Autonomous search in a social and ubiquitous Web

Andrei Ciortea, Simon Mayer, Simon Bienz, Fabien Gandon, Olivier Corby

Personal and Ubiquitous Computing
https://doi.org/10.1007/s00779-020-01415-1

Autonomous search in a social and ubiquitous Web

Andrei Ciortea[1,2] · Simon Mayer[1,3] · Simon Bienz[3] · Fabien Gandon[2] · Olivier Corby[2]

Received: 31 October 2019 / Accepted: 8 May 2020

Abstract

Recent W3C recommendations for the Web of Things (WoT) and the Social Web are turning hypermedia into a homogeneous information fabric that interconnects heterogeneous resources: devices, people, information resources, abstract concepts, etc. The integration of multi-agent systems with such hypermedia environments now provides a means to distribute autonomous behavior in worldwide pervasive systems. A central problem then is to enable autonomous agents to discover *heterogeneous resources* in *worldwide* and *dynamic* hypermedia environments. This is a problem in particular in WoT environments that rely on open standards and evolve rapidly—thus requiring agents to adapt their behavior at run time in pursuit of their design objectives. To this end, we developed a hypermedia search engine for the WoT that allows autonomous agents to perform *approximate search queries* in order to retrieve relevant resources in their environment in *(weak) real time*. The search engine crawls dynamic WoT environments to discover and index device metadata described with the W3C WoT Thing Description, and exposes a SPARQL endpoint that agents can use for approximate search. To demonstrate the feasibility of our approach, we implemented a prototype application for the maintenance of industrial robots in worldwide manufacturing systems. The prototype demonstrates that our semantic hypermedia search engine enhances the flexibility and agility of autonomous agents in a social and ubiquitous Web.

Keywords Autonomous agents · Hypermedia search engines · Web of Things · Semantic Web

1 Introduction

The Web of Things (WoT) fosters innovation and rapid prototyping in the Internet of Things (IoT): developers can use standard Web technologies to create and execute mashups of devices and digital services—so-called physical mashups [20]. WoT systems are often required to evolve rapidly as the availability of devices—and their services—fluctuates. This is particularly true for constrained devices that are often duty-cycled, and for mobile devices (as well as people) that physically move between spatial domains together with the (localized) services they provide. These inherent dynamics can make a WoT system evolve every few seconds or even faster.

In such settings, the manual definition and maintenance of static physical mashups (e.g., via tools such as Node-RED[1] and IFTTT[2]) become impractical: (i) once deployed, they cannot adapt to dynamic environments, and (ii) manually "wiring" the WoT cannot scale well to large numbers of heterogeneous devices (see also [6, 33]). The W3C WoT Thing Description (TD) helps mitigate these limitations through *interaction affordances* and *hypermedia controls* [25]: it allows physical mashups to be defined in terms of abstract interaction patterns rather than specific protocols and device APIs. The resulting physical mashups are then more flexible as they are loosely coupled to the underlying device APIs, but they still have to be defined and maintained manually. Ideally, WoT systems would be able to adapt to large and dynamic settings in an *autonomous* manner—with minimal human intervention.

This article is an extended version of our publication at the *9th International Conference on the Internet of Things (IoT 2019)* [2] and positions our search engine for the Web of Things in the broader context of a *social* and *ubiquitous* Web, which is presented in Section 3.

✉ Andrei Ciortea
andrei.ciortea@unisg.ch

[1] University of St. Gallen, St. Gallen, Switzerland

[2] Inria, Université Côte d'Azur, CNRS, I3S, Sophia Antipolis, France

[3] ETH Zürich, Zürich, Switzerland

[1] http://nodered.org/, accessed: April 15, 2020.

[2] http://www.ifttt.com/, accessed: April 15, 2020.

Such autonomous systems have been studied to a large extent in the scientific literature on distributed artificial intelligence (AI) and, in particular, multi-agent systems (MAS) [50]. In the past, we have shown that MAS research already provides models, programming paradigms, languages, and tooling that can be used to engineer more adaptive WoT systems (e.g., see [9]). However, autonomous agents operating in such systems need to make decisions based on the systems' current state: they need to find and use available resources in real time and with minimal out-of-band information in order to achieve their goals in an autonomous and flexible manner.

We hypothesize that, similarly to how people need hypermedia search engines to find resources on the Web required to achieve their everyday goals (online shopping, travel planning, etc.), autonomous agents will also require hypermedia search engines to help them achieve their goals in the WoT. This analogy is particularly relevant in the context of the W3C WoT [28], which relies on *interaction affordances* as fundamental building blocks. Hypermedia search is still insufficiently investigated in the WoT. A common solution for resource discovery in the WoT is the use of directories, such as the CoRE Resource Directory [42] or the *Thing Directory*.[3] Autonomous agents could then query directories individually or as a federation. However, existing approaches for federated query processing assume that the complete federation is known beforehand [1]—an assumption that fails in an *open and dynamic* WoT. Hypermedia search, on the other hand, facilitates the *flexible* discovery of resources on the Web—an important property for sustaining large-scale, open, and long-lived systems.

We developed a hypermedia search engine for the WoT that allows autonomous agents to perform *approximate search queries* in *(weak) real time* in order to find resources in their environment that are required to achieve their goals. The search engine crawls hypermedia environments and keeps track of their evolution in order to discover device metadata described with the W3C WoT TD. The discovered descriptions are indexed and exposed to clients via a SPARQL endpoint that can process approximate queries. The search engine is based on *Corese* [13], an open-source inference and query engine for Linked Data,[4] together with our own implementation of a hypermedia crawler for dynamic WoT environments. To demonstrate the feasibility of our approach, we implemented a demonstrator based on a concrete scenario for the maintenance of industrial robots in *worldwide* manufacturing systems. The demonstrator shows that our search engine allows agents to cope better with dynamic WoT environments and to pursue their goals in a more flexible and agile manner—therefore enhancing their autonomous behavior in WoT environments.

This paper is structured as follows. We discuss related work on searching the WoT in Section 2. In Section 3, we then present our approach for creating a hypermedia-driven Social Ubiquitous Web and define the search problem in this context. We give an overview of the design and implementation of our system in Section 4. We present our application scenario and the demonstrator deployment in Section 5, and discuss the benefits and limitations of our approach in Section 6.

2 Background and related work

In this section, we first introduce several concepts from MAS research that we use throughout the rest of this paper—with a focus on defining a conceptual bridge between MAS and WoT systems. We then discuss related work on searching the WoT.

2.1 From multi-agent systems to autonomous WoT systems

In AI research, an *agent* is commonly defined as an entity "situated in some environment, that is capable of flexible autonomous action in order to meet its design objectives" [24]. *Autonomy* is central to this definition and refers to the agent's ability to operate on its own, without the need of direct intervention from people or other agents. The agent is situated in an external environment that it can perceive via sensors and influence via actuators. A distinctive feature of an autonomous agent is its flexibility in the pursuit of some design objectives [24]: the agent is *reactive* by responding to changes in the environment in a timely fashion, *proactive* by exhibiting goal-directed behavior and taking the initiative when appropriate, and social by interacting with humans or other agents in order to achieve complex tasks that would surpass its individual capabilities. In distributed AI, a multi-agent system is then a system conceptualized in terms of agents that are situated in a shared environment and interact with one another to achieve their design objectives [24, 50].

Agent-oriented programming was first articulated as a paradigm in [43], but its origins can be traced back to the mid-1980s [19]. A well-known meta-model for designing and programming MAS—that we use in our approach—is *Agents & Artifacts* (A&A) [40]. In A&A, the agents' environment is considered a first-class abstraction in the MAS: a component designed and programmed with clear-cut responsibilities, such as mediating interaction among

[3] https://github.com/thingWeb/thingWeb-directory/, accessed: April 15, 2020.

[4] https://project.inria.fr/corese/

agents or access to the deployment context (devices, digital services, etc.).

The *environment* is modeled as a dynamic set of workspaces, where a workspace is a dynamic set of artifacts. An artifact is a computational object that exposes:

- *Observable properties*: state variables that can be perceived by the agent;
- *Observable events*: non-persistent, fire-and-forget signals that carry information and can be perceived by the agent;
- *Operations*: environment actions provided to the agent; operations can change the values of observable properties or they can trigger events.

The set of all interactions an agent can have with its environment is determined by the artifacts available at run time. Agents use artifacts in pursuit of their goals, and they can create and destroy artifacts at run time.

It is worth to note the similarity between the artifact model defined by A&A and the *Web Thing* model defined by the W3C WoT TD [25]. Both models define three types of interaction affordances, namely observable properties, observable events, and operations or actions, with the Web Thing model being slightly more generic: a W3C WoT TD can expose writable properties, whereas artifact properties are read-only. Applying the W3C WoT TD to decouple artifacts from devices is thus straightforward and provides a conceptual bridge for deploying autonomous agents in WoT environments [9]. A central problem then is to provide agents with the search facility that would allow them to find relevant resources in an efficient manner [8].

The Web has already raised a lot of interest in MAS research, in particular in the context of service-oriented computing [45]. Although related, this work is outside of the scope of this paper, which focuses on the more specific problem of autonomous search in a social and ubiquitous Web. A concise overview of the last decade of research on Web-based MAS is available in [8].

2.2 Searching the Web of Things

There is already a considerable body of research on searching the IoT/WoT and several surveys are available, such as [41] and the more recent [53]. The latter, in particular, provides an extensive review of search techniques for the WoT.

Keyword-based search techniques for the WoT (e.g., [47, 49, 52]) typically target human users—given that choosing meaningful keywords is then of central importance. These techniques are thus less suitable for machines (or autonomous agents), which would rather benefit from approaches that support approximate search. Other search techniques rely on location-based clustering—often in combination with keyword- or tag-based search—and follow the assumption that in the WoT there is a high degree of locality of interactions among human users and devices (e.g., [27, 32]). In another approach, Dyser [36] focuses on real-time search given dynamic sensor readings in WoT environments and uses statistical models to predict the state of registered resources: these models induce a ranking on known resources that determine which are contacted first by the engine to find out whether their actual current state matches the query.

More recent approaches include directory-based discovery mechanisms, such as the CoRE Resource Directory [42] and the Thing Directory,[5] which can be queried individually or as a federation. Most interestingly, Thing Directories store device metadata described using the W3C WoT TD [25] and can expose SPARQL endpoints. Nevertheless, existing approaches for federated SPARQL query processing assume that the complete federation is known beforehand [1]—an assumption that fails in an open and dynamic WoT.

Hypermedia-based discovery via crawling, on the other hand, has proven practical for coping with an open Web. A crawling-based mechanism for WoT devices was proposed in DiscoWoT [31], which allows WoT devices to be crawled in order to discover their properties and any exposed interfaces. However, DiscoWoT assumes that an entry point for the WoT device to be crawled (e.g., its IRI) is known beforehand. To discover devices and other resources, the SPITFIRE architecture [38] suggested crawling the (semantic) WoT periodically, but the crawling process is not discussed in detail—and given the dynamicity of WoT systems, periodical crawling seems impractical.

Another approach aiming to crawl the WoT at Web-scale was proposed in WOTS2E [26], which uses meta-crawling (i.e., it relies on popular search engines such as Google or Bing) to discover SPARQL endpoints that contain WoT-related datasets and ontologies. WOTS2E focuses on the global discovery of relevant SPARQL endpoints (rather than individual devices), which is complementary to our proposal (see also Section 6).

To the best of our knowledge, a search engine for dynamic WoT environments that would allow machines to perform approximate queries in real time is not yet available.

3 Searching a Social Ubiquitous Web

Hypermedia is emerging as a homogeneous information fabric that interconnects everything: not only documents,

[5] https://github.com/thingWeb/thingWeb-directory/, accessed: April 15, 2020.

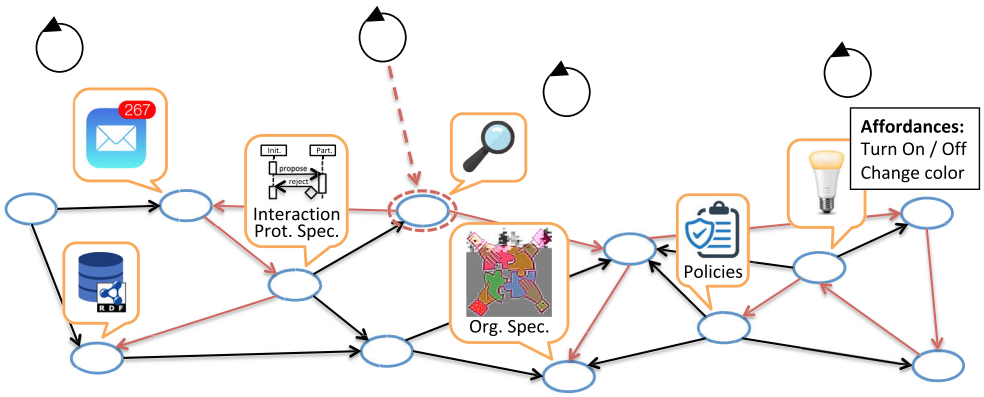


Fig. 1 Hypermedia MAS: Agents are situated in a shared environment designed as a distributed hypermedia application (sourced from [8])

but also people, autonomous agents, devices, data, organizations, etc. [30]. This evolution enables the engineering of a new generation of sociotechnical systems for the Web: systems in which both people and autonomous agents are first-class citizens situated in a worldwide hypermedia environment that spans across the physical–digital space. We refer to such systems as Hypermedia Multi-Agent Systems [7, 8].

In this section, we motivate and position our contribution in the broader research context of a social and ubiquitous Web. We first present Hypermedia MAS in Section 3.1. In Section 3.2, we then present a specific type of Hypermedia MAS, namely socio-technical networks [10], which define a least common denominator for the Web of Things and the Web of People—and thus enable the seamless integration of these two facets of the Web. This seamless integration defines the search space for our autonomous agents: a hypermedia-driven Social Ubiquitous Web. Within this context, in Section 3.3, we then motivate and define in more detail the search problem we tackle.

3.1 Hypermedia multi-agent systems

Hypermedia MAS have been introduced as an emerging class of MAS that are designed for the Web and are aligned with the Web architecture [7, 8]. In a Hypermedia MAS, such as the one depicted in Fig. 1, agents are situated in a distributed hypermedia environment that they can navigate and use in pursuit of their goals—much like we, as human agents, achieve most of our everyday goals (e.g., shopping, travel planning) by browsing and acting on Web resources. The agents' hypermedia environment is weaved into the fabric of the Web, which enables its seamless distribution across the Web. The Web thus provides the underpinning that interconnects all entities within and across Hypermedia MAS.

3.1.1 The hypermedia environment as a first-class abstraction

In Hypermedia MAS, we conceive of the agents' environment as a first-class abstraction[6] [40]: it is an *information layer* that provides agents with various functionalities, such as mediating interaction, communication, and coordination among agents; mediating access to the external environment (e.g., devices, digital services); providing an abstraction layer for modeling, representing, and programming artifacts that agents can use to achieve their goals.

Unlike typical environments in MAS, however, the environment in a Hypermedia MAS uses hypermedia to drive interaction in the system: agents navigate the hypermedia environment to discover other entities in the MAS as well as the means to interact with those entities. The hypermedia environment thus serves as a conceptual bridge between MAS and the Web architecture [8].

From the agents' viewpoint, hypermedia enables a seamless distribution of MAS on the Web: similar to how people navigate and use Web pages regardless of their location, autonomous agents use hypermedia controls to discover and interact with other entities (other agents, tools, etc.) regardless of their location. Hypermedia controls allow autonomous agents to discover at run time the affordances of entities in their environment, such as the operations exposed by a light bulb (cf. Fig. 1). In the WoT, such affordances can be described in a standard manner using the W3C WoT TD [25].

Hypermedia can also be used to advertise in the environment various other resources that agents can

[6] This design choice draws from a line of research on engineering *agent environments* [51].

discover and consume at run time, such as specifications of interaction protocols (e.g., using BSPL [44]), specifications of organizations (e.g., using MOISE [23]), and data licensing policies [48] (cf. Fig. 1). Such resources can be designed into the environment to further reduce coupling in MAS: for instance, engineers in different parts of the world could develop and deploy autonomous agents and interaction protocols independently from one another. To discover such heterogeneous resources, agents can navigate the hypermedia environment themselves (e.g., see [7, 11]), or they can use hypermedia search engines (cf. Fig. 1)—an approach that has proven successful on the Web [5]. Agents can also *manipulate* the hyperlinks in their hypermedia environment in order to "rewire" the MAS to fit their needs. We return to this discussion in Section 3.3.

3.1.2 Hypermedia MAS on the Web

In REST-style hypermedia systems, such as the Web, a resource is the key abstraction of information [16]. In a Hypermedia MAS, the entire observable state of the MAS (e.g., agents, artifacts, relations among them) is projected into the distributed hypermedia environment in a resource-oriented manner, for instance as an RDF graph [14] (cf. Fig. 1). This makes the Hypermedia MAS crawlable and searchable, and allows agents to interact with the MAS in a uniform manner by consuming and producing hypermedia. To illustrate the latter, one agent can send a message to another by writing an RDF representation of the message in the hypermedia (e.g., using an OWL ontology for describing messages). To receive messages, an agent can observe a resource that represents its mailbox in the hypermedia. To turn on a light bulb, an agent can manipulate the state of a resource that represents the light bulb in the hypermedia. All these resources—and the means to interact with these resources—can be discovered by crawling the hypermedia environment of the MAS.

All resources in a Hypermedia MAS are identified using IRIs [15] such that they can be referenced globally. The uniform identification of resources is essential to allow agents to reference and interact with other entities in the MAS regardless of context. For instance, if an agent or a light bulb in their environment is identified via an IRI, they can be referenced without the need for contextual information such as how to interpret platform-specific identifiers or low-level network information (e.g., IP addresses of hosts)—in non-hypermedia MAS, this is not generally the case.[7] The uniform representation of resources (e.g., in RDF) allows hiding any implementation-specific details behind standardized knowledge models. For instance, the state of a light bulb could be represented in the hypermedia environment using RDF together with a standard ontology. An agent could then interact with the light bulb by interpreting and manipulating its semantic representation either directly or via some intermediary tool (e.g., via an artifact). The relations among entities in a hypermedia MAS (e.g., agents, artifacts, organizations, datasets) are also represented explicitly in the hypermedia environment in a uniform manner—such that they can be crawled, manipulated, and reasoned about.

The uniform identification and representation of resources in Hypermedia MAS, together with the explicit and uniform representation of relations among those resources, allow creating a homogeneous information fabric that (i) interconnects heterogeneous resources in a Social Ubiquitous Web, and (ii) can be reliably crawled and indexed for resource retrieval.

3.2 Bridging the Web of Things and the Web of People

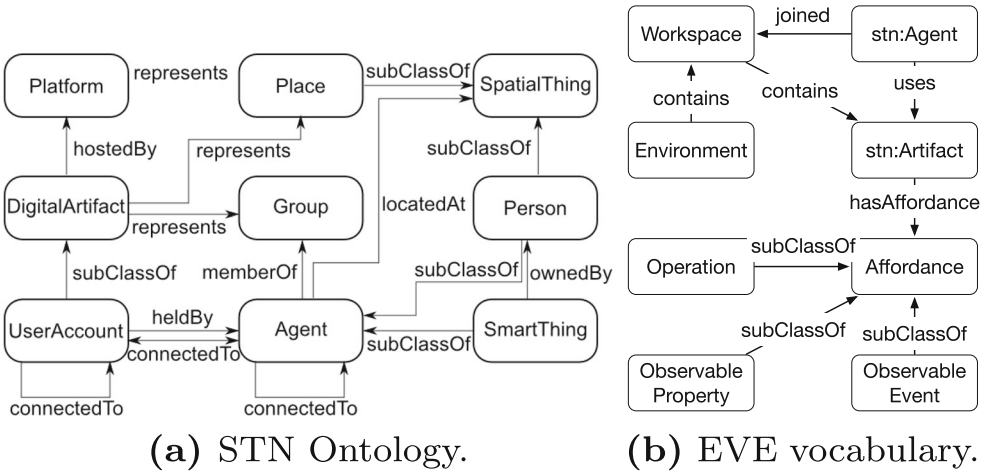
A specific type of Hypermedia MAS is the Socio-Technical Network (STN). The STN model, which was formally defined in [10], provides a least common denominator for the Web of People and the Web of Things: it defines concepts and terms that capture the commonality from existing online social platforms, W3C recommendations and open standards for the Social Web,[8] and research on social aspects in the WoT. The *STN Ontology*, which defines these concepts and terms, is available online.[9] In previous work, it was demonstrated that the STN ontology can be used to create a homogenous hypermedia overlay that spans across silos in the WoT and the Social Web [11]—allowing agents (and any other software clients) to navigate across the otherwise "walled gardens" of the Web.

An overview of the core concepts and properties defined in the *STN Ontology* is depicted in Fig. 2a. In more recent work, the STN ontology was extended with a vocabulary for describing agent environments on the Web (EVE), which is depicted in Fig. 2b. Both the STN ontology and the EVE vocabulary have their roots in the A&A meta-model (see Section 2.1)—and we used both to design the distributed hypermedia environment in our system, which is presented in Section 4.

[7] See [7] for a more detailed discussion.

[8] Such as the activity of the W3C Social Web Working Group (https://www.w3.org/Social/WG) and Interest Group (https://www.w3.org/Social/IG).

[9] https://w3id.org/stn



(a) STN Ontology. (b) EVE vocabulary.

Fig. 2 An overview of the core concepts and properties used to design the distributed environment in our Hypermedia MAS. Term prefixes are omitted for legibility (see Table 1 in Section 4.2). Figure 2b is sourced from [7]

Informally, STNs are dynamic networks of agents and artifacts. All entities in an STN (`stn:Agents`, `stn: Artifacts`, etc.) are interrelated in a meaningful manner via typed relations. Typed relations allow STNs to be navigated in an informed manner. The *STN Ontology* defines `stn:connectedTo` as a generic relation between agents, which can be further extended with domain- and application-specific relation types. On top, the EVE vocabulary adds new concepts and properties that further enhance navigability. For instance, the properties `eve:joined` and `eve:contains` can be used to represent explicitly in the environment hyperlinks to all `stn:Agents` and `stn:Artifacts` in the system—thus making them discoverable via crawling. The affordances of artifacts are also described explicitly in the hypermedia environment using the EVE vocabulary (cf. Fig. 2b)—in conjunction with other vocabularies that can be used to describe implementations of affordances via hypermedia controls, such as the W3C WoT TD [25] and Hydra [29].

By enabling the seamless integration of the Web of Things and the Web of People, STNs define a flexible backbone for a Social Ubiquitous Web—and can be crawled and indexed to retrieve resources that are relevant for people and autonomous agents.

3.3 Searching for heterogeneous resources in a Social Ubiquitous Web

Along its evolution, the Web required increasingly sophisticated machines to sustain its growth. In the early days of the Web, hypermedia by itself was sufficient to allow people to publish documents and to discover documents by following hyperlinks. But as the number of documents grew, new mechanisms were developed to manage this growth: from manually maintained Web directories (e.g., the now defunct Yahoo! Directory, DMOZ), to automatic search engines that use crawlers and information retrieval algorithms to exploit the hypermedia structure of the Web (e.g., via PageRank [37]). As we now move toward a Social Ubiquitous Web, this evolution raises new challenges that are not addressed by traditional hypermedia search engines.

First, unlike the documentary Web, the Social Ubiquitous Web is populated with non-textual resources (e.g., agents, devices, people) that cannot be simply indexed and ranked based on term frequency—this is particularly the case in the WoT (see also [32]). Rather, it would be more appropriate to index semantic descriptions of such resources, which can also capture relevant contextual information (e.g., the current location of a maintenance engineer). Furthermore, the Social Ubiquitous Web is already populated with heterogeneous resources that can be described using a plurality of Web ontologies (one of the main challenges in the Semantic Web community [17]). For instance, to adjust the brightness in a room, an autonomous agent may derive the need to find resources that can increase the room's light level—such as light bulbs or window blinds. The resources can be manufactured by different vendors and described using various ontologies. This motivates the need for methods that support approximate query processing when searching the Social Ubiquitous Web.

Second, the Social Ubiquitous Web is expected to evolve more rapidly than the documentary Web—particularly in the context of the WoT, since the nature of WoT services is more sporadic and transient (e.g., due to physical locality, sleeping nodes, and other contextual factors). This motivates the need for methods that support searching for

resources in real time, a research topic that has already been identified in early work on search engines for the WoT (e.g., [32, 35]).

4 System architecture

To address the challenges identified in the previous section, we developed a hypermedia search engine that allows machines to perform approximate queries for finding relevant resources in their WoT environments in (weak) real time. To achieve this, our approach integrates results from research on the WoT, MAS, and the Semantic Web.

Figure 3 depicts an overview of our system. Following our discussion in the previous section, we design and program the system as a Hypermedia MAS: people and autonomous agents are situated in a distributed hypermedia environment, and we model devices and any other tools that agents use to achieve their goals as artifacts in this environment. All entities in our system (agents, artifacts, workspaces, etc.) are represented as Web resources described in RDF and projected into the distributed hypermedia environment, which enables their system-wide discovery via crawling. We present the main components of our system in what follows.

4.1 Agents & Artifacts container

We use an Agents & Artifacts (A&A) container for programming and running agents and artifacts in W3C WoT environments. The A&A Container is developed using JaCaMo [3], a MAS platform that includes the reference implementation for the A&A meta-model (see Section 2.1). Among others, JaCaMo provides developers with a customizable architecture for cognitive agents, a language for programming cognitive agents, and a Java-based framework for programming artifacts according to A&A.

4.1.1 Agent architecture

We model the autonomous agents in our system as Belief-Desire-Intention (BDI) agents [4]—a type of cognitive agent designed and programmed in terms of mental attitudes: beliefs held about the world, goals desired to be achieved, and plans used to achieve goals. The BDI agent architecture—which is the mainstream architecture for cognitive agents in MAS research—thus provides developers with a formal "human-oriented" level of abstraction that facilitates not only designing and programming, but also inspecting and debugging autonomous behavior. It therefore facilitates the engineering of autonomous WoT systems. Another important feature of the BDI agent architecture that makes it a good fit for our approach is that it can balance goal-directed and reactive behavior: BDI agents commit to goals by executing plans, but they can still react to events and changes in the environment while executing their plans. The JaCaMo platform allows agents to observe artifacts in their environment: changes in an artifact's state and signals emitted by the artifact are reflected in the observing agent's belief base.

The typical program of a BDI agent is composed of the agent's initial sets of beliefs, goals, and plans—all of which can evolve at run time. Multiple languages and frameworks are available for programming BDI agents. One

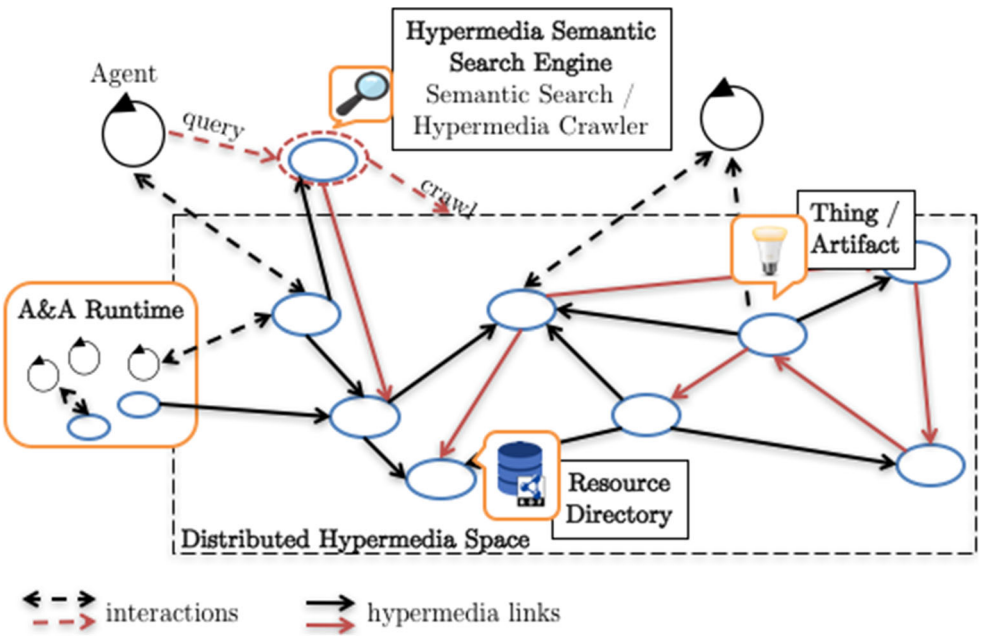


Fig. 3 Conceptual overview of our system

```
1  /* Initial beliefs */
2  green_color(0.409, 0.518).
3  red_color(0.4, 0.2).
4  // (...)
5
6  /* Initial goals */
7  // Initializes the agent, and may lead to
       the
8  //"!notify_engineer" sub-goal being created
9  !start.
10
11 /* Plans */
12 // (...)
13 +!notify_engineer(ArtifactName,CIEx,CIEy) :
       true <-
14   act("http://iotschema.org/SetColour", [
15     ["http://iotschema.org/CIExData", CIEx],
16     ["http://iotschema.org/CIEyData", CIEy]
17   ])[artifact_name(ArtifactName)];
18   .wait(2000);
19   act("http://iotschema.org/TurnOff", [
20   ])[artifact_name(ArtifactName)].
```

Listing 1 Extract from the Jason program of a maintenance agent in our demonstrator. The IoT Schema IRIs in this listing are used only for illustrative purposes

of the most prominent agent programming languages is AgentSpeak(L) [39] and its more recent extended version commonly known as Jason [4]. Jason is the language used in the JaCaMo platform on which our A&A container is based. BDI agents in our system are equipped with libraries of Jason plans, where a Jason plan has the form:

`triggering_event : application_context <- plan_body .`

For illustrative purposes, Listing 1 shows an extract from a Jason program used by a maintenance agent in our demonstrator (see Section 5 for details). The `!start` goal on line 9 in Listing 1 is the entry point into our agent program. Depending on the evolution of the system at run time, the `!start` goal may eventually lead to the sub-goal `!notify_engineer(...)` to be created—that is, if the agent decides it is necessary to notify an on-site engineer of a malfunction (cf. scenario in Section 5.2).

In our demonstrator, malfunctions are signaled visually to on-site engineers via light bulbs. The creation of the `!notify_engineer(...)` sub-goal would then trigger the execution of the plan on lines 13–20 of Listing 1: the agent turns on a light bulb with a given color code for 2 s, and then turns off the light bulb. The initial set of beliefs in our agent program includes, among others, the CIE 1931 XY color codes [46] to be used when notifying on-site engineers (lines 2–3).[10]

4.1.2 Infrastructure artifacts

The A&A container provides agents with several types of infrastructure artifacts that they can use: *browser artifacts*, *crawler artifacts*, and *finder artifacts*.

[10]The color code values used in our demonstrator correspond to nuances of green and red used by Philips Hue.

Browser artifacts serve as facades that allow agents to interact with artifacts discovered at run time in their hypermedia environment as they would interact with any artifact in a local workspace (see Section 2.1) on the JaCaMo platform. Browser artifacts are instantiated with IRIs of W3C WoT TDs, and we refer to them as "browser" artifacts because they perform functions similar to Web browsers: they retrieve and parse W3C WoT TDs, expose interaction affordances to agents, and translate agents' actions to interactions with the *Web Thing* [25] being described (e.g., via HTTP or CoAP). Unlike regular JaCaMo artifacts, browser artifacts expose metadata (e.g., about the supported types of actions provided to agents) via *observable properties*. To perform actions, such as the action of changing the color of a light bulb in Listing 1 (line 14), agents use a generic `act` operation provided by the browser artifact. The `act` operation takes as arguments the IRI of the action type to be executed as well as IRIs of any required parameter types specified in the W3C WoT TD used by the browser artifact. If the W3C WoT TD provides multiple hypermedia controls for the same action type, the first hypermedia control is used.

Agents can use crawler artifacts to configure the search engine, for instance by providing seeds to be crawled or by setting the link types to be followed when crawling (see Section 4.3.1), and they can use finder artifacts to perform search queries (see Section 4.3.2). The role of crawler and finder artifacts is to simplify the agents' logic by encapsulating all logic required to access the HTTP and SPARQL endpoints exposed by our semantic hypermedia search engine.

4.2 Distributed hypermedia environment

All resources in our system (agents, workspaces, artifacts, devices, etc.) and relations among them are described in RDF using the STN ontology and the EVE vocabulary (see Section 3.2). In addition, we use the W3C WoT TD [25] for describing devices and other artifacts together with the interaction affordances they provide. Throughout the rest of this paper, we use the prefix bindings in Table 1.

We map W3C WoT TDs to descriptions of artifacts as defined by the A&A meta-model (see discussion in Section 2.1)—i.e., a Web resource can be both a `td:Thing`

Table 1 Prefix bindings

Prefix	IRI
td:	http://www.w3.org/ns/td#
stn:	http://w3id.org/stn/core#
eve:	http://w3id.org/eve#
ex:	http://example.org/#

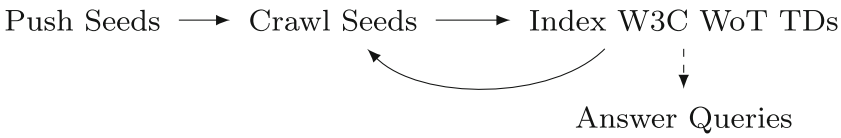


Fig. 4 From seeding to query-answering

and an `eve:Artifact`. This mapping allows autonomous agents to use Web Things as they would use any other artifacts programmed with JaCaMo (via the browser artifacts presented in the previous section).

Both people and autonomous agents can manipulate the hypermedia environment, for instance by adding devices to the system. The hypermedia environment is hosted on *Yggdrasil* [7], our prototypical platform for programming hypermedia environments for autonomous agents, and can be distributed across multiple nodes. The version of Yggdrasil used in our demonstrator is on GitHub[11] and provides two core functionalities:

- It serves as a repository for semantic descriptions of hypermedia environments; each Yggdrasil node exposes a REST HTTP API for creating, updating, and deleting RDF representations of *environment*, *workspace*, and *artifact* abstractions (cf. A&A meta-model in Section 2.1);
- It acts as a hub that (partially) implements the W3C WebSub recommendation [18]; agents—or indeed any software client, such as our A&A container—can use this functionality to observe resources in the environment.

4.3 Semantic hypermedia search engine

Autonomous agents in such distributed hypermedia environments need to be able to conduct searches for a broad range of goals and require structured query and result capabilities to achieve their goals. To this end, we developed a hypermedia search engine for the WoT that autonomous agents can use to perform *approximate search queries* in *(weak) real time*. The search engine consists of two components: (i) our own implementation of a *hypermedia crawler* for dynamic WoT environments, and (ii) *Corese* [13], an open-source inference and query engine for Linked Data.[12] The search engine implements an event-driven non-blocking architecture using *Vert.x*.[13] The hypermedia crawler and the query engine are loosely coupled (and deployed in separate Vert.x verticles), which enhances the evolvability of the system. The semantic hypermedia search engine of our demonstrator is on GitHub.[14]

Figure 4 depicts an overview of the functioning of our search engine: (i) seed IRIs are pushed to the search engine via crawler artifacts; (ii) the seed IRIs are dereferenced and crawled to discover any available W3C WoT TDs; (iii) the discovered W3C WoT TDs are indexed and (iv) queried using SPARQL. We elaborate on these steps in the following.

4.3.1 Hypermedia crawler

We designed and implemented a crawler that navigates distributed hypermedia environments to discover any resources described with the W3C WoT TD. The seeds for initiating the crawling process can be provided by any entity in the system: humans, autonomous agents, resource directories, etc. We enriched the Yggdrasil platform (see Section 4.2) with the functionality to automatically register seeds with the crawler whenever a component is added to a hypermedia environment it hosts (e.g., the IRI of a newly created workspace). This functionality allows the crawler to keep track of the evolution of the distributed hypermedia environment more efficiently as it can rely on Yggdrasil nodes to push notifications whenever parts of the distributed hypermedia environment need to be (re-)crawled.

In addition to seeds, humans and autonomous agents can also configure the crawler with the *link types* to be followed when navigating hypermedia environments, such as links among W3C WoT TDs, links between workspaces and contained artifacts, or any link types defined in the context of domain- and application-specific STNs (see Section 3.2). This functionality allows the crawler (i) to be customized for hypermedia environments described with various ontologies, and (ii) to navigate large-scale hypermedia environments more efficiently.

The crawler exposes two HTTP endpoints: `/links` for pushing IRIs denoting link types that should be followed when crawling hypermedia environments, and `/registrations` for pushing seeds for the crawling process (IRIs of artifacts, workspaces, etc.). Agents can access these endpoints using the crawler artifacts provided by the A&A container (see Section 4.1). After being seeded, the crawler dereferences the registered IRIs to obtain resource representations. From these representations, it extracts links to be followed and continues the crawling recursively in a depth-first manner. We study optimized crawling plans in another work [22] and leave the integration of more sophisticated crawling techniques for

[11] https://github.com/Interactions-HSG/yggdrasil/tree/iot2019/

[12] See https://project.inria.fr/corese/ and also http://corese.inria.fr/

[13] https://vertx.io/, accessed: April 15, 2020.

[14] https://github.com/Interactions-HSG/wot-search/

future development. The crawler stores discovered W3C WoT TDs as RDF data to be indexed and queried.

4.3.2 Semantic query engine

The W3C WoT TDs discovered by our crawler are indexed and queried using Corese[15] [13]. The query engine exposes a SPARQL endpoint `/search` that autonomous agents can use to search for artifacts needed to achieve their goals. Agents access the SPARQL endpoint using the finder artifacts provided by the A&A container (see Section 4.1).

A feature of Corese that is central to our system is the ability to process approximate queries: if there is no exact answer for a query, Corese can approximate the semantics of the query or its structure, or both [12]. To illustrate this feature, let there be an OWL ontology that describes industrial robots in which the `UR5` and `UR10` series of single-armed robots from Universal Robots are sibling subclasses of `SingleArmedRobots`, while the `Baxter` series of two-armed robots from Rethink Robotics is a subclass of `TwoArmedRobots` (where `SingleArmedRobots` and `TwoArmedRobots` are sibling subclasses of `RobotsWithArms`. If an agent searches for a `UR5` robot and none is available, Corese uses the ontological distance between the classes (as they are defined in the class hierarchy) to approximate a `UR10` robot as being semantically closer to a `UR5` than a `Baxter` robot is. We refer the interested reader to [12] for further details on all the query approximation techniques used by Corese—Corese also provides several other features that could be leveraged for searching the WoT, such as federated queries over heterogeneous data sources (see also Section 6).

5 Prototypical deployment

We deployed a demonstrator based on a concrete application scenario in which agents have to cope with open and dynamic WoT environments: the maintenance of industrial robots in worldwide manufacturing systems, in which production sites are distributed across the globe [9]. For our scenario, we consider two types of robots: *manufacturing robots* and *maintenance robots*. Manufacturing robots might require maintenance tasks, and they can delegate such tasks either to maintenance robots or to maintenance engineers. To this end, the manufacturing robots thus have to find in real time what heterogeneous resources (robots or engineers) are available across production sites, and to decide what maintenance tasks can (and should) be fulfilled by a robot and what tasks would require an engineer considering also their locality (e.g., engineers can travel between production sites).

A video of our demonstrator is on YouTube,[16] and the source code is on GitHub.[17] In the following, we first present our deployment setup and then discuss the demonstrator scenario.

5.1 Deployment setup

We deployed our system in a laboratory at the University of St. Gallen. We used two devices in our deployment: a PhantomX AX-12 Reactor Robot Arm controlled via an HTTP API[18] that can be accessed from the Internet, and a Philips Hue light bulb controlled via an HTTP API exposed by a Philips Hue bridge in the local network. We deployed a hypermedia environment distributed across two Yggdrasil nodes running on a MacBook Pro machine in the local network. We deployed the search engine on the same machine together with two A&A containers that host the agents in our demonstrator. Even though in this setup all software components are deployed on the same machine, the components interact with one another via HTTP and could be easily deployed across the Internet.

5.2 Demonstrator scenario

Each of the two Yggdrasil nodes in our deployment hosts the hypermedia environment of a production site—Site A and Site B. The Philips Hue light bulb is deployed on Site A and the PhantomX robot is deployed on Site B. Both the light bulb and the robot are modeled as artifacts that agents can observe and use. In our scenario, agents use the light bulb to signal malfunctions to on-site engineers. We deploy two autonomous agents for each production site: on Site A, we deploy a maintenance agent tasked with monitoring and maintaining industrial robots across all production sites; and on Site B, we deploy a manufacturing agent tasked with controlling the robot arm during normal operation. Each agent runs in one A&A container.

In what follows, we present our demonstrator across three phases (cf. demonstrator video).

5.2.1 Phase 1

In the first phase, the maintenance agent on Site A is launched, starts to *observe the light bulb* (see Section 4.1), and seeds the crawler with the IRI of its local workspace on Site A.

[15] https://github.com/Wimmics/corese

[16] https://youtu.be/iuTzzMA-7FI

[17] https://github.com/Interactions-HSG/wot-search-manufacturing/

[18] https://github.com/Interactions-HSG/leubot

The crawler then crawls the workspace by following all `stn:connectedTo` links (meaning here: a workspace is connected to another work-space) and all `eve:contains` links (meaning here: an artifact is contained in a workspace), which lead the crawler to discover a second workspace on Site B that contains the robot artifact.

The maintenance agent on Site A searches for robots to be monitored—across all existing production sites—by querying the search engine for artifacts that are robotic devices. The ability of Corese to process approximate queries allows the maintenance agent to query for devices of type `ex:RoboticDevice` and to receive as a result the PhantomX robot on Site B, which is of type `ex:Ax12ReactorArm`—in the vocabulary used for this demonstrator, `ex:Ax12ReactorArm` is a subclass of `ex:RoboticDevice`.

Once the robot artifact on Site B is found, the maintenance agent at Site A starts observing it to receive any events it might generate. The manufacturing agent on Site B is launched at the end of Phase 1 and starts operating the robot.

5.2.2 Phase 2

The robot at Site B malfunctions and issues a maintenance event. This event is pushed from the Yggdrasil node on Site B to the A&A container on Site A, which dispatches the event to the maintenance agent due to the subscription created earlier. When the event is received, the maintenance agent queries the search engine to find maintenance suppliers that can perform the required maintenance task.

Since no maintenance robot is deployed at Site B, the search returns an empty result.

The agent then notifies any maintenance engineer that might be available on Site A by switching on the light bulb with a red color (cf. Listing 1). A maintenance engineer travels to Site B, repairs the manufacturing robot, and the robot resumes its tasks.

5.2.3 Phase 3

A maintenance robot is deployed on Site B and registered with the local Yggdrasil node, which automatically updates the workspace of Site B. Yggdrasil pushes the IRI of the newly added robot to the crawler, which discovers the robot.

Similar to Phase 2, the manufacturing robot malfunctions and issues a maintenance event, which is dispatched to the maintenance agent on Site A. Upon receiving the event, the maintenance agent queries again the search engine and receives as a result the newly installed maintenance robot on Site B. The maintenance agent delegates the task to the maintenance robot and informs manufacturing engineers on Site A by switching on a green light (cf. Listing 1).

6 Discussion and limitations

The deployed demonstrator proves the two key elements of our approach. First, the maintenance agent is able to use *approximate search queries* to find industrial robots in a hypermedia environment distributed across two production sites. In our setup, the Yggdrasil nodes for the two production sites run on the same machine, but they could be easily distributed across the Web – as they are in the Yggdrasil demonstrator presented in [7]. Second, our prototypical search engine has the ability to keep track of the evolution of the hypermedia environments it indexes, which allows the maintenance agent to perform searches in *(weak) real time*: when the maintenance robot is added to the environment, Yggdrasil notifies the search engine, which crawls and indexes the robot. In the future, we intend to implement a similar mechanism for tracking components that are removed from the environment. We say searches are performed in *weak real time* because the notification-based mechanism used in our prototype would be insufficient for keeping track of large-scale, rapidly evolving environments. In the future, complementary mechanisms could be added to improve real-time search. For instance, predictive crawling (e.g., see [21]) could be used to crawl and index fast-changing areas in the environment more frequently, or to determine which parts of an environment should be prioritized during crawling.

Our search engine crawls WoT environments to discover and index W3C WoT TDs. In most cases, the TDs would describe devices, but they could also describe resource directories, such as the *Thing Directory*.[19] Ou r current implementation would treat a discovered Thing Directory as any regular Thing in the environment—and thus leaves it to agents to use the Thing Directory if they are able to do so. In the future, we intend to extend our search engine with the ability to automatically query SPARQL endpoints discovered in the environment at run time. Corese already supports federated SPARQL queries, and we study the automatic discovery and querying of SPARQL endpoints in another work [34]. Our current implementation also does not check the correctness of discovered TDs beyond RDF syntax—for instance, to check if a given TD is usable and corresponds to the Web Thing being described, or if a described device is operational. We leave it as future work to investigate such mechanisms.

As a direction for future research, we intend to investigate the ranking of resources based on agents' goals and current context. For instance, an agent having the goal to increase the brightness in a room could do so using either light bulbs or window blinds, but the relevance of

[19] https://github.com/thingWeb/thingWeb-directory/, accessed: September 08, 2019.

these resources is also contextual: opening the window blinds during nighttime would have little impact on the room's light level. Going further, in our current approach, agents rely on libraries of plans programmed by developers in order to "bridge" their goals to relevant resources. Currently, this knowledge has to be programmed into the agents (cf. Listing 1); it can be obtained at run time from other agents (if available), or could potentially be inferred at the expense of added complexity (e.g., via automated planning). Providing agents with a context-aware search engine that can process goal-oriented queries (rather than resource-oriented queries) would further enhance the agents' flexibility in achieving their goals.

7 Conclusions

We hypothesize that similar to how hypermedia search enhances people's ability to achieve their everyday goals through the Web (for online shopping, travel planning, etc.), semantic hypermedia search can enhance the autonomous behavior of software agents in a Social Ubiquitous Web. To this end, we designed and implemented a prototypical search engine that allows autonomous agents to use approximate search queries for finding relevant resources in their WoT environment in (weak) real time. We demonstrate these features in a maintenance scenario for a prototypical agent-based manufacturing system deployed in one of our laboratories at the University of St. Gallen. Our demonstrator shows that—through these features—the search facility enhances the agents' flexibility and agility in achieving their goals.

We presented an approach to create a hypermedia-driven Social Ubiquitous Web on top of interlinked socio-technical networks. Agents can configure our prototypical search engine to crawl any relation types in these networks, and they can manipulate the networks to "rewire" their environments according to their needs. The current search engine prototype is indexing only W3C WoT TDs, but the same search facility could serve a broader range of purposes. For instance, autonomous agents could use the search engine to discover how to interact with one another based on declarative specifications of agent interaction protocols (e.g., in a formal language such as BSPL [44]) or of multi-agent organizations (e.g., in a formal language such as MOISE OML [23]). Such resources could be designed into the hypermedia environment to further enhance autonomous behavior in a social and ubiquitous Web.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Acosta M, Hartig O, Sequeda J (2018) Federated RDF query processing. Springer International Publishing, Cham, pp 1–8. https://doi.org/10.1007/978-3-319-63962-8_228-1
2. Bienz S, Ciortea A, Mayer S, Gandon F., Corby O (2019) Escaping the streetlight effect: semantic hypermedia search enhances autonomous behavior in the Web of Things. In: 9th International conference on the internet of things. https://www.alexandria.unisg.ch/257439/
3. Boissier O, Bordini RH, Hübner JF, Ricci A, Santi A (2013) Multi-agent oriented programming with jacamo. Sci Comput Program 78(6):747–761. https://doi.org/10.1016/j.scico.2011.10.004. http://www.sciencedirect.com/science/article/pii/S016764231100181X
4. Bordini RH, Hübner JF, Wooldridge M (2007) Programming multi-agent systems in AgentSpeak using Jason, vol 8. Wiley
5. Brin S, Page L (1998) The anatomy of a large-scale hypertextual Web search engine. Comput Netw ISDN Syst 30(1):107–117. https://doi.org/10.1016/S0169-7552(98)00110-X. http://www.sciencedirect.com/science/article/pii/S016975529800110X. Proceedings of the Seventh International World Wide Web Conference
6. Ciortea A, Boissier O, Ricci A (2017) Beyond physical mashups: autonomous systems for the Web of Things. In: Proceedings of the eighth international workshop on the Web of things, WoT 2017. ACM, New York, pp 16–20. https://doi.org/10.1145/3199919.3199924
7. Ciortea A, Boissier O, Ricci AWeyns D, Mascardi V, Ricci A (eds) (2019) Engineering world-wide multi-agent systems with hypermedia. Springer International Publishing, Cham
8. Ciortea A, Mayer S, Gandon F, Boissier O, Ricci A, Zimmermann A (2019) A decade in hindsight: the missing bridge between multi-agent systems and the World Wide Web. In: Proceedings of the 18th international conference on autonomous agents and multiagent systems, AAMAS 2019, Montreal, Canada, May 13-17, 2019. International foundation for autonomous agents and multiagent systems
9. Ciortea A, Mayer S, Michahelles F (2018) 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), pp 813–822. https://www.alexandria.unisg.ch/255802/
10. Ciortea A, Zimmermann A, Boissier O, Florea AM (2015) Towards a social and ubiquitous Web: a model for socio-technical networks. In: 2015 IEEE/WIC/ACM international conference on Web intelligence and intelligent agent technology (WI-IAT), vol 1, pp 461–468. https://doi.org/10.1109/WI-IAT.2015.205
11. Ciortea A, Zimmermann A, Boissier O, Florea AM (2016) Hypermedia-driven socio-technical networks for goal-driven discovery in the Web of Things. In: Proceedings of the seventh international workshop on the Web of Things, WoT '16. ACM, New York, pp 25–30. https://doi.org/10.1145/3017995.3018001
12. Corby O, Dieng-Kuntz R, Gandon F, Faron-Zucker C (2006) Searching the semantic Web: approximate query processing based on ontologies. IEEE Intell Syst 21(1):20–27. https://doi.org/10.1109/MIS.2006.16
13. Corby O, Gaignard A, Faron-Zucker C, Montagnat J (2012) KGRAM versatile inference and query engine for the Web of linked data. In: IEEE/WIC/ACM International conference on Web intelligence, Macao, pp 1–8. https://hal.archives-ouvertes.fr/hal-00746772
14. Cyganiak R, Wood D, Lanthaler M (2014) RDF 1.1 concepts and abstract syntax, W3C recommendation 25 February 2014. W3C Recommendation World Wide Web Consortium (W3C). http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

15. Duerst M, Suignard M (2005) Internationalized resource identifiers (IRIs) RFC 3987 (proposed standard). http://www.ietf.org/rfc/rfc3987.txt
16. Fielding RT, Taylor RN (2002) Principled design of the modern Web architecture. ACM Trans Internet Technol 2(2):115–150. https://doi.org/10.1145/514183.514185
17. Gandon F (2018) A survey of the first 20 years of research on semantic Web and linked data. Revue des Sciences et Technologies de l'Information - Série IS. Ingénierie des Systèmes d'Information. https://doi.org/10.3166/ISI.23.3-4.11-56, https://hal.inria.fr/hal-01935898
18. Genestoux J, Parecki A (2018) WebSub, W3C Recommendation 23 January 2018. W3C Recommendation World Wide Web Consortium (W3C). https://www.w3.org/TR/2018/REC-Websub-20180123/
19. Georgeff MP, Lansky AL (1987) Reactive reasoning and planning. In: AAAI, vol 87, pp 677–682
20. Guinard D, Trifa V, Pham T, Liechti O (2009) Towards physical mashups in the Web of things. In: 2009 Sixth international conference on networked sensing systems (INSS). IEEE, pp 1–4
21. Han S, Brodowsky B, Gajda P, Novikov S, Bendersky M, Najork M, Dua R, Popescul A (2019) Predictive crawling for commercial Web content. In: Proceedings of the 2019 World Wide Web conference, pp 627–637
22. Huang H, Gandon F (2019) Learning URI selection criteria to improve the crawling of linked open data. In: ESWC2019 - The 16th extended semantic Web conference., Portoroz. https://hal.inria.fr/hal-02073854
23. Hübner JF, Sichman JS, Boissier O (2007) Developing organised multiagent systems using the MOISE+ model: programming issues at the system and agent levels. Int J Agent-Oriented Softw Eng 1(3/4):370–395. https://doi.org/10.1504/IJAOSE.2007.016266
24. Jennings NR, Wooldridge M (1998) Applications of intelligent agents. Springer, Berlin, pp 3–28. https://doi.org/10.1007/978-3-662-03678-5_1
25. Kaebisch S, Kamiya T, McCool M, Charpenay V, Kovatsch M (2020) Web of Things (WoT) thing description, W3C Recommendation 9 April 2020. W3C Recommendation World Wide Web Consortium (W3C). https://www.w3.org/TR/2020/REC-wot-thing-description-20200409/
26. Kamilaris A, Yumusak S, Ali MI (2016) Wots2e: a search engine for a semantic Web of Things. In: 2016 IEEE 3rd World forum on internet of things (WF-IoT), pp 436–441, https://doi.org/10.1109/WF-IoT.2016.7845448
27. Kansal A, Nath S, Liu J, Zhao F (2007) SenseWeb: an infrastructure for shared sensing. IEEE MultiMedia 14(4):8–13. https://doi.org/10.1109/MMUL.2007.82
28. Kovatsch M, Matsukura R, Lagally M, Kawaguchi T, Toumura K, Kajimoto K (2020) Web of Things (WoT) architecture, W3C recommendation 9 April 2020. W3C Recommendation World Wide Web Consortium (W3C). https://www.w3.org/TR/2020/REC-wot-architecture-20200409/
29. Lanthaler M, Gütl C (2013) Hydra: a vocabulary for hypermedia-driven Web APIs. In: Proceedings of the WWW2013 workshop on linked data on the Web, CEUR WS, vol 996. http://ceur-ws.org/Vol-996/papers/ldow2013-paper-03.pdf
30. Mayer S, Ciortea A, Ricci A, Robles MI, Kovatsch M, Croatti A (2018) Hypermedia to connect them all: autonomous hypermedia agents and socio-technical interactions. Internet Technol Lett 1(4):e50. https://doi.org/10.1002/itl2.50
31. Mayer S, Guinard D (2011) An extensible discovery service for smart things. In: Proceedings of the second international workshop on Web of things, WoT '11. ACM, New York, pp 7:1–7:6. https://doi.org/10.1145/1993966.1993976
32. Mayer S, Guinard D, Trifa V (2012) Searching in a Web-based infrastructure for smart things. In: 2012 3rd IEEE international conference on the internet of things, pp 119–126, https://doi.org/10.1109/IOT.2012.6402313
33. Mayer S, Verborgh R, Kovatsch M, Mattern F (2016) Smart configuration of smart environments. IEEE Trans Autom Sci Eng 13(3):1247–1255. https://www.alexandria.unisg.ch/255762/
34. Michel F, Faron-Zucker C, Corby O, Gandon F (2019) Enabling automatic discovery and querying of Web APIs at Web scale using linked data standards. In: WWW 2019 - LDOW/LDDL workshop of the world wide Web conference, San Francisco, https://doi.org/10.1145/3308560.3317073. https://hal.archives-ouvertes.fr/hal-02060966
35. Ostermaier B, Römer K, Mattern F, Fahrmair M, Kellerer W (2010) A real-time search engine for the Web of Things. In: 2010 internet of things (IOT), pp 1–8, https://doi.org/10.1109/IOT.2010.5678450
36. Ostermaier B, Römer ., Mattern F, Fahrmair M, Kellerer W (2010) A real-time search engine for the Web of Things. In: Proceedings of Internet of Things 2010 international conference (IoT 2010), Tokyo
37. Page L, Brin S, Motwani R, Winograd T (1999) The pagerank citation ranking: bringing order to the Web. Technical Report 1999-66 Stanford InfoLab. http://ilpubs.stanford.edu:8090/422/. Previous number = SIDL-WP-1999-0120
38. Pfisterer D, Romer K, Bimschas D, Kleine O, Mietz R, Truong C, Hasemann H, Kröller A, Pagel M, Hauswirth M, Karnstedt M, Leggieri M, Passant A, Richardson R (2011) Spitfire: toward a semantic Web of Things. IEEE Commun Mag 49(11):40–48. https://doi.org/10.1109/MCOM.2011.6069708
39. Rao AS (1996) Agentspeak (l): Bdi agents speak out in a logical computable language. In: European Workshop on modelling autonomous agents in a multi-agent world. Springer, pp 42–55
40. Ricci A, Piunti M, Viroli M (2011) Environment programming in multi-agent systems: an artifact-based perspective. Auton Agent Multi-Agent Syst 23(2):158–192
41. Römer K, Ostermaier B, Mattern F, Fahrmair M, Kellerer W (2010) Real-time search for real-world entities: a survey. Proc IEEE 98(11):1887–1902
42. Shelby Z, Koster M, Bormann C, van der Stok P, Amusüss C (2020) CoRE resource directory. Internet Draft. Internet Engineering Task Force (IETF). https://tools.ietf.org/html/draft-ietf-core-resource-directory-24
43. Shoham Y (1993) Agent-oriented programming. Artif Intell 60(1):51–92
44. Singh MP (2011) Information-driven interaction-oriented programming: Bspl, the blindingly simple protocol language. In: The 10th International conference on autonomous agents and multiagent systems - volume 2, AAMAS '11, pp 491–498. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC. http://dl.acm.org/citation.cfm?id=2031678.2031687
45. Singh MP, Huhns MN (2006) Service-oriented computing: semantics, processes, agents. Wiley. https://doi.org/10.1002/0470091509
46. Smith T, Guild J (1931) The c.i.e. colorimetric standards and their use. Trans Opt Soc 33(3):73–134. https://doi.org/10.1088/1475-4878/33/3/301
47. Tan CC, Sheng B, Wang H, Li Q (2010) Microsearch: a search engine for embedded devices used in pervasive computing. ACM Trans Embed Comput Syst 9(4):43,1–43,29. https://doi.org/10.1145/1721695.1721709
48. Villata S, Gandon F (2012) Licenses compatibility and composition in the Web of Data. In: Proceedings of the Third international conference on consuming linked data - volume 905, COLD'12.

CEUR-WS.org, Aachen, pp 124–135. http://dl.acm.org/citation.cfm?id=2887367.2887378
49. Wang H, Tan CC, Li Q (2010) Snoogle: a search engine for pervasive environments. IEEE Trans Parallel Distrib Syst 21(8):1188–1202. https://doi.org/10.1109/TPDS.2009.145
50. Weiss G (2000) Multiagent systems: a modern approach to distributed artificial intelligence. MIT Press
51. Weyns D, Omicini A, Odell J (2007) Environment as a first class abstraction in multiagent systems. Autonom Agents Multi-Agent Syst 14(1):5–30
52. Yap KK, Srinivasan V, Motani M (2005) Max: human-centric search of the physical world. In: Proceedings of the 3rd international conference on embedded networked sensor systems, SenSys '05. ACM, New York, pp 166–179. https://doi.org/10.1145/1098918.1098937
53. Zhou Y, De S, Wang W, Moessner K (2016) Search techniques for the Web of Things: a taxonomy and survey. Sensors 16(5):600