Velocity on the Web - a PhD Symposium

Riccardo Tommasini
supervised by Emanuele Della Valle
Politecnico di Milano, Department of Electronic, Informatic, and Bioengineering
Milan, Italy
riccardo.tommasini@polimi.it

ABSTRACT

It is a streaming world: a new generation of Web Applications is pushing the Web infrastructure to evolve and process data as soon as they arrive. However, the Web of Data is not appealing to the growing number of Web Applications demanding to tame Data Velocity. To solve these issues, we need to introduce new key abstractions, i.e., stream and events. Moreover, we need to investigate how to identify, represent and process streams and events on the Web. In this paper, we discuss why taming Velocity on the Web of Data. We present a Design Science research plan that builds on the state of the art of Stream Reasoning and RDF Stream Processing. Finally, we present our research results, for representing and processing stream and events on the Web.

KEYWORDS

Web of Data, Stream Processing, Complex Event Processing, Stream Reasoning

ACM Reference Format:
Riccardo Tommasini. 2019. Velocity on the Web - a PhD Symposium. In *Companion Proceedings of the 2019 World Wide Web Conference (WWW '19 Companion), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3308560.3314192

1 PROBLEM

Data Velocity is the challenge to process data as soon as they arrive, and before it is too late. A new generation of Web applications is raising in a streaming world [41], where there is the need to process streams continuously and detect events reactively. Relevant application scenarios like environmental monitoring [1], social media trend analysis [24], and anomaly detection in financial markets [33] prove that Data Velocity is a critical challenge for the Web [17, 23].

The Web of Data (WoD) is an extension of the World Wide Web (Web) that encourages data sharing and interoperability. The fundamental notion of the Web are Resources to discover, to represent, and to process. WoD builds on the Web infrastructure. It uses URIs to enable discovery by identifying resources; it uses HTTP to enable links following (i.e., de-referencing) and processing.

Velocity is related to the entire Web data infrastructure, e.g., representing data, managing, delivering to recipients, reacting to changes, and responding to failures promptly. The Web underlying infrastructure is evolving accordingly by adding new protocols such as HTTP Long-Polling[1], HTTP Streaming[2], and WebSockets[3] that provide data continuously, i.e., in a *streaming* fashion; Server-sent events[4] and WebHooks[5] that supply data reactively, i.e., in an *event-driven* fashion. Henceforth we call HTTP extended with these new protocols HTTP*.

Web resources are heterogeneous. Therefore, WoD provides principles to enable interoperability: (i) exchange data using the Resource Description Framework (RDF)[6] a machine-readable data model (ii) build vocabularies using proper knowledge representation languages as RDF Schema[7] or the Web Ontology Language[8] (OWL); (iii) share vocabulary to annotate data semantically and (iv) use SPARQL[9] to process data in its broadest sense.[10]

The WoD is an evolving environment [21], but its key abstractions – i.e., RDF Triples, Named Graphs and RDF Datasets – are not adequate to tame data velocity. The WoD community proposed *RDF triples*, i.e., $< subject, predicate, object >$, to publish rich and interconnected Web documents. The community proposed *Named Graph* [10], i.e., set of RDF triples (graphs) identified by URIs, to organize document adding contextual metadata. Indeed, metadata are useful to support trust, syndication, and provenance, and they can boost data processing [18]. The WoD community proposed *RDF Datasets*, i.e., collections of Named RDF Graphs, to support the Linked Open Data (LOD) initiative. Indeed, LOD users need to discover, represent, and process multiple RDF graphs at once [5].

Problem Statement: the Web of Data is currently not appealing to the growing number of Web applications demanding to tame Velocity [7]. To solve this problem, the WoD needs new adequate key abstractions, i.e., *Streams* and *Events* [11]. Moreover, we need to investigate *how to discover, represent, and process* stream and events on the Web.

The research goals implied by this problem requires the design of artifacts that improve the WoD. Therefore, we run our investigation following a design science methodology [45], i.e., the *design cycle*, which is reported in Figure 1 and detailed in Section 3.

2 STATE OF THE ART

Previous attempts to tame velocity on the Web go under the umbrella of Stream Reasoning [17, 23]. Although there are proposals

This paper is published under the Creative Commons Attribution 4.0 International (CC-BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.
WWW '19 Companion, May 13–17, 2019, San Francisco, CA, USA
© 2019 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC-BY 4.0 License.
ACM ISBN 978-1-4503-6675-5/19/05.
https://doi.org/10.1145/3308560.3314192

[1] https://realtimeapi.io/hub/http-long-polling/
[2] https://realtimeapi.io/hub/http-streaming/
[3] https://www.w3.org/TR/websockets/
[4] https://www.w3.org/TR/2009/WD-eventsource-20090421/
[5] https://www.w3.org/TR/websub/
[6] https://www.w3.org/TR/rdf11-concepts/
[7] https://www.w3.org/TR/rdf-schema/
[8] https://www.w3.org/TR/owl2-overview/
[9] https://www.w3.org/TR/sparql11-overview/
[10] On the Web of Data, processing certainly means querying, but also reasoning, managing data access and storage, orchestration, choreography, and federation.

that discuss how to discover [4, 25, 29] and represent stream and events [26, 30, 44], most of the work focus on processing RDF Streams [9, 15, 16, 23] and Events [3, 13] (RSP). Due to lack of space, we cannot list all the relevant works; we invite the interested readers to consult the recent surveys [14, 23].

Discover. Sequeda and Corcho [29] proposed Linked Streaming Data as a mechanism to identify sensors and their observations. The paper provides a set of requirements and a proposal for URIs referencing space and time. Barbieri and Della Valle [4] propose to identify streams using URIs to enable discovery as for Linked Data. The authors focus on RDF Streams, and they model them using two kinds of named graphs. Each element of the stream is an Instantaneous Graphs (iGraphs), identified by a timestamp. The Stream Graphs (sGraphs) describes the current content of the window over the RDF Stream, and it references the iGraphs using the properties *rdfs:seeAlso* and *receivedAt*.

Represent. The Semantic Web proposals for representing events include, but are not limited to, the Event Ontology [25], Linking Open Descriptions of Events (LODE) [30] and Simple Event Model [44]. However, neither of these focuses on the events as a way to tame velocity. Only Rinne et al. [26] proposal for an ontology design pattern for events has the primary intent of reactive detection. Proposals for stream focus more on the content[11] than on stream as resources. In [4] Barbieri e Della Valle sketches a vocabulary to use for sGraphs. Moreover, Keskisarkka [22] proposed a SPIN extension for RSP which introduces some terminology based on RSP-QL (a reference model for modelling RSP engines). Recently, Schema.org included DataFeed[12] and DataFeedItem[13] as classes to model streams of content.

Process. The generality of the term “processing" calls for more specificity. Our investigation includes: *Querying*, i.e., event detection and continuous transformations as defined in the Information Flow Processing state-of-the-art [11] (IFP)[14]; *Data Management*, i.e., persisting and accessing streams and events, but also enabling orchestration, and choreography of services that produce/consume streams or events; and *Reasoning*, i.e., inference algorithms to enrich streams and derive implicit events.

RSP attempts to tame velocity and variety simultaneously [42] combining solutions from IFP and Semantic Web. RDF Streams are formalized as infinite sequences of pairs (G, t) where G is an RDF Graph and t is a non-decreasing timestamp [16]. Dell'Aglio et al. proposed a reference model called RSP-QL [16] that captures formal and operational semantics of continuous extensions of SPARQL [23]. RSP-QL extends the concepts of RDF Graph and RDF Dataset to enable the continuous semantics. Moreover, it adds stream-oriented operators such as Time Windows.RSEP-QL [13] extends RSP-QL with CEP operators to capture the semantics of RSP dialects for processing Events [2]. It introduces the concept of Basic Event Pattern as an extension of SPARQL basic graph pattern with temporal annotations.

[11] https://www.w3.org/TR/vocab-ssn/
[12] https://schema.org/DataFeed
[13] https://schema.org/DataFeedItem
[14] Due to the lack of space we cannot include a comprehensive background on IFP, the interested reader can consult [11] for a recent survey.

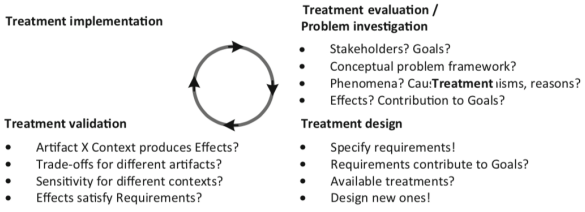

Figure 1: Design Cycle from [45].

Despite the attempts to formalize and uniform the approaches for RSP, existing dialects do not follow a common design methodology. In particular, they do not satisfy language design principles [12].

Stuckenschmidt et al. [32] envisioned the idea of cooperation of stream and event processors. They elaborated on orchestration proposing Cascading Reasoning, i.e., a hierarchical approach that combines multiple stream/event processor into layers of different expressiveness. They elaborated on choreography proposing Networks of Stream Reasoners cooperating on the Web. Finally, Calbimonte et al. [8] further develop this second idea towards a reactive Web Stream Processing. However, none of these visions reached the level of maturity to be validated and evaluated.

Works on reasoning divide into query rewriting for RSP [9] and incremental maintenance of ontology materialization [15]. Both the approaches aim at boosting reasoning performance to meet velocity requirements. Works on query rewriting attempt to extend the mapping languages with the concept of Window. Works on incremental reasoning focus on extending reasoners' capability with IFP concepts, while the idea of introducing reasoning features in an IFP engine is stil an unexplored path.

3 METHODOLOGY

For our investigation, we follow a Design Science (DS) [45] research methodology. DS is the investigation of the interaction of *artifacts*, e.g., algorithms, methods, techniques, with a *problem context* in order to improve it. Such interaction, called *treatment*, is designed by researchers to solve a problem. The application of treatments to the original problem context is called *implementation*.

Artifacts and treatments are designed following a subset of the engineering cycle [45], i.e., the *design cycle*, which is reported in Figure 1. It starts with a research goal. DS research goals divide into *treatment evaluation* and *problem investigation*. The the design cycles continuous with *treatment design* and *validation*.

Problem Investigation happens before implementation and requirements analysis. DS identifies *Design Problems*, which are the problems to (re-)design an artifact so that it better contributes to achieving some goal, and *Knowledge Questions*, i.e., explanatory or descriptive questions about the world.

Knowledge questions do not call for a change in the world, and the answer is a unique falsifiable proposition. On the contrary, design problems call for a change in the world. A solution is a design, i.e., a decision about what to do that is generally not unique.

Treatment Design. Design problems assume a problem context and some stakeholders to interview for deriving *requirements*, i.e., the desired property of the artifact. The researcher has to formulate

contribution arguments, i.e., sustain that an artifact which satisfies the requirements contribute to stakeholder goals.

Starting from a design problem we can ask knowledge questions about the artifact, the context, and the treatment, e.g., we can ask about the performance of the of the artifact. Similarly, knowledge questions can lead to a design problem, e.g., building a prototype of the artifact or simulating the context. These are all *empirical knowledge questions*, which require data about the world to answer them. To this extent, DS identifies *instrument design problems* that are the problem to design an instrument that support the investigation, e.g., a simulation or a prototype.

Treatment Validation aims at predicting how an artifact will interact with its context, without actually observing an implemented artifact in a real-world context. Validation is done under controlled conditions, before implementation, and using a *validation model*, i.e., exposing an artifact model to problem context models.

Validation include asking empirical knowledge questions about i) the effects produced by the treatment; ii) whether or not these effects satisfy the requirements; iii) the trade-offs between different artifacts applied to same problem context, and iv) the sensitivity between the same artifact applied to different contexts.

Research methods used for validation are (i) Expert opinion, which usually corresponds to peer-reviewing. (ii) Single-Case Mechanism Experiments, which studies the response of a validation model to a stimulus under controlled conditions and regarding the model internal. (iii) Statistical Difference-Making Experiments, in which a researcher compares the outcomes of different treatments applied to the same model context, and; (iv) Technical Action Research (TAR), which studies the properties of an artifact under development used by a researcher under real-world conditions.

Treatment Evaluation investigates the treatment as applied by stakeholders in the real world after implementation. Research methods used for evaluation since include (i) Surveys; (ii) Observational Case Studies; (iii) Single-Case Mechanism Experiments and (iv) Statistical Difference-Making Experiments.

4 PROPOSED APPROACH

To extend the Web of Data to tame velocity, we need to prescribe to discover, represent and process Stream and Events as resources.

On the one hand, considering recent extensions to the Web architecture, a discovery approach based on HTTP* and URIs is reasonable [29, 42]. Moreover, we should take into account what done by the Stream Reasoning community [14] on extending RDF and SPARQL query language for processing streams and events.

On the other hand, representing and processing in its broadest sense 10 are still open challenges towards taming velocity on the WoD. Thus, we formulate the following research question.

Research Question: *Can we enable representation and processing of stream and events to tame velocity in the Web of Data?*

Two problems arise from this research question: the representation problem and the processing problem for streams and events.

Moreover, the lack of a systematic approach of empirical research in the field [27] highlights the need for designing a research instrument that can help to answer many empirical knowledge questions. We refer to this as validation problem.

4.1 Representation Problem

Representing streams and events on the Web of Data is a design problem that we formulate as: *improve the Web of Data by designing an artifact, which satisfies Web developers' requirements to represent streams and events.*

WoD prescriptions for knowledge representation are RDFS and OWL2 and, thus, our solution must be compatible with them. Therefore, *we assume that existing knowledge representation languages for the Web of Data can capture streams and events semantics.*

As presented in Section 2, works on stream representation focus on the content, and a shared terminology to describe streams on the Web is still missing. Moreover, several ontological models compete for representing events, but they were designed without taking into account Velocity. Therefore, we plan to

- RP.1 design and share a vocabulary for streams descriptions that satisfies Web developers' requirements.
- RP.2 survey existing event models to propose an artifact that allows event definitions without neglecting Data Velocity [7].

4.2 Processing Problem

Processing streams and events as we discussed in Section 2 is also a design problem that we can formulate as: *improve the Web of Data by designing an artifact, which satisfies Web developers' requirements to process streams and events.*

In Section 2, we highlight how existing solutions for querying streams and detecting events on the Web do not follow common design principles. In particular, stream and events are not first-class citizen in the query languages. Moreover, cooperation of web stream and event processors calls for further investigations. Finally, works on reasoning focused on extending reasoners to meet velocity requirements rather than extend stream and event processing engines with inference capabilities. Therefore, we plan to

- PP.1 re-design an RSP approach that follows common language design principles for querying streams and events [12];
- PP.2 investigate techniques for orchestration and choreography of Web Stream Processors that can cooperatively solve problems in a decentralized Web environment [32].
- PP.3 investigate how to extend event processing languages and event-based architecture with reasoning capabilities to meet the performance requirements for treating velocity [31].

4.3 Validation Problem

The validation problem is an instrument design problem that we can formulate as: *improve empirical research for velocity on the WoD by designing an artifact, which satisfies researcher requirements, for validating artifacts that represent and process streams and events.*

Validation of the representation problem is based on design principles such as Tom Gruber's [20]. However, the for knowledge representation artifacts the improvement to the problem context is quantified by adoption. Technical Action Research (TAR) is a validation method that can (i) foster the artifacts adoption, while (ii) improving the artifact design with a strict feedback loop.

Validation for the processing problems, henceforth referenced as benchmarking, requires instead to measure precise performance requirements [31]. Although there are principles for domain-specific

benchmarking [19], existing benchmarks lack a systematic and comparative research approach. In particular, (i) prototypes might be unavailable and (ii) do not follow common design guidelines; (iii) the manual effort to set up the benchmark is vast, and (iv) the experimental environment does not guarantee reproducibility and repeatability. These issues make it hard to run the necessary validation, i.e., Single-Case Mechanism Experiments or Statistical Difference Making Experiments (see Section 3). Thus, we plan to

- VP.1 design an experimental environment that guarantees reproducibility and repeatability of experimental results;
- VP.2 enable Single-Case Mechanism Experiments by design an artifacts that can reproduce the variety of RSP engines without neglecting common design principles;
- VP.3 enable Statistical Difference Making Experiments by allowing fast-prototyping of alternative validation models and, thus, rapid exploration of the solution space; and
- VP.4 organize TAR sessions to using research instruments to validate representation and processing artifacts.

5 RESULTS

In this section, we discuss how we applied Design Science (DS) research methodology to our research plan. In particular, we focus on how we implemented the design cycle for the representation and processing problems, how we validated and how we evaluated the designed treatments.

5.1 Representation Problem

Regarding the representation problem, we designed a Domain-Specific Language (DSL) for event definition and processing called OBEP [35], and a vocabulary for describing streams called VoCaLS [36]. Following the design cycle, we collected requirements and we validated the treatments by expert opinion respectively for the DSL in [34], and for the vocabulary in [28].

```
<> a vocals:StreamDescriptor ; dcat:dataset :MilanTrafficStream .

:MilanTrafficStream a vocals:RDFStream ;
  vocals:hasEndpoint :MilanTrafficStreamEndpoint ;
  dcat:publisher <www.3cixty.eu>;
  dcat:description "Stream produced by traffic sensors in Milan"^.

:MilanTrafficStreamEndpoint a vocals:StreamEndpoint ;
  dcat:license <https://creativecommons.org/licenses/by-nc/4.0/> ;
  dcat:format frmt:JSON-LD ;
  dcat:accessURL "ws://example.org/traffic/milan".
```

Listing 1: Example of RDF Stream using VoCaLS.

We addressed 4.1 in [36], where we inquired WoD developers about their requirements for stream representation, and we provided evidence of the vocabulary usability. Listing 1 shows an example of stream description using VoCaLS that enables web stream processing scenarios not possible before.

We addressed 4.1 in [35], where we compared our event definition DSL with existing RSP event processing language, i.e., EP-SPARQL and RSEP-QL. We show that our approach, based on Description Logic (DL) reasoning, is more straightforward and more succinct than existing ones. DL is the foundation of OWL2 and, thus, the results respect the assumption made in Section 4. Listing 2 shows an example OBEP event declaration that is based on rules and, thus, can tame Velocity in the form of Event-Condition-Action [7].

```
EVENT :SmokeDetectionEvent subClassOf
   (hasContext some (hasProperty some Smoke))
EVENT :HighTemperaturEvent subClassOf TemperaturEvent and
   (result some (hasValue >= 40))
NAMED EVENT :FireEvent
MATCH :HighTemperaturEvent seq :SmokeDetectionEvent  WITHIN (5m)
```

Listing 2: Example of OBEP Language

5.2 Processing Problem

Regarding the processing problem, we planned to design several artifacts that address different aspects of processing, i.e., querying (4.2), data management (4.2) and reasoning (4.2).

Our investigation related to 4.2 develops in [35, 37, 38]. In [35], we also provide the specification of an event processing algebra that treats events as first-class citizens. Preliminary validation of the approach compares it with existing models to represent events. In [37], we highlighted the issues related to designing a query language based on RSP-QL formalization. A language that treats streams as first-class citizen should keep the languages constructs minimal, homogeneous, symmetric and orthogonal [12]. In [38], we proposed a first artifact prototype for such an RSP-QL engine called YASPER that will support our empirical investigation.

Our investigation related to 4.2 develops in [6], where we showed the feasibility of Cascading Reasoning, i.e., an orchestration approach, envisioned by Stuckenschmidt et al. [32]. The approach we designed combines exiting RSP and Stream Reasoning solutions into a layered architecture. To validate the treatment, we run Single-Case Mechanism Experiments using our prototype.

Last but not least, our investigation related to 4.2 develops in [6, 35] where we investigated how to combine existing stream reasoning approaches using RDF Streams. Moreover, an approach for event-based hierarchical reasoning is under review.

5.3 Validation Problem

Regarding the validation problem, we formalized a systematic and comparative research approach [39] and a Web environment for experimentation [40]. Our approach is based on two notions, i.e., Experiment and Baseline [39]. The former characterizes the model context for the validation, while the latter represents a versatile validation model [45].

To address 4.3, we developed an open-source Web environment called RSPLab [40] that guarantees reproducibility and repeatability of experimental results using vitalization techniques. The researcher needs to know the prototype internals in order to explain its response out of a stimulus to address 4.3. The researcher needs to fast-prototype many alternatives that can be compared systematically [39] to address 4.3.

Finally, we are organizing a series of tutorials to apply TAR with a knowledgeable audience. The initial sections include RW 2018, ISWE 2018, and The Web Conference 2019 [42].

6 CONCLUSION AND FUTURE WORK

In this paper, we discussed why and how extending the the Web of Data for taming Velocity. Our research goal is to allow Web developers to discover, represent and process streams and events.

To achieve it, we presented a design-science research approach that elaborates into a representation problem, a processing problem,

and a validation problem. We designed a vocabulary for stream description called VoCaLS [36] and a DSL for event definition called OBEP [35] to solve the representation problem. We developed our investigation for processing problem in three different directions, i.e., querying [35, 37, 38], reasoning [6, 35] and orchestration [6]. We designed an research instrument [40]and a methodology [39] to solve the validation problem.

Our research plan now focuses on validation and evaluation [45]. We aim at using Technical Action Research to validate VoCaLS and OBEP. To this extent, we are integrating VoCaLS and OBEP within our tutorials [43], and extending RSPLab to track usability metrics.

We plan to design a unified approach for querying that keeps the stream/event operations orthogonal of the remaining part of the algebra. We plan to evaluate such an approach running Statistical Difference-Making Experiments.

An approach that exploits VoCaLS descriptions to make a choreography of Web Stream Processors on a web environment is currently under review. We plan to validate the approach running Single-Case Mechanism Experiments.

REFERENCES

[1] Damminda Alahakoon and Xinghuo Yu. 2016. Smart Electricity Meter Data Intelligence for Future Energy Systems: A Survey. *IEEE Trans. Industrial Informatics* 12, 1 (2016), 425–436.

[2] Darko Anicic. 2012. *Event Processing and Stream Reasoning with ETALIS.* Ph.D. Dissertation. Karlsruhe Institute of Technology.

[3] Darko Anicic, Paul Fodor, Sebastian Rudolph, and Nenad Stojanovic. 2011. EP-SPARQL: a unified language for event processing and stream reasoning. In *WWW*. ACM, 635–644.

[4] Davide Francesco Barbieri and Emanuele Della Valle. 2010. A Proposal for Publishing Data Streams as Linked Data - A Position Paper. In *LDOW (CEUR Workshop Proceedings)*, Vol. 628. CEUR-WS.org.

[5] Christoph Böhm, Johannes Lorey, and Felix Naumann. 2011. Creating voiD descriptions for Web-scale data. *J. Web Sem.* 9, 3 (2011), 339–345.

[6] Pieter Bonte, Riccardo Tommasini, Emanuele Della Valle, Filip De Turck, and Femke Ongenae. 2018. Streaming MASSIF: Cascading Reasoning for Efficient Processing of IoT Data Streams. *Sensors* 18, 11 (2018), 3832.

[7] François Bry and Michael Eckert. 2007. Twelve Theses on Reactive Rules for the Web. In *Event Processing (Dagstuhl Seminar Proceedings)*, Vol. 07191.

[8] Jean-Paul Calbimonte. 2014. RDF Stream Processing: Let's React. In *OrdRing@ISWC (CEUR Workshop Proceedings)*, Vol. 1303. CEUR-WS.org, 1–10.

[9] Jean-Paul Calbimonte and Óscar Corcho. 2014. Evaluating SPARQL Queries over Linked Data Streams. In *Linked Data Management*. Chapman and Hall/CRC.

[10] Jeremy J. Carroll, Christian Bizer, Patrick J. Hayes, and Patrick Stickler. 2005. Named graphs. *J. Web Sem.* 3, 4 (2005), 247–267.

[11] Gianpaolo Cugola and Alessandro Margara. 2012. Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv.* 44, 3 (2012), 15:1–15:62.

[12] C. J. Date. 1984. Some Principles of Good Language Design (with especial reference to the design of database languages). *SIGMOD Record* 14, 3 (1984), 1–7.

[13] Daniele Dell'Aglio, Minh Dao-Tran, Jean-Paul Calbimonte, Danh Le Phuoc, and Emanuele Della Valle. 2016. A Query Model to Capture Event Pattern Matching in RDF Stream Processing Query Languages. In *EKAW (Lecture Notes in Computer Science)*, Vol. 10024. 145–162.

[14] Daniele Dell'Aglio, Emanuele Della Valle, Frank van Harmelen, and Abraham Bernstein. 2017. Stream reasoning: A survey and outlook. *Data Science* Preprint (2017).

[15] Daniele Dell'Aglio and Emanuele Della Valle. 2014. Incremental Reasoning on RDF Streams. In *Linked Data Management*. Chapman and Hall/CRC, 413–435.

[16] Daniele Dell'Aglio, Emanuele Della Valle, Jean-Paul Calbimonte, and Óscar Corcho. 2014. RSP-QL Semantics: A Unifying Query Model to Explain Heterogeneity of RDF Stream Processing Systems. *Int. J. Semantic Web Inf. Syst.* 10, 4 (2014).

[17] Stefano Germano, Thu-Le Pham, and Alessandra Mileo. 2015. Web Stream Reasoning in Practice: On the Expressivity vs. Scalability Tradeoff. In *RR (Lecture Notes in Computer Science)*, Vol. 9209. Springer, 105–112.

[18] Olaf Görlitz and Steffen Staab. 2011. SPLENDID: SPARQL Endpoint Federation Exploiting VOID Descriptions. In *COLD (CEUR Workshop Proceedings)*, Vol. 782.

[19] Jim Gray (Ed.). 1993. *The Benchmark Handbook for Database and Transaction Systems (2nd Edition).* Morgan Kaufmann.

[20] Thomas R. Gruber. 1995. Toward principles for the design of ontologies used for knowledge sharing? *Int. J. Hum.-Comput. Stud.* 43, 5-6 (1995), 907–928.

[21] Tobias Käfer, Ahmed Abdelrahman, Jürgen Umbrich, Patrick O'Byrne, and Aidan Hogan. 2013. Observing Linked Data Dynamics. In *ESWC (Lecture Notes in Computer Science)*, Vol. 7882. Springer, 213–227.

[22] Robin Keskisärkkä. 2016. Representing RDF Stream Processing Queries in RSP-SPIN. In *International Semantic Web Conference (Posters & Demos) (CEUR Workshop Proceedings)*, Vol. 1690. CEUR-WS.org.

[23] Alessandro Margara, Jacopo Urbani, Frank van Harmelen, and Henri E. Bal. 2014. Streaming the Web: Reasoning over dynamic data. *J. Web Sem.* 25 (2014), 24–44.

[24] Michael Mathioudakis and Nick Koudas. 2010. TwitterMonitor: trend detection over the twitter stream. In *SIGMOD Conference*. ACM, 1155–1158.

[25] Yves Raimond and Samer Abdallah. 2007. The event ontology.

[26] Mikko Rinne, Eva Blomqvist, Robin Keskisärkkä, and Esko Nuutila. 2013. Event Processing in RDF. In *WOP (CEUR Workshop Proceedings)*, Vol. 1188.

[27] Thomas Scharrenbach, Jacopo Urbani, Alessandro Margara, Emanuele Della Valle, and Abraham Bernstein. 2013. Seven Commandments for Benchmarking Semantic Flow Processing Systems. In *ESWC (Lecture Notes in Computer Science)*, Vol. 7882. Springer, 305–319.

[28] Yehia Abo Sedira, Riccardo Tommasini, and Emanuele Della Valle. 2017. Towards VoIS: A Vocabulary of Interlinked Streams. In *DeSemWeb@ISWC (CEUR Workshop Proceedings)*, Vol. 1934. CEUR-WS.org.

[29] Juan F. Sequeda and Óscar Corcho. 2009. Linked Stream Data: A Position Paper. In *SSN (CEUR Workshop Proceedings)*, Vol. 522. CEUR-WS.org, 148–157.

[30] Ryan Shaw, Raphaël Troncy, and Lynda Hardman. 2009. LODE: Linking Open Descriptions of Events. In *ASWC (Lecture Notes in Computer Science)*, Vol. 5926. Springer, 153–167.

[31] Michael Stonebraker, Ugur Çetintemel, and Stanley B. Zdonik. 2005. The 8 requirements of real-time stream processing. *SIGMOD Record* 34, 4 (2005), 42–47.

[32] Heiner Stuckenschmidt, Stefano Ceri, Emanuele Della Valle, and Frank van Harmelen. 2010. Towards Expressive Stream Reasoning. In *Semantic Challenges in Sensor Networks (Dagstuhl Seminar Proceedings)*, Vol. 10042. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany.

[33] Kia Teymourian, Malte Rohde, and Adrian Paschke. 2012. Knowledge-based processing of complex stock market events. In *EDBT*. ACM, 594–597.

[34] Riccardo Tommasini, Pieter Bonte, Emanuele Della Valle, Erik Mannens, Filip De Turck, and Femke Ongenae. 2016. Towards Ontology-Based Event Processing. In *OWLED (Lecture Notes in Computer Science)*, Vol. 10161. Springer, 115–127.

[35] Riccardo Tommasini, Pieter Bonte, Emanuele Della Valle, Femke Ongenae, and Filip De Turck. 2018. A Query Model for Ontology-Based Event Processing over RDF Streams. In *EKAW (Lecture Notes in Computer Science)*, Vol. 11313. Springer.

[36] Riccardo Tommasini, Yehia Abo Sedira, Daniele Dell'Aglio, Marco Balduini, Muhammad Intizar Ali, Danh Le Phuoc, Emanuele Della Valle, and Jean-Paul Calbimonte. 2018. VoCaLS: Vocabulary and Catalog of Linked Streams. In *International Semantic Web Conference (2) (Lecture Notes in Computer Science)*, Vol. 11137. Springer, 256–272.

[37] Riccardo Tommasini and Emanuele Della Valle. 2017. Challenges & Opportunities of RSP-QL Implementations. In *WSP/WOMoCoE@ISWC (CEUR Workshop Proceedings)*, Vol. 1936. CEUR-WS.org, 48–57.

[38] Riccardo Tommasini and Emanuele Della Valle. 2017. Yasper 1.0: Towards an RSP-QL Engine. In *International Semantic Web Conference (Posters, Demos & Industry Tracks) (CEUR Workshop Proceedings)*, Vol. 1963. CEUR-WS.org.

[39] Riccardo Tommasini, Emanuele Della Valle, Marco Balduini, and Daniele Dell'Aglio. 2016. Heaven: A Framework for Systematic Comparative Research Approach for RSP Engines. In *ESWC (Lecture Notes in Computer Science)*, Vol. 9678. Springer, 250–265.

[40] Riccardo Tommasini, Emanuele Della Valle, Andrea Mauri, and Marco Brambilla. 2017. RSPLab: RDF Stream Processing Benchmarking Made Easy. In *International Semantic Web Conference (2) (Lecture Notes in Computer Science)*, Vol. 10588. Springer, 202–209.

[41] Emanuele Della Valle, Stefano Ceri, Frank van Harmelen, and Dieter Fensel. 2009. It's a Streaming World! Reasoning upon Rapidly Changing Information. *IEEE Intelligent Systems* 24, 6 (2009), 83–89.

[42] Emanuele Della Valle, Daniele Dell'Aglio, and Alessandro Margara. 2016. Taming velocity and variety simultaneously in big data with stream reasoning: tutorial. In *DEBS*. ACM, 394–401.

[43] Emanuele Della Valle, Riccardo Tommasini, and Marco Balduini. 2018. Engineering of Web Stream Processing Applications. In *Reasoning Web (Lecture Notes in Computer Science)*, Vol. 11078. Springer, 223–226.

[44] Willem Robert van Hage, Véronique Malaisé, Roxane Segers, Laura Hollink, and Guus Schreiber. 2011. Design and use of the Simple Event Model (SEM). *J. Web Sem.* 9, 2 (2011), 128–136.

[45] Roel Wieringa. 2014. *Design Science Methodology for Information Systems and Software Engineering.* Springer.