# Repurposing Manufacturing Lines on the Fly with Multi-agent Systems for the Web of Things

Andrei Ciortea
Siemens Corporate Technology
Berkeley, CA 94704, USA
Univ. Lyon, MINES Saint-Étienne,
CNRS Lab Hubert Curien UMR 5516
Saint-Étienne, France
andrei.ciortea@emse.fr

Simon Mayer
Siemens Corporate Technology
Berkeley, CA 94704, USA
Pro2Future GmbH and Graz
University of Technology
Graz, Austria
simon.mayer@pro2future.at

Florian Michahelles
Siemens Corporate Technology
Berkeley, CA 94704, USA
florian.michahelles@siemens.com

## ABSTRACT

Multi-agent systems (MAS) have long been envisioned as a key enabling technology in manufacturing, but this promise is yet to be realized: the lack of proper models, architectures, tooling, and the high level of expertise required for designing and programming agent-based manufacturing systems have hindered their large-scale acceptance. The emerging Web of Things (WoT), now being standardized at the W3C and IETF, provides new research opportunities that could help MAS enter the mainstream. In this paper, we present an approach to design scalable and flexible agent-based manufacturing systems that integrates automated planning with multi-agent oriented programming for the WoT: autonomous agents synthesize production plans using semantic descriptions of Web-based artifacts and coordinate with one another via multi-agent organizations; engineers can program and repurpose the systems on the fly via an intuitive Web user interface. The systems use the Web as an application architecture (and not just as a transport layer), which facilitates the seamless integration of geographically distributed production cells. To demonstrate our approach, we implemented a prototypical production cell that uses industry-grade robots and an augmented reality interface for human workers. Together, these contributions demonstrate a means to achieve an intriguing vision for the forthcoming fourth industrial revolution: a global collective intelligence for manufacturing.

## KEYWORDS

Multi-agent Oriented Programming; Semantic Web; Web of Things; Industry 4.0

**ACM Reference Format:**
Andrei Ciortea, Simon Mayer, and Florian Michahelles. 2018. Repurposing Manufacturing Lines on the Fly with Multi-agent Systems for the Web of Things. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 10 pages.

## 1 INTRODUCTION

The dynamic reconfiguration of manufacturing systems has been a major topic in academic research for several decades [40, 44, 70], but the more recent Industry 4.0 movement [38] further stresses its significance: we are witnessing an accelerating trend towards highly customized products across a broad range of industrial domains. For industry, *mass-customization* means that products in lot sizes of as little as a single item now have to be manufactured at the price of mass-produced goods. This development challenges the practice of designing, verifying, implementing, and commissioning manufacturing lines with target lifespans of more than thirty years. Such systems are well optimized for mass-production, but they are tightly coupled and can hardly be reused or reconfigured for new products – even variations of the same product (e.g., a new product generation) usually require a complete re-engineering of the manufacturing line. In contrast, mass-customization calls for the rapid and dynamic reconfiguration of manufacturing lines.

*Multi-agent systems (MAS)* have long been envisioned as a suitable technology for flexible and adaptive manufacturing lines [39], but the actual number of mature (TRL[1] 8-9) applications deployed to date is small [39, 60]. The lack of proper models, architectures, tooling, and the high level of expertise required to program agent-based manufacturing systems have hindered their acceptance in the industry [47]. Thus, we still witness a large gap between the existing body of research on agent-based manufacturing and applications deployed in the field. In recent years, however, a number of key research topics have been identified that could help bridge this deployment gap. In particular, the *Internet of Things (IoT)* and *cyber-physical systems* have been recognized as potential catalysts for MAS in manufacturing [38, 45, 58]. The integration of MAS with Web services and Semantic Web technologies has also been identified as a potential enabler for (more flexible) agent-based manufacturing systems [47]. More recently, the Web is now emerging as the de facto means for enabling IoT devices and services to interoperate at the application layer, initiative known as the *Web of Things (WoT)* [31]. We believe that the integration of concepts from MAS and the WoT will facilitate the deployment of agent-based manufacturing systems.

In this paper, we present an approach to design scalable and flexible agent-based manufacturing systems that integrates automated planning with *multi-agent oriented programming* for WoT systems. Engineers can repurpose the systems on the fly via an intuitive Web user interface that allows for a trade-off between programming effort and the time required to infer production plans. To achieve this, our approach relies on two novel elements: (i) it integrates *Belief-Desire-Intention (BDI)* agents with first-principles planning

[1]NASA Technology Readiness Level [49]

*Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden.* 

for *Web-based artifacts*, and (ii) it applies semantic multi-agent organizations for the dynamic reconfiguration of manufacturing lines. To demonstrate our approach, we implemented an agent-based manufacturing system for a prototypical production cell that integrates two industry-grade robots and an augmented reality interface for human workers. We used this system to deploy a manufacturing line for assembling stools with various leg configurations.

This paper is structured as follows. In Section 2, we discuss the various research lines that are relevant to our work. We then present our approach in further detail in Section 3. In Section 4, we report on the implementation of our prototype manufacturing line and discuss the strengths and limitations of our approach. We conclude in Section 5.

## 2 BACKGROUND AND RELATED WORK

In this section, we discuss related work from several relevant areas, including the WoT, engineering of Web-based MAS, automated planning and acting in BDI agents, and agent-based manufacturing systems. In particular, we highlight where an integration across these areas is beneficial for creating practically viable agent-based manufacturing systems that are both scalable and flexible.

### 2.1 Industry 4.0 and the Web of Things

The *fourth industrial revolution*, often referred to as "Industrie 4.0" or "Industry 4.0" [38], emerged around 2011 in close relationship with developments in the *Internet of Things (IoT)*: it promotes the networking and interoperability of industrial devices, and the decentralization of decision processes in manufacturing environments. Early research in the IoT was focused on enabling devices to connect at the network layer via the *Internet Protocol (IP)* [16, 69]. Over the past years, however, it became apparent that devices also need to interoperate at the *application layer* in order to unlock the full potential of the IoT vision. On account of its scalable and loosely coupled architecture, the World Wide Web is now widely recognized in academia and industry as a suitable middleware for enabling application-layer interoperability in the IoT via Web standards and protocols (e.g., HTTP [19], CoAP [68]) – effectively interconnecting devices into a *Web of Things (WoT)* [31]. The WoT vision is currently being standardized through combined efforts of the W3C[2], IETF [37], and IRTF[3], with strong support from industrial outfits.

### 2.2 Web-based Multi-agent Systems

The World Wide Web has also raised a lot of interest in the AAMAS community. There has been extensive research on integrating MAS and Web services [27, 34, 35, 73] – and thus using the Web as an infrastructure for distributed MAS. Most of the existing approaches to engineering Web-based MAS use the Web as a transport layer – this includes all MAS platforms that implement the FIPA specification for using HTTP as a message transport protocol [23] (e.g., [14, 17, 29]) as well as those approaches that implement the WS-* standards[4] (SOAP, WSDL, UDDI etc.) [2, 6, 65, 74]. However, systems that use the Web only as a transport layer are misaligned with REST, the architectural style of the Web (see Section 6.5.3 in [20] for a detailed discussion). Consequently, they do not inherit its architectural properties (e.g., scalability, loose coupling), and make limited use of the existing Web infrastructure (e.g., mechanisms for load balancing, caching) or any of its future extensions. In addition, when compared to WS-* services, REST services are lighter and thus better suited for constrained devices [80], and also easier to learn and use for developers [30], both of which are important considerations when designing and programming agent-based manufacturing systems. For all these reasons, the WoT community adopted REST services for integrating devices into the Web [31], and a large part of the broader IoT community followed suit.

More recent approaches to engineering Web-based MAS have also turned to services that adhere to some of the REST principles (e.g., [1, 28, 57]), but they generally do not use hypermedia or hypermedia-driven interaction, one of the core tenets of REST and the Web architecture – a.k.a. *Hypermedia As The Engine of Application State (HATEOAS)* [21]. Two exceptions are [9] and [15]: the former uses HATEOAS to design agent environments for Internet-scale MAS, and the latter applies the *linked data principles* [3] (which partly reflect the HATEOAS principle) to agent environments in order to bring autonomous agents on the Web. Note that both approaches consider the *agent environment* as a first-class abstraction in MAS [78] and a means to deploy MAS on the Web.

In recent years, the use of HATEOAS has been gaining momentum in Web service design, for instance through approaches such as Hydra [42] or RESTdesc [76]. The main benefit of using HATEOAS is that it reduces coupling between clients and service providers, allowing them to be deployed and to evolve independently from one another. These characteristics are particularly important in systems that rely on machine-to-machine interactions, such as manufacturing systems. One approach of particular interest to our work is the *WoT Thing Description* currently being standardized in the *W3C WoT Working Group*; we elaborate on this point in Section 3.1.1.

### 2.3 Planning, Acting, and BDI Agents

Much of the research on autonomous agents, automated reasoning and planning can be traced back to the mid-80s [24], or even the early 70s [22], to the seminal work conducted at the Stanford Research Institute. However, this research has yet to gain mainstream acceptance. Motivated by the low deployment of automated planning techniques in fielded applications, Ghallab et al. proposed recently to shift the research focus on integrating planning and acting (rather than further improving the performance of search algorithms) [25]. To this aim, they introduced an approach that is inspired by and formalizes the *Procedural Reasoning System (PRS)* [36, 61]. A thorough overview of automated planning, and in particular integrating planning and acting, is available in [26].

Another line of research looks at integrating automated planning in *Belief-Desire-Intention (BDI)* agents (a recent thorough survey of this research line is available in [54]). Note that most implementations of the BDI agent architecture have also been inspired by the PRS [24], and one of the most prominent programming languages for BDI agents is *AgentSpeak(L)* [64], which formalizes the operational semantics of PRS in a restricted first-order predicate logic. Unsurprisingly, this line of research is thus closely related to the one mentioned previously.

[2] https://www.w3.org/WoT/WG/, accessed: 09.04.2018.
[3] https://datatracker.ietf.org/rg/t2trg/documents/, accessed: 09.04.2018.
[4] It is now well recognized that WS-* services use the Web only as a transport layer [62].

The above research lines generally focus on *actors* (or *agents*, respectively). In most cases, the *agent environment* is not considered as a first-class abstraction in these systems.[5] To help illustrate this point, for instance, Meneguzzi and Luck [55] introduced an approach that enables a BDI agent to synthesize new plans from plans existing in its plan library. This approach thus enables the BDI agent to extend its capabilities at runtime (e.g., to deal with unforeseen situations). However, the BDI agent cannot synthesize plans from environmental actions discovered at runtime in an *open environment*, such as a WoT environment.

A recent proposal that integrates *hierarchical task network (HTN)* planning with BDI agents and also models the agent environment is PLACE [32], a programming language for agents and their environment. This approach defines a meta-model that is very similar to the *Agents & Artifacts* meta-model [66], but in PLACE an *artifact* is a logical entity and not a physical or computational entity in the environment, such as a device or a Web service.

## 2.4 Agent-Based Manufacturing Systems

Manufacturing is one of the most well-studied application domains for MAS [60]. Multiple surveys of research in this area have been published over the last decade [44, 50, 59, 70, 71], more recent ones in [39, 46, 47]. However, the actual number of mature (TRL[6] 8-9) applications deployed to date is small [39, 60]. Early systems that were successfully deployed as part of large industrial demonstrators had to be programmed and maintained by experts highly-skilled in MAS technology, and any changes required significant work [47]. Consequently, the accumulated costs of such systems – a key factor for acceptance [39] – were not sustainable.

The design of flexible and adaptive manufacturing systems, which would require less human intervention, is still an active area of research. In addition to the surveys mentioned above, recent works cover topics such as realizability of product specifications (e.g., [13]), production planning and scheduling (e.g., [18]), and agent-based architectures (e.g., [8, 67]). Of particular interest to our work, in [67] the authors report on the implementation of a manufacturing line for printed circuit boards that is based on the *JaCaMo meta-model* for *multi-agent oriented programming (MAOP)* [4]. This contribution validates the applicability of MAOP (and JaCaMo) to legacy manufacturing systems, but also highlights the additional difficulty of learning the *MAOP paradigm* (as defined in [4]), even when compared to more traditional agent-oriented programming.

In an attempt to design more flexible manufacturing systems, researchers have also turned towards Web services and the Semantic Web [43, 47]: industrial devices are abstracted as Web services to reduce coupling on the shop floor, and ontologies are used to extend the runtime behavior of a deployed system while minimizing manual integration efforts. Planning agents can discover and use semantic descriptions of services to synthesize production plans. This evolution towards Web services in manufacturing is now recognized as an important step [39, 47], with more recent developments focusing on both *on-device* services and *cloud-based* services (a.k.a. *cloud manufacturing*) [10, 11, 48, 79]. However, many efforts in this direction were influenced by the WS-* standards (e.g., see [43]) and thus suffer from the limitations discussed in Section 2.2. Furthermore, integrating planning and acting (discussed in Section 2.3) remains an open challenge.

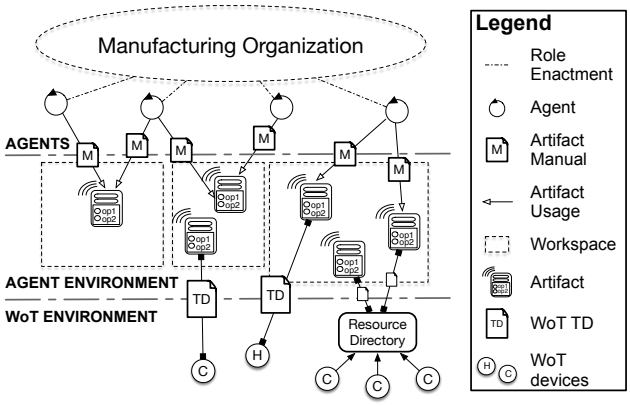



**Figure 1: The various layers of abstraction in our approach.**

# 3 APPROACH

Following our analysis of relevant related work, we integrated these recent developments into an approach to design scalable and flexible agent-based manufacturing systems. The novelty of our approach is two-fold: (i) it integrates PRS-like BDI agents with first-principles planning for Web-based artifacts, and (ii) it applies semantic multi-agent organizations for repurposing manufacturing lines on the fly.

Figure 1 depicts an overview of the layers of abstraction used in our approach. The manufacturing systems use the Web as an application architecture, which induces *scalability* and *evolvability* (cf. Section 2.2, see also [21]). Then, the various programming abstractions (e.g., agents, artifacts, organizations), which were inspired by the JaCaMo meta-model for MAOP [4], further enhance the *modularity* of the systems. Together, modularity, BDI reasoning[7], and automated planning enable the flexible pursuit of manufacturing goals, while the use of high-level programming abstractions (e.g., organizations) and automated planning allows engineers to repurpose the systems on the fly (see also Section 4.1.2).

In the following, in Section 3.1 we first discuss considerations for designing the agent environment and single-agent planning. In Section 3.2, we elaborate on using semantic multi-agent organizations for the dynamic reconfiguration of manufacturing lines.

## 3.1 Integrating BDI Agents with Planning for Web-based Artifacts

Our approach to integrate planning into PRS-like BDI agents goes beyond the state-of-the-art by considering the *agent environment* as a first-class abstraction (cf. Section 2.3): to achieve their manufacturing goals, agents either use plans programmed by engineers, or

[5] Over the past decade, however, the *agent environment* has gained broad recognition as a first-class abstraction in MAS [77, 78].

[6] NASA Technology Readiness Level [49]

[7] The main feature of the BDI agent architecture that makes it suitable for our approach is that it balances reactive and proactive behavior [24]: in pursuit of their manufacturing goals, BDI agents can still react to input from factory workers or to changes in their environment.

they synthesize plans from semantic descriptions of *Web-based artifacts* discovered at runtime. Both programmed and inferred plans are represented in *AgentSpeak* [5]. Engineers can thus use the same language both for writing plans and for inspecting or editing plans synthesized by agents at runtime.

*3.1.1 WoT Environments for Manufacturing.* Following the WoT initiative, industrial devices on the shop floor are abstracted as Web resources, and controlling the devices is then performed by interacting with those resources. For instance, to move a robotic arm to a specific location, a client can use any standard Web interaction protocol, such as HTTP [19] or CoAP [68], to issue a request to update the state of the resource representing the robotic arm. We use the *WoT Thing Description (TD)*[8] currently being standardized in the *W3C WoT Working Group* to enable hypermedia-driven interaction with devices: similar to how Web browsers use HTML forms to interact with heterogeneous Web servers, software clients can use *WoT TDs* to interact with heterogeneous industrial devices on the shop floor. Devices can register their TDs with a resource directory[9] (cf. Figure 1, *WoT Environment*), which enables their dynamic discovery as they are added or replaced on the shop floor.

On the surface, this approach of exposing and advertising Web services on the shop floor might seem very similar to the ones relying on the WS-* standards (SOAP, WSDL, UDDI, etc.), but there are significant differences at the architectural level: by embracing a resource-oriented design and hypermedia-driven interaction, devices are effectively integrated into the Web, and the deployed manufacturing systems can fully benefit from the existing Web infrastructure. This Web-based design is central to our approach: it reduces coupling on the shop floor and enables the seamless integration of geographically distributed production cells.

*3.1.2 Web-based Artifacts.* The proposed architecture for agent-based manufacturing systems follows the *Agents & Artifacts* meta-model [66], in which the *agent environment* is a first-class abstraction: a component designed and programmed with clear-cut responsibilities, which include providing agents with mechanisms for interaction and coordination and with access to industrial devices on the shop floor. The agent environment is composed of dynamic sets of artifacts (i.e., *workspaces*, cf. Figure 1), where artifacts can be both physical entities (e.g., devices on the shop floor) or digital entities (e.g., a planner). Artifacts that model devices are loosely coupled to the WoT environment via WoT TDs.

Note that even though the artifact abstraction[10] and the TD model have been developed independently in two different research communities (and for different purposes), they have similar features: both models expose a uniform interface defined in terms of *observable properties*, *events*, and *operations* (or *actions*, respectively).[11] Therefore, mapping artifacts to TDs is straightforward and provides for a clean vertical integration that facilitates the deployment of MAS in WoT environments.

The use of artifacts as first-class abstractions is a key design choice in our approach. First, the separation of concerns between agents and artifacts simplifies the writing and generation of plans – for instance, plans consist of high-level artifact operations and are insulated from the low-level details of executing HTTP requests. Second, artifacts enhance component reusability since the same artifact can be reused to encapsulate functionally similar devices from different manufacturers. Third, artifacts enhance modularity and enable the independent deployment and evolution of agents and artifacts. All these properties are essential for designing flexible manufacturing systems that can be repurposed on the fly.

*3.1.3 Planning with Artifacts in AgentSpeak.* To achieve their manufacturing goals, agents either use plans that have already been programmed by an engineer, or they use an automated planner to synthesize production plans from semantic descriptions of artifacts available in their environment at runtime. All plans, either programmed or inferred, are represented in *AgentSpeak* [5]. To use a planner, an agent has to specify: (i) the goal to be achieved (i.e., the intended system state), (ii) a serialization of the agent's belief base (i.e., the current system state per the agent's beliefs), and (iii) a list of manuals for available artifacts, where an *artifact manual* consists of semantic descriptions of the artifact's operations. To maintain the alignment with Web standards, all knowledge used in the planning process is represented in RDF [12] using Web ontologies.

```
{
  ?objectState a st:State ;
    log:includes {
        ex:plate ex:locatedIn ex:FinalObjectLocation .
        ex:leg1 ex:attachedTo ex:plate .
        ex:leg2 ex:attachedTo ex:plate .
        ex:leg3 ex:attachedTo ex:plate .
    }.
} => {}.
```

**Listing 1: Manufacturing goal for a stool with three legs.**

The representation of a goal state typically includes an RDF description of an object (or part) to be produced. For instance, Listing 1 shows the goal of manufacturing a stool with three legs (represented as an *N3 rule*[12]): the stool is composed of a plate with three legs attached to it, and the plate is placed at a specific location. This declarative goal can either be provided explicitly as an RDF graph by an engineer (with proper tool support, see [41, 51]), or it can be selected/composed by non-technical workers via a graphical user interface (such as the interfaces we used in [53] and [52]).

An artifact manual consists of a set of *N3 rules*, where each rule describes an artifact operation in terms of: *preconditions* for performing the operation, the *artifact interface* used to perform the operation, and the operation's *postconditions*. For illustrative purposes, Listing 2 shows an N3 rule that describes the operation of moving an empty robotic arm to a location given in a three dimensional Cartesian coordinate system.[13] The rule also provides a description of how to perform the operation using the interface of a *CArtAgO artifact* [66] (see Section 4 for implementation details), where the interface description includes the name of the operation to be called, its input parameters, and their data types (cf. Listing 2).

[8] https://www.w3.org/TR/wot-thing-description/, accessed: 09.04.2018.
[9] https://tools.ietf.org/html/rfc6690, accessed: 09.04.2018.
[10] As defined by *Agents & Artifacts* [66].
[11] The Thing Description model is slightly more generic however: TD properties are writable, whereas artifact properties are read-only.
[12] https://www.w3.org/TeamSubmission/n3/, accessed: 09.04.2018.
[13] An artifact operation can have multiple N3 rules attached to it. For instance, we defined a rule similar to the one in Listing 2 to describe the operation of moving the robotic arm when an object is grabbed. The two rules thus describe using the same artifact operation in two different contexts (the differences between them being in terms of preconditions and postconditions).

For this purpose, we developed an OWL ontology for describing CArtAgO environments, but any ontology for any other platform that implements the *Agents & Artifacts* meta-model could also be used.

```
{
  ?objectState a st:State ;
      log:includes {
        ex:PLCArmGripper ex:locatedIn ?gripperposition ;
            ex:grabbed "none".
    } .
    ?gripperposition a ex:PhysicalLocation3D .
    ?destination a ex:PhysicalLocation3D ;
        ex:hasX ?x ; ex:hasY ?y ; ex:hasZ ?z .
}
=>
{
  [ a cartago:Operation ;
      cartago:hasName "move" ;
      cartago:hasInputParameters [
          a rdf:Seq ;
          rdf:_1 "?x"^^xsd:decimal ; rdf:_2 "?y"^^xsd:decimal ;
          rdf:_3 "?z"^^xsd:decimal ;
      ]
  ] .

  [ a st:StateChange ;
      st:replaced {
          ex:PLCArmGripper ex:locatedIn ?destination .
      } ;
      st:parent ?objectState
  ] .
} .
```

**Listing 2: Semantic description of an artifact operation for moving an empty robotic arm.**

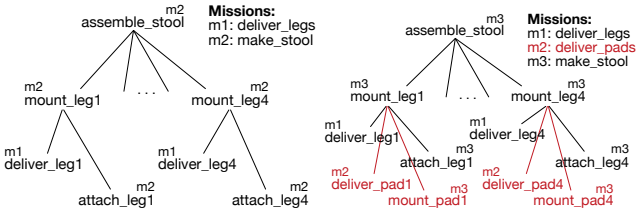
If a solution is found, the planner returns an *AgentSpeak* plan whose body is a sequence of artifact operations. The agent can then add the new plan to its library of plans and execute it to achieve its manufacturing goal. For illustrative purposes, Listing 3 shows a synthesized *AgentSpeak* plan for moving an object from one location to another. In this example, the agent retrieved the semantic description of the `deliver_pad1` goal (which is used to invoke the automated planning operation) from the specification of a *manufacturing organization* (discussed next).

```
+!deliver_pad1 : true <-
    move(-3, 20, 14)[artifact_name("plc_arm")];
    grab[artifact_name("plc_arm")];
    move(20, -3, 14)[artifact_name("plc_arm")];
    release[artifact_name("plc_arm")].
```

**Listing 3: Synthesized AgentSpeak plan for moving an object using a robotic arm artifact.**

## 3.2 Manufacturing Organizations

In the previous section, we discussed production planning from a centralized, single-agent perspective. A decentralized, multi-agent design: (i) can decrease production planning time by splitting the search space into independent sub-spaces, and (ii) can further enhance modularity and the independent development and deployment of components. However, adopting a multi-agent design also raises the problem of agent coordination.




**(a) Stool with 4 legs.** **(b) Stool with 4 padded legs.**

**Figure 2: Stool assembly schemes.**

To address this problem, we use multi-agent organizations, and in particular MOISE [33], to coordinate communities of industrial agents. A MOISE organization is defined on three dimensions [33]: a *structural dimension*, which defines roles and groups within the organization; a *functional dimension*, which defines goals, missions (i.e., sets of goals), and coordination schemes; and a *normative dimension*, which assigns missions to roles via norms (e.g., obligations, permissions). Norms provide a means to hot-deploy coordination schemes in running MAS (assuming all agents are norm-aware), which enables the dynamic reconfiguration of manufacturing lines.

We model MOISE organizations in RDF using the ontology introduced in [81]. This design choice allows us to reuse standard tooling for storing and querying organizational specifications, and enables extensibility – for instance, to attach RDF descriptions to MOISE goals that can then be used for automated planning.

*3.2.1 Agentification and Coordination.* Ascribing agency to system components (a.k.a. "agentification" [72]) is a key design choice in the engineering of MAS. In agent-based manufacturing systems, two main agentification approaches have been recognized [70]: *functional decomposition*, in which agents encapsulate functional modules (e.g., planning, scheduling, material handling, transportation management), and *physical decomposition*, in which agents represent physical entities (e.g., industrial devices on the shop floor).[14] In the functional decomposition design, agents tend to share many state variables, while in the physical decomposition design agents tend to manage independent sets of state variables and require few interactions with one another.

We adopt the latter design and conceive of a manufacturing system as a socio-technical system in which agents can be either human workers or autonomous software entities that manage one or more industrial devices (i.e., *physical* artifacts). We assume agents manage independent sets of state variables, and thus agent coordination can happen before the planning phase.

To illustrate our use of multi-agent organizations, Figure 2a depicts a MOISE coordination scheme (represented as a goal decomposition tree) for manufacturing a four-legged stool: assembling the stool requires mounting each of the four legs, and mounting each leg requires that the leg is first delivered to a designated location and then attached to the stool plate. In this example, non-leaf goals are decomposed into sub-goals achieved in *sequence*, but a goal could also be decomposed into sub-goals that are achieved in *parallel* (see [33]). Goals are grouped into *missions* that are assigned

[14] Complex systems, however, generally follow hybrid approaches [70].

**Algorithm 1** Algorithm for probing the deployment of a manufacturing scheme.

```
1: function ComputeSchemeDiff
2:     for all roles relevant to a scheme do
3:         determine set G of goals role r is obliged to achieve
4:         for all known agents do
5:             agent_rating ← number of achievable goals from G
6:     return ratings for all known agents
```

via *obligation norms* to two *roles* (cf. Figure 2a): `human_worker` and `leg_transporter_robot`. Roles provide an indirection level that effectively decouples the manufacturing organization from individual agents (and thus increases its reusability).

*3.2.2 Probing Manufacturing Organizations.* Manufacturing organizations provide formal specifications that can also be used *to probe* if an agent-based manufacturing system is able to manufacture a given product. If the deployed system cannot readily manufacture the product, the organizational specification can help determine what are the missing capabilities of the production cell, or even determine components that could be procured from or delegated to other production cells.

To demonstrate this capability, we developed Algorithm 1 for probing the deployment of manufacturing organizations designed with MOISE [33]. The core idea is that the *meaning of a role* in a MOISE organization is determined by the set of all *goals* attached to that *role* via *norms*. Therefore, to probe the deployment of a manufacturing scheme, the algorithm selects all roles that are relevant for that scheme and, for each role, it evaluates all known agents based on the number of goals that they can achieve for that role (in our implementation the evaluation is based on semantic descriptions of agents and the goals they can achieve). A complete (or partial) solution for deployment can then be presented to an engineer, who can decide on a course of action for manufacturing the product.

For illustrative purposes, Figure 2b depicts a manufacturing organization for a stool with four *padded* legs. In contrast to the scheme in Figure 2a, manufacturing this stool requires an agent that can enact the role of `pad_transporter_robot` to deliver the pads. Furthermore, the `make_stool` mission now includes additional goals for mounting the pads. The system can evaluate this organizational specification using Algorithm 1 in order to propose a solution for its deployment to an engineer.

## 4 IMPLEMENTATION AND DEPLOYMENT

To demonstrate our approach, we implemented an agent-based manufacturing system for a prototypical production cell that integrates two industry-grade robots and an *augmented reality (AR)* interface for human workers. We present the main system components in Section 4.1, and discuss the prototype deployment in Section 4.2. A video that demonstrates the main features of our research prototype is available online.[15]

[15] https://youtu.be/tfAVDpYn_ow, accessed: 09.04.2018.

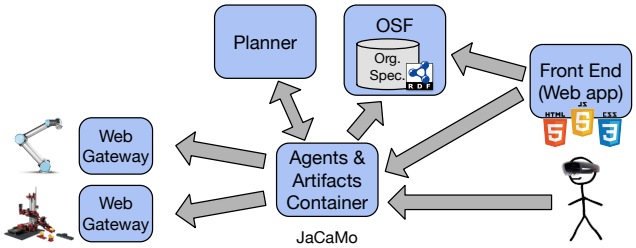



**Figure 3: System architecture.**

### 4.1 System Architecture

Figure 3 depicts the main components of the implemented system, which interact with one another via HTTP. To store and manage semantic models (e.g., organizational specifications, agent descriptions), we use the *Open Semantic Framework* (OSF) [51], a standard-compliant industrial knowledge management platform. Software agents and the artifacts they use run in an *Agents & Artifacts Container* implemented with the JaCaMo platform [4], and schedule planning jobs on a remote machine. The automated planning system is based on the Eye Reasoner [75], which is wrapped in a REST HTTP API.

When participating in manufacturing organizations, human workers interact with the system via an AR interface for *Microsoft HoloLens*[16] devices. For simplicity, in our implementation human workers are proxied by software agents: the agents forward to humans all their obligations (e.g., to achieve goals), and use *organizational artifacts* on behalf of humans to signal the fulfillment of those obligations (e.g., the achievement of goals). Obligations are displayed to human workers in textual form, and workers use voice commands to notify when they have fulfilled their obligations.

In the next two sections, we first describe the *Agents & Artifacts Container* and then present our system's engineering user interface that can be used for repurposing deployed systems.

*4.1.1 Agents & Artifacts Container.* The *Agents & Artifacts (A&A) Container* uses a custom extension of the MOISE framework [4] for reading organizational specifications in RDF (rather than the usual XML-based specifications). The A&A Container provides agents with two infrastructure artifacts: a *planner artifact* and a *Web interface artifact*.

Agents use the *planner artifact* to synthesize plans in a non-blocking manner: they schedule planning jobs, and once a job is finished, they receive a notification with the result. Agents are thus effectively decoupled from the automated planning system. If a solution is found, the notification payload includes a representation of the plan as an *RDF Sequence* [7] of CArtAgO operation descriptions (such as the one shown in Listing 2). The RDF sequence is translated to an *AgentSpeak* plan and added to the calling agent's library of plans.

The *Web interface artifact* exposes a REST HTTP API that is used by both the AR interface and the engineering front end to interact with the MAS. For instance, clients can use this API to send notifications to individual agents, or to create organizational

[16] https://www.microsoft.com/en-us/hololens, accessed: 09.04.2018.

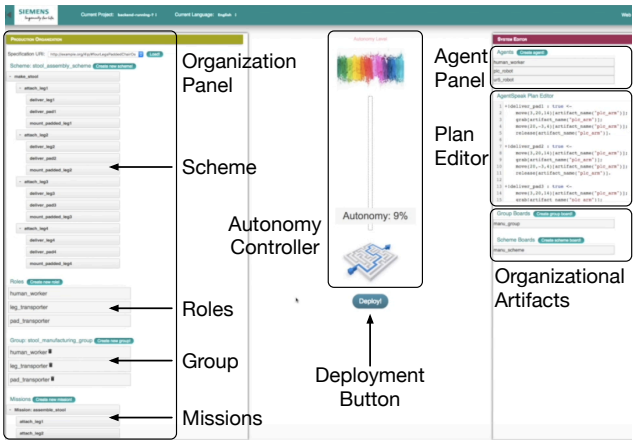



**Figure 4: Engineering front end. Users can set the general autonomy level of the system via a central controller (center) and can guide/constrain the system by specifying organizations (left) and programming concrete plans (right).**

artifacts (i.e., *group boards* or *scheme boards* [4]). When a valid HTTP request is received, the *Web interface artifact* processes the resource representation in the request payload and dispatches notifications to all interested agents. For instance, the representation of a group board can include role allocations for agents – in which case, when a group board is created, all participating agents are notified of its creation and the roles they are required to enact in it.

*4.1.2 Engineering Front End.* To enable high-level control by a production engineer, the system exposes a graphical Web user interface (see Figure 4) that can be used from stationary and mobile devices alike. This interface is structured around a central *autonomy controller* (center of Figure 4) that is used for setting the global autonomy level of the system. If set to the maximum, the system attempts to synthesize a production plan purely based on first-principles planning and using the semantic descriptions of discoverable Web-based artifacts with no further input from the engineer (except the manufacturing goal). For lower settings of the controller, the interface is enriched with more and more options that enable engineers to guide/constrain the system: they can define (arbitrarily deep) goal decomposition trees, organizational roles, groups, missions, and assign missions to roles (left panel of Figure 4), and even program plans in *AgentSpeak* (right panel). When satisfied with the system configuration, a production engineer can start the manufacturing process by clicking the *"Deploy!"* button.

## 4.2 Prototype Deployment

In what follows, we report on the deployment of our prototype. First, we introduce the concrete use case scenario of assembling customized furniture, and then we present our prototypical production cell and the scenario implementation. Lastly, we discuss the strengths and current limitations of our approach.

*4.2.1 Flexible Assembly of Customized Furniture.* To validate our prototype implementation, we considered the assembly of customized furniture. This market seems particularly appealing for flexible manufacturing systems both due to high potential and high demand for customization: even when restricting our scope to only a few different types of furniture, colors, leg configurations (e.g., three-legged vs. four-legged stools), and types of floor protector pads, the manufacturing line already faces several hundreds of product variations. This scenario thus warrants the use of manufacturing systems that are capable of switching between product variants and can be extended with additional industrial devices. The concrete example we use is switching between the assembly of several different variants of a wooden stool: our system should be able to continuously manufacture *stools with three legs*; this behavior can then be interrupted by requests to assemble stools with a *four-leg configuration*, which does not require any adjustment of the manufacturing line itself but merely of the behaviors of involved agents; furthermore, the system should react appropriately to assembly requests for stools with *padded legs* (in any leg configuration) – in this case, the system should transparently add another agent (a second robot) that delivers felt pads to the assembly station.

*4.2.2 Prototypical Production Cell.* We deployed the presented system in a prototypical production cell in our laboratory. The cell contains two handling robots that interact with human workers. The robots are controlled by systems that are representative for a state-of-the-art production cell in a real manufacturing environment. The first of the two handling robots is a *Universal Robotics UR5* robot with an attached *Robotiq 2-Finger Gripper* that is connected to the rest of the system via ROS [63], a Linux-based robot programming framework that includes modules for movement planning, optimization, and visualization. The second robot in our laboratory production cell is a *Fischertechnik ROBO TX Automation Robot* that is controlled via an industrial controller (a *Siemens S7-300 PLC*) and is programmed by means of the *STEP7* language using ladder logic. Both robot controllers expose REST HTTP APIs for integration with the rest of the system (cf. Figure 3).[17]

This setup allows us to demonstrate that our system enables *collaborative robots* (i.e., the UR5), which are widely perceived as important future participants in industrial production processes, to interact with devices that are controlled via *conventional factory automation systems* such as PLCs. Furthermore, by building on top of the WoT, our system can be seamlessly extended with any Web-enabled devices that expose WoT TDs for their Web APIs.

*4.2.3 Scenario Implementation.* We implemented the stool assembly scenario in our prototypical production cell using the UR5 robot for delivering stool legs, the Fischertechnik robot for delivering pads, and a human worker for assembling the stool.[18] The engineering front end shown in Figure 4 displays the system configuration for a stool with four padded legs. When the autonomy lever is at 99%, the only input required by the system is an RDF description of the final goal (similar to the one shown in Listing 1), which is retrieved from OSF. In this case, the system uses a single planning agent that controls both robots to deliver the parts. As the engineer lowers the autonomy lever, the front end gradually refines the organizational specification with knowledge retrieved

[17] For more technical details about the setup of the laboratory production cell we refer interested readers to [53].
[18] A demonstrator video is available online: https://youtu.be/tfAVDpYn_ow, accessed: 09.04.2018.

**Table 1: Agent ratings for roles in the manufacturing organization for assembling a stool with four padded legs.**

| Role | human worker | ur5 robot | fischertechnik robot |
|---|---|---|---|
| human_worker | 9/9 | 0/9 | 0/9 |
| leg_transporter_robot | 4/4 | 4/4 | 0/4 |
| pad_transporter_robot | 4/4 | 0/4 | 4/4 |

from OSF. In Figure 4, the autonomy lever is at 9% and the front end displays a 2-level goal decomposition tree, which is the complete coordination scheme for this scenario.

The engineer can easily switch production between stools with various leg configurations. For instance, when switching production from *3-legged stools* to *4-legged stools* (without pads), the new product variant fits within the capabilities of the deployed manufacturing line. The engineer can adjust the 3-legged stool manufacturing organization using the front end, or by loading the corresponding organizational specification from OSF. If the engineer next switches to *4-legged stools with pads*, the production now exceeds the capabilities of the manufacturing line. When probing the corresponding manufacturing organization, the system retrieves from OSF semantic descriptions of known agents to compute and display the role ratings in Table 1 (i.e., the number of goals an agent can achieve for each role). The engineer can use this information to bring online the Fischertechnik robot for delivering the pads, and can then deploy the manufacturing organization.

*4.2.4 Discussion and Limitations.* Our setup demonstrates that the proposed design for agent-based manufacturing systems attains the required properties that motivate our work. First, the deployed systems are modular and flexible, which allows them to be repurposed on the fly – for instance, in response to high customization demands in markets such as furniture manufacturing. Second, this approach facilitates the programming and repurposing of manufacturing lines: engineers use high-level concepts to introduce knowledge into deployed systems via an intuitive interface that allows for a flexible trade-off between programming effort vs. the computational effort spent by the system to arrive to a solution (if any). This approach thus has the potential to significantly lower the engineering costs – a key factor for acceptance of MAS technology in manufacturing [39]. We leave it as future work to perform an empirical evaluation of the effectiveness of this interface and to study the use of domain terminology for further increasing the usability of MAS concepts for production engineers.

The WoT provides an effective means to deal with heterogeneity on the shop floor: WoT TDs allow the *A&A Container* to discover at runtime how to interact with heterogeneous industrial devices (PLCs, robots etc.). Furthermore, by piggybacking on the Web architecture, the deployed manufacturing systems inherit the properties of the Web as an Internet-scale and long-lived system [21, 31]. We currently argue that these architectural properties are preserved based on our theoretical understanding of the Web architecture, but in the future we intend to build and deploy large-scale, geographically distributed demonstrators to support this claim.

The automated planning system used in our implementation inherits assumptions that are specific to the classical planning domain, such as [26]: the environment is static, there is no explicit model of time (e.g., how long an action will last), concurrency is not considered (only discrete sequences of states and actions), all actions are deterministic (the planning algorithm assumes that it can predict with certainty what state is produced if an action $a$ is performed in state $s$). Some of these assumptions are relaxed by our integration with BDI reasoning (e.g., coping with dynamic environments, recovery from failed actions). In addition, our modular and loosely coupled architecture enables the easy replacement of the current automated planning system with any other system (e.g., systems that consider temporal models [26]).

At the moment, we do not address the use of variables in synthesized AgentSpeak plans: all arguments to artifact operations are passed by value (cf. Listing 3), which limits the reusability of synthesized plans. Closely related, we also do not refine the application context of inferred plans: the inferred context is always `true`, and thus agents do not have a proper means to select inferred plans when achieving their goals (this issue has already been addressed in related work, e.g. [56]). We intend to address these issues in the future.

## 5 CONCLUSIONS AND OUTLOOK

Manufacturing has been envisioned as a major application domain for MAS since the late 1980s, yet the actual number of mature applications deployed to date is small. This paper presents an approach to design scalable and flexible agent-based manufacturing systems that attempts to bridge this deployment gap by integrating automated planning with MAOP for the WoT. The use of MAOP with WoT TDs enhances the modularity of manufacturing systems – and together, modularity, BDI reasoning, and automated planning allow the systems to pursue manufacturing goals in a flexible manner. The use of high-level programming abstractions and automated planning allows to hide the system complexity behind an intuitive interface for programming and reconfiguring organizations of industrial agents on the fly. This can effectively reduce the time and effort required to repurpose manufacturing lines for small batch sizes – and thus lowers the engineering costs, a key factor for the adoption of MAS in manufacturing. We demonstrated these properties in a fully functional prototype that was deployed in a laboratory production cell using realistic industrial equipment.

The WoT plays a central role in our approach: it enhances interoperability and reduces coupling among devices on the shop floor. Furthermore, it allows manufacturing systems to exploit the existing Web infrastructure and inherit its architectural properties – thus promoting the vision of world-wide manufacturing systems. Together with MAS and the use of AR for seamless collaboration between human workers and industrial agents, we believe this integration outlines an intriguing vision for the fourth industrial revolution: a global collective intelligence for manufacturing.

## ACKNOWLEDGMENTS

This work was supported by the Austrian FFG as part of the COMET K1 Centre $Pro^2Future$.

## REFERENCES


[1] Abdullah Althagafi. 2012. *Designing a Framework for RESTful Multi-Agent Systems.* Master's thesis. Department of Computer Science, University of Saskatchewan, Saskatoon, Canada.

[2] Estefania Argente, Vicente Botti, Carlos Carrascosa, Adriana Giret, Vicente Julian, and Miguel Rebollo. 2011. An abstract architecture for virtual organizations: The THOMAS approach. *Knowledge and Information Systems* 29, 2 (01 Nov 2011), 379–403. https://doi.org/10.1007/s10115-010-0349-1

[3] Christian Bizer, Tom Heath, and Tim Berners-Lee. 2011. Linked Data: The Story so Far. In *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, Amit Sheth (Ed.). IGI Global, 205–227.

[4] Olivier Boissier, Rafael H. Bordini, Jomi F. Hübner, Alessandro Ricci, and Andrea Santi. 2013. Multi-agent oriented programming with JaCaMo. *Science of Computer Programming* 78, 6 (2013), 747 – 761. https://doi.org/10.1016/j.scico.2011.10.004

[5] Rafael H. Bordini, Jomi F. Hübner, and Michael Wooldridge. 2007. *Programming multi-agent systems in AgentSpeak using Jason.* John Wiley & Sons.

[6] Lars Braubach and Alexander Pokahr. 2013. Conceptual Integration of Agents with WSDL and RESTful Web Services. In *Programming Multi-Agent Systems*, Mehdi Dastani, Jomi F. Hübner, and Brian Logan (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 17–34.

[7] Dan Brickley and Ramanathan V. Guha. 2014. *RDF Schema 1.1.* W3C Recommendation. World Wide Web Consortium (W3C). http://www.w3.org/TR/2014/REC-rdf-schema-20140225/

[8] J.C. Chaplin, O.J. Bakker, L. de Silva, D. Sanderson, E. Kelly, B. Logan, and S.M. Ratchev. 2015. Evolvable Assembly Systems: A Distributed Architecture for Intelligent Manufacturing. *IFAC-PapersOnLine* 48, 3 (2015), 2065 – 2070. https://doi.org/10.1016/j.ifacol.2015.06.393 15th IFAC Symposium on Information Control Problems in Manufacturing.

[9] Andrei Ciortea, Olivier Boissier, Antoine Zimmermann, and Adina Magda Florea. 2017. Give Agents Some REST: A Resource-oriented Abstraction Layer for Internet-scale Agent Environments. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS '17)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1502–1504. http://dl.acm.org/citation.cfm?id=3091282.3091342

[10] Armando Colombo, Thomas Bangemann, Stamatis Karnouskos, Jerker Delsing, Petr Stluka, Robert Harrison, Francois Jammes, and Jose L. Lastra. 2014. *Industrial Cloud-Based Cyber-Physical Systems: The IMC-AESOP Approach.* Springer Publishing Company, Incorporated. https://doi.org/10.1007/978-3-319-05624-1

[11] A. W. Colombo, S. Karnouskos, O. Kaynak, Y. Shi, and S. Yin. 2017. Industrial Cyberphysical Systems: A Backbone of the Fourth Industrial Revolution. *IEEE Industrial Electronics Magazine* 11, 1 (March 2017), 6–16. https://doi.org/10.1109/MIE.2017.2648857

[12] Richard Cyganiak, David Wood, and Markus Lanthaler. 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/

[13] Lavindra de Silva, Paolo Felli, Jack C Chaplin, Brian Logan, David Sanderson, and Svetan Ratchev. 2016. Realisability of Production Recipes. In *Proceedings of the 22nd European Conference on Artificial Intelligence*. IOS Press, 1449–1457. http://doi.org/10.3233/978-1-61499-672-9-1449

[14] Oğuz Dikenelli. 2008. SEAGENT MAS Platform Development Environment. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Demo Papers (AAMAS '08)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1671–1672. http://dl.acm.org/citation.cfm?id=1402744.1402758

[15] Oğuz Dikenelli, Oylum Alatlı, and Rıza Cenk Erdur. 2015. Where Are All the Semantic Web Agents: Establishing Links Between Agent and Linked Data Web Through Environment Abstraction. In *Agent Environments for Multi-Agent Systems IV*, Danny Weyns and Fabien Michel (Eds.). Springer International Publishing, Cham, 41–51.

[16] Adam Dunkels. 2003. Full TCP/IP for 8-bit Architectures. In *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services (MobiSys '03)*. ACM, New York, NY, USA, 85–98. https://doi.org/10.1145/1066116.1066118

[17] Jose Exposito, Joan Ametller, and Sergi Robles. 2010. Configuring the JADE HTTP MTP. http://jade.tilab.com/documentation/tutorials-guides/configuring-the-jade-http-mtp/. (2010). Accessed: 20.04.2018.

[18] Paolo Felli, Lavindra De Silva, Brian Logan, and Svetan Ratchev. 2017. Process Plan Controllers for Non-deterministic Manufacturing Systems. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. AAAI Press, 1023–1030. http://dl.acm.org/citation.cfm?id=3171642.3171788

[19] R. Fielding and J. Reschke. 2014. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. RFC 7231 (Proposed Standard). (June 2014). http://www.ietf.org/rfc/rfc7231.txt

[20] Roy T Fielding. 2000. *Architectural styles and the design of network-based software architectures.* Ph.D. Dissertation. University of California, Irvine.

[21] Roy T. Fielding and Richard N. Taylor. 2002. Principled Design of the Modern Web Architecture. *ACM Trans. Internet Technol.* 2, 2 (May 2002), 115–150. https://doi.org/10.1145/514183.514185

[22] Richard E. Fikes and Nils J. Nilsson. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2, 3 (1971), 189 – 208. https://doi.org/10.1016/0004-3702(71)90010-5

[23] Foundation for Intelligent Physical Agents. 2002. FIPA Agent Message Transport Protocol for HTTP Specification. http://www.fipa.org/specs/fipa00084/SC00084F.html. (2002). Document number: SC00084F.

[24] Michael P. Georgeff and Amy L. Lansky. 1987. Reactive Reasoning and Planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 2 (AAAI'87)*. AAAI Press, 677–682. http://dl.acm.org/citation.cfm?id=1856740.1856792

[25] Malik Ghallab, Dana Nau, and Paolo Traverso. 2014. The actor's view of automated planning and acting: A position paper. *Artificial Intelligence* 208 (2014), 1 – 17. https://doi.org/10.1016/j.artint.2013.11.002

[26] Malik Ghallab, Dana Nau, and Paolo Traverso. 2016. *Automated Planning and Acting* (1st ed.). Cambridge University Press, New York, NY, USA.

[27] Nicholas Gibbins, Stephen Harris, and Nigel Shadbolt. 2004. Agent-based Semantic Web Services. *Web Semantics: Science, Services and Agents on the World Wide Web* 1, 2 (2004), 141 – 154. https://doi.org/10.1016/j.websem.2003.11.002 2003 World Wide Web Conference.

[28] Abdelkader Gouaïch and Michael Bergeret. 2010. REST-A: An Agent Virtual Machine Based on REST Framework. In *Advances in Practical Applications of Agents and Multiagent Systems*, Yves Demazeau, Frank Dignum, Juan M. Corchado, and Javier Bajo Pérez (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 103–112.

[29] Miguel Escrivá Gregori, Javier Palanca Cámara, and Gustavo Aranda Bada. 2006. A Jabber-based Multi-agent System Platform. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '06)*. ACM, New York, NY, USA, 1282–1284. https://doi.org/10.1145/1160633.1160866

[30] Dominique Guinard, Iulia Ion, and Simon Mayer. 2012. In Search of an Internet of Things Service Architecture: REST or WS-*? A Developers' Perspective. In *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, Alessandro Puiatti and Tao Gu (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 326–337.

[31] D. Guinard, V. Trifa, and E. Wilde. 2010. A resource oriented architecture for the Web of Things. In *2010 Internet of Things (IOT)*. 1–8. https://doi.org/10.1109/IOT.2010.5678452

[32] Muhammad Adnan Hashmi, Muhammad Usman Akram, and Amal El Fallah Seghrouchni. 2017. PLACE: Planning based Language for Agents and Computational Environments. In *Proceedings of the 5th International Workshop on Engineering Multi-Agent Systems (EMAS)*. ACM.

[33] Jomi F. Hübner, Jaime S. Sichman, and Olivier Boissier. 2007. Developing organised multiagent systems using the MOISE+ model: programming issues at the system and agent levels. *International Journal of Agent-Oriented Software Engineering* 1, 3-4 (2007), 370–395. https://doi.org/10.1504/IJAOSE.2007.016266

[34] M. N. Huhns. 2002. Agents as Web services. *IEEE Internet Computing* 6, 4 (Jul 2002), 93–95. https://doi.org/10.1109/MIC.2002.1020332

[35] M. N. Huhns and M. P. Singh. 2005. Service-oriented computing: key concepts and principles. *IEEE Internet Computing* 9, 1 (Jan 2005), 75–81. https://doi.org/10.1109/MIC.2005.21

[36] F. F. Ingrand, M. P. Georgeff, and A. S. Rao. 1992. An architecture for real-time reasoning and system control. *IEEE Expert* 7, 6 (Dec 1992), 34–44. https://doi.org/10.1109/64.180407

[37] Isam Ishaq, David Carels, Girum K. Teklemariam, Jeroen Hoebeke, Floris Van den Abeele, Eli De Poorter, Ingrid Moerman, and Piet Demeester. 2013. IETF Standardization in the Field of the Internet of Things (IoT): A Survey. *Journal of Sensor and Actuator Networks* 2, 2 (2013), 235–287. https://doi.org/10.3390/jsan2020235

[38] Henning Kagermann, Wolfgang Wahlster, and Johannes Helbig. 2013. Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry. (2013). Final report of the Industrie 4.0 Working Group.

[39] S. Karnouskos and P. Leitão. 2017. Key Contributing Factors to the Acceptance of Agents in Industrial Environments. *IEEE Transactions on Industrial Informatics* 13, 2 (April 2017), 696–703. https://doi.org/10.1109/TII.2016.2607148

[40] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy, and H. Van Brussel. 1999. Reconfigurable Manufacturing Systems. *CIRP Annals* 48, 2 (1999), 527 – 540. https://doi.org/10.1016/S0007-8506(07)63232-6

[41] Matthias Kovatsch, Yassin Nasir Hassan, and Simon Mayer. 2015. Practical semantics for the Internet of Things: Physical states, device mashups, and open questions. In *5th International Conference on the Internet of Things, IOT 2015, Seoul, South Korea, 26-28 October, 2015*. 54–61. https://doi.org/10.1109/IOT.2015.7356548

[42] Markus Lanthaler and Christian Gütl. 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

[43] J. L. M. Lastra and M. Delamer. 2006. Semantic web services in factory automation: fundamental insights and research roadmap. *IEEE Transactions on Industrial Informatics* 2, 1 (Feb 2006), 1–11. https://doi.org/10.1109/TII.2005.862144

[44] Paulo Leitão. 2009. Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering Applications of Artificial Intelligence* 22, 7 (2009), 979 – 991. https://doi.org/10.1016/j.engappai.2008.09.005 Distributed Control of Production Systems.

[45] Paulo Leitão, Armando Walter Colombo, and Stamatis Karnouskos. 2016. Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges. *Computers in Industry* 81 (2016), 11 – 25. https://doi.org/10.1016/j.compind.2015.08.004 Emerging ICT concepts for smart, safe and sustainable industrial systems.

[46] Paulo Leitão and Stamatis Karnouskos. 2015. *Industrial Agents: Emerging Applications of Software Agents in Industry* (1st ed.). Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands.

[47] Paulo Leitão, Vladimír Mařík, and Pavel Vrba. 2013. Past, Present, and Future of Industrial Agent Applications. *IEEE Transactions on Industrial Informatics* 9, 4 (Nov 2013), 2360–2372. https://doi.org/10.1109/TII.2012.2222034

[48] Yuqian Lu, Xun Xu, and Jenny Xu. 2014. Development of a Hybrid Manufacturing Cloud. *Journal of Manufacturing Systems* 33, 4 (2014), 551 – 566. https://doi.org/10.1016/j.jmsy.2014.05.003

[49] John C. Mankins. 1995. Technology Readiness Levels: A White Paper. (April 1995). NASA, Office of Space Access and Technology, Advanced Concepts Office.

[50] Vladimír Mařík and Jiří Lažanskỳ. 2007. Industrial applications of agent technologies. *Control Engineering Practice* 15, 11 (2007), 1364 – 1380. https://doi.org/10.1016/j.conengprac.2006.10.001 Special Issue on Manufacturing Plant Control: Challenges and Issues.

[51] S. Mayer, J. Hodges, D. Yu, M. Kritzler, and F. Michahelles. 2017. An Open Semantic Framework for the Industrial Internet of Things. *IEEE Intelligent Systems* 32, 1 (Jan 2017), 96–101. https://doi.org/10.1109/MIS.2017.9

[52] Simon Mayer, Nadine Inhelder, Ruben Verborgh, Rik Van de Walle, and Friedemann Mattern. 2014. Configuration of smart environments made simple: Combining visual modeling with semantic metadata and reasoning. In *4th International Conference on the Internet of Things, IOT 2014, Cambridge, MA, USA, October 6-8, 2014*. 61–66. https://doi.org/10.1109/IOT.2014.7030116

[53] Simon Mayer, Dominic Plangger, Florian Michahelles, and Simon Rothfuss. 2016. UberManufacturing: A Goal-Driven Collaborative Industrial Manufacturing Marketplace. In *Proceedings of the 6th International Conference on the Internet of Things*. ACM, 111–119.

[54] Felipe Meneguzzi and Lavindra De Silva. 2015. Planning in BDI agents: a survey of the integration of planning algorithms and agent reasoning. *The Knowledge Engineering Review* 30, 1 (2015), 1–44. https://doi.org/10.1017/S0269888913000337

[55] Felipe Meneguzzi and Michael Luck. 2008. Composing High-Level Plans for Declarative Agent Programming. In *Declarative Agent Languages and Technologies V*, Matteo Baldoni, Tran Cao Son, M. Birna van Riemsdijk, and Michael Winikoff (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 69–85.

[56] Felipe Meneguzzi and Michael Luck. 2009. Leveraging New Plans in AgentSpeak(PL). In *Declarative Agent Languages and Technologies VI*, Matteo Baldoni, Tran Cao Son, M. Birna van Riemsdijk, and Michael Winikoff (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 111–127.

[57] Dejan Mitrović, Mirjana Ivanović, Zoran Budimac, and Milan Vidaković. 2014. Radigost: Interoperable web-based multi-agent platform. *Journal of Systems and Software* 90 (2014), 167 – 178. https://doi.org/10.1016/j.jss.2013.12.029

[58] László Monostori, Botond Kádár, T Bauernhansl, S Kondoh, S Kumara, G Reinhart, O Sauer, G Schuh, W Sihn, and K Ueda. 2016. Cyber-physical systems in manufacturing. *CIRP Annals* 65, 2 (2016), 621 – 641. https://doi.org/10.1016/j.cirp.2016.06.005

[59] László Monostori, József Váncza, and Soundar RT Kumara. 2006. Agent-Based Systems for Manufacturing. *CIRP Annals* 55, 2 (2006), 697 – 720. https://doi.org/10.1016/j.cirp.2006.10.004

[60] Jörg P. Müller and Klaus Fischer. 2014. Application Impact of Multi-agent Systems and Technologies: A Survey. In *Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks*, Onn Shehory and Arnon Sturm (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 27–53. https://doi.org/10.1007/978-3-642-54432-3_3

[61] Dana S. Nau, Malik Ghallab, and Paolo Traverso. 2015. Blended Planning and Acting: Preliminary Approach, Research Challenges. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15)*. AAAI Press, 4047–4051. http://dl.acm.org/citation.cfm?id=2888116.2888281

[62] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. 2008. Restful Web Services vs. "Big"' Web Services: Making the Right Architectural Decision. In *Proceedings of the 17th International Conference on World Wide Web (WWW '08)*. ACM, New York, NY, USA, 805–814. https://doi.org/10.1145/1367497.1367606

[63] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. 2009. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, Vol. 3.

[64] Anand S. Rao. 1996. AgentSpeak(L): BDI agents speak out in a logical computable language. In *Agents Breaking Away*, Walter Van de Velde and John W. Perram (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 42–55.

[65] Alessandro Ricci, Enrico Denti, and Michele Piunti. 2010. A Platform for Developing SOA/WS Applications As Open and Heterogeneous Multi-agent Systems. *Multiagent Grid Syst.* 6, 2 (April 2010), 105–132. http://dl.acm.org/citation.cfm?id=1835414.1835418

[66] Alessandro Ricci, Michele Piunti, and Mirko Viroli. 2011. Environment programming in multi-agent systems: an artifact-based perspective. *Autonomous Agents and Multi-Agent Systems* 23, 2 (01 Sep 2011), 158–192. https://doi.org/10.1007/s10458-010-9140-7

[67] M. L. Roloff, M. R. Stemmer, J. F. Hübner, R. Schmitt, T. Pfeifer, and G. Hüttemann. 2014. A multi-agent system for the production control of printed circuit boards using JaCaMo and Prometheus AEOlus. In *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*. 236–241. https://doi.org/10.1109/INDIN.2014.6945514

[68] Z. Shelby, K. Hartke, and C. Bormann. 2014. The Constrained Application Protocol (CoAP). RFC 7252 (Proposed Standard). (June 2014). http://www.ietf.org/rfc/rfc7252.txt

[69] Z. Shelby, P. Mahonen, J. Riihijarvi, O. Raivio, and P. Huuskonen. 2003. NanoIP: the zen of embedded networking. In *Communications, 2003. ICC '03. IEEE International Conference on*, Vol. 2. 1218–1222 vol.2. https://doi.org/10.1109/ICC.2003.1204570

[70] Weiming Shen, Qi Hao, Hyun Joong Yoon, and Douglas H Norrie. 2006. Applications of agent-based systems in intelligent manufacturing: An updated review. *Advanced engineering INFORMATICS* 20, 4 (2006), 415–431.

[71] Weiming Shen, Lihui Wang, and Qi Hao. 2006. Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 36, 4 (July 2006), 563–577. https://doi.org/10.1109/TSMCC.2006.874022

[72] Yoav Shoham. 1993. Agent-oriented programming. *Artificial Intelligence* 60, 1 (1993), 51 – 92. https://doi.org/10.1016/0004-3702(93)90034-9

[73] Munindar P. Singh and Michael N. Huhns. 2006. *Service-Oriented Computing: Semantics, Processes, Agents*. John Wiley & Sons. https://doi.org/10.1002/0470091509

[74] Dante I. Tapia, Sara Rodríguez, Javier Bajo, and Juan M. Corchado. 2009. FUSION@, A SOA-Based Multi-agent Architecture. In *International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008)*, Juan M. Corchado, Sara Rodríguez, James Llinas, and José M. Molina (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 99–107.

[75] R. Verborgh and J. De Roo. 2015. Drawing Conclusions from Linked Data on the Web: The EYE Reasoner. *IEEE Software* 32, 3 (May 2015), 23–27. https://doi.org/10.1109/MS.2015.63

[76] Ruben Verborgh, Thomas Steiner, Davy Van Deursen, Sam Coppens, Joaquim Gabarró Vallés, and Rik Van de Walle. 2012. Functional Descriptions as the Bridge between Hypermedia APIs and the Semantic Web. In *Proceedings of the Third International Workshop on RESTful Design*. ACM, 33–40. https://doi.org/10.1145/2307819.2307828

[77] Danny Weyns and Fabien Michel. 2015. Agent Environments for Multi-agent Systems – A Research Roadmap. In *Agent Environments for Multi-Agent Systems IV*, Danny Weyns and Fabien Michel (Eds.). Springer International Publishing, Cham, 3–21.

[78] Danny Weyns, Andrea Omicini, and James Odell. 2007. Environment as a first class abstraction in multiagent systems. *Autonomous Agents and Multi-Agent Systems* 14, 1 (01 Feb 2007), 5–30. https://doi.org/10.1007/s10458-006-0012-0

[79] Dazhong Wu, Matthew John Greer, David W. Rosen, and Dirk Schaefer. 2013. Cloud manufacturing: Strategic vision and state-of-the-art. *Journal of Manufacturing Systems* 32, 4 (2013), 564 – 579. https://doi.org/10.1016/j.jmsy.2013.04.008

[80] Dogan Yazar and Adam Dunkels. 2009. Efficient Application Integration in IP-based Sensor Networks. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings (BuildSys '09)*. ACM, New York, NY, USA, 43–48. https://doi.org/10.1145/1810279.1810289

[81] Alexandra-Madalina Zarafin, Antoine Zimmermann, and Olivier Boissier. 2012. Integrating Semantic Web Technologies and Multi-Agent Systems: A Semantic Description of Multi-Agent Organizations. In *Proceedings of the First International Conference on Agreement Technologies (CEUR WS)*, Vol. 918. 296–297. http://ceur-ws.org/Vol-918/111110296.pdf