Requirements and Specifications for Robots, Linked Data and all the REST

René Schubotz, Christian Vogelgesang, André Antakli, Dmitri Rubinstein, Torsten Spieldenner
German Research Center for Artificial Intelligence
Saarbrücken, Germany

ABSTRACT

The use of autonomous robots in both industry and every day life has increased significantly in the recent years. A growing number of robots is connected to and operated from networks, including the World Wide Web. Consequently, the Robotics community is exploring and adopting REST (Representational State Transfer) architectural principles and considers the use of Linked Data technologies as fruitful next step. However, we observe a lack of concise and stable specifications of how to properly leverage the RESTful paradigm and Linked Data concepts in the Robotics domain. Introducing the notion of *Linked Robotic Things*, we provide a minimalistic, yet well-defined specification covering a minimal set of requirements with respect to the use of HTTP and RDF.

1 INTRODUCTION

Robotic systems are widely used in modern society and have significant impact to economics and social aspects. However, most of these robots are not able to act in a context aware manner. For example, many industry robots execute hard coded programs in a caged working space. With the increasing demand for flexibility in production processes and enhanced human-robot collaboration, robotic systems have to interact naturally with their environment, other robots and humans. To achieve this, robotic systems must acquire deeper knowledge about their environment, for example by tapping into other software and information systems.

In this respect, Kamei et al. [17] argue that for the successful design of flexible, extendable, re-usable applications, robots are required to be provided as abstracted resource in a "cloud of robots". Subsequently, parts of the robots, the actions that can be performed, and tasks that are to be fulfilled by performing the described actions need to be provided in a unified format.

Consequently, the Robotics community is exploring and adopting REST (Representational State Transfer) architectural principles and considers the use of Linked Data technologies as fruitful next step. However, we observe a lack of concise and stable specifications of how to properly leverage the RESTful paradigm and Linked Data concepts in the Robotics domain.

In the scope of this paper, we identify a set of desiderata that we consider vital for successfully implementing Linked Data-driven robotic systems and applications. Introducing the notion of *Linked Robotic Things*, we provide a minimalistic, yet well-defined specification covering a minimal set of requirements with respect to the desiderata identified. Our contributions include in particular:

(1) the *Robotic Thing Model*, a minimalistic core specification for truly RESTful "Web Robotics".
(2) the notion of *Linked Robotic Things* as a semantic extension of Web-enabled robots.
(3) the *Linked Robotic Thing Model*, a basic contract allowing clients to automatically discover and interoperate with Linked Robotic Things.

The remainder of this paper is organized as follows. We outline the development of "Web Robotics" and report on recent research in the Linked Data community (cf. Section 2). Next, several high-level desiderata for Linked Data-driven robotic systems and applications (cf. Section 3) are discussed. Section 4 presents a minimal specification of Web-enabled robots meeting the criteria for level 3 of the Richardson Maturity Model. We propose a semantic extension of Web-enabled robots in Section 5, and define the *Linked Robotic Thing Model* in Section 6. Future work is outlined in Section 7 , and we conclude with summary in Section 8.

2 FROM TELELABS TO WEB ROBOTICS

Active since 1994, the UWA Telelabs Project offers remote access to an industrial robot for tele-manipulations in a blocks world and initiated the subfield of "Internet Robotics" [40]. Evolving from internet-based interfaces to robotic systems [7, 16, 29], to networked control middlewares [1, 5, 46], service-oriented approaches using various instantiations of the web service protocol stack [4, 6, 22, 23, 28] and finally to RESTful services for robotic applications [23, 34, 42, 45], researchers in the field consider the use of the World Wide Web and its associated technologies as a scalable robotics application platform [18, 36, 53, 55]. Rather than exposing real-world data and functionality through vertical system designs, proponents of "Web Robotics" suggest to apply the REST architectural style [11, 37] to Web-enabled robots in the physical world and thus to make them an integral part of the Web.

Following the very same line of argument, the Linked Data community explores the relationship between software and information systems and the Web. Initially focusing on aligning SPARQL with

REST architectural principles [54], interest quickly revolves around the notion of *Linked Services*. Coining the nomenclature, Pedrinaci et al. [33] anticipate RESTful services that process and generate Linked Data and concomitantly expose their "inputs and outputs, their functionality, and their non-functional properties" following Linked Data principles. After discussion and refutation [30] of the potential impedance mismatch between RESTful services and the Resource Description Framework (RDF), considerable effort has been dedicated to the formal description of Linked Services (cf. [48]).

However, facing the lack of any standardization or widespread adoption of Linked Services, a fatum shared with Semantic Web Services [50], the scientific community currently favours addressing more utile techniques for data integration and system interoperation. To name a few, we briefly list RDF constraint languages for describing and constraining the contents of RDF graphs [21, 32, 35, 38, 44], novel RESTful Linked Data interfaces as well as client-side query execution techniques [47, 49, 51, 52], the specification of integration and interaction patterns for building resource-oriented Linked Data applications [13–15, 26, 39], or high-performance processing techniques for dynamic Linked Data [19, 20].

3 TOWARDS LINKED ROBOTICS

Recent developments in Linked Data and Hypermedia research might help in setting priorities for a roadmap towards Linked Data in Robotics. In the following, we discuss several high-level desiderata for Linked Data-driven robotic systems and applications that we feel free to coin *Linked Robotics*.

D1 **Level 3 Richardson Maturity Model:** The larger share of RESTful interface designs in industry and academia is focused on devising resource identifier schemes and discussing the semantics of HTTP verbs in context thereof. Such designs are oftentimes in violation with the "URI Opacity Axiom" [2] and usually neglect the principle of "Hypermedia As The Engine Of Application State" (HATEOAS) [31]. Hence they rarely meet the criteria for level 3 of the Richardson Maturity Model [12]. Since we aim to provide a core specification for "Linked Robotics", we must assume clients with no or limited prior knowledge and consequently establish 3-RMM[1].

D2 **Well-defined interaction patterns:** The cornerstones for successful distributed systems are well-defined protocol interfaces and interaction patterns. In this respect, some communities provide guidelines and specifications for their domain, e.g. the Richardson Maturity Model [12] details on the mandatory properties of RESTful systems, Berners-Lee [3] suggests a cumulative "5 Star" deployment scheme for Linked Data systems, Guinard et al. [13, 14] offer guidelines for the Web of Things, and Speicher et al. [39] define a set of rules for read-write Linked Data on the Web. We are not aware of equipollent work in the "Web Robotics" community.

D3 **Event-drivenness and real-time data:** Real-time and event-based computing play major roles in various topics in Robotics including real-time control, human-robot interactions or sensor perception and fusion. REST architectural principles, however, enforce stateless request-response messaging patterns and, therefore, seemingly violate some core requirements of Robotics. Yet, there is a number of established REST-conformant techniques for providing event-triggered callbacks and protocol upgrades to full-duplex streaming communication. We believe these techniques to be potentially suitable, but they require further evaluation and elaboration in the Robotics domain.

D4 **Semantic descriptions of robotic configurations and scenarios:** With Linked Data we have the ability to integrate all addressable web resources with the Web representation of robotic systems. A real added value results from exposing the entire robotic world model with configurations, abilities, sensor data, knowledge about the environment and the current tasks of a robot in a semantically useful way. Additional resources can be integrated to provide a better understanding of the robotic world model for other Web-based systems.

D5 **Unifying binary and semantic robotic data:** The semantic description of robotic configurations and scenarios has received some attention [25, 27, 41, 43], yet, the issue is far from resolved. The Linked Data Community is working incessantly on techniques for mapping existing data in the RDF data model in order to lower the barrier for generating Linked Data. While there is tremendous progress in mapping structured data [8] and semi-structured data [9], there is a surprising gap with respect to binary data, e.g. robotic perception data, that necessitates further thought.

D6 **Declarative integration:** We aim to use declarative means to describe robotic data and functionality, and wish to achieve an ecosystem in which providers of data and functionality interlink without the need for direct coordination. An ideal system would automatically determine how to combine data and functionality to ultimately arrive at the stated goal of a user. However, such a system has proven elusive [24, 50]. We thus opt for a declarative coordination of distributed data and functionality into coherent Linked Data-driven robotic applications.

Despite considerable uptake of REST architectural principles in the Robotics community [23, 34, 42, 45], guidelines or specifications for building RESTful robotic systems and applications are missing. In the following sections, we outline a minimalistic core specification for Linked Robotics (cf. overview in Figure 1) and envision its usage in conjunction with one or more domain-specific refinements.

4 ROBOTIC THING MODEL

Robotic Things (RT) are digital representations of robotic systems or applications accessible via RESTful interfaces. HTTP requests to access, modify, create or delete Robotic Things are accepted and processed by Robotic Thing servers. Such servers can manage two kinds of Robotic Things, those resources whose state is represented using RDF (cf. Section 5) and those using other formats. In the following, we specify the use of HTTP for accessing, updating, creating and deleting resources from servers that expose Robotic Things.

[1]We use X-RMM as a shorthand for Level-X Richardson Maturity Model.

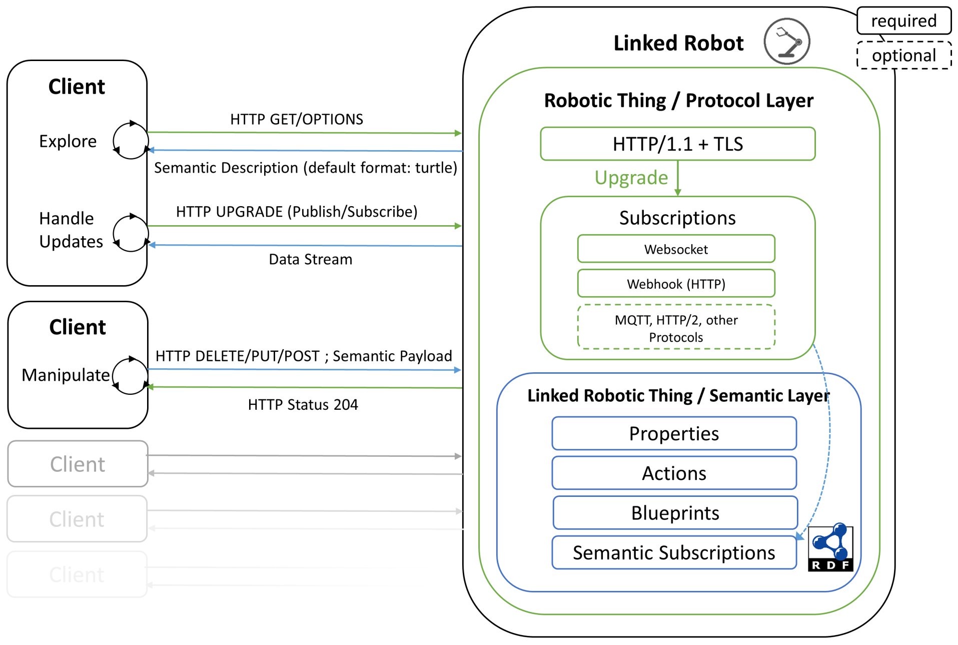

Figure 1: Overview of the Linked Robotic Thing Model.

R0.1 A RT server must at least be HTTP/1.1 conformant server.
R0.2 A RT server should support secure connections with HTTP over TLS. RT servers that are exposed on the Internet must always implement HTTPS or other security mechanisms in addition or in place of HTTPS.
R0.3 A Robotic Thing must have a URI endpoint for itself and all of its sub-resources.
R0.4 A RT server must support GET on its root resource.
R0.5 A RT server must support GET for retrieving resource representations, POST for resource creation, PUT for resource updates or state changes, and DELETE for resource removal.
R0.6 A RT server must implement HTTP status codes 200, 400 and 500 providing at least generic indications for successful actions, client-side errors or server-side errors.
R0.7 A RT server should return a HTTP status code 204 for all successful PUT, POST, and DELETE requests.
R0.8 A RT server must use the response Location header field to provide information about the location of a newly created resource for all successful POST requests.
R0.9 A Robotic Thing must advertise associated sub-resources using the HTTP response Link header field, and a link relation type of related (that is, rel="related").
R0.10 A RT server must support OPTIONS for each of its exposed resources and must advertise permitted resource interactions in the response Allow header field.
R0.11 A RT server must use entity tags as response ETag header values for all responses that contain representations and for responses to HEAD requests.
R0.12 A RT server must implement the WebSocket protocol [10].
R0.13 A RT server should implement a HTTP callback mechanism, e.g. WebHooks[2] or RestHooks[3].
R0.14 A RT server must support the request Upgrade header field for subscription creation.
R0.15 A request Upgrade header field value of 'websocket' must result in the creation of a WebSocket-based communication as specified in [10].
R0.16 A request Upgrade header field value of 'callback' must result in the creation of a HTTP callback, and requires the request Callback header field to specify a callback URI.
R0.17 A RT server should support the Accept header field in subscription creation requests to allow for content-negotiation.

Discussion. Starting with basic requirements (R0.1-R0.3) for reaching 1-RMM, we satisfy the criteria for 2-RMM with (R0.4-R0.7), and employ best practices (R0.8-R0.10) in order to provide hypermedia controls at the protocol level. On the whole, we establish a maturity level of 3-RMM (D1). The semantics of HTTP operations is clarified (R.05), and their admissibility is either specified by default (R0.4) or advertised via the Options Header (R0.10). In conjunction with the definition of expectable HTTP return and error codes (R0.6-R0.7), we provide a simple, yet well-defined interaction model (D2). The ETag header (R0.11) is widely used for cache validation and conditional requests, and can be used for performance improvements of polling algorithms (D3). With the potential for protocol upgrades or event-triggered callback mechanisms (R0.12-R0.17), we strive for communication paradigms more suitable for event-driven and real-time data (D3). The remaining desiderata (D4-D6) can not be fulfilled within the Robotic Thing Model. We therefore extend the notion of Robotic Things in the next section.

[2]http://www.webhooks.org/
[3]http://resthooks.org/

```
ros:RevoluteJointShape
  rdf:type sh:NodeShape ;
  sh:targetClass ros:RevoluteJoint ;
  sh:property [
    sh:path [ sh:alternativePath ( ros:rotation hybrit:property ) ] ;
    sh:minCount "1"^^xsd:integer ;
    sh:maxCount "1"^^xsd:integer ;
    sh:class hybrit:Property ;
    sh:node spatial:AxisAngleShape ;
  ] ;
  sh:property [
    sh:path [ sh:alternativePath ( ros:jointReferenceFrame hybrit:property ) ] ;
    sh:minCount "1"^^xsd:integer ;
    sh:maxCount "1"^^xsd:integer ;
    sh:class hybrit:Property ;
    sh:node maths:CoordinateSystemShape ;
  ] ;
  sh:property [
    sh:path ros:preceedingLink ;
    sh:minCount "1"^^xsd:integer ;
    sh:maxCount "1"^^xsd:integer ;
    sh:class ros:Link ;
  ] ;
  sh:property [
    sh:path ros:succeedingLink ;
    sh:minCount "1"^^xsd:integer ;
    sh:maxCount "1"^^xsd:integer ;
    sh:class ros:Link ;
  ] ;
```

Figure 2a: A SHACL-based Blueprint specifying a joint.

```
@base <http://example.org/baxter/> .

<right_s1>
  rdf:type hybrit:RoboticThing , ros:RevoluteJoint ;
  hybrit:blueprint ros:RevoluteJointShape ;
  hybrit:subscriptions <right_s1/subscriptions> ;
  hybrit:action <right_s1/actions/rotate> ;
  hybrit:property
    <right_s1/friction> ,
    <right_s1/rotation> ,
    <right_s1/jointReferenceFrame> ;
  rdfs:seeAlso <> ;
  dcterms:title "right_s1" ;
  ros:friction <right_s1/friction> ;
  ros:jointReferenceFrame  <right_s1/jointReferenceFrame> ;
  ros:rotation <right_s1/rotation> ;
  xhv:prev <right_upper_shoulder> ;
  xhv:next  <right_lower_shoulder> ;
  ros:preceedingLink <right_upper_shoulder> ;
  ros:succeedingLink  <right_lower_shoulder> .
```

Figure 2b: A Blueprint-conformant Linked Robotic Thing.

5 LINKED ROBOTIC THINGS

Linked Robotic Things are Robotics Things, but have the unique quality that at least one of their representations is based on RDF. Furthermore, Linked Robotic Things are often associated with a number of specific sub-resources that a client can use to learn about the Linked Robotic Thing's topology and interface constraints, to discover and execute available actions, to access and manipulate a set of parameters, or to subscribe for event notifications in order to observe particular state changes.

5.1 Blueprints

Any useful interface definition for a Linked Robotic Thing should enable the definition of its RDF graph topology, allow to express structural constraints on its RDF representations, and communicate expected and expectable graph patterns for admissible RESTful interactions.

```
@base <http://example.org/baxter/> .

<right_s1/rotation>
  rdf:type spatial:AxisAngle , spatial:Rotation3D , hybrit:Property ;
  rdfs:seeAlso <right_s1> ;
  spatial:rotationAngle [
    rdf:type maths:Scalar ;
    vom:numericalValue "0"^^xsd:float ;
    vom:unit maths:Radian
  ] ;
  spatial:rotationAxis  [
    rdf:type maths:Vector3D ;
    maths:x "0"^^xsd:float ;
    maths:y "0"^^xsd:float ;
    maths:z "1"^^xsd:float
  ] .
```

Figure 3: A Property describing a joint's rotation parameter.

Unfortunately, the formulation and validation of such interface constraints using RDF(S) or OWL2 is ineligible due to the Open World Assumption adopted by RDF(S) and OWL2, and the lack of a Unique Name Assumption. In fact, neither RDF(S) nor OWL2 provide constraint validation semantics in our sense.

However, RDF shape constraint languages [21, 32, 35, 38, 44] provide just enough expressiveness and, first and foremost, constraint validation semantics for our purposes. We intend to use RDF shape expressions to describe the topology of Linked Robotics Things, to specify and validate constraints on HTTP message bodies, and besides description and validation, for code generation and data integration.

We collect a number of such RDF shape expressions using a dereferenceable (sub-) resource of a Linked Robotic Thing, called its *Blueprint*. Blueprints provide completely machine-processable documentation about a Linked Robotic Thing and its interfaces. For illustrational purposes, we give a simple SHACL-based blueprint of a revolute joint (cf. Figure 2a) that can be used to validate the Linked Robotic Thing instance of Figure 2b.

5.2 Properties

Linked Robotic Things typically expose a collection of configuration parameters and dynamic variables governing overall behaviour and execution. Such configuration parameters and dynamic variables are essential components of a robotic system, hence, we make them uniquely identifiable and resolvable in form of so-called *Properties*.

A *Property* is a dereferenceable (sub-) resource of a Linked Robotic Thing that describes and keeps track of a specific parameter or variable, e.g. a joint's rotation (cf. Figure 3) or a link's moment of mass inertia, and enables clients to automatically inspect and potentially manipulate state and behaviour. Properties may detail on the type of the exposed quantity, its time-stamped values, units of measurement, dimensions or data types. Implementers are encouraged to re-use existing vocabularies such as QUDT[4], VOM[5] or UO[6].

5.3 Subscriptions

Besides plain GET requests on Robotic Things and their potentially time-varying Properties (cf. Section 5.2), a Robotic Thing server allows clients to subscribe to event notifications via the HTTP

[4]http://qudt.org/
[5]http://vocab.arvida.de/2015/06/vom
[6]http://purl.bioontology.org/ontology/UO

```
@base <http://example.org/baxter/right_s1/subscriptions> .

<>
  rdf:type ldp:Container ;
  ldp:contains <840c14b0-6def-11e7-907b-a6006ad3dba0> .

<840c14b0-6def-11e7-907b-a6006ad3dba0>
  rdf:type hybrit:Subscription , hybrit:WebCallback ;
  hybrit:onResource <right_s1/rotation> ;
  hybrit:callbackURL "http://www.example.org/receiver"^^xsd:anyURI ;
  hybrit:mediaType "text/turtle" ;
  dcterms:identifier "840c14b0-6def-11e7-907b-a6006ad3dba0" .
```

Figure 4: A subscription on a joint rotation parameter.

protocol upgrade mechanism. New subscriptions are created by issuing a GET request on the respective resource with the request Upgrade header field value set to 'websocket' or 'callback', and the request Accept header field set for proper content-negotiation. In case of creating a 'callback' subscription, the request Callback header field must specify a callback URI.

A Linked Robotic Thing provides minimalistic support for the management of subscriptions via a dereferenceable *Subscriptions* (sub-) resource. The Subscriptions resource is a Linked Data Platform container and allows to list current subscriptions using a GET request as well as cancelling a subscription using a DELETE request. For each current subscription, the Subscriptions resource must provide a RDF description (cf. Figure 4), e.g. detailing on the subscription type, additional protocol specific information, and the resource being subscribed to.

5.4 Actions

An *Action* enables a Linked Robotic Thing to advertise its potential resource state transitions to a client. These advertisements are exposed in dereferenceable *Action* (sub-) resources that provide the information necessary for a client to select and execute an appropriate Action so that a certain desired goal is achieved. Since we aim to exchange this information in a machine-processable way at runtime instead of being hardcoded into the client at design time, clients can be decoupled from the RT server and adapt to changes more easily.

An Action must effectively describe and constrain the data it may consume for and produce upon a successful invocation. Such specifications may link to a Linked Robotic Thing's Blueprint (cf. Section 5.1), or directly given using a RDF shape expression (cf. Figure 5a). Regardless of the specific formulation, the client must expect the server to validate the client-provided input against the expressed constraints. The server may reject an invocation if this validation fails. The server may ignore any additional triples not covered by the formulated constraints.

A *Binding* describes a means that the client can use to execute an Action using a specific communication protocol stack. In the scope of this paper, Bindings are restricted to HTTP requests. The predicate hybrit:binding is used to a link an Action to one or more Bindings. Based on a chosen Binding, a client may employ simple code generation techniques in order to execute the associated Action (cf. Figure 5b).

As a final remark, we point out that Actions can be offered at any level of abstraction, thus, they can be summarised to higher level Actions.

```
@base <http://example.org/baxter/> .

_:body rdf:type http-core:Body .

<right_s1/action/rotate>
  rdf:type hybrit:Action , hybrit:AsynchronousAction ;
  hybrit:consumes [
    rdf:type sh:NodeShape ;
    sh:targetNode _:body ;
    sh:targetClass maths:Scalar ;
    sh:property [
      sh:path vom:numericalValue ;
      sh:minCount "1"^^xsd:integer ;
      sh:maxCount "1"^^xsd:integer ;
      sh:datatype xsd:float ;
    ] ;
    sh:property [
      sh:path vom:unit ;
      sh:minCount "1"^^xsd:integer ;
      sh:maxCount "1"^^xsd:integer ;
      sh:minInclusive 0 ;
      sh:maxExclusive 360 ;
      sh:hasValue maths:Degree ;
    ] ;
  ] ;
  hybrit:produces <right_s1/action/rotate/producesshape> ;
  hybrit:binding [
    rdf:type http-core:Request ;
    http-core:mthd http-methods:PUT ;
    http-core:requestURI  <right_s1/action/rotate> ;
    http-core:body _:body ;
  ] .
```

Figure 5a: An Action allowing to rotate a joint.

```
POST /baxter/right_s1/action/rotate HTTP/1.1
Host: http://example.org

<>
  rdf:type maths:Scalar ;
  vom:numericalValue "43.6"^^xsd:float ;
  vom:unit vom:Degree .
```

Figure 5b: An Action execution rotating the joint.

6 LINKED ROBOTIC THING MODEL

The Linked Robotic Thing model is intended as a contract between clients and Linked Robotic Things exposed on the Web, thus, allowing clients to automatically discover and use their properties and action possibilities. In the following, we specify a set of simple rules and conventions for Linked Robotic Things.

R1.1 A Linked Robotic Thing is a Robotic Thing.
R1.2 A RT server must respond with a Turtle representation[7] of the requested Linked Robotic Thing, unless HTTP content-negotiation requires a different outcome.
R1.3 Linked Robotic Things must have at least one rdf:type set explicitly in their RDF representations.
R1.4 In the absence of special knowledge of the application or domain, clients must assume that any RDF representation of a Linked Robotic Thing can have multiple rdf:type triples with different objects.
R1.5 RDF representations of Linked Robotic Things should reuse existing vocabularies instead of creating their own duplicate vocabulary terms. Standard vocabularies such as RDF and

[7] We prefer Turtle syntax for its legibility as default representation. Implementers are free to support more RDF surface syntaxes via HTTP content-negotiation.

RDF Schema, Dublin Core or Linked Data Platform, should be used whenever possible.

R1.6 Clients should always assume that different Linked Robotic Things of the same type may not all have the same set of predicates in their RDF representations, and the set of predicates that are used in the state of any one Linked Robotic Thing is not limited to any pre-defined set.

R1.7 A RT server must not require clients to implement inferencing in order to recognize the subset of content defined by a Linked Robotic Thing's RDF representation. Other specifications built on top of this may require clients to implement inferencing.

R1.8 A Linked Robotic Thing must expose a Subscriptions container resource using a RDF triple whose subject is the Linked Robotic Thing's URI, whose predicate is hybrit:subscriptions and whose object is the URI for the Subscriptions container resource.

R1.9 A Linked Robotic Thing should expose a collection of Property resources using RDF triples whose subject is the Linked Robotic Thing's URI, whose predicate is hybrit:property and whose objects are the URIs for the Property resources.

R1.10 A Linked Robotic Thing should expose a collection of Action resources using RDF triples whose subject is the Linked Robotic Thing's URI, whose predicate is hybrit:action and whose objects are the URIs for the Action resources.

R1.11 A Linked Robotic Thing may expose a Blueprint resource using an RDF triple whose subject is the Linked Robotic Thing's URI, whose predicate is hybrit:blueprint and whose object is the URI for the Blueprint resource.

R1.12 A Linked Robotic Thing may provide additional meta data to precisely describe the meaning of individual building blocks of a model in a machine-understandable way.

Discussion. Linked Robotic Things are - by definition (R1.1-R1.2) - a semantic extension of Robotic Things (D4). We specify some minimal expectations (R1.3-R1.6) regarding a Linked Robotic Thing's description (D4), and furthermore provide a light-weight, intuitive conceptual model with standardized link relations (R1.8-R1.11). This not only allows to (re-)use vocabularies and ontologies suitable for a more refined semantic descriptions (D4), but also serves as a minimalistic interaction model (D2) offering hypermedia controls (D1) in addition to (R0.8-R0.10). Using HTTP content-negotiation (R1.2), a Linked Robotic Thing may be available in several different representations (R1.1) including binary formats (D5). We explicitly reflect the semantic aspects (R1.8, R0.12-R.017) of available event-based and real-time interfaces (D3). Using Blueprints as machine-processable descriptions of Linked Robotics Things (R1.11) and Actions (R1.10) as advertisements of their potential state transitions, we aim for improved system interoperation (D6).

7 FUTURE WORK

With HTTP/2 and Google QUIC[8], a number of interesting protocols for event-driven and real-time communication arise. We intend a more detailed investigation and evaluation of these protocols in the context of the Linked Robotic Things (cf. D3, Section 3).

[8]https://datatracker.ietf.org/wg/quic/documents/

Robotic applications make heavy use of binary data, but there is surprisingly little work on generic mappings between application-specific binary data and RDF. We see possible research contributions on how to lower the barrier between binary data and Linked Data (cf. D5, Section 3).

The primary focus of this paper is on the design of Linked Robotic Thing servers. While we are convinced that our specifications will help to simplify the implementation of clients, we feel that the actual design of such clients, and how they efficiently integrate in a heterogeneous environment of Linked Robotic Things and other Web resources, should receive more attention.

The Linked Robotic Things Model does not make strong assumptions (R0.2) regarding any security or privacy concerns. At this point, we rely on the standard Web security mechanisms, such as HTTPS, and standard Web infrastructure. We thus see future research in how to include security and privacy aspects into the Linked Robotic Things Model.

8 CONCLUSION

Considering developments in the field of networked robotics, from first tele-operated systems in the 1990's, to nowadays cloud-based RESTful Robotics applications, we have identified how these RESTful Robotics applications may benefit from Linked Data approaches. Such have already been investigated in the Linked Service community, and have also been considered to be applied to Web Robotics applications. To overcome the lack of clear and concise specifications of how to employ Linked Data concepts in the field of Robotics, in this paper, we have made the following contributions:

We have proposed a number of desiderata for robotic systems in Linked Data Environments and have proposed a multi-layer model to address these: A protocol layer to ensure basic communication, and a (machine-readable) semantic layer that provides additional information about the resource. To ensure the basic communication on the protocol layer, we proposed a definition of a Robotic Thing. Building on that, we have defined a set of requirements on the semantic level to create a common understanding, how server and clients interact with each other and how we can explore the world model of a Linked Robot Thing in an efficient and simple way. Thus, the implementation and communication between systems using this model are much more robust, simpler and testable.

We have discussed the requirements on both levels, compared them to the previously stated desiderata, and shown that all desiderata can be fulfilled by the given requirements. We thus present an exhaustive, concise, well-defined specification for Linked Robotic Things as building blocks for Linked Robotic applications on the World Wide Web.

ACKNOWLEDGMENTS

This work is supported by the Federal Ministry of Education and Research of Germany in the project Hybr-iT (Förderkennzeichen 01IS16026A).

REFERENCES

[1] Noriaki Ando, Takashi Suehiro, Kosei Kitagaki, Tetsuo Kotoku, and Woo-Keun Yoon. 2005. RT-middleware: distributed component middleware for RT (robot technology). In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE, 3933–3938. http://ieeexplore.ieee.org/abstract/document/1545521/

[2] Tim Berners-Lee. 1996. Univeral Resource Identifiers – Axioms of Web architecture. (Dec. 1996). https://www.w3.org/DesignIssues/Axioms.html

[3] Tim Berners-Lee. 2010. Is your linked open data 5 star. *https://www. w3. org/DesignIssues/LinkedData. html* (2010).

[4] Radhia Bouziane, Labib Sadek Terrissa, Soheyb Ayad, and Jean-Franois Breth. 2016. A ROS-based approach for robot as a service (Web services based solution). (2016). https://www.researchgate.net/profile/Labib_Sadek_Terrissa2/publication/311583836_Web_services_based_solution_for_robot_as_a_service_in_the_cloud/links/584f0ef208aed95c25099630.pdf

[5] Herman Bruyninckx. 2001. Open robot control software: the OROCOS project. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, Vol. 3. IEEE, 2523–2528. http://ieeexplore.ieee.org/abstract/document/933002/

[6] Yinong Chen, Zhihui Du, and Marcos García-Acosta. 2010. Robot as a service in cloud computing. In *Service Oriented System Engineering (SOSE), 2010 Fifth IEEE International Symposium on*. IEEE, 151–158. http://ieeexplore.ieee.org/abstract/document/5570010/

[7] Barney Dalton and Ken Taylor. 2000. Distributed robotics over the internet. *IEEE Robotics & Automation Magazine* 7, 2 (2000), 22–27. http://ieeexplore.ieee.org/abstract/document/848264/

[8] Souripriya Das, Seema Sundara, and Richard Cyganiak. 2012. R2RML: RDB to RDF mapping language. (2012). http://www.citeulike.org/group/14833/article/11522782

[9] Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. 2014. RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data.. In *LDOW*. https://www.researchgate.net/profile/Ruben_Verborgh/publication/264274087_RML_A_Generic_Language_for_Integrated_RDF_Mappings_of_Heterogeneous_Data/links/53d8fd2b0cf2631430c38a7b.pdf

[10] Ian Fette. 2011. The websocket protocol. (2011). https://tools.ietf.org/html/rfc6455

[11] Roy Thomas Fielding. 2000. *Architectural styles and the design of network-based software architectures*. Ph.D. Dissertation. University of California, Irvine. http://jpkc.fudan.edu.cn/picture/article/216/35/4b/22598d594e3d93239700ce79bce1/7ed3ec2a-03c2-49cb-8bf8-5a90ea42f523.pdf

[12] Martin Fowler. 2010. Richardson Maturity Model: steps toward the glory of REST. *Online at http://martinfowler. com/articles/richardsonMaturityModel. html* (2010).

[13] Dominique Guinard, Vlad Trifa, Friedemann Mattern, and Erik Wilde. 2011. From the internet of things to the web of things: Resource-oriented architecture and best practices. *Architecting the Internet of things* (2011), 97–129. http://link.springer.com/content/pdf/10.1007/978-3-642-19157-2.pdf#page=129

[14] Dominique Guinard, Vlad Trifa, and Erik Wilde. 2010. A resource oriented architecture for the web of things. In *Internet of Things (IOT), 2010*. IEEE, 1–8. http://ieeexplore.ieee.org/abstract/document/5678452/

[15] Michael Hausenblas. 2009. Linked data applications. *First Community Draft, DERI* (2009). https://wtlab.um.ac.ir/images/e-library/linked_data/Linked%20Data%20Applications.pdf

[16] Huosheng Hu, Lixiang Yu, Pui Wo Tsui, and Quan Zhou. 2001. Internet-based robotic systems for teleoperation. *Assembly Automation* 21, 2 (2001), 143–152. http://www.emeraldinsight.com/doi/pdf/10.1108/01445150110388513

[17] Koji Kamei, Shuichi Nishio, Norihiro Hagita, and Miki Sato. 2012. Cloud networked robotics. *IEEE Network* 26, 3 (2012), 28–34. http://dblp.uni-trier.de/db/journals/network/network26.html#KameiNHS12

[18] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg. 2015. A Survey of Research on Cloud Robotics and Automation. *IEEE Transactions on Automation Science and Engineering* 12, 2 (April 2015), 398–409. https://doi.org/10.1109/TASE.2014.2376492

[19] Felix Leif Keppmann, Tobias Käfer, Steffen Stadtmüller, René Schubotz, and Andreas Harth. 2014. High performance linked data processing for virtual reality environments. In *Proceedings of the 2014 International Conference on Posters & Demonstrations Track-Volume 1272*. CEUR-WS. org, 193–196. http://dl.acm.org/citation.cfm?id=2878502

[20] Felix Leif Keppmann, Tobias Käfer, Steffen Stadtmüller, Rene Schubotz, and Andreas Harth. 2014. Integrating highly dynamic RESTful linked data APIs in a virtual reality environment. In *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*. IEEE, 347–348. http://ieeexplore.ieee.org/abstract/document/6948482/

[21] Holger Knublauch and Dimitris Kontokostas. 2017. Shapes Constraint Language (SHACL). (2017). https://www.w3.org/TR/shacl/

[22] Anis Koubâa. 2014. A Service-Oriented Architecture for Virtualizing Robots in Robot-as-a-Service Clouds.. In *ARCS*. 196–208. https://books.google.de/books?hl=de&lr=&id=U8O6BQAAQBAJ&oi=fnd&pg=PA196&dq=A+Service-Oriented+Architecture+for+Virtualizing+Robots+in+Robot-as-a-Service+Clouds&ots=B10oKSPyTT&sig=VP6Hz54Rs6J9SOHiJ4pj01_K6kE

[23] Anis Koubaa. 2015. ROS as a service: web services for robot operating system. *Journal of Software Engineering for Robotics* 6, 1 (2015), 1–14. https://www.researchgate.net/profile/Anis_Koubaa/publication/309668701_ROS_As_a_Service_Web_Services_for_Robot_Operating_System/links/581c59c508aeccc08aec5689/ROS-As-a-Service-Web-Services-for-Robot-Operating-System.pdf

[24] Reto Krummenacher, Martin Hepp, Axel Polleres, Christoph Bussler, and Dieter Fensel. 2005. WWW or What is Wrong with Web services. In *Web Services, 2005. ECOWS 2005. Third IEEE European Conference on*. IEEE, 9–pp. http://ieeexplore.ieee.org/abstract/document/1595733/

[25] Lars Kunze, Tobias Roehm, and Michael Beetz. 2011. Towards semantic robot description languages. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 5589–5595. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5980170

[26] Markus Lanthaler and Christian Gütl. 2013. Hydra: A Vocabulary for Hypermedia-Driven Web APIs. *LDOW* 996 (2013). https://pdfs.semanticscholar.org/6f29/4df1ea316c73e68ebaaf7966661daa13cfe2.pdf

[27] Gi Hyun Lim, Il Hong Suh, and Hyowon Suh. 2011. Ontology-based unified robot knowledge for service robots in indoor environments. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 41, 3 (2011), 492–509. http://ieeexplore.ieee.org/abstract/document/5605259/

[28] Sachiko Nakagawa, Noboru Igarashi, Yosuke Tsuchiya, Masahiko Narita, and Yuka Kato. 2012. An implementation of a distributed service framework for cloud-based robot services. In *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 4148–4153. http://ieeexplore.ieee.org/abstract/document/6389225/

[29] Lung Ngai, Wyatt S. Newman, and Vincenzo Liberatore. 2002. An experiment in internet-based, human-assisted robotics. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, Vol. 2. IEEE, 2190–2195. http://ieeexplore.ieee.org/abstract/document/1014864/

[30] Kevin R. Page, David C. De Roure, and Kirk Martinez. 2011. REST and Linked Data: a match made for domain driven development?. In *Proceedings of the Second International Workshop on RESTful Design*. ACM, 22–25. http://dl.acm.org/citation.cfm?id=1967435

[31] Savas Parastatidis, Jim Webber, Guilherme Silveira, and Ian S. Robinson. 2010. The role of hypermedia in distributed system development. In *Proceedings of the First International Workshop on RESTful Design*. ACM, 16–22. http://dl.acm.org/citation.cfm?id=1798379

[32] Peter F. Patel-Schneider. 2017. ASHACL: Alternative Shapes Constraint Language. *arXiv:1702.01795 [cs]* (Feb. 2017). http://arxiv.org/abs/1702.01795 arXiv: 1702.01795.

[33] Carlos Pedrinaci and John Domingue. 2010. Toward the Next Wave of Services: Linked Services for the Web of Data. *J. ucs* 16, 13 (2010), 1694–1719. http://www.jucs.org/jucs_16_13/toward_the_next_wave/jucs_16_13_1694_1719_pedrinaci.pdf

[34] Steffen Planthaber. 2015. Rocks new http-based API for robot control. In *Proceedings of the RIC Project Day Workgroup "Framework & Standardization". RIC Project Day, March 19, Bremen (DFKI Documents, D)*, Vol. 15-01. Selbstverlag, 44–51.

[35] Eric Prud'hommeaux, Jose Emilio Labra Gayo, and Harold Solbrig. 2014. Shape expressions: an RDF validation and transformation language. In *Proceedings of the 10th International Conference on Semantic Systems*. ACM, 32–40. http://dl.acm.org/citation.cfm?id=2660523

[36] Partha Pratim Ray. 2016. Internet of Robotic Things: Concept, Technologies, and Challenges. *IEEE Access* 4 (2016), 9489–9500. http://ieeexplore.ieee.org/abstract/document/7805273/

[37] Leonard Richardson and Sam Ruby. 2008. *RESTful web services*. " O'Reilly Media, Inc.". https://books.google.de/books?hl=de&lr=&id=XUaErakHsoAC&oi=fnd&pg=PP1&dq=Richardson+L,+Ruby+S+(2007)+RESTful+Web+Services.+O%27Reilly+Media,+Inc&ots=5keoGiuHsw&sig=KCGDIn0ps0S70f_OgLtcWsavP5M

[38] Arthur G. Ryman, Arnaud Le Hors, and Steve Speicher. 2013. OSLC Resource Shape: A language for defining constraints on Linked Data. *LDOW* 996 (2013). http://ceur-ws.org/Vol-996/papers/ldow2013-paper-02.pdf

[39] Steve Speicher, John Arwe, and Ashok Malhotra. 2015. Linked data platform 1.0. *W3C Recommendation, February* 26 (2015).

[40] Kenneth Taylor and Barney Dalton. 2000. Internet robots: A new robotics niche. *IEEE Robotics & Automation Magazine* 7, 1 (2000), 27–34. http://ieeexplore.ieee.org/abstract/document/833572/

[41] Moritz Tenorth and Michael Beetz. 2013. KnowRob: A knowledge processing infrastructure for cognition-enabled robots. *The International Journal of Robotics Research* 32, 5 (2013), 566–590. http://journals.sagepub.com/doi/abs/10.1177/0278364913481635

[42] Moritz Tenorth, Ulrich Klank, Dejan Pangercic, and Michael Beetz. 2011. Web-enabled robots. *IEEE Robotics & Automation Magazine* 18, 2 (2011), 58–68. http://ieeexplore.ieee.org/abstract/document/5876223/

[43] M. Tenorth, A. C. Perzylo, R. Lafrenz, and M. Beetz. 2013. Representation and Exchange of Knowledge About Actions, Objects, and Environments in the RoboEarth Framework. *IEEE Transactions on Automation Science and Engineering* 10, 3 (July 2013), 643–651. https://doi.org/10.1109/TASE.2013.2244883

[44] Dominik Tomaszuk. 2017. RDF Validation: A Brief Survey. In *Beyond Databases, Architectures and Structures. Towards Efficient Solutions for Data Analysis and Knowledge Representation (Communications in Computer and Information Science)*. Springer, Cham, 344–355. https://doi.org/10.1007/978-3-319-58274-0_28

[45] Russell Toris, Julius Kammerl, David V. Lu, Jihoon Lee, Odest Chadwicke Jenkins, Sarah Osentoski, Mitchell Wills, and Sonia Chernova. 2015. Robot Web Tools: Efficient messaging for cloud robotics. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 4530–4537. https://doi.org/10.1109/IROS.2015.7354021

[46] Hans Utz, Stefan Sablatnog, Stefan Enderle, and Gerhard Kraetzschmar. 2002. Miro-middleware for mobile robot applications. *IEEE Transactions on Robotics and Automation* 18, 4 (2002), 493–497. http://ieeexplore.ieee.org/abstract/document/1044362/

[47] Joachim Van Herwegen, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. 2015. Query execution optimization for clients of triple pattern fragments. In *European Semantic Web Conference*. Springer, 302–318. http://link.springer.com/chapter/10.1007/978-3-319-18818-8_19

[48] Ruben Verborgh, Andreas Harth, Maria Maleshkova, Steffen Stadtmüller, Thomas Steiner, Mohsen Taheriyan, and Rik Van de Walle. 2014. Survey of semantic description of rest APIs. In *rest: Advanced Research Topics and Practical Applications*. Springer, 69–89. http://link.springer.com/chapter/10.1007/978-1-4614-9299-3_5

[49] Ruben Verborgh, Olaf Hartig, Ben De Meester, Gerald Haesendonck, Laurens De Vocht, Miel Vander Sande, Richard Cyganiak, Pieter Colpaert, Erik Mannens, and Rik Van de Walle. 2014. Low-cost queryable linked data through triple pattern fragments. In *Proceedings of the 2014 International Conference on Posters & Demonstrations Track-Volume 1272*. CEUR-WS. org, 13–16. http://dl.acm.org/citation.cfm?id=2878457

[50] Ruben Verborgh, Erik Mannnens, and Rik Van de Walle. 2015. Bottom-up Web apis with self-descriptive responses. In *Proceedings of the First Karlsruhe Service Summit Workshop-Advances in Service Research, Karlsruhe, Germany, February 2015*, Vol. 7692. KIT Scientific Publishing, 143. https://books.google.de/books?hl=de&lr=&id=z24GBwAAQBAJ&oi=fnd&pg=PA143&ots=QK-W8Fpc0b&sig=-JMS6Sa4fJbqD4592PnG64woRUo

[51] Ruben Verborgh, Miel Vander Sande, Pieter Colpaert, Sam Coppens, Erik Mannens, and Rik Van de Walle. 2014. Web-Scale Querying through Linked Data Fragments.. In *LDOW*. https://www.researchgate.net/profile/Ruben_Verborgh/publication/264274086_Web-Scale_Querying_through_Linked_Data_Fragments/links/53f498b10cf2fceacc6e918d.pdf

[52] Ruben Verborgh, Miel Vander Sande, Olaf Hartig, Joachim Van Herwegen, Laurens De Vocht, Ben De Meester, Gerald Haesendonck, and Pieter Colpaert. 2016. Triple Pattern Fragments: A low-cost knowledge graph interface for the Web. *Web Semantics: Science, Services and Agents on the World Wide Web* 37 (2016), 184–206. http://www.sciencedirect.com/science/article/pii/S1570826816000214

[53] Markus Waibel, Michael Beetz, Javier Civera, Raffaello D'Andrea, Jos Elfring, Dorian Gálvez-López, Kai Häussermann, Rob Janssen, J.M.M. Montiel, Alexander Perzylo, Björn SchieÃ§le, Moritz Tenorth, Oliver Zweigle, and René De Molengraft. 2011. RoboEarth. *IEEE Robotics & Automation Magazine* 18, 2 (June 2011), 69–82. https://doi.org/10.1109/MRA.2011.941632

[54] Erik Wilde and Michael Hausenblas. 2009. RESTful SPARQL? You name it!: aligning SPARQL with REST and resource orientation. In *WEWST '09*. ACM Press, 39–43. https://doi.org/10.1145/1645406.1645412

[55] Oliver Zweigle, René van de Molengraft, Raffaello d'Andrea, and Kai Häussermann. 2009. RoboEarth: connecting robots worldwide. In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*. ACM, 184–191. http://dl.acm.org/citation.cfm?id=1655958