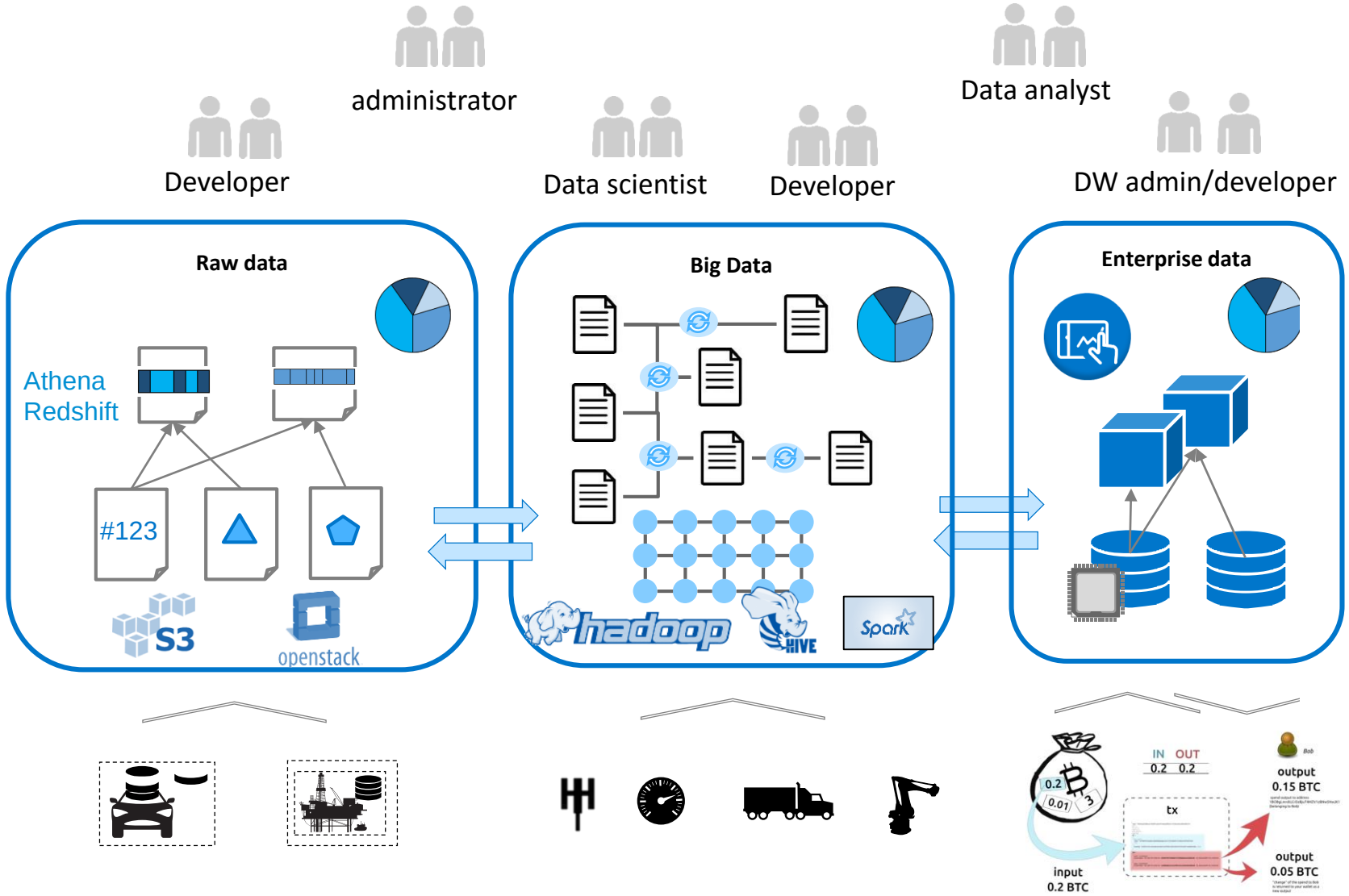
Agile Data Management Challenges in Enterprise Big Data Landscape

Eric Simon, SAP Big Data

October, 2017

SAP Run Simple

Evolution Towards Enterprise Big Data Landscape



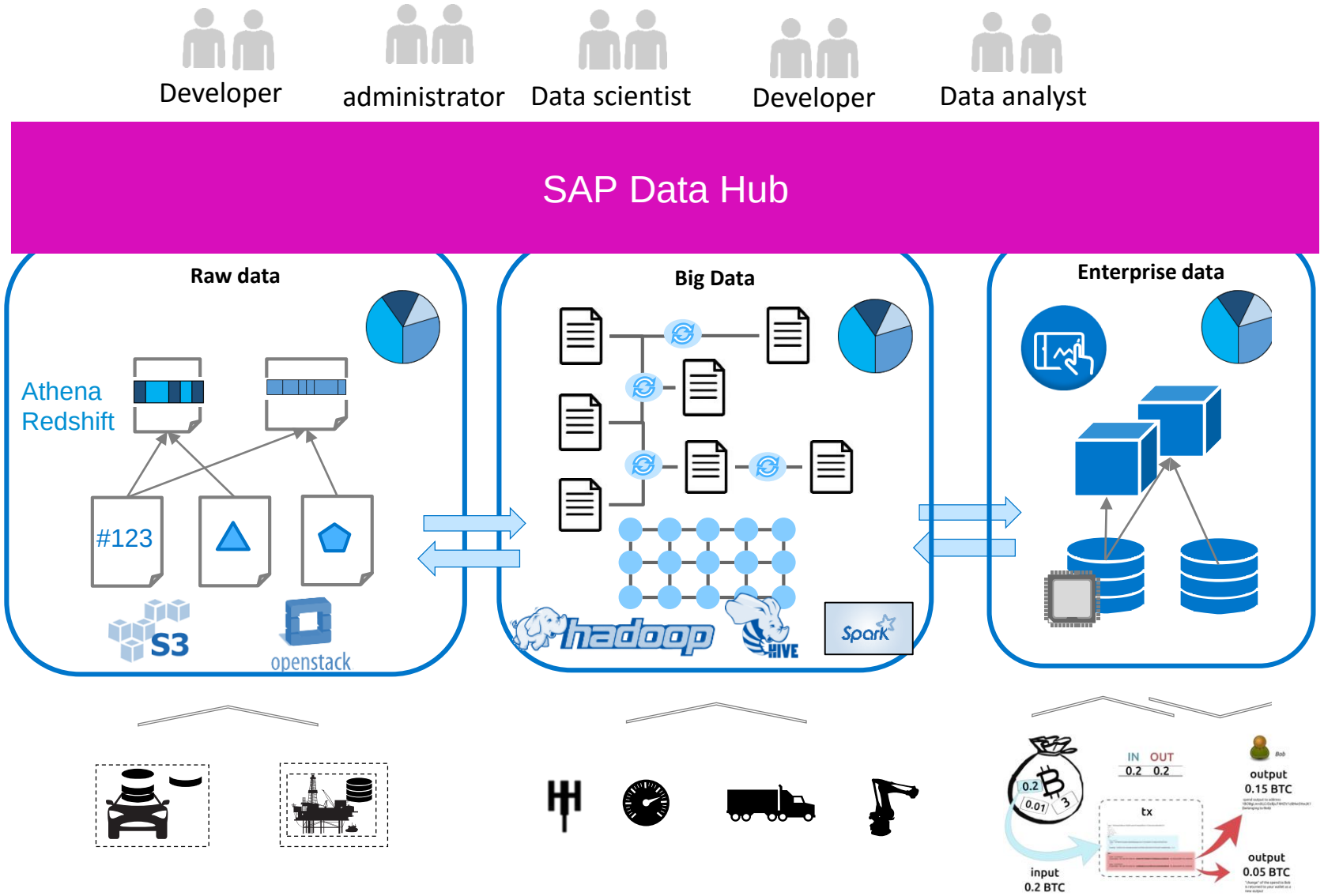
Diversity of

- Technology stacks
- Technical requirements
- Consumer persona

Data processing spread

- Within and between stacks
- Functional overlap
- Data volume and velocity

Evolution Towards Enterprise Big Data Landscape



Diversity of

- Technology stacks
- Technical requirements
- Consumer persona

Data processing spread

- Within and between stacks
- Functional overlap
- Data volume and velocity

SAP Data Hub

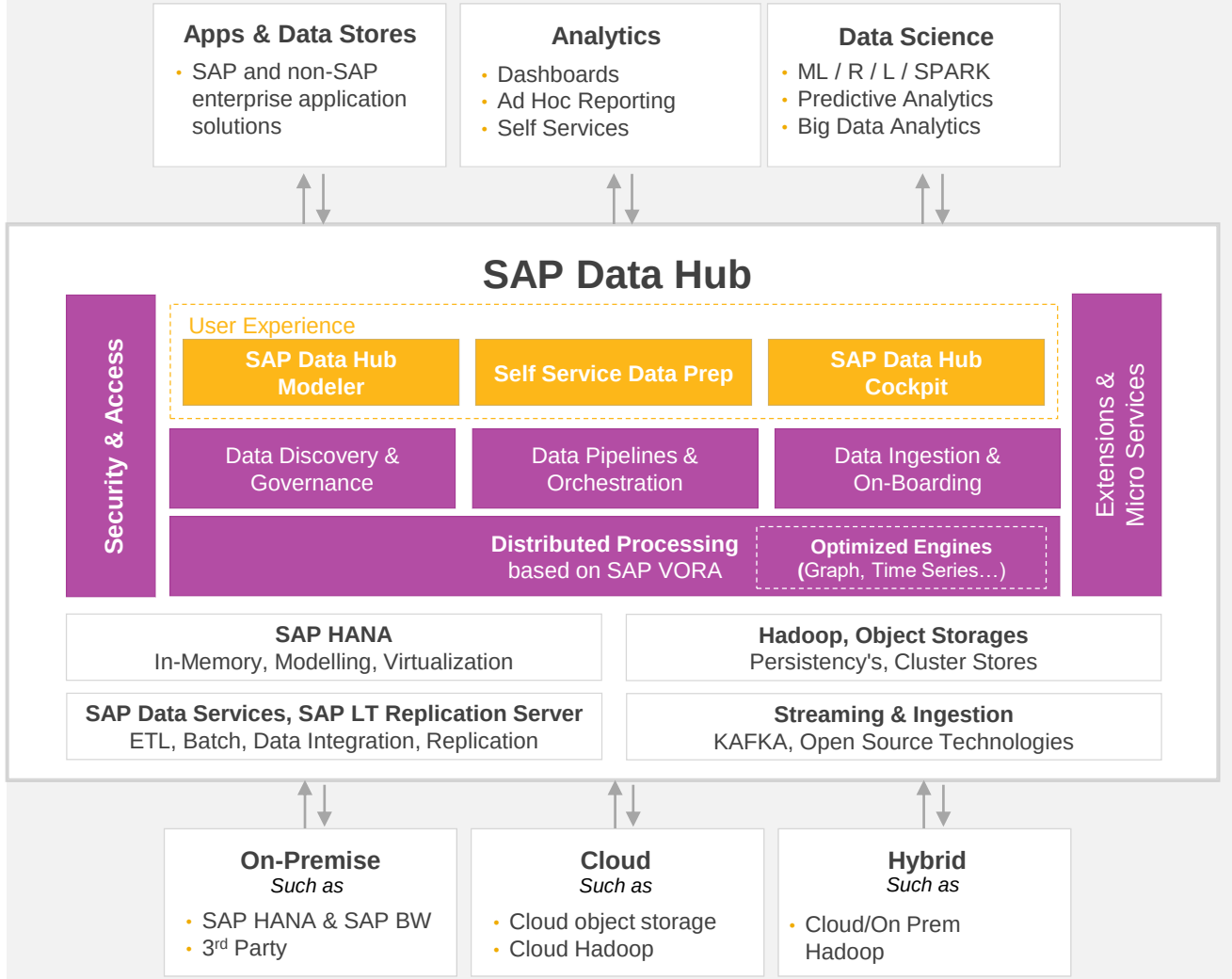
Overview and Key Functionality

Efficient Data Sharing – Leverage existing connections and integration tools, while adding new connections easily and flexibly. Access on-premise, cloud, hybrid data sources.

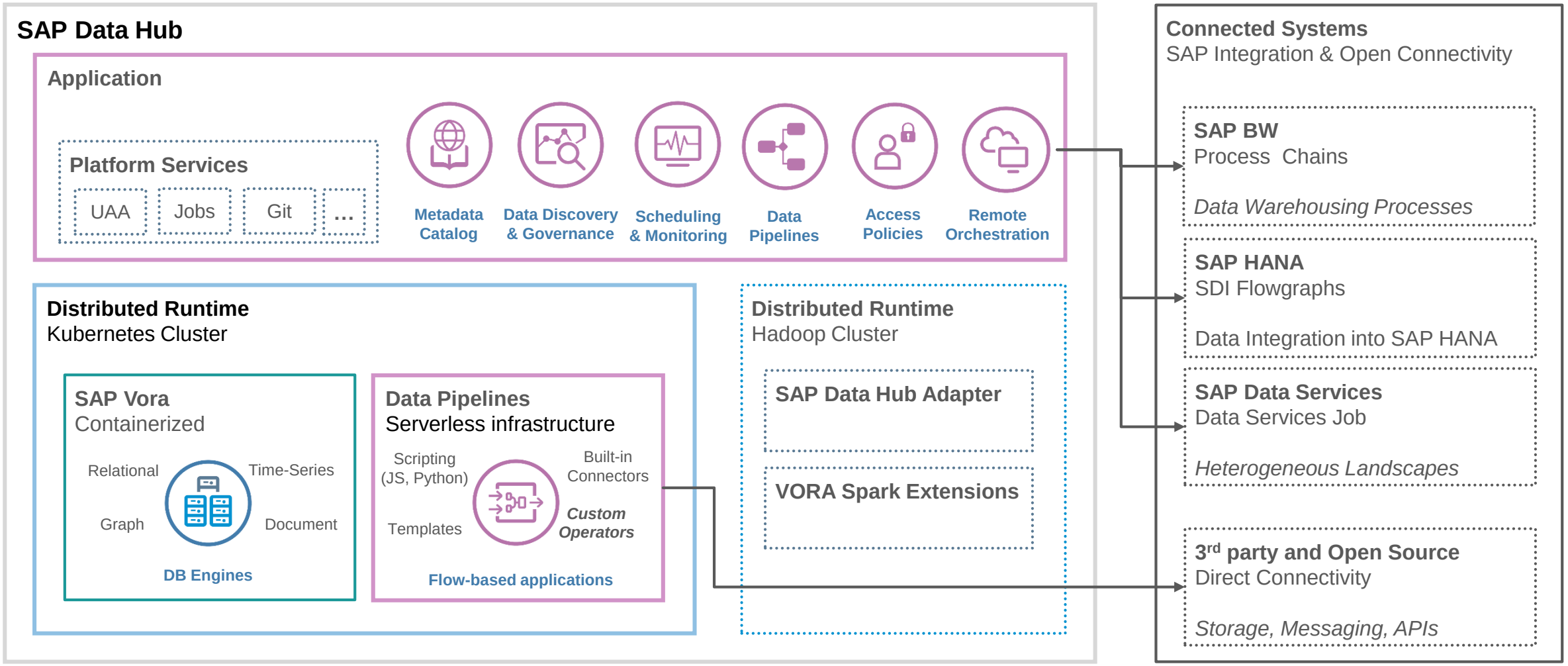
Centralized Governance – metadata repository of information stored in the connected landscape. Offering profiling, data lineage, and search capabilities

Data Pipelines – flow based applications consisting of reusable and configurable operations, e.g. data transformations, native code execution, trained models, connectors. Executes where data resides.

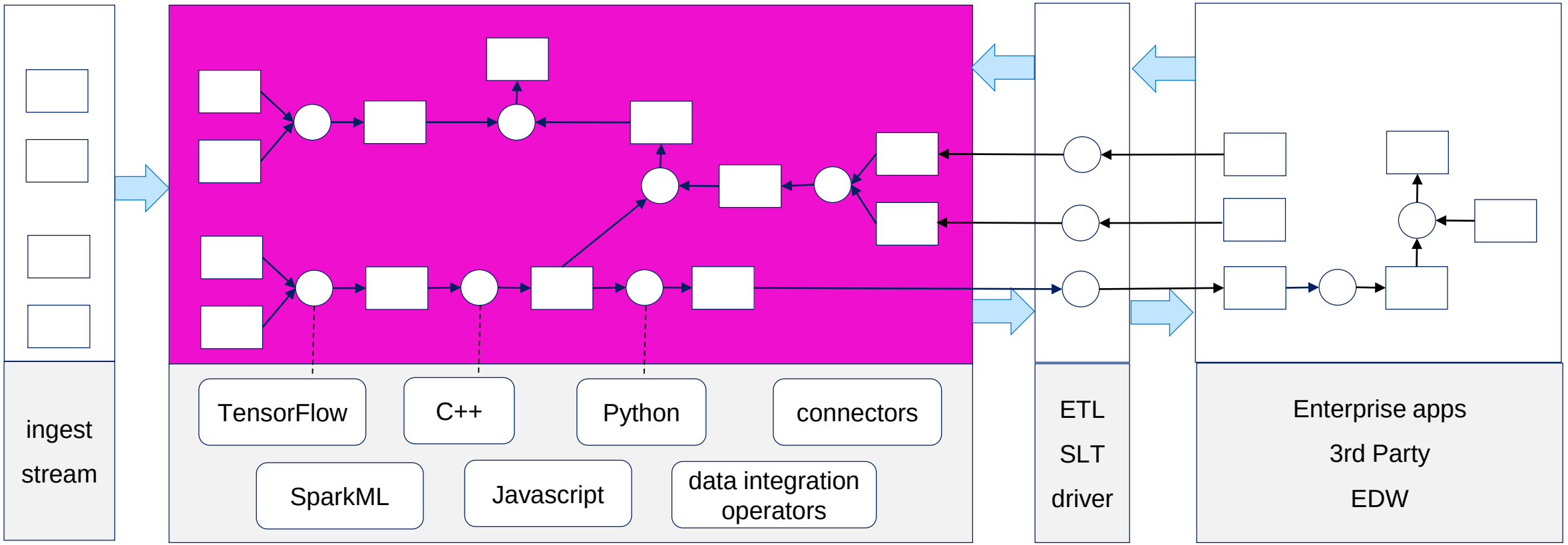
Workflows – orchestrate processes across the data landscape, e.g. executing data pipelines, triggering SAP BW Process Chains, SAP Data Services Jobs and many more



SAP Data Hub
Architecture Overview

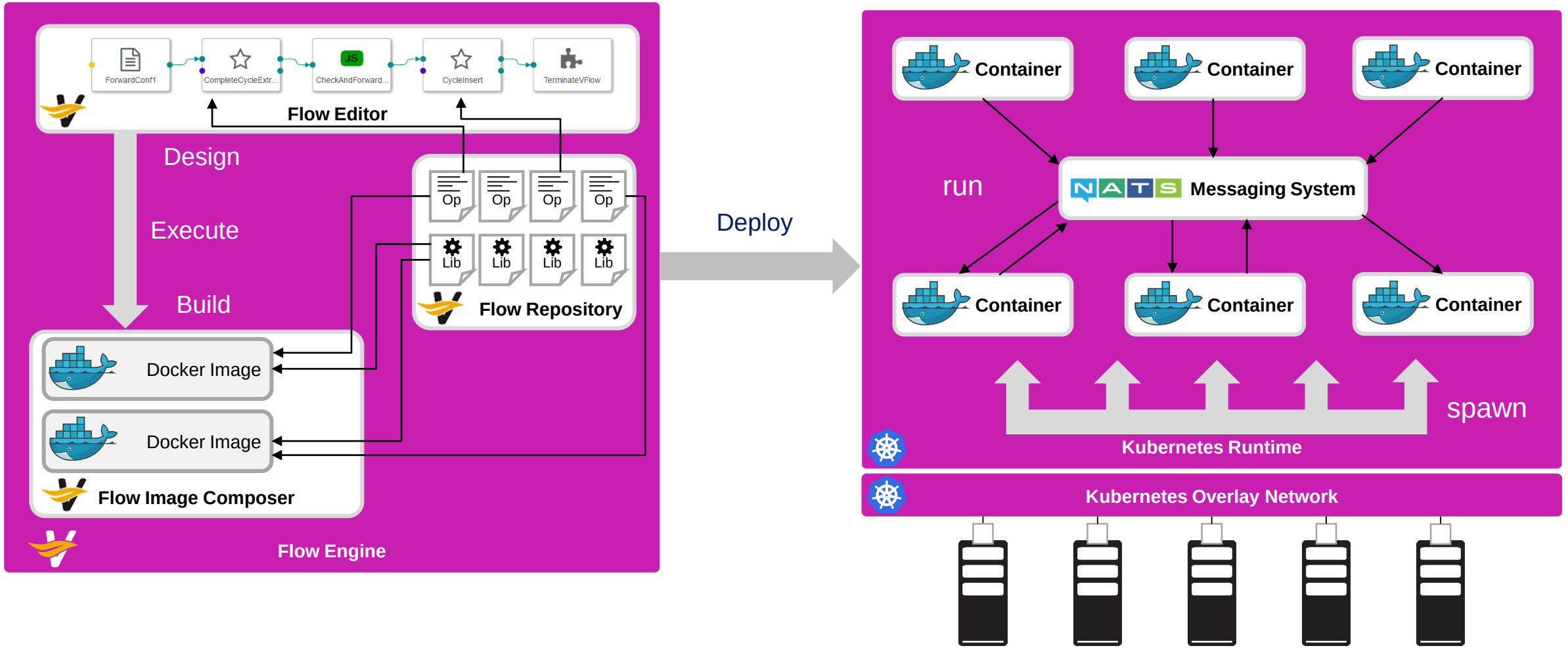


SAP Data Hub
Positioning of Data Pipelines



SAP Data Hub

Data Pipeline Design, Build and Deploy



Key Open Data Management Challenges

Unified data governance at scale

- Monitor data quality
- Understand the provenance and usage of data
- Understand what data exist and their purpose

How to bring big data processing and creation of analytics to the masses ?

- Ease the creation of customized datasets by business users and data scientists
- Ease the development of big data pipelines without compromising performance (*left out of this talk*)

Monitoring Data Quality at Scale – Assess Data Content

IT cannot protect anymore the “perimeter” !

- Data go beyond EDW and DM, good and bad data co-exist, GIGO principle is not sustainable
- Data quality must get closer to the data consumers - “make data quality everyone’s business”

Provide means to assess data content

- Profile the data and discover its meaning (e.g., what’s in a column or a record?)
  - Profiling uses predefined rules and reference data, (e.g., "100 Whatever St. 94103 San Francisco”)
  - How to collect and maintain good reference domain data at affordable price?
  - How to detect entities in a dataset?, metrics, units?
- Assess the “trust” of data
  - Measure of trust is based on data content, its provenance and usage
- Assess the “pertinence” of data for reuse
  - Data transformations may “drop” data or destroy the characteristics of data in some pipeline; more pertinent to reuse pipeline’s input data

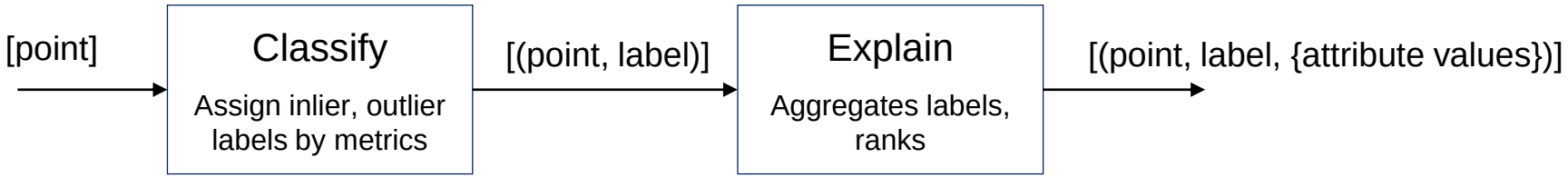
Monitoring Data Quality at Scale – Continuous Monitoring

Continuously monitoring the quality of data

- Current methods based on IT-governed quality rules … do not scale
  – Statistical methods: infer patterns or ranges of values in a field, discover association rules in data samples
  – User-defined rules: field constraints, row-level data dependencies. Order of 100 – 1,000 rules

- Involve the user in error detection and learn “subjective” quality monitoring rules
  – E.g., learn a data dependency rule between 2 fields

- Discover anomalies in the data when no rule exists and high velocity precludes manual inspection
  – Detect changes in features used by a ML pipeline (e.g., drop from 90% to 60% of non-nulls in a feature)
  – Generic classification and interpretation operators for fast data stream analysis



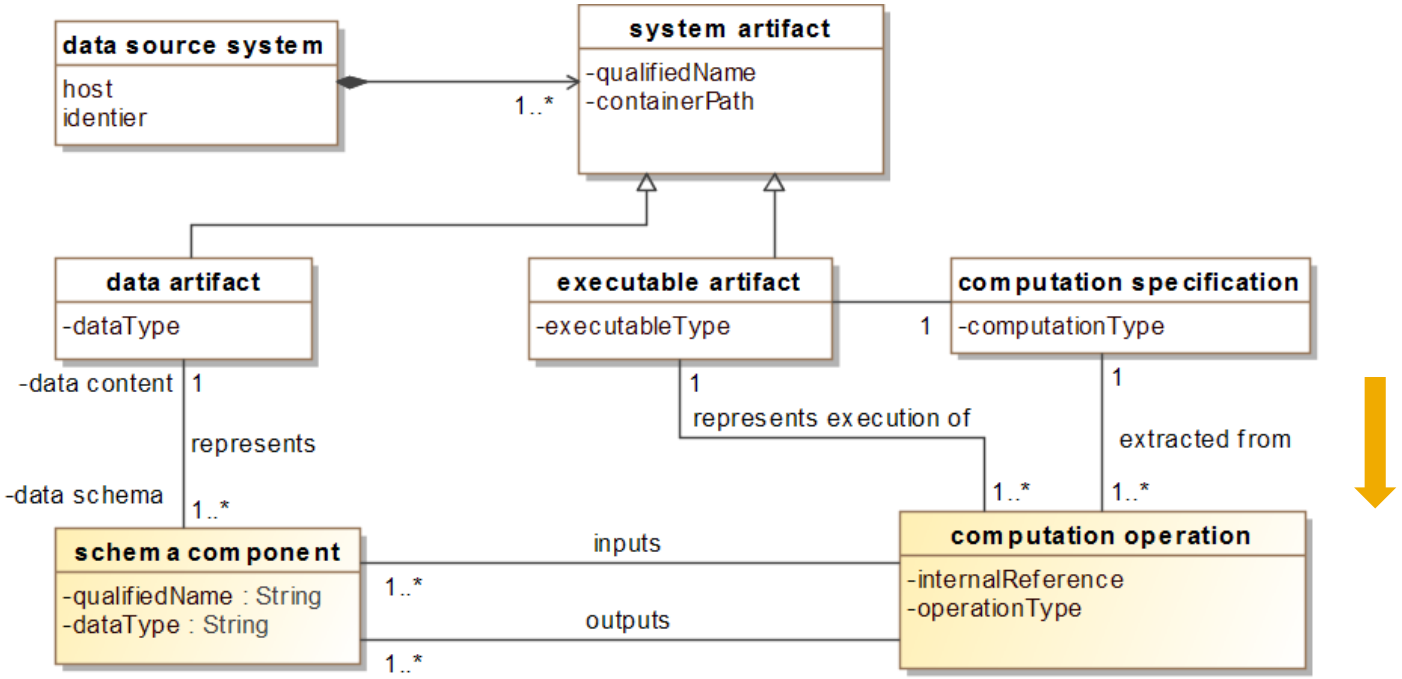
(see Macrobase paper, SIGMOD 2017))

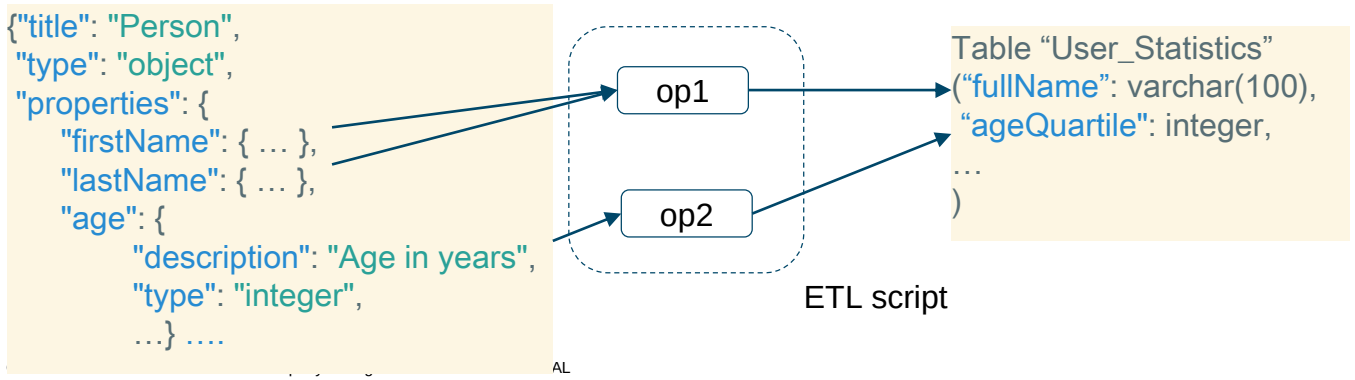
Monitoring Data Quality at Scale – Repair the Data

Repair quality issues

- No automatic correction/repair ! Don’t loose the data! ⇒ human-driven remediation
- Notify errors, group them and present to users for review
  – for predefined rules (e.g., bad address), repair actions can be suggested; in others cases we can’t
- How to conduct root cause analysis ? ⇒ errors can be detected lately in the data pipeline – lineage analysis
- What to do with user feedback?: fix data?, fix data transformation?, create intermediate data ?, …
- How to propagate the repair action? ⇒ impact analysis

Schema-level Data Lineage – The Usual Approach





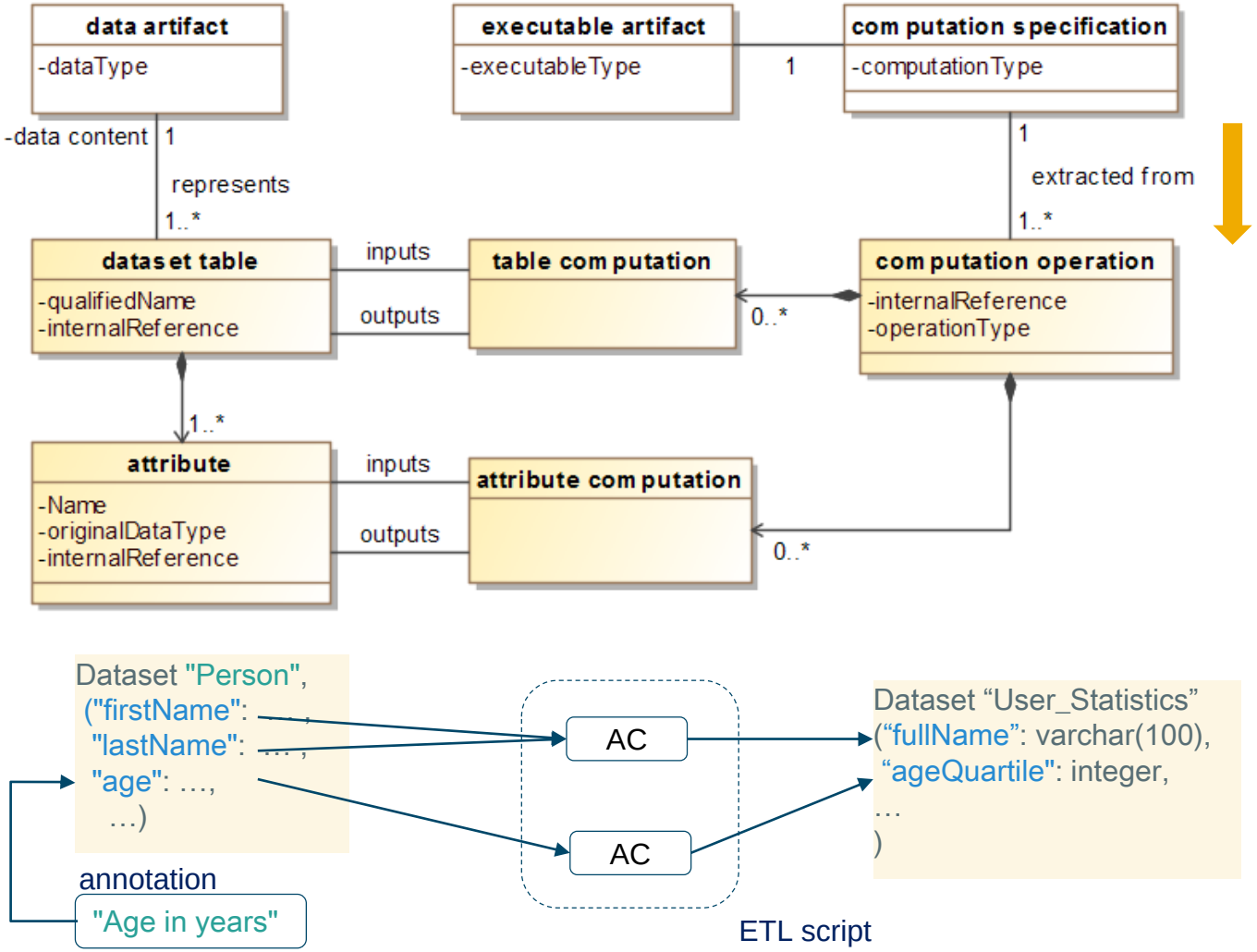
Traces the lineage between schema components through computation operations

- **Data lineage extractors** dedicated to artifact type and computation specifications
- **Data lineage storage** contains results of extractors and dedicated graph data structures
- Provides lineage or impact analysis to developers/designers

Not well suited for data governance scenarios

- No uniform view of data lineage because data models are heterogeneous
  - E.g., 320 schema components in SAP IS !
- Impact and analysis hard to interpret because too many sorts of operations
  - E.g., which schema components contribute values for a given schema component?

Uniform Schema-level Data Lineage



Provide a dataset abstraction level

- Table-based representation of data artifacts
- Uniform abstract representation of operations
- **Data lineage extractors** generate a uniform representation of lineage

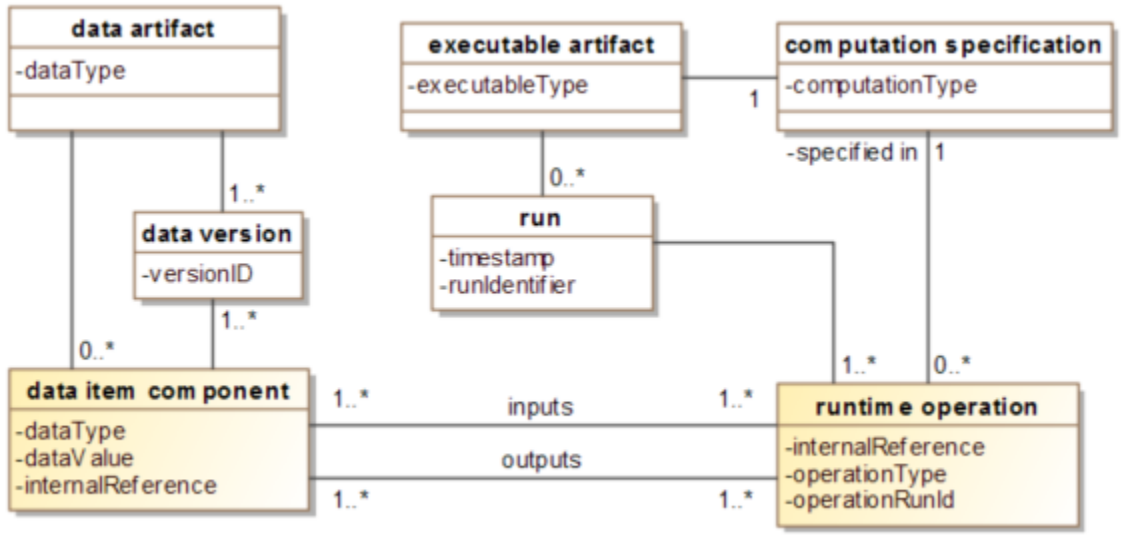
Benefits:

- Uniform metadata catalog (incl. data lineage) – coupled with data wrappers to visualize the data
- Basis for expressing additional semantic information – Annotations, tags, relationships, …
- Facilitates lineage and impact analysis algorithms (heterogeneity is resolved)

Schema-level Data Lineage – Open Issues

- Discover lineage from « blackbox » operators
  - When no extractor is available or no access to computation specification
  - Discover attribute lineage from data analysis and dataset metadata properties
- Capture dynamic inputs
  - Inputs of operators provided at runtime (e.g., name of a dataset, or query string “select … from”)
  - Dynamic queries generated during execution of computation operations
- Enable application developers to express lineage within computation specifications
  - Provide a universal API to capture lineage at runtime
- Assure reproducibility of results and sustainable explanations
  - Manage versions of system artifacts

Instance-level Data Lineage



Traces the lineage between data item components through runtime computations

- Captures “exact” lineage even with dynamic inputs
- Reproducible results, sustainable explanations
- Provides fine-grain lineage to data analysts
- Provides debugging facilities to developers, designers

Eager approach:

- generates and stores data lineage at runtime
  - e.g., by adapting the code of computations, wrapping library of data manipulations, or using light-tracing and replay on-demand in trace mode
- Issues: performance overhead of computations, size of data lineage storage

Lazy approach

- generates instance-level data lineage on-demand by generating dynamic queries that use schema-level lineage and knowledge of computation operation semantics
- Issues: not always possible so lack of precision, looses reproducibility

Instance-level Data Lineage – Open Issues

Mixed approach that maximizes the precision of the lineage while minimizing performance overhead of computations and space overhead of data lineage storage

- Multiple tradeoffs to handle!
- Solutions exist in the context of relational and data cleansing operations
- Optimized techniques exist for eager lineage in big data pipelines or scientific workflows

Integration with domain-specific instance-level data lineage solutions

- solutions for data pipelines in Hadoop Map-Reduce or Spark
- scientific workflow management systems

Data Governance Queries over Unified Metadata Catalog

Governance query example

- *Find reports on “internal financial” topic, accessible to “financial analysts”, and for each report return: the primary data source systems and the business owner*

Evaluation

- Retrieve datasets such that:
  - dataset.type = ‘report’ *// authored property*
  - dataset.topic = ‘internal financials’ *// authored property*
  - exists x such that accessList (dataset.qualifiedName, x) and jobProfile (x, y) and y = ‘financial analyst’
- For each retrieved dataset, find its source datasets using lineage analysis, and return properties:
  - (sourceDataset.name, sourceDataset.system.name)
  - Dataset.businessOwner *// authored property*

Requires schema-level lineage, metadata properties, and security policies

Data Governance Queries over Unified Metadata Catalog

Governance query example

- *What is the freshness of the datasets directly used by application "A" with respect to the freshness of their primary source of data?*

Evaluation

- Retrieve datasets such that:
  - Contains(dataset.application, 'A') *// authored property*
- For each retrieved dataset, find source datasets used to generate it and return:
  MAX(setof(sourceDataset.latestRefreshDate)) – dataset.latestRefreshDate

Requires instance-level lineage, metadata properties, and runtime execution metadata

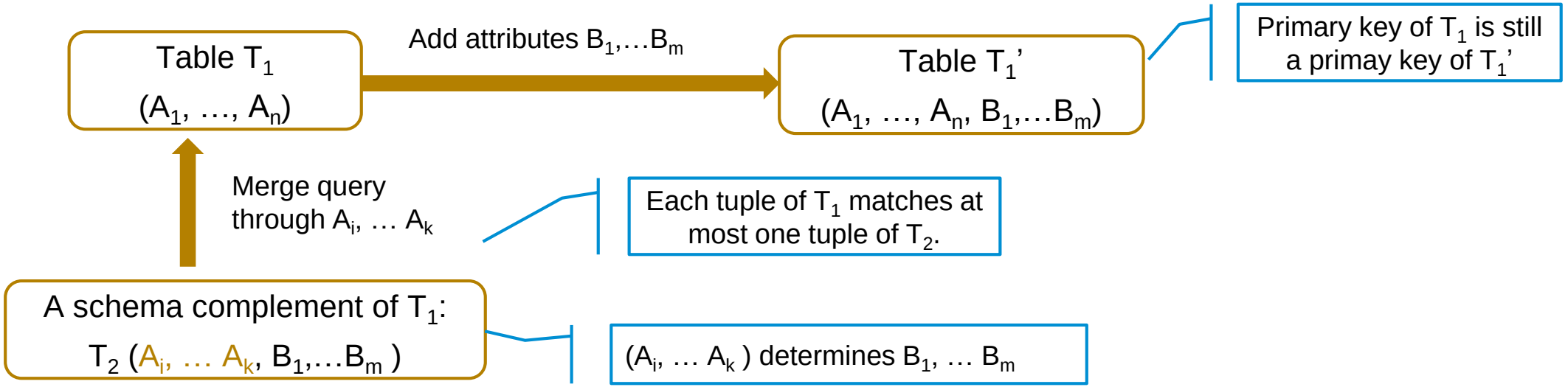
Ease the Creation of Customized Datasets

Users are empowered to create their data because

- More users need more data but IT cannot handle all user demands
- Governed data do not match user needs

Problem: help users customize and share their data

- A frequent operation is “schema complement” (e.g., combine analytics, feature engineering)



[A. Das Sarma and al, "Finding Related Tables" , VLDB 2012]

Discover and Rank Generalized Schema Complements

PRODUCT_SALES

PRODUCT	CITY	STATE	COUNTRY	CONTINENT	REVENUE	UNIT
Plato Salt	Dublin	Ohio	United States	North America	1,000	Euro
Skinner Cola	Dublin	California	United States	North America	5,000	USD
Plato Salt	Vancouver	British Columbia	Canada	North America	3,500	USD
Booker 1% Milk	Vancouver	British Columbia	Canada	North America	600	USD
Plato Salt	Dublin	-	Ireland	Europe	1,200	USD
Plato Salt	Dublin	-	Belarus	Europe	7,000	USD
Plato Salt	Paris	-	France	Europe	1,000	Euro

POPULATION

COUNTRY	YEAR	COUNTRY_POPULATION
France	2015	64,395,345
France	2016	64,668,129
Canada	2015	35,939,927
Canada	2016	36,286,378
United States	2015	321,773,631
United States	2016	324,118,787

Problem:
POPULATION **is not** a schema complement, although it can be used to generate schema complements

Discovery of Schema Complements – Open Issues

Understand relations between datasets

- Extract relationships from metadata and computation specifications (e.g., data lineage)
- Discover entities and associations using data profiling

Generalized schema complements

- Handle analytic datasets (hierarchical dimensions and measures)
- Query-generated schema complements

Adapted strategies

- Customized analytics require different strategies than feature engineering for ML
- Personalized strategies

Summary

SAP Data Hub vision

- Rich data sharing and connectivity capabilities
- Enterprise level centralized governance data landscape
- Dataflow pipelines with server-less execution
- Orchestration of heterogeneous and distributed tasks

Data management challenges

- Data quality monitoring must scale with number of users and datasets, and velocity of data
  - essential for both data governance and the development and maintenance of data pipelines
- Data lineage is critical
  - Spectrum of solutions needed depending on the use case
- Empowered users need sophisticated solutions to customize their data
  - Finding generalized schema complements is a first direction

Thank you.

Contact information:

Eric Simon
eric.simon@sap.com

Run Simple