ECA2LD: From Entity-Component-Attribute runtimes to Linked Data applications

Torsten Spieldenner[1,2], René Schubotz[1], and Michael Guldner[1]

[1] German Research Center for Artificial Intelligence (DFKI)
[2] Saarbrücken Graduate School of Computer Science
firstname.lastname@dfki.de

Abstract. Large-scale IoT applications, like Smart Cities, are ever-changing pieces of software. Software built for such IoT environments needs a thorough design to be adaptable to the changes in the underlying systems. The Entity-Component-Attribute (ECA) pattern is well-suited for the design of changeable and maintainable software artifacts. However, the nature of large-scale IoT applications does not only enforces changeable, but also interoperable design of software components. For this, W3C working groups propose to use the Web as an IoT convergence platform. To unleash its full potential and to help to tackle pressing cross-domain interoperability issues, this emerging Web of Things is expected to evolve into a Semantic Web of Things which will heavily rely on Linked Data principles. While the generation of Linked Data from data storage layers has undergone thorough research, the Linked Data compliant exposure of dynamic run-time environments is to this day incomplete. Towards this end, we formalize the ECA design pattern and present an generic and auto-generatable mapping from ECA runtimes to a structure compliant with the W3C Linked Data Platform. This structural mapping may be declaratively augmented by domain-specific semantics, and lifts a software design pattern highly suitable for large-scale IoT applications to Semantic Web of Things.

Keywords: Entity-Component-Attribute model; Linked Data; Resource Description Framework (RDF); Interoperability

1 Introduction

The continual change that software undergoes during its lifetime is generally called evolution, and the degree to which it is easy or hard to change existing software is often called *changeability* [17, 14]. Especially in perdurable and large scale Internet of Things (IoT) platforms, software design with the intention to optimize changeability is imperative [27]. Towards this end, the architectural pattern of *Entity-Component-Attribute (ECA) based software design* is particularly well-suited [26]. Focusing on the principle of "composition over inheritance", the role of an entity is no longer determined by class inheritance or attribution hierarchy, but dynamically determined by the set of attached components. This significantly improves changeability of entities and reuse of components.

ECA patterns have been successfully applied in the design of changeable IoT platforms and applications [19], however, the enablement of *seamless cross-domain interoperability* between independently developed IoT applications and platforms, one of the central challenges facing IoT [18], is not directly addressed by this design paradigm.

In this respect, the W3C Web of Things Working Group[3] proposes to use the Web as an IoT convergence platform. The group develops initial standards for this *Web of Things* (WoT) [2] by defining 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.

The *Semantic Web of Things* (SWoT) [21] will heavily rely on Linked Data principles [11] to semantically describe IoT entities in terms of their actions, properties, events and metadata [22] independent of the underlying IoT technology stacks. For large-scale scenarios and environments, the development of IoT platforms and applications on the SWoT will require software tooling that enables

1. (semi-) automated mappings from IoT application data layouts to RDF
2. declarative augmentation of Linked Data with application-specific semantics
3. exposition of dynamic IoT runtime data as Linked Data

While there are numerous works investigating (semi-) automated mappings from heterogeneous data structures and serializations to the RDF data model [6, 3, 16, 8, 15], little research has been conducted on the dynamic mapping of runtime environments [20, 12] to RDF. In particular, we are not aware of any works on how to leverage ECA-driven software applications on the Semantic WoT.

Contributions. This paper makes the following contributions. We formalize the notation of an ECA system, and detail on the automated *structural mapping* between ECA runtimes and the W3C Linked Data Platform[4]. Next, we explain how domain experts can declaratively augment these generated structural mappings with domain-specific semantics. Finally, we outline how a Linked Data client may materialize these application-specific RDF triples either locally or by delegation to a suitable Linked Data Service.

Structure. In the remainder of the paper, we first discuss current research in mapping approaches from non-RDF sources to RDF data. We then present a formalization of the Entity-Component-Attribute model as basis for changeable software in Section 3. In Section 4, we briefly outline the W3C Linked Data

[3] https://www.w3.org/WoT/WG/
[4] https://www.w3.org/TR/ldp/

Platform standard, and present our automated mapping for structural interoperability between ECA data and the Linked Data Platform in Section 5. Section 6 outlines a declarative augmentation of the automated structural mapping with semantics from domain-specific vocabularies. We conclude with summary in Section 7.

2 Related Work

The available literature investigating (semi-) automated data mappings from various data structures and serializations into the RDF data model is vast. However, little research has been conducted on the dynamic mapping of runtime environments [20, 12] to RDF. In what follows, we briefly survey the related literature.

RDBMS. Numerous work investigates how to translate between RDF datagraphs and relational databases. The W3C specifies a *Direct Mapping*[5] (DM) from relational database structures directly to RDF. *R2RML*[6] is a similar approach that enables customization of the mapping. Automatic procedures have been proposed to create the R2RML mapping, which yields RDF graphs similar to the results of Direct Mapping [8]. Bizer et al. [6] present *D2RQ*, an approach that translates semantic queries into native queries against non-RDF databases and maps the result to RDF.

OOP. Focusing on business logic rather than data storage layers, other work also investigated mappings between RDF data sources, and *object-oriented programming* (OOP) languages. Bartolos et al. [4] discuss mappings between object-oriented classes, and present a possibility to automatically create a class model from ontologies. The resulting class model then operates as access to the underlying RDF data. *ActiveRDF* by Oren et al. [20] provides an object-oriented API for scripting languages to operate on RDF datasets. However, they found that concepts of object-oriented programming are at times to strict to allow for an automated mapping. Limiting concepts are for example class inheritance rules, encapsulation, or class attribution. Hillairet et al. [12] show how to match the at this time widely used Eclipse Modeling Framework (EMF) with RDF data sources. Approaches that operate directly on OOP concepts struggle with OOP concepts like class inheritance, encapsulation.

Semi-structured Data. A third class of RDF mapping approaches uses as source semi-structured data, such as XML files, or comma separated values (CSV). Apache Any23[7], offered as library, Web service, or command line tool, translates a variety of source formats, such as CSV and YAML, to RDF representations in Turtle, Notation 3, and others. Dimou et al. [9] present RML, an extension to the R2RML W3C standard, to map between heterogeneous semi-structured

[5] https://www.w3.org/TR/rdb-direct-mapping/
[6] https://www.w3.org/TR/r2rml/
[7] https://any23.apache.org/

data and RDF. Gupta et al. [10] provide Karma, a semi-automatic RDF extraction framework not only from relational database, but also from sources given as CSV, XML, or JSON.

Despite its wide application in different domains, and work that investigates how application design profits from semantic information inherent to ECA-based software, there exists to our knowledge no particular analysis about how to leverage Entity-Component-Attribute driven applications to Linked Data.

3 Changeability by Entity-Component-Attribute designs

In the following, we will give an outline of requirements we see for the design of changeable large-scale software projects. We give a formal definition of the established Entity-Component-Attribute model and discuss how it is suitable to fulfill the requirements towards changeable software.

3.1 Changeable software requirements

We derive requirements towards changeable software from the notion of *aspect-oriented software design* as presented by Kiczales et. al [13]. For this, it is necessary to avoid *cross-cutting concerns* in the code. As main pitfalls that break changeability of software, aspect-oriented design distinguishes:

Code scattering: Code that implements a concept or logic is distributed over several modules or classes. As a result, adding, changing, or removing logic from an application requires to change several modules at once. Code scattering is avoided by modeling software and its data in distinct modules for every task.

Code tangling: While code that implements a certain feature may be entirely contained in its own module, dependencies between modules can still break changeability of software. This happens for example if code of one module refers to, or makes calls to, code in another module. If one module changes its interface to which other modules make calls, all other modules need to be changed as well. A way to avoid code tangling is a data-centralistic approach by which separate software modules operate on a shared data layer, without direct calls between the modules.

3.2 Entity-Attribute Models

Above requirements are met by Entity-Attribute based software design. The understanding of Entity-Attribute models varies in literature. In the following, we summarise available variants and formalize the notation of Entity-Attribute models.

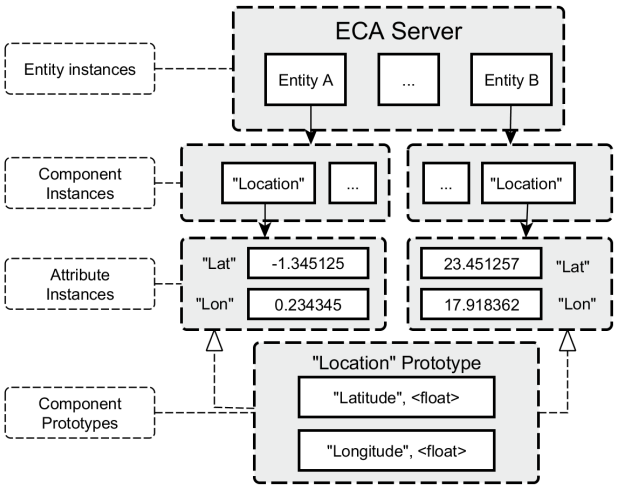

Fig. 1: Entity-Component-Attribute model based on the example of a component that defines a "Location".

Common for all variations is the notion of an *entity* as an empty data container which is closer specified by a set of typed *attributes* that carry the actual values. The IoT Context Broker by Moltchanov et. al [19] keeps to the levels of entities and attributes.

The systems RealXtend [1, 7] and FiVES [25] include the notion of *components* (*Entity-Component-Attribute pattern, ECA*; see also Fig. 1). Components can be considered as prototypes of attribute sets that belong to the same concept. When a component is attached to an entity instance, a new instance of the component and the respective attribute set is created from this prototype. RealXtend and FiVES share this design with game engines like Unity3D[8] and Unreal Engine[9].

RealXtend equips components with application logic that is directly contained as code in the component implementation. Same holds for game engines.

The work by Wiebusch et al. [26] as well as the FiVES server system consider the Entity-Component-Attribute model as data model only. Logic is implemented in independent *systems* (referred to as *plugins* in FiVES), with a careful design of how external logic accesses the data.

We adapt the understanding of components as by Wiebusch, and FiVES, with logic implemented separately to avoid the tight coupling between components and their specific implementation as in RealXtend or game engines. We derive from this the following formal definition of the ECA architectural pattern.

[8] http://www.unity3d.com
[9] http://www.unrealengine.com

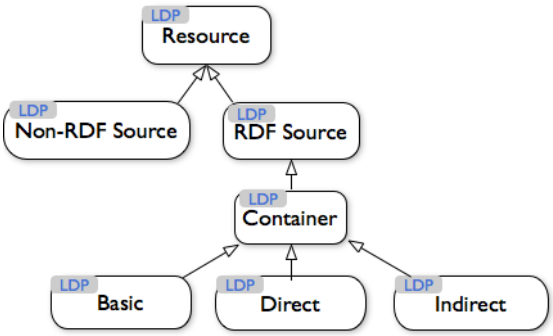

Fig. 2: High-level structure of the Linked Data Platform[10]

Let $\mathbf{e}$ denote an entity instance, and $\mathcal{P}_\mathbf{C}$ denote the set of all component prototypes. Then we define the following sets:

$\mathbf{E}$ is the set of all entity instances. An entity instance is defined as $\mathbf{e} = (n_\mathbf{e}, \mathbf{C_e})$, with $n_\mathbf{e} \in \Sigma^+$ being the unique identifier for $\mathbf{e}$ over alphabet Σ, and $\mathbf{C_e}$ being the set of component instances attached to $\mathbf{e}$.

$\mathbf{C_e}$ is the set of all component instances attached to an entity instance $\mathbf{e}$. A component instance is defined as $\mathbf{c} = (n_\mathbf{c}, p_\mathbf{c}, \mathbf{A_{c,e}})$, with $n_\mathbf{c} \in \Sigma^+$ being the unique identifier for $\mathbf{c}$, $p_\mathbf{c} \in \mathcal{P}_\mathbf{C}$ being the prototype that $\mathbf{c}$ is an instance of, and $\mathbf{A_{c,e}}$ being the set of all attribute instances attached to $\mathbf{c}$.

$\mathbf{A_{c,e}}$ the set of all attribute instances attached to a component instance $\mathbf{c}$. An attribute instance is defined as $\mathbf{a} = (n_\mathbf{a}, v, t)$, with $n_\mathbf{a} \in \Sigma^+$ being the unique identifier for $\mathbf{a}$, v denoting the attribute instance's current *value*, and t denoting the ECA runtime type of $\mathbf{a}$.

The role of an entity within the application is by this entirely determined by the set of attached components. Components are implemented and operate independent of each other. This avoids the issue of code scattering. The *composition-over-inheritance* principle of the model also avoids code-tangling, as it eliminates class inheritance and attribution hierarchies.

4 The Linked Data Platform

The W3C *Linked Data Platform* recommendation provides best practices for read-write Linked Data applications on the Web. It describes how to model applications in terms of a minimal set of RDF resources (cf. Figure 2). Moreover access patterns to these different resources are specified for Linked Data clients.

[10] Image taken from https://www.w3.org/TR/ldp/#fig-ldpc-types

The basic element of LDP is a `ldp:Resource`. Every `ldp:Resource` must be an HTTP endpoint with at least HTTP/1.1 protocol compatibility, and accept at least HTTP GET requests, and others depending on the type of resource. From the `ldp:Resource` are derived a number of resource types with the following roles:

`ldp:RDFSource` exposes general RDF data. Upon a `HTTP GET` request, it `MUST` return a full RDF graph in `text/turtle` format (or `application/ld+json`, if requested). `HTTP PUT` or `HTTP PATCH` can be used to update triples in the graph that is provided by the resource.

`ldp:Container` is a `ldp:RDFSource` that manages a set of LDP Resources and provides information about access, modification, and filtering of the contained elements.

`ldp:BasicContainer` is a `ldp:Container` that specifies linked documents in the form of *Containment Triples* of the form (*container-uri*, `ldp:contains`, *document-uri*). The `ldp:BasicContainer` does not require to specifically state the semantic relationship between the container resource itself, and its containing elements.

The W3C LDP recommendation specifies more concepts. In the scope of this paper, however, we will only make use of above concepts.

5 Structural Interoperability with ECA-based Systems

In the following, we detail on the automated structural mapping between ECA runtime environments and W3C LDP compliant Linked Data servers. By repeated application of a set of mapping rules (cf. ① to ④), structural interoperability [23] between ECA runtimes and LDP servers is established.

We assume the existence of functions $\nu : \Sigma^+ \rightarrow \texttt{IRI}$ and $\rho : \mathcal{P}_{\mathbf{C}} \rightarrow \texttt{IRI}$ for minting fresh `IRI`s from identifiers and component prototypes. Although several guidelines exist for minting `IRI`s[11], we do not make assumptions on ν or ρ.

$$\frac{(n_{\mathbf{e}}, \mathbf{C_e}) \in \mathbf{E} \quad \forall (n_{\mathbf{c}}, p_{\mathbf{c}}, \mathbf{A_{c,e}}) \in \mathbf{C_e}}{\begin{array}{l} \nu(n_{\mathbf{e}})\ \texttt{rdf:type}\ \texttt{ldp:BasicContainer}\ . \\ \nu(n_{\mathbf{e}})\ \texttt{dct:identifier}\ \text{“}n_{\mathbf{e}}\text{”}\hat{}\hat{}\texttt{xsd:String}\ . \\ \nu(n_{\mathbf{e}})\ \texttt{ldp:hasMemberRelation}\ \texttt{dct:hasPart}\ . \\ \nu(n_{\mathbf{e}})\ \texttt{dct:hasPart}\ \nu(n_{\mathbf{c}})\ . \end{array}}\ ①$$

① Each entity instance $\mathbf{e} = (n_{\mathbf{e}}, \mathbf{C_e})$ is mapped to a `ldp:BasicContainer` with `IRI` $\nu(n_{\mathbf{e}})$. This entity container maintains a membership triple ($\nu(n_{\mathbf{e}})$,

[11] https://www.w3.org/TR/cooluris/

dct:hasPart, $\nu(n_{\mathbf{c}})$) for each component instance $(n_{\mathbf{c}}, p_{\mathbf{c}}, \mathbf{A}_{\mathbf{c},\mathbf{e}})$ attached to $\mathbf{e}$.

$$\frac{(n_{\mathbf{c}}, p_{\mathbf{c}}, \mathbf{A}_{\mathbf{c},\mathbf{e}}) \in \mathbf{C}_{\mathbf{e}} \quad \forall (n_{\mathbf{a}}, v, t) \in \mathbf{A}_{\mathbf{c},\mathbf{e}}}{\begin{array}{l}\nu(n_{\mathbf{c}})\ \texttt{rdf:type}\ \texttt{ldp:BasicContainer}\ .\\ \nu(n_{\mathbf{c}})\ \texttt{dct:identifier}\ \text{“}n_{\mathbf{c}}\text{”}\hat{}\hat{}\texttt{xsd:String}\ .\\ \nu(n_{\mathbf{c}})\ \texttt{dct:isPartOf}\ \nu(n_{\mathbf{e}})\ .\\ \nu(n_{\mathbf{c}})\ \texttt{ldp:hasMemberRelation}\ \texttt{dct:hasPart}\ .\\ \nu(n_{\mathbf{c}})\ \texttt{dct:hasPart}\ \nu(n_{\mathbf{a}})\ .\\ \nu(n_{\mathbf{c}})\ \texttt{rdfs:isDefinedBy}\ \rho(p_{\mathbf{c}})\ .\end{array}}\ ②$$

② Each component instance $\mathbf{c} = (n_{\mathbf{c}}, p_{\mathbf{c}}, \mathbf{A}_{\mathbf{c},\mathbf{e}})$ is mapped to a `ldp:BasicContainer` with `IRI` $\nu(n_{\mathbf{c}})$. This component container uses `dct:isPartOf` to indicate its containing entity container $\nu(n_{\mathbf{e}})$ and maintains a membership triple ($\nu(n_{\mathbf{c}})$, dct:hasPart, $\nu(n_{\mathbf{a}})$) for each attribute instance $(n_{\mathbf{a}}, v, t)$ attached to $\mathbf{c}$. In addition, we use `rdfs:isDefinedBy` to indicate an authoritative resource $\rho(p_{\mathbf{c}})$ semantically defining the component container $\nu(n_{\mathbf{c}})$. We detail on $\rho(p_{\mathbf{c}})$ in the next section.

$$\frac{(n_{\mathbf{a}}, v, t) \in \mathbf{A}_{\mathbf{c},\mathbf{e}}}{\begin{array}{l}\nu(n_{\mathbf{a}})\ \texttt{rdf:type}\ \texttt{ldp:RDFResource}\ .\\ \nu(n_{\mathbf{a}})\ \texttt{dct:identifier}\ \text{“}n_{\mathbf{a}}\text{”}\hat{}\hat{}\texttt{xsd:String}\ .\\ \nu(n_{\mathbf{a}})\ \texttt{dct:isPartOf}\ \nu(n_{\mathbf{c}})\ .\\ \nu(n_{\mathbf{a}})\ \texttt{rdf:value}\ \text{“}\mu(v)\text{”}\hat{}\hat{}\nu(t)\ .\end{array}}\ ③$$

③ Each attribute instance $(n_{\mathbf{a}}, v, t) \in \mathbf{A}_{\mathbf{c},\mathbf{e}}$ is represented by a `ldp:RDFResource` with `IRI` $\nu(\mathbf{n_a})$. This attribute resource uses `dct:isPartOf` to indicate its containing component container $\nu(n_{\mathbf{c}})$. The triple ($\nu(n_{\mathbf{a}})$, `rdf:value`, “$\mu(v)$”$\hat{}\hat{}\nu(t)$) encodes the attribute's current value v and type t as a typed literal “v”$\hat{}\hat{}\nu(t)$ (cf. ④).

④ Since the RDF datatype abstraction is compatible with XML Schema, we rely on the data type support between an ECA runtime enviroment and XML Schema Types for datatype conversion. Given an attribute $(n_{\mathbf{a}}, v, t) \in \mathbf{A}_{\mathbf{c},\mathbf{e}}$, we denote by $\nu(t)$ the datatype `IRI` of the RDF-compatible XSD type corresponding to t. The lexical form $\mu(v)$ may be any lexical form, ie. a Unicode string in Normal Form C, from $\nu(t)$'s lexical space that represents the same value as v. Extensions that handle domain-specific or user-defined datatypes beyond the RDF-compatible XSD types are expected to behave as outlined here.

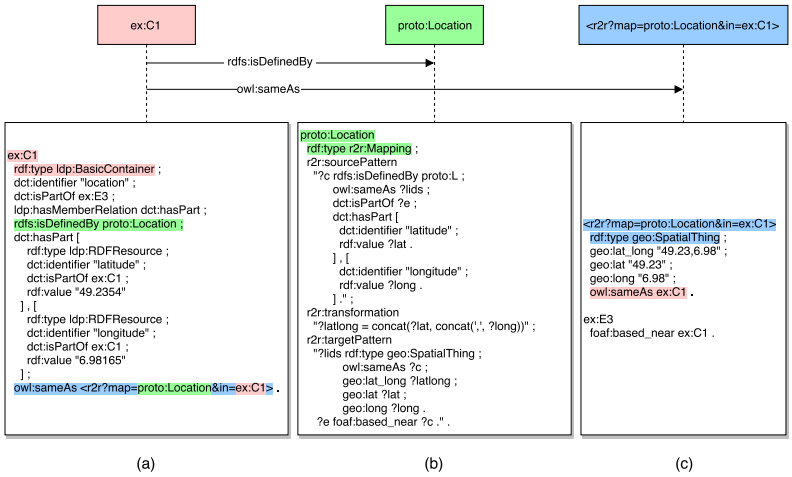

Fig. 3: Structural mapping and semantic augmentation of a "Location" component (cf. Figure 1).

6 Augmenting Domain-specific Semantics

The ECA model induces an implicit semantic understanding of the underlying data. The prototype of a component assigns to a component a specific concept that is modeled by that component. For example, Figure 1 defines a component prototype that describes a location in geo-coordinates.

Rules ① to ④ provide a structural mapping from ECA runtime objects to Web resources described using the LDP vocabulary. Our structural mapping is generic and auto-generatable, but so far it does not express said application-specific semantics that are contained in the ECA model.

A domain expert tasked with semantic augmentation thus requires support for specifying expressive RDF mappings that enable fine-grained term correspondences, literal transformations and structural graph transformations at dataset-level. Ideally, these RDF mappings should be dereferencable and executable, self-contained and interoperably represented as RDF triples. Natural candidates for expressing and executing such RDF mappings are SPIN SPARQL[12], RIF in RDF[13], the LDIF framework[14] or the R2R framework[15].

In the scope of this paper and without loss of generality, we describe and publish such RDF mappings using the R2R Mapping Language [5]. Similar to SPARQL CONSTRUCT queries, a `r2r:Mapping` (cf. Figure 3(b)) has a `r2r:sourcePattern`, `r2r:transformation` and a `r2r:targetPattern`.

[12] https://www.w3.org/Submission/2011/SUBM-spin-sparql-20110222/
[13] https://www.w3.org/TR/rif-in-rdf/
[14] http://ldif.wbsg.de/
[15] http://wifo5-03.informatik.uni-mannheim.de/bizer/r2r/

The source pattern is matched against data generated from rules ① to ④ (cf. Figure 3(a)) and produces a set of variable bindings. Transformations define how variable bindings are transformed before being inserted into the target pattern. The target pattern is used to produce the triples resulting from the `r2r:Mapping` (cf. Figure 3(c)).

Rule ② uses `rdfs:isDefinedBy` to indicate an authoritative resource $\rho(p_{\mathbf{c}})$ defining all instances of a component prototype $p_{\mathbf{c}} \in \mathcal{P}_{\mathbf{C}}$. Hence, a domain expert can publish her RDF mapping under $\rho(p_{\mathbf{c}})$ and make it discoverable for Linked Data clients.

By retrieving a representation of $\rho(p_{\mathbf{c}})$, a Linked Data client will be instructed on how to locally render additional application-specific RDF triples. Note that execution of a RDF mapping may also be delegated to a suitable Linked Data Service (LIDS) [24]. We suggest `owl:sameAs` (cf. Figure 3(a)) to indicate the respective LIDS invocation `IRI`.

7 Conclusion

This paper presents a generic and auto-generatable structural mapping between Entity-Component-Attribute (ECA) runtimes and the W3C Linked Data Platform. First, we discuss the Entity-Component-Attribute model as suitable choice for changeable software, followed by a formal definition of the ECA design pattern. From this, a generic and auto-generatable structural mapping between ECA runtimes and the W3C Linked Data Platform is provided. Building upon this basic level of structural interoperability, we explain how domain experts may declaratively specify and publish expressive RDF mappings in order to convey the application-specific semantics of the respective ECA runtime objects. By executing the published RDF mappings, a Linked Data client is instructed on how to semantically interpret the dynamically exposed ECA runtime objects. A prototype implementation of the presented approach is available.

Acknowledgment

The work presented in this paper received funding from the European Union's project FI-NEXT under grant agreement no. 732851, and by the Federal Ministry of Education and Research of Germany in the project Hybr-iT under support code 01IS16026A.

References

1. Toni Alatalo. An entity-component model for extensible virtual worlds. *IEEE Internet Computing*, 15(5):30–37, 2011.

2. Dominique ard and Vlad Trifa. 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, 2009.
3. Sören Auer, Sebastian Dietzold, Jens Lehmann, Sebastian Hellmann, and David Aumueller. Triplify: light-weight linked data publication from relational databases. In *Proceedings of the 18th international conference on World wide web*, pages 621–630. ACM, 2009.
4. Peter Bartalos and Maria Bielikova. An approach to object-ontology mapping. In *IIT. SRC–Student Research Conference*, pages 9–16, 2007.
5. Christian Bizer and Andreas Schultz. The R2R framework: Publishing and discovering mappings on the Web. *COLD*, 665, 2010.
6. Christian Bizer and Andy Seaborne. D2RQ-treating non-RDF databases as virtual RDF graphs. In *Proceedings of the 3rd international semantic web conference (ISWC2004)*, volume 2004. Proceedings of ISWC2004, 2004.
7. Toni Dahl, Timo Koskela, Seamus Hickey, and Jarkko Vatjus-Anttila. A Virtual World Web Client utilizing an Entity-Component model. In *NGMAST*, pages 7–12. IEEE, 2013.
8. Luciano Frontino de Medeiros, Freddy Priyatna, and Oscar Corcho. Mirror: Automatic R2RML mapping generation from relational databases. In *International Conference on Web Engineering*, pages 326–343. Springer, 2015.
9. Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. Rml: A generic language for integrated rdf mappings of heterogeneous data. In *LDOW*, 2014.
10. Shubham Gupta, Pedro Szekely, Craig A Knoblock, Aman Goel, Mohsen Taheriyan, and Maria Muslea. Karma: A system for mapping structured sources into the semantic web. In *Extended Semantic Web Conference*, pages 430–434. Springer, 2012.
11. Tom Heath and Christian Bizer. Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology*, 1(1):1–136, 2011.
12. Guillaume Hillairet, Frédéric Bertrand, Jean Yves Lafaye, et al. Bridging EMF applications and RDF data sources. In *Proceedings of the 4th International Workshop on Semantic Web Enabled Software Engineering, SWESE*, 2008.
13. Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-oriented programming. In *Proceedings of the European Conference on Object-Oriented Programming (ECOOP)*, Finland, June 1997. Springer-Verlag, Berlin, Germany.
14. Herwig Mannaert, Jan Verelst, and Kris Ven. Towards evolvable software architectures based on systems theoretic stability. *Software: Practice and Experience*, 42(1):89–116, 2012.
15. Franck Michel, Loïc Djimenou, Catherine Faron-Zucker, and Johan Montagnat. Translation of relational and non-relational databases into RDF with xR2RML. In *11th International Confenrence on Web Information Systems and Technologies (WEBIST'15)*, pages 443–454, 2015.
16. Franck Michel, Johan Montagnat, and Catherine Faron-Zucker. *A survey of RDB to RDF translation approaches and tools.* PhD thesis, I3S, 2014.
17. Parastoo Mohagheghi and Reidar Conradi. An empirical study of software change: origin, acceptance rate, and functionality vs. quality attributes. In *Empirical Software Engineering, 2004. ISESE'04. Proceedings. 2004 International Symposium on*, pages 7–16. IEEE, 2004.

18. Saraju P Mohanty, Uma Choppali, and Elias Kougianos. Everything you wanted to know about smart cities: The internet of things is the backbone. *IEEE Consumer Electronics Magazine*, 5(3):60–70, 2016.
19. Boris Moltchanov and Oscar Rodriguez Rocha. A context broker to enable future IoT applications and services. In *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2014 6th International Congress on*, pages 263–268. IEEE, 2014.
20. Eyal Oren, Benjamin Heitmann, and Stefan Decker. ActiveRDF: Embedding Semantic Web data into object-oriented languages. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):191–202, 2008.
21. D. Pfisterer, K. Romer, D. Bimschas, O. Kleine, R. Mietz, C. Truong, H. Hasemann, A. Krller, M. Pagel, M. Hauswirth, M. Karnstedt, M. Leggieri, A. Passant, and R. Richardson. Spitfire: toward a semantic web of things. *IEEE Communications Magazine*, 49(11):40–48, November 2011.
22. René Schubotz, Christian Vogelgesang, André Antakli, Dmitri Rubinstein, and Torsten Spieldenner. Requirements and specifications for Robots, Linked Data and all the REST. In *Proceedings of 2nd Workshop on Linked Data in Robotics and Industry 4.0. (LIDARI-2017), located at Semantics 2017, Amsterdam, Netherlands*. CEUR, 2017.
23. Amit P Sheth. Changing focus on interoperability in information systems: from system, syntax, structure to semantics. In *Interoperating geographic information systems*, pages 5–29. Springer, 1999.
24. Sebastian Speiser and Andreas Harth. Integrating linked data and services with linked data services. In *Proceedings of the 8th Extended Semantic Web Conference on The Semantic Web: Research and Applications - Volume Part I*, ESWC'11, pages 170–184, Berlin, Heidelberg, 2011. Springer-Verlag.
25. Torsten Spieldenner, Michael Guldner, Sergiy Byelozyorov, and Philipp Slusallek. FiVES: An aspect-oriented Virtual Environment Server. In *Proceedings of the 2017 International Conference on Cyberworlds. International Conference on Cyberworlds (CyberWorlds-2017), September 20-22, Chester, United Kingdom*. IEEE Xplore, 2017.
26. Dennis Wiebusch and Marc Erich Latoschik. Decoupling the entity-component-system pattern using semantic traits for reusable realtime interactive systems. In *Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), 2015 IEEE 8th Workshop on*, pages 25–32. IEEE, 2015.
27. Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32, 2014.