GIMA Geographical Information Management and Applications

Data Lineage in the RVB

P.M.E. Gottenbos

TU Delft, Utrecht University, Twente University, Wageningen University

Rijksvastgoedbedrijf
Ministerie van Binnenlandse Zaken en Koninkrijksrelaties
Rijksvastgoedbedrijf voor NL

Universiteit Utrecht TU Delft UNIVERSITY OF TWENTE. WAGENINGEN UR

Table of contents

List of Abbreviations

Abbreviation	Meaning
API	Application programming interface
CRS	Coordinate reference system
CSV	Comma-separated values
DAG	Directed acyclic graph
ETL	Extract, transform, load
EIF	European Interoperability Framework
FME	Feature Manipulation Engine
FMW	FME workspace file
GIS	Geographic information system
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
LE_Algorithm	Lineage extension algorithm
LE_Processing	Lineage extension processing metadata
LI_ProcessStep	Lineage process step (ISO 19115 element)
LI_Source	Lineage source (ISO 19115 element)
OGC	Open Geospatial Consortium
OWL-S	Ontology Web Language for Services
RDF	Resource Description Framework
RVB	Rijksvastgoedbedrijf
SRID	Spatial reference identifier
UI	User interface
UML	Unified Modeling Language
W3C PROV	W3C Provenance data model
WPS	Web Processing Service
XML	Extensible Markup Language

Abstract

The Rijksvastgoedbedrijf (RVB), the Dutch national real estate agency, relies on spatial Extract–Transform–Load (ETL) workflows, in particular those built with the Feature Manipulation Engine (FME), to manage complex geospatial data. However, it currently lacks a standards-compliant, automated method to track data lineage throughout these workflows. This gap hampers traceability, reproducibility, and compliance with data governance standards, thereby impeding effective spatial data management and informed decision-making.

To address this challenge, a data lineage model tailored to FME-based spatial ETL workflows was developed. The model integrates the ISO 19115 geospatial metadata standard for lineage and leverages GeoSPARQL to define a dictionary of commonly used spatial transformations. Implementation of the model utilizes custom Python-based parsers to automatically extract lineage information from FME workspaces and their runtime log files. These parsers translate FME transformer steps into standardized provenance records, which are stored in JSON format and visualized through a Flask-based web interface. The system captures both spatial- and attribute-level operations, including geometry calculations, coordinate reprojections, and spatial filtering operations.

Evaluation on real-world RVB workflows and an evaluation done according to ISO 19157 confirms that the proposed approach effectively enhances traceability, auditability, and the semantic interpretability of spatial ETL processes. Key limitations were identified, such as partial coverage of certain FME transformer types and the lack of versioned lineage history. Despite these limitations, the solution significantly improves transparency and reproducibility in spatial data governance practices. It thereby provides a scalable foundation for integrating robust provenance tracking into operational geospatial infrastructure.

Keywords: *Geospatial ETL, Data Lineage, FME Workflows, ISO 19115, GeoSPARQL, Spatial Data Governance*.

1. Introduction

Data lineage also known as data provenance is the documented history of a dataset: where it came from, how it was transformed, and by whom and when those transformations occurred (Tang et al., 2019). In practice, data lineage answers five questions about any output layer: what changed, where, how, who, and when (Closa et al., 2019). Because organizations use this trace to prove quality, ownership, and regulatory compliance, lineage sits within the broader practice of data governance—the policies, standards, and operating routines that keep data trustworthy and accountable across its lifecycle (Jamedzija et al., 2021). Together, data lineage and governance make complex data work auditable and reproducible.

Geospatial data intensifies these requirements. Government agencies routinely integrate cadastral boundaries, building footprints, infrastructure layers, and policy zones, all of which evolve over time and must remain spatially consistent. Typical processing involves coordinate reference system transformations, spatial joins, topological edits, and geometry calculations. Each of these transformations adds semantic and geometric complexity that is poorly captured by lineage approaches designed for tabular data. As a result, organizations need lineage that can describe not only which Table joined which, but also which geometry operations occurred on what features, in which CRS, and with what parameters.

Within the Rijksvastgoedbedrijf (RVB), the Dutch national real estate agency, spatial ETL is a core operational capability. RVB maintains diverse spatial datasets and automates frequent updates to keep them authoritative and fit for purpose. Safe Software's Feature Manipulation Engine (FME) is central to this practice (Breunig et al., 2020). FME's transformer-based model allows complex geoprocessing chains to be assembled visually and run reliably at scale. However, while FME excels at executing spatial workflows, it does not natively export those workflows' transformation histories as structured, standards-compliant lineage records. Provenance is present implicitly—in workbench diagrams and execution logs—but not captured in a form that supports consistent governance, cross-system integration, or independent verification.

As the RVB is a government company, the RVB depends on transparent and well-documented data workflows to uphold their data quality, validate historical decisions, and meet compliance requirements (e.g., satisfying metadata standards like ISO 19115 for geospatial information). However, as their ETL workflows grow in complexity and involve multiple stakeholders, the absence of a formalized lineage-tracking mechanism increases the risk of errors, slows problem resolution, and compromises trust in spatial data products. This gap is especially critical when spatial data is shared externally. For example, with municipalities, contractors, or regulatory authorities, where provenance information is essential for defensible and auditable decisions.

This thesis responds to that gap by designing and evaluating a spatially aware, standards-aligned lineage model tailored to FME-based workflows. The remainder of this opening section explains why spatial ETL is different from general ETL (Figure 1.1), situates FME in RVB's operational reality, and then states the problem this research addresses.

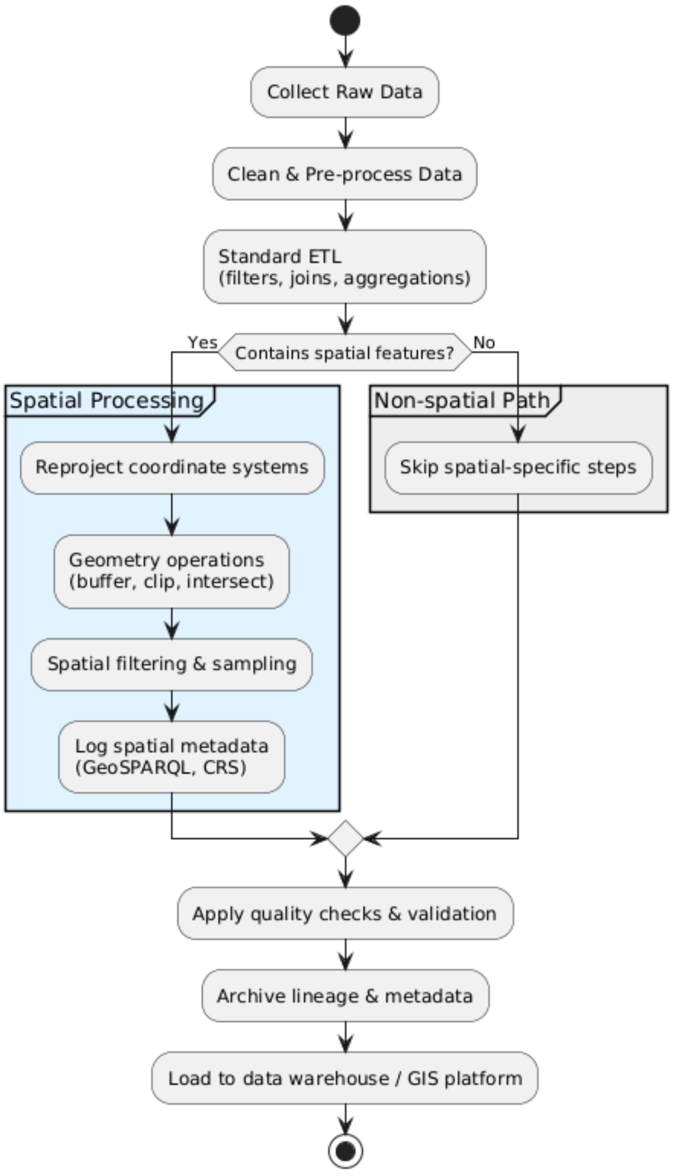

Collect Raw Data
Clean & Pre-process Data
Standard ETL (filters, joins, aggregations)
Contains spatial features?
Yes
No
Spatial Processing
Reproject coordinate systems
Geometry operations (buffer, clip, intersect)
Spatial filtering & sampling
Log spatial metadata (GeoSPARQL, CRS)
Non-spatial Path
Skip spatial-specific steps
Apply quality checks & validation
Archive lineage & metadata
Load to data warehouse / GIS platform

Figure 1.1 - Standard ETL pipeline vs spatial ETL pipeline

1.1. The Rise of Spatial ETL

ETL refers to the extraction of data from source systems, the transformation of that data according to business and quality rules, and the loading of the results into target stores; in spatial contexts, geometry operations and coordinate handling are common transformation steps (Kimball & Ross, 2013). An ETL tool is the engine that takes raw input from many places, applies an ordered chain of processing steps, and then writes the combined data to a centralized system such as a data warehouse (Kumaran, 2021). ETL is divided into three main phases: the first is extracting data from varying sources. This can include databases, flat files, APIs etc. This extracted data is then transformed into a standard format, making sure that the data can be used. Lastly this data is then loaded into a target storage system such as a data warehouse, which means that the data can now be queried and used in reports and analysis.

ETL tools are essential for creating valuable insights through data integration however standard ETL tools are not equipped to handle geospatial transformations (Chee, 2024). This is because these spatial transformations introduce additional complexities, such as geometry handling and coordinate reprojections (Di et al., 2013). To address these complexities, specialized Spatial ETL tools such as FME, GeoKettle, and ArcGIS Data Interoperability have emerged since the late 20th century, continually evolving to handle increasingly sophisticated geospatial requirements. The complexity of spatial transformations, such as reprojections and spatial joins, is clearly illustrated in Figure 1.1, comparing the workflow steps of standard ETL pipelines with those of spatial ETL processes.

- Coordinate System Reprojections: Converting spatial data between different coordinate reference systems (CRS) to ensure compatibility across datasets.
- Spatial Joins and Merging: Spatial joins and merging operations uniquely consider geometric intersections and spatial proximity, unlike traditional joins based solely on database attributes.
- Topological Edits: Performing geometry-based operations such as buffering, clipping, and intersection analysis (Viera, 2024).

These operations introduce dependencies between geometry, attributes, and spatial logic that are not adequately captured by conventional lineage models, which primarily focus on schema or record-level transformations.

Although spatial ETL tools provide powerful functionalities, a significant gap remains in lineage tracking which is the ability to trace and document how data has been transformed and integrated. Without robust lineage tracking, organizations face risks in verifying data provenance, undermining confidence in data-driven decisions, especially in fields requiring regulatory compliance or environmental assessments (Closa et al., 2019). This presents a challenge for organizations that require auditability and traceability in their workflows. Without a mechanism to capture transformation history, it becomes difficult to verify how datasets were created or which operations contributed to a given output (Dai et al., 2017). An example of an FME workbench can be seen in Figure 1.2 and an example of a geospatial transformation can be seen in Figure 1.3.

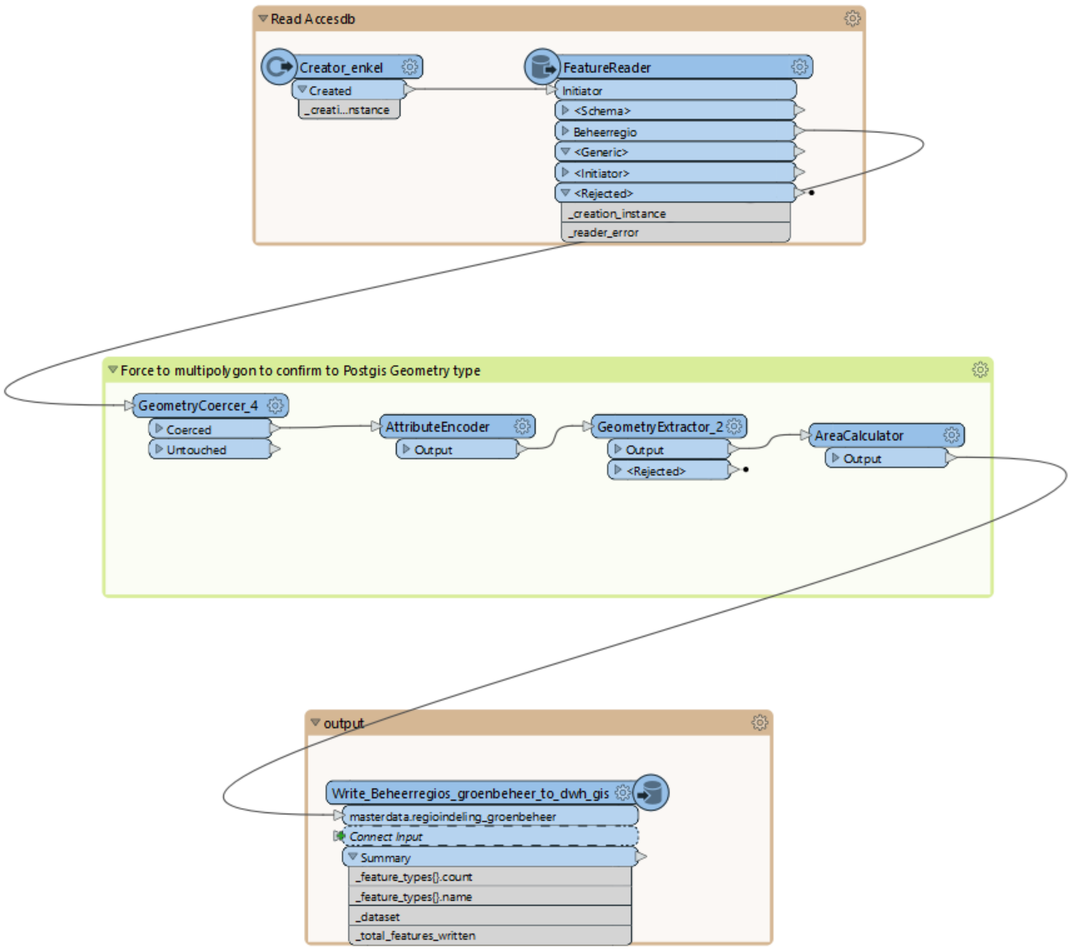
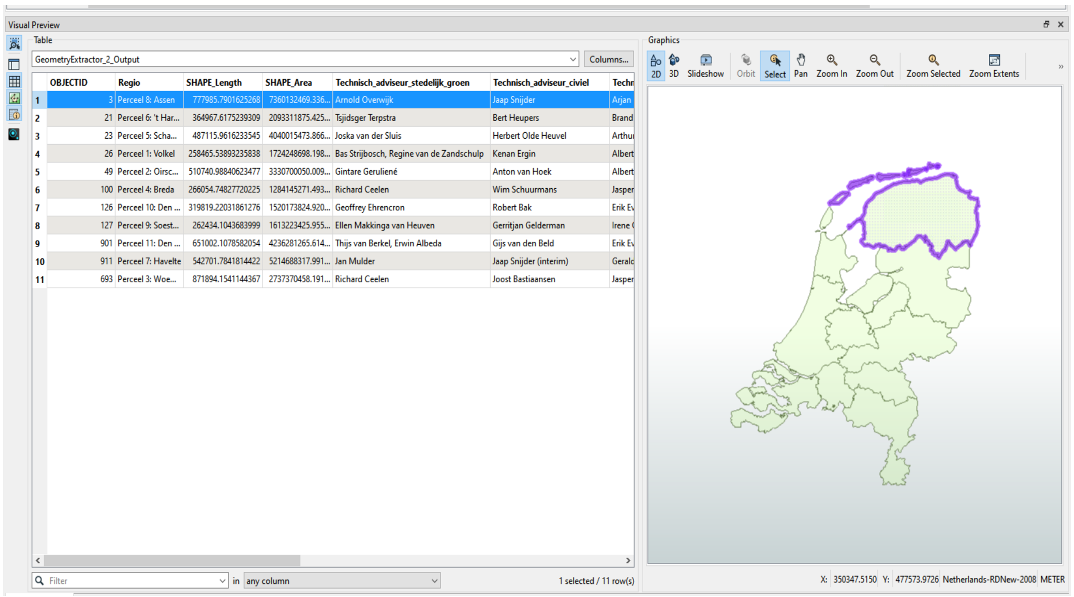


Figure 1.2 - Geospatial processes done in FME



GeometryExtractor_2_Output

	OBJECTID	Regio	SHAPE_Length	SHAPE_Area	Technisch_adviseur_stedelijk_groen	Technisch_adviseur_civiel	Techn
1	3	Perceel 8: Assen	777985.7901625268	7360132469.336...	Arnold Overwijk	Jaap Snijder	Arjan
2	21	Perceel 6: 't Har...	364967.6175239309	2093311875.425...	Tsjidsger Terpstra	Bert Heupers	Brand
3	23	Perceel 5: Scha...	487115.9616233545	4040015473.866...	Joska van der Sluis	Herbert Olde Heuvel	Arthu
4	26	Perceel 1: Volkel	258465.53893235838	1724248698.198...	Bas Strijbosch, Regine van de Zandschulp	Kenan Ergin	Albert
5	49	Perceel 2: Oirsc...	510740.98840623477	3330700050.009...	Gintare Gerulienė	Anton van Hoek	Albert
6	100	Perceel 4: Breda	266054.74827720225	1284145271.493...	Richard Ceelen	Wim Schuurmans	Jasper
7	126	Perceel 10: Den ...	319819.22031861276	1520173824.920...	Geoffrey Ehrencron	Robert Bak	Erik Ev
8	127	Perceel 9: Soest...	262434.1043683999	1613223425.955...	Ellen Makkinga van Heuven	Gerritjan Gelderman	Irene
9	901	Perceel 11: Den ...	651002.1078582054	4236281265.614...	Thijs van Berkel, Erwin Albeda	Gijs van den Beld	Erik Ev
10	911	Perceel 7: Havelte	542701.7841814422	5214688317.991...	Jan Mulder	Jaap Snijder (interim)	Gerald
11	693	Perceel 3: Woe...	871894.1541144367	2737370458.191...	Richard Ceelen	Joost Bastiaansen	Jasper

1 selected / 11 row(s)

X: 350347.5150 Y: 477573.9726 Netherlands-RDNew-2008 METER

Figure 1.3 – Example of a geospatial transformation done in FME

Given the increasing need for accountability in data-driven decision-making, the ability to track lineage in Spatial ETL processes is becoming a necessity rather than an optional feature (Closa et al., 2021). Addressing the lineage tracking gap will enable organizations, such as RVB, to improve data governance by clearly documenting data transformations and origins, thereby enhancing transparency in workflows, accuracy in spatial analyses, and accountability in audit processes and regulatory compliance.

1.2. Problem statement

Despite the increasing adoption of spatial ETL tools to manage complex geospatial workflows, current systems lack integrated mechanisms to capture and represent data lineage in a structured, interoperable manner. This limitation is particularly evident in tools such as FME, which, while technically advanced in handling spatial transformations, do not natively support systematic documentation of transformation logic, data flow, or responsibility tracking. The absence of such functionality represents both a technical shortcoming in operational tools and a scholarly gap in geospatial data lineage research**.**

From a technical standpoint, FME's workflow design is inherently visual and execution-driven, with lineage implicitly encoded in the structure of the workspace. Although this visual model supports user understanding during development, it does not produce machine-readable outputs that capture transformation steps in formats aligned with governance standards (e.g., ISO 19115). Furthermore, changes to spatial attributes and geometries—such as reprojections, spatial joins, or topological edits—are not exported in a reusable lineage log. This hinders auditability, reproducibility, and long-term traceability of spatial datasets.

Several studies have explored lineage tracking for ETL workflows, but their applicability to spatial data remains limited. For example, Dai (2017) proposes lineage tracking using minimal attribute sets, which are the smallest collection of attributes needed to effectively track data transformations. This approach suits basic tabular processes but does not scale adequately to complex spatial transformations such as reprojections, spatial joins, or geometric edits.

Similarly, Tang (2019) examines lineage tracking methods within large-scale ETL environments, but the scope is explicitly restricted to non-spatial data structures, neglecting spatial logic and geometry handling complexities inherent to tools like FME.

Building on these efforts, Jamedzija (2021) introduces a push-based API to support real-time lineage capture, with a focus on data ownership and access control. In this context, data processes and data ownership refer to assigning responsibility for managing datasets and workflows. This allows for more accountability and increased quality of the data. While this is valuable for tracking responsibility in ETL pipelines, the approach remains focused on tabular data and does not include spatial-specific operations such as coordinate reprojections or topological edits.

Furthermore, while Jamedzija (2021) mentions data governance, its interpretation primarily revolves around permission management and access-level control. Broader governance aspects, such as structured transformation documentation, metadata integrity, and alignment with standards like ISO 19115, an international standard defining the structure and format of geographic metadata to ensure interoperability and transparency in spatial data management, remain insufficiently addressed. Together, these studies show progress in the general ETL lineage, but none provide a dedicated solution for spatial data. The gap lies in the absence of an automated, ETL-integrated lineage model designed specifically for geospatial workflows, which must address unique spatial requirements such as accurate geometry handling, coordinate reference system transformations, spatial relationship tracking (e.g., topological changes, spatial joins), and detailed metadata capture aligned with geospatial standards.

Moreover, even when data governance is mentioned in these studies, it is typically framed in terms of permissioning or data ownership, without attention to spatial transformation semantics**,** geometric data integrity**, or** metadata conformance to geospatial standards. As a result, existing models fail to support critical use cases in geospatial domains—such as determining how a cadastral boundary was reprojected, which features were spatially joined, or how overlapping layers were intersected in producing a final dataset.

The implications for organizations like the RVB are significant:

- **Lack of transformation traceability**: If an output dataset contains incorrect or outdated spatial information (e.g., property boundaries), there is no systematic way to identify the specific operation or workflow that introduced the error.
- **Reduced auditability and legal defensibility**: When spatial datasets are shared with municipalities or ministries, the absence of lineage information undermines trust and accountability, especially in contexts that require regulatory validation.
- **Increased risk to data quality and governance compliance**: Without lineage records, it becomes difficult to enforce access rules, verify user responsibilities, or comply with interoperability models such as ISO 19115 or the European Interoperability Framework (EIF).
- **Inefficiencies in debugging and operational processes**: Manual inspection of FME workflows to trace data transformations is time-consuming, error-prone, and often inconsistent, especially when datasets are shared across teams or organizational units.

These limitations collectively compromise core governance principles such as traceability, accountability, and auditability, and hinder the effective use of spatial data in decision-making, especially in high-stakes domains like land management, infrastructure planning, or environmental regulation. For the RVB, where spatial data directly supports property valuation, legal boundary definition, and public asset oversight, the absence of robust lineage tracking is not merely an operational inconvenience—it is a critical vulnerability in their data governance framework.

Addressing this gap requires a lineage solution that is spatially aware**,** workflow-integrated**,** and aligned with established governance standards**.** Such a solution must support the capture and representation of transformation logic at both the attribute and geometry levels, enabling end-to-end traceability of spatial data products. The following section defines the research objectives and outlines how this gap will be addressed through the development of a tailored lineage model for spatial ETL workflows in FME.

1.3. Research objectives

Given the identified limitations in existing lineage tracking solutions for spatial ETL (Section 1.2), the primary objective of this research is to design an automated lineage model tailored specifically to geospatial workflows. The goal is therefore to bridge the gap between generic data lineage frameworks and the complex, geometry-driven nature of spatial data transformations.

At a theoretical level, the research seeks to contribute a conceptual model for spatial data lineage that incorporates geometry-level operations**,** semantic transformation tracking, and standards-aligned metadata capture. By extending lineage concepts beyond traditional tabular structures, the model addresses a critical gap in the existing literature. Namely, the absence of spatial logic in current provenance and governance frameworks.

From a practical perspective, the research will result in a functioning prototype capable of capturing and exporting lineage metadata from real-world FME workflows. This includes tracking spatial operations such as coordinate reprojections, spatial joins, and topological edits, as well as recording attribute-level changes and associated user or system actions. The resulting lineage records will be aligned with international metadata and governance standard of the ISO 19115**,** ensuring applicability in institutional settings like the RVB.

By achieving these objectives, the research is expected to deliver the following contributions:

- **Improved traceability** across spatial ETL workflows, allowing organizations to reconstruct and validate transformation histories from source to final dataset.
- **Enhanced auditability**, with lineage records enabling verification of compliance with internal processes, regulatory requirements, and external reporting obligations.
- **Stronger accountability** by clearly documenting who modified spatial data, when, and through which transformations—supporting both internal governance and external transparency.
- **Increased interoperability and standard compliance**, ensuring that lineage outputs can be integrated into existing data governance systems and exchanged with institutional partners.

For the RVB and similar organizations, the proposed model will provide a robust mechanism to embed data governance principles directly within operational workflows, reducing reliance on ad hoc or manual lineage tracking approaches. In doing so, it offers a scalable and transferable solution that strengthens the reliability and defensibility of geospatial data products.

1.4. Research question and scope

In line with objectives stated in Section 1.3, the research is guided by one main question and a set of sub-questions. The main research question is:

How can a geospatial data lineage model be designed and evaluated to support end-to-end traceability, capture, and visualization of spatial transformations in geospatial ETL processes such as FME, thereby enhancing data governance and operational efficiency within organizations?

To systematically address this broad question, we break it down into six sub-questions:

1. What are the technical and organizational requirements and priorities for geospatial data lineage?

2. What data lineage models currently exist for tracking lineage in both spatial and non-spatial contexts, and what are their strengths and weaknesses in supporting spatial use cases?
3. What are the existing challenges and limitations in tracking data lineage within geospatial ETL tools, specifically when using tools such as FME?
4. How can a geospatial data lineage model be designed and implemented in FME to enable end-to-end traceability of spatial transformations?
5. How does the proposed model align with the ISO 19157 quality assessment framework, under the four core dimensions usability, accuracy, consistency and completeness?
6. How can the proposed model be tested to ensure that it meets the identified requirements for geospatial data lineage, and what methods or metrics can be used to evaluate its effectiveness in real-world scenarios?

By answering these sub-questions, the research will comprehensively cover understanding the problem context, devising a solution, and assessing its value. The scope of the work is deliberately focused on 2D geospatial ETL processes workflows like those in FME that handle vector and attribute data and does not extend to domains like 3D modelling or real-time sensor data (which have their own lineage challenges). The emphasis is on a conceptual and prototype implementation that could be generalized, rather than a full production software product. Nonetheless, the outcomes are intended to be practically relevant, the model should be implementable with reasonable effort in tools such as FME and compatible with existing governance standards so that it can integrate into the RVB's data management framework.

The research will be structured according to a design-oriented methodology to develop a geospatial data lineage model tailored for FME. The methodological steps to achieve this are as follows: (1) to identify the requirements (both technical and organizational) for geospatial data lineage tracking in an operational setting such as the RVB, (2) to review existing data lineage and provenance models, from both spatial and non-spatial domains, to inform the design, (3) to develop a taxonomy or schema of lineage information that captures spatial transformations in FME workflows, (4) to prototype a tool that extracts this lineage information (e.g., by instrumenting FME workflows or extending FME's logging capabilities), and (5) to validate the approach using real-world use cases from the RVB (e.g., applying the model to a set of representative FME workspaces and evaluating how well the lineage captured reflects the actual processes). Through these steps, shown in Figure 1.4, the research will produce a data lineage model tailored to spatial ETL and demonstrate its value in improving data governance outcomes.

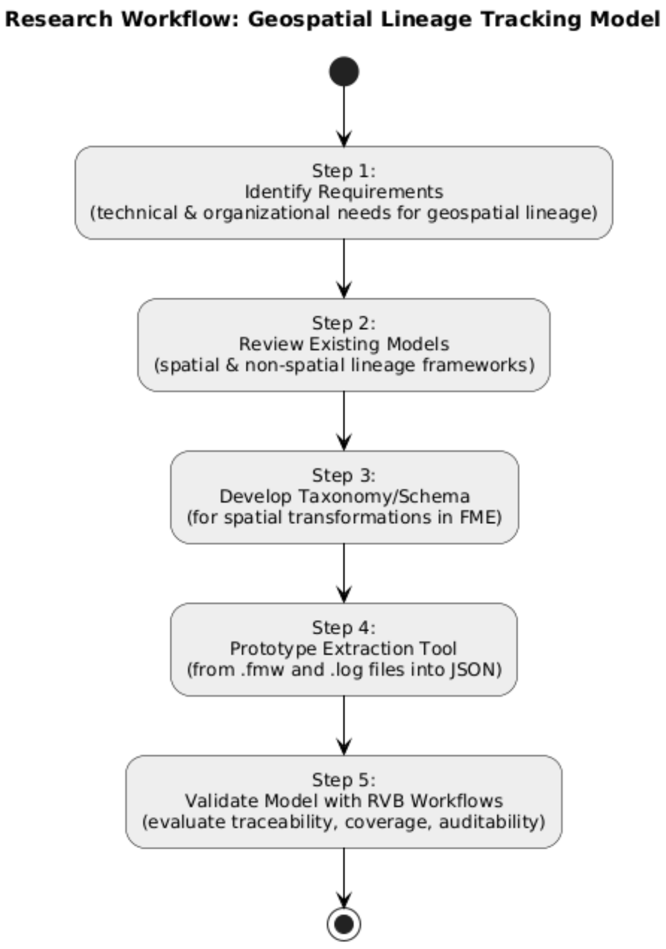

Research Workflow: Geospatial Lineage Tracking Model
Step 1:
Identify Requirements
(technical & organizational needs for geospatial lineage)
Step 2:
Review Existing Models
(spatial & non-spatial lineage frameworks)
Step 3:
Develop Taxonomy/Schema
(for spatial transformations in FME)
Step 4:
Prototype Extraction Tool
(from .fmw and .log files into JSON)
Step 5:
Validate Model with RVB Workflows
(evaluate traceability, coverage, auditability)

Figure 1.4 – Research workflow

1.5. Roadmap

This thesis is organized as follows:

Chapter 2: **Literature Review** – Critically reviews state-of-the-art data lineage standards and existing tools, with emphasis on identifying gaps specific to spatial ETL workflows. This review establishes what current models and systems offer and where they fall short in addressing geospatial transformation provenance.

Chapter 3: Methodology and Design – Translates the identified gaps into a set of design requirements and proposes a solution architecture. In particular, it details a dual-parser lineage architecture tailored to FME, which forms the conceptual blueprint for the lineage model and its implementation.

Chapter 4: Results and Evaluation – Presents the outcomes of applying the prototype lineage model to real-world RVB case studies. The chapter includes both quantitative and qualitative evaluations of the model's performance, using ISO 19157-derived metrics to assess data quality dimensions.

Chapter 5: Discussion – Discusses the implications of the findings for data governance, standards alignment, and practical usability. It also addresses the limitations of the current model and how the results fulfil (or fall short of) the research objectives.

Chapter 6: Conclusion and Future Work – Concludes the thesis by answering the main research question and each sub-question, highlighting how the research objectives were met. It also outlines an implementation roadmap for the RVB and suggests directions for future work to extend the geospatial lineage model beyond this study's scope.

2. Systematic Literature Review

This chapter presents a systematic literature review to evaluate the extent to which current lineage frameworks and systems address the structural and semantic needs of geospatial data transformations. In other words, we ask: How well do existing models capture the unique aspects of spatial ETL processes, and where do they fall short? Answering this question will highlight the limitations and gaps. In alignment with the research objectives (see Chapter 1), this review will identify limitations in present solutions and thereby inform the design requirements for a new FME-based spatial lineage model. To provide a structured analysis, the review is organized into three parts based on the first three research questions.

- **Part I:** Which lineage models and standards (e.g., W3C PROV, ISO 19115, GeoSPARQL) currently exist, and how do they represent spatial semantics such as geometry, CRS, and topological operations?
  This section examines standard-based data lineage models such as W3C PROV, ISO 19115, and GeoSPARQL. It assesses their structural integrity, semantic capabilities, and fitness for describing spatial processes and metadata in a consistent, standards-aligned manner.
- **Part II:** How do practical systems and ETL tools—particularly FME—capture lineage at both design-time and runtime, and what levels of granularity do they support?
  This section reviews platform-specific implementations of lineage tracking, particularly in spatial ETL tools like FME. It highlights current capabilities, limitations, and design trade-offs in capturing spatial transformations, including real-world examples and emerging trends.

- **Part III:** Where do existing, approaches fail relative to ISO 19115/GeoSPARQL alignment, operational feasibility, and integration into governance workflows?
  Drawing on gaps identified in the preceding sections, this final part introduces the conceptual foundation for a novel, standards-compliant lineage model. This proposed model integrates ISO 19115 and GeoSPARQL to enable interpretable, machine-readable, and FME-compatible provenance representations.

Together, these three research questions form a comprehensive foundation for understanding the current landscape of geospatial data lineage and for motivating the development of an operationally feasible, standards-aligned model suited to spatial ETL environments.

2.1. Part I – Conceptual Models and Theoretical Foundations

This section examines common frameworks and ontologies for spatial data lineage, evaluating their structure, scope, and suitability for capturing geospatial data provenance. Four main models are widely cited in the literature for provenance and geospatial metadata: W3C PROV, ISO 19115, OWL-S, and GeoSPARQL. Each offers distinct capabilities and has specific constraints in the context of spatial ETL workflows. The following review highlights how well it represents spatial transformations and what gaps remain for real-world adaption for FME.

2.1.1. W3C PROV

The W3C PROV model is a flexible, graph-based standard for representing provenance information across domains (World Wide Web Consortium, 2013). Its object-oriented structure centers on three core classes: prov:Entity, prov:Activity, and prov:Agent, with relationships such as wasGeneratedBy, used, and wasAttributedTo linking them into directed provenance graphs (Figure 2.1; Belhajime et al., 2012).

One of PROV's strengths is its abstract treatment of entities. A dataset, an individual feature, or even a single attribute value can each be modeled as a prov:Entity, enabling provenance capture at multiple granularities within the same framework (Closa et al., 2017). This scalability is particularly relevant for spatial ETL, where workflows may require lineage at:

- **Dataset level** (e.g., a shapefile representing cadastral parcels)
- **Feature level** (a single cadastral boundary within that dataset)
- **Attribute or geometry level** (the geometry of a parcel or a specific attribute such as area)

By not prescribing fixed semantic categories for entities or activities, PROV allows ETL designers to tailor lineage graphs to the operational and governance needs of their organization. This flexibility makes it theoretically well-suited for documenting the chain of spatial data transformations in platforms such as FME.

However, core PROV has important limitations for geospatial contexts. Geometry, coordinate reference systems (CRS), and topological operations are not first-class concepts in the standard. The model offers a generic prov:Location attribute, but this provides no structured way to represent geometries or spatial relationships. As a result, spatial semantics must be incorporated via external ontologies, such as GeoSPARQL for geometry and spatial predicates. Without such extensions, PROV records for spatial ETL risk being semantically shallow, making it difficult to capture the specific parameters of operations like reprojection, buffering, or intersection.

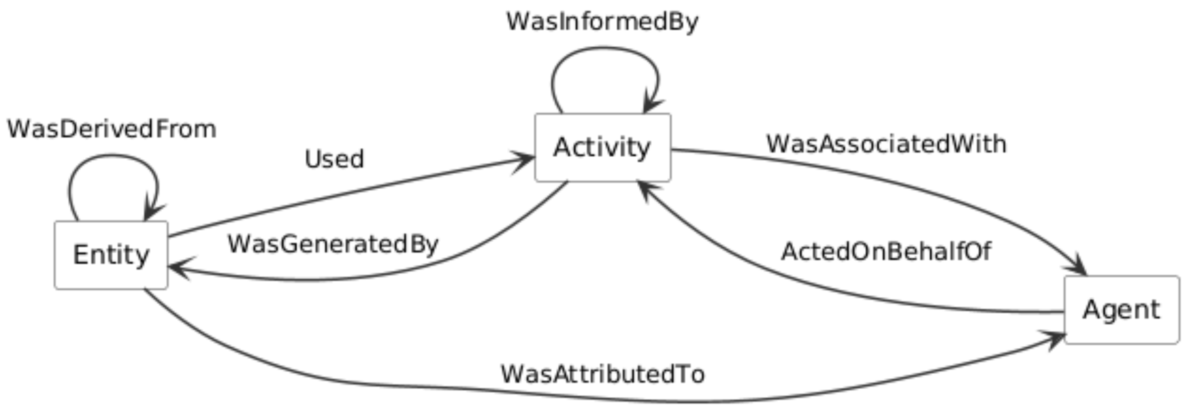


Figure 2.1 – PROV-model (Belhaijme et al., 2012)

2.1.2. ISO 19115

The ISO 19115 standard is the international standard for geospatial metadata, including lineage documentation. In ISO 19115 lineage information is captured in a structured section called LI_Lineage (Closa et al, 2017). The lineage model from 2003 included two main components the LI_Source which describes the datasets used and LI_ProcessStep which describes the process or transformation applied. This basic model provides a way to document the general process and source data in a structured and standardized manner. In 2009 an extension was introduced to improve the overall lineage detail. This was done through adding in additional fields such as LE_ProcessStep and LE_Algorithm (Jiang et al., 2018). The process step allows one to add in information on the procedure name, parameters and processing time (Closa et al., 2017). It supports metadata elements such as CRS, spatial extent, resolution, and lineage descriptions. Crucially, ISO 19115 is designed to serialize metadata in XML, aligning well with spatial ETL tools such as FME, which also serialize workflows as XML-based FMW files. A complete view of the ISO 19115 can be seen in figure 2.2.

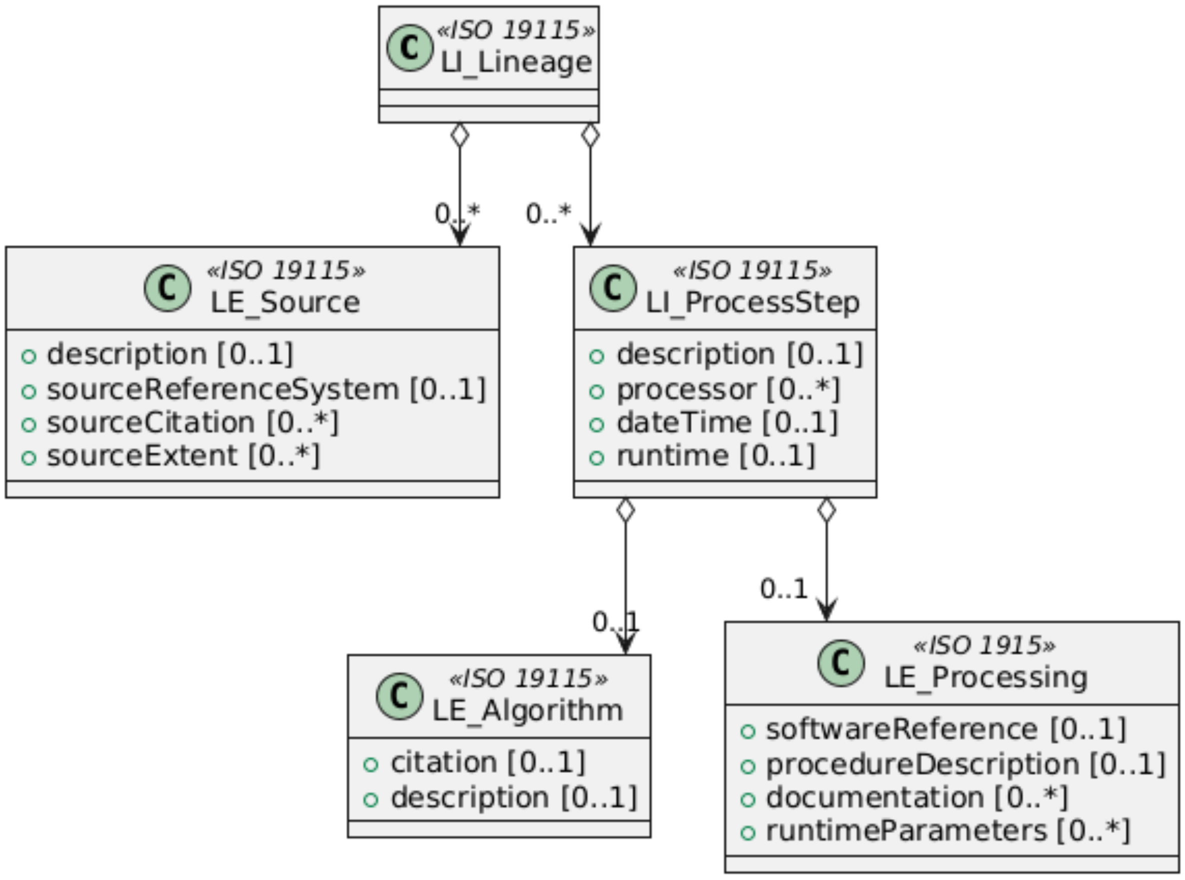


Figure 2.2 – ISO 19115 overview

Despite its strengths ISO 19115 has some limitations. Closa et al. mentioned that ISO lineage is less structured than W3C PROV and as a result results in more nestled structures when the lineage gets more complex (Closa et al., 2019). For example ISO allows one to specify lineage at various scopes (“dataset”, “feature type”, “feature instance”, “attribute instance”) (Jiang et al., 2018) In theory this covers granularity, but in practice nesting lineage for every feature or attribute becomes unwieldy – the metadata XML would become extremely large and complex if one tried to document provenance for each feature in a dataset. As noted by Closa et al. (2017), the hierarchical tree structure “generates a very deep structure that hinders comprehensibility” when used for fine-grained (feature-level) provenance (Closa et al, 2017). As a result, ISO lineage is most often used at the dataset level (or dataset series level), providing a summary of how the entire dataset was derived, rather than a detailed record of each operation in a workflow (Granell et al., 2013).

2.1.3. OWL-S

OWL-S (Ontology Web Language for Services) is an OWL-based ontology for describing web services’ capabilities, inputs, outputs, and process workflows. It provides a semantic framework to annotate what a service does and how it can be invoked, facilitating automated service discovery and chaining. In the mid-2000s, researchers in the GIS domain explored OWL-S to achieve semantic interoperability of geo-services (Fallahi et al., 2008). Lemmens (2006), for example, applied OWL-S to geospatial processes, demonstrating that one can create an ontology of geo-operations and use OWL-S to mark-up services like coordinate transformations or map overlay operations. By doing so, each operation’s functional meaning (e.g. “buffers a polygon by distance X”) is formally described, enabling reasoning engines to understand and even compose complex geoprocessing workflows based on these descriptions. This is particularly valuable for semantic transformation logic – OWL-S can capture the intent of a transformation (e.g., a “clip” or “merge” function) in a way that is software-agnostic and machine-readable. Studies have shown that using OWL-S and related ontologies can improve the discovery of appropriate geo-processing services and the automated chaining of them into larger workflows (Lemmens, 2006).

Despite these strengths in semantic description, OWL-S has significant shortcomings for data lineage tracking. It was not built to record the history of data through a sequence of operations; rather, it focuses on describing types of services and how to invoke them. OWL-S by itself does not maintain provenance relationships (which output came from which input, which agent executed it, etc.) in the manner that PROV or ISO lineage do. Lemmens’ work and others illustrate OWL-S as a planning/annotation tool, not a logging mechanism for executed processes. In a geospatial ETL context like FME, OWL-S could be used to semantically annotate each transformer (e.g., label a “Bufferer” transformer as an instance of a “BufferOperation” class), thereby clarifying its role. However, OWL-S would not automatically link that transformer to a specific output dataset or timestamp without an additional provenance model. In essence, traceability is not OWL-S’s function. It lacks a notion of data instances flowing through a chain – the very essence of lineage. Therefore, while OWL-S contributes a rich semantic layer (answering what kind of operation each step is in a formal sense), it must be integrated with a provenance framework to capture when, by whom, and with which inputs/outputs those operations occurred. This makes it a complementary piece in lineage modelling, rather than a standalone solution.

2.1.4. GeoSPARQL

GeoSPARQL is an OGC standard (first adopted in 2012) that defines a vocabulary and query extension for representing geospatial information in RDF graphs (Abhayaratna et al., 2020).

While GeoSPARQL is not a provenance or lineage model per se, it provides the necessary geospatial vocabulary (classes, properties, and functions) to describe geometry, spatial relationships, and spatial reference systems in a Semantic Web context. In a provenance knowledge graph dealing with spatial data, GeoSPARQL can play a crucial role by enabling one to capture the "where" aspect of data and transformations: the geometry of datasets, the spatial region affected by a process, the coordinate reference systems, and topological relations between inputs and outputs.

Key features of GeoSPARQL include:

- A class **geo:Feature** for spatial features (which could be anything with a geometry, e.g., a road segment, a land parcel, a dataset that covers a region).
- A class **geo:Geometry** for geometric objects (points, lines, polygons, etc.).
- The property **geo:hasGeometry** linking a Feature to its Geometry.
- A set of spatial relationship predicates (e.g., geo:sfWithin, geo:sfIntersects, etc., based on the Simple Features relations) and spatial functions that can be used in SPARQL queries (like geof:distance, geof:buffer, etc. in GeoSPARQL 1.1).
- Support for specifying the Coordinate Reference System (CRS) of geometry literals via URI identifiers.

GeoSPARQL can also help describe spatial transformations themselves. For instance, a transformation such as reprojection can be partly described by indicating that the output geometry is in a different CRS than the input geometry (GeoSPARQL allows specifying the CRS URIs in the geometry literal).

GeoSPARQL enriches geospatial provenance by providing the language to describe spatial properties of data within the provenance model. It addresses the "spatial context" aspect that pure provenance models leave unspecified. By combining GeoSPARQL with PROV or other lineage representations, we achieve a more complete lineage record: not only do we know which operations produced a dataset, but also what the spatial characteristics of those inputs and outputs were (coverage, location, geometry shapes). This is particularly useful when tracking lineage in scenarios like spatial data integration (where overlapping regions matter) or change detection workflows (where you want to know the footprint of data that was compared).

2.1.5. Summary

Each model brings a unique perspective to capturing geospatial provenance. Table 2.1 summarizes the comparative strengths of W3C PROV, ISO 19115, OWL-S, and GeoSPARQL across key criteria relevant to spatial ETL lineage: process traceability, spatial metadata support, semantic description of transformations, and geometry-level detail.

Table 2.1**:** Comparison of provenance models/ontologies relevant to geospatial ETL lineage. PROV and ISO 19115 address, respectively, execution-time provenance and metadata lineage containers, while OWL-S can provide a domain-specific process vocabulary and GeoSPARQL contributes geometry representations and standardized spatial relations/functions.

Criteria	W3C PROV	ISO 19115	OWL-S	GeoSPARQL
Process traceability	Strong—domain-neutral provenance with Entities, Activities, Agents; supports edges like wasGeneratedBy, used, wasDerivedFrom. Granularity depends on how you model Activities, no native spatial context.	Good at dataset/metadata level—LI_Lineage with LI_ProcessStep and LI_Source records stepwise lineage in metadata; typically, descriptive (free text + roles).	Limited for lineage—describes service/process types (inputs/outputs, preconditions/effects) but not execution provenance chains. Use with a provenance model for run histories.	None—does not model processes or workflow order; must be combined with PROV/ISO to represent "how/when".
Spatial metadata	Minimal—no built-in CRS, extent, or data quality; can be extended in a domain profile.	Strong—designed for geospatial metadata (identification, extent, quality, spatial reference); lineage sits naturally in this schema.	Minimal—domain-neutral; you can reference spatial ontologies, but OWL-S itself has no CRS/extent elements.	Moderate—first-class geometry literals (WKT/GML/GeoJSON) and some geometry properties (e.g., spatial resolution/accuracy in 1.1), but not full dataset-level metadata/quality reports.
Semantic transformation logic	Basic—asserts that an Activity occurred; no standard vocabulary of geo-operations (needs extensions or domain ontologies).	Descriptive—can label/classify `LI_ProcessStep` but does not standardize operations like "buffer" or "clip".	Enables process vocabularies—provides a formal process model so you can define a domain ontology of operations, but ships with no geospatial operation set by default.	Strong (spatial)—standardized spatial relations and functions for reasoning and queries.
Geometry-level detail	Weak—no native geometry types; geometry must be modeled externally.	Moderate—can record spatial extent/CRS as metadata for datasets/outputs, but not per-feature geometry lineage.	Weak—can describe a service that produces/consumes geometries, but not geometry representation or predicates.	Excellent—geometry classes/literals and rich spatial predicates and functions; queryable in SPARQL.
Execution / temporal capture	Yes—prov:startedAtTime, `prov:endedAtTime`, qualified associations; good for run histories.	Limited—LI_ProcessStep can include date/time, responsibility and description, but implementations are often free-text; weaker for fine-grained run events.	No—models process types and I/O, not actual executions; requires pairing with a provenance model.	N/A—focuses on spatial descriptions and functions, not temporal/run events (combine with PROV/ISO).

Evaluating the above models highlights that no single standard fully meets the needs of spatial data lineage in complex ETL workflows. W3C PROV provides a rigorous structure for process documentation but lacks spatial awareness; ISO 19115 ensures essential geospatial context but is semantically shallow regarding transformation logic; OWL-S richly describes process semantics but without lineage chaining; and GeoSPARQL offers detailed spatial descriptors yet no notion of temporal or process flow. To capture the lineage of FME-based workflows in a complete, standards-aligned manner, a hybrid approach is necessary. This thesis adopts ISO 19115 combined with GeoSPARQL as the conceptual foundation for the lineage framework. The choice is grounded in both the comparative analysis above and recommendations in recent literature.

2.2. Part II – Practical Systems and Tool-based lineage models

Building on the conceptual frameworks discussed earlier, this section examines how data lineage is implemented in models, with an emphasis on geospatial ETL tools like FME. It reviews existing systems and prototypes that attempt to capture or visualize data lineage, highlighting their current capabilities, limitations, and design trade-offs. By analysing these practical implementations – including real-world examples and emerging industry trends one can identify how well they support spatial transformations and where gaps remain. This practical review sets the stage for the prototype development in later chapters by pinpointing gaps that the proposed model will address.

2.2.1. Search Strategy and Selection Criteria

Relevant literature is sourced from academic databases including Google Scholar, Scopus, and Web of Science, using focused keywords such as *data lineage*, *geospatial ETL*, *geospatial data governance*, and *transformation tracking*. A structured three-stage filtering strategy is applied to refine the selection:

- **Primary inclusion criteria:** Publications that present ETL-focused lineage workflows or applied technical methodologies.
- **Secondary inclusion criteria:** Studies discussing lineage in ETL workflows but don't have a concrete solution, only mention ETL in passing.
- **Exclusion criteria:** Works with no relation to ETL workflows.

2.2.2. Relevance Categorization and Key Contributions

To support comparative analysis, reviewed publications are classified into three tiers based on their applicability to the research objectives. The search strategy can be seen in figure 2.3.

High Relevance (Green):

- Studies that present tools, frameworks, or models directly applicable to automated data lineage tracking within geospatial ETL workflows. This includes literature that discusses methodologies for capturing transformation steps, spatial data flows, and metadata extraction, specifically tailored for tools like FME or similar spatial ETL environments.

Moderate Relevance (Yellow):

- Literature that offers conceptual alignment with data lineage and governance models but provides limited detail on geospatial-specific challenges or practical implementations in spatial transformation contexts. These works might address general ETL lineage tracking principles that are only partially applicable to spatial data scenarios.

Low Relevance (Red):

- Works with minimal technical contribution or relevance to the challenges of geospatial data transformations. These studies may discuss ETL lineage tracking in non-spatial contexts or contribute only marginally to the understanding of spatial transformation logic (such as geometry operations or coordinate changes).

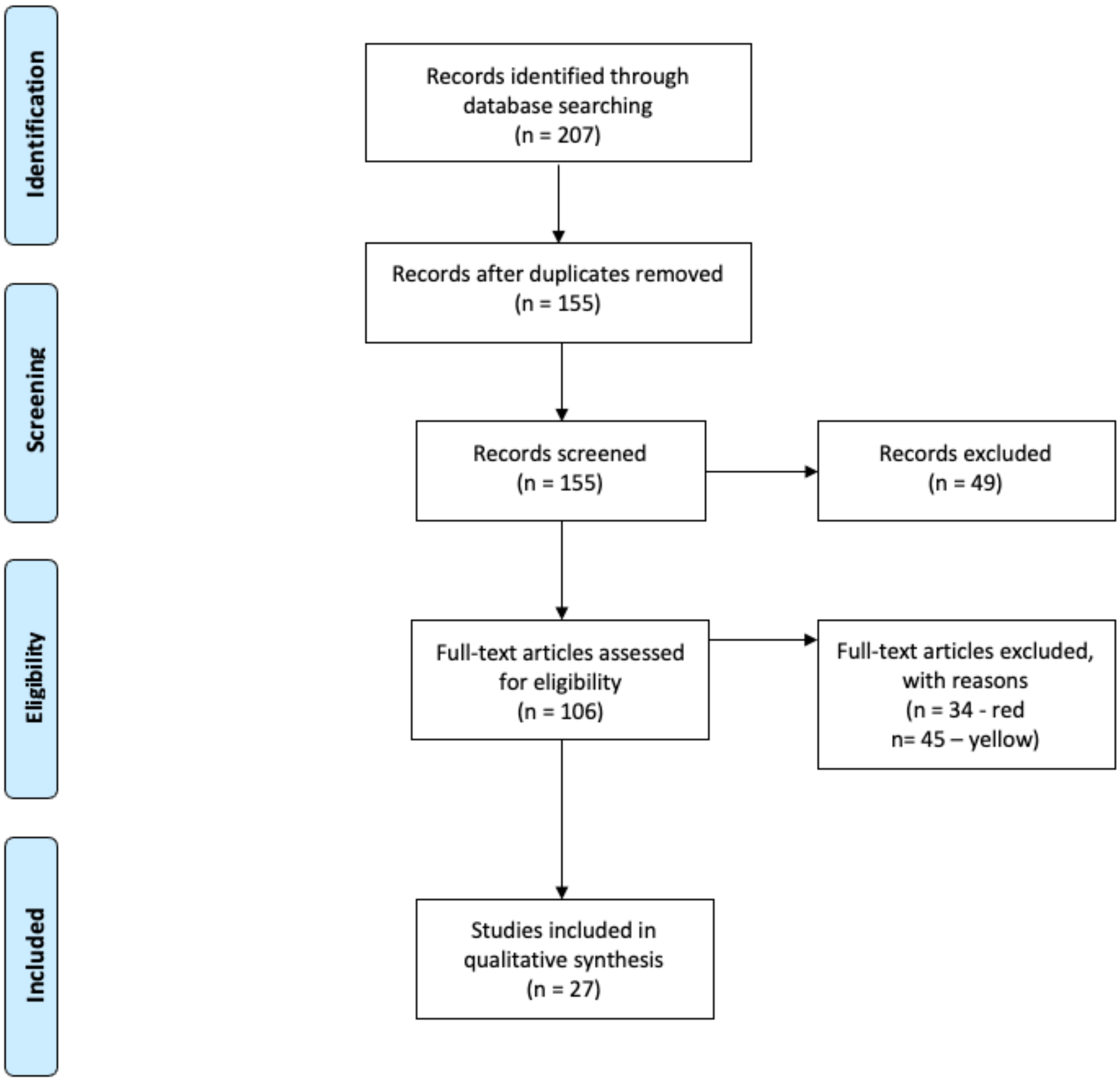

Identification
Screening
Eligibility
Included
Records identified through database searching (n = 207)
Records after duplicates removed (n = 155)
Records screened (n = 155)
Records excluded (n = 49)
Full-text articles assessed for eligibility (n = 106)
Full-text articles excluded, with reasons (n = 34 - red n= 45 – yellow)
Studies included in qualitative synthesis (n = 27)

Figure 2.3 – PRISMA methodology for the found literature

2.2.1. Summary of Key Publications (2015–2025)

Table 2.2 presents a matrix of high-relevance studies on automated geospatial data lineage. It outlines each publication's year, authors, methodology, the tool or platform context, the spatial focus (e.g., whether it captures feature-level geometry/attribute transformations), and key contributions.

Table 2.2 - Key publications on automated geospatial data lineage (2015–2025), with focus on methodologies, platforms, and contributions relevant to spatial ETL workflows.

Author, year & title	Methods	Key findings	Gaps in literature	Tool/platform used	Spatial specificity	Formats handled	Degree of automation
Closa et al., 2019 — "A provenance metadata model integrating ISO geospatial lineage and the OGC WPS"	Conceptual model + implementation in MiraMon; integrates ISO 19115(-1/-2) lineage with OGC WPS	Demonstrates automatic capture of process/source lineage via WPS; improves completeness of geoprocessing provenance	Needs a spatial query vocabulary (topology), not covered by ISO lineage alone	MiraMon WPS + ISO 19115(-1/-2)	Spatial	Raster and vector (use case demo is raster)	High (automatic within WPS)
Zhang, Jiang, Zhao, Yue & Zhang, 2020 — "Coupling OGC WPS and W3C PROV for provenance-aware geoprocessing workflows"	Coupled WPS–PROV model + XML schema + prototype	Combined WPS and PROV yields more complete, interoperable workflow provenance	No native spatial topology predicates; relies on external vocabularies for geometry semantics	OGC WPS + W3C PROV	Spatial	WPS process inputs/outputs (data-type agnostic)	Medium–High (model + implemented workflow)
Schoenenwald et al., 2021 — "Collecting and visualizing data lineage of Spark jobs"	Uses Spline Agent to digest Spark execution plans; property-graph storage; web UI	Reliable coarse-/fine-grained Spark lineage; practical visualization principles	Spark-centric; no geospatial semantics	Apache Spark + Spline	Non-spatial (general ETL)	Spark SQL pipelines	High (agent-based, automatic)
Tang et al., 2019 — "SAC: A System for Big Data Lineage Tracking"	Spark-Atlas-Connector; metadata model + REST + UI	Cross-job lineage across HBase/HDFS/Hive/SQL/ML/streaming; production deployment	Coarse-grained; Spark/Atlas-dependent; no geometry/topology semantics	Apache Spark + Apache Atlas (SAC)	Non-spatial	HDFS, Hive, HBase, Kafka	High (automatic)
Zheng, Alawini & Ives, 2019 — "Fine-Grained Provenance for Matching & ETL (PROVision)"	Provenance system for matching/ETL; extends DB provenance to capture equivalences & selective eval	Achieves cell/record-level provenance for complex matching & ETL	Relational focus; no geospatial geometry lineage	PROVision prototype	Non-spatial	Relational Tables	High (system implementation)
Jamedžija & Đurić, 2021 — "Moonlight: A Push-based API for Tracking Data Lineage in Modern ETL processes"	Push-based REST API; jobs call API per step	Low integration overhead across platforms; simple ingestion model	Requires manual/API instrumentation; no spatial semantics	Moonlight API server	Non-spatial	JSON/API events	Medium (semi-automatic)

Closa, Masó, Julià & Pons, 2021 — "Geospatial Queries on Data Collection Using a Common Provenance Model"	Combines ISO 19115-2 lineage elements with PROV relations; queryable graph + demo	Shows how ISO lineage classes + PROV relations support graph queries over geospatial provenance	Not ETL-tool-specific; limited discussion of runtime evidence capture	ISO 19115-2 + W3C PROV	Spatial	General geospatial datasets (vector/raster)	Medium (implemented query system)

2.2.2. Trends in Geospatial Lineage Tracking Approaches

Although the limitations highlighted in section 2.5 indicate that no tool is fully capable of capturing, storing and visualizing data lineage in a usable way, the tools do allow for insights that are core to the building of a new tool.

Automation in lineage tooling

Multiple studies reviewed in the literature emphasize the critical role of automation in enhancing the efficiency and reliability of data lineage tools. A notable example is presented by Closa et al. (2019), who developed a GIS-based solution utilizing a Web Processing Service (WPS) to automatically extract lineage from both raster and vector datasets. Their tool is compliant with ISO 19115-1 and ISO 19115-2 metadata standards, ensuring standardized lineage documentation across diverse geospatial outputs.

The preference for automated lineage captures over manual methods is grounded in several practical advantages. First, an automation eliminates human error in documenting transformation steps and metadata attributes. This is particularly important in complex workflows where oversight can compromise data integrity or traceability. Additionally, automated tools enforce a consistent structure and method for collecting and presenting lineage information. This uniformity is vital for standardization and enables comparative analysis, facilitating data audits, and ensuring clarity when datasets are shared across teams or organizations.

Second, from an operational standpoint, automation accelerates lineage extraction, integrates seamlessly into real-time or batch processing environments, and supports scalable implementation across large datasets. In business contexts where decision-making relies on reliable analytics, the benefits of automatic lineage; accuracy, repeatability, and clarity, are not just technical preferences but operational necessities.

Visualization and user interaction

Beyond the automation of data lineage capture, the literature also emphasizes the importance of making lineage information usable and accessible to end users. As noted by Malaverri et al. (2012), high-quality lineage is only truly valuable if it can be easily interpreted and applied by analysts, data managers, and other stakeholders within an organization. Usability extends beyond technical accuracy as it encompasses clarity, accessibility, and relevance in presentation.

One of the primary challenges in this area is that raw lineage data, especially in formats such as JSON, can be overwhelming or unintelligible to non-technical users. While structured logs may retain all necessary details, their utility is limited without appropriate presentation. To address this, several studies advocate for the use of visual representations, such as diagrams, graphs, or flow-based models, to communicate lineage. These formats not only reduce cognitive load but also allow for quicker comprehension of complex workflows, dependencies, and data transformations.

When visualizations are generated through automated tools, as discussed in the previous section, the result is a consistent and standardized lineage output. This standardization ensures that every user interacts with the lineage data in the same visual context, enabling easier comparison, validation, and reuse across projects. In practical terms, this transforms data lineage from a back-end traceability mechanism into a front-line asset that supports data governance, quality assurance, and collaborative analysis.

Quality and reproducibility

A recurring theme across the literature is the critical role of reproducibility in data lineage frameworks (Closa et al., 2021). Closely tied to automation and standardization, reproducibility ensures that, given the same inputs and processes, identical outputs can be regenerated reliably. This is especially feasible when the lineage capture is fully automated, as consistent metadata collection and transformation tracking eliminate variability introduced through manual methods.

Reproducibility strengthens the credibility of provenance data by enabling independent validation of results. It allows analysts and auditors to trace not only what was done, but also to re-execute workflows to verify outcomes. From a business operations standpoint, this supports accountability, transparency, and quality assurance in geospatial data processing.

In addition to verification, reproducibility contributes to operational security. By maintaining precise records of when changes were made, by whom, and to which components of a dataset, organizations gain insights into potential vulnerabilities, such as the propagation of sensitive or restricted source data into downstream products. This kind of traceability is fundamental for compliance and for managing data access policies.

While some GIS platforms are beginning to embed reproducibility features, current implementations remain limited. For example, ArcGIS Pro provides geoprocessing history logs that track certain operations. However, these logs still require manual metadata exports and do not deliver a fully queryable, standardized provenance graph. As such, current tools demonstrate the potential for reproducibility but fall short of delivering fully integrated, platform-independent solutions. This highlights an area for continued development, especially in building tools that can deliver reproducibility, automation, and usability as part of a cohesive data lineage framework.

2.2.3. Comparative Insights from Tools and Platforms

While FME is a powerful spatial ETL platform, it still lacks a native, queryable lineage repository, its execution logs record Reader, Transformer, and Writer events but do not constitute a standards-compliant provenance store. Community threads from 2022 and 2025 (Safe Software, 2025) confirm that users must build ad-hoc solutions to export these logs into external catalogues, underscoring the demand for an integrated lineage system.

There is one tool on the market that has an integrated lineage that is of AWS. As cloud infrastructure becomes more integrated into enterprise data strategies, the need for detailed and continuously updated data lineage has grown. AWS DataZone addresses this by offering an API-driven environment for managing and governing data across organizational and external boundaries. A key feature of AWS DataZone is its support for column-level lineage tracking, which records the full lifecycle of data as it moves through various stages. This includes initial ingestion of raw data, transformation during ETL processes, and final usage in analytics, reporting, or client-facing systems. The lineage information is not static; it is actively updated and visualized through interactive graphs, making the data flow transparent and easier to audit. These lineage views are tightly connected to governance policies. Every transformation step and access point are logged, which supports data quality management, access control, and compliance requirements.

In practice, this means that teams can trace specific data values back to their origin, verify processing steps, and ensure that outputs are based on trusted inputs. Compared to traditional metadata catalogues, DataZone's integration of real-time lineage and automated updates reduces manual documentation work. It also improves visibility across workflows, especially when data moves between services or teams. This supports reproducibility, simplifies error tracking, and enhances security by allowing organizations to identify and isolate weak points in data flows. AWS DataZone currently provides one of the most complete implementations of end-to-end lineage in a production cloud environment. Its ability to scale across platforms while maintaining updated lineage graphs makes it a strong candidate for organizations aiming to mature their data governance strategy (Amazon Web Services, 2024; Interlandi et al., 2018).

However, this tool is firstly not based around spatial data but only focuses on general data lineage. Second, as AWS is a company that has multiple platforms, the RVB would have to adopt these multiple platforms for this tool to work effectively. This is not only against the RVB policy to not have a large reliance on one vendor but also would increase the risk of vendor lock-in. This is ultimately why this tool is not directly applicable to spatial data, and hence the RVB.

2.3. Part III - Towards a Lineage Model for FME Workflows

Drawing on the gaps identified in the review; this section introduces a conceptual foundation for a hybrid spatial lineage model. The need for this model arises due to current models lacking integration with standards and ETL tools. This research proposes a hybrid lineage model that explicitly addresses these shortcomings. The model is designed to be both standards-aligned and operationally compatible with FME-based workflows.

The proposed approach integrates ISO 19115 for metadata structure with GeoSPARQL as a semantic ontology. ISO 19115 is widely adopted for describing geospatial datasets and their lineage (Di et al., 2013; Closa et al., 2019), but it does not natively capture the meaning of spatial transformations. However, ISO 19115 is natively structured in XML, which aligns with FME's `.FMW` file architecture. This makes it well-suited for extracting and representing the static components of a workflow. Including the origin of datasets, the spatial extent of operations, and the formal history of processing steps. However, while ISO 19115 excels at documenting structural lineage, it does not provide the semantics of spatial operations themselves. For example, it cannot distinguish between a spatial join, a buffer operation, or a reprojection in terms of functional intent.

Therefore, to complement ISO 19115's structural metadata approach GeoSPARQL's ontology was combined with ISO 19115 standards. While GeoSPARQL was originally designed for semantic web and linked data contexts, it has increasingly been used to enhance semantic querying and reasoning in spatial infrastructures. For example, He et al. (2014) proposed adding geospatial-based semantics to spatial data infrastructures (SDIs) to track transformation logic across services. Similarly, Ivanova et al. (2017) and Battle et al. (2012) identified

GeoSPARQL as a key enabler for interoperable provenance queries in distributed geoprocessing systems. These studies demonstrate that GeoSPARQL can serve not just as a query language, but also as a lightweight ontology for formally describing the intent and logic of spatial workflows.

In the proposed model, GeoSPARQL is used in exactly this way: to characterize each spatial operation in the FME workflow with a semantic description. Each process step in the lineage is annotated with a GeoSPARQL term that denotes its geospatial function. For example, an FME Clipper transformer (which trims features to a boundary) can be represented by the GeoSPARQL function vocabulary term `geof:difference`. Likewise, the FME Reprojector transformer (which changes a dataset's CRS) corresponds to a `geof:transform` operation. By encoding each transformation step with such semantic annotations, the lineage model gains a high level of transparency. Rather than merely logging that a step occurred, it records what kind of spatial operation was performed, in a formal notation that both humans and machines can interpret (Open Geospatial Consortium, 2024).

Using this existing vocabulary ensures that the lineage captures spatial operations in a consistent, interoperable manner. This alignment offers several benefits. First, it supports more interpretable and audiTable lineage than flat transformer logs. Second, it enhances interoperability with other systems that already support GeoSPARQL. Third, it creates a foundation for semantic filtering, reasoning, and validation of workflow correctness based on spatial logic.

To illustrate the practicality of this approach, consider a simplified real-world scenario: an FME workspace takes a set of raw satellite images, reprojects them from WGS84 to a national coordinate system, clips them to a study area boundary, and then mosaics the results. Using the hybrid lineage model, we would capture an ISO 19115 lineage record with three process steps (Reprojection, Clipping, Mosaicking), each with its input sources and output. Enhancing this, each step would be annotated. For example, Step 1 has a GeoSPARQL semantic tag indicating a coordinate transformation*,* Step 2 a spatial clipping (geometry difference)*,* Step 3 an aggregation/merging operation*.* At runtime, FME's log might also provide the exact parameter values (e.g., the EPSG codes of source/target CRS, the polygon used for clipping, the list of image IDs mosaicked), which the model can record as well. The end result is therefore a comprehensive, standards-based lineage graph in JSON formatting. This data can allow users to query the data, "show me all workflows that produced data in Coordinate System X" or "find the workflow steps that involve mosaic operations and which images they combined," and so on.

By combining ISO 19115 for structural metadata with GeoSPARQL for spatial semantics, the proposed hybrid model delivers a standards-aligned lineage approach that is directly compatible with spatial ETL workflows. This integration addresses the key gaps identified earlier. Embedding GeoSPARQL enables the model to capture the semantic intent of spatial operations and at the same time, structuring the model around ISO 19115 ensures consistency with established geospatial metadata standards.

2.3.1. Lineage Extraction Methods in FME

To bridge the conceptual model into a practical solution for FME workflows, we identify four complementary methods for extracting data lineage in FME-based spatial ETL processes. FME does not natively provide a comprehensive, standards-based lineage repository, so a combination of static and dynamic extraction techniques is needed. The four practical methods are: (1) Static parsing of FME workspace files (.FMW), (2) FME built-in logging, (3) targeted transformer logging, and (4) FMW flow job history. Table 2.3 summarizes these methods across key evaluation dimensions (lineage completeness, debugging effectiveness, maintainability, and standards compatibility), followed by a discussion of how a hybrid strategy leverages their advantages.

Table 2.3 – Comparison of FME Lineage Extraction Methods (static vs. dynamic approaches).

Method	Evidence scope	Granularity & completeness	Debugging / tracing value	Ops & maintainability	Standards mapping effort	Design decision for this thesis
FMW file parsing (read .fmw workspace)	Design-time (declared Readers/Writers/Transformers; connections; published parameters)	High for configuration graph; no data-dependent branching or run outcomes	Good for impact analysis and dependency graphs before execution	Needs a parser; Table across runs; update when FME schema changes	Low–Medium: map to ISO 19115 lineage (LI_Lineage / LI_ProcessStep / LI_Source); add GeoSPARQL only when naming spatial functions	Use as the primary source for the process graph and static metadata
Built-in FME job logs (Workbench/Flow)	Runtime (what executed; warnings/errors; feature counts; timings)	Moderate–High for executed steps; detail depends on log level; not full transformer semantics	Strong for post-run debugging; shows which sources/targets ran and when	Generated by default; parsers must tolerate minor format/version changes; easy to centralize/rotate	Medium: transform text/JSON into ISO 19115 process steps with `dateTime`/responsibility; GeoSPARQL not native	Use as the primary source for timestamps, counts, and execution status
Embedded per-transformer logging (Logger / PythonCaller)	Runtime, targeted (you choose which steps emit structured events)	High for instrumented steps (operation name, key params, sample stats); zero for uninstrumented steps	Excellent for localizing errors and validating semantics (e.g., buffer distance actually used)	Adds overhead if blanket-applied; keep scoped and configurable; version-safe if you centralize a reusable subgraph	Medium: transform text/JSON into ISO 19115 process steps with `dateTime`/responsibility; GeoSPARQL not native	Not Used
FME Flow job history / archived logs (files or REST)	Runtime, historical (multi-run audit; parameters; status)	Mirrors built-in logs; completeness depends on retention and log level	Effective for compliance, SLA, and longitudinal lineage	Low: Flow auto-archives; REST gives structured access; straightforward to warehouse	Medium: transform text/JSON into ISO 19115 process steps with `dateTime`/responsibility; GeoSPARQL not native	Not Used

Each of these methods supports different aspects of provenance tracking in FME workflows. Design-time lineage can be obtained by parsing the static .FMW workspace files, which yields the declared structure of the workflow (readers, transformers, writers and their connections) without needing to run it. This static approach cleanly captures the intended data flow and transformation steps configured by the user, aligning well with formal metadata elements from the ISO 19115 lineage schema. For example, an FME Reader or Writer in the workspace corresponds to an ISO 19115 source description (LI_Source), and each configured transformer can be represented as an LI_ProcessStep entry describing a processing step. By extracting these elements, the static parser builds a structural provenance. The limitation of this approach is that it does not reflect any runtime parameters such as time of run, execution time, etc.

In contrast, runtime lineage is best captured through FME's logging mechanisms or feature caches, which document what happened when the workspace executed. The FME log records each step of execution, including the initiation of readers and writers, the invocation of transformers, numbers of features processed. These dynamic details are crucial for validating the integrity of the lineage. For instance, confirming that all expected steps were executed. Feature caching through a custom bult insert for FME goes even further by preserving the state of the data at each transformation step, which ensures complete ground-truth lineage for that run. Both, logs and caches can reveal runtime-specific behaviours.

Given the strengths and weaknesses of each method mentioned in Table 2.3, the most effective solution is a dual-parser strategy that combines a static and a dynamic approach. In the proposed framework, a static parser first extracts the metadata from the .FMW workspace such as the readers, transformers, writers, and their configured parameters, producing a baseline lineage structure that is compliant with ISO 19115's core lineage. This satisfies the requirement for standards alignment and provides a container for all declared process steps and source references. Then, a dynamic parser (operating on FME's execution log outputs) supplements this model with operational details – such as actual runtime parameters, timestamps for each process step, feature counts run times. By merging these two sources, we obtain a comprehensive and semantically enriched lineage model that covers both the planned workflow and its executed reality. This hybrid approach also enables semantic annotations of the transformations: as the dynamic parser recognizes specific FME transformers at runtime, it can tag them with corresponding concepts from an ontology (GeoSPARQL) to capture the spatial operation semantics. This strategy lays the groundwork for the implementation of an automated lineage extraction tool, as will be detailed in Chapter 3.

2.4. Criteria creation for model

The criteria for the model are based on ISO 19157 that defines a set of quality measure for evaluating geospatial data. Through treating the lineage information generated by the model as a type of geospatial dataset, one can evaluate its quality using the same criteria that ISO 19157 defines for data (International Organization for Standardization, 2013):

- **Completeness:** the degree to which all required lineage information is present, and no extra information is included. This therefore covers both data omission and data commission, which is excess data that should not be present (Gerhard, 2006). For the model, completeness therefore means capturing all relevant FME process steps without skipping any and avoiding recording any false or irrelevant steps. This can be checked, through establishing if every transformation in a workflow that is supposed to be logged in the lineage is actually captured, and that nothing extra was added.

- **Logical Consistency:** the adherence of the lineage information to the defined rules and schemas. This includes conceptual consistency (conformance to the conceptual schema or ontology), domain consistency (using valid values and references), and format consistency (correct structure/format of the output) in the context of the model. For example, the model should assign each FME transformer to the correct GeoSPARQL definition and produce a lineage output that conforms to the JSON schema. Any violation (e.g. a transformer mapped to an incorrect class, or an invalid coordinate reference system code in the output) would indicate a lack of consistency.
- **Accuracy:** the correctness of the quantitative attributes in the lineage metadata. One can consider several aspects when considering accuracy: positional accuracy, this refers to the correct recording of spatial reference information (e.g. coordinate reference system) for each data source or transformation. Temporal accuracy, meaning the correct timestamps and ordering of process steps are documented. Thematic accuracy, which here refers to the correctness of attribute-related information, such as classification of operations or feature counts. Overall, the model's lineage should timestamp each process step in the proper sequence, and it should accurately reflect values like feature counts or class labels as reported by FME.
- **Usability:** the degree to which the lineage information is understandable and useful for end-users. In the criteria, usability encompasses the interpretability of the lineage, the scenario fit, and governance alignment. These aspects ensure the model is not only technically sound but also practically valuable for data governance.

Ureña-Cámara et al. (2019) propose a method for checking the quality of geographic metadata by adapting ISO 19157's quality elements to metadata records (Urena-Camara et al., 2019; IGN France International, 2013). Their approach introduces specific metrics, for example, counting the proportion of metadata elements present (completeness) or the rate of errors/inconsistencies in metadata fields, and then evaluate those metrics against target thresholds to determine if the metadata meets the quality expectations. Other studies that have used the ISO 19157 criteria help to establish baselines for the criteria are those by Drobinjak which gives insight into the thresholds that can be adopted (Drobnjak et al., 2016). As a result, the criteria that are expressed in Table 2.4 were chosen and established. By measuring completeness, consistency, accuracy, and usability of the model's outputs, a foundation was created for objectively validating the obtained lineage data in Chapter 4 and 5.

Table 2.4 – ISO 19157 criteria and their standards and specific measures

Code	ISO data-quality element	Sub-element	Description	Tool-specific measure	Tolerance / target
A01	Completeness	Commission	Excess data present in a dataset – features, attributes, or relationships that should not exist.	False-positive rate in transformer mapping (1 − P) = Count FP transformers ÷ total mapped	≤ 5 % (P ≥ 95 %)
A02		Omission	Required features, attributes, or relationships missing from a dataset.	1 − COV = (Actual Steps − Captured Steps) ÷ Actual Steps	≤ 5 % (COV ≥ 95 %)
A03	Logical consistency	Conceptual consistency	Adherence to the rules of the conceptual schema.	Semantic accuracy of GeoSPARQL class assignment P = TP / (TP + FP) on class labels	≥ 95 %
A04		Domain consistency	Attribute values fall within their permitted domains.	Validity of CRS & ontology URIs AC = Filled valid-URI fields ÷ expected	≥ 90 %
A05		Format consistency	Degree to which data conform to the agreed physical storage structure.	JSON-schema validation pass rate (no errors in JSON output)	100 %
A06	Positional accuracy	Absolute / external	Closeness of reported coordinate values to accepted true values.	Presence of sourceReferenceSystem attribute (CRS recorded) AC = CRS filled in LI_Source records ÷ total LI_Source records	≥ 90 %
A07	Temporal accuracy	Accuracy of time measurement	Closeness of reported time measurements to accepted true values.	Millisecond resolution of `dateTime` stamps	All timestamps present, ± 1 s

A08		Temporal consistency	Correctness of the order of events.	Chronological order of LI_ProcessStep records	0 inversions allowed
A09		Temporal validity	Data values fall within the correct time period.	Run time lies within declared job window (compare `dateTime` against job start–end)	Pass / Fail
A10	Thematic accuracy	Classification correctness	Correctness of class assignments for features or attributes.	Correct GeoSPARQL mapping P = TP / (TP + FP) on class labels	≥ 95 %
A11		Non-quantitative attribute correctness	Whether a non-quantitative attribute is correct or incorrect.	Field names & units correctly copied AC = Filled non-numeric fields ÷ expected	≥ 90 %
A12		Quantitative attribute accuracy	Closeness of a quantitative attribute to its accepted true value.	Feature-count delta between FME log and lineage	Expect ± 0 difference
A13	Usability		Quality information describing a dataset's suitability for a particular application **or** conformance to user-specific requirements.	Scenario Fit, Interpretability, & Governance Alignment QC UX (post-test survey)	Mean score ≥ 4 / 5

2.5. Closing Summary & Gap Statement

This chapter evaluated the current landscape of data lineage models and tooling in the context of spatial ETL workflows. It assessed both conceptual frameworks—such as W3C PROV, ISO 19115, GeoSPARQL, and OWL-S—and practical implementations, including log-based lineage in FME and cloud-native platforms like AWS DataZone. The focus was on their ability to support spatial data transformations across varying use cases.

Among the conceptual models, W3C PROV provides a strong foundation for capturing process-oriented provenance, particularly in terms of agents, activities, and entities. However, it lacks native support for spatial semantics, limiting its applicability in geospatial workflows. ISO 19115, by contrast, is explicitly designed for geospatial metadata and integrates well with XML-based representations such as those used in FME workspaces, making it structurally compatible with spatial ETL environments. GeoSPARQL contributes spatial reasoning capabilities through SPARQL extensions and geometry vocabularies but does not independently support provenance chaining. OWL-S facilitates semantic annotation of processes but operates largely outside of lineage-specific contexts, offering little alignment with data transformation histories.

From a tooling perspective, the review confirmed that no existing platform, including for ETL tools such as FME, provides a native, standards-aligned, and spatially expressive lineage model. Most general-purpose solutions are optimized for tabular or big data scenarios, with limited support for spatial data such as geometry manipulation, coordinate reference system (CRS) transformations, or topological operations. Even platforms with visual and automated lineage capabilities, such as AWS DataZone, lack spatial data integration and introduce risks around vendor lock-in, constraints that are particularly problematic for companies like the RVB.

Three critical deficiencies were identified in the current ecosystem. First, there is a consistent absence of formal spatial semantics: most lineage systems fail to represent spatial transformations in a machine-readable and human-interpretable way. Second, reproducibility is constrained by limited metadata granularity; even where transformations are logged, the detail is often insufficient to fully reconstruct spatial workflows. Third, tool-level fragmentation undermines integration: in FME, for instance, lineage metadata is split between static workspace files and runtime logs, with no unified model or persistent repository. Furthermore, no existing solution successfully combines ISO 19115 and GeoSPARQL in a way that delivers both standards compliance and operational feasibility.

2.5.1. Proposed Research Contribution

To address these gaps, the next chapter introduces a dual-parser lineage model purpose-built for spatial ETL environments. This model integrates static parsing of FME `.FMW` files with dynamic extraction from execution-time logs, producing a comprehensive and queryable lineage graph. The proposed framework is aligned with existing standards: ISO 19115 is used to structure core metadata, while GeoSPARQL provides the semantic foundation for representing spatial relationships and operations. The approach is designed to be fully compatible with FME's internal metadata model, enabling integration without requiring significant changes to existing workflows.

3. Methodology

Based on the gaps and requirements identified in the previous chapter, this research adopts a design-oriented methodology to develop and evaluate a geospatial data lineage solution. The approach is structured in three main phases: conceptual model design, prototype development, and validation. In the following overview, we outline each phase and how it contributes to strengthening data governance at the RVB.

3.1 **Conceptual Model Design**: The first phase focused on developing a comprehensive lineage framework grounded in established standards. Guided by the findings from the literature review and criteria defined in Chapter 2, the model's design uses ISO 19115 (and its extended lineage elements from ISO 19115-2) as the foundational schema for geospatial metadata. On top of this baseline, the design incorporates GeoSPARQL constructs to semantically represent FME's geospatial operations.

3.2 **Technical Implementation**: In the second phase, the conceptual model was translated into a working prototype system. A custom dual-parser tool was developed to extract lineage information from FME workflows by processing both the static workspace definition (.FMW file) and the dynamic execution log (.log file) in parallel. One parser component scans the FME workspace file to capture design-time metadata (readers, transformers, writers and their configurations), while another component parses the runtime log to capture execution details such as transformer order, parameter values, and processing timestamps. The extracted details from these two sources are then mapped into the predefined conceptual model structure. In practice, each FME Reader and Writer in a workflow is mapped to an ISO 19115 `LI_Source` (documenting input or output dataset, coordinate reference system, etc.), and each transformation step (FME Transformer) is recorded as an `LI_ProcessStep` enriched with GeoSPARQL descriptors for the spatial operation performed. The two streams of metadata are merged into a unified JSON lineage record conforming to the conceptual model.

3.3 **Deployment and Evaluation**: The final phase involved deploying the lineage prototype and evaluating its performance and usefulness in real-world scenarios. The JSON lineage output was integrated into a lightweight web application (built with Flask) to visualize and interact with the captured lineage. This interface allowed users to browse the end-to-end transformation graph of an FME workflow – for example, tracing which source datasets were used, what transformations (buffer, clip, join, etc.) were applied, and how results were produced. The research then validated the lineage model and tool using actual FME workflows provided by the RVB. Several representative ETL workflows were run through the system to check if the captured lineage met the evaluation criteria from ISO 19157 which was set up in chapter 2 to assess if the model accurately reflected the transformations that occurred.

In summary, this three-phase methodology ensured that the research not only designed a lineage model aligned with geospatial standards and FME's capabilities but also implemented a functional prototype and evaluated it. Together, these sections provide a comprehensive account of how the lineage model was realized and assessed within FME-based spatial ETL workflows.

3.1. ISO + GeoSPARQL integration

To ensure that the lineage data captured is interoperable so that it can be used in the RVB, it is necessary to select a standard provenance model that is native to geospatial data. In specific, the ISO 19115 was chosen, which was later updated to 19115-2 to include more elements as it is specialized standards for geospatial data. The main class used is the provenance metadata class `LI_Lineage`, from which 3 subclasses are defined:

- The process step (`LI_ProcessStep`), which captures the processing information. This includes the execution time, status, and parameters.
- The source (`LI_Source`), which is used to store the input and output information such as the reference system and the name of the data source.
- The output is placed into (LE_Source). This is not native to ISO 19115 but was added for FME parsing as FME has readers and writers. This would then map to the writer's element of FME to keep this clear distinction.

As this is a generic geospatial lineage data model, the main question remains how this translates into FME. An FME workbench has a standard workflow of a reader, transformers, and a writer.

- **FME Reader → LI_Source (Input):** Each Reader in an FME workflow is responsible for pulling in a source dataset. In the model, every FME Reader is represented as an LI_Source entry capturing the input dataset's identity and properties. For example, the source's name or path becomes the LI_Source's description or citation, and its coordinate reference system can be recorded in the source's metadata.
- **FME Transformer → LI_ProcessStep:** Each Transformer (the FME component that transforms or manipulates data) is mapped to an LI_ProcessStep in the lineage model. The LI_ProcessStep records the overall description of the process, the date time the process was run and the overall runtime.
- **FME Writer → LE_Source (Output):** Each FME writer creates the final output that will then be written to a Datawarehouse. Like the LI_Source this documents the source reference system, the extent and the overall description of the output.

While this captures a foundational structure, it does not fully encapsulate the semantic intent of spatial operations embedded in FME workflows. This is because semantics such as geometric operations like intersections, reprojections, or spatial predicates such as overlaps and within are not accurately captured.

To address this gap, the lineage model design incorporates semantic enrichment by integrating concepts from the OGC GeoSPARQL standard. GeoSPARQL provides a formal ontology and functions for representing geospatial operations and relationships in a machine-interpretable way. GeoSPARQL extends SPARQL with geospatial functions and vocabulary, giving the lineage model a standard semantic layer that matches FME's spatial operations. The use of GeoSPARQL was chosen based on three main factors:

1. **Standards Alignment**:
   As a recognized international standard maintained by OGC, it is suitable for use within governments and companies and ensures consistent updates (Open Geospatial Consortium, 2024).

2. **Ontology**:
   It provides a clear ontology for describing common geospatial operations, which links closely to those found in FME.
3. **Compatibility**:
   Many of the spatial transformations in FME correspond to those found in GeoSPARQL (OGC, 2024).

Our approach extends the ISO model by utilizing the additional lineage classes defined in ISO 19115-2, namely LE_Processing and LE_Algorithm, as hooks for GeoSPARQL annotations. ISO 19115-2 introduced these classes to allow detailed documentation of the processing procedure and algorithm used in a process step. We leveraged them in the following manner:

- Each **LI_ProcessStep/LE_ProcessStep** in the model is extended with an **LE_Algorithm** entry that describes the underlying spatial operation in a formal way. Rather than just a text description, we insert a reference to a corresponding GeoSPARQL function or concept. This mapping of FME transformers to GeoSPARQL terms was done by creating a lookup dictionary of 68 GeoSPARQL functions against FME's library of transformers it was established that 17 unique GeoSPARQL functions had an FME transformer equivalent and only those were semantically annotated. Transformers without a direct equivalent were left unannotated to avoid inaccurate semantics.
- The **LE_Processing** class is used to group the algorithm information and any parameters or processing notes for the step. Within each process step entry, LE_Processing serves as a container indicating that, for example, "this step performed a spatial buffer operation using GeoSPARQL's `geof:buffer` function". Technical parameters (like the buffer distance, coordinate reference details, etc.) can also be included here or in the LE_Algorithm description, ensuring that the semantics are accompanied by the quantitative details of execution.

By integrating GeoSPARQL in this way, the lineage model gains a semantic layer on top of the ISO structural layer. This hybrid approach (ISO 19115 for structure + GeoSPARQL for meaning) yields a lineage record that is both human-readable and machine-interpretable. An example of this worked into the ISO 19115 workflow can be seen in Figure 3.2.

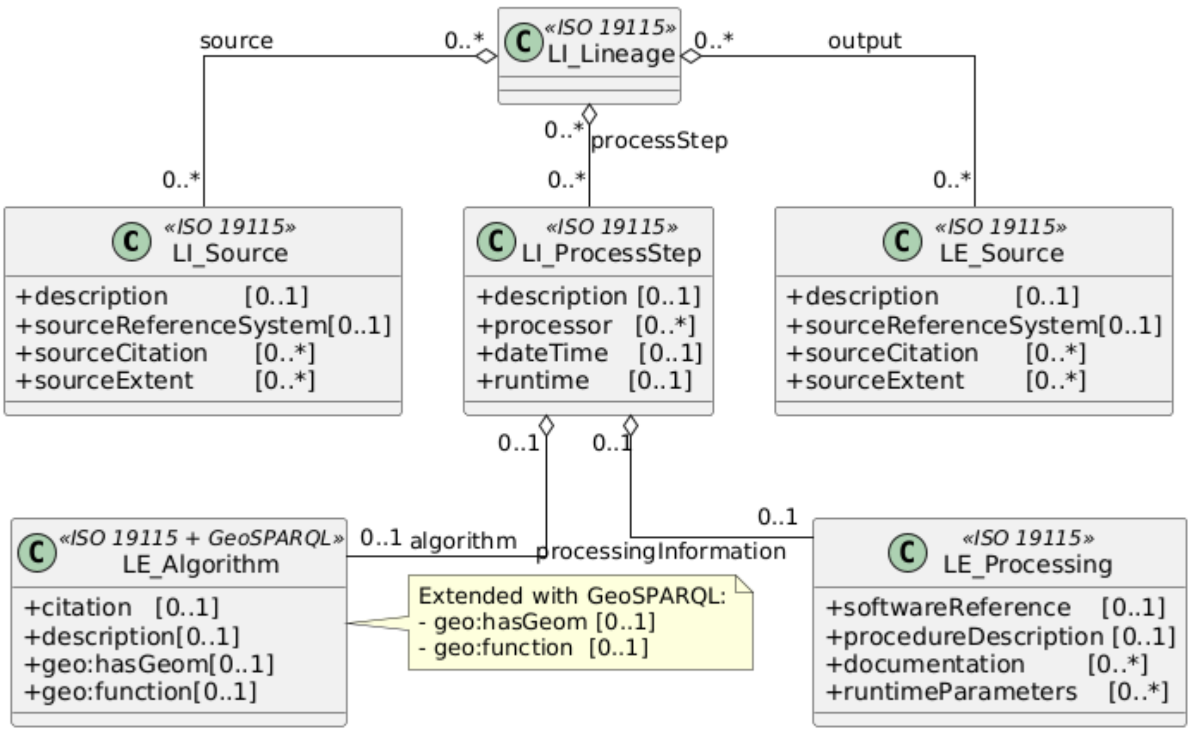


Figure 3.2 - ISO 19115-2 lineage model with GeoSPARQL Extensions

With the mapping in place, the parser (developed in Section 3.4) applies it during the lineage extraction process: as it iterates through the transformers in an FME workspace, it checks each transformer's type against the `GEO_REL_MAP`. If a match is found (i.e. the transformer has a GeoSPARQL equivalent), the parser augments that process step's metadata by inserting the GeoSPARQL function into the lineage record. Concretely, the parser creates an LE_Algorithm entry for the process step and adds two key pieces of information: the `geo:function` corresponding to the transform (e.g. `geof:buffer` for a Bufferer transformer), and the `geo:hasGeometry` property linking to the geometry or geometric parameter involved. By including both the operation and the geometry, we capture *what* was done and *to what* it was done, aligning with GeoSPARQL's modelling of geospatial operations. This semantic annotation is then embedded in the output lineage JSON. Only transformers that have a defined GeoSPARQL equivalent receive this treatment – ensuring that every spatial operation we can semantically describe is described using a standard term. In this way, the ISO lineage structure is complemented, not replaced: the LI_ProcessStep still records the basic details (e.g. a description of the step, timing, etc.), and now an LE_Algorithm within it carries the GeoSPARQL term that precisely characterizes the spatial operation performed. Figure 3.2 (from the conceptual model) illustrates this approach, showing how a standard ISO lineage model can be extended with GeoSPARQL concepts in the algorithm and processing elements.

Table 3.1 below presents a selection of the mapping between FME transformers and their equivalent GeoSPARQL functions (using the GeoSPARQL function namespace prefix geof. This sample demonstrates how common spatial transformers are semantically annotated in our model:

FME Transformer	Equivalent GeoSPARQL Function
Bufferer	geof:buffer
Clipper	geof:difference
Intersector	geof:intersection
Unioner	geof:union
AreaCalculator	geof:area
LengthCalculator	geof:length

Table 3.1: Examples of FME transformers and their corresponding GeoSPARQL functions. The GeoSPARQL function names (right column) provide a semantic definition of the operation performed by each FME transformer (left column).

In these examples, the semantic enrichment is straightforward: for instance, an FME Bufferer transformer (which creates a buffer polygon around a geometry) is annotated with the GeoSPARQL function geof:buffer, indicating the standard buffer operation as defined by OGC. Similarly, the Intersector transformer (finding geometric intersections) is linked to geof:intersection, and the AreaCalculator (computing the area of a geometry) corresponds to geof:area. By incorporating these terms, any lineage record exported from the system can be understood in terms of standard geospatial operations. This means that external systems or analysts familiar with GeoSPARQL can readily interpret the lineage: for example, they would know a step labelled `geof:buffer` is performing a buffer, even without deep knowledge of FME's internal naming. The full mapping covers 17 unique GeoSPARQL functions (see Appendix A for the complete list of mapped transformers), which include most of the frequently used spatial transformations in FME.

The process of annotating FME process steps with GeoSPARQL terms was implemented as part of the custom parser (detailed in Section 3.4). Conceptually, for each transformer encountered in an FME workspace file, the parser follows these steps: (1) retrieve the transformer's type/name, (2) consult the `GEO_REL_MAP` dictionary to see if a GeoSPARQL function is defined for that type, and (3) if so, insert the corresponding `geo:function` (and related geometry information) into the lineage metadata for that step. For example, if the parser reads a transformer of type "Bufferer", it finds `"Bufferer": "geof:buffer"` in the dictionary and thus adds an entry in the output JSON like: `"LE_Algorithm": { "geo:function": "geof:buffer", "geo:hasGeometry": [...] }` within that process step's record. This effectively tags the step with a semantic identifier.

While the GeoSPARQL integration markedly improves the semantic detail of the lineage model, there are some limitations and design decisions to acknowledge. For example, not all FME transformers have a direct equivalent in the GeoSPARQL standard. A few specialized transformers or those dealing with non-geometric data (e.g. attribute manipulation, format conversion) fall outside the GeoSPARQL vocabulary. In our implementation, such transformers are omitted from semantic mapping. Essentially this means that the parser filters out any transformer that lacks a GeoSPARQL definition when compiling the semantic lineage. This selective capture was intentional: it keeps the lineage graph focused on spatial operations (addressing the "granularity" concern that logging every single operation can overwhelm the user) and avoids mislabelling any process with an inexact term. The downside is a slight reduction in completeness but as the main focus is on spatial transformations this is acceptable.

The next section (Section 3.3) details the Metadata Capture and Parser Implementation, describing how FME workbench files and logs are parsed to automatically generate the lineage records following this model, and how the GeoSPARQL mapping is programmatically applied to real workflow data.

3.2. Technical Implementation

The parsers' structure outlines in figure 3.1 and then further explained in section 3.4. Figure 3.1 illustrates how design-time metadata (`.FMW`) and runtime logs (`.log`) feed into two parallel parsers, merge into a JSON store, and are visualized via a Flask app.

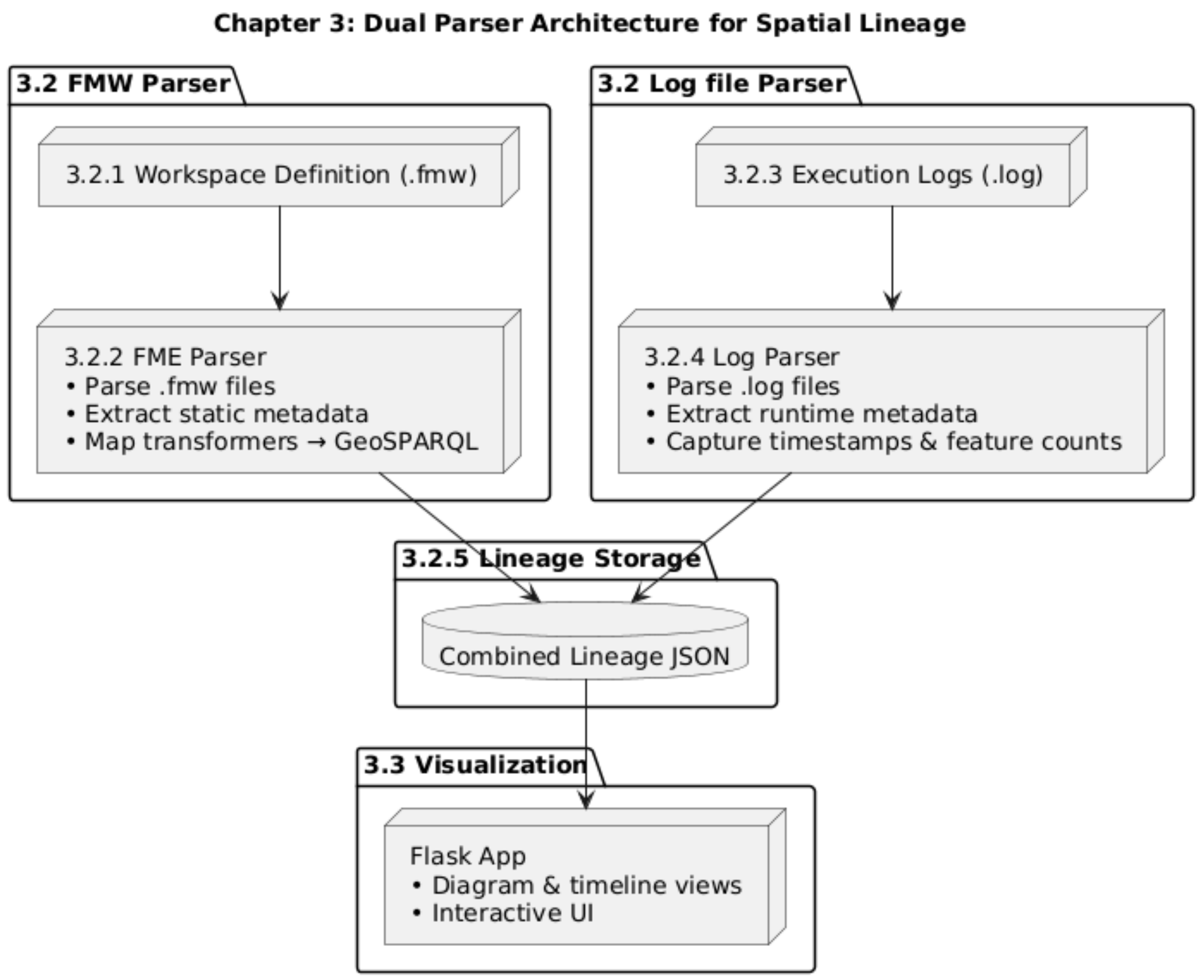

Chapter 3: Dual Parser Architecture for Spatial Lineage
3.2 FMW Parser
3.2.1 Workspace Definition (.fmw)
3.2.2 FME Parser
• Parse .fmw files
• Extract static metadata
• Map transformers → GeoSPARQL
3.2 Log file Parser
3.2.3 Execution Logs (.log)
3.2.4 Log Parser
• Parse .log files
• Extract runtime metadata
• Capture timestamps & feature counts
3.2.5 Lineage Storage
Combined Lineage JSON
3.3 Visualization
Flask App
• Diagram & timeline views
• Interactive UI

Figure 3.1 - Methodology flow for FME lineage model. Workflows defined in the .FMW and .log files are parsed in parallel, merged into a single JSON store, and visualized through the Flask app.

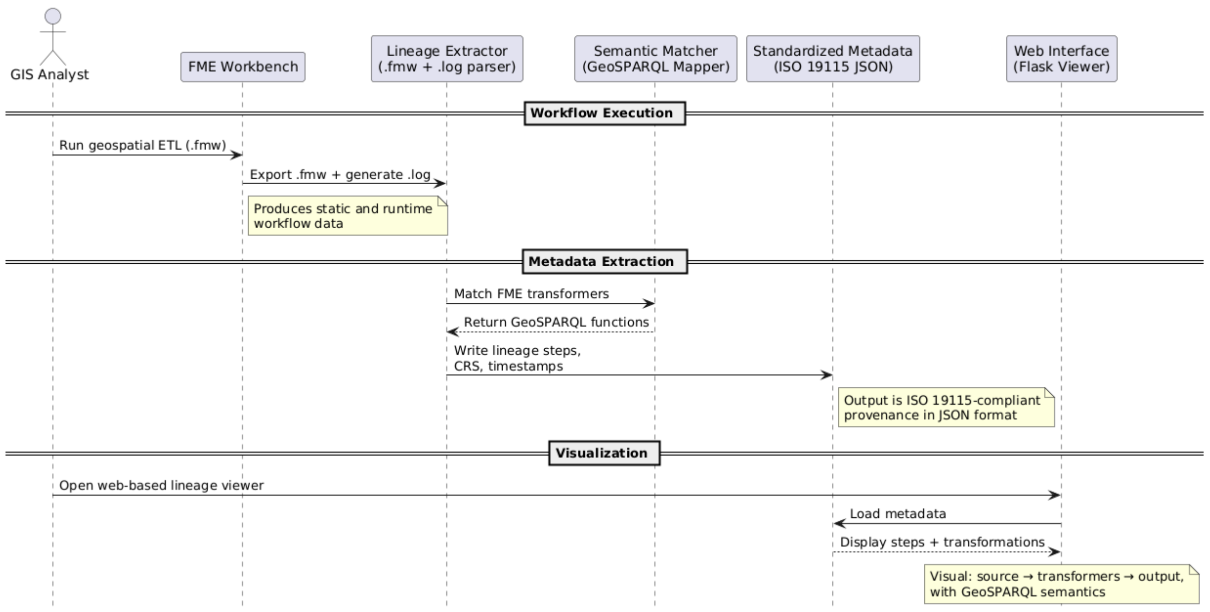

GIS Analyst
FME Workbench
Lineage Extractor (.fmw + .log parser)
Semantic Matcher (GeoSPARQL Mapper)
Standardized Metadata (ISO 19115 JSON)
Web Interface (Flask Viewer)
Workflow Execution
Run geospatial ETL (.fmw)
Export .fmw + generate .log
Produces static and runtime workflow data
Metadata Extraction
Match FME transformers
Return GeoSPARQL functions
Write lineage steps, CRS, timestamps
Output is ISO 19115-compliant provenance in JSON format
Visualization
Open web-based lineage viewer
Load metadata
Display steps + transformations
Visual: source → transformers → output, with GeoSPARQL semantics

Figure 3.2 - Overview of the workings of the parser

3.2.1. FMW Parser: Static Metadata Extraction

The first part of the tool is a parser that was designed to extract the static metadata of the FME workspace through parsing of the `.FMW` files. These files, as mentioned, are in the form of XML and define the readers, writers, and transformers used.

The FMW parser was created in Python and outputs a JSON structure aligning with the conceptual model created in section 3.3.

Input and Preprocessing

FME workspaces contain internal metadata lines prefixed by `#!` and `#`, which are not part of the valid XML tree. The parser removes these prefixes and truncates the content after the closing `</WORKSPACE>` tag to isolate the relevant structure:

```
# Remove comment lines and FME prefixes
if line.startswith('#! '):
    cleaned.append(line[3:])
elif line.startswith('#!'):
    cleaned.append(line[2:])
elif line.startswith('#'):
    continue
To ensure XML validity, a missing XML declaration is automatically added if not present:
if not xml_content.strip().startswith('<?xml'):
    xml_content = '<?xml version="1.0" encoding="UTF-8"?>\n' + xml_content
```

Parsing Strategy

The cleaned XML is parsed using `ElementTree.fromstring`, and traversal focuses on the `<TRANSFORMER>` elements within the `<TRANSFORMERS>` section. These are used to extract input sources, spatial process steps, and outputs.

Extracting Input Sources (LE_Source)

Input datasets are represented by `FeatureReader` transformers. The parser identifies them by filtering for the `TYPE="FeatureReader"` attribute and extracts relevant metadata such as name, source citation, and description:

```
if transformer.get('TYPE') == 'FeatureReader':
    for output_feat in transformer.findall('OUTPUT_FEAT'):
        if name and name not in ['<SCHEMA>', '<OTHER>', 'INITIATOR', '<REJECTED>']:
            readers.append({
                "name": name,
                "sourceCitation": ...,
                ...
            })
```

These are stored under the `LE_Source` element in the output JSON.

Extracting Spatial Process Steps (LI_ProcessStep)

The parser loops through all transformers and checks if each has a geospatial function by referencing a predefined dictionary `GEO_REL_MAP`, which maps FME transformer types to GeoSPARQL functions:

```
geo_relation = GEO_REL_MAP.get(ttype, None)
if geo_relation:
    lineage.append({
        "description": ttype,
        "LE_Algorithm": {
            "geo:function": geo_relation,
            ...
        },
        ...
    })
```

Each matching transformer becomes a `LI_ProcessStep`, enriched with two nested components:

- `LE_Algorithm`: holds the semantic operation using GeoSPARQL
- `LE_Processing`: holds technical metadata such as software version and identifier

Extracting Output Datasets (LI_Source)

Output datasets are identified via `FeatureWriter` transformers. Similar to the input extraction logic, metadata such as writer name, target dataset, and writer parameters are recorded:

```
if t.get('TYPE') == 'FeatureWriter':
    sources.append({
        "name": dataset_param.get('PARM_VALUE', ''),
        "description": parms.get('XFORMER_NAME', ''),
        ...
    })

These are stored in the second LE_Source entry, representing the output.
The resulting JSON structure is organized as follows:
"verversen.FMW": {
  "LE_Source": [
    {
      "name": "Beheerregio",
      "description": "FeatureReader",
      "sourceCitation": "$(DataDir)<solidus>*.mdb"
    }
  ],
  "LI_ProcessStep": [
    {
      "description": "GeometryExtractor",
      "LE_Algorithm": {
        "geo:function": "geof:asGeoJSON"
      },
      "LE_Processing": {
        "identifier": "TR001",
        "softwareReference": "FME 2023.1.0.0"
      }
    }
  ],
  "LI_Source": [
    {
      "description": "PostGISWriter",
      "sourceCitation": "stg_klein.invasieve_exoten"
    }
  ]
}
```

This output is saved to a JSON file and becomes the input for visualization and further processing.

3.2.2. Log Parser

While the FMW parser captures the overall structure of the FME workbench it does not provide the execution metadata that can only be obtained through log files after the FME workbench has been run. Therefore, another parser was developed that can parse the outputted log files.

Implementation Strategy

The parser uses Python's `re` module to identify relevant log lines using pattern matching. It is composed of three core functions:

Input Metadata Extraction

The `parse_input_dataset` function looks for lines that mention the FME workspace name and coordinate system. These values are used to populate the `LE_Source` entry with real-time metadata:

```
if "FME_MF_NAME" in msg:
    m_FMW = re.search(r"FME_MF_NAME is '([^']+\.FMW)'", msg)
    input_info['sourceCitation'] = m_FMW.group(1)
    input_info['description'] = 'FME Workspace'

if "|INFORM|Coordinate System" in line:
    m_coord = re.search(r"\|INFORM\|Coordinate System `([^`]+)'", line)
    input_info['sourceReferenceSystem'] = m_coord.group(1)
Timestamps are extracted from the beginning of each log line using:
timestamp = line.strip().split('|')[0].strip()
```

Process Step Extraction

The `parse_transformation_steps` function identifies the execution of FME transformers using a regular expression that matches log lines with transformer activity:

```
transformer_re = re.compile(r'^([A-Za-z0-9_]+?)\s*\(([^)]+)\)')
```

For each match, the parser checks whether the transformer name exists in the GeoSPARQL mapping (`FME_TO_GEOSPARQL`). If a match is found, the transformer is recorded as a process step with its runtime timestamp and semantic function:

```
for fme_name, geosparql_func in FME_TO_GEOSPARQL.items():
    if fme_name in step_name:
        geo_relation = geosparql_func
```

Only spatially meaningful transformations are retained by filtering for non-null `geoRelation` values.

Output Dataset Extraction

The `parse_output_dataset` function matches log lines containing writer activity, feature statistics, and target Tables. For example:

- Writer detection:

```
writer_re = re.compile(r'\b([A-Za-z0-9]+Writer)\b')
```

- Table name extraction:

```
Table_re = re.compile(r"Table '([^']+)'")
```

- Feature statistics:

```
stats_re = re.compile(r"STATS \|([^|]+)\s+([0-9,]+)")
```

All outputs are stored in a list, each enriched with its name, timestamp, coordinate system, and, where available, feature count.

An enriched `LI_ProcessStep` entry from the log parser might look like:

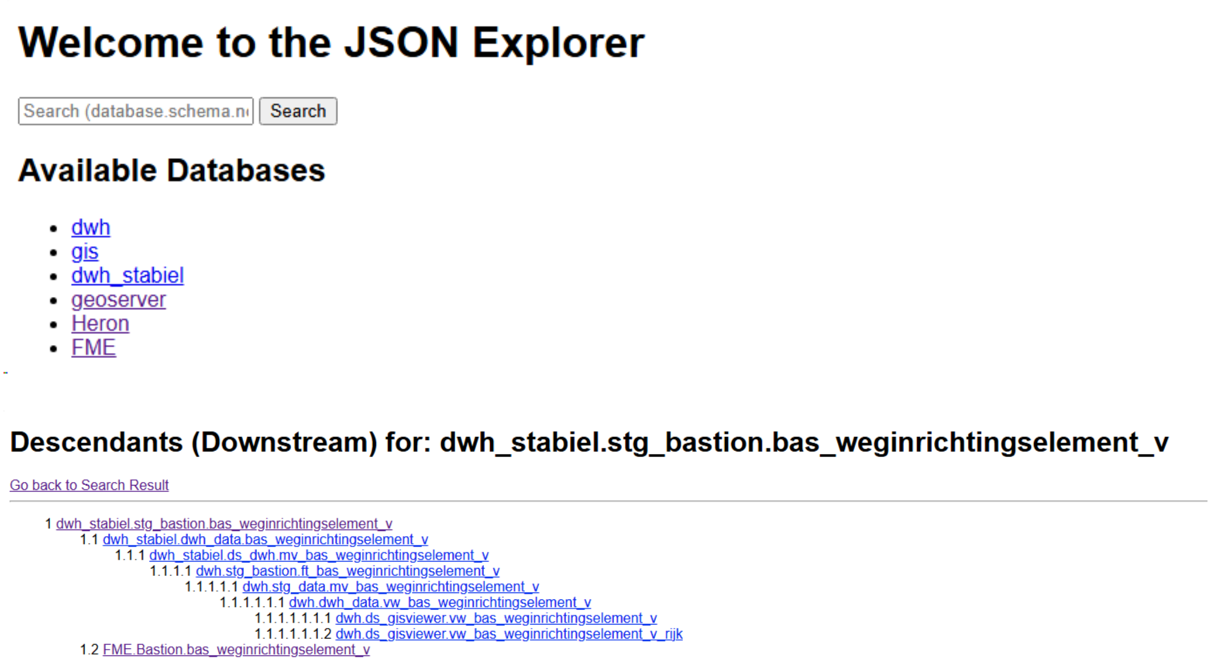
```
{
  "description": "LengthCalculator",
  "processor": "LengthCalculator",
  "geoRelation": "geof:length",
  "dateTime": "2024-10-30 14:35:52"
}
{
  "description": "PostGISWriter",
  "sourceCitation": "stg_klein.invasieve_exoten",
  "sourceReferenceSystem": "EPSG:28992",
  "feature_count": 3827,
  "dateTime": "2024-10-30 14:36:02"
}
```

3.2.3. Lineage storage and visualization

To improve accessibility for end users, a lightweight web-based visualization tool was developed using Flask. The application provides an interactive interface for browsing, inspecting, and interpreting the contents of the lineage model. It was originally created during my internship at the RVB to provide insight into the lineage across their entire Datawarehouse. Through the interface, users can view lineage trees tracing data flows from the Datawarehouse to GeoServer and ultimately to the Heron viewer (Gottenbos, 2025).

As part of this work, the application was extended to incorporate FME scripts into the lineage. This enhancement enables users to follow the complete flow of data from its source – as defined within the FME workspaces – through the Data Warehouse, and on to the final viewer. This provides a more complete representation of the technical workflow, connecting the ETL processes to their published outputs.

An example of the interface and a downstream search result is shown in Figure 3.3. Section 4 presents the Flask interface for each of the three individual examples, illustrating how these lineage views are delivered to end users.



Welcome to the JSON Explorer

Available Databases

- dwh
- gis
- dwh_stabiel
- geoserver
- Heron
- FME

Descendants (Downstream) for: dwh_stabiel.stg_bastion.bas_weginrichtingselement_v

Go back to Search Result

```
1 dwh_stabiel.stg_bastion.bas_weginrichtingselement_v
    1.1 dwh_stabiel.dwh_data.bas_weginrichtingselement_v
        1.1.1 dwh_stabiel.ds_dwh.mv_bas_weginrichtingselement_v
            1.1.1.1 dwh.stg_bastion.ft_bas_weginrichtingselement_v
                1.1.1.1.1 dwh.stg_data.mv_bas_weginrichtingselement_v
                    1.1.1.1.1.1 dwh.dwh_data.vw_bas_weginrichtingselement_v
                        1.1.1.1.1.1.1 dwh.ds_gisviewer.vw_bas_weginrichtingselement_v
                        1.1.1.1.1.1.2 dwh.ds_gisviewer.vw_bas_weginrichtingselement_v_rijk
    1.2 FME.Bastion.bas_weginrichtingselement_v
```

Figure 3.3 – Example of Flask application visualization the lineage data

4 Results and Evaluation

This section presents the extracted data lineage structures generated by the dual-parser implementation described in Chapter 3. The goal is to empirically demonstrate how FME-based spatial ETL workflows can be transformed into semantically enriched, standards-aligned provenance records. By aligning transformer outputs with ISO 19115 and GeoSPARQL, the model supports auditability, traceability, and machine-readable workflow reconstruction. The output structures are evaluated across three use cases, each designed to test the model's ability to capture both declarative and operational metadata.

4.1 Overview of Selected Workflows

To evaluate the practical application of the proposed lineage model, a set of representative FME workflows from the RVB were selected. These workflows serve as test cases to assess the model's ability to capture, document, and visualize spatial data transformations in operational settings. The selected cases differ in spatial complexity, governance relevance, and transformation logic, allowing for a full evaluation of the parser's accuracy and generalizability. The cases were evaluated based on the evaluation criteria defined in section 2.5.

The lineage model outputs a hierarchical JSON structure consisting of three primary entities, derived from ISO 19115-2 and extended with GeoSPARQL semantics:

- **LE_Source**: Describes input datasets, including identifiers, coordinate reference systems (CRS), source citations, and optional timestamps. In FME this is representative of the reader. Is not native to ISO 19115 however this helps to distinguish the input from the output.
- **LI_Source:** Describes the output dataset, including identifiers, coordinate reference systems (CRS), source citations, and optional timestamps. This maps to FME's readers.
- **LI_ProcessStep**: Captures the spatial operations performed, including a textual description, the transformer identifier, execution time (where available), and a semantic mapping to a corresponding spatial function. This maps to FME's transformers.
- **LE_Algorithm / LE_Processing**: These subcomponents detail the semantic logic (`geo:function`) and software environment (e.g., transformer name, version, runtime parameters).

GeoSPARQL mappings are embedded under `LE_Algorithm.geo:function`, enabling the functional intent of FME transformers to be represented in a queryable and standards-compliant form. A representative excerpt is shown below:

```
{
  "LI_ProcessStep": [
    {
      "description": "AreaCalculator",
      "LE_Algorithm": {
        "geo:function": "geof:area"
      },
      "LE_Processing": {
        "identifier": "TR_004",
        "softwareReference": "FME 2023.1"
      },
      "dateTime": "2025-06-04 10:17:13"
    }
  ]
}
```

This output confirms that both procedural lineage (i.e., which operation occurred) and semantic lineage (i.e., what type of transformation was applied) are captured, supporting cross-platform interoperability and governance validation.

4.1.1 Verversen Regio Indeling Groenbeheer (VRIG)

This FME workflow is used to renew the data for the green spaces which are governed by the Dutch government. This case is selected because it plays a key role in the weekly operations of the RVB as stakeholders use this data to establish policies surrounding these green spaces.

The extracted lineage was structured according to the framework, with the three core data blocks:

- LE_Source: Captures the input dataset, with its CRS of Netherlands-RDNew-2008.
- LI_ProcessesSteps: Each transformer which has a correlating GeoSPARQL definition was parsed, including the functions and the date time stamps.
- LI_Source: Represents the output dataset including where it is written to with its CRS extent if applicable. For this case that is master data Regio Indeling Groenbeheer in the Datawarehouse GIS.

The GeoSPARQL definition mapping was successful, and the following spatial operations were detected and mapped.

Transformer	GeoSPARQL definition	Description
Aggregator	Geof:aggCentroid	Aggregates geometries
Geometry Extractor	Geof:geoJSON	Extracts geometries

The JSON was visualized according to the enriched ISO framework created introduced in section 3.2. Figure 4.1 shows the FME workbench that will be parsed. Figure 4.2 Is the UML diagram of the parsed lineage, showing how the LE_Source flows through the LI_processSteps to the LI_Source. An example of the JSON output has been provided below; the full JSON script can be found in the Appendix C. Figure 4.3 shows the Flask interface example of the parsed JSON that users can view and click through.

```
{
    "Verversen_attributen_Regioindeling_Groenbeheer.FMW": {
        "LE_Source": [
            {
                "name": "Beheerregio",
                "sourceReferenceSystem": "Netherlands-RDNew-2008",
                "description": "FeatureReader",
                "sourceCitation": "$(DataDir)<solidus>*.mdb",
                "dateTime": "2025-05-28 15:04:13"
            }
        ],
        "LI_ProcessStep": [
            {
                "description": "Aggregates geometries.",
                "processor": "Aggregator_2",
                "dateTime": "2025-05-28 15:04:13",
                "LE_Algorithm": {
                    "citation": "FME",
                    "description": "Aggregates geometries.",
                    "geo:hasGeom": "Yes",
                    "geo:function": "geo:aggCentroid"
                },
                "LE_Processing": {
                    "softwareReference": "FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)",
                    "procedureDescription": "Processing step for geospatial data",
                    "documentation": "Aggregates geometries.",
                    "runtimeParameters": {
                        "runtime": 0.0,
                        "startTime": "2025-05-28 15:04:13",
                        "endTime": "2025-05-28 15:04:13"
                    }
                },
```

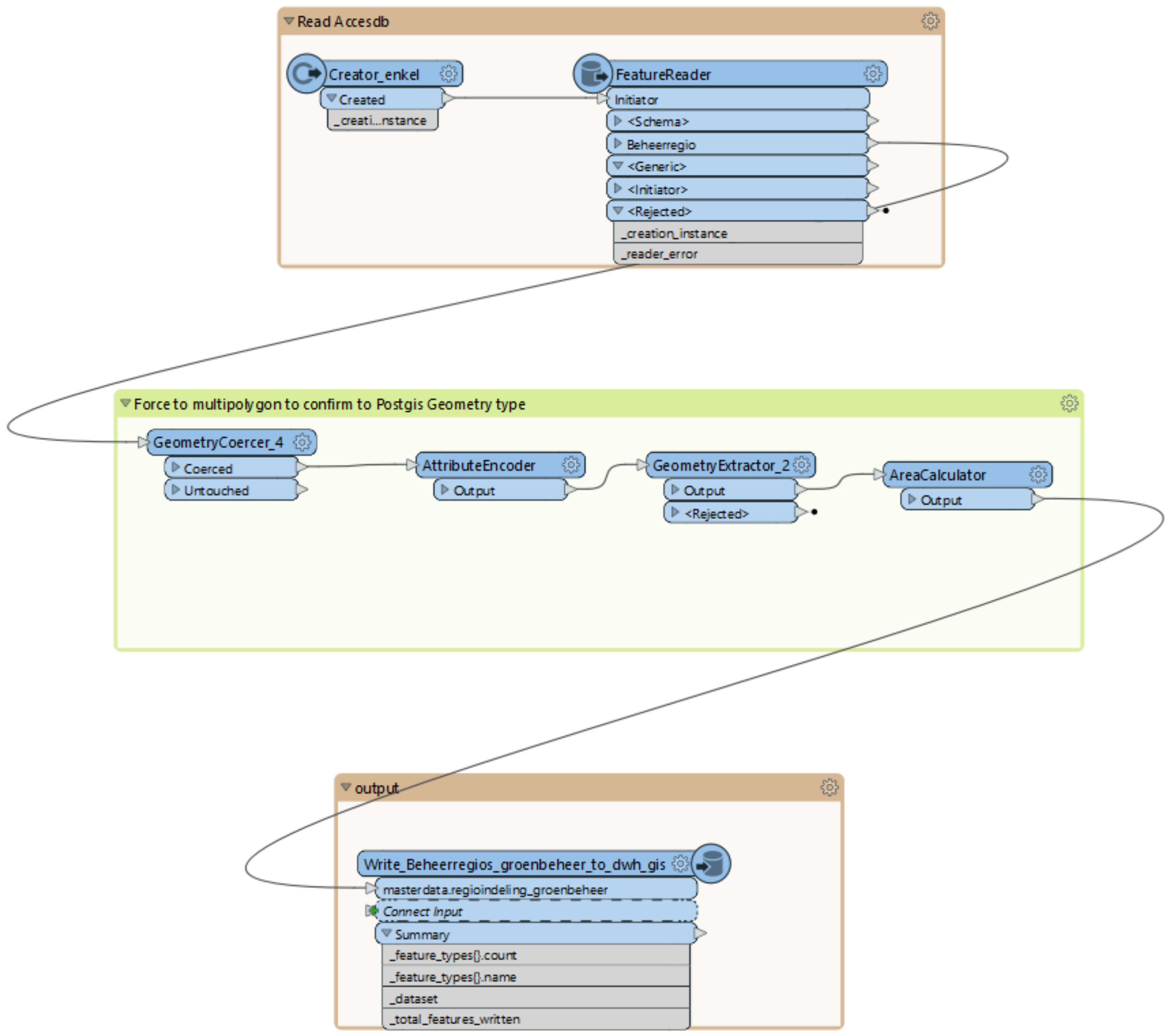


Figure 4.1 – Regio-indeling Groenbeheer FME workbench

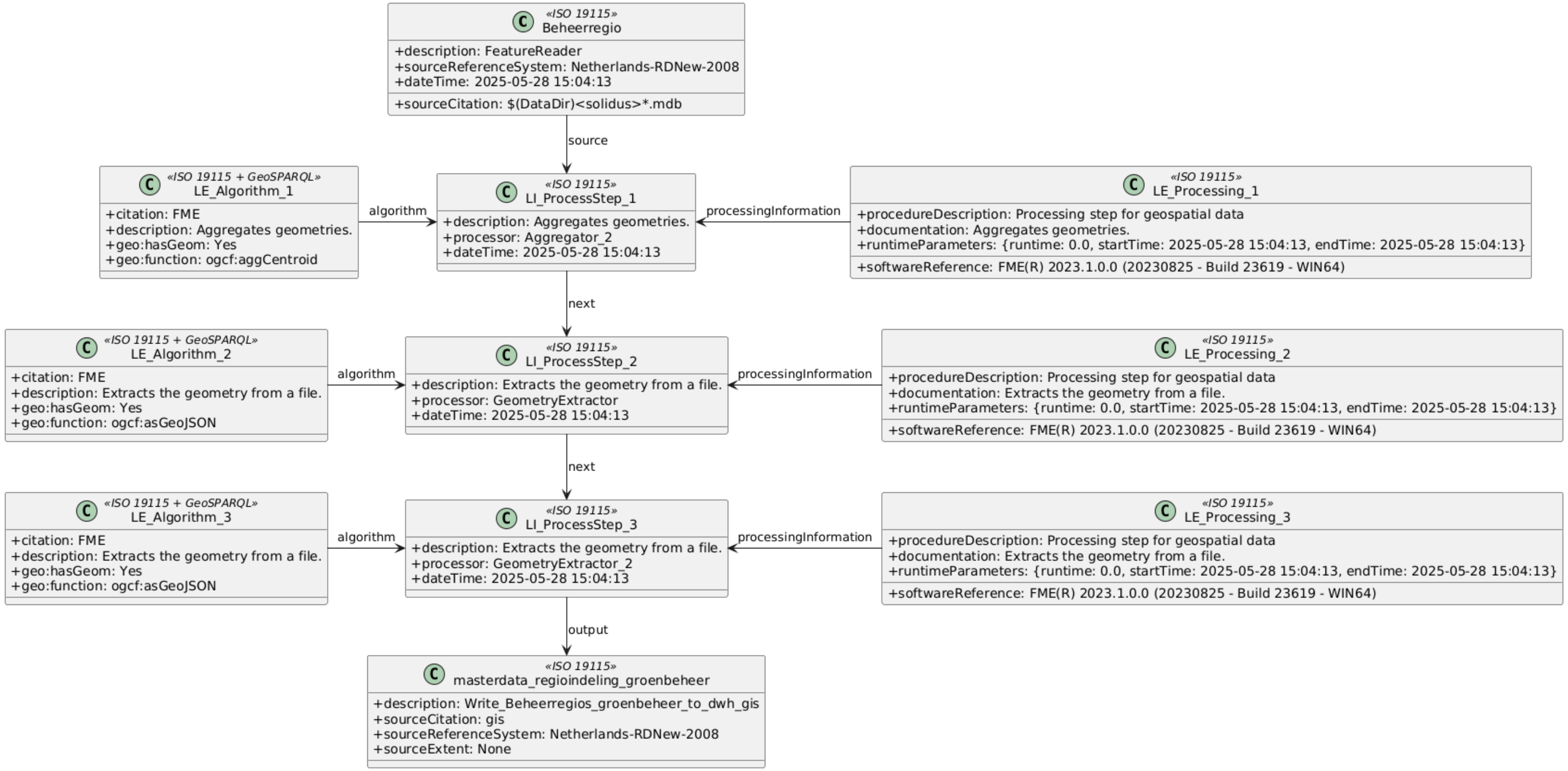
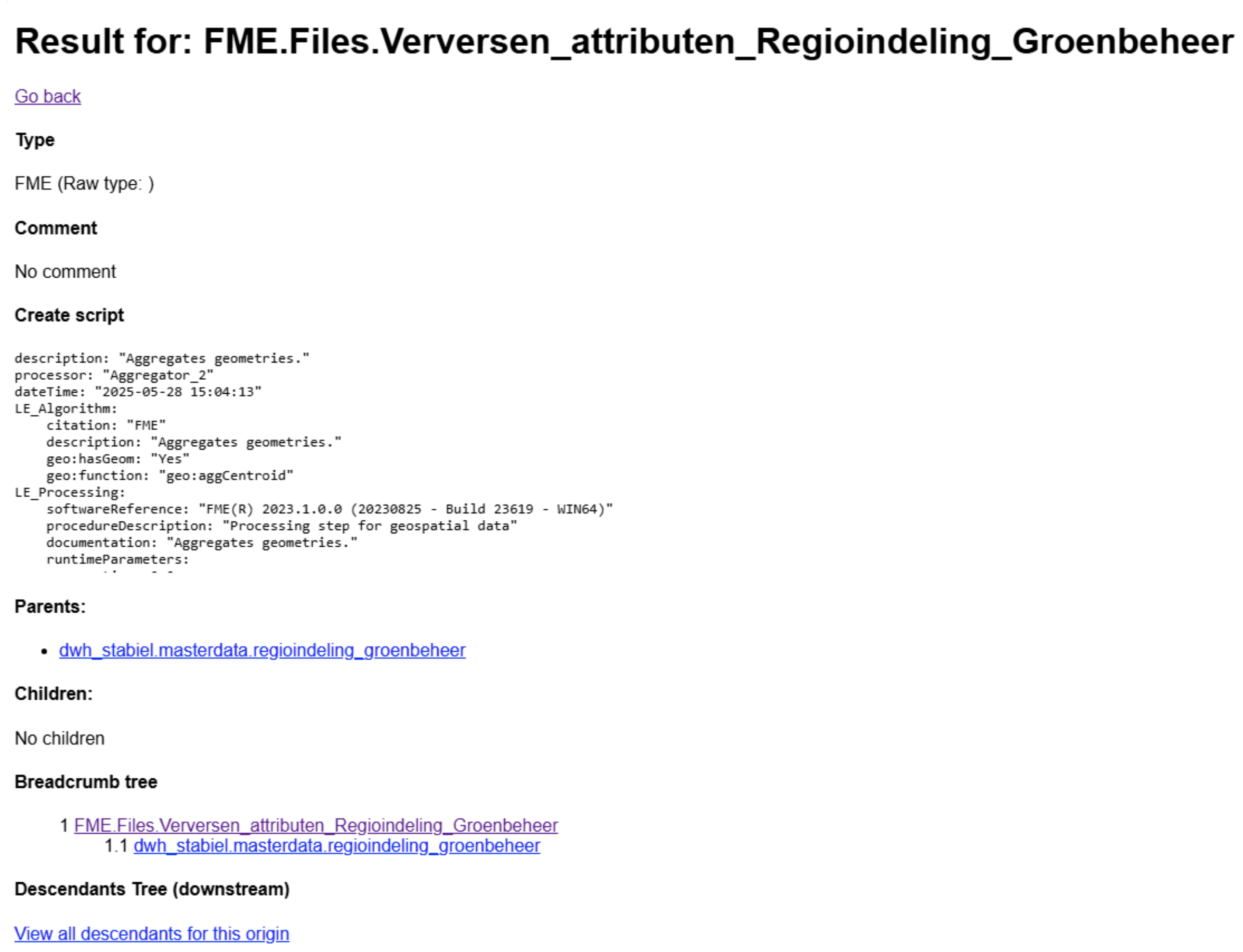

«ISO 19115» Beheerregio
+description: FeatureReader
+sourceReferenceSystem: Netherlands-RDNew-2008
+dateTime: 2025-05-28 15:04:13
+sourceCitation: $(DataDir)<solidus>*.mdb
source
«ISO 19115 + GeoSPARQL» LE_Algorithm_1
+citation: FME
+description: Aggregates geometries.
+geo:hasGeom: Yes
+geo:function: ogcf:aggCentroid
algorithm
«ISO 19115» LI_ProcessStep_1
+description: Aggregates geometries.
+processor: Aggregator_2
+dateTime: 2025-05-28 15:04:13
processingInformation
«ISO 19115» LE_Processing_1
+procedureDescription: Processing step for geospatial data
+documentation: Aggregates geometries.
+runtimeParameters: {runtime: 0.0, startTime: 2025-05-28 15:04:13, endTime: 2025-05-28 15:04:13}
+softwareReference: FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)
next
«ISO 19115 + GeoSPARQL» LE_Algorithm_2
+citation: FME
+description: Extracts the geometry from a file.
+geo:hasGeom: Yes
+geo:function: ogcf:asGeoJSON
algorithm
«ISO 19115» LI_ProcessStep_2
+description: Extracts the geometry from a file.
+processor: GeometryExtractor
+dateTime: 2025-05-28 15:04:13
processingInformation
«ISO 19115» LE_Processing_2
+procedureDescription: Processing step for geospatial data
+documentation: Extracts the geometry from a file.
+runtimeParameters: {runtime: 0.0, startTime: 2025-05-28 15:04:13, endTime: 2025-05-28 15:04:13}
+softwareReference: FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)
next
«ISO 19115 + GeoSPARQL» LE_Algorithm_3
+citation: FME
+description: Extracts the geometry from a file.
+geo:hasGeom: Yes
+geo:function: ogcf:asGeoJSON
algorithm
«ISO 19115» LI_ProcessStep_3
+description: Extracts the geometry from a file.
+processor: GeometryExtractor_2
+dateTime: 2025-05-28 15:04:13
processingInformation
«ISO 19115» LE_Processing_3
+procedureDescription: Processing step for geospatial data
+documentation: Extracts the geometry from a file.
+runtimeParameters: {runtime: 0.0, startTime: 2025-05-28 15:04:13, endTime: 2025-05-28 15:04:13}
+softwareReference: FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)
output
«ISO 19115» masterdata_regioindeling_groenbeheer
+description: Write_Beheerregios_groenbeheer_to_dwh_gis
+sourceCitation: gis
+sourceReferenceSystem: Netherlands-RDNew-2008
+sourceExtent: None

Figure 4.2 – Regio Indeling Groenbeheer UML diagram of parsed lineage f



Result for: FME.Files.Verversen_attributen_Regioindeling_Groenbeheer

Go back

Type

FME (Raw type:)

Comment

No comment

Create script

```
description: "Aggregates geometries."
processor: "Aggregator_2"
dateTime: "2025-05-28 15:04:13"
LE_Algorithm:
    citation: "FME"
    description: "Aggregates geometries."
    geo:hasGeom: "Yes"
    geo:function: "geo:aggCentroid"
LE_Processing:
    softwareReference: "FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)"
    procedureDescription: "Processing step for geospatial data"
    documentation: "Aggregates geometries."
    runtimeParameters:
```

Parents:

- dwh_stabiel.masterdata.regioindeling_groenbeheer

Children:

No children

Breadcrumb tree

1 FME.Files.Verversen_attributen_Regioindeling_Groenbeheer
    1.1 dwh_stabiel.masterdata.regioindeling_groenbeheer

Descendants Tree (downstream)

View all descendants for this origin

Figure 4.3 – Regio Indeling Groenbeheer Flask visualization

4.1.2 Capaciteitskaart Elekriciteitsnet (CE)

This workflow is one that is more complex than the Regio Indeling Groenbeheer which is the reason it was chosen to serve as an example of the tool. It has multiple inputs and multiple outputs and serves the RVB through providing information on areas of congestion in the electricity lines the ingoing lines and the outgoing powerlines.

The extracted lineage was structured according to the framework, with the three core data blocks:

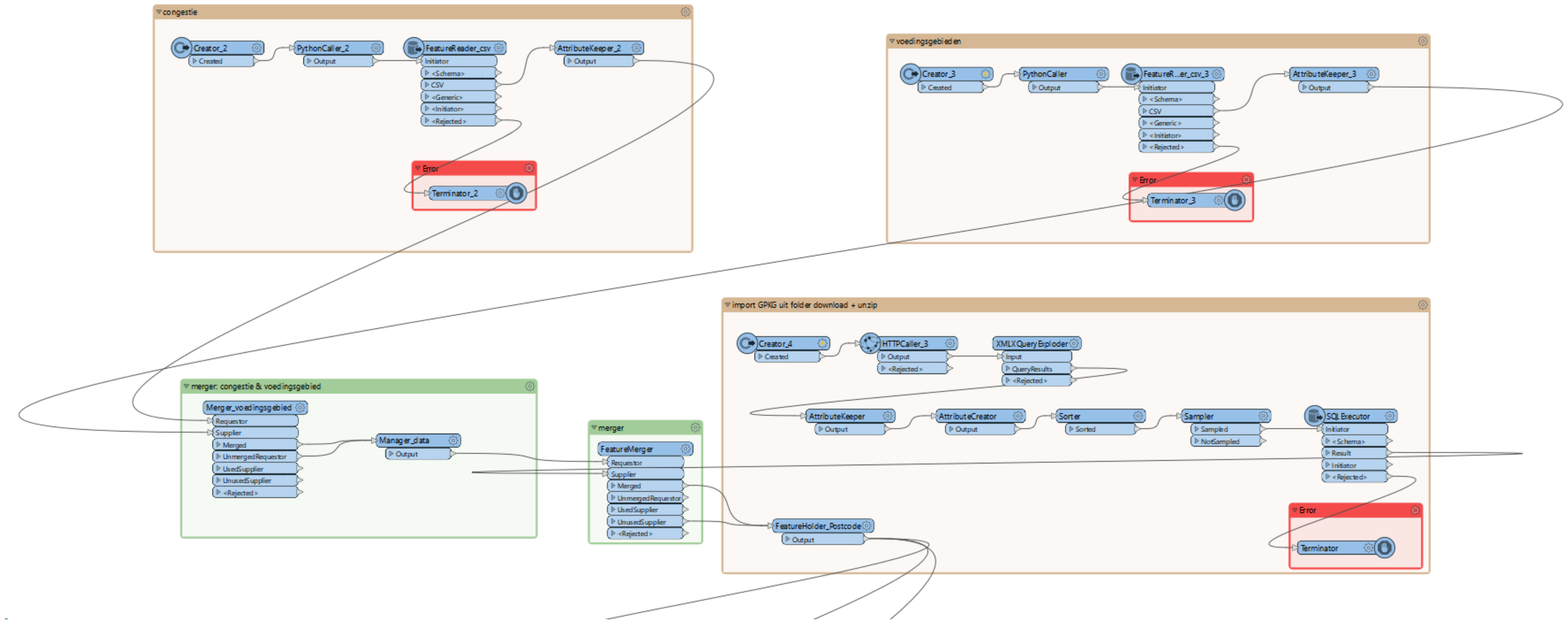
- LE_Source: Captures the input dataset, with its CRS of Netherlands-RDNew-2008.
- LI_ProcessesSteps: Each transformer which has a correlating GeoSPARQL definition was parsed, including the functions and the date time stamps.
- LI_Source: Represents the output dataset including where it is written to with its CRS extent if applicable. For this case that is three separate outputs

The GeoSPARQL definition mapping was successful, and the following spatial operations were detected and mapped.

Transformer	GeoSPARQL definition	Description
Aggregator	geo:aggcentroid	Aggregates geometries
Dissolver	geo:union	Dissolves (Unions) overlapping geometries into one.

The FME workbench that was parsed can be seen in figure 4.4. This data was the visualized in a UML diagram that visualizes the JSON structure according to the ISO framework. This can be seen in figure 4.5. This figure was simplified for this report as the full figure is too large to add, two of the six processing steps were chosen to visualize the overall processes. An example of the JOSN output can be seen below and the full JSON can be found in appendix B. the Flask interface for this parsed lineage can be seen in figure 4.6.

```
    "Capaciteitskaart_Elektriciteitsnet.FMW": {
        "LE_Source": [
            {
                "name": "CSV",
                "sourceReferenceSystem": "Netherlands-RDNew-2008",
                "description": "FeatureReader",
                "sourceCitation": "f:\\dataservices\\data\\BP23\\congestie_pc6.csv",
                "dateTime": "2025-06-02 13:20:47"
            },
        ],
        "LI_ProcessStep": [
            {
                "description": "Dissolves (unions) overlapping or contiguous geometries into
                "processor": "Dissolver_voedingsgebied",
                "dateTime": "2025-06-02 13:18:19",
                "LE_Algorithm": {
                    "citation": "FME",
                    "description": "Dissolves (unions) overlapping or contiguous geometries into
                    "geo:hasGeom": "Yes",
                    "geo:function": "geo:union"
```


congestie
Creator_2
Created
PythonCaller_2
Output
FeatureReader_csv
Initiator
<Schema>
CSV
<Generic>
<Initiator>
<Rejected>
AttributeKeeper_2
Output
Error
Terminator_2
voedingsgebieden
Creator_3
Created
PythonCaller
Output
FeatureR...er_csv_3
Initiator
<Schema>
CSV
<Generic>
<Initiator>
<Rejected>
AttributeKeeper_3
Output
Error
Terminator_3
import GPKG uit folder download + unzip
Creator_4
Created
HTTPCaller_3
Output
<Rejected>
XMLXQueryExploder
Input
QueryResults
<Rejected>
AttributeKeeper
Output
AttributeCreator
Output
Sorter
Sorted
Sampler
Sampled
NotSampled
SQLExecutor
Initiator
<Schema>
Result
Initiator
<Rejected>
Error
Terminator
FeatureHolder_Postcode
Output
merger: congestie & voedingsgebied
Merger_voedingsgebied
Requestor
Supplier
Merged
UnmergedRequestor
UsedSupplier
UnusedSupplier
<Rejected>
Manager_data
Output
merger
FeatureMerger
Requestor
Supplier
Merged
UnmergedRequestor
UsedSupplier
UnusedSupplier
<Rejected>

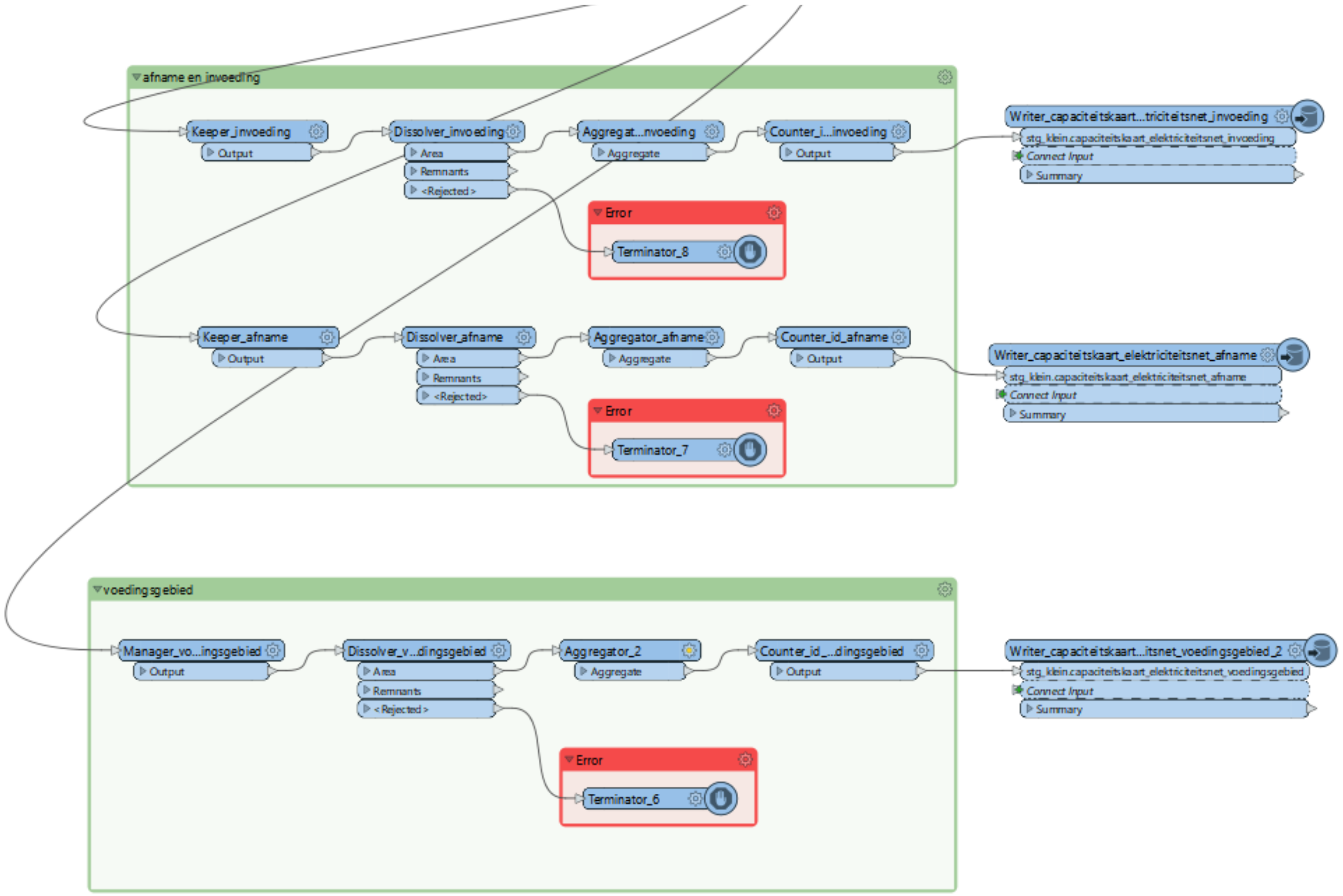


Figure 4.4 - Capaciteitskaart Elektriciteitsnet FME workbench

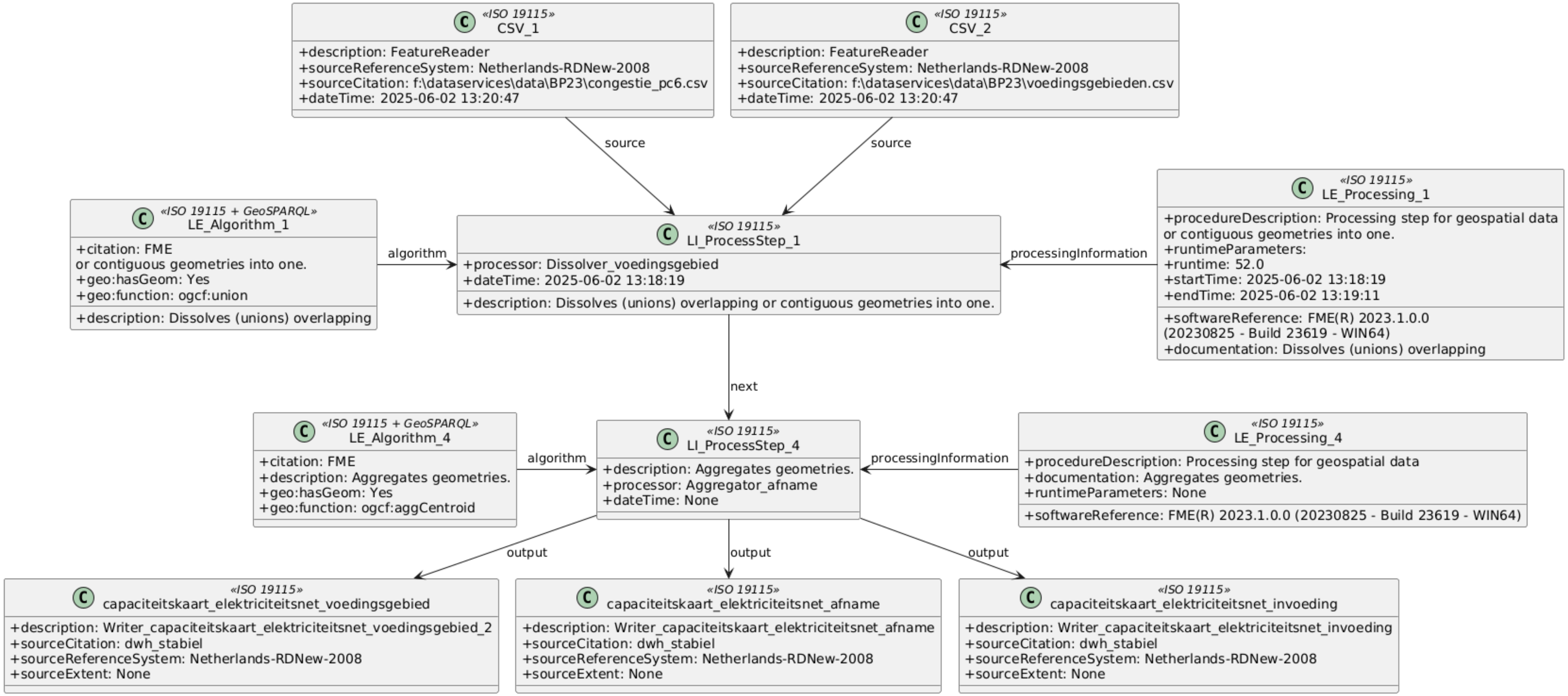
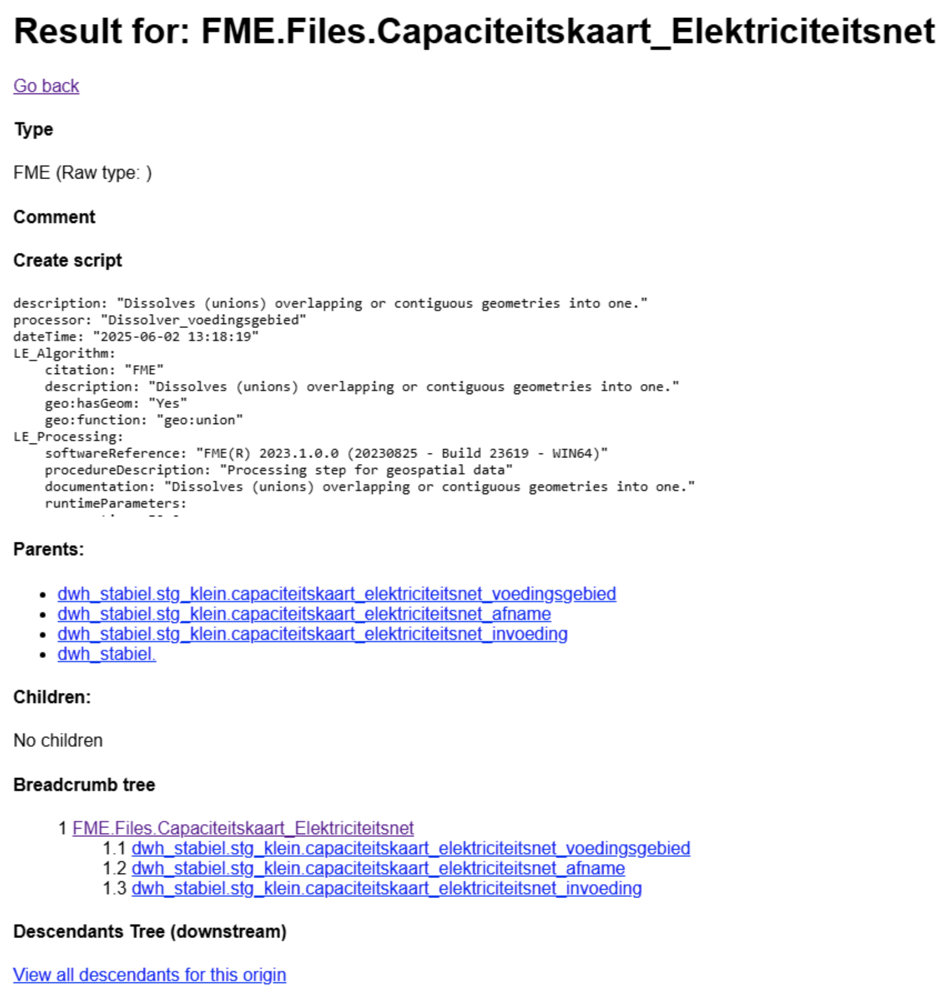

«ISO 19115»
CSV_1
+description: FeatureReader
+sourceReferenceSystem: Netherlands-RDNew-2008
+sourceCitation: f:\dataservices\data\BP23\congestie_pc6.csv
+dateTime: 2025-06-02 13:20:47
«ISO 19115»
CSV_2
+description: FeatureReader
+sourceReferenceSystem: Netherlands-RDNew-2008
+sourceCitation: f:\dataservices\data\BP23\voedingsgebieden.csv
+dateTime: 2025-06-02 13:20:47
source
source
«ISO 19115 + GeoSPARQL»
LE_Algorithm_1
+citation: FME
or contiguous geometries into one.
+geo:hasGeom: Yes
+geo:function: ogcf:union
+description: Dissolves (unions) overlapping
algorithm
«ISO 19115»
LI_ProcessStep_1
+processor: Dissolver_voedingsgebied
+dateTime: 2025-06-02 13:18:19
+description: Dissolves (unions) overlapping or contiguous geometries into one.
processingInformation
«ISO 19115»
LE_Processing_1
+procedureDescription: Processing step for geospatial data
or contiguous geometries into one.
+runtimeParameters:
+runtime: 52.0
+startTime: 2025-06-02 13:18:19
+endTime: 2025-06-02 13:19:11
+softwareReference: FME(R) 2023.1.0.0
(20230825 - Build 23619 - WIN64)
+documentation: Dissolves (unions) overlapping
next
«ISO 19115 + GeoSPARQL»
LE_Algorithm_4
+citation: FME
+description: Aggregates geometries.
+geo:hasGeom: Yes
+geo:function: ogcf:aggCentroid
algorithm
«ISO 19115»
LI_ProcessStep_4
+description: Aggregates geometries.
+processor: Aggregator_afname
+dateTime: None
processingInformation
«ISO 19115»
LE_Processing_4
+procedureDescription: Processing step for geospatial data
+documentation: Aggregates geometries.
+runtimeParameters: None
+softwareReference: FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)
output
output
output
«ISO 19115»
capaciteitskaart_elektriciteitsnet_voedingsgebied
+description: Writer_capaciteitskaart_elektriciteitsnet_voedingsgebied_2
+sourceCitation: dwh_stabiel
+sourceReferenceSystem: Netherlands-RDNew-2008
+sourceExtent: None
«ISO 19115»
capaciteitskaart_elektriciteitsnet_afname
+description: Writer_capaciteitskaart_elektriciteitsnet_afname
+sourceCitation: dwh_stabiel
+sourceReferenceSystem: Netherlands-RDNew-2008
+sourceExtent: None
«ISO 19115»
capaciteitskaart_elektriciteitsnet_invoeding
+description: Writer_capaciteitskaart_elektriciteitsnet_invoeding
+sourceCitation: dwh_stabiel
+sourceReferenceSystem: Netherlands-RDNew-2008
+sourceExtent: None

Figure 4.5 – UML diagram of the parsed lineage of Capaciteitskaart Elektriciteitsnet



Result for: FME.Files.Capaciteitskaart_Elektriciteitsnet

Go back

Type

FME (Raw type:)

Comment

Create script

```
description: "Dissolves (unions) overlapping or contiguous geometries into one."
processor: "Dissolver_voedingsgebied"
dateTime: "2025-06-02 13:18:19"
LE_Algorithm:
    citation: "FME"
    description: "Dissolves (unions) overlapping or contiguous geometries into one."
    geo:hasGeom: "Yes"
    geo:function: "geo:union"
LE_Processing:
    softwareReference: "FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)"
    procedureDescription: "Processing step for geospatial data"
    documentation: "Dissolves (unions) overlapping or contiguous geometries into one."
    runtimeParameters:
```

Parents:

- dwh_stabiel.stg_klein.capaciteitskaart_elektriciteitsnet_voedingsgebied
- dwh_stabiel.stg_klein.capaciteitskaart_elektriciteitsnet_afname
- dwh_stabiel.stg_klein.capaciteitskaart_elektriciteitsnet_invoeding
- dwh_stabiel.

Children:

No children

Breadcrumb tree

1 FME.Files.Capaciteitskaart_Elektriciteitsnet
  1.1 dwh_stabiel.stg_klein.capaciteitskaart_elektriciteitsnet_voedingsgebied
  1.2 dwh_stabiel.stg_klein.capaciteitskaart_elektriciteitsnet_afname
  1.3 dwh_stabiel.stg_klein.capaciteitskaart_elektriciteitsnet_invoeding

Descendants Tree (downstream)

View all descendants for this origin

Figure 4.6 - Flask interface of the data lineage of Capaciteitskaart Elektriciteitsnet

4.1.3 Zon op Dak data naar PostGIS (ZoD)

This FME workflow is used to renew the data for the theoretical and space on the roofs of Dutch houses for solar panels currently. This case is selected because it shows the variation in the types of datasets that are used on a daily basis in the RVB.

The GeoSPARQL definition mapping was successful, and the following spatial operations were detected and mapped.

Transformer	GeoSPARQL definition	Description
FeatureJoiner	geof:join	Joins datasets

The FME workbench that was parsed can be seen in figure 4.7. The JSON was visualized according to the enriched ISO framework created introduced in section 3.2. An example can be seen in figure 4.8. An example of the JSON output has been provided below; the full JSON script can be found in the Appendix D. The Flask-based interface for this output was visualized and can be seen in figure 4.9.

```
{
    "ZonOpDak_data_naar_PostGIS.FMW": {
        "LE_Source": [
            {
                "name": "CSV",
                "sourceReferenceSystem": "Netherlands-RDNew-2008",
                "description": "FeatureReader",
                "sourceCitation": "F:\\dataservices\\data\\BP07\\1.csv",
                "dateTime": "2025-08-05 12:19:13"
            },
            {
                "name": "CSV",
                "sourceReferenceSystem": "Netherlands-RDNew-2008",
                "description": "FeatureReader",
                "sourceCitation": "F:\\dataservices\\data\\BP07\\1.csv",
                "dateTime": "2025-08-05 12:19:13"
            },
            {
                "name": "stg_data.mv_bouwwerk",
                "sourceReferenceSystem": "Netherlands-RDNew-2008",
                "description": "FeatureReader",
                "sourceCitation": "F:\\dataservices\\data\\BP07\\2.csv",
                "dateTime": "2025-08-05 12:19:13"
            }
```

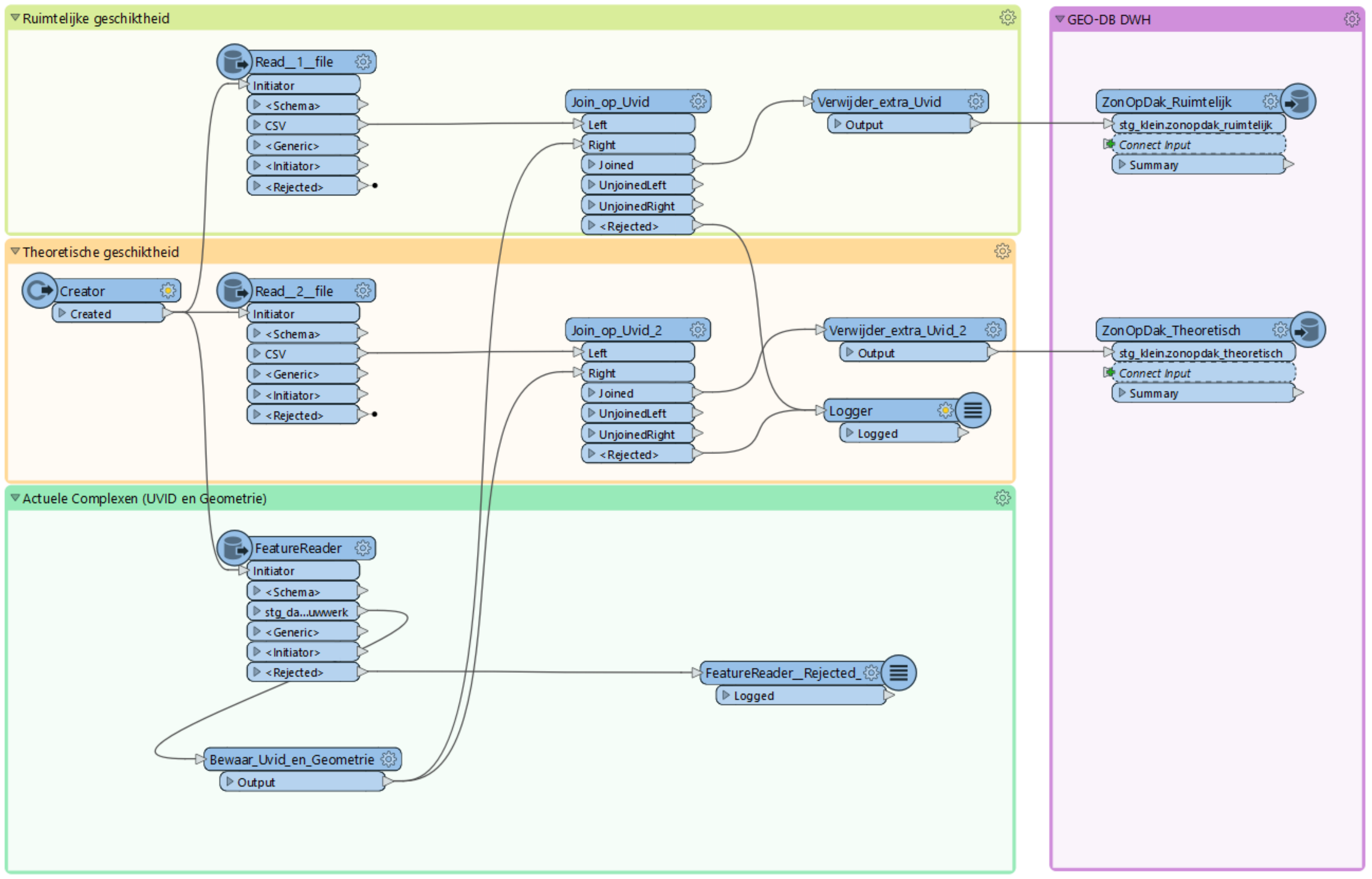


Figure 4.7 – Zon op Dak naar PostGIS FME workbench

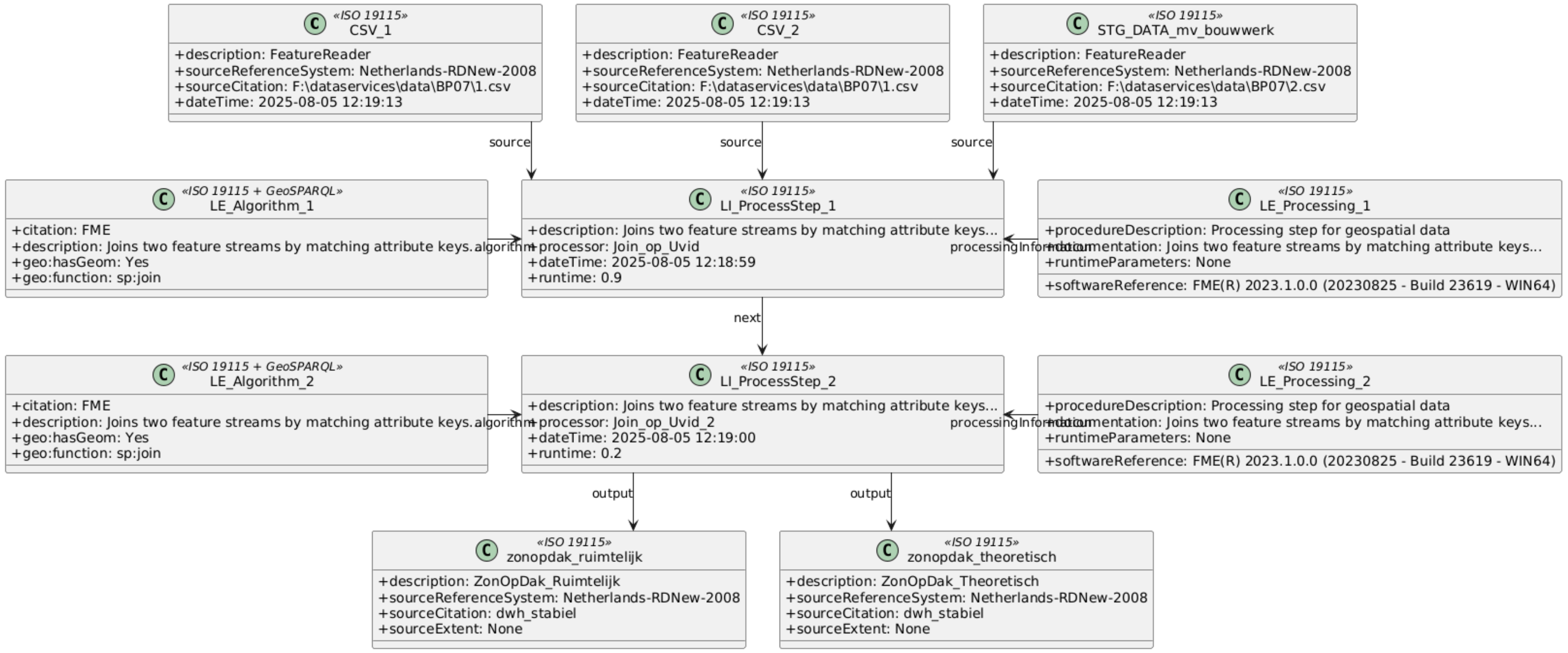
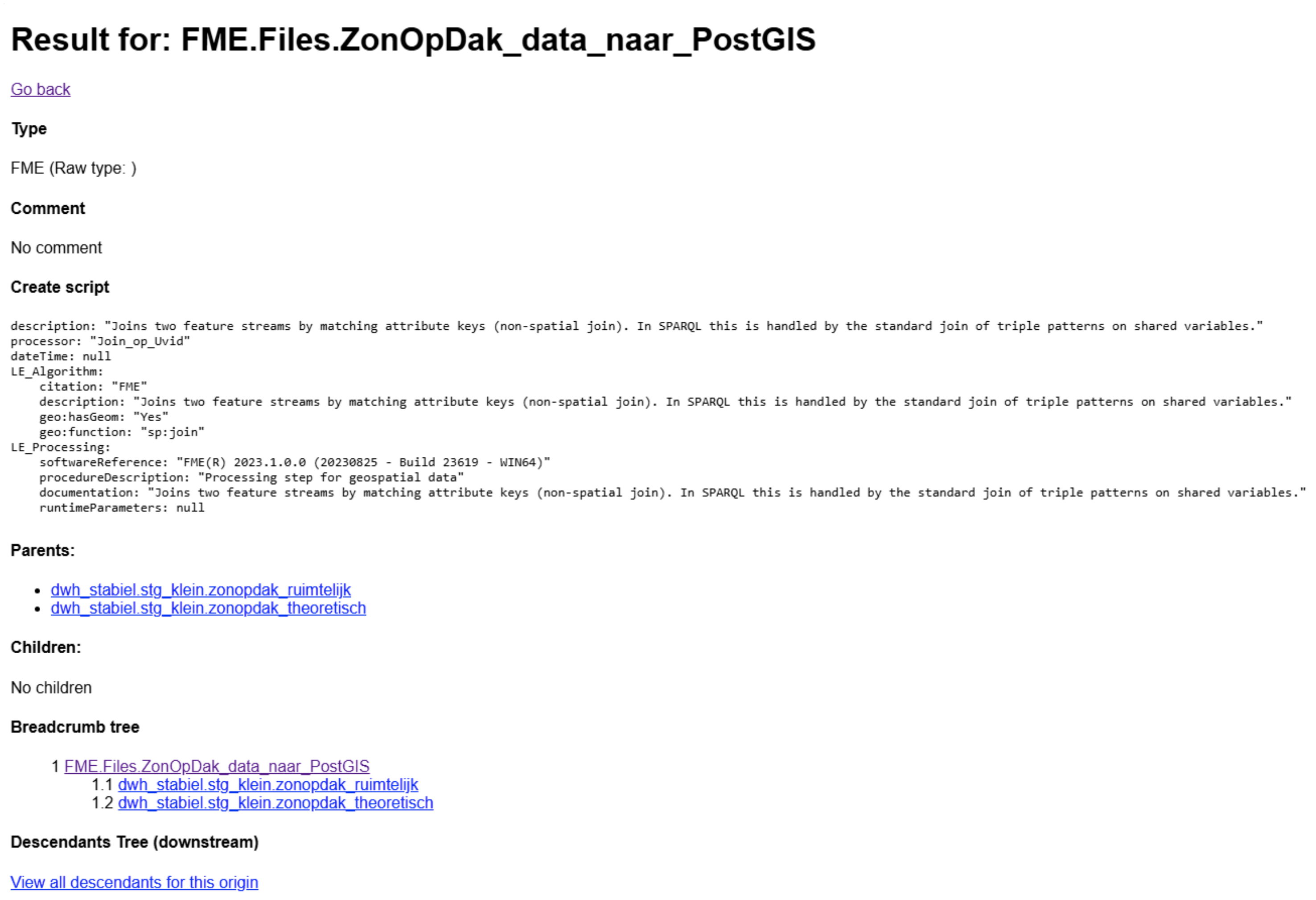

«ISO 19115» CSV_1
+description: FeatureReader
+sourceReferenceSystem: Netherlands-RDNew-2008
+sourceCitation: F:\dataservices\data\BP07\1.csv
+dateTime: 2025-08-05 12:19:13
«ISO 19115» CSV_2
+description: FeatureReader
+sourceReferenceSystem: Netherlands-RDNew-2008
+sourceCitation: F:\dataservices\data\BP07\1.csv
+dateTime: 2025-08-05 12:19:13
«ISO 19115» STG_DATA_mv_bouwwerk
+description: FeatureReader
+sourceReferenceSystem: Netherlands-RDNew-2008
+sourceCitation: F:\dataservices\data\BP07\2.csv
+dateTime: 2025-08-05 12:19:13
source
source
source
«ISO 19115 + GeoSPARQL» LE_Algorithm_1
+citation: FME
+description: Joins two feature streams by matching attribute keys.
+geo:hasGeom: Yes
+geo:function: sp:join
algorithm
«ISO 19115» LI_ProcessStep_1
+description: Joins two feature streams by matching attribute keys...
+processor: Join_op_Uvid
+dateTime: 2025-08-05 12:18:59
+runtime: 0.9
processingInformation
«ISO 19115» LE_Processing_1
+procedureDescription: Processing step for geospatial data
+documentation: Joins two feature streams by matching attribute keys...
+runtimeParameters: None
+softwareReference: FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)
next
«ISO 19115 + GeoSPARQL» LE_Algorithm_2
+citation: FME
+description: Joins two feature streams by matching attribute keys.
+geo:hasGeom: Yes
+geo:function: sp:join
algorithm
«ISO 19115» LI_ProcessStep_2
+description: Joins two feature streams by matching attribute keys...
+processor: Join_op_Uvid_2
+dateTime: 2025-08-05 12:19:00
+runtime: 0.2
processingInformation
«ISO 19115» LE_Processing_2
+procedureDescription: Processing step for geospatial data
+documentation: Joins two feature streams by matching attribute keys...
+runtimeParameters: None
+softwareReference: FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)
output
output
«ISO 19115» zonopdak_ruimtelijk
+description: ZonOpDak_Ruimtelijk
+sourceReferenceSystem: Netherlands-RDNew-2008
+sourceCitation: dwh_stabiel
+sourceExtent: None
«ISO 19115» zonopdak_theoretisch
+description: ZonOpDak_Theoretisch
+sourceReferenceSystem: Netherlands-RDNew-2008
+sourceCitation: dwh_stabiel
+sourceExtent: None

Figure 4.8 – Zon op Dak naar PostGIS UML diagram of parsed lineage



Result for: FME.Files.ZonOpDak_data_naar_PostGIS

Go back

Type

FME (Raw type:)

Comment

No comment

Create script

```
description: "Joins two feature streams by matching attribute keys (non-spatial join). In SPARQL this is handled by the standard join of triple patterns on shared variables."
processor: "Join_op_Uvid"
dateTime: null
LE_Algorithm:
    citation: "FME"
    description: "Joins two feature streams by matching attribute keys (non-spatial join). In SPARQL this is handled by the standard join of triple patterns on shared variables."
    geo:hasGeom: "Yes"
    geo:function: "sp:join"
LE_Processing:
    softwareReference: "FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)"
    procedureDescription: "Processing step for geospatial data"
    documentation: "Joins two feature streams by matching attribute keys (non-spatial join). In SPARQL this is handled by the standard join of triple patterns on shared variables."
    runtimeParameters: null
```

Parents:

- dwh_stabiel.stg_klein.zonopdak_ruimtelijk
- dwh_stabiel.stg_klein.zonopdak_theoretisch

Children:

No children

Breadcrumb tree

1 FME.Files.ZonOpDak_data_naar_PostGIS
    1.1 dwh_stabiel.stg_klein.zonopdak_ruimtelijk
    1.2 dwh_stabiel.stg_klein.zonopdak_theoretisch

Descendants Tree (downstream)

View all descendants for this origin

Figure 4.9 – Zon op Dak visualized in the Flask application

4.2 Quantitative Evaluation

To assess the parser's accuracy and compliance to the standards this section will apply the evaluation criteria to the metrics established in section 2.5 to the use cases. This will allow insights into how well the parser adheres to the structure and if it accurately parsers the required data.

The evaluation criteria are centred around the six core quality measures adapted from the ISO 19157. These include completeness, logical consistency, positional accuracy, temporal accuracy, temporal validity, thematic accuracy and quantitative attribute accuracy. For each of the 3 use cases the outputs were evaluated and can be seen in the Tables 4.1-4.3 respectively.

4.2.1 Verversen Regio Indeling Groenbeheer

Code	ISO data-quality element	Sub-element	Tool-specific measure	Tolerance / target	Results	Comments
A01	Completeness	Commission	False-positive rate in transformer mapping (1 − P) = Count FP transformers ÷ total mapped	≤ 5 % (P ≥ 95 %)	P = 100%	No transformers were incorrectly mapped.
A02		Omission	1 − P = (Actual Steps − Captured Steps) ÷ Actual Steps	≤ 5 % (P ≥ 95 %)	P = 100%	All expected transformers were captured.
A03	Logical consistency	Conceptual consistency	Semantic accuracy of GeoSPARQL class assignment P = TP / (TP + FP) on class labels	≥ 95 %	P = 100%	All mappings are valid
A04		Domain consistency	Validity of CRS & ontology URIs AC = Filled valid-URI fields ÷ expected	≥ 90 %	AC = 99%	One mapping was None, this was the extent, which this dataset did not have
A05		Format consistency	JSON-schema validation pass rate (no errors in JSON output)	100 %	100%	No errors were found in the output format
A06	Positional accuracy	Absolute / external	Presence of sourceReferenceSystem attribute (CRS recorded) AC = CRS filled in LI_Source records ÷ total LI_Source records	≥ 90 %	AC = 100%	Both the source and the output had the correct CRS
A07	Temporal accuracy	Accuracy of time measurement	Millisecond resolution of `dateTime` stamps	All timestamps present, ± 1 s	Passed	All of the time stamps were accurate
A08		Temporal consistency	Chronological order of LI_ProcessStep records	0 inversions allowed	Passed	

A09		Temporal validity	Run time lies within declared job window (compare `dateTime` against job start–end)	Pass / Fail	Passed	It was run in under 2 seconds
A10	Thematic accuracy	Classification correctness	Correct GeoSPARQL mapping P = TP / (TP + FP) on class labels	≥ 95 %	P = 100%	
A11		Non-quantitative attribute correctness	Field names & units correctly copied AC = Filled non-numeric fields ÷ expected	≥ 90 %	AC = 100%	All of the names and times were correctly parsed
A12		Quantitative attribute accuracy	Feature-count between FME log and lineage	Expect ± 0 difference	Passed	The log and the parser matched up well

This workflow hits most of the targets set in thematic accuracy as well as temporal accuracy. However, A04 there was one none extent mapped. This was because the spatial extent of this dataset was not known.

4.2.2 Capaciteitskaart Elektriciteitsnet

Code	ISO data-quality element	Sub-element	Tool-specific measure	Tolerance / target	Results	Comments
A01	Completeness	Commission	False-positive rate in transformer mapping (1 − P) = Count FP transformers ÷ total mapped	≤ 5 % (P ≥ 95 %)	P = 100%	No transformers were incorrectly mapped.
A02		Omission	1 − P = (Actual Steps − Captured Steps) ÷ Actual Steps	≤ 5 % (P ≥ 95 %)	P = 100%	All expected transformers were captured.
A03	Logical consistency	Conceptual consistency	Semantic accuracy of GeoSPARQL class assignment P = TP / (TP + FP) on class labels	≥ 95 %	P = 100%	All mappings are valid
A04		Domain consistency	Validity of CRS & ontology URIs AC = Filled valid-URI fields ÷ expected	≥ 90 %	AC = 90%	2 mapping types were none – the extent, which this dataset does not have as well as the date time stamp for the
A05		Format consistency	JSON-schema validation pass rate (no errors in JSON output)	100 %	100%	No errors were found in the output format
A06	Positional accuracy	Absolute / external	Presence of sourceReferenceSystem attribute (CRS recorded) AC = CRS filled in LI_Source records ÷ total LI_Source records	≥ 90 %	AC = 100%	Both the source and the output had the correct CRS
A07	Temporal accuracy	Accuracy of time measurement	Millisecond resolution of `dateTime` stamps	All timestamps present, ± 1 s	Failed	Timestamp was missing from LE_Processing_4
A08		Temporal consistency	Chronological order of LI_ProcessStep records	0 inversions allowed	Passed	

A09		Temporal validity	Run time lies within declared job window (compare `dateTime` against job start–end)	Pass / Fail	Passed	
A10	Thematic accuracy	Classification correctness	Correct GeoSPARQL mapping P = TP / (TP + FP) on class labels	≥ 95 %	P = 100%	
A11		Non-quantitative attribute correctness	Field names & units correctly copied AC = Filled non-numeric fields ÷ expected	≥ 90 %	AC = 100%	All of the names were correctly parsed
A12		Quantitative attribute accuracy	Feature-count between FME log and lineage	Expect ± 0 difference	Passed	The log and the parser matched up well

The workflow scored on target in most of the completeness and logistical accuracy however there were gaps in A07 on the timestamp presence. As well as A04 as certain fields were missing. The GeoSPARQL ontology was correctly achieved, and the visual interoperability of the lineage was confirmed through stakeholder analysis and review.

4.2.3 Zon op Dak

Code	ISO data-quality element	Sub-element	Tool-specific measure	Tolerance / target	Results	Comments
A01	Completeness	Commission	False-positive rate in transformer mapping (1 − P) = Count FP transformers ÷ total mapped	≤ 5 % (P ≥ 95 %)	P = 100%	
A02		Omission	1 − P = (Actual Steps − Captured Steps) ÷ Actual Steps	≤ 5 % (P ≥ 95 %)	P = 100%	
A03	Logical consistency	Conceptual consistency	Semantic accuracy of GeoSPARQL class assignment P = TP / (TP + FP) on class labels	≥ 95 %	P = 100%	
A04		Domain consistency	Validity of CRS & ontology URIs AC = Filled valid-URI fields ÷ expected	≥ 90 %	AC = 90%	Two types of mappings were none the spatial extent, which is not known as well as the run time parameters for the processing.
A05		Format consistency	JSON-schema validation pass rate (no errors in JSON output)	100 %	100%	No errors were found in the output format
A06	Positional accuracy	Absolute / external	Presence of sourceReferenceSystem attribute (CRS recorded) AC = CRS filled in LI_Source records ÷ total LI_Source records	≥ 90 %	AC = 100%	Both the source and the output had the correct CRS
A07	Temporal accuracy	Accuracy of time measurement	Millisecond resolution of `dateTime` stamps	All timestamps present, ± 1 s	Failed	The run time parameters were missing from both processing steps

A08		Temporal consistency	Chronological order of LI_ProcessStep records	0 inversions allowed	Passed	
A09		Temporal validity	Run time lies within declared job window (compare `dateTime` against job start–end)	Pass / Fail	Passed	It was run in under 10 seconds
A10	Thematic accuracy	Classification correctness	Correct GeoSPARQL mapping P = TP / (TP + FP) on class labels	≥ 95 %	P = 100%	
A11		Non-quantitative attribute correctness	Field names & units correctly copied AC = Filled non-numeric fields ÷ expected	≥ 90 %	AC = 100%	All of the names and times were correctly parsed
A12		Quantitative attribute accuracy	Feature-count between FME log and lineage	Expect ± 0 difference	0	The log and the parser matched up

The workflow scored well overall in Completeness, and thematic accuracy. However, scored lower on temporal accuracy and logical consistency. This was as a result of the run time parameters missing from both process steps. This indicates that there is a parsing error in the model.

4.3 Qualitative Evaluation

In line with the ISO 19157 quality-assessment framework the evaluation is organized around the 4 core dimensions, completeness, logical consistency, accuracy and usability. For each ISO criteria, it was rated **P** (Pass) or **F** (Fail).

4.3.1 Completeness

The completeness of the data is the extent to which the information is accurately represented in the model based on the available data.

Code	Description	VRIG	CE	ZoD
A01	Commission	P	P	P
A02	Omission	P	P	P

The model achieved a perfect 6/6 successful assessments demonstrating full completeness across both test scenarios.

Manual cross checks against the ETL logs confirmed that every relevant process step, and in and output sources were correctly parsed in the model. There were also no omissions in the data, indicating that the data provenance is correctly preserved and analysts can use the lineage.

4.3.2 Consistency

Consistency assesses whether the information recorded in the lineage model obeys internal rules, logical constraints and agreed coding conventions (ISO 19157 logical-consistency subclass).
For this study we focused on three critical facets (A03–A05):

Code	Description	VRIG	CE	ZoD
A03	Conceptual consistency	P	P	P
A04	Domain consistency	P	P	P
A05	Format consistency	P	P	P

Overall, the conceptual model upheld consistency in the parsing. There was a domain discrepancy in the CE workflow as there were 2 mapping types that were none. This was the extent, which is a result of this dataset not having an extent. However also the date and time stamp were missing from some of the transformation steps.

4.3.3 Accuracy

This subsection evaluates the conceptual accuracy of the proposed lineage model. Conceptual accuracy here is defined as the degree to which the model represents the semantics of spatial operations, attribute transformations, metadata structures, and transformation sequences observed in real-world Extract-Transform-Load (ETL) processes. The conceptual accuracy was assessed according to the ISO 19157 data-quality elements A06-A12.

Code	**Description_**	**VRIG**	**CE**	**ZoD**
A06	Logical / schematic consistency	P	P	P
A07	Domain consistency	P	F	F
A08	Format consistency	P	P	P
A09	Topological consistency	P	P	P
A10	Temporal consistency	P	P	P
A11	Thematic consistency	P	P	P
A12	Maintenance consistency	P	P	P

The lineage model achieved 16 / 18 successful assessments (VRIG 7 / 7; CE 6 / 7; ZoD 6/7), demonstrating conceptual accuracy across the tested ETL scenarios.

Tests A06 and A08 – A12 all passed, demonstrating that the framework reliably preserves every key dimension: schema, encoding, topology, temporal lineage, and thematic labels. Direct integration of GeoSPARQL functions—such as `geo:union` and `geof:area`—supports fine-grained spatial queries, while the paired `prov:startedAtTime` and `prov:endedAtTime` fields retain a precise record of execution order. Only test A07 failed in the CE workflow upon manual review the issue was traced to a missing timestamp log on one transformer. While it was only one it does demonstrate that the tool will need to be debugged before going into production.

4.3.4 Usability

Usability (ISO 19157, A13) reflects the degree to which the lineage information is fit for its intended purpose which is supporting governance staff, data engineers and analysts in day-to-day tasks. An interview was conducted with an employee from the RVB who reviewed and analysed the current model. This employee from the RVB then wrote the following review:

"The core of our architecture consists of FME scripts (and other ETL tooling), a PostgreSQL database, a GeoServer, and a viewer. More than 500 map layers are managed and made available to users through this architecture. Many layers in the viewer have different names in GeoServer, combine data from multiple data sources within the database and originate from different ETL tools such as FME. It often took a lot of time to manually track these data flows.

The data lineage tool has not only made it significantly easier to understand how data flows operate but also serves as a communication tool during projects. Additionally, the tool can act as input for other tooling, such as a cleaning tool for system maintenance. Thanks to the data lineage tool, we already begun the cleanup process.

To fully benefit from these advantages, it is essential that the data model tool functions properly. Currently, the tool works for our environment. However, due to changes in architecture and design choices, a certain level of maintenance will be required. Currently, the data model tool only supports the use of the FeatureWriter transformer to write data to our database. Even though we currently always use a FeatureWriter, this could change in the future. This may force us to decide whether to maintain the data lineage tool, limit its functionality or temporarily hold on to certain parts of our architecture to ensure the data lineage tool remains operational. Choosing the last option would limit our flexibility, but we gain insight, system maintenance capabilities and a communication tool." (Staring, 2025)

This indicates that the current version is usable and understandable by the current employee at the RVB. However future changes to the infrastructure and or tooling could cause the data lineage tool to have to be changed or adapted.

5 Discussion

The development and evaluation of the spatial data lineage model reveal several limitations spanning technical, semantic, and practical dimensions. While the model successfully demonstrated feasibility within the RVB use cases, these limitations highlight areas where the approach falls short of a comprehensive, universal solution. This section discusses the key technical constraints of the parser implementation, the semantic gaps, and practical challenges related to generalizability and user adoption. Together, these insights lay the groundwork for the implications and recommendations that follow.

5.1 Technical Limitations

The current static parser, which analyses the FMW files faced challenges in its parsing since the files are non-standardized and inconsistent format. An FME document is not a straightforward XML document, rather it is a text structure that contains XML-like content. This means that the script cannot be parsed without preprocessing of the file, however conventional XML parsers cannot be applied due to the slight variation from traditional XML. Therefore, the specialized parser was created, however as it is specialized this makes it prone to problems if FME changes its formatting. An example can be seen in recent version changes of FME workspaces where a shift occurred from versions and users lost some of their created transformers due to the change logging methods (Zornig, 2015). Highlighting how slight changes in the logic can impact the structure of the FMW file, and hence the parser that was created in this study.

The log parser that extracts the lineage from FME's runtime logs has its own set of limitations. One of the main issues stems from the reliance on log files being complete and available. This means that if a translation job fails or is aborted, the log file may be incomplete, which can lead to incomplete metadata extraction. In such cases, the parser might only capture the portion of the workflow that executed before the failure, reducing lineage completeness for failed runs. Another issue, similar to that of the FMW parser is that log files do not have a standardized structure, instead they are free-form text with human readable messages. This is also stated by FME on their website stating that not everything is shown in sequence and that there is no standardized format for a log file (Safe Software, 2025). Therefore, this is also prone to shifts between versions and updates on FME platform, which limits the parsers' overall reliability.

5.2 Semantic Limitations

A central objective of this work was to enhance traditional lineage records by embedding spatial semantics, achieved through a combined use of the ISO 19115 lineage model and the GeoSPARQL ontology. This hybrid design delivers clear benefits, but it also exposes notable semantic limitations and gaps.

The ISO 19115 standard provides robust structural lineage metadata, recording the occurrence of process steps, the source datasets, parameters, and related information however did not specify which type of geospatial operation occurred. This approach sought to address this through mapping FME transformers to corresponding GeoSPARQL definitions where possible. This aligns with current research highlighting GeoSPARQL as a key enabler for interoperable provenance across GIS workflows (Ivanova et al., 2017; Battle et al., 2012). However semantic coverage remains incomplete as GeoSPARQL was not designed as a comprehensive taxonomy of geoprocessing operations. As a result, certain FME transformers had no direct GeoSPARQL equivalent. Future work could investigate alternatives to GeoSPARQL that can fully map all the transformations done in FME.

5.3 Practical and Evaluative Limitations

The prototype lineage model was developed and evaluated in the context of the RVB's internal FME workflows, which limits the diversity of the scenarios tested. Although it was successful for the scenarios of the RVB, which encompass large and complex workflows on a national scale. It does bring into question the generalizability of the model in a different setting. The model's overall design does offer flexibility and is standards based therefore it suggests that it can be adapted to organizations that are using FME. However, this was not tested in this study therefore the approaches robustness and efficiency in a broader context remains to be tested.

The evaluation of the model was done through using the ISO 19157 derived criteria and feedback from employees from the RVB. This confirmed the technical correctness and that it helped to improve the transparency of the data lineage, however it does mean that the scope is limited to the finite set of workflows from the RVB. A full usability assessment would have been beneficial where efficiency was monitored and measured within the RVB to establish the overall benefits.

6 Conclusion

This study set out to close a clear provenance gap in the RVB spatial-data workflows. A standards-aligned lineage model has been designed, implemented, and validated for FME-based Extract–Transform–Load (ETL) processes. Below, each original goal and research question is revisited in turn, followed by an overall synthesis, the main limitations, and recommendations for future work.

6.1 Answers to Research Questions

6.1.1 What are the technical and organisational requirements and priorities for geospatial data lineage?

The technical requirements can be grouped into three main components: the compliance with standards, the automation of the lineage extraction and the performance of the tool. As established in chapter two the lineage should be formatted according to international standards of the ISO to adhere to governance policies (Closa et al., 2017). As adhering to these standards ensures long-term interoperability. Further it was found in chapter two that it is critical to automate the data lineage tracking process as it increases the efficiency and reliability of data lineage tools (Closa et al., 2019). The last technical requirement that was identified was the need for quality testing criteria, these were identified through the ISO 19157 that defines a set of quality measure for evaluating geospatial data (Urena-Camara et al., 2019).

The organisational requirements can be grouped into two main areas: governance alignment and that it works with the current systems. First, as mentioned previously the lineage needs to be formatted according to the ISO 19115. Second, the lineage solution must work with RVB's existing lineage data tracker for their data warehouse so that provenance information becomes a routine decision-support asset rather than a stand-alone model (see Chapter 2, section 2.3).

6.1.2 What data lineage models currently exist for tracking lineage in both spatial and non-spatial contexts, and what are their strengths and weaknesses in supporting spatial use cases?

A literature review done in chapter 2 revealed that within the current systems there are two keyways to document data lineage the ISO 19115 and the PROV. Some research even suggests merging the two to increase the breadth and depth of the data lineage recorded (Jiang et al., 2018). The strength in this is that both the agents, temporal validity, and the overall lineage is correctly recorded. However, a weakness identified was that there is no direct ontology for geospatial transformations that can work with these.

When researching varying models currently in production one can see an emerging trend, that they are all automated and visually represent the lineage for the user. Good examples that were discussed were AWS and the provenance metadata model focusing on raster data (Closa et al., 2019). However, both types have gaps in their ability to provide a full geospatial provenance description and state that future models should look at exploiting other types of catalogues and ontologies to accurately represent the who, what, where, when and how of data lineage (Closa et al., 2019). Therefore, this paper suggests using GeoSPARQL as an ontology for defining the relevant spatial operations in combination with ISO 19115 standards.

6.1.3 What are the existing challenges and limitations in tracking data lineage within geospatial ETL tools, specifically when using tools such as FME?

In chapter two, two main points were established which are existing challenges in tracking data lineage, specifically in FME, mainly the lack of a built-in lineage export and establishing granularity vs overview. FME logs every reader, writer and transformer at run-time, yet it offers no built-in mechanism to export this information in a standards-compliant form (e.g., ISO 19115). Users on the Safe Software community forum note that there is currently no easy way to obtain the data lineage and export it to another platform. Further, general-purpose data-lineage literature highlights the "granularity issue," namely the difficulty of choosing between coarse, dataset-level snapshots and fine, feature- or attribute-level histories. FME compounds the problem because a single workspace can execute hundreds of transformers in one translation thread. Capturing every intermediate geometry yields graphs so dense that they become unwieldy for analysts. Striking the right balance therefore remains an open design question. Therefore, a model was created that automatically parses the data lineage and only extracts lineage from transformers whose definition is defined in the GeoSPARQL ontology.

6.1.4 How can a geospatial data lineage model be designed and implemented in FME to enable end-to-end traceability of spatial transformations?

Chapter 3 sets out a three-layer architecture that delivers end-to-end traceability for FME-based ETL processes while meeting the technical and organisational requirements identified earlier. Section 3.2 goes into more detail on how the model fuses together the ISO 19115 and the GeoSPARQL for its geospatial definitions. Through this the model is now able to record when changes occurred, which datasets were involved, how the data was transformed, and in which layer it was done. Section 3.3 details a Python service that is triggered automatically when FME Server completes a workspace. The parser reads the static FMW file, and its run-time log maps every reader, writer and transformer to its GeoSPARQL definition counterpart. The lineage extracted is then all placed into a JSON file. This JSON is then used as the basis for visualization and integration into a searchable system which analysts can use.

Overall, through the combination of log files and FMW files into a python parser FME data lineage can now be structured in a transparent, standard compliant way, for analytic use within the RVB.

6.1.5 How can the proposed model be tested to ensure that it meets the identified requirements for geospatial data lineage, and what methods or metrics can be used to evaluate its effectiveness in real-world scenarios?

The ISO 19157 standards were identified in section 2.5 as the requirements for spatial datasets. This method of evaluation was then used to evaluate the model on four main aspects. Completeness, consistency, accuracy and usability. The paragraphs below summarise how each criterion was operationalized, applied and reviewed in chapters 4 and 5.

The completeness was assessed through comparing the parsed lineage to the original files. This was done through checking for false positives or omissions neither of which were found in the examples. The consistency was checked by comparing the original code to the GeoSPARQL mappings to check if the mappings were done correctly. It was found that all the GeoSPARQL mappings were correct, however one CRS mapping failed. The accuracy tests focused on whether the lineage captured accurately represents the FME data. The execution timestamps that were extracted from the log file were compared to the parsed values. No anomalies were found; it was representative of the live runs done.

Lastly usability was evaluated through a review conducted by the RVB employee. Here the outcome presented that currently the model works well for their workflows and that it meets their current needs but future expansions or changes to their workflows could mean the model needs to be adapted. Taken together the results show that the model is currently complete, structurally sound, easy to use and accurate.

6.2 Recommendation for Future Work

The current implementation provides a standards-aligned foundation for capturing spatial data lineage within FME workflows, but several areas for future enhancement have been identified:

Versioned Lineage Storage

While the current model captures lineage for each run of an FME workflow, it does not retain historical versions of workflows or their outputs. Introducing a versioning mechanism—such as git-style lineage snapshots or delta logging—would enable users to track changes over time, compare workflow revisions, and reconstruct past data states. This is particularly relevant in governance contexts where audit trails or rollback capabilities are essential.

Extended GeoSPARQL Mappings and Ontological Coverage

The current GeoSPARQL mapping covers 17 unique FME transformers, but several high-value spatial operations remain unmapped due to lack of equivalent functions in the standard. A future extension could involve:

- Defining custom ontology terms for FME-specific logic.
- Proposing enhancements to GeoSPARQL to incorporate commonly used spatial ETL operations (e.g., spatial predicates within compound filters).
- A different ontology that better matches the FME transformers.

This would improve semantic completeness and enhance the accuracy of provenance graphs.

Support for Complex Workflows and Enhanced UI

Support for multi-input/multi-output workflows can be improved by implementing relationship mapping between individual datasets and their downstream outputs. This could be achieved by tagging inputs/outputs with unique identifiers and visualizing their flow paths. Furthermore, the user interface could be expanded with features such as filter-by-geometry-type, operation search, and comparative lineage views across versions.

These future extensions will enhance the model's completeness, usability, and interoperability, positioning it as a core enabler of traceable, standards-based spatial data governance.

References

Abhayaratna, Joseph & Brink, Linda & Car, Nicholas & Atkinson, Rob & Homburg, Timo & Knibbe, Frans & Wagner, Anna & Bonduel, Mathias & Rasmussen, Mads & Thiery, Florian. (2020). OGC Benefits of Representing Spatial Data Using Semantic and Graph Technologies.

Amazon Web Services. (2024, March 18). Introducing end-to-end data lineage (preview) *visualization in Amazon DataZone*. AWS News Blog. https://aws.amazon.com/blogs/big-data/introducing-end-to-end-data-lineage-preview-visualization-in-amazon-datazone

Apache NiFi Documentation. (2023). Overview of Apache NiFi – provenance features. https://nifi.apache.org/docs/nifi-docs/html/nifi-user-guide.html#provenance

AWS Big Data Blog (Nguyen, K., et al.). (2022). *Build data lineage for data lakes using AWS Glue, Amazon Neptune, and Spline*. https://aws.amazon.com/blogs/big-data/build-data-lineage-for-data-lakes-using-aws-glue-amazon-neptune-and-spline

Battle, R., & Kolas, D. (2012). Enabling the geospatial semantic web with parliament and geosparql. *Semantic Web*, *3*(4), 355-370.

Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S. & Zhao, J. (2012). PROV-O: The PROV Ontology .

Breunig, M., Bradley, P. E., Jahn, M., Kuper, P., Mazroob, N., Rösch, N., ... & Jadidi, M. (2020). Geospatial data management research: Progress and future directions. *ISPRS International Journal of Geo-Information*, *9*(2), 95.

Closa, G., Masó, J., Julià, N., & Pons, X. (2021). Geospatial queries on data collection using a common provenance model. ISPRS International Journal of Geo-Information, 10(3), 139.

Closa, G., Masó, J., Pross, B., & Pons, X. (2017). W3C PROV to describe provenance at the dataset, feature, and attribute levels in a distributed environment. *Computers, Environment and Urban Systems, 64*, 103–117. https://doi.org/10.1016/j.compenvurbsys.2017.03.001

Closa, G., Masó, J., Zabala, A., Pesquer, L., & Pons, X. (2019). A provenance metadata model integrating ISO geospatial lineage and the OGC WPS: Conceptual model and implementation. *Transactions in GIS, 23*(6), 1241–1269. https://doi.org/10.1111/tgis.12555

Closa, J., Masó, J., & Pons, X. (2017). Describing geospatial provenance using ISO and W3C standards. *International Journal of Digital Earth, 10*(12), 1212–1227. https://doi.org/10.1080/17538947.2016.1239773

Closa, J., Masó, J., & Pons, X. (2019). Automated lineage capture in geospatial processes using WPS and ISO standards. In *Proceedings of the 2019 AGILE Conference on Geographic Information Science*. https://doi.org/10.1007/978-3-030-14745-7_22

Dai, C. F., Zhang, R., Li, P., Wang, W. Q., Cao, Z. W., & Acm. (2017, Dec 20-22). A Minimal Attribute Set-oriented Data Provenance Method. [International conference on big data and internet of things (bdiot 2017)]. International Conference on Big Data and Internet of Things (BDIOT), London, ENGLAND.

Di, L., Shao, Y., & Kang, L. (2013). Implementation of geospatial data provenance in a web service workflow environment with ISO 19115 and ISO 19115-2 lineage model. *IEEE transactions on geoscience and remote sensing*, *51*(11), 5082-5089.

Drobnjak, S., Sekulović, D., Amović, M., Gigović, L., & Regodić, M. (2016). Central geospatial database analysis of the quality of road infrastructure data. Geodetski vestnik, 60(2), 270-284.

ESRI. (n.d.). Geoprocessing history—ArcGIS Pro | Documentation. Retrieved from https://pro.arcgis.com/en/pro-app/latest/help/analysis/geoprocessing/basics/geoprocessing-history.html

Fallahi, Gholam Reza, et al. "An ontological structure for semantic interoperability of GIS and environmental modeling." International Journal of Applied Earth Observation and Geoinformation 10.3 (2008): 342-357.

Gottenbos, P (2025). Internship report. Data Lineage in the Rijksvastgoed Bedrijfs DWH

Joos, G. (2006). Data quality standards. In XXIII FIG Congress (pp. 1-10).

Open Geospatial Consortium [OGC]. (2024). *OGC GeoSPARQL – A Geographic Query Language for RDF Data* (OGC Standard No. 22-047, Version 1.1). Wayland, MA:

OGC. Retrieved July 3, 2025, from https://www.ogc.org/standards/geosparql/

He, L., Yue, P., Di, L., Zhang, M., & Hu, L. (2014). Adding geospatial data provenance into SDI—a service-oriented approach. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *8*(2), 926 936.https://doi.org/10.1109/JSTARS.2015.2400414

Henzen, C., Mäs, S., & Bernard, L. (2013). Provenance information in geodata infrastructures: Metadata visualization with MetaViz. In *Provenance and Annotation of Data and Processes, Lecture Notes in Computer Science (Vol. 7902)* (pp. 87–99). Springer. https://doi.org/10.1007/978-3-642-38729-0_9

International Organization for Standardization. (2013). ISO 19157:2013 Geographic information -- Data quality.

International Organization for Standardization. (2014). ISO 19115-1: Geographic information — Metadata — Part 1: Fundamentals (Technical Report ISO 19115-1:2014).

International Organization for Standardization. (2019). Geographic information - Metadata – Part 2: Extensions for acquisition and processing (ISO 19115-2:2019).

IGN France International. (2014, June 1). *Survey quality control – Implementation of ISO 19157:2013: Development of Abu Dhabi Land Survey Act* (Version 1.03, Draft). Department of Municipal Affairs, Emirate of Abu Dhabi.

Interlandi, M., Ekmekji, A., Shah, K., Gulzar, M. A., Tetali, S. D., Kim, M., Millstein, T., & Condie, T. (2018). Adding data provenance support to Apache Spark. *Vldb Journal*, *27*(5), 595-615. https://doi.org/10.1007/s00778-017-0474-5

International Organization for Standardization. (2016). *Geographic information — Metadata — Part 3: XML schema implementation for fundamental concepts* (ISO/TS 19115-3:2016). https://www.iso.org/standard/32579.html

Ivánová, I., Armstrong, K., & McMeekin, D. (2017, December). Provenance in the next-generation spatial knowledge infrastructure. In Proceedings of the 22nd International Congress on Modelling and simulation (MODSIM 2017), Hobart, Tasmania, Australia (pp. 3-8).

Jamedzija, M., & Duric, Z. (2021). Moonlight: A Push-based API for Tracking Data Lineage in Modern ETL processes. 2021 20th International Symposium INFOTEH-JAHORINA, INFOTEH 2021 – Proceedings.

Jiang, L., Yue, P., Kuhn, W., Zhang, C., & Guo, X. (2018). Advancing interoperability of geospatial data provenance on the web: Gap analysis and strategies. *Computers & Geosciences, 117*, 21–31. https://doi.org/10.1016/j.cageo.2018.04.003

Staring, K. (2025). Employee Review of the Data lineage Tool.

Kimball, R., & Ross, M. (2013). The data warehouse toolkit (3rd ed.). Wiley.

Kumaran, R. (2021). ETL Techniques for Structured and Unstructured Data. *International* Research Journal of Engineering and Technology (IRJET), 8, 1727-1735.

Lemmens, R., Wytzisk, A., de By, R., Granell, C., Gould, M., & Van Oosterom, P. (2006).

Integrating semantic and syntactic descriptions to chain geographic services. IEEE Internet Computing, 10(5), 42-52.

Malaverri, J., Medeiros, C. B., & Lamparelli, R. A. C. (2012). A provenance approach to assess the quality of geospatial data. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on GeoStreaming (IWGS '12)* (pp. 53–56). https://doi.org/10.1145/2245276.2232116

Masó, J., & Closa, J. (2021). Geospatial queries on data collection using a common provenance model. In *Proceedings of the 2021 AGILE Conference on Geographic Information Science*. https://agile-online.org/index.php/conference/proceedings/proceedings-2021

Safe Software. (2025, March 8). Make FME data lineage information discoverable by other data catalog systems [Idea post]. *Safe Software Community*. https://community.safe.com/ideas/make-fme-data-lineage-information-discoverable-by-other-data-catalog-systems-37486

Safe Software. (n.d.). *Interpreting the log*. In *FME Form Documentation* (2025.1). https://docs.safe.com/fme/html/FME-Form-Documentation/FME-Form/Workbench/Interpreting_the_Log.htm#:~:text=Sequence

Safe Software Community. (2022). Forum thread: Populating a data lineage system with information from FME workspaces. *Safe Software Community*. https://community.safe.com/s/question/0D5S000002M7U6OSAV

Schoenenwald, A., Kern, S., Viehhauser, J., & Schildgen, J. (2021). Collecting and visualizing data lineage of Spark jobs. *Datenbank-Spektrum*, *21*(3), 179-189. https://doi.org/10.1007/s13222-021-00387-7

Tang, M., Shao, S., Yang, W., Liang, Y., Yu, Y., Saha, B., & Hyun, D. (2019). SAC: A system for big data lineage tracking. Proceedings - International Conference on Data Engineering.

Ureña-Cámara, M. A., Nogueras-Iso, J., Lacasta, J., & Ariza-López, F. J. (2019). A method for checking the quality of geographic metadata based on ISO 19157. International Journal of Geographical Information Science, 33(1), 1-27.

Zhang, L., Jing, N., Zhao, J., Zhang, M., & Yang, B. (2020). A provenance model for spatial data processing based on W3C PROV and FME. *ISPRS International Journal of Geo-Information, 9*(6), 372. https://doi.org/10.3390/ijgi9060372

Zhang, M., Jing, Z., Yue, P., Zhao, X., & Zhang, T. (2020). Coupling OGC WPS and W3C PROV for provenance-aware geoprocessing workflows. *Computers & Geosciences, 138*, 104419. https://doi.org/10.1016/j.cageo.2020.104419

Zornig, J. (2015, January 18). FME Desktop saves Transformers uncommented in Workspace *File*. Geographic Information Systems Stack Exchange. https://gis.stackexchange.com/questions/130689/fme-desktop-saves-transformers-uncommented-in-workspace-file

Appendix A: GeoSPARQL mapping functions list

FME Transformer	GeoSPARQL Function
AreaCalculator	geof:area
BoundsExtractor	geof:envelope
BoundingBoxReplacer	geof:envelope
Bufferer	geof:buffer
Clipper	geof:difference
Intersector	geof:intersection
Dissolver	geof:union
Unioner	geof:union
LengthCalculator	geof:length
SpatialFilter	geo:sfContains
GeometryPartExtractor	geof:geometryN
GeometryPropertyExtractor	geof:geometryType
Affiner	geof:transform
CoordinateSystemExtractor	geof:getSRID
CentroidReplacer	geof:centroid
ConvexHullReplacer	geof:convexHull
DistanceCalculator	geof:distance

Appendix B: Example of JSON for Capaciteitskaart Elektriciteitsnet

```json
{
    "Capaciteitskaart_Elektriciteitsnet.FMW": {
        "LE_Source": [
            {
                "name": "CSV",
                "sourceReferenceSystem": "Netherlands-RDNew-2008",
                "description": "FeatureReader",
                "sourceCitation": "f:\\dataservices\\data\\BP23\\congestie_pc6.csv",
                "dateTime": "2025-06-02 13:20:47"
            },
            {
                "name": "CSV",
                "sourceReferenceSystem": "Netherlands-RDNew-2008",
                "description": "FeatureReader",
                "sourceCitation": "f:\\dataservices\\data\\BP23\\voedingsgebieden.csv",
                "dateTime": "2025-06-02 13:20:47"
            }
        ],
        "LI_ProcessStep": [
            {
                "description": "Dissolves (unions) overlapping or contiguous geometries into
one.",
                "processor": "Dissolver_voedingsgebied",
                "dateTime": "2025-06-02 13:18:19",
                "LE_Algorithm": {
                    "citation": "FME",
                    "description": "Dissolves (unions) overlapping or contiguous geometries into
one.",
                    "geo:hasGeom": "Yes",
                    "geo:function": "geo:union"
                },
                "LE_Processing": {
                    "softwareReference": "FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)",
                    "procedureDescription": "Processing step for geospatial data",
                    "documentation": "Dissolves (unions) overlapping or contiguous geometries into
one.",
                    "runtimeParameters": {
                        "runtime": 52.0,
                        "startTime": "2025-06-02 13:18:19",
                        "endTime": "2025-06-02 13:19:11"
                    }
                },
                "runtime": 52.0
            },
            {
                "description": "Dissolves (unions) overlapping or contiguous geometries into
one.",
                "processor": "Dissolver_invoeding",
                "dateTime": "2025-06-02 13:18:19",
                "LE_Algorithm": {
                    "citation": "FME",
                    "description": "Dissolves (unions) overlapping or contiguous geometries into
one.",
                    "geo:hasGeom": "Yes",
                    "geo:function": "geo:union"
                },
                "LE_Processing": {
                    "softwareReference": "FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)",
                    "procedureDescription": "Processing step for geospatial data",
                    "documentation": "Dissolves (unions) overlapping or contiguous geometries into
one.",
                    "runtimeParameters": {
                        "runtime": 147.0,
                        "startTime": "2025-06-02 13:18:19",
                        "endTime": "2025-06-02 13:20:46"
                }
            },
            "runtime": 147.0
        },
        {
            "description":  "Dissolves  (unions)  overlapping  or  contiguous  geometries  into 
one.",
            "processor": "Dissolver_afname",
            "dateTime": "2025-06-02 13:18:19",
            "LE_Algorithm": {
                "citation": "FME",
                "description": "Dissolves (unions) overlapping or contiguous geometries into 
one.",
                "geo:hasGeom": "Yes",
                "geo:function": "geo:union"
            },
            "LE_Processing": {
                "softwareReference": "FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)",
                "procedureDescription": "Processing step for geospatial data",
                "documentation": "Dissolves (unions) overlapping or contiguous geometries into 
one.",
                "runtimeParameters": {
                    "runtime": 101.0,
                    "startTime": "2025-06-02 13:18:19",
                    "endTime": "2025-06-02 13:20:00"
                }
            },
            "runtime": 101.0
        },
        {
            "description": "Aggregates geometries.",
            "processor": "Aggregator_afname",
            "dateTime": "2025-06-02 13:20:01",
            "LE_Algorithm": {
                "citation": "FME",
                "description": "Aggregates geometries.",
                "geo:hasGeom": "Yes",
                "geo:function": "geo:aggCentroid"
            },
            "LE_Processing": {
                "softwareReference": "FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)",
                "procedureDescription": "Processing step for geospatial data",
                "documentation": "Aggregates geometries.",
                "runtimeParameters": {
                    "runtime": 0.0,
                    "startTime": "2025-06-02 13:20:01",
                    "endTime": "2025-06-02 13:20:01"
                }
            },
            "runtime": 0.0
        },
        {
            "description": "Aggregates geometries.",
            "processor": "Aggregator_invoeding",
            "dateTime": "2025-06-02 13:20:47",
            "LE_Algorithm": {
                "citation": "FME",
                "description": "Aggregates geometries.",
                "geo:hasGeom": "Yes",
                "geo:function": "geo:aggCentroid"
            },
            "LE_Processing": {
                "softwareReference": "FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)",
                "procedureDescription": "Processing step for geospatial data",
                "documentation": "Aggregates geometries.",
                "runtimeParameters": {
                    "runtime": 0.0,
                    "startTime": "2025-06-02 13:20:47",
                    "endTime": "2025-06-02 13:20:47"
                }
            },
            "runtime": 0.0
        },
        {
            "description": "Aggregates geometries.",
            "processor": "Aggregator_2",
            "dateTime": "2025-06-02 13:19:12",
            "LE_Algorithm": {
                "citation": "FME",
                "description": "Aggregates geometries.",
                "geo:hasGeom": "Yes",
                "geo:function": "geo:aggCentroid"
            },
            "LE_Processing": {
                "softwareReference": "FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)",
                "procedureDescription": "Processing step for geospatial data",
                "documentation": "Aggregates geometries.",
                "runtimeParameters": {
                    "runtime": 0.0,
                    "startTime": "2025-06-02 13:19:12",
                    "endTime": "2025-06-02 13:19:12"
                }
            },
            "runtime": 0.0
        }
    ],
    "LI_Source": [
        {
            "name": "capaciteitskaart_elektriciteitsnet_voedingsgebied",
            "description": "Writer_capaciteitskaart_elektriciteitsnet_voedingsgebied_2",
            "sourceReferenceSystem": "Netherlands-RDNew-2008",
            "sourceCitation": "dwh_stabiel",
            "sourceExtent": null
        },
        {
            "name": "capaciteitskaart_elektriciteitsnet_afname",
            "description": "Writer_capaciteitskaart_elektriciteitsnet_afname",
            "sourceReferenceSystem": "Netherlands-RDNew-2008",
            "sourceCitation": "dwh_stabiel",
            "sourceExtent": null
        },
        {
            "name": "capaciteitskaart_elektriciteitsnet_invoeding",
            "description": "Writer_capaciteitskaart_elektriciteitsnet_invoeding",
            "sourceReferenceSystem": "Netherlands-RDNew-2008",
            "sourceCitation": "dwh_stabiel",
            "sourceExtent": null
        }
    ]
  }
}
```

Appendix C: Example of JSON for Groenbeheer

```
{
    "Verversen_attributen_Regioindeling_Groenbeheer.FMW": {
        "LE_Source": [
            {
                "name": "Beheerregio",
                "sourceReferenceSystem": "Netherlands-RDNew-2008",
                "description": "FeatureReader",
                "sourceCitation": "$(DataDir)<solidus>*.mdb",
                "dateTime": "2025-05-28 15:04:13"
            }
        ],
        "LI_ProcessStep": [
            {
                "description": "Aggregates geometries.",
                "processor": "Aggregator_2",
                "dateTime": "2025-05-28 15:04:13",
                "LE_Algorithm": {
                    "citation": "FME",
                    "description": "Aggregates geometries.",
                    "geo:hasGeom": "Yes",
                    "geo:function": "geo:aggCentroid"
                },
                "LE_Processing": {
                    "softwareReference": "FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)",
                    "procedureDescription": "Processing step for geospatial data",
                    "documentation": "Aggregates geometries.",
                    "runtimeParameters": {
                        "runtime": 0.0,
                        "startTime": "2025-05-28 15:04:13",
                        "endTime": "2025-05-28 15:04:13"
                    }
                },
            },
            {
                "description": "Extracts the geometry from a file.",
                "processor": "GeometryExtractor",
                "dateTime": "2025-05-28 15:04:13",
                "LE_Algorithm": {
                    "citation": "FME",
                    "description": "Extracts the geometry from a file.",
                    "geo:hasGeom": "Yes",
                    "geo:function": "geo:asGeoJSON"
                },
                "LE_Processing": {
                    "softwareReference": "FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)",
                    "procedureDescription": "Processing step for geospatial data",
                    "documentation": "Extracts the geometry from a file.",
                    "runtimeParameters": {
                        "runtime": 0.0,
                        "startTime": "2025-05-28 15:04:13",
                        "endTime": "2025-05-28 15:04:13"
                    }
                },
                "runtime": 0.0
            },
            {
                "description": "Extracts the geometry from a file.",
                "processor": "GeometryExtractor_2",
                "dateTime": "2025-05-28 15:04:13",
                "LE_Algorithm": {
                    "citation": "FME",
                    "description": "Extracts the geometry from a file.",
                    "geo:hasGeom": "Yes",
                    "geo:function": "geo:asGeoJSON"
                },
                "LE_Processing": {
                    "softwareReference": "FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)",
                    "procedureDescription": "Processing step for geospatial data",
                    "documentation": "Extracts the geometry from a file.",
                    "runtimeParameters": {
                        "runtime": 0.0,
                        "startTime": "2025-05-28 15:04:13",
                        "endTime": "2025-05-28 15:04:13"
                    }
                },
                "runtime": 0.0
            }
        ],
        "LI_Source": [
            {
                "name": "masterdata.regioindeling_groenbeheer",
                "description": "Write_Beheerregios_groenbeheer_to_dwh_gis",
                "sourceReferenceSystem": "Netherlands-RDNew-2008",
                "sourceCitation": "gis",
                "sourceExtent": null
            }
        ]
    }
```

```
}
```

Appendix D: Example of JSON for Zon op Dak

```
{
    "ZonOpDak_data_naar_PostGIS.FMW": {
        "LE_Source": [
            {
                "name": "CSV",
                "sourceReferenceSystem": "Netherlands-RDNew-2008",
                "description": "FeatureReader",
                "sourceCitation": "F:\\dataservices\\data\\BP07\\1.csv",
                "dateTime": "2025-08-05 12:19:13"
            },
            {
                "name": "CSV",
                "sourceReferenceSystem": "Netherlands-RDNew-2008",
                "description": "FeatureReader",
                "sourceCitation": "F:\\dataservices\\data\\BP07\\1.csv",
                "dateTime": "2025-08-05 12:19:13"
            },
            {
                "name": "stg_data.mv_bouwwerk",
                "sourceReferenceSystem": "Netherlands-RDNew-2008",
                "description": "FeatureReader",
                "sourceCitation": "F:\\dataservices\\data\\BP07\\2.csv",
                "dateTime": "2025-08-05 12:19:13"
            }
        ],
        "LI_ProcessStep": [
            {
                "description": "Joins two feature streams by matching attribute keys (non-spatial
join). In SPARQL this is handled by the standard join of triple patterns on shared variables.",
                "processor": "Join_op_Uvid",
                "dateTime": null,
                "LE_Algorithm": {
                    "citation": "FME",
                    "description": "Joins two feature streams by matching attribute keys (non-
spatial join). In SPARQL this is handled by the standard join of triple patterns on shared
variables.",
                    "geo:hasGeom": "Yes",
                    "geo:function": "sp:join"
                },
                "LE_Processing": {
                    "softwareReference": "FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)",
                    "procedureDescription": "Processing step for geospatial data",
                    "documentation": "Joins two feature streams by matching attribute keys (non-
spatial join). In SPARQL this is handled by the standard join of triple patterns on shared
variables.",
                    "runtimeParameters": null
                }
            },
            {
                "description": "Joins two feature streams by matching attribute keys (non-spatial
join). In SPARQL this is handled by the standard join of triple patterns on shared variables.",
                "processor": "Join_op_Uvid_2",
                "dateTime": null,
                "LE_Algorithm": {
                    "citation": "FME",
                    "description": "Joins two feature streams by matching attribute keys (non-
spatial join). In SPARQL this is handled by the standard join of triple patterns on shared
variables.",
                    "geo:hasGeom": "Yes",
                    "geo:function": "sp:join"
                },
                "LE_Processing": {
                    "softwareReference": "FME(R) 2023.1.0.0 (20230825 - Build 23619 - WIN64)",
                    "procedureDescription": "Processing step for geospatial data",
                    "documentation": "Joins two feature streams by matching attribute keys (non-spatial join). In SPARQL this is handled by the standard join of triple patterns on shared variables.",
                    "runtimeParameters": null
                }
            }
        ],
        "LI_Source": [
            {
                "name": "zonopdak_ruimtelijk",
                "description": "ZonOpDak_Ruimtelijk",
                "sourceReferenceSystem": "Netherlands-RDNew-2008",
                "sourceCitation": "dwh_stabiel",
                "sourceExtent": null
            },
            {
                "name": "zonopdak_theoretisch",
                "description": "ZonOpDak_Theoretisch",
                "sourceReferenceSystem": "Netherlands-RDNew-2008",
                "sourceCitation": "dwh_stabiel",
                "sourceExtent": null
            }
        ]
    }
}
```

Appendix E: Code

```
import os
import xml.etree.ElementTree as ET
import json
from io import StringIO
import re

# 1. Geospatial-relation mapping
GEO_REL_MAP = {
    "AreaCalculator": {
        "geoRelation": "geof:area",
        "description": "Computes the area of polygonal geometries (units follow CRS)."
    },
    "BoundsExtractor": {
        "geoRelation": "geof:envelope",
        "description": "Returns the axis-aligned minimum bounding rectangle."
    },
    "BoundingBoxReplacer": {
        "geoRelation": "geof:envelope",
        "description": "Replaces the geometry with its envelope."
    },
    "Bufferer": {
        "geoRelation": "geof:buffer",
        "description": "Buffers geometry by a distance expressed in CRS units."
    },
    "Clipper": {
        "geoRelation": "geof:difference",
        "description": "Subtracts the clipping geometry from the source geometry."
    },
    "Intersector": {
        "geoRelation": "geof:intersection",
        "description": "Computes the geometric intersection of inputs."
    },
    "Dissolver": {
        "geoRelation": "geof:union",
        "description": "Unions overlapping or contiguous geometries into one."
    },
    "Unioner": {
        "geoRelation": "geof:union",
        "description": "Combines multiple geometries into their geometric union."
    },
    "LengthCalculator": {
        "geoRelation": "geof:length",
        "description": "Computes length of linear geometries (CRS units)."
    },
    "SpatialFilter": {
        "geoRelation": "geo:sfContains",
        "description": "Topological predicate: geometry A contains geometry B."
    },
    "GeometryPartExtractor": {
        "geoRelation": "geof:geometryN",
        "description": "Returns the N-th component geometry from a collection."
    },
    "GeometryPropertyExtractor": {
        "geoRelation": "geof:geometryType",
        "description": "Returns the geometry’s type (e.g., Point, Polygon)."
    },
    "Affiner": {
        "geoRelation": "geof:transform",
        "description": "Transforms geometry coordinates using an affine transform."
    },
    "CoordinateSystemExtractor": {
        "geoRelation": "geof:getSRID",
        "description": "Retrieves the Spatial Reference ID of the geometry."
    },
    "CentroidReplacer": {
        "geoRelation": "geof:centroid",
        "description": "Replaces geometry with its centroid."
    },
    "ConvexHullReplacer": {
        "geoRelation": "geof:convexHull",
        "description": "Replaces geometry with its convex hull."
    },
    "DistanceCalculator": {
        "geoRelation": "geof:distance",
        "description": "Returns distance between geometries (CRS units)."
    }
}


# Function to parse log files
def parse_log_file(log_file_path):
    log_data = {
        "dateTime": None,
        "sourceReferenceSystem": None,
        "sourceCitation": []
    }
    with open(log_file_path, "r", encoding="utf-8") as f:
        lines = f.readlines()
        for line in lines:
            # Extract dateTime
            if re.match(r"^\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}", line):
                log_data["dateTime"] = line.split("|")[0].strip()

            # Extract sourceReferenceSystem
            if "|INFORM|" in line and "Coordinate System" in line:
                match = re.search(r"\|INFORM\|Coordinate System `([^`]+)'", line)
                if match:
                    log_data["sourceReferenceSystem"] = match.group(1)

            # Extract sourceCitation (dataset paths)
            if "Opening dataset" in line:
                match = re.search(r"Opening dataset '([^']+)'", line)
                if match:
                    log_data["sourceCitation"].append(match.group(1))

    return log_data

def parse_FMW_xml(path):
    # STEP A: Read & strip the FME "#!" comment prefixes
    with open(path, 'r', encoding='utf-8') as f:
        raw = f.readlines()
    cleaned = []
    for line in raw:
        # Remove "#!" prefixes and any lines starting with "#"
        if line.startswith('#! '):
            cleaned.append(line[3:])
        elif line.startswith('#!'):
            cleaned.append(line[2:])
        elif line.startswith('#'):
            continue  # Skip lines starting with "#"
        else:
            cleaned.append(line)

    # STEP B: Truncate after the closing </WORKSPACE>
    last_ws  =  max((i  for  i,  l  in  enumerate(cleaned)  if  l.strip().startswith('</WORKSPACE>')),
default=None)
    if last_ws is None:
        raise ValueError("Closing </WORKSPACE> tag not found in the XML file.")
    xml_content = ''.join(cleaned[: last_ws + 1])

    # STEP C: Validate and clean XML content
    # Ensure the XML starts with a valid declaration
    if not xml_content.strip().startswith('<?xml'):
        xml_content = '<?xml version="1.0" encoding="UTF-8"?>\n' + xml_content

    # STEP D: Parse XML
    try:
        root = ET.fromstring(xml_content)
    except ET.ParseError as e:
        raise ValueError(f"Failed to parse XML: {e}")

    workspace_name = os.path.basename(path).replace('.xml', '.FMW')
    result = {workspace_name: {}}

    # Extract LAST_SAVE_BUILD parameter for softwareReference
    software_reference = None
    for line in cleaned:
        if "LAST_SAVE_BUILD" in line:
            software_reference = line.split('=')[1].strip().strip('"')
            break

    # LE_Source: Extract input sources from <TRANSFORMER> sections
    readers = []
    transformers = root.find('TRANSFORMERS')
    if transformers is not None:
        for transformer in transformers.findall('TRANSFORMER'):
            if transformer.get('TYPE') == 'FeatureReader':
                for output_feat in transformer.findall('OUTPUT_FEAT'):
                    name = output_feat.get('NAME', '')
                    # Filter out irrelevant terms
                    if name and name not in ['<SCHEMA>', '<OTHER>', 'INITIATOR', '<REJECTED>']:
                        readers.append({
                            "name": name,
                            "sourceReferenceSystem": None,
                            "description": transformer.get('TYPE', ''),
                            "sourceCitation":
transformer.find('./XFORM_PARM[@PARM_NAME="DATASET"]').get('PARM_VALUE', ''),
                        })
    result[workspace_name]["LE_Source"] = readers

    # LI_ProcessStep: Add LE_Processing as a nested heading
    lineage = []
    if transformers is not None:
        for t in transformers.findall('TRANSFORMER'):
            ttype = t.get('TYPE', '')
            parms   =   {xp.get('PARM_NAME',   ''):   xp.get('PARM_VALUE',   '')   for   xp   in
t.findall('XFORM_PARM')}
            geo_relation_data = GEO_REL_MAP.get(ttype, {})
            geo_relation = geo_relation_data.get("geoRelation")
            description = geo_relation_data.get("description", "No description available.")
            if geo_relation:  # Only include transformers with a geoRelation
                lineage.append({
                    "description": description,
                    "processor": parms.get('XFORMER_NAME', ''),
                    "dateTime": None,
                    "LE_Algorithm": {
                        "citation": None,
                        "description": None,
                        "geo:hasGeom": "Yes",
                        "geo:function": geo_relation,
                    },
                    "LE_Processing": {
                        "identifier": t.get('ID', ''),
                        "softwareReference": software_reference,
                        "procedureDescription": "Processing step for geospatial data",
                        "documentation": description,
                        "runtimeParameters": None
                    }
                })
    result[workspace_name]["LI_ProcessStep"] = lineage
    # LI_Source: FeatureWriter transformers as sources
    sources = []
    if transformers is not None:
        for t in transformers.findall('TRANSFORMER'):
            if t.get('TYPE') == 'FeatureWriter':
                parms = {xp.get('PARM_NAME', ''): xp.get('PARM_VALUE', '') for xp in
t.findall('XFORM_PARM')}
                dataset_param = t.find('./XFORM_PARM[@PARM_NAME="WRITER_FEATURE_TYPE_PARAMS"]')
                name = dataset_param.get('PARM_VALUE', '') if dataset_param is not None else None
                # Truncate the name to only include the portion before the first colon
                if name and ':' in name:
                    name = name.split(':')[0]
                sources.append({
                    "name": name,
                    "description": parms.get('XFORMER_NAME', ''),
                    "sourceReferenceSystem": None,
                    "sourceCitation": parms.get('DATASET', ''),
                    "sourceExtent": None,
                })
    result[workspace_name]["LI_Source"] = sources

    return result
# Combine log data with parsed XML data
def combine_data(xml_data, log_data):
    for workspace, details in xml_data.items():
        # Add dateTime and sourceReferenceSystem to LE_Source
        for i, source in enumerate(details.get("LE_Source", [])):
            source["dateTime"] = log_data.get("dateTime")
            source["sourceReferenceSystem"] = log_data.get("sourceReferenceSystem")
            # Assign sourceCitation from log_data if available
            if i < len(log_data.get("sourceCitation", [])):
                source["sourceCitation"] = log_data["sourceCitation"][i]

    return xml_data
def main():
    # Hard-coded path to your XML workspace
    workspace_xml = r"H:\Mijn
documenten\Repo\postgresql\geostep_scripts\Capaciteitskaart_Elektriciteitsnet\Capaciteitskaart_E
lektriciteitsnet.FMW"
    log_file_path = r"H:\Mijn documenten\Repo\postgresql\datamodel_scripts\Data_model_tool\log
files\Capaciteitskaart_Elektriciteitsnet.log"
    output_file = r"H:\Mijn
documenten\Repo\postgresql\datamodel_scripts\Data_model_tool\FMW_parser\FMW_output.json"

    # Parse XML and log file
    parsed = parse_FMW_xml(workspace_xml)
    log_data = parse_log_file(log_file_path)

    # Combine data
    combined_data = combine_data(parsed, log_data)

    # Save to JSON file
    with open(output_file, 'w', encoding='utf-8') as fout:
        json.dump(combined_data, fout, indent=4, ensure_ascii=False)

    print(f"Output saved to {output_file}")

if __name__ == "__main__":
    main()
```