Unlocking Semantics as a FAIR Implementation Profile. Reflections and Lessons Learned Building User-centric Semantic Authoring Applications

Selena Baset*, Didier Clement

Hoffmann-La Roche, Grenzacherstrasse 124, 4070 Basel, Switzerland

Abstract

This paper presents insights gained utilizing semantic technologies as a FAIR implementation profile for clinical metadata standards at Roche Pharma Product Development Data Sciences. It highlights the journey undertaken to democratize FAIR data authoring and achieve higher-order FAIRification all while addressing the associated challenges. The paper includes a specific use case from the clinical data collection standards domain, illustrating the automation potential of semantic model-driven development for implementing FAIR user-centric applications. The authors aim to share valuable lessons learned and challenges faced, providing insights for organizations seeking to advance FAIR practices and transform their data landscapes in similar endeavors.

Keywords

Clinical metadata standards, FAIR implementation profile, native RDF authoring tools

1. Introduction

Semantic Web Technologies, with their approach to linked metadata, offer substantial potential as a fit-for-purpose FAIR implementation profile [1] in large life sciences organizations committed to a FAIR data transformation [2]. Yet, in our experience to date, most of the semantic-based approaches to FAIRification are either retrospective, where data traverses a few systems before receiving proper contextual metadata or permanent unique identifiers, or they lack scalability as more often than not, the necessary know-how to create the data FAIR from the start is confined to circles of semantic practitioners. In both cases, semantic FAIRification is perceived as a labor-intensive afterthought, creating a barrier to widespread adoption.

Breaking that barrier and unlocking the potential of semantic technologies as a FAIR implementation profile was an objective we have set for ourselves over the last couple of years as we embarked on a journey of building a semantic-model-driven development framework for clinical metadata standards at Roche Pharma Product Development Data Sciences. We recognize that successful FAIR adoption commences with the leadership mindset embracing the FAIR principles and culminates in a corresponding technological shift. Advocating for the pivotal role of semantic technologies in catalyzing that shift, our focus was on:

SWAT4HCLS'24: 15th International Semantic Web Applications and Tools for Health Care and Life Sciences Conference, February 26-29, 2024, Leiden, The Netherlands

*Corresponding author.

selena.baset@roche.com (S. Baset); didier.clement@roche.com (D. Clement)

- Democratizing the FAIR data authoring process by embedding semantic applications in user-centric operational contexts.
- Aiming for higher-order FAIRification; not merely retrofitting metadata but leveraging semantic inference and automation to proactively incorporate FAIR considerations into the design and development of data systems and applications.

In this paper, we discuss our experience developing authoring applications based on ontologies. These applications are smoothly incorporated into the everyday workflows of clinical standards subject matter experts with limited or no background in semantics. We provide an overview of what we denote by semantic model-driven development, highlighting challenges and specificities. Additionally, using a use case from the clinical data collection standards domain, we demonstrate how the automation capabilities brought by semantic inference facilitate higher-order FAIRification by accelerating the creation of FAIR user-centric applications.

2. The Starting Point: Roche's Metadata Repository

Our starting point was an existing metadata repository known at Roche as the Global Data Standards Repository (GDSR). GDSR is Roche's homegrown multi-tenant metadata repository for clinical data standards, be it the industry-wide standards such as the standards from the Clinical Data Interchange Standards Consortium (CDISC) or any Roche-specific data standards. The metadata standards in GDSR are expressed as RDF knowledge graphs, stored in a triple store, cataloged, versioned and then served to users via a web browser and to applications downstream via a ReST API.

The GDSR system architecture inherently adhered to the FAIR principles in its metadata content [3]. With this as our starting point, achieving FAIRification for our clinical data standards appeared to be a solved problem — or almost so. The end-to-end FAIRification setup was still sub-optimal because the data standards owners were unable to independently maintain the metadata content. They needed an information architect or a knowledge engineer to create and update the underlying knowledge graph representation. With increased adoption and emerging use cases for semantified standard models, the dependency on having semantic expertise to maintain GDSR content evolved into a bottleneck and a dual inefficiency. On the data owners' side, the data standards experts were not empowered to take full control over the data life cycle, and on the semantic expert side, instead of working on more impactful modeling work, a good portion of their time was spent processing change request tickets.

At this point, It was clear that reaping the full FAIRification benefits GDSR offered was predicated on having the right native RDF authoring and governance tools in the hands of its content owners. This is how a key component in the GDSR ecosystem, the GDSR Workbench, was born.

3. Semantic Model-driven Development

Our decision to pursue a model-driven approach stemmed from the need for a systematic and efficient development process for the envisioned GDSR Workbench. One that leverages

automation not only to cover existing data standards models but also to accommodate future ones, ensuring adaptability and scalability.

Similarly to model-driven development in an object-oriented paradigm [4], semantic model-driven development can be seen as the approach to software development that starts with an ontology as a formal domain representation and builds on top of it with a series of semantic artifacts configuration steps to populate, link, manage and expose data as instances of the concepts defined in the ontology and adhering to its constraints. Having adopted this as a methodology when building the GDSR Workbench, we ended up building a framework that we could reuse every time a new FAIRification use case emerged.

The development process (Figure 1) for a typical FAIR authoring application using the GDSR Workbench framework consists of:

1. If the underlying ontology is not in place already, an ontology creation phase where an ontologist works closely with a domain expert on a set of competency-based questions to define the ontology as a formal machine-understandable representation of the domain of interest.
2. An artifact configuration phase that covers:
    - Writing SPARQL queries as a catalog of parameterized named queries. The queries, with metadata annotations, are then used for populating the various user interface views, for quality control and report generation or for downstream consumption.
    - Writing SHACL shapes on top of the ontology classes exposed to the end users. The shapes are used for validating the graph data mutations according to the ontology definitions and for automatically generating user interface views.
    - Adding the necessary URI redirect rules in Roche's identifiers service for domain registry and resource dereferenceability.

From here on, no further development is needed. The remaining parts are left to the GDSR system to put all the pieces together and generate the final authoring and governance tools that correspond to the ontology. We equipped GDSR with the orthogonal system components and inference rules that understood the underlying semantics of the above artifacts and acted as mini language interpreters generating the user interface forms and transactional validation behaviour that mirrored the SHACL shapes.

Under the hood, SHACL shapes are translated into another GDSR data structure, the GDSR Facets, which are used by the ReST API as the actual vehicle between the SHACL-driven front end and the triple store. Facets are some of the earliest foundational pieces of the GDSR system

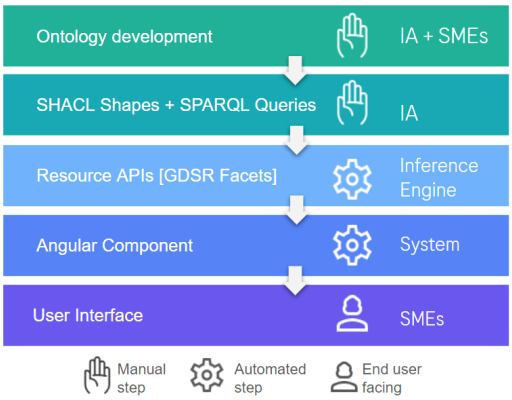

Figure 1: The GDSR Workbench model-driven artifact development process

built at a time when the more standard GraphQL mutations did not exist yet [5]. While facets are still needed by the GDSR core component for executing CRUD operations [1], they have faded into the background and are now automatically inferred from the SHACL layer.

4. Challenges and Proposed Solutions

While adopting a semantic model-driven approach for developing FAIR authoring applications introduced numerous automation possibilities and a systematic means to standardize editing interfaces, it also came with its own set of hurdles and challenges. In the following sections, we highlight some of the noteworthy ones and the solution approach we have taken to tackle them.

4.1. Bridging an expressiveness gap

SHACL shapes provided us with a very good base to describe the different authoring forms and editing constraints an object is subject to but we still encountered some expressiveness limitations when exclusively relying on SHACL constraints semantics [6]. In our case, the SHACL shapes missed the semantics to express some of the user-interface aspects relevant to our application's operational context. For example, how to control the visual layout of the related resources in the current resource view? How to express what properties of the resource are mutable and under which assumptions? Which properties are relevant for validation but should be hidden from the user in the user interface?

To fully automate the user interface generation process starting from the SHACL shapes, more expressive power was needed to complement the shape definition with the additional constraints to dictate aspects such as these in the aforementioned questions. It was for that reason that we defined our own Roche extension to SHACL namespace. An example SHACL shape with some Roche-specific extensions is shown in Listing 1.

```
@prefix schema: <https://id.roche.com/example-domain/schema/> .
@prefix gsh: <https://id.roche.com/gdsr-shacl/> .
# Other standard prefixes are omitted

:ExampleShape
  a sh:NodeShape ;
  sh:targetClass schema:ExampleClass ;
  # a gsh extension to determine if resources of the target class are mutable
  gsh:editable true ;
  # a gsh extension to specifiy the graph in which permutations will be saved
  gsh:targetGraph <https://id.roche.com/example-domain/example-graph> ;
  # a gsh extension prob. to determine if resources should be indexed for full text
      search
  gsh:searchable true;
  # a gsh extension to calculate user-friendly dynamic label expressions
  gsh:labelExpression [gsh:concat([sh:path schema:propA] [sh:path schema:propB])] ;
```

[1] CRUD is a commonly used acronym in software development that stands for Create, Read, Update, Delete.

```
16
17    rdfs:label "Running label to be used in the UI" ;
18    rdfs:comment "A longer description of the resource type to be shown in the UI" ;
19
20    sh:property [
21        a sh:PropertyShape ;
22        sh:name "Property A" ;
23        sh:order 1 ;
24        sh:path schema:propA ;
25        # some properties are hidden from the user in the UI
26        gsh:hidden true ;
27        sh:datatype xsd:string ;
28      ] ;
29    sh:property [
30        a sh:PropertyShape ;
31        sh:name "Property B" ;
32        sh:order 2 ;
33        sh:path schema:propB ;
34        #editibility can be set in a cascading style (property then class)
35        # it can also be resolved dynamically from blank node expressions
36        gsh:editable [  gsh:coalesce ( [ sh:path schema:isExtensible ; ] false ) ; ] ;
37        sh:datatype xsd:string ;
38        #cardinality constraints are applicable in the UI form validation (e.g.
              mandatory fields)
39        sh:minCount 1 ;
40      ] ;
41    sh:property [
42        a sh:PropertyShape ;
43        sh:name "Child property A" ;
44        sh:order 3 ;
45        # collecting children of the current resource via the inverse path
46        sh:path [sh:inversePath schema:childProbA ;] ;
47        sh:node :ChildNodeShape ;
48        #the relationship type dictates the applicable cascade CRUD validations
49        gsh:relation gsh:Aggregation ;
50        #the display mode for childer resources. E.g: tabular details or chip labels
51        gsh:displayMode gsh:DisplayMode.Detaila ;
52      ] ;
53    .
```

Listing 1: An example of a SHACL shape definition with additional Roche-specific extensions

In some cases, the expressiveness gap manifested when the standard SHACL shape definition resulted in a non-deterministic translation of the user transaction into a graph mutation. An example we can give here is the standard `sh:alternativePath` used in SHACL complex property paths. To avoid such non-deterministic scenarios, we equipped the GDSR Workbench engine with the necessary shape validation rules to report faulty shape configurations (in the example here, an alternative path used in a mutable property definition).

4.2. Bridging a procedural gap

Like other languages in the Semantic Web sphere, SHACL is inherently declarative, emphasizing the 'what' over the 'how'. This serves the purpose of knowledge representation and validation perfectly: declaratively describe the problem domain, its entities, their relationships and what constraints apply to a given solution. The systematic procedural logic of how the solution is found is left to OWL reasoners, SPARQL query execution engines, SHACL inference engines, etc. In our case, however, adopting a pure declarative approach would have fallen short of the expectation. In many cases, we were faced with situations where a user action in the interface should trigger a series of mutations to be handled in a transaction where synchronous system-to-system communication was involved. In these scenarios, and many similar ones, we needed to have more imperative control over the how, and when, not just the what.

For that purpose, we introduced the concept of an action shape, a subclass of the standard SHACL shape but with the additional semantics of having an action component attached to it. The procedural logic behind the action component itself can be written using any programming language of choice as long as it is encapsulated and reachable via an HTTP(s) endpoint. A simplified example of action shape to notify a remote system of an event (e.g.: a certain mutation to a resource) is shown in Listing 2 below.

```
@prefix schema: <https://id.roche.com/example-domain/schema/> .
@prefix pec: <https://id.roche.com/gdsr/procedural-extensions-components/> .
@prefix gsh: <https://id.roche.com/gdsr-shacl/> .
# other standard prefixes are omitted

:ExampleShape
  a sh:NodeShape ;
  sh:targetClass schema:Class ;
  gsh:action :NotifyAction;
  # remaining parts of the shape definition are ommitted
.
:NotifyAction
  a gsh:Action ;
  gsh:component pec:APIEndpoint ;
  pec:httpVerb "GET" ;
  pec:param [
      pec:name "event" ;
      pec:value "event x" ;
    ] ;
  pec:param [
      pec:name "resourceURI" ;
      pec:value sh:this ;
    ] ;
  pec:remoteURL "https://remote-system/notify" ;
  rdfs:comment "The comment here can be used to show additional information to the
      user in the UI" ;
  rdfs:label "Notify the remote system that event x has occurred" ;
.
```

Listing 2: An example of an action shape component

4.3. Validating under a half-open world assumption

When we started configuring the authoring interfaces using SHACL shapes, SHACL's closed-world assumption was just what we needed to validate transactions on top of linked data in an operational context. We relied on it to define custom uniqueness constraints and rules applicable for cascade deletion. As we covered more and more configuration scenarios, however, we realized that the closed-world assumption was too restrictive in certain cases where the related linked resource resided in a different "world" such as a neighboring integrated system or an archived version of the metadata. After careful analysis of the different false positives SHACL violations we collected over time, we finally adopted a hybrid approach to validation. By default, the close-world assumption prevails and in the cases where falling back to an open-world assumption was needed, we declared it in the object property in question using a dedicated predicate from Roche's SHACL extension as shown in Listing 3.

```
@prefix schema: <https://id.roche.com/example-domain/schema/> .
@prefix gsh: <https://id.roche.com/gdsr-shacl/> .
# other standard prefixes are omitted

:ShapeWithRemoteLinkedResources
  a sh:NodeShape ;
  sh:targetClass schema:Class ;
  rdfs:label "A shape definition for targets with remote linked resources" ;

  sh:property [
    a sh:PropertyShape ;
    sh:name "Remote resource" ;
    sh:path schema:probA ;
    sh:datatype rdfs:Resource ;
    gsh:relation gsh:Association ;
    gsh:location [gsh:variable :remoteDateset];
  ] ;
  # remaining parts of the shape definition are ommited
  # ...
.
```

Listing 3: An example of a shape definition where the closed-world assumption is amended for remote properties.

The `gsh:location` here tells the workbench engine that the linked resource resides in a remote location. Depending on the relationship type, that remote location can be dynamically resolved at runtime so that the applicable cascade validation rules are enforced.

5. A Clinical Data Standards Application Use Case

To offer a more tangible perspective on the opportunities explored in this paper, we present a use case from the clinical data standards domain. The use case scenario presented below, along with the subsequent solution sections, should illustrate the automation potential of our semantic model-driven development framework and how it helps achieve higher-order FAIRification by accelerating the creation of FAIR user-centric applications.

5.1. An introduction to the use case scenario

Like every step in any clinical trial, data collection is a process that lends itself to harmonization. At Roche, collected data should conform to collection standards that are based on the CDISC-controlled terminologies or their Roche sponsor extension. This means that every variable's permissible value is bound to fall within pre-established enumerated value domains. The controlled terminologies and the underlying schemas can vary depending on whether the variables are collected as part of a questionnaire in an electronic Case Report Form (eCRF) or as one of the multitude of other non-CRF data structures such as lab assays, digital measures, CT-scans, ophthalmologic grading parameters, etc.

The majority Roche's metadata standards are largely aligned with the existing CDISC standards but when it comes to data collection, Roche maintains its own set of global metadata standards that covers a wide range of therapeutic areas and collected data formats[2].

While dictating the permissible values for a given non-CRF variable in isolation from others is readily achievable using the existing standard models, dictating the permissible combinations of values for variables collected together poses a more significant challenge. It requires maintaining Value Level Metadata (VLM) according to schemas that vary dynamically following the different data collection settings and the applicable version of the standards at a given point in time.

5.2. A lost cause for FAIRification?

Given the current scope of Roche's non-CRF standards, populating and maintaining dozens of different VLM schemas seemed infeasible. Before the introduction of the semantic model-driven framework, and despite the FAIR metadata representation of the underlying collection variables and controlled terminologies, maintaining combinations for all VLM schemas in a machine-readable manner required a dedicated solution for each schema. Defining the schemas themselves in a semantic form was relatively straightforward, but the challenge lay in maintaining permissible combinations as populated knowledge graphs. We were faced with the choice of either having to provide the data standards managers with new custom-built authoring tools for each VLM schema; or having to handle the population of all VLM knowledge graphs on our end. Neither option justified the resource investment. Consequently, only three selected VLM schemas were maintained in the metadata repository, while all other schemas were manually kept in a tabular sheet, disconnected from the FAIR source, posing the risk of possibly outdated copies circulating freely and non-compliance issues encountered downstream.

[2]Roche's eCRF and non-CRF standard models were established before the CDASH collection standards counterparts from CDISC were publicly released.

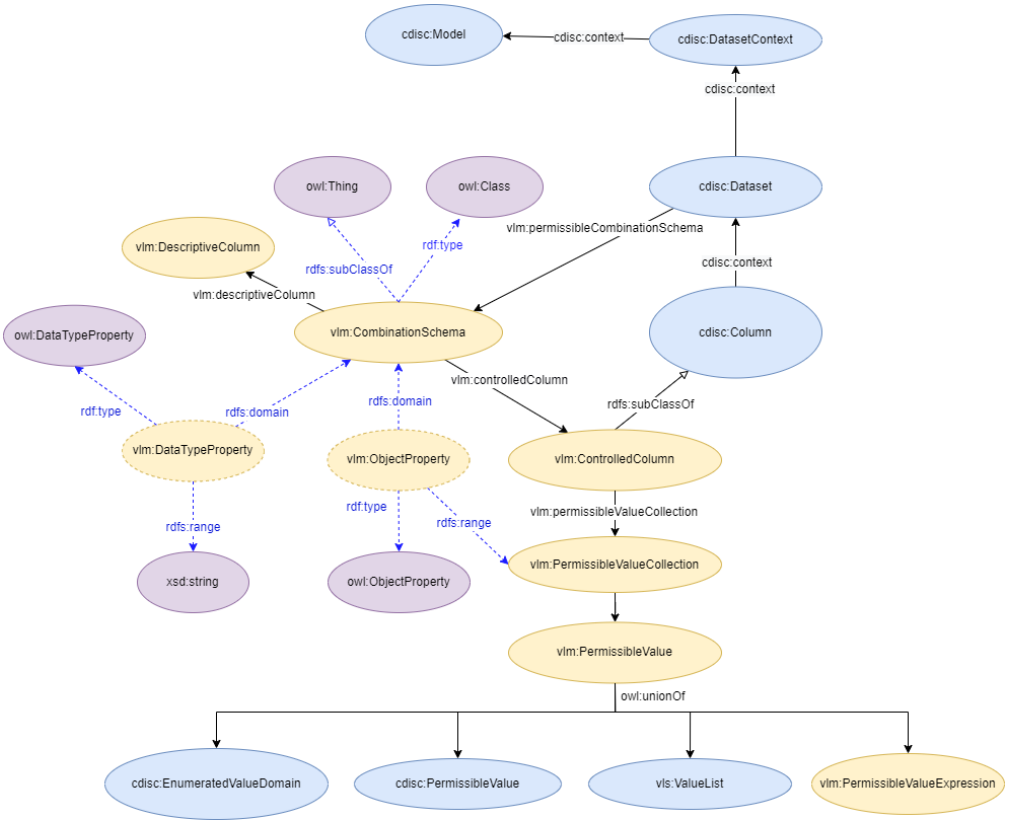

Figure 2: A simplified visualization of the meta VLM ontology

5.3. Restoring the lost FAIRness

Leveraging the semantic model-driven framework we have progressively implemented, and drawing inspiration from concepts in the meta-programming paradigm [7], we felt equipped with the tools needed to automate the solution for constructing the authoring and governance tools for the data standards manager to maintain the VLM combination independently and in a FAIR manner —fully connected to the standard data source. The following subsections outline the different aspects of the implemented solution.

5.3.1. The different levels of modelling abstractions

Right from the start, we took an abstract perspective on the problem of modeling the different non-CRF VLM schemas for different therapeutic areas and non-CRF standard models. Instead of developing a new ontology for different occurrences of the VLM problem, we developed a more abstract meta-VLM ontology. Similar to how a meta-program possesses knowledge of itself and can manipulate itself [7], the meta VLM ontology comprises a small number of stem classes and properties that capture the essence of what constitutes a VLM schema, an ontology that defines a generic schema for other VLM schemas (see the blue dotted connections in the

simplified ontology visualization in Figure 2). We then configured a small workbench user interface on top of this stem to allow the standards data manager to provide the input necessary to automatically create new VLM schemas.

5.3.2. An automated implementation process

Every time the data manager declared a new VLM schema in the user interface, the VLM ontology automatically expanded with the help of semantic inference, creating new classes and properties. For these classes and properties just-created, a few more inference rules automated the artifacts generation process including the SPARQL and SHACL required for creating the GDSR editor user interfaces. The resulting end-to-end automated pipeline for the artifact development process is shown in Figure 3.

To address the legacy VLM combinations maintained in tabular sheets, we implemented a separate dedicated parser component that dynamically understood the VLM schemas created. This component performed a series of reverse look-up operations to the metadata repository, translating from free-text values into the URIs of the linked entities.

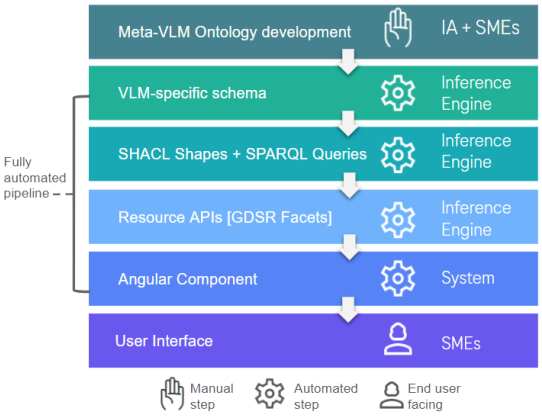

Figure 3: The end-to-end automated pipeline to create VLM authoring interfaces using the GDSR Workbench framework

5.3.3. Restored metadata lineage and eliminated risks

Having undergone the automated pipeline outlined above, the legacy tabular content is now fully converted to machine-readable metadata. The resulting knowledge graph is appended to the overarching clinical standards knowledge graph in GDSR where the semantic link to the non-CRF standard variables and controlled terminologies is restored. This not only enforces data integrity and opens new possibilities for standards compliance assessment but also serves as a significant efficiency booster and time-saver for the data standards manager maintaining the VLM combinations. It practically eliminates the need to manually propagate changes to the legacy tabular VLM sheet following an update to the standard source, thereby eliminating the risk of the VLM not being aligned with the metadata repository.

6. Related Work

The insights presented in this paper were obtained working hands-on in the specific context of Roche's metadata repository and its satellite projects. Rather than being research-oriented in nature, our efforts were geared toward tackling the hurdles and bottlenecks faced scaling our

metadata FAIRification initiatives across the different areas of Roche's clinical trial development. That being said, we consider our work to be largely aligned with the viewpoint of the research community and that of our industry peers; especially those underlining the importance of adhering to the FAIR principles and the role Semantic Technologies play in there [2] [1] [8].

On a more technical note, a thorough review of the literature is yet to be conducted but to our knowledge, most of the work on augmenting SHACL shapes with procedural logic is attributed to Knublauch et al. at TopQuadrant, Inc [9]. The approach they take is based on JavaScript extensions to the W3C SHACL standards. The full specifications of the proposed extension mechanism are accessible under the corresponding W3C Working Group notes.

7. Summary

This paper documented our efforts to harness semantic technologies as a FAIR implementation profile for metadata standards applications in life sciences organizations. We stressed the importance of proactively incorporating FAIR considerations into the design and development of the FAIR data systems. Using a real-world use case scenario, we illustrated how we leveraged automation to build native FAIR authoring tools seamlessly embedded in user-centric contexts. By sharing the challenges we encountered, we aim to provide some insights for other organizations pursuing similar efforts to advance FAIR practices and transform their data landscapes.

8. A Note on Reproducibility

We appreciate the lack of open access to the work behind the insights shared in this paper. Open-sourcing a selection of our standard ontologies and SHACL namespace extension is potentially possible but pending a feasibility assessment due to known inter-dependencies with other components of the GDSR ecosystem. Nonetheless, a more elaborate exchange of ideas and experiences with the community remains a privilege we are always after and we encourage interested readers to reach out via email with any feedback or questions.

Acknowledgments

The insights shared in this paper are the outcome of the long and dedicated efforts of a team comprising semantic experts, information architects, software and QA engineers. We express our sincere gratitude to all those who have contributed to and supported this endeavor. In particular, we'd like to thank the following key contributors for adding to the GDSR story over the years:

Adam Kozac , Celso Pereira , Huw Mason , Ivan Robinson , Jacek Caban , Javier Fernandez , Marcelina Kasprusz, Michal Kalinowski , Michal Pietrusinki , Nelia Lasierra and Szymon Tyka.

We equally appreciate the great support of our sponsors and the continued fruitful collaboration with our key stakeholders. In particular, we'd like to call out the whole Data Standards and Governance Group at Roche Pharma division.

References

[1] E. Schultes, B. Magagna, K. M. Hettne, R. Pergl, M. Suchánek, T. Kuhn, Reusable fair implementation profiles as accelerators of fair convergence, in: G. Grossmann, S. Ram (Eds.), Advances in Conceptual Modeling, Springer International Publishing, Cham, 2020, pp. 138–147.

[2] J. Wise, A. G. de Barron, A. Splendiani, B. Balali-Mood, D. Vasant, E. Little, G. Mellino, I. Harrow, I. Smith, J. Taubert, K. van Bochove, M. Romacker, P. Walgemoed, R. C. Jimenez, R. Winnenburg, T. Plasterer, V. Gupta, V. Hedley, Implementation and relevance of fair data principles in biopharmaceutical rd, Drug Discovery Today 24 (2019) 933–938. URL: https://www.sciencedirect.com/science/article/pii/S1359644618303039. doi:https://doi.org/10.1016/j.drudis.2019.01.008.

[3] J. D. Fernández, N. Lasierra, D. Clement, H. Mason, I. Robinson, Enabling fair clinical data standards with linked data, in: The Semantic Web: ESWC 2020 Satellite Events: ESWC 2020 Satellite Events, Heraklion, Crete, Greece, May 31–June 4, 2020, Revised Selected Papers 17, Springer, 2020, pp. 303–306.

[4] B. Hailpern, P. Tarr, Model-driven development: The good, the bad, and the ugly, IBM Systems Journal 45 (2006) 451–461. doi:10.1147/sj.453.0451.

[5] G. Brito, T. Mombach, M. T. Valente, Migrating to graphql: A practical assessment, in: 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2019, pp. 140–150. doi:10.1109/SANER.2019.8667986.

[6] J. Corman, J. L. Reutter, O. Savković, Semantics and validation of recursive shacl, in: D. Vrandečić, K. Bontcheva, M. C. Suárez-Figueroa, V. Presutti, I. Celino, M. Sabou, L.-A. Kaffee, E. Simperl (Eds.), The Semantic Web – ISWC 2018, Springer International Publishing, Cham, 2018, pp. 318–336.

[7] P. Hill, J. Gallagher, Meta-programming in logic programming, Handbook of logic in artificial intelligence and logic programming 5 (1998) 421–497.

[8] A. Jacobsen, R. de Miranda Azevedo, N. Juty, D. Batista, S. Coles, R. Cornet, M. Courtot, M. Crosas, M. Dumontier, C. T. Evelo, et al., Fair principles: interpretations and implementation considerations, 2020.

[9] H. Knublauch, P. Maria, Shacl javascript extensions. working group note, World Wide Web Consortium (W3C) (2017).

A. Online Resources

- CDISC Standards
- Pistoia Alliance
- SHACL JavaScript Extensions - W3C Working Group Notes 08 June 2017
- PHUSE EU Connect 2023 - Conference Proceedings