Knowledge multiplies when shared — when calling things by their right name: improving the validation and exchange of genetic data in research and diagnostics

Fokkema, I.F.A.C.

CHAPTER 9

Direct data exchange of complex genetic datasets; the VarioML framework, an update

Ivo F.A.C. Fokkema[1], Juha Muilu.

1 - Department of Human Genetics, Leiden University Medical Center, Leiden, the Netherlands.
2 - Biocomputing Platforms Ltd, Espoo, Finland.

9.1 Abstract

As genetic data sharing becomes increasingly complex, standardized and interoperable data formats are essential. To align with modern data exchange practices, the VarioML framework, originally designed based on an XML-based format for complex genetic datasets, has been extended with a native JSON implementation. This update enhances interoperability, allowing seamless integration with modern bioinformatics tools and automated submissions to the LOVD3 database. JSON Schema definitions were developed to ensure data validation. Unlike alternative formats such as Phenopackets, VarioML maintains a flexible, ontology-driven structure suitable for both patient-centered and variant-centered representations. This update ensures that VarioML remains a robust solution for LSDB interoperability and future genetic data exchange. By implementing VarioML JSON in our APIs, we take steps towards integrating the FAIR principles in LOVD.

9.2 Introduction

Gene variant databases, or Locus-Specific Databases (LSDBs), and related resources are increasingly interconnected through automated data exchange mechanisms. These connections may link distributed systems for bi-directional exchange[1] or facilitate federated queries via a central hub.[2] However, most of these integrations focus solely on DNA variant sharing, relying on simple data formats. Given the complexity of genetic datasets in LSDBs, including case-level patient and phenotype data, a more structured format is necessary to take full advantage of this information. Such a format would not only allow automated submissions to databases like the Leiden Open Variation Database (LOVD), but also improve data interoperability for solving increasingly complex genetic cases where automated access to highly detailed case-level data is required.

In 2012, we introduced the VarioML framework,[3] a flexible XML-based data format designed to store and exchange complex LSDB datasets. VarioML was structured to support both variant-centered and LSDB-centered views, and was adopted in Cafe Variome,[4] Molgenis,[5] and LOVD2.[6] A key feature of VarioML is its ontology-driven standardization, which enables seamless data exchange across systems using differing internal representations.

Since the design of VarioML, XML has been largely replaced by JavaScript Object Notation (JSON), a more widely adopted programming language-agnostic format for structured data.[7] While the original VarioML framework included an XML-to-JSON conversion method, this process was only partially automated and still required XML as the primary format. To modernize VarioML and support direct JSON implementation, we developed a native JSON version of VarioML.

Here, we describe the design and implementation of VarioML in JSON, its integration into LOVD3 for automated database submissions, and the development of JSON Schema definition files to enable fully automated validation of VarioML JSON datasets. The resulting toolkit provides a modern, flexible approach to exchanging and validating complex LSDB datasets.

9.3 Materials and Methods

The primary structural difference between XML and JSON lies in how they store and organize data. While both formats support single values, lists, and nested elements, XML also allows the use of attributes, which do not exist in JSON. As a result, converting XML to JSON requires attributes to be mapped into JSON keys. Additionally, when an XML element contains both attributes and a textual value, a special key must be introduced in JSON to store that value. Another key distinction is that XML permits direct repetition of elements, whereas JSON

```
<seq_changes>
    <variant type="cDNA">
        <gene source="HGNC" accession="IVD" />
        <ref_seq source="genbank" accession="NM_002225.3" />
        <name scheme="HGVS">
            c.97C>T
        </name>
    </variant>
</seq_changes>
```

Figure 9.1: **An example of an XML format with attributes, a textual value, and a repeatable element.** The attributes in this example are the "type" of the "variant" element, the "source" and "accession" of the "gene" element, the "source" and "accession" of the "ref_seq" element, and the "scheme" of the "name" element. Attributes do not exist in the JSON format. The name's value, "c.97C>T", would normally be stored under the key "name" in JSON. However, since the "name" element has a "scheme" attribute, the JSON format will require a different method to store the value. Note that the variant element is repeatable; any "seq_changes" element can contain multiple "variant" elements. This is possible in XML but not in JSON, as JSON keys have to be unique.

does not allow duplicate keys, requiring an additional level of nesting to handle repeatable elements. An example of an XML representation of LOVD data is shown in Figure 9.1.

During the development of the JSON-native version of VarioML, we based our design on a VarioML data file representing the LOVD3 data model for a full database submission. Several key modifications were made compared to the XML approach. Elements that are allowed to occur multiple times, such as the "variant" element, were renamed using plural nouns (e.g., "variants") and structured as JSON arrays to align with standard JSON best practices. XML attributes were converted into key-value pairs within their respective elements, with special care taken to prevent naming conflicts. When an XML element contained both attributes and a textual value, the value was explicitly stored under a key named "value" rather than "string" as described in the original VarioML design, as not all values are strings, making "value" a more generic and widely applicable alternative. The JSON format for the data displayed in Figure 9.1 is shown in Figure 9.2.

Once the JSON format for a full case-level submission was finalized, it was implemented in the LOVD3 submission API, allowing fully automated data submissions from European diagnostic laboratories. Following this, the variant-centered VarioML JSON format was developed and implemented in a new LOVD3 API that enables registered LOVD instances to exchange data with the central LOVD server and generate bulk data downloads in VarioML JSON format. To ensure compliance with the VarioML standard and enable automated validation, JSON

```
{
    "seq_changes": {
        "variants": [
            {
                "type": "cDNA",
                "gene": {
                    "source": "HGNC",
                    "accession": "IVD"
                },
                "ref_seq": {
                    "source": "genbank",
                    "accession": "NM_002225.3"
                },
                "name": {
                    "scheme": "HGVS",
                    "value": "c.97C>T"
                }
            }
        ]
    }
}
```

Figure 9.2: **The JSON output for the data displayed in Figure 9.1.** The "variant" element is repeatable, so was renamed to "variants" and implemented as an array.

Schema definition files were created based on the existing XML Schema: `LSDB.json`, which defines all supported fields and their values and contains the structure of a complete LSDB submission, and `variant.json`, which defines the structure for variant-centered data while referencing `LSDB.json` for field definitions. Additionally, `GA4GH.json` was developed as a JSON Schema for a GA4GH Data Connect API endpoint that returns VarioML data.

All schema files are publicly available at the GitHub repository: `github.com/VarioML/VarioML`.

9.4 Results and Discussion

JSON has become the most widely used data exchange format due to its built-in support across all major programming languages and the availability of extensive online documentation, tools, and validators. By converting VarioML to a JSON-native format, we significantly improved its interoperability with modern bioinformatics tools and infrastructure. The

resulting format is highly flexible, supporting both patient-centered and variant-centered implementations, and has been integrated into LOVD3 to enable fully automated submissions of case-level datasets as well as seamless data exchange between registered LOVD instances and the central server.

Initially, to determine the best XML-to-JSON conversion method, we tested six different XML-to-JSON tools, generating five distinct JSON outputs. Each JSON result was then converted back to XML to assess reversibility, which succeeded in only one case. The tested tools encountered problems converting repeatable elements, element attributes, and text values in elements containing attributes. None of the tested tools based their conversion on the XML schema, resulting in inconsistent conversions of repeatable elements. Based on these findings, we opted to design a native JSON format prioritizing readability and usability over direct XML reversibility. The subsequent required renaming of repeatable elements (e.g., "variant" to "variants") aids developers in adopting the format, as it is immediately clear that the data format allows for multiple variants.

During the development of this JSON implementation, the Global Alliance for Genomics and Health (GA4GH) introduced Phenopackets, a data exchange format designed for complex patient-centered data structures.[8] Phenopackets has several limitations compared to VarioML. It is strictly a patient-centered format and does not support variant-centered representations, limiting its broader applicability. Additionally, its model includes oversimplifications that reduce its usefulness for resources requiring a higher level of detail. For instance, genetic data in Phenopackets must always be linked to a diagnosis, preventing the storage of genetic information for undiagnosed patients or healthy individuals. Furthermore, it lacks an internal structure to explicitly link variant expressions across DNA, RNA, and protein levels, making it impossible to indicate which descriptions correspond to the same underlying genomic change. While some degree of simplification is necessary when designing data models, every abstraction inherently reduces flexibility. These limitations in Phenopackets not only prevent proper conversion of existing data into its format but also hinder accurate reconstruction of stored data. To address similar challenges in VarioML while maintaining adaptability, the VarioML model implements the "evidence_code" element, allowing users to store additional observational data when needed. This ensures that critical information can still be captured, even if it does not fit directly within the predefined model, and helps maintain full bidirectional compatibility between different data representations.

A major advantage of VarioML is its strong reliance on ontologies, which ensures that all data fields and values are fully standardized. This makes VarioML datasets easily convertible into FAIR-compliant formats (Findable, Accessible, Interoperable, and Reusable). The FAIR principles aim to maximize data interoperability, particularly for federated queries across

distributed systems.[9] Since FAIR-compliant datasets can be stored as JSON-LD (JSON for Linked Data), native VarioML JSON files can be transformed into FAIR-compliant files requiring relatively little effort. By implementing VarioML within LOVD APIs, we have taken an important step toward full integration of the FAIR principles in genetic variant databases.

Although the current JSON implementation of VarioML is primarily designed for LOVD3-based datasets, the VarioML framework was developed for broader applicability. To support adoption in the wider scientific community, we plan to extend the JSON version to fully capture the complexity of the original VarioML framework. Additionally, we aim to improve documentation and develop an online validation tool, allowing researchers and developers to verify compliance of both data files and API endpoints with the VarioML standard.

Examples, JSON Schema files, and the original XML implementation are available in the VarioML GitHub repository: `github.com/VarioML/VarioML`.

9.5 References

[1] Ivo F.A.C. Fokkema, Mark Kroon, Julia A. López Hernández, Daan Asscheman, Ivar Lugtenburg, Jerry Hoogenboom, and Johan T. den Dunnen. *The LOVD3 platform: efficient genome-wide sharing of genetic variants*. European Journal of Human Genetics 2021; 29 (12) 1796–1803.

[2] Global Alliance for Genomics and Health. *A federated ecosystem for sharing genomic, clinical data*. Science 2016; 352 (6291) 1278–1280.

[3] Myles Byrne, Ivo F.A.C. Fokkema, Owen Lancaster, Tomasz Adamusiak, Anni Ahonen-Bishopp, David Atlan, Christophe Béroud, Michael Cornell, Raymond Dalgleish, Andrew Devereau, George P. Patrinos, Morris A. Swertz, Peter E.M. Taschner, Gudmundur A. Thorisson, Mauno Vihinen, Anthony J. Brookes, and Juha Muilu. *VarioML framework for comprehensive variation data representation and exchange*. BMC Bioinformatics 2012; 13 (1) 254.

[4] Owen Lancaster, Tim Beck, David Atlan, Morris Swertz, Dhiwagaran Thangavelu, Colin Veal, Raymond Dalgleish, and Anthony J. Brookes. *Cafe Variome: General-Purpose Software for Making Genotype-Phenotype Data Discoverable in Restricted or Open Access Contexts*. Human Mutation 2015; 36 (10) 957–964.

[5] K. Joeri van der Velde, Floris Imhann, Bart Charbon, Chao Pang, David van Enckevort, Mariska Slofstra, Ruggero Barbieri, Rudi Alberts, Dennis Hendriksen, Fleur Kelpin, Mark de Haan, Tommy de Boer, Sido Haakma, Connor Stroomberg, Salome Scholtens, Gert-Jan van de Geijn, Eleonora A.M. Festen, Rinse K. Weersma, and Morris A. Swertz. *MOLGENIS Research: Advanced bioinformatics data software for non-bioinformaticians*. Bioinformatics 2019; 35 (6) 1076–1078.

[6] Ivo F.A.C. Fokkema, Peter E.M. Taschner, Gerard C.P. Schaafsma, J. Celli, Jeroen F.J. Laros, and Johan T. den Dunnen. *LOVD v.2.0: the next generation in gene variant databases*. Human Mutation 2011; 32 (5) 557–563.

[7] Jiwen Xin, Cyrus Afrasiabi, Sebastien Lelong, Julee Adesara, Ginger Tsueng, Andrew I. Su, and Chunlei Wu. *Cross-linking BioThings APIs through JSON-LD to facilitate knowledge exploration*. BMC Bioinformatics 2018; 19 (1).

[8] Julius O.B. Jacobsen, Michael Baudis, Gareth S. Baynam, Jacques S. Beckmann, Sergi Beltran, Orion J. Buske, Tiffany J. Callahan, Christopher G. Chute, Mélanie Courtot, Daniel Danis, Olivier Elemento, Andrea Essenwanger, Robert R. Freimuth, Michael A. Gargano, Tudor Groza, Ada Hamosh, Nomi L. Harris, Rajaram Kaliyaperumal, Kevin C.Kent Lloyd, Aly Khalifa, Peter M. Krawitz, Sebastian Köhler, Brian J. Laraway, Heikki Lehväslaiho, Leslie Matalonga, et al. *The GA4GH Phenopacket schema defines a computable representation of clinical data*. Nature Biotechnology 2022; 40 (6) 817–820.

[9] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J.G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, et al. *The FAIR Guiding Principles for scientific data management and stewardship*. Scientific Data 2016; 3.