Agent-based Web Supported Simulation of Human-robot Collaboration

André Antakli, Torsten Spieldenner, Dmitri Rubinstein, Daniel Spieldenner,
Erik Herrmann, Janis Sprenger and Ingo Zinnikus
German Research Centre for Artificial Intelligence (DFKI), Campus D3 2, 66123 Saarbruecken, Germany

Keywords: Human Robot Collaboration, Multi-Agent Systems, Linked Data, 3D Simulation.

Abstract: In the production industry, in recent years more and more hybrid teams of workers and robots are being used to improve flexible processes. The production environment of the future will include hybrid teams in which workers cooperate more tightly together with robots and virtual agents. The virtual validation of such teams will require simulation environments in which various safety and productivity issues can be evaluated. In this paper, we present a framework for 3D simulation of hybrid teams in production scenarios based on an agent framework that can be used to evaluate critical properties of the planned production environment and the dynamic assignment of tasks to team members. The framework is embedded in a web-based distributed infrastructure that models and provides the involved components (digital human models, robots, visualization environment) as resources. We illustrate the approach with a use case in which a human-robot team works together in an aircraft manufacturing scenario.

1 INTRODUCTION

Human-robot collaboration (HRC) is expected to increase the automation level in the manufacturing industry which has been putting considerable research and development work into the implementation of hybrid teams for several years. A usual procedure in the planning and design phase when preparing the rollout is to simulate critical aspects of the collaboration processes in advance. For HRC, configurations of hybrid teams and their internal work distribution and interactions have to be validated in advance in order to reduce costs and time in real production and to avoid injuries as far as possible. In addition to the specific assembly planning, the establishment of hybrid teams asks for optimal orchestration of individual actors with fundamentally different characteristics. For these reasons, a highly configurable simulation environment to model and validate such configurations is needed.

3D simulations can be used to evaluate safety conditions, provide risk assessment, show spatial relationships and path regularities, convey time-dependencies, but they are also intuitive for the end-user, e.g. manufacturing planners, in terms of understanding and interaction (Mourtzis et al., 2015). Currently tools for planning, simulation and validation of complex production processes and flows of materials are available, but these only allow modeling plain action sequences of actors. When actors in a setting have various options to react to changing circumstances (which humans are capable of), each possible behavior has to be manually specified in detail. This becomes even more important with the increasing variety of products ('batch size 1'), leading to an enormous number of potentially diverging action sequences. Furthermore, these solutions look at each actor separately, which means they do not interact directly with each other, making a simulation of collaborative behavior very hard or even virtually impossible. In contrast, simulated production processes involving hybrid teams should be highly configurable and the simulated actors have to be—to a certain degree—autonomous and must interact directly with each other. Simulating dynamic and adaptive behavior of individual team members requires a paradigm that allows to intuitively model autonomous entities that can independently decide on how to achieve a given goal depending on the simulation state, exploring the space of possible (inter-)action sequences. If autonomous entities are to be simulated, agent systems are usually used to model and execute their behavior and interactions. If independent actors are not only to be visualized abstractly—e.g. with premodeled animations to evaluate ergonomic and spatial feasibility—approaches are needed that dynamically synthesize diversified motions that are as close to reality as possible, especially for the simulation of

Antakli, A., Spieldenner, T., Rubinstein, D., Spieldenner, D., Herrmann, E., Sprenger, J. and Zinnikus, I.
Agent-based Web Supported Simulation of Human-robot Collaboration.
DOI: 10.5220/0008163000880099
In *Proceedings of the 15th International Conference on Web Information Systems and Technologies (WEBIST 2019)*, pages 88-99
ISBN: 978-989-758-386-5

human actors.

When answering questions regarding the feasibility of a configuration, there must be guarantees about the adequacy of the simulated robot behavior, and, as far as possible, also for the simulated human. Hence, there is a requirement to integrate robot control and simulation software in order to make sure that the simulated processes represent the 'real' behavior of the simulated entities adequately. Most industrial robots are endowed with their own specific robotic software resp. their own data format, which means that to enable data exchange between several simulation subsystems, data needs to be lifted to a common representation. We propose to use a Web-based infrastructure where components are represented as resources. The resource-oriented infrastructure can be utilized for integrating simulation subsystems, but also during execution on the shop floor for information exchange between production control, robots and sensor/actuator systems.

In the following we present our approach to a highly configurable simulation environment that should support end-users, such as manufacturing planners, to optimally prepare, evaluate and improve collaboration of hybrid teams in production lines. Beside of a visualization environment, in which the end-user can flexibly manipulate the scenery objects at run-time and validate time constraints, we use approaches to dynamically generate human motions based on motion capture data and to enable intuitive modeling of agent behavior for orchestration and control of simulated hybrid teams in such a dynamic environment. The integration of the components is based on Web technology and standards which provide an abstraction layer for the communication and data exchange between the subsystems.

This paper is structured as follows. After pointing out the Related Work in Section 2, Section 3 shows our framework and the interplay of the different approaches. In section 4 we introduce the AJAN agent service and our motion synthesis system. In section 5, a use-case scenario in the context of hybrid teams in manufacturing is presented. Finally, we discuss open issues in section 6 and conclude in section 7.

2 RELATED WORK

In a recent survey on HRC, (Villani et al., 2018) distinguish basic *safety measures*, human-robot *coexistence* and human-robot *collaboration*. In coexistence, humans and robots share a common workspace, but have possibly independent goals. In collaboration, at least parts of the actions of the individual team members are coordinated towards reaching a shared goal, such as e.g. lifting and mounting heavy parts together in an assembly line. Fundamental for both, coexistence and collaboration, are in fact safety mechanisms which guarantee that collisions are avoided or as harmless as possible.

Several approaches focus on risk assessment in HRC scenarios based on built-in safety measures to reduce hazards. (Pedrocchi et al., 2013) and (Gopinath and Johansen, 2016) consider HRC mainly from the robot-oriented perspective of collision avoidance. (Awad et al., 2017) presents a design method for facilitating and automating risk assessment in HRC scenarios. Although risk assessment is an important aspect of HRC, especially in industrial contexts, it is rather a prerequisite for designing adaptable HRC workspaces. Given safety measures on the robotic side, the question still remains which impact these measures have on the coordination and collaboration of workers and robots in flexible environments.

Several commercial systems allowing the configuration and 3D simulation of target processes in production environments in the shop-floor context, e.g. Tecnomatix[1], FlexSim[2], visTABLEtouch[3] or SIMUL8[4]. DELMIA[5] for example, is a tool which allows additionally the validation of 'produced' products and the evaluation of manufacturing processes. DELMIA is also used in (Kashevnik et al., 2016) for prototyping CPPS environments. (Ore et al., 2014) present an extension of the IMMA tool for ergonomic assessment of HRC scenarios. (Fritzsche et al., 2014) describes EMA, a tool for ergonomic assessment of worker behavior which has recently been extended with robotic capabilities to simulate collaborations. The main deficit of these industrial frameworks is the limited support for specifying dynamical and mutual dependencies between the actors which is required for evaluating team-oriented behavior especially in critical situations (Tsarouchi et al., 2016).

3 SIMULATION FRAMEWORK ARCHITECTURE

In this section, we describe the overall architecture of the simulation framework (see Figure 1). It consists of components for modeling and executing the

[1]Tecnomatix: plm.automation.siemens.com/Tecnomatix
[2]FlexSim: www.FlexSim.com/FlexSim
[3]visTABLEtouch: www.vistable.de/visTABLEtouch-software
[4]SIMUL8: www.SIMUL8.com
[5]DELMIA: www.transcat-plm.com/software/ds-software/delmia

behavior of actors in simulated assembly scenarios (including workers, robots, etc.) using the multi-agent system AJAN (see Section 4.1); a component for generating worker motions using motion captured data, and the robot control component *Robot Operating System* (ROS) with an integrated simulation unit (Gazebo). The simulation of the actors is visualized in Unity3D[6], a widespread game engine. The 3D scene modeled with Unity3D also represents the real task environment. Accordingly, it makes the virtual working tools available to the actors. The integration of these subsystems is realized based on standard Web technologies, enhanced with semantic features, establishing a *resource-oriented architecture*.

3.1 Resource Oriented Architecture

Obviously, the nature of software shown in Figure 1, their data and provided network interfaces, differ widely, ranging from robotic operation systems, over highly dynamic data stores, to 3D game engines for interactive simulation and visualization. This variation calls for an integration layer to achieve *structural interoperability*, if not even *semantic interoperability* (Sheth, 1999) between the different components, meaning that applications are not only using structural compatible data layouts, but data can be mutually exchanged and understood between applications.

In particular in the domain of IoT, the W3C Web of Things Working Group[7] proposes to use the Web as convergence and integration platform. The so emerging *Web of Things (WoT)* (Guinard and Trifa, 2009) defines a Web-based abstraction layer for IoT platforms, protocols, data models and communication patterns. To unleash its full potential, the emerging WoT is expected to evolve into a Semantic Web of Things (SWoT) (Pfisterer et al., 2011). The SWoT will heavily rely on Linked Data principles (Heath and Bizer, 2011) to semantically describe IoT entities in terms of their actions, properties, events and metadata (Schubotz et al., 2017) independent of the underlying IoT platform.

This vision extends also to server-client communication. Verborgh et al. claim that for clients to act as *intelligent agents*, it must be given that client applications are able to explore and understand server functionality and data autonomously (Verborgh et al., 2011). Considering AJAN agents as intelligent clients that operate based on the provided application data, these requirements need also to be fulfilled by our framework's network API. In this respect, providing server data as HTTP resources that fulfill Fielding's *hypermedia constraints* (Fielding and Taylor, 2002), and with this, comply to a level 3 Richardson maturity model (Parastatidis et al., 2010), has been found a suitable way to match the requirements identified by Verborgh et al.

In the following, we outline how we achieve to lift the software components indicated in Figure 1 to a Linked Data representation to achieve a *Resource Oriented Architecture (ROA)* for the framework toolchain. The Linked Data layer of the resulting architecture ensures structural interoperability of the different tools' runtime data. By ensuring Level 3 Richardson Maturity Model compliance, we ensure more over that data can be autonomously explored and interpreted by AJAN agents.

3.1.1 Data Publishing

We employ *ECA2LD* (Spieldenner et al., 2018) to lift the stand-alone World server, as well as Unity3D, to RDF[8]. ECA2LD performs an automatic structural mapping from Entity-Component-Attribute (ECA) based runtimes, as it is the case for Unity3D and the standalone worldserver, to a Linked-Data representation in compliance with the Linked Data Platform W3C recommendation[9].

As result of the mapping, Entities (resp. *game objects* in Unity3D), attached Components, and selected Attributes that model relevant information, such as 3D position in space, sensor data, and others, are all represented by individual resources with an HTTP endpoint. Relevant information about the resources is provided by RDF triples that describe the structure and type of data. Relations between resources, such as Entity-Component relations, are established as links between resources. Being provided with an arbitrary resource as entry point, clients are by this able to autonomously explore the provided server data by following the established links. Moreover, the RDF description of provided data enables clients to identify subsets of resources of interest for their interaction via a SPARQL Query interface[10].

The robotic systems, driven by the ROS operation system, are accessed by a RESTful layer that complies to the *Linked Robotic Things* model, presented by Schubotz et al. (Schubotz et al., 2017). Comparable to the Linked Data Platform representation that is generated from ECA based runtimes by ECA2LD, the Linked Robotic Thing defines a Web model for robotic data that fulfills level 3 Richardson Maturity Model. In short, with the Linked Robotic

[6]Unity3D: https://unity3d.com
[7]https://www.w3.org/WoT/WG/
[8]https://www.w3.org/RDF/
[9]https://www.w3.org/TR/ldp/
[10]https://www.w3.org/TR/rdf-sparql-query/

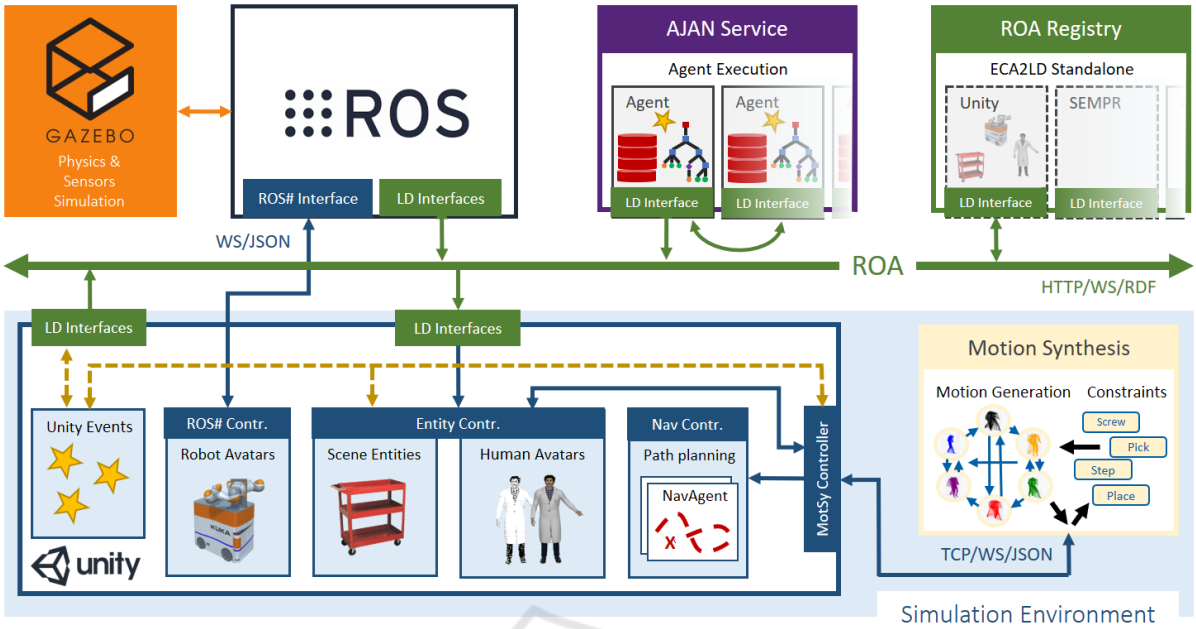

Figure 1: High level architecture of the collaborative robotic simulation framework. The different components from the heterogenous software pool are linked by a unified resource-oriented Linked Data layer.

```
<s> sub:endpoint [WebSocketURI]
  [WebSocketURI] sub:protocol sub:WebSocket
  [WebSocketURI] rdf:format <messageFormat>

<s> sub:endpoint [WebHookURI]
  [WebHookURI] sub:protocol sub:WebHook
  [WebHookURI] rdf:format <messageFormat>
```

Figure 2: The information about subscription channels provided in RDF by Linked Data resources.

Things model, we are able to semantically describe ROS robots in terms of components (such as sensors, joints) and actions, and provide associated LD-compliant APIs. Each of these concepts is provided with an individual HTTP resource that, in addition to the standard HTTP verbs, offers subscription mechanisms to constantly read and write data from and into the robotic application. This data may include, but is not limited to, streaming data from robotic sensors, reading or writing joint values, or sending commands to the robotic execution system.

Using the above mentioned methodologies for data publication on a common Web layer for both robotic and non-robotic systems, we achieve a unified, mutually understood data representation. Independent of the underlying application, clients are able to explore data by following links, also between applications. By this, we achieve full structural and data interoperability on the Web as integration layer.

3.1.2 Interfaces to Data Resources

To interact with data resources following HTTP operations are available: GET, to read out resource information; PUT, to update resource information; POST, to create a resource; and DELETE, to delete the resource. Information about further interaction possibilities with the resource can be obtained by using the HTTP OPTIONS operation. For example, the ECA2LD model offers an RDF description for a subscription mechanism that goes beyond the standard HTTP vocabulary (see Fig. 2). In addition to subscribing to resource changes via WebHook or WebSocket, an RDF-based data sheet describing the transferred data model is also offered.

3.2 Robot Control and Simulation

As described in Section 3.1.1 commands to and from the robotic execution system are sent with RESTful operations. In our implementation we use ROS, a popular open-source robotics middleware. ROS implements many robotic components and algorithms like *navigation stacks* (e.g. SLAM navigation) and *motion planning* (e.g. MoveIt). Furthermore, ROS supports robot simulation by integrating the Gazebo Robot Simulator (Koenig and Howard, 2004) which provides Gazebo services by exposing them as ROS services. In order to use the robot simulation we only need to configure ROS to use Gazebo instead of a real robot. This guarantees that the robot behavior in the

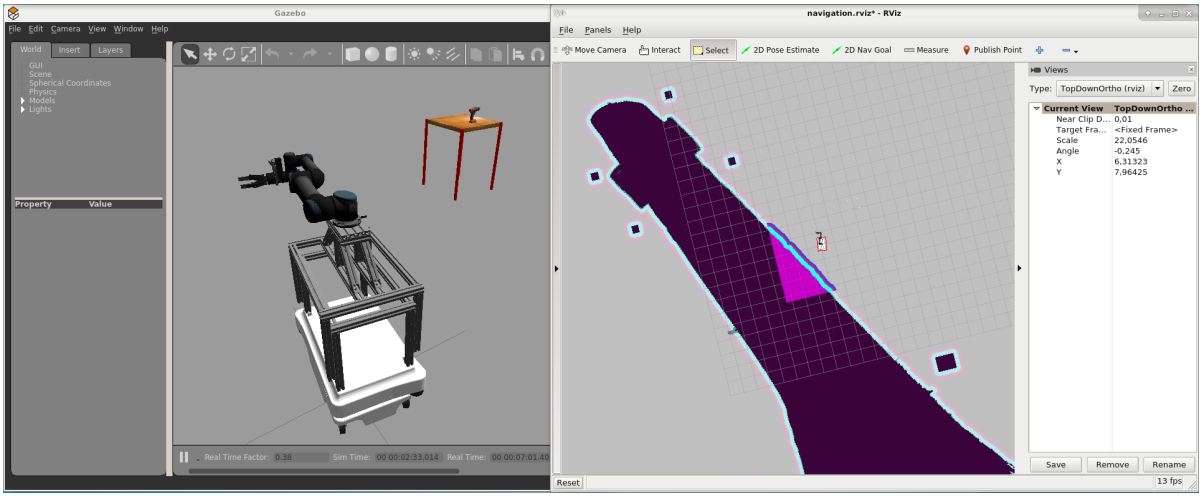

Figure 3: Running ROS (RVIZ visualization of navigation component on the right) and Gazebo (left) for the wing assembly simulation, to control a mobile one arm robot.

simulation is at least close to the 'real' robot behavior. For controlling the robot commands can be sent via the ROA to navigation planning, movement planning and picking components. When ROS is started with the Gazebo simulator, the commands are executed by a simulated robot. To visualize robots ROS#[11] is used to import URDF (Unified Robot Description Format) models into Unity3D.

3.2.1 Interfaces to Data Resources

```
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix hybrit: <https://hybr-it-projekt.de/ns/hybr-it#> .
@prefix ldp: <http://www.w3.org/ns/ldp#> .
@prefix maths: <http://vocab.arvida.de/2015/06/maths/vocab#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ros: <http://ros.org/#> .
@prefix rosf: <http://ros.org/rosfield#> .
@prefix spatial: <http://vocab.arvida.de/2015/06/spatial/vocab#> .
@prefix subscription: <https://hybr-it-projekt.de/ns/subscription#> .
@prefix vom: <http://vocab.arvida.de/2015/06/vom/vocab#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://localhost:9091/lrt/topics/simple_move_base/> a ros:Topic ;
    ros:type "hybrit_movement/SimpleMoveBase" ;
    hybrit:subscriptions <http://localhost:9091/lrt/topics/simple_move_base/subscriptions> ;
    hybrit:websocketUrl "http://localhost:9091/lrtws/topics/simple_move_base"^^xsd:anyURI .
```

```
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix hybrit: <https://hybr-it-projekt.de/ns/hybr-it#> .
@prefix ldp: <http://www.w3.org/ns/ldp#> .
@prefix maths: <http://vocab.arvida.de/2015/06/maths/vocab#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ros: <http://ros.org/#> .
@prefix rosf: <http://ros.org/rosfield#> .
@prefix spatial: <http://vocab.arvida.de/2015/06/spatial/vocab#> .
@prefix subscription: <https://hybr-it-projekt.de/ns/subscription#> .
@prefix vom: <http://vocab.arvida.de/2015/06/vom/vocab#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://localhost:9091/lrt/topics/simple_move_base_result/> a ros:Topic ;
    ros:type "hybrit_movement/SimpleMoveBaseResult" ;
    hybrit:subscriptions <http://localhost:9091/lrt/topics/simple_move_base_result/subscriptions> ;
    hybrit:websocketUrl "http://localhost:9091/lrtws/topics/simple_move_base_result"^^xsd:anyURI .
```

Figure 4: *Linked Robotic Things* model representation of a ROS robot move action goal topic *SimpleMoveBase* and result topic *SimpleMoveBaseResult*.

As a concrete example we describe here the API used to execute a move action of the ROS robot platform. ROS is built on top of a publish-subscribe system and ROS actions are executed by sending messages to specific topics and subscribing to messages on specific topics. The movement action offers two topics, a goal topic with a target pose and a result topic that informs us of the success or failure of an action. Accordingly to the *Linked Robotic Things* model both topics are accessible via a REST API and their RDF representation can bee seen in Figure 4. We extend the original concept with a *hybrit:websocketUrl* predicate in order to be able to specify WebSocket endpoint different to the endpoint of the resource. This can be necessary when the WebSocket endpoint is implemented by a technology other than HTTP endpoint.

3.3 Unity3D Simulation Environment

The frameworks as used according to Section 3.2 so far only cover robotic components and plan execution. We maintain and simulate remaining objects and entities which are not part of the robotic world in an interactive 3D run-time environment in the Unity3D[12] game engine. This allows for including avatars of human workers, as well as rigid bodies for tools and working material that is crucial for completion of the modelled task, but not directly linked to any robotic system. The simulated three-dimensional environment is visualized in an planning-editor and contains all objects to be displayed with their respective states. The editor interface provides means to the end-user to manipulate 3D objects and properties of the production units at run-time.

To generate realistic human animations in our simulation, we use a data-driven approach based on

[11]Open Source C# library for communicating with ROS: https://github.com/siemens/ros-sharp

[12]Unity: https://unity3d.com

motion capture data. The motion data is manually segmented into actions and automatically processed into a directed graph of parameterized motion models based on the approach presented by Min et al. (Min and Chai, 2012) and will be discussed in more detail in Section 4.2. The respective motion simulation is directly integrated into the Unity3D simulation environment.

We implemented a set of Unity3D scripts that use *ECA2LD* (cf. Section 3.1.1) to publish data of Unity3D game objects directly in terms of Linked Data Platform resources. Connected applications can by this retrieve simulation relevant data, i.e. states of simulated objects, via a Web based Linked Data integration layer that complies to the concepts described in (Spieldenner et al., 2018).

For the individual control of the simulated actors of a hybrid team, AJAN agents are used in our approach. AJAN is a multi-agent Web service developed for the intelligent orchestration of Linked Data[13] (LD) resources and was already used for various human simulations in 3D worlds, see (Antakli et al., 2018), (Zinnikus et al., 2017). To access simulated entities managed in unity3D, AJAN uses their LD abstraction layer provided by ECA2LD.

4 AGENT-BASED ORCHESTRATION

4.1 Distributed Control with AJAN

The multiagent system (MAS) paradigm has already proven that it can be used to realize advanced distributed applications in environments with a high diversity like IoT or LD domains, see (Bosse, 2016),(Xu et al., 2013),(Khriyenko and Nagy, 2011),(Diaconescu and Wagner, 2015),(Garcia-Sanchez et al., 2008). Individual agents of a MAS are autonomous, interconnected and to a certain extent intelligent units, which perceive their environment and decide independently how to interact with it. The MAS paradigm is predestined to implement a higher value ”intelligent” functionality of semantically described heterogeneous domains on application level, while hiding the deployment context from the user. AJAN (Accessible Java Agent Nucleus) is an agent system designed to interact with LD domains. RDF/SPARQL enhanced Behavior Trees (BT) are used as an agent behavior model to dynamically explore such domains and to query and orchestrate LD resources.

4.1.1 AJAN Agent Model

An AJAN agent has one or more behaviors, each executed in a single thread and consisting of a SPARQL-BT (see Section 4.1.2) and a corresponding behavior RDF database; one agent specific knowledge base (KB), storing inter-behavior knowledge like the agent status; one or more events, each holding RDF data in the form of named graphs for behaviors; and one or more HTTP endpoints. These endpoints are the agent's interfaces to its LD-domain and forward incoming RDF messages as named graphs in form of events. Behaviors can be linked to these events. If an event occurs, the behaviors linked to it are executed. While executing a SPARQL-BT, it can access special incoming event data by querying its named graph. Each Behavior can also create events to trigger other behaviors. In addition, the agent state can be checked and manipulated during execution, as well as interacting with LD resources.

By using the AJAN plug-in system, AI methods can be integrated as behavior primitives for behavioral modeling. For example, a SPARQL-BT can be synthesized during GraphPlan-based (Meneguzzi et al., 2004) action planning, or by using a SPIN-rule engine [14], the agent KB can be extended.

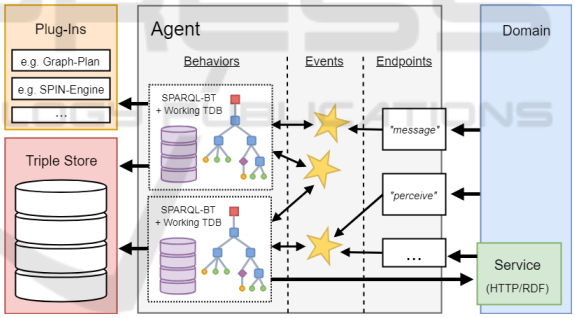

Figure 5: AJAN-Agent model overview.

4.1.2 AJAN Behavior Model

For modeling agent behavior in LD domains, AJAN uses an extension of the Behavior Trees (BT) paradigm widely used in industry and robotics (see (Marzinotto et al., 2014), (Paxton et al., 2017), (Nguyen et al., 2013)), the SPARQL-BT approach. SPARQL-BTs, as one might expect, are a combination of the BT paradigm with SPARQL, which is first mentioned in (Schreiber et al., 2017). Basically, BTs are used in AJAN to perform contextual SPARQL queries for state checking, updating, or constructing RDF data used for action executions. Furthermore, SPARQL-BTs are defined in RDF,

[13]Linked Data: https://www.w3.org/standards/semanticweb/data

[14]Using the engine integrated in rdf4j: http://rdf4j.org

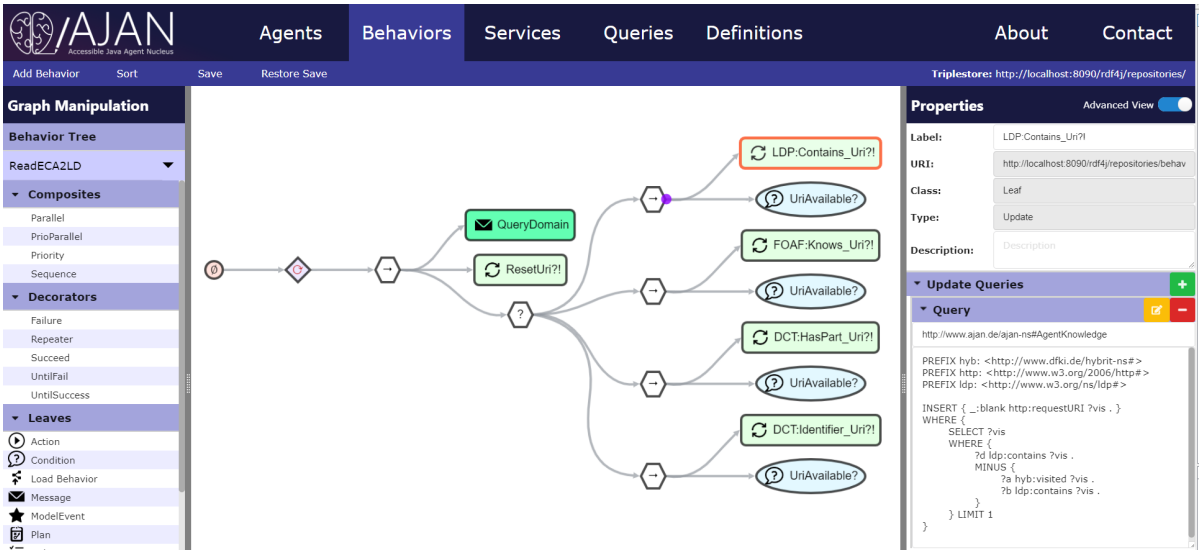

Figure 6: SPARQL-BT displayed in the AJAN Editor to examine an ECA2LD domain. The GET node (turquoise) in the iterating BT first queries an previously selected ECA2LD resource (URI). The resulting RDF graph is then queried via a Selector node (labeled with '?') for different relations shown, from which an unvisited ECA2LD resource is selected for the next iteration.

whereby a semantic description of the behaviors they implement is available and to meet the requirements of the LD paradigm. SPARQL-BTs use standard BT composite and decorator nodes and are processed like typical BTs[15], but this approach defines three main new leaf node types to work on RDF-based datasets and resources using SPARQL queries. Thus, a SPARQL-BT always has one or more RDF Triple Stores that can be accessed via SPARQL endpoints that follow the W3C standardized SPARQL protocol (W3C, 2008).

SPARQL-BT Condition. *A SPARQL-BT Condition is a BT leaf node that makes a binary statement about the presence of a graph-pattern in an RDF dataset. It returns two states after execution: SUCCEEDED and FAILED and can be used to formulate state conditions of an agent. Thereby, it performs one SPARQL 1.1 ASK query on a defined RDF dataset. The dataset can be a default graph or a named graph and is represented by its SPARQL endpoint URI. To define a SPARQL ASK query, the complete language space of the SPARQL 1.1 language with regard to ASK operations in (W3C, 2013a) can be used.*

SPARQL-BT Update. *This leaf node returns two states after execution: SUCCEEDED and FAILED and can be used to create, delete or update RDF data in a Triple Store. Thereby, it performs one SPARQL 1.1 UPDATE query on a defined RDF dataset. The dataset can be a default graph or a named graph and is represented by its SPARQL endpoint URI. To define a SPARQL UPDATE query, the complete language space of the SPARQL 1.1 UPDATE language in (W3C, 2013b) can be used.*

SPARQL-BT Action. *A SPARQL-BT Action leaf node sends a RDF dataset via HTTP or WebSocket to an external LD-resource which is defined by a URI endpoint and an action description[16]. This dataset is defined using a SPARQL 1.1 CONSTRUCT query. The complete SPARQL 1.1 language space with regard to CONSTRUCT operations in (W3C, 2013a) can be used for this purpose. The RDF response resulting from the executed external resource is then inserted into a named graph of the agents knowledge base. In this context, named graphs define the source of the received result. A SPARQL-BT Action node returns three states as comparable action nodes of other BT implementations, SUCCEEDED, FAILED and RUNNING. Actions that do not immediately receive a result from executed LD resources, are so-called asynchronous actions.*

Beside of the SPARQL-BT nodes presented, further nodes were realized, e.g. to dynamically choose and execute AJAN behaviors or to interact with LD-

[15]Behavior Tree framework based on libGDX: https://github.com/libgdx/gdx-ai/wiki/Behavior-Trees

[16]The description of resource actions respectively affordances is oriented to the action language *A* defined in (Gelfond and Lifschitz, 1998).

resources like AJAN-agents, using HTTP methods like GET, POST, PUT, OPTIONS or DELETE. The ECA2LD approach presented in Section 3.1 provides distributed data semantically and refers to their raw form. In order to enable AJAN to interact with a ECA2LD domain, it was extended with further SPARQL-BT primitives to query data and to perform domain actions over WebSockets. Since the data addressed is not only stored in RDF, an RML based[17] mapping AJAN-plugin was established. With this plugin XML, JSON or CVS data can be translated into RDF to make it usable for SPARQL-BTs. In figure 6 a SPARQL-BT is presented with which an AJAN agent can autonomously explore an ECA2LD domain to broaden its event and action horizon. Given the URI of a ECA2LD entry point, it is possible to iteratively query it over given links in order to dynamically include the LD-resources found into the agent planning process.

4.2 Worker Simulation

Realistic worker simulation is a prerequisite for the evaluation of hybrid teams. Whereas many available tools provide manikins based on predefined behavior we synthesize digital human models from motion capture data and learned behavior using neural networks.

The worker motion synthesis functionality is implemented as an external service that controls the state of the human workers in the simulation by sending a continuous stream of poses to Unity3D. Depending on whether the web browser is the target platform of the Unity3D application, either a TCP or a WebSocket connection can be used by the server. The behavior defined by the agent system is translated into worker motions by providing a sequence of actions with a set of spatial constraints in a custom JSON format to the REST interface of the motion synthesis service. The spatial constraints can be automatically derived from the 3D environment based on annotated scene objects and standard path planning functionality of the game engine.

To produce realistic motions we use machine learning models trained on reference motion capture data. Depending on the type of action, the synthesis service can either apply a statistical model-based method (Min and Chai, 2012) or phase-functioned neural networks (Holden et al., 2017) to generate the motions. For manipulation actions with constraints on the hands, such as picking, fastening of screws or actions involving tools, we apply the statistical motion synthesis method in combination with inverse kinematics to produce motion clips. To accelerate the statistical motion synthesis for the real-time application, we prepare a search data structure for each motion model that enables a fast look up of a motion example given constraints before it is further optimized (Herrmann et al., 2017). For walking motions, however, we apply the phase-functioned neural network which produces a smooth sequence of poses of arbitrary length given a reference path.

The motion synthesis runs in a separate thread from the animation server that synchronizes the state of the worker motion with Unity3D. This way the motion for each agent can be generated ahead of time and stored in a pose queue. As soon as the pose queue is empty the motion synthesis service will notify AJAN that the action was completed and start looping an idle motion until the next task is specified. In case that the worker needs to react to a change in the environment the pose queue can be emptied earlier by specifying a new task before the current task is complete.

5 APPLICATION SCENARIO

Figure 7: Visualization of the wing assembly application scenario with one robot and two workers.

Figure 8: Visualization of the hybrid team interaction.

The collaboration of workers with a robot to assemble raceways in an commercial airplane wing (see Figure 7) demonstrate the challenges faced in a simulation environment for human robot interaction. Due to the overhead position of the parts to assemble, it is difficult for human workers to maintain an ergonomic working posture while handling heavy or unwieldy

[17]RML, a extension of the W3C-recommended R2RML mapping language: http://rml.io

parts. Having a fully simulated environment allows for checking for bottle necks or potential dangerous situations that might occur in a real life application. In our case, two human workers are simulated, performing the task of picking up material from shelves in the factory building, while a robot (steered by a *ROS* application, see Figure 3) takes care of providing them with tools at appropriate times. In the scenario considered here, a worker and a robot are simulated. The worker is responsible for placing the raceways, whereas a robot stands ready to aid the worker by bringing working material and pass it to the worker in an ergonomically convenient fashion.

5.1 Scenario Realization

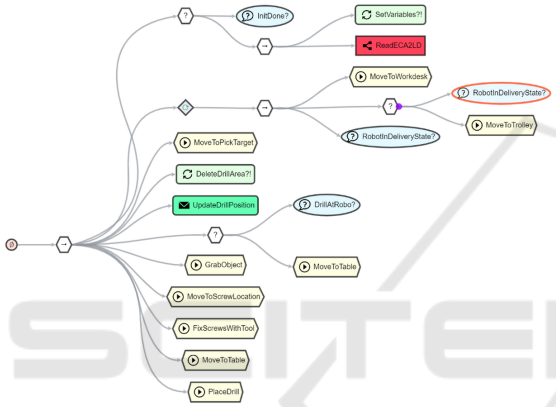

Figure 9: Visualization of the hybrid team interaction.

The scenario is visualized using Unity3D, accessing the Unity3D animation system and using out of the box web communication to integrate our systems. The workers are animated using synthesized motions provided by our *Motion Synthesis*. This also ensures that all the worker movements are reasonable and can be performed by a living human being in a real world scenario. The behavior of each worker is modelled using *SPARQL-BTs*, not only providing a blueprint for the task to be completed but also allowing to react to changes in the environment in real time. Every actor in the scene shares common knowledge synchronized via the ROA, and each change in the world state is considered there as well. For this purpose each actor applies the same SPARQL-BT shown in Figure 6 to explore the ROA. In Figure 9 a SPARQL-BT of a worker is displayed, where this BT is integrated (red leaf node). The AJAN-controlled worker runs this SPARQL-BT at the beginning to find simulated objects in the Unity3D scene, other AJAN agents, or the mentioned ROS robot, to get status updates but also to interact with them. The screwdriver position is queried in the turquoise node – if it is carried by robot (see Figure 8), the worker approaches the robot to take over the screwdriver.

The underlying information comes from various resources that the agent collects, and makes them available through its KB for behavioral execution. For example, the information that the screwdriver is located at the robot comes from the Unity environment. Its actual position (which is JSON-based and still needs to be transformed to RDF using the AJAN mapping plugin) is derived from the position of the robot arm actuator that holds the screwdriver, originating from the ROS robot itself. Both subscribed resource endpoints were referenced from the previously explored ECA2LD-based ROA.

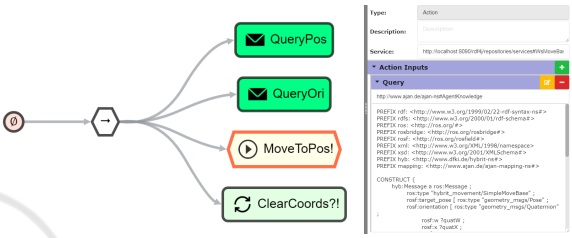

Figure 10: SPARQL query to create the input to execute a move action of the ROS robot.

The robot is controlled by an AJAN agent as well, sending instructions to a ROS environment and taking feedback from ROS into account to decide upon the success of a desired action. Figure 10 shows a SPARQL-BT that queries a predefined URI of a named transformation in the Unity3D environment. This transformation is then sent to the LD interface (see Figure 4) of the ROS robot navigation component using a SPARQL-BT action. For this purpose, the requested transformation is converted into the ROS coordinate system with the help of a construct query and sent to the robot as an RDF-based ROS message. The SPARQL query described can be seen in Figure 11, where the orientation (see lines 16-19) and the position of the mobile robotic platform (see lines 21-23) is set. To receive the result of the navigation action, i.e. whether the robot has found a path to the transferred target and then reached it, AJAN listens to the offered Result WebSocket Endpoint.

A possible transformation is, for example, a previously defined position in which the screwdriver can be picked by the robot to deliver it to the worker.

5.2 Application Examples

Given a specified team behavior with roles and individual tasks assigned, several aspects of the team performance can be simulated and evaluated. Evaluation criteria are e.g. time to fulfil a specific task sequence,

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ros: <http://ros.org/#>
PREFIX rosbridge: <http://ros.org/rosbridge#>
PREFIX rosf: <http://ros.org/rosfield#>
PREFIX xml: <http://www.w3.org/XML/1998/namespace>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX hyb: <http://www.dfki.de/hybrit-ns#>
PREFIX mapping: <http://www.aian.de/aian-mapping-ns#>

CONSTRUCT {
    hyb:Message a ros:Message ;
        ros:type "hybrit_movement/SimpleMoveBase" ;
        rosf:target_pose [ ros:type "geometry_msgs/Pose" ;
        rosf:orientation [ ros:type "geometry_msgs/Quaternion" ;
            rosf:w ?quatW ;
            rosf:x ?quatX ;
            rosf:y ?quatZ ;
            rosf:z ?invQuaY ] ;
        rosf:position [ ros:type "geometry_msgs/Point" ;
            rosf:x ?posZ ;
            rosf:y ?invPosX ;
            rosf:z ?posY ] ];
        rosf:frame_id "map" ;
        rosf:action_name "move_base" ;
        rosf:id "1234" .
}
WHERE {
    ?robot a hyb:Robot .
    ?robot mapping:position ?position .
    ?position a mapping:Position .
    ?position mapping:posX ?posX .
    ?position mapping:posY ?posY .
    ?position mapping:posZ ?posZ .
    ?robot mapping:orientation ?orientation .
    ?orientation mapping:quatW ?quatW .
    ?orientation mapping:quatX ?quatX .
    ?orientation mapping:quatY ?quatY .
    ?orientation mapping:quatZ ?quatZ .
    BIND(?posX * (-1.0) AS ?invPosX) .
    BIND(?quatY * (-1.0) AS ?invQuaY) .
}
```

Figure 11: SPARQL query to create the input to execute a move action of the ROS robot platform.

reachability of positions and objects, or ergonomic assessment of poses based on visual appearance with respect to different body sizes. In the application scenario presented above, a number of observations and experiences have been made:

- in certain configurations and job assignments, the robot is often slowed down since trajectories of workers and robot interfere. A deliberate change of the assignments helps to reduce certain interferences;
- the robot can grasp a tool from a specified position but the ensuing grasp pose leads to a frequent slip of the tool in the gripper. To cope with this a worker has to correct the grip or take over the tool immediately;
- depending on a position assigned or calculated, the robot is supposed to grasp a tool from a commissioning area which cannot be reached. Changing the position increases the success rate;
- in some cases, the same issue occurred for the human worker. Here, ergonomic aspects are assessment based on visual appearance;
- the robot might damage the wing during navigation if robot arm is not properly in driving position. A safe area has been introduced which the robot is not allowed to enter;
- we discovered that certain task assignments could be improved based on the information in the ROA, e.g. it is better to assign a task dynamically to a worker when a robot is too far away from a commissioning table;
- if available, sensor information can be fed into the ROA to evaluate the impact on the team behavior;
- finally, it is possible to take into account the context: is the configuration of team tasks able to deal with e.g. a change in the clock rate or problems due to missing supplies.

6 DISCUSSION

Programming a robot even for simulation purposes is a complex endeavor. For the connection of a ROS-controlled robot to the ROA, a corresponding control software must nevertheless be available. In order to obtain adequate robot behavior, the relevant sensor information must be generated, to which the controller can react. In some cases this can be generated relatively easily in the Unity3D environment, for other sensor systems this task is very complex and time consuming.

The ROA is designed to abstract from the underlying technology. As reported, we use a robot controller based on ROS. For other commercial robot systems with their proprietary software a connection to the ROA is future work.

When simulating human behavior to evaluate hybrid teams, the unpredictability of human behavior is a problem. Several intended human responses may be uncritical, but in some situations a safety-critical moment may result. One possible way to take this into account is to randomize human behavior and generate a variety of actions to test effects on robot behavior.

7 CONCLUSION

We presented a framework for the 3D simulation of hybrid teams in production scenarios. Using an agent framework for modeling dynamic behavior and motion synthesis for the animation of human worker behavior we developed a platform that enables end-users (e.g. manufacturing planners) to specify and coordinate team-based production processes where critical situations can be created and evaluated. By clearly specified Linked Data representations and developed tools to lift applications to this representation, the presented architecture is highly extendable and flexible. It allows incorporating external services as e.g. a ROS-based robot control software to ensure an appropriate simulation of the behavior of the robots in-

volved. Based on the usage of RDF and the 3-RMM compliance, the agent system AJAN is able to autonomously understand new software components in the architecture. Adapting the architecture to new domain-specific applications is then reduced to modelling fitting SPARQL-BTs that operate on the new data. Our approach has been applied to an aerospace industry use case in an air plane assembly line.

ACKNOWLEDGEMENTS

The work described in this paper has been partially funded by the German Federal Ministry of Education and Research (BMBF) through the projects Hybr-iT under the grant 01IS16026A, and REACT under the grant 01/W17003.

REFERENCES

Antakli, A., Hermann, E., Zinnikus, I., Du, H., and Fischer, K. (2018). Intelligent distributed human motion simulation in human-robot collaboration environments. In *Proceedings of the 18th International Conference on Intelligent Virtual Agents*, IVA '18, pages 319–324, New York, NY, USA. ACM.

Awad, R., Fechter, M., and van Heerden, J. (2017). Integrated risk assessment and safety consideration during design of HRC workplaces. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–10.

Bosse, S. (2016). Mobile Multi-agent Systems for the Internet-of-Things and Clouds Using the JavaScript Agent Machine Platform and Machine Learning as a Service. In *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 244–253.

Diaconescu, I. M. and Wagner, G. (2015). Modeling and Simulation of Web-of-Things Systems as Multi-Agent Systems. In Mueller, J. P., Ketter, W., Kaminka, G., Wagner, G., and Bulling, N., editors, *Multiagent System Technologies*, volume 9433, pages 137–153. Springer International Publishing, Cham.

Fielding, R. T. and Taylor, R. N. (2002). Principled design of the modern web architecture. *ACM Transactions on Internet Technology (TOIT)*, 2(2):115–150.

Fritzsche, L., Schönherr, R., and Illmann, B. (2014). Interactive simulation and ergonomics assessment of manual work with EMA–applications in product development and production planning. In *Advances in Applied Digital Human Modeling. AHFE*, pages 49–58.

Garcia-Sanchez, F., Fernández-Breis, J. T., Valencia-Garcia, R., Gómez, J. M., and Martinez-Béjar, R. (2008). Combining Semantic Web technologies with Multi-Agent Systems for integrated access to biological resources. *Journal of Biomedical Informatics*, 41(5):848–859.

Gelfond, M. and Lifschitz, V. (1998). Action languages. *Electronic Transactions on AI*, 3.

Gopinath, V. and Johansen, K. (2016). Risk assessment process for collaborative assembly–a job safety analysis approach. *Procedia CIRP*, 44:199–203.

Guinard, D. and Trifa, V. (2009). Towards the web of things: Web mashups for embedded devices. In *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences), Madrid, Spain*, volume 15.

Heath, T. and Bizer, C. (2011). Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology*, 1(1):1–136.

Herrmann, E., Manns, M., Du, H., Hosseini, S., and Fischer, K. (2017). Accelerating statistical human motion synthesis using space partitioning data structures. *Computer Animation and Virtual Worlds*, 28(3-4):e1780.

Holden, D., Komura, T., and Saito, J. (2017). Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)*, 36(4):42.

Kashevnik, A., Teslya, N., Yablochnikov, E., Arckhipov, V., and Kipriianov, K. (2016). Development of a prototype cyber physical production system with help of smart-m3. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 4890–4895.

Khriyenko, O. and Nagy, M. (2011). Semantic web-driven agent-based ecosystem for linked data and services. In *Proceedings of the Third International Conferences on Advanced Service Computing*, pages 25–30.

Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154.

Marzinotto, A., Colledanchise, M., Smith, C., and Ögren, P. (2014). Towards a unified behavior trees framework for robot control. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5420–5427.

Meneguzzi, F. R., Zorzo, A. F., and da Costa Móra, M. (2004). Propositional planning in BDI agents. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, SAC '04, pages 58–63, New York, NY, USA. ACM.

Min, J. and Chai, J. (2012). Motion graphs++: a compact generative model for semantic motion analysis and synthesis. *ACM Transactions on Graphics (TOG)*, 31(6):153.

Mourtzis, D., Papakostas, N., Mavrikios, D., Makris, S., and Alexopoulos, K. (2015). The role of simulation in digital manufacturing: applications and outlook. *International Journal of Computer Integrated Manufacturing*, 28(1):3–24.

Nguyen, H., Ciocarlie, M. T., Hsiao, K., and Kemp, C. C. (2013). ROS commander (ROSCo): Behavior creation for home robots. *2013 IEEE International Conference on Robotics and Automation*, pages 467–474.

Ore, F., Hanson, L., Delfs, N., and Wiktorsson, M. (2014). Virtual evaluation of industrial human-robot cooperation: An automotive case study. In *3rd International Digital Human Modeling Symposium (DHM2014), May 20-22, Odaiba, Japan*.

Parastatidis, S., Webber, J., Silveira, G., and Robinson, I. S. (2010). The role of hypermedia in distributed system development. In *Proceedings of the First International Workshop on RESTful Design*, pages 16–22. ACM.

Paxton, C., Hundt, A., Jonathan, F., Guerin, K., and Hager, G. D. (2017). CoSTAR: Instructing collaborative robots with behavior trees and vision. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 564–571. IEEE.

Pedrocchi, N., Vicentini, F., Matteo, M., and Tosatti, L. M. (2013). Safe human-robot cooperation in an industrial environment. *International Journal of Advanced Robotic Systems*, 10(1):27.

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., and Richardson, R. (2011). Spitfire: toward a semantic web of things. *IEEE Communications Magazine*, 49(11):40–48.

Schreiber, W., Zürl, K., and Zimmermann, P., editors (2017). *Web-basierte Anwendungen Virtueller Techniken: Das ARVIDA-Projekt – Dienste-basierte Software-Architektur und Anwendungsszenarien für die Industrie*. Springer Vieweg.

Schubotz, R., Vogelgesang, C., Antakli, A., Rubinstein, D., and Spieldenner, T. (2017). Requirements and Specifications for Robots, Linked Data and all the REST. In *Proceedings of Workshop on Linked Data in Robotics and Industry 4.0. Workshop on Linked Data in Robotics and Industry 4.0 (LIDARI-2017)*. CEUR.

Sheth, A. P. (1999). Changing focus on interoperability in information systems: from system, syntax, structure to semantics. In *Interoperating geographic information systems*, pages 5–29. Springer.

Spieldenner, T., Schubotz, R., and Guldner, M. (2018). Eca2ld: From entity-component-attribute runtimes to linked data applications. In *Proceedings of the International Workshop on Semantic Web of Things for Industry 4.0. Extended Semantic Web Conference (ESWC-2018), International Workshop on Semantic Web of Things for Industry 4.0, located at 15th ESWC Conference 2018, June 3-7, Heraklion,, Crete, Greece*. Springer.

Tsarouchi, P., Makris, S., and Chryssolouris, G. (2016). Human—robot interaction review and challenges on task planning and programming. *International Journal of Computer Integrated Manufacturing*, 29(8):916–931.

Verborgh, R., Steiner, T., Van Deursen, D., Van de Walle, R., and Vallés, J. G. (2011). Efficient runtime service discovery and consumption with hyperlinked restdesc. In *2011 7th International Conference on Next Generation Web Services Practices*, pages 373–379. IEEE.

Villani, V., Pini, F., Leali, F., and Secchi, C. (2018). Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, 55:248–266.

W3C (2008). SPARQL 1.1 Protocol. https://www.w3.org/TR/sparql11-protocol/. [Online; accessed 19-May-2019].

W3C (2013a). SPARQL 1.1 Query Language. https://www.w3.org/TR/2013/REC-sparql11-query-20130321/. [Online; accessed 19-May-2019].

W3C (2013b). SPARQL 1.1 Update. https://www.w3.org/TR/sparql11-update/. [Online; accessed 19-May-2019].

Xu, X., Bessis, N., and Cao, J. (2013). An Autonomic Agent Trust Model for IoT systems. *Procedia Computer Science*, 21:107–113.

Zinnikus, I., Antakli, A., Kapahnke, P., Klusch, M., Krauss, C., Nonnengart, A., and Slusallek, P. (2017). Integrated semantic fault analysis and worker support for cyber-physical production systems. In *2017 IEEE 19th Conference on Business Informatics (CBI)*, volume 1, pages 207–216. IEEE.