge inside the vehicle and sharing it between cars. In: Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications, pp. 79–86 (2011)
90. Yazdizadeh, A., Farooq, B.: Smart mobility ontology: Current trends and future directions. Handbook of Smart Cities, 1–36 (2020)
91. Katsumi M, Fox M. Ontologies for transportation research: A survey. Transportation Research Part C: Emerging Technologies. 2018;89:53–82.
92. Chen, W., Kloul, L.: An advanced driver assistance test cases generation methodology based on highway traffic situation description ontologies. In: International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management, pp. 93–113 (2018). Springer
93. Viktorović, M., Yang, D., Vries, B.d.: Connected Traffic Data Ontology (CTDO) for intelligent urban traffic systems focused on connected (Semi) autonomous vehicles. Sensors 20(10), 2961 (2020)
94. Fernando, R.: The impact of Planned Special Events (PSEs) on urban traffic congestion. EAI Endorsed Transactions on Scalable Information Systems 6(23) (2019)

95. Chen C, Zhao X, Liu H, Ren G, Zhang Y, Liu X. Assessing the influence of adverse weather on traffic flow characteristics using a driving simulator and VISSIM. Sustainability. 2019;11(3):830.