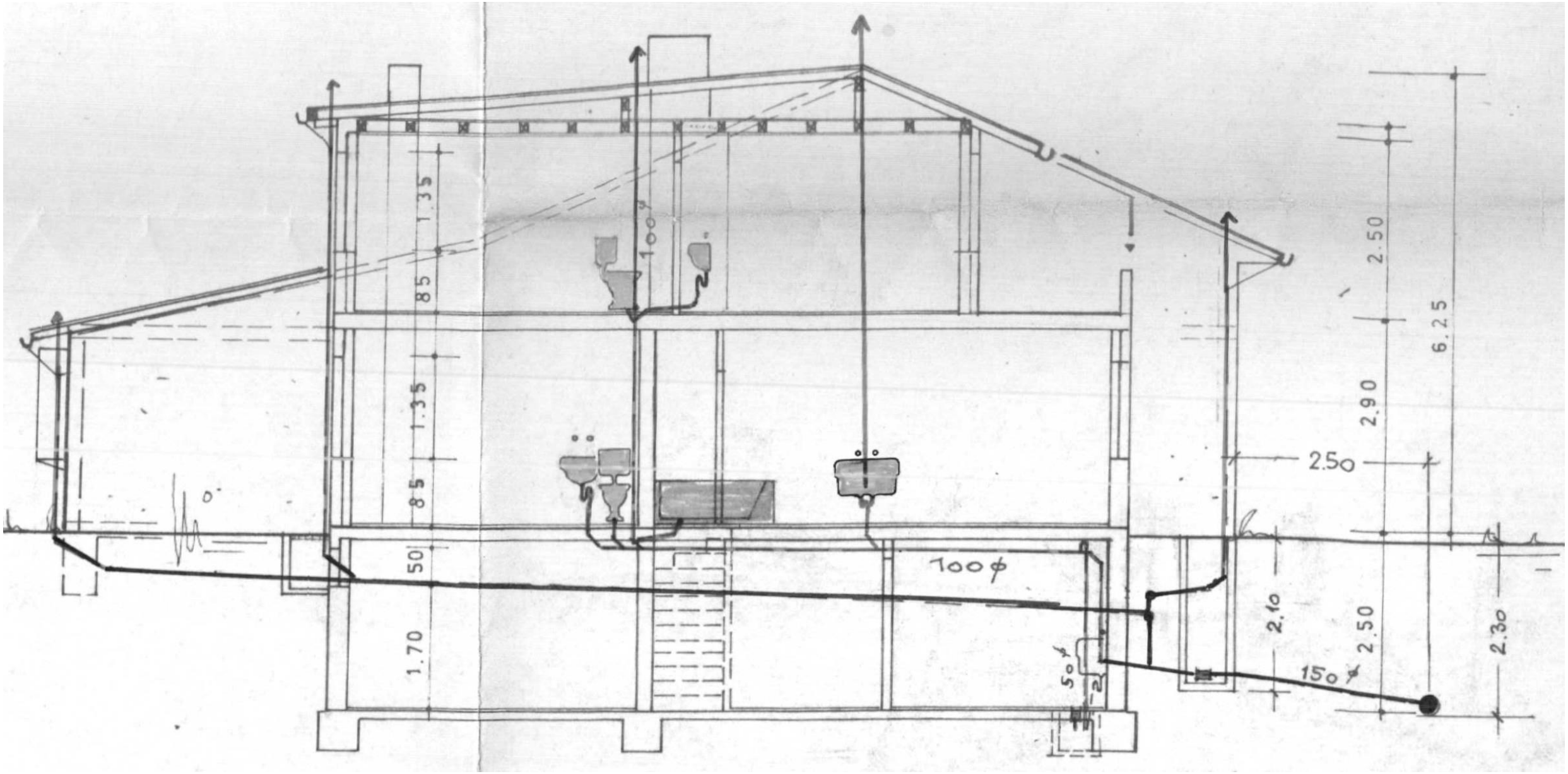
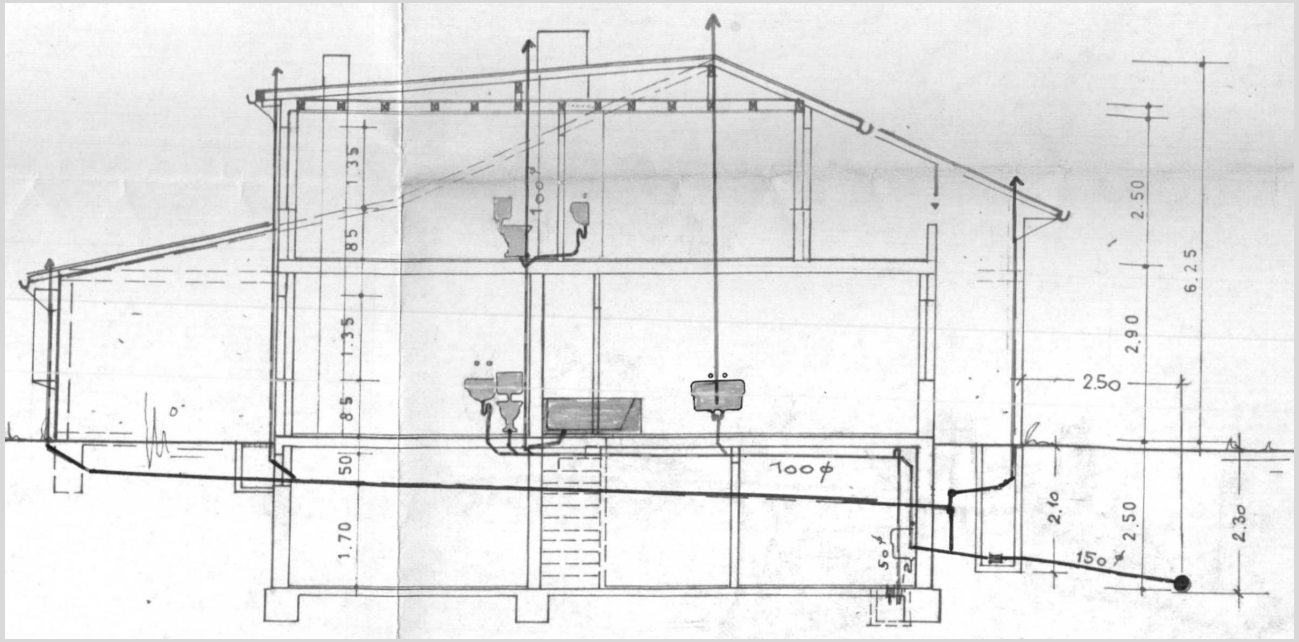
Data Vault Pipeline Description

Dokumentformat zur Entkopplung von DWH Entwicklungswerkzeugen

Matthias Wegner, cimt ag


1.35
85
1.35
85
50
1.70
100φ
700φ
50φ
2.50
2.90
6.25
2.50
2.10
2.50
150
2.30


1.35
85
1.35
85
50
1.70
100φ
2.50
2.90
6.25
2.50
2.10
2.50
2.30
150φ
50φ

- Motivation
- Konkreter Blick
- Demo
- Designprinzip
- Github Projekt
- Nutzung

cimt

cimt

Motivation

Der Data Warehouse Entwicklungsprozess

cimt

Anforderung

- Ziel & Informationsbedarf
- Herleitungs/ Transformations-beschreibung
- Datenherkunft

Modellierung

- Profiling der Quelldaten
- RAW Vault Modell
- Mart Modell
- Business Vault Modell

Implementierung

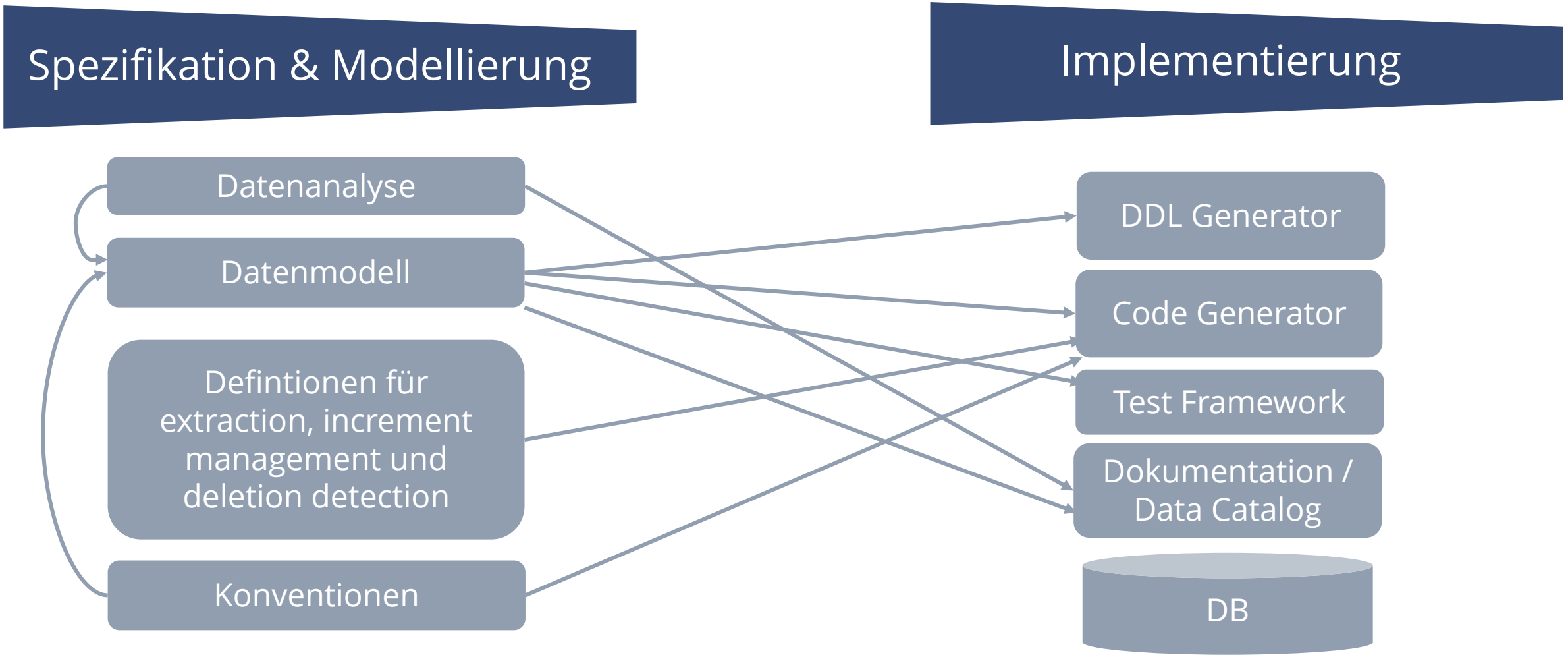
- Technische Anbindung
- Inkrementverwaltung
- Fetch
- Parse
- Stage
- Load
- Export/Send

Betrieb

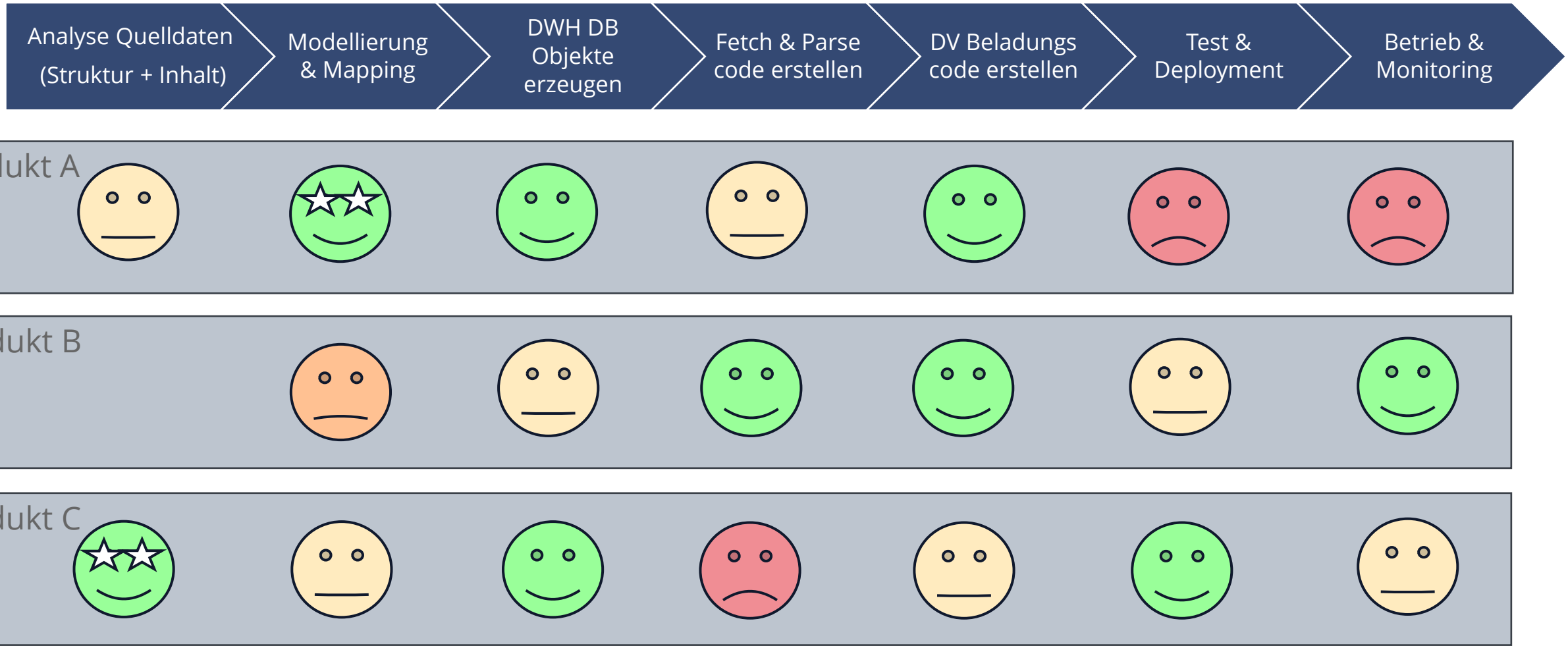
- Deployment
  - DB Objekte
  - Pipeline/Job
- Scheduling
- Monitoring
- Störungs-management

Werkzeugkasten der Entwicklungsteams

cimt

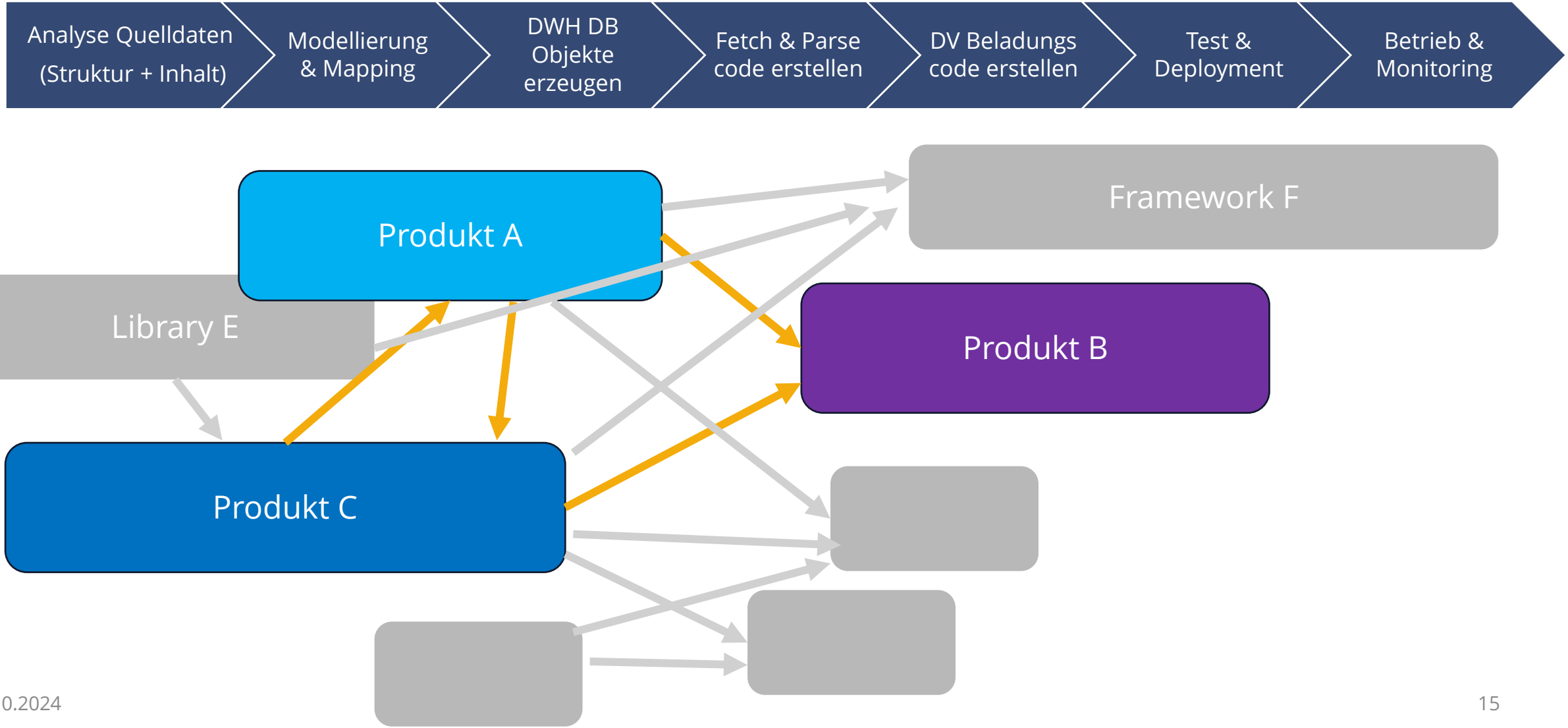


„Data Vault Produkte“ - Funktionsabdeckung

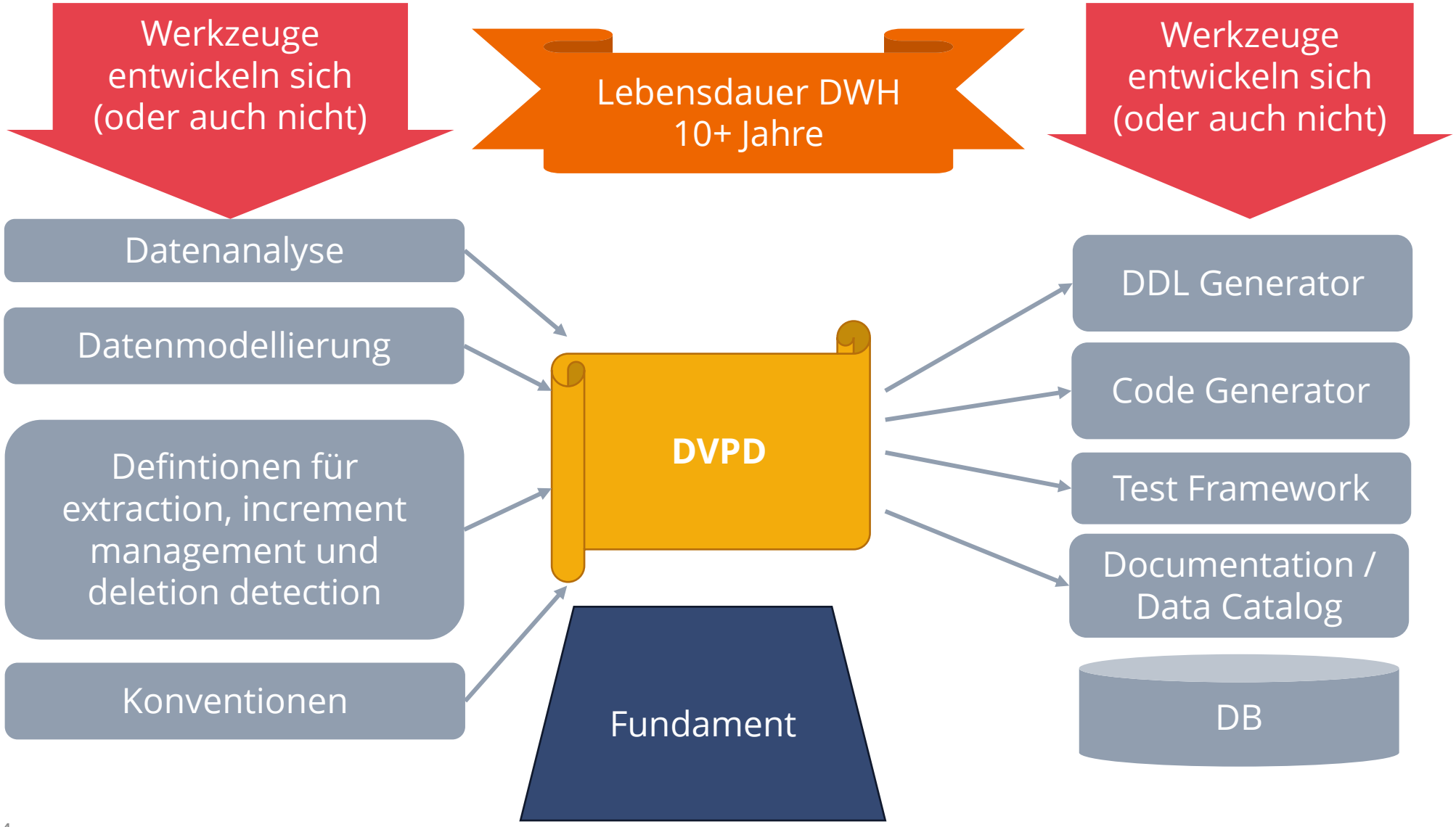


Produktzusammenstellung

cimt

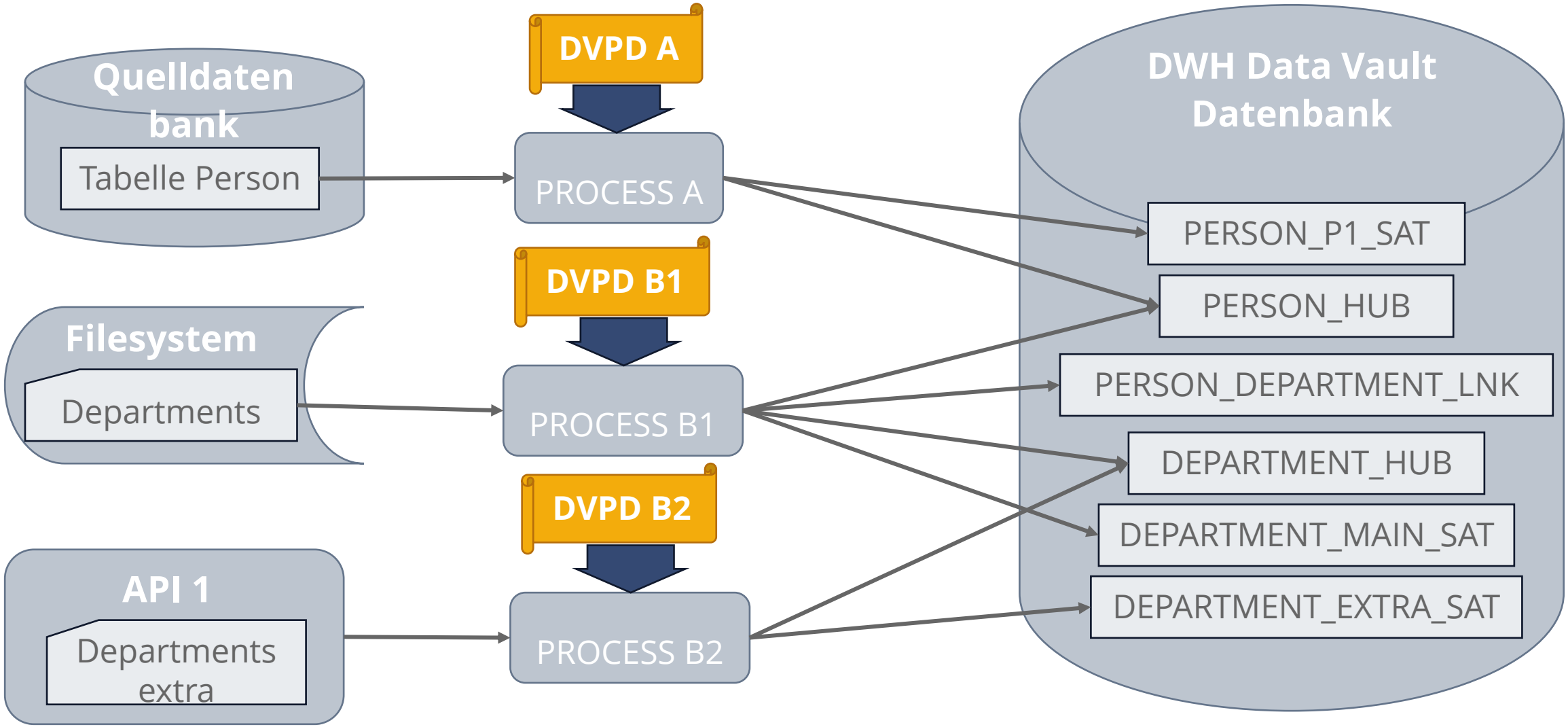


Data Warehouses leben lange



Wie sieht eine DVPD konkret aus ?

Eine DVPD beschreibt den konkreten physischen Ladeprozess



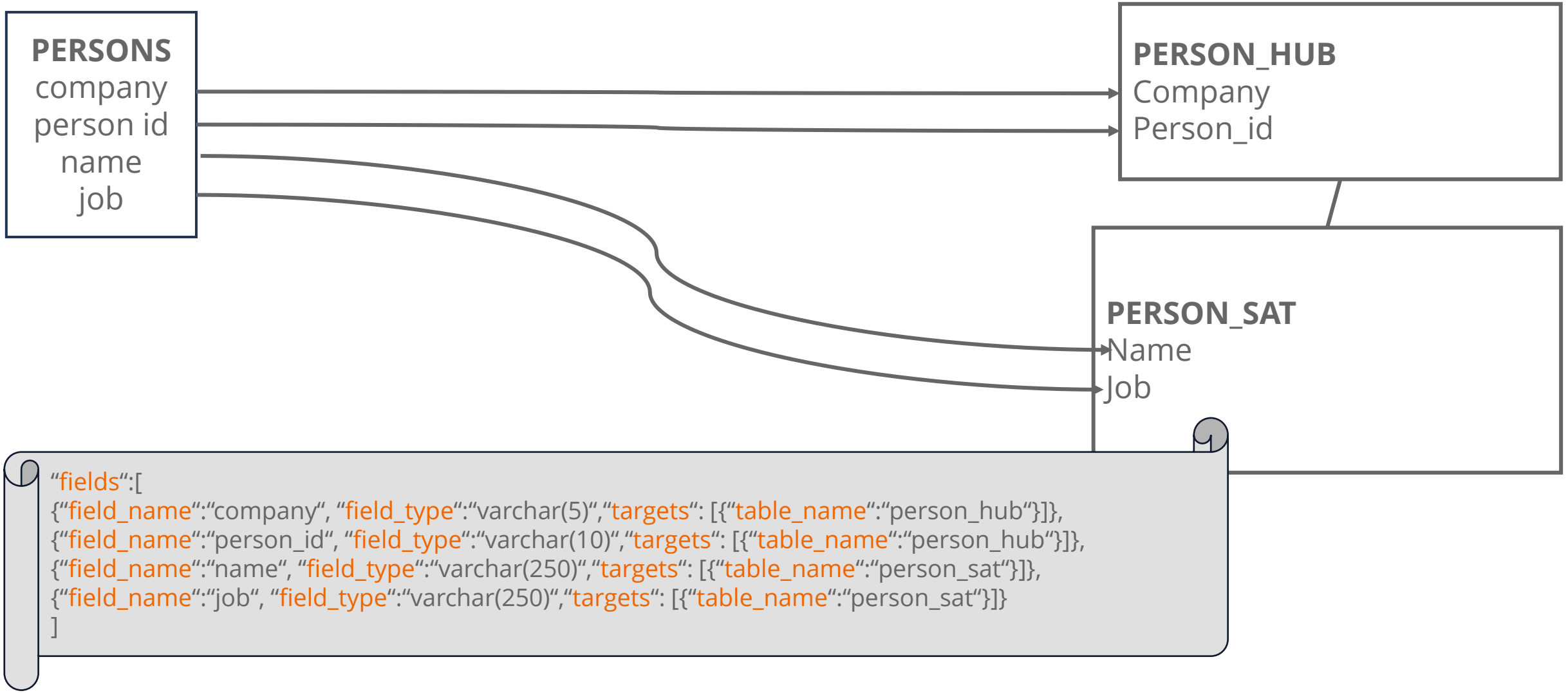
Einfache Modellbeschreibung …

PERSONS
company
person id
name
job

```
"fields":[
{"field_name":"company", "field_type":"varchar(5)"},
{"field_name":"person_id", "field_type":"varchar(10)"},
{"field_name":"name", "field_type":"varchar(250)"},
{"field_name":"job", "field_type":"varchar(250)"}
]
```

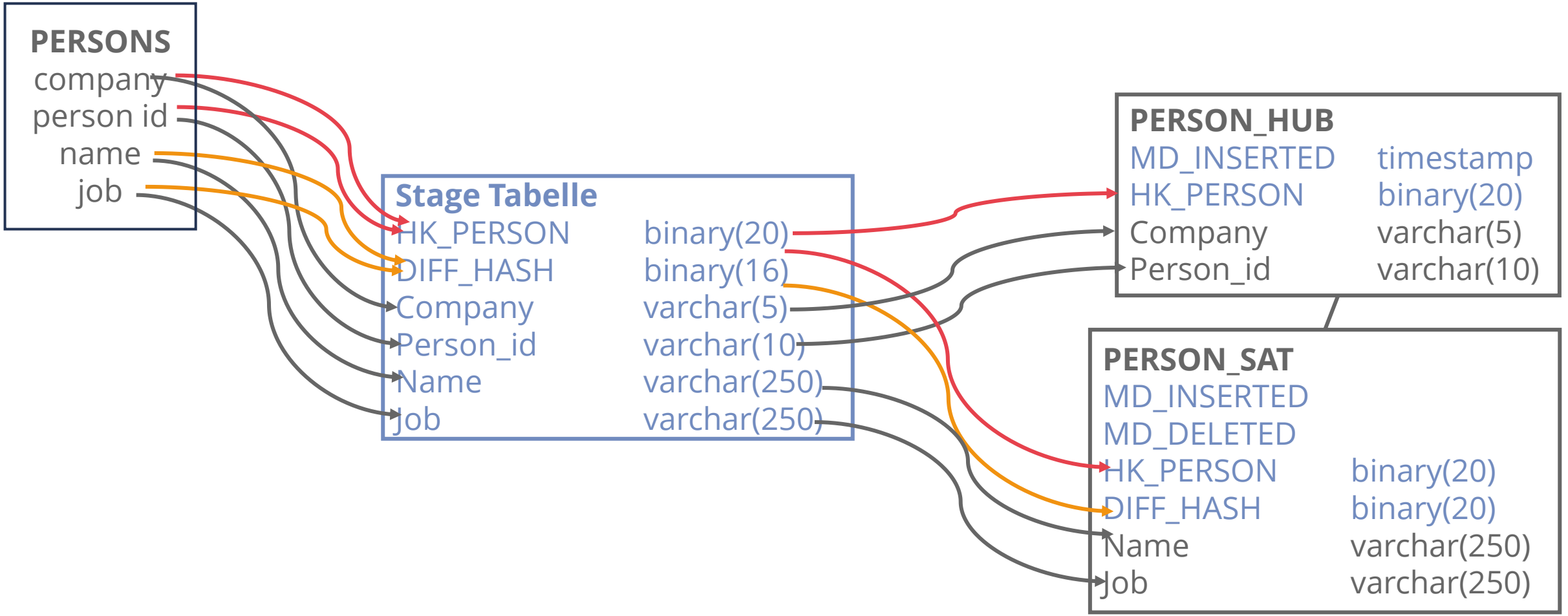
```
"data_vault_model":
{"table_name":"person_hub ", "table_stereotype":"hub"},
{"table_name":"person_sat ", "table_stereotype":"sat",
"satellite_parent_table":"person_hub"},
]
```

Einfache Mappingbeschreibung




```
"fields":[
{"field_name":"company", "field_type":"varchar(5)","targets": [{"table_name":"person_hub"}]},
{"field_name":"person_id", "field_type":"varchar(10)","targets": [{"table_name":"person_hub"}]},
{"field_name":"name", "field_type":"varchar(250)","targets": [{"table_name":"person_sat"}]},
{"field_name":"job", "field_type":"varchar(250)","targets": [{"table_name":"person_sat"}]}
]
```

Aus der einfachen Beschreibung prinzipbedingte Schlussfolgerungen ziehen



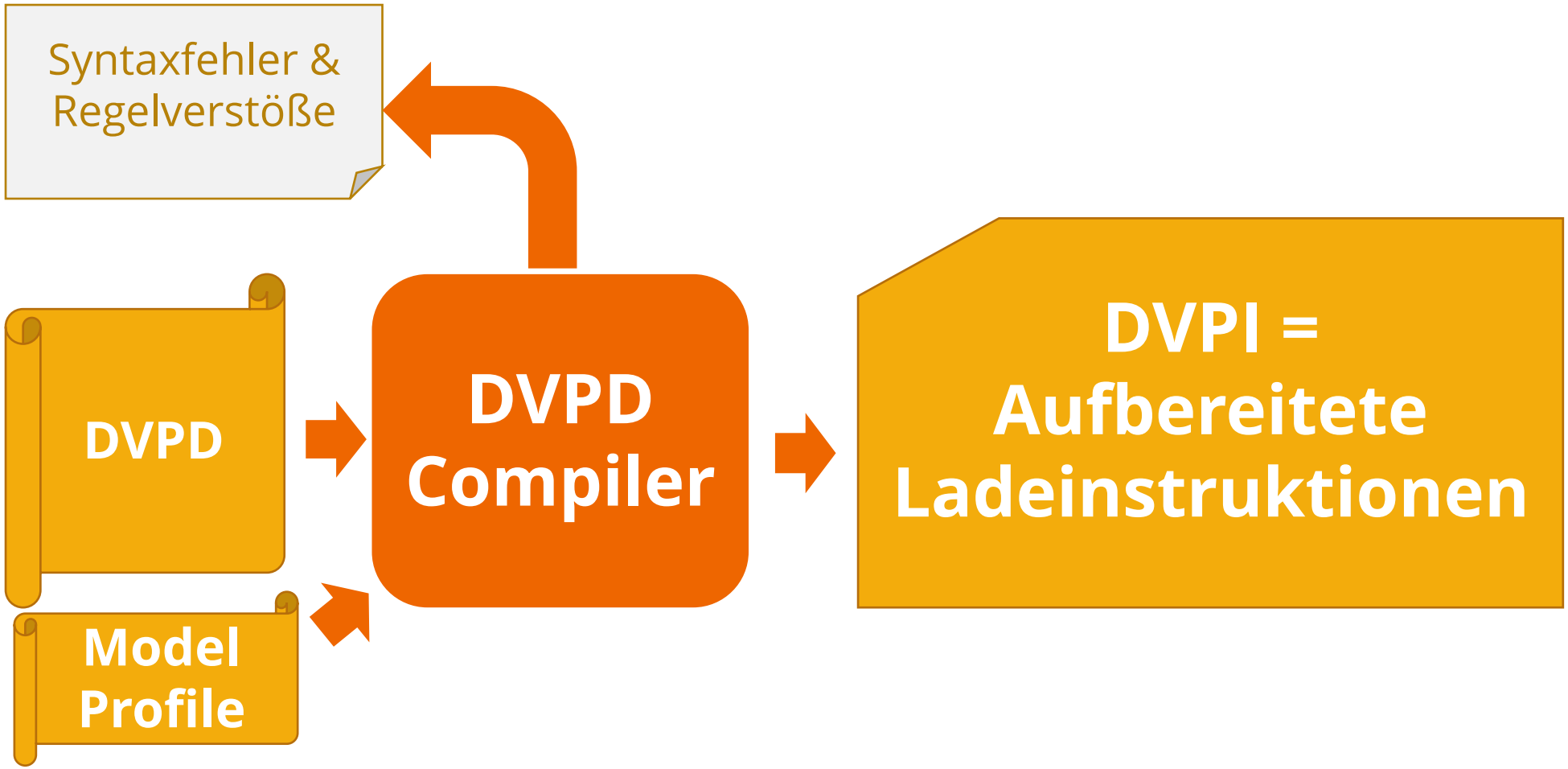
JSON Syntax

{
"dvpd_version":"0.6.0",
"stage_properties":[{"stage_schema":"stage_rvlt"}],
"pipeline_name":"hr_person",
"record_source_name_expression":"company.org.persons",
"data_extraction":{"fetch_module_name":"oracle_db_table_full", "source_system_name":"ora_company",
"source_schema":"org_structure", "source_table":"persons"},
"data_vault_model":[
{"schema_name":"rvlt_company",
"tables":[
{"table_name":"person_hub","table_stereotype":"hub"},
{"table_name":"person_sat","table_stereotype":"sat", "satellite_parent_table":"person_hub"}
]
}
]
"fields":[
{"field_name":"company", "field_type":"varchar(5)", "targets":[{"table_name":"person_hub"}]},
{"field_name":"person_id","field_type":"varchar(10)", "targets":[{"table_name":"person_hub"}]},
{"field_name":"name", "field_type":"varchar(250)","targets":[{"table_name":"person_sat"}]},
{"field_name":"job", "field_type":"varchar(250)","targets":[{"table_name":"person_sat"}]}
],
}

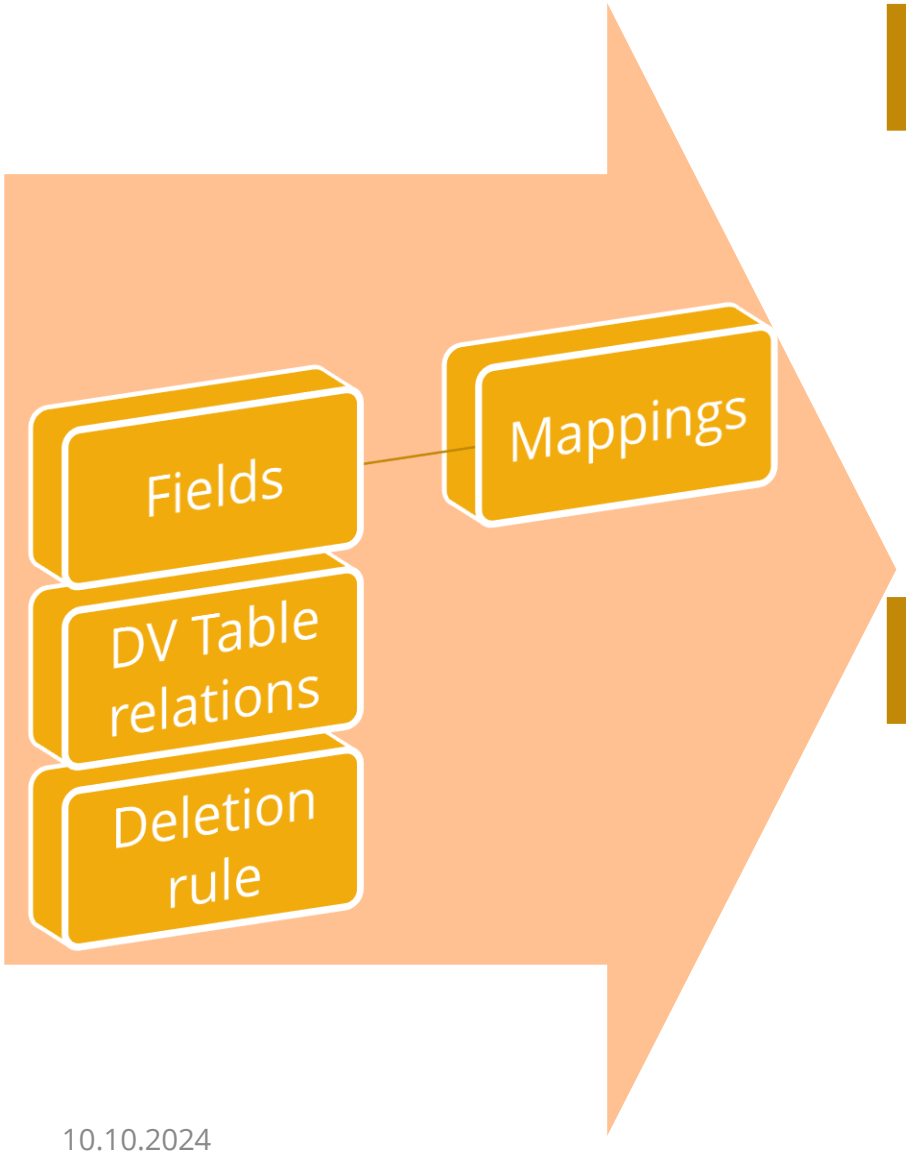
Alles (!) was eine Beladung beschreibt

Von Quelle bestimmt	▪ Abrufverfahren und Transport aus der Quelle ▪ Inkrementmethode ▪ Erkennung von gelöschten Daten ▪ Datenformat und Zerlegung in Felder (Parsing)
Von Data Vault Methode und Zielplattform bestimmt	▪ Zieldatenmodel ▪ Mapping

Compiler und
Data Vault Pipeline Instruction (DVPI)

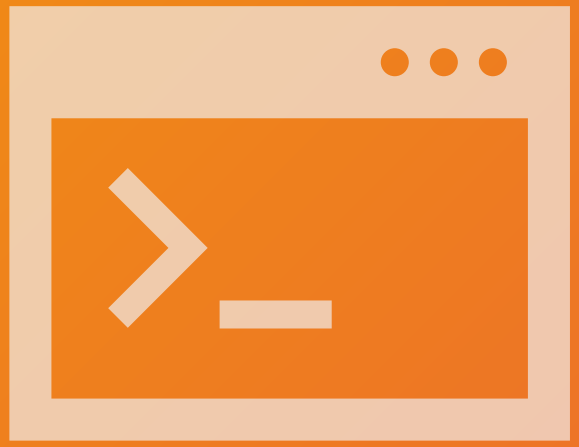


Struktur der DVPI



- Data Vault Tables
  - Table properties
  - Columns (meta, hash, content)
- Parse sets
  - Fields
  - Hashes
    - Field list & ordering properities
  - Stage table columns
    - Field mappings
  - 1-n Load operations per table
    - field-stage-column mappings
    - Deletion rule

cimt



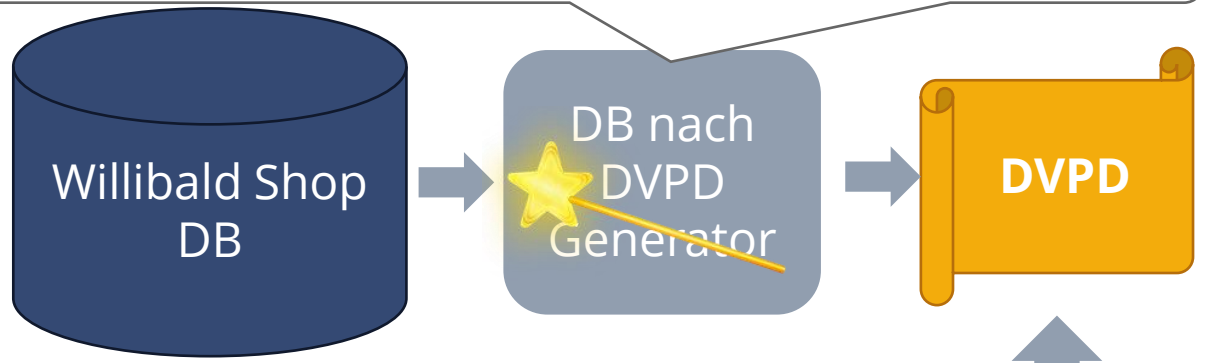
Demonstration

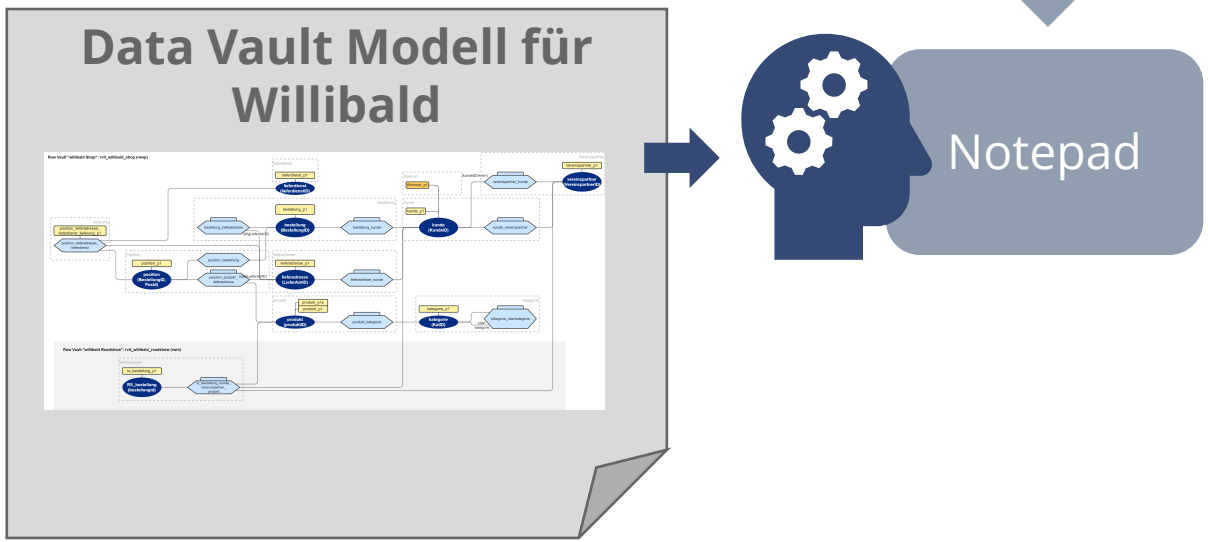
DEMONSTRATION

„Generieren einer DVPD aus der Quellstruktur“

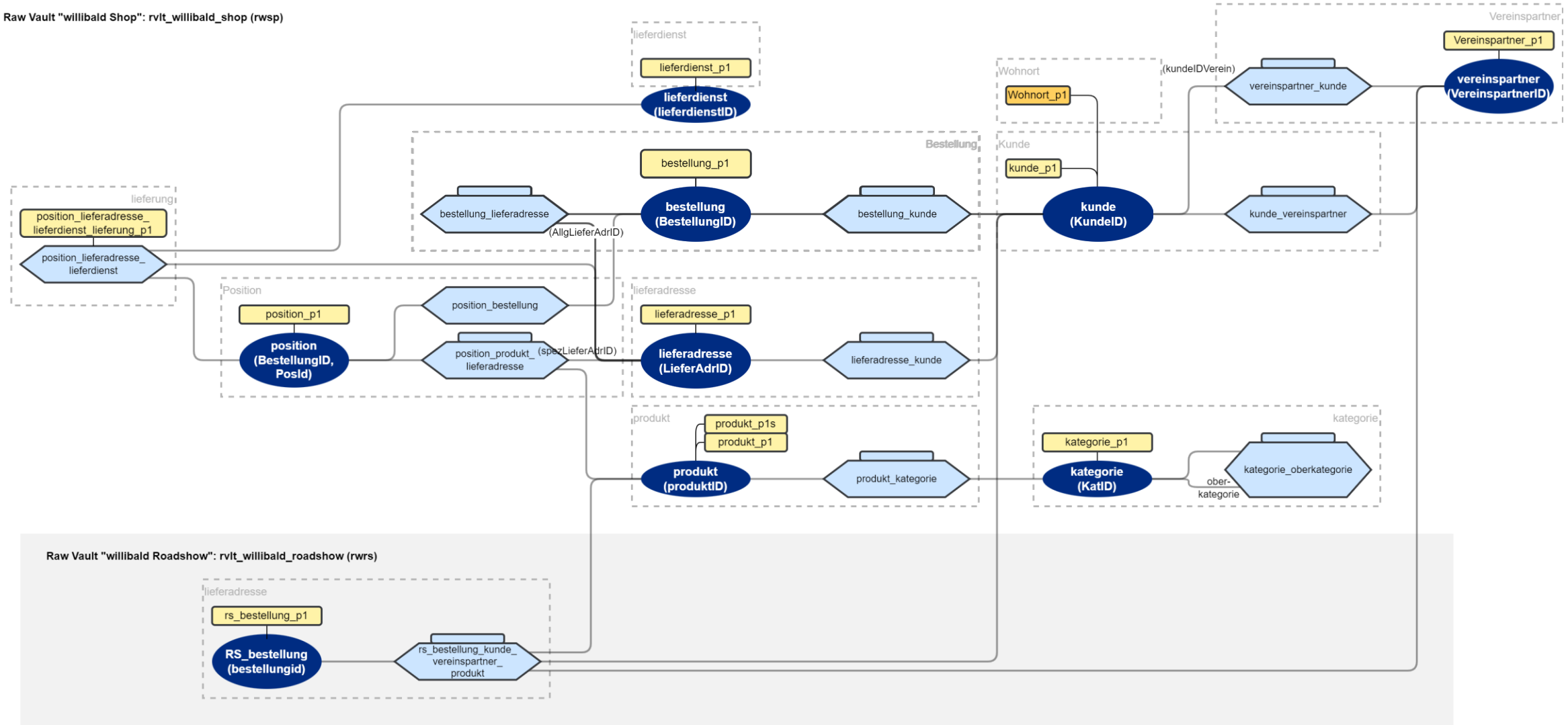
cimt

```
dvpd_generate_from_db willibald_shop_p1 bestellung
```

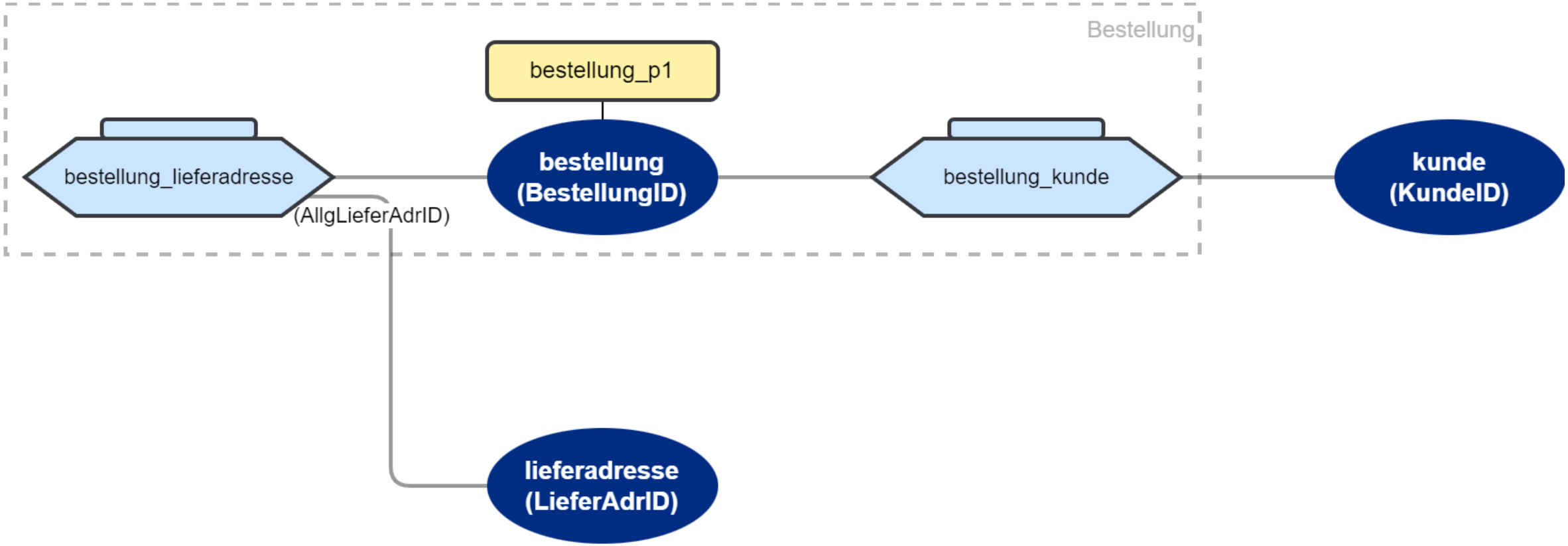




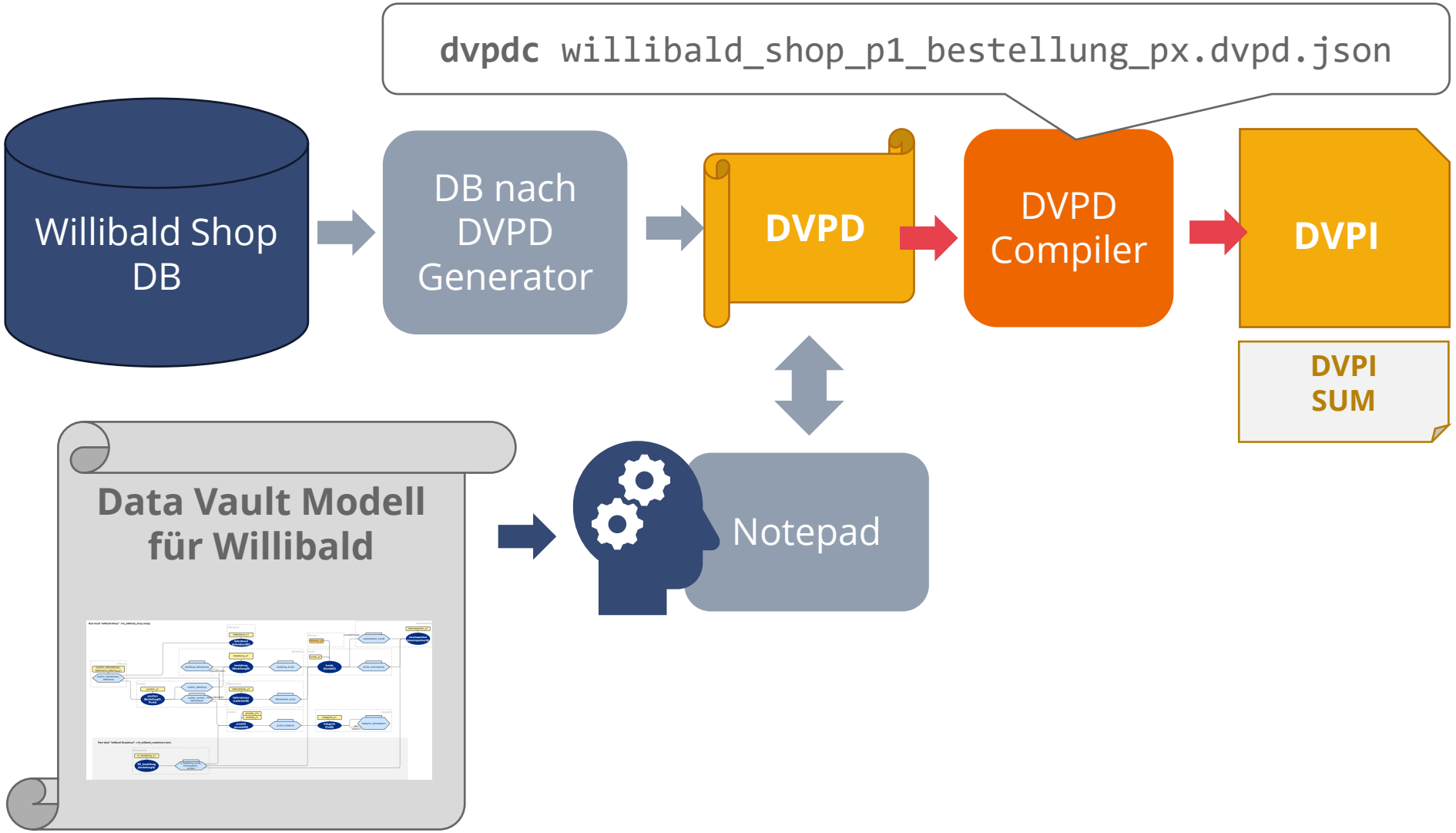
Modell der Pipeline



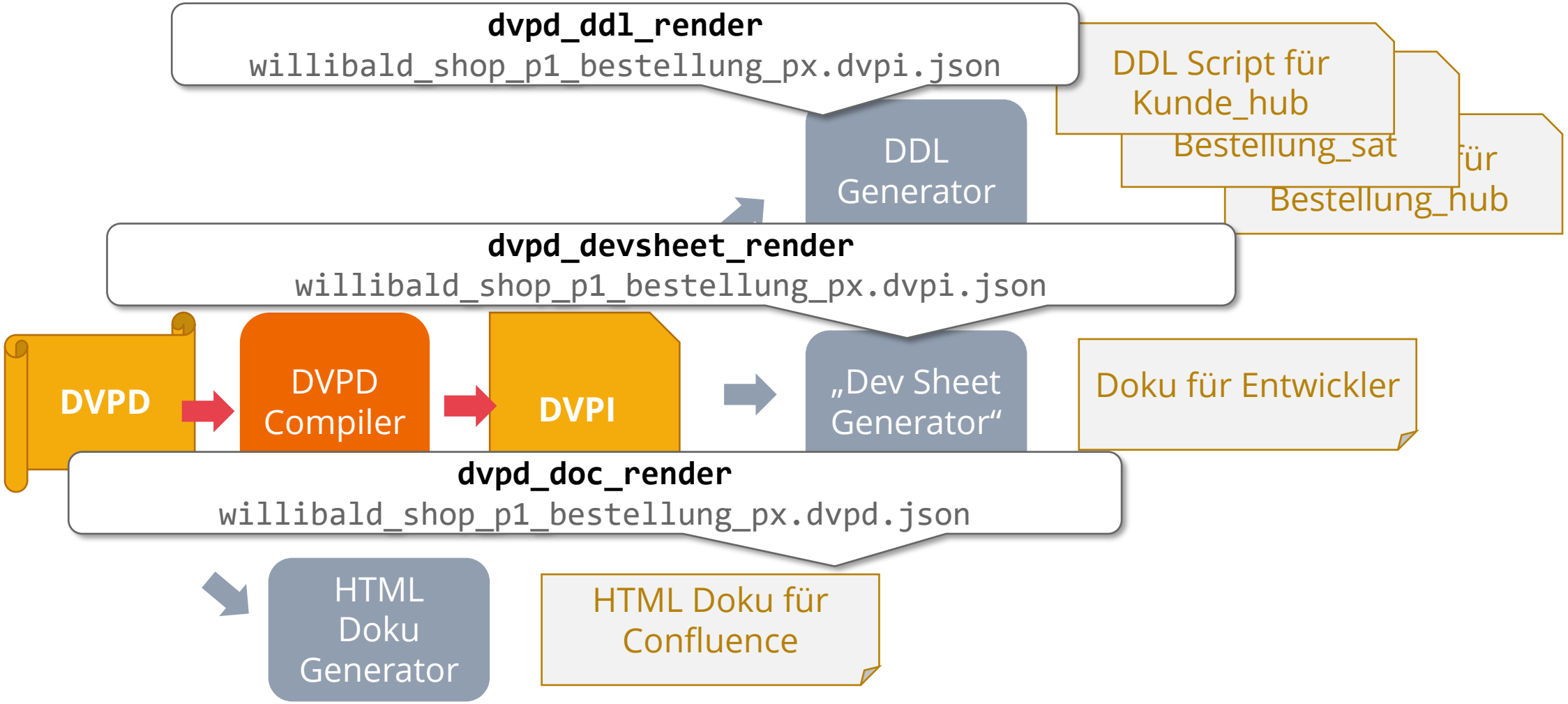
Modell der Pipeline



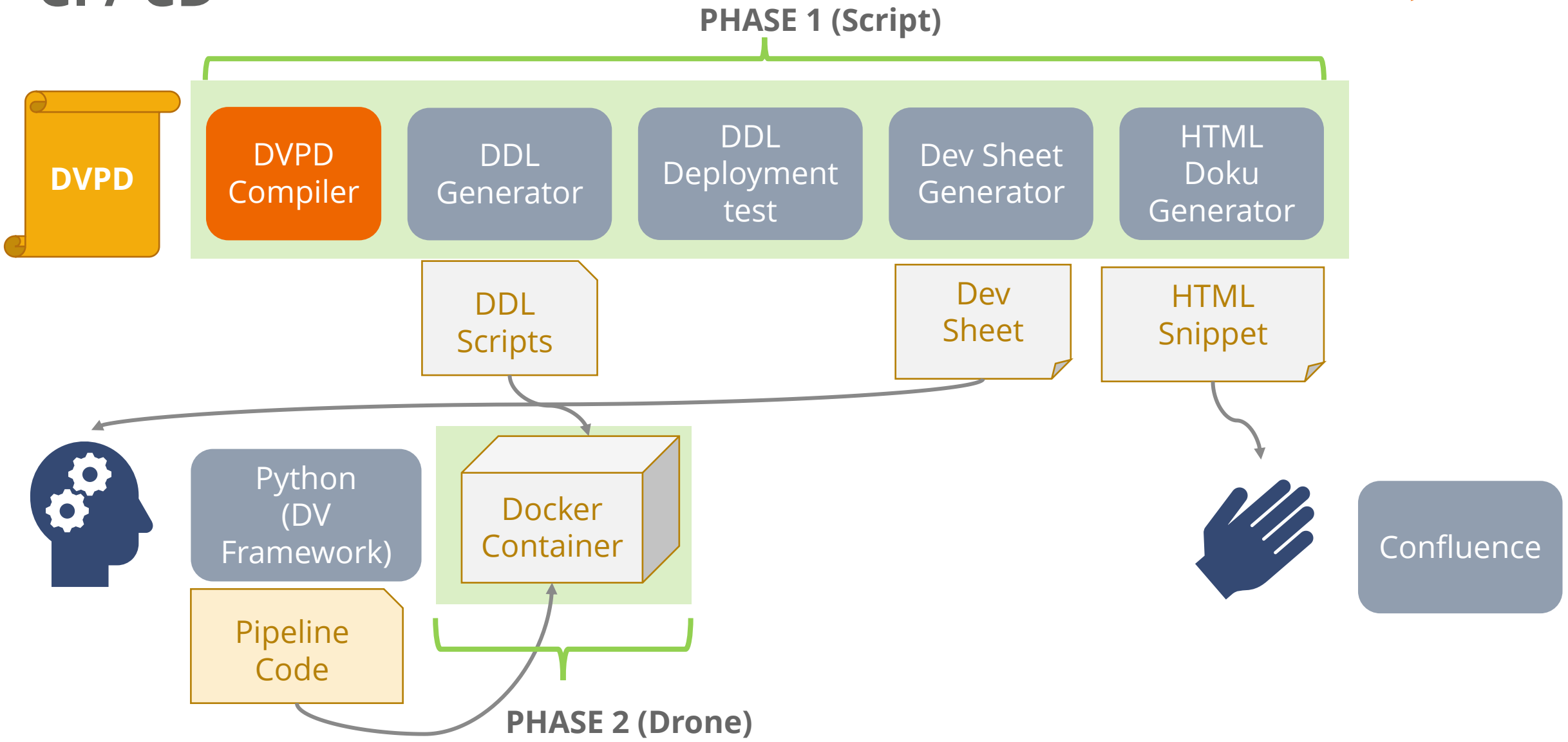
Prüfung der DVPD mit dem Compiler



Demo „Ergebnisse aus der DVPD“



CI / CD


PHASE 1 (Script)
DVPD
DVPD Compiler
DDL Generator
DDL Deployment test
Dev Sheet Generator
HTML Doku Generator
DDL Scripts
Dev Sheet
HTML Snippet
Python (DV Framework)
Docker Container
Pipeline Code
Confluence
PHASE 2 (Drone)

cimt

Designprinzipien der DVPD Syntax

„Den ganzen Weg gehen“

DVPD =

cimt

Komplette Beschreibung eines Datenladeprozesses

Vollständige Unterstützung der Data Vault Methode

Standpunkt der Syntax ist die Modelltransformation

DVPD =

cimt

Komplette Beschreibung eines Datenladeprozesses

Vollständige Unterstützung der Data Vault Methode

Standpunkt der Syntax ist die Modelltransformation

DVPD =

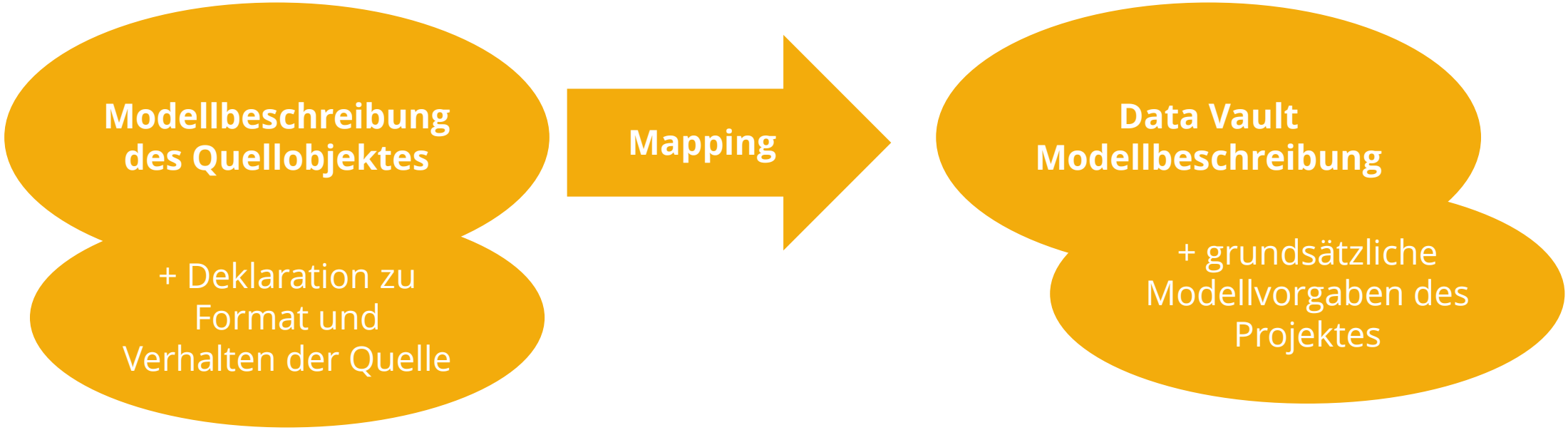
cimt

Komplette Beschreibung eines Datenladeprozesses

Vollständige Unterstützung der Data Vault Methode

Standpunkt der Syntax ist die Modelltransformation

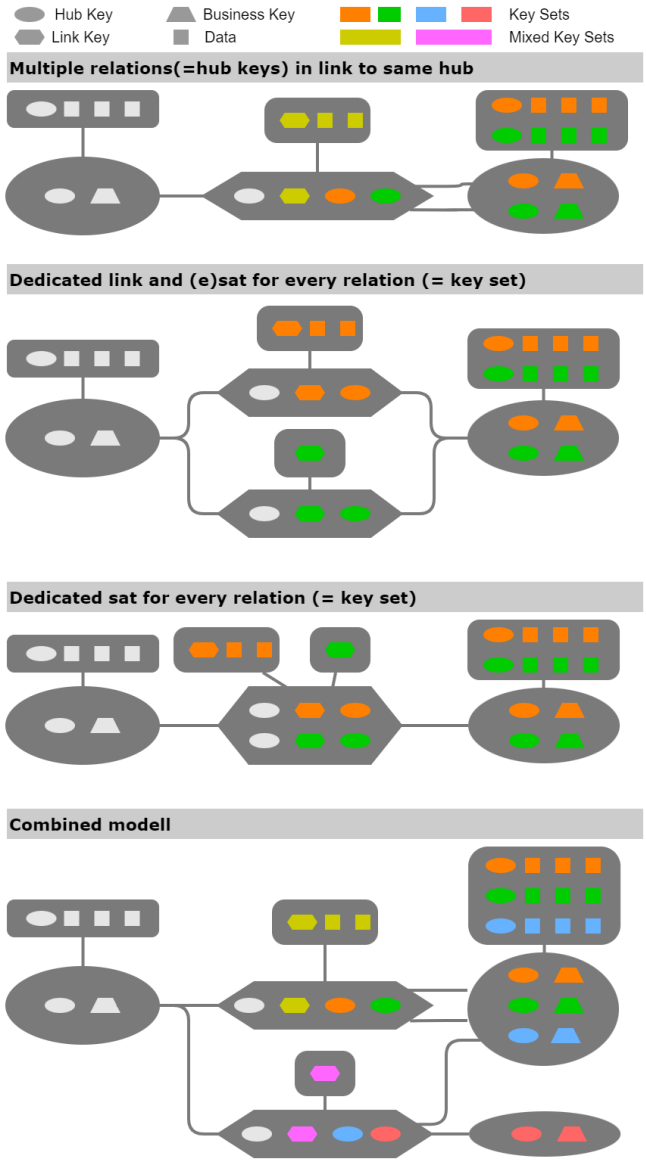
Standpunkt der Syntax ist die Modelltransformation



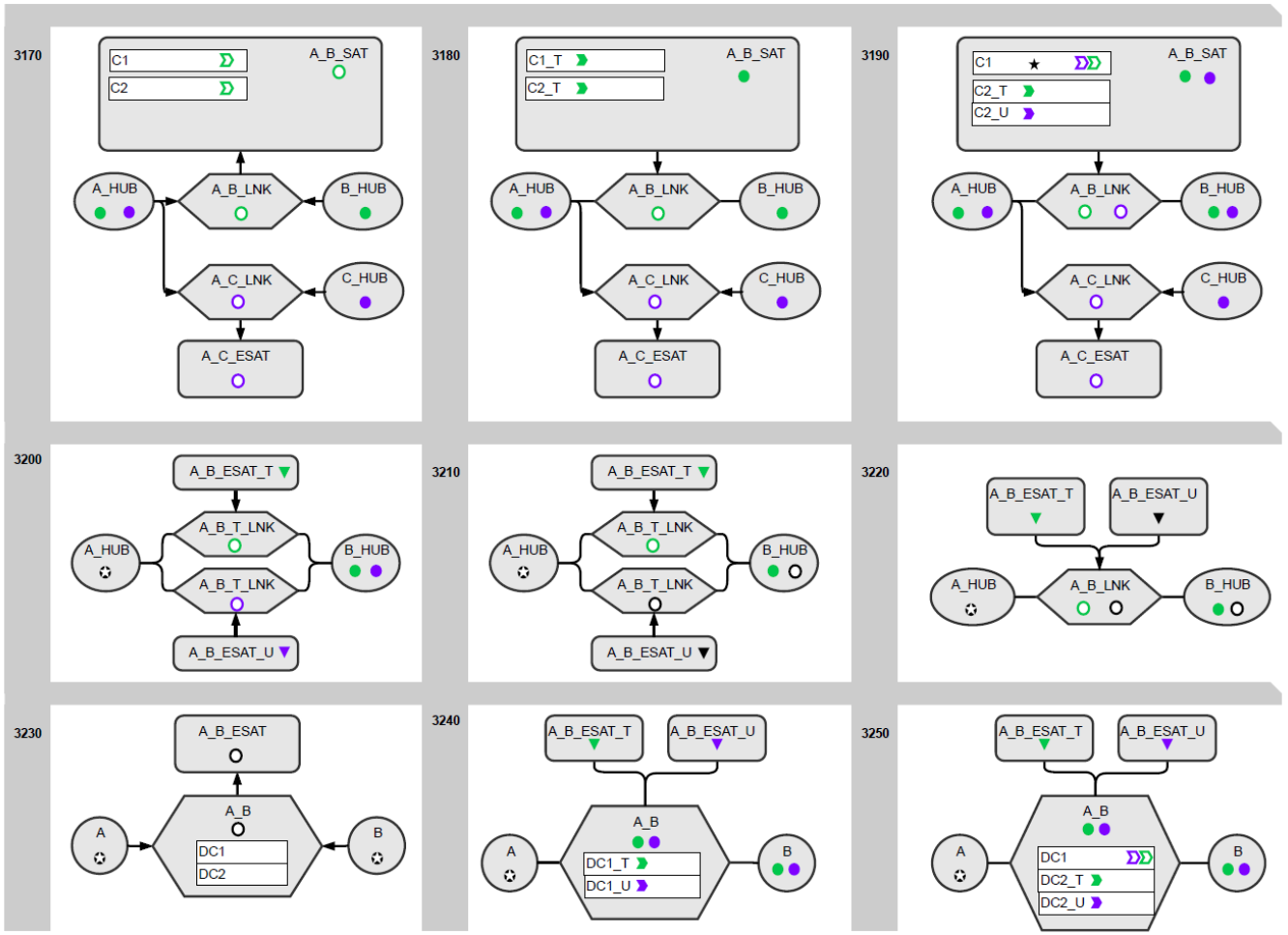
Methodische Selbstverständlichkeiten und Best Practices müssen nicht deklariert werden

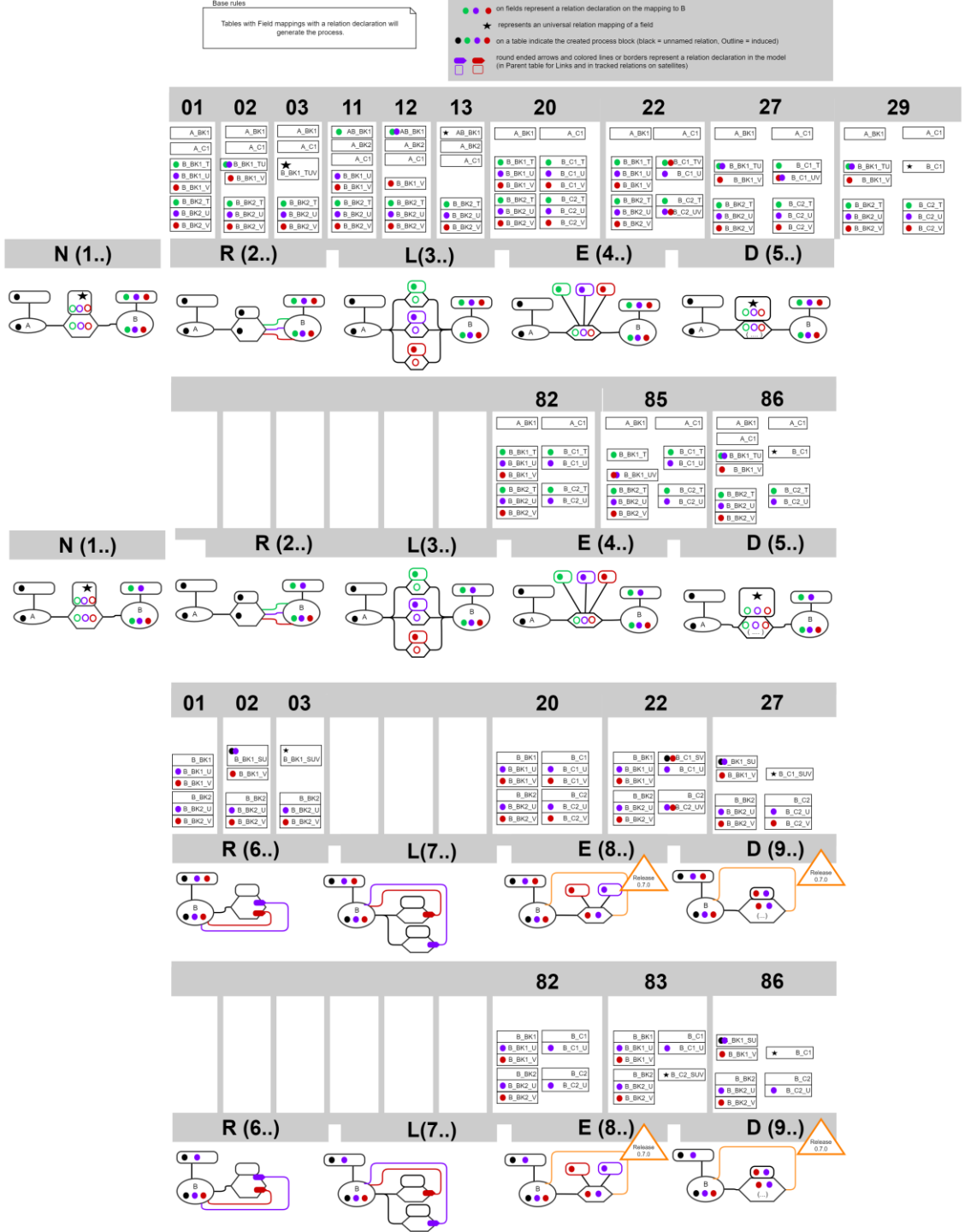
Vollständige Unterstützung der Data Vault Methode

- **„Building a Scalable Data Warehouse with Data Vault 2.0“ von Linstedt/Olschimke**
  ist die Grundlage
  (Data_Vault_method_coverage_and_syntax_examples)
- Kombinatorische Analysen und Ableitungen
  - Varianten von Data Vault Modellen
  - Varianten von Datenmappings
  - Begrenzt auf 1 Datenquellobjekt
- Ergänzt um Erfahrungen aus Projekten und der DV Community
  - Varianten bzgl. Modelldetails



Kombinatorische Analysen (Ausschnitt)



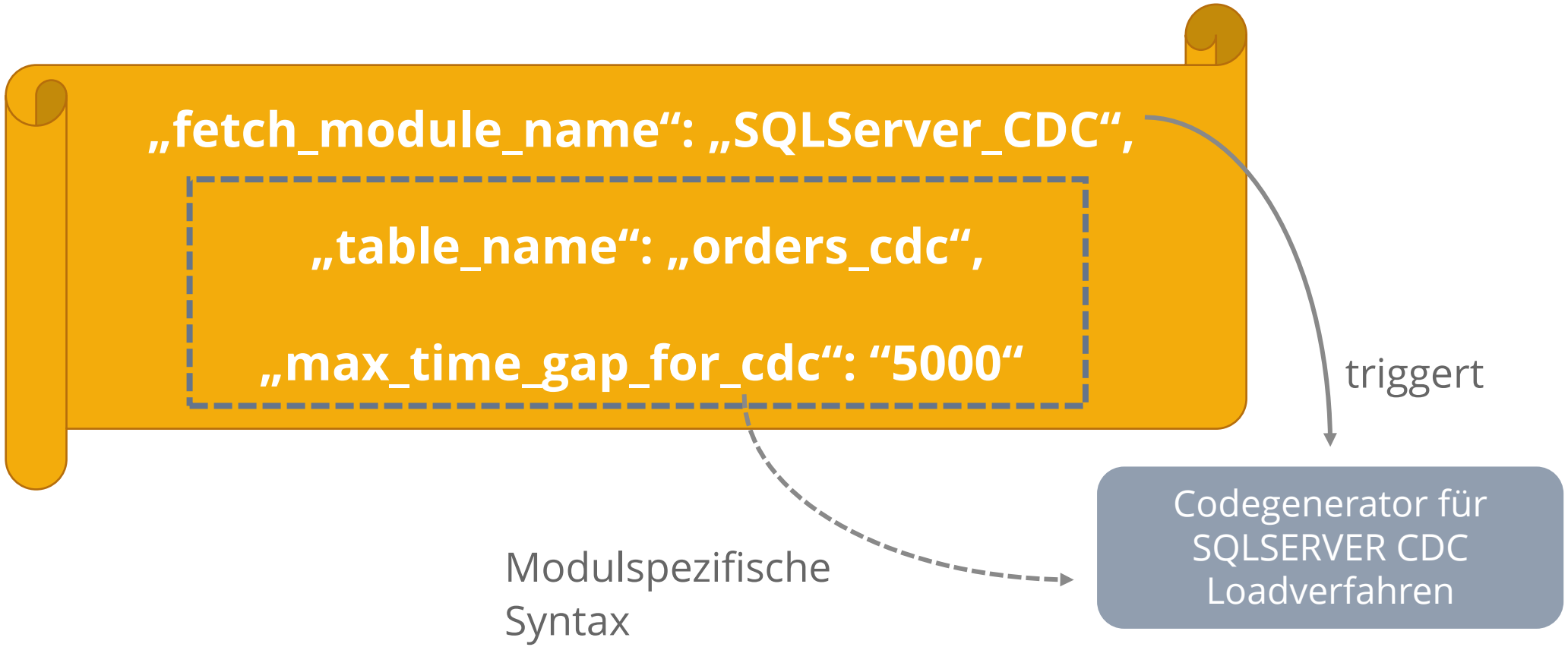


Vielfalt der Quellformate und Verhalten

cimt

Transport	Formate	Inkrement	Datenlöschung
File	DB Table	Container / Message	Expliziter Datensatz
DB Session	File with Table	Zeitlinie / Wertintervall	Schlüssel-abgleich
Stream	Dokument (Json/XML)	Change Data Capture	Regelwerk
„normale“API	Excel Eskapade	. . .	…
…	…		

Modulprinzip + offene Syntax = Flexibilität



Das optimale Verhältnis zwischen hartem Programmcode und generischen Komponenten hängt vom Projekt ab

Quellbezogene Parameter

Abrufmodul, hart codiert für diese Technologie und Inkrementmethode

```
{
  "dvpd_version":"0.6.0",
  "stage_properties":[{"stage_schema":"stage_rvlt"}],
  "pipeline_name":"hr_person",
  "record_source_name_expression":"company.org.persons",
  "data_extraction":{"fetch_module_name":"oracle_db_table_full", "source_system_name":"ora_company",
                     "source_schema":"org_structure", "source_table":"persons"},
  "data_vault_model":[
    {"schema_name":"rvlt_company",
     "tables":[
         {"table_name":"person_hub","table_stereotype":"hub"},
         {"table_name":"person_sat","table_stereotype":"sat", "satellite_parent_table":"person_hub"}
         ]
    }
  ]
  "fields":[
    {"field_name":"company",   "field_type":"varchar(5)",   "targets":[{"table_name":"person_hub"}]},
    {"field_name":"person_id","field_type":"varchar(10)",  "targets":[{"table_name":"person_hub"}]},
    {"field_name":"name",      "field_type":"varchar(250)","targets":[{"table_name":"person_sat"}]},
    {"field_name":"job",       "field_type":"varchar(250)","targets":[{"table_name":"person_sat"}]}
    ],
}
```

Parameter für das Abrufmodul

Dies sind auch Parameter für den Datenabruf (Parsing ist bei DB Anbindung nicht notwendig)

cimt

Warum JSON ?

weit verbreitetes Verfahren

Interpretierbar durch
Menschen und Maschinen

cimt

Das DVPD Entwicklungsprojekt

https://github.com/cimt-ag/
data_vault_pipelinedescription

Inhalte des Konzepts

Dokumentation

- Konzept
  - Anforderungsrahmen
  - Design
  - **Regelwerk**
  - Workflow Beispiele
- **Syntaxreferenz**
- Begleitende Erkenntnisse
  - Data Mapping Taxonomie
  - Deletion Detection Szenarien

DVPD
DVPI

Referenzimplementierung

- Compiler
  - Syntax- und Konsistenzprüfung
  - Ableitungsverfahren
  - Ergebnisbereitstellung
- Automatisierter Test
  - Testfälle
  - Referenzergebnisse
  - Vergleich
- Generatorbeispiele

Bisher
232

DDL
DBT(!)

https://github.com/cimt-ag/data_vault_pipelinedescription

Lizenz und Weiterverwendung

Dokumentation der Spezifikation

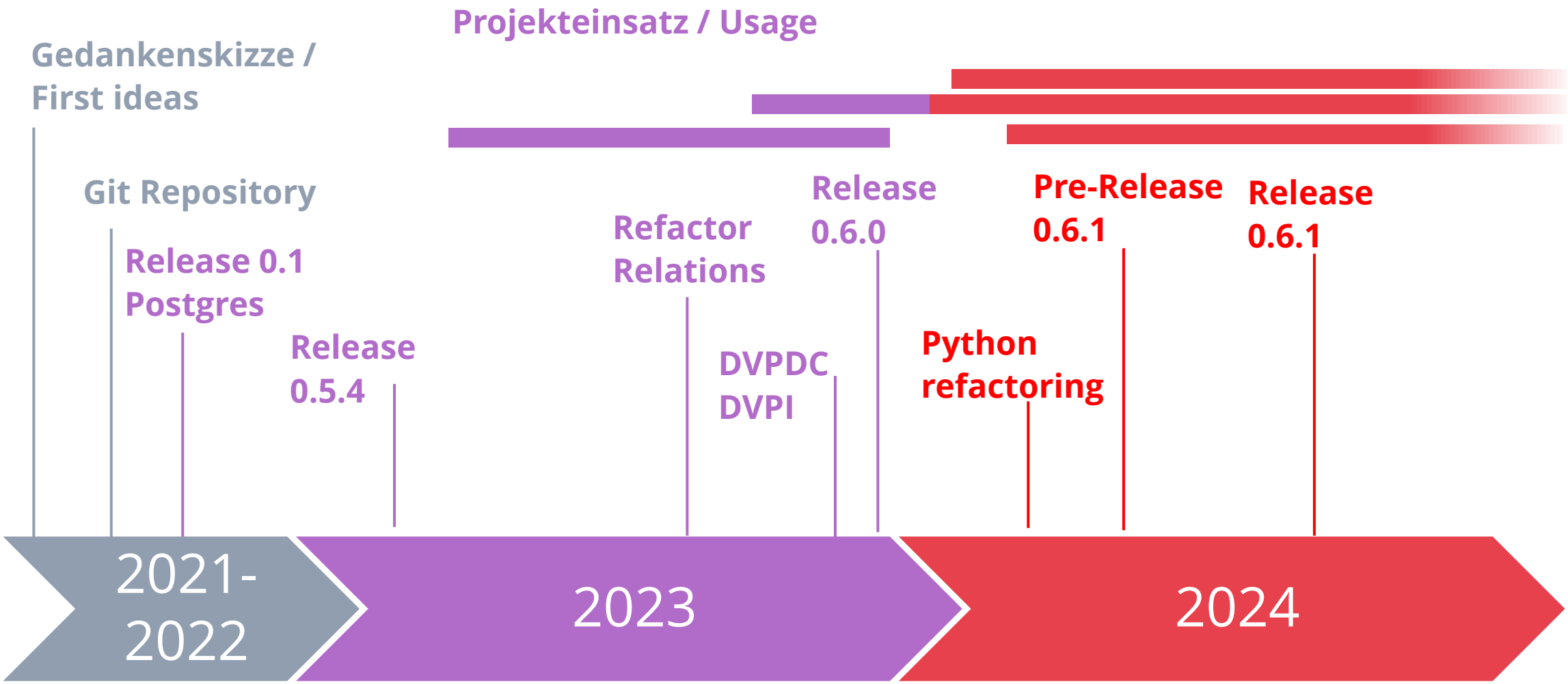
- Creative Commons BY-ND 4.0
  - Attribution
  - NoDerivatives

Referenzimplementierung

- Apache License, Version 2.0

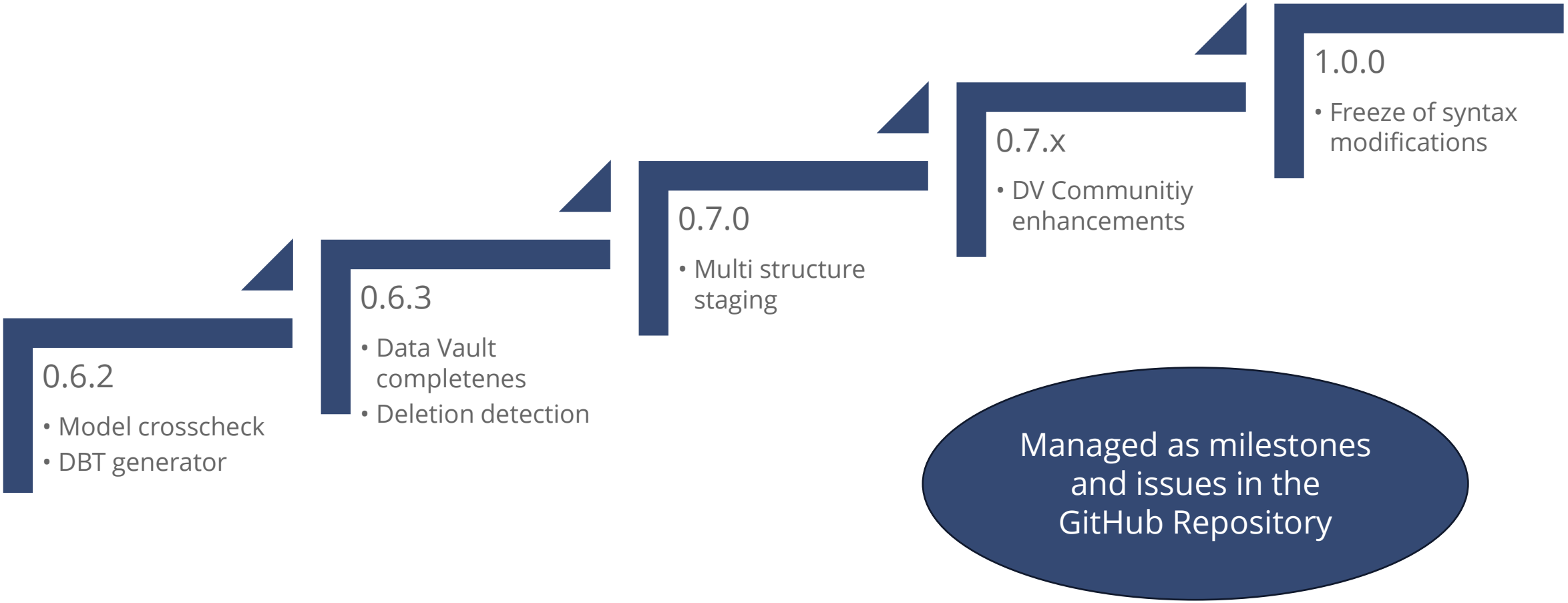
Weiterentwicklung des Konzeptes erfolgt unter der
redaktionellen Führung der cimt ag, Matthias Wegner.
Neue Versionen werden unter den gleichen Bedingungen veröffentlicht.

Historie/History



Roadmap

cimt

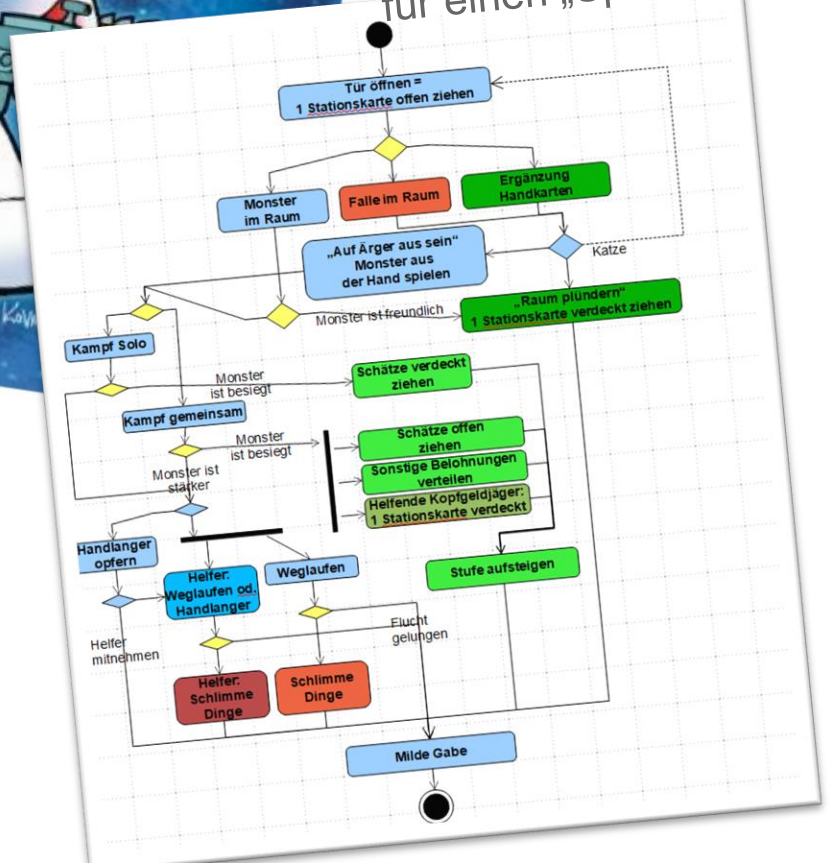


Wie kann man es nutzen ?

Erster „Vollkontakt“ mit DVPD

- Dokumentation ist ein harter Einstiegspfad
  - Komplexe Konzepte dominieren die Dokumentation, werden aber selten notwendig
- Bei der Anwendung wird’s plötzlich einfach und einleuchtend
- Coaching kann den Weg über die Lernkurve erleichtern

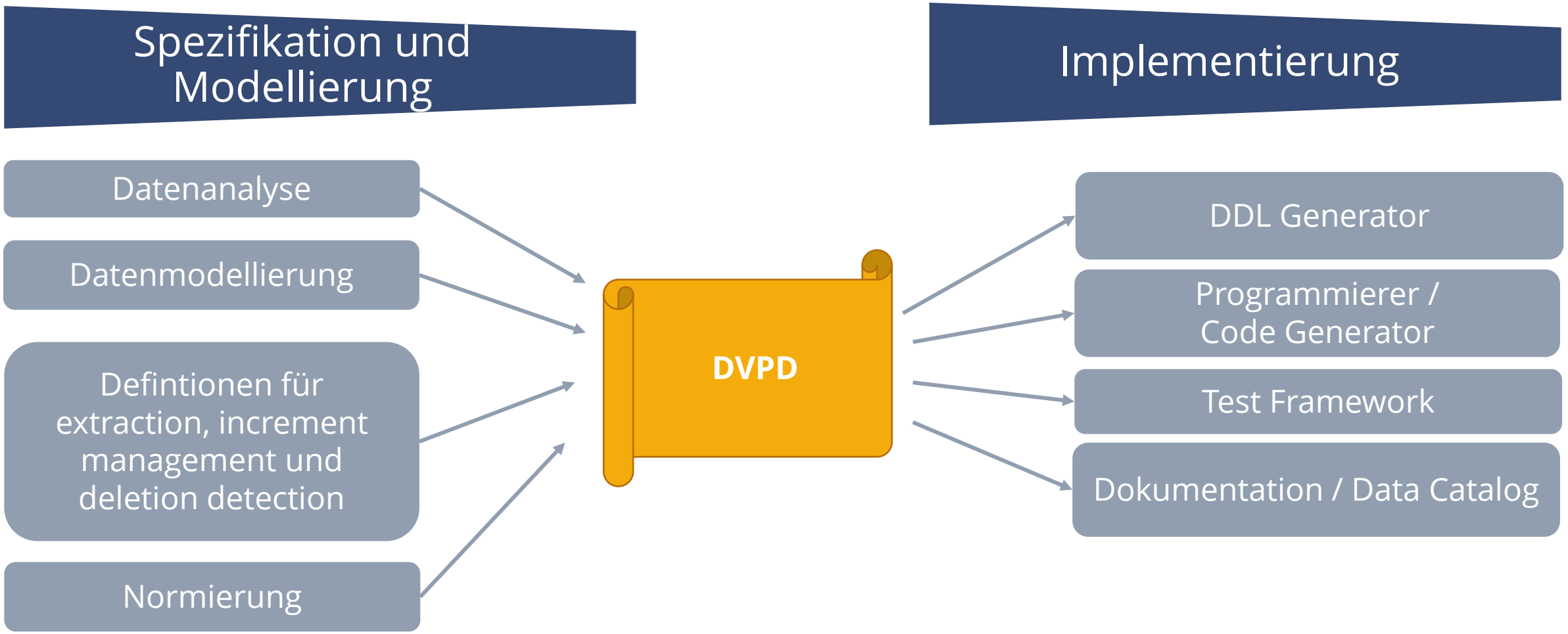
Ablaufplan für einen „Spielzug“



Mitwirken


Gestalten
Erweiterungen beisteuern
Feedback und Anforderungen in die Community geben
Runterladen + Lesen + Nutzen

Flexibilität der Toolchain ermöglichen



Freiheit für Erweiterungen

Prüfung plattformspezifischer Parameter und Limitierungen

Prüfung auf Namenskonventionen

Generator für Modelldiagramme

Erweiterung für Modellanpassung für DSGVO

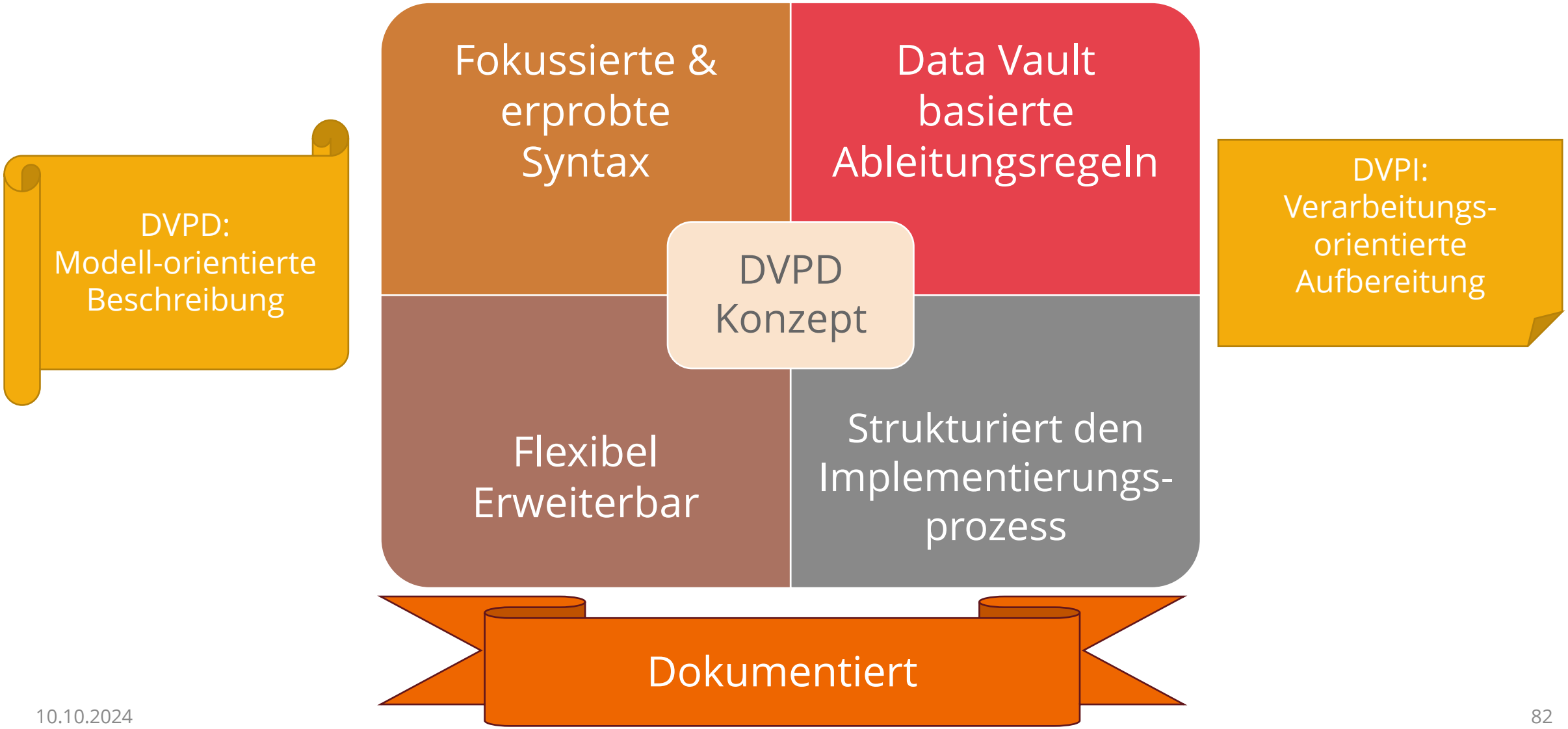
Generator für Tabellennamen und Data Vault Spaltennamen

Abgleich mit anderen Modellen

Generator für Testdaten

Verzeichnis aller DVPD/DVPI

DVPD als Leitlinie & Werkzeug



cimt

Q & A

Vielen Dank

Kontakt:
matthias.wegner@cimt-ag.de

LinkedIn: Matthias Wegner

https://github.com/cimt-ag/
data_vault_pipelinedescription/discussions