HAW HAMBURG

BACHELOR THESIS
Inga Buniatyan

Evaluation von ETL-Tools für kleine bis mittelgroße Datenintegrationsprojekte

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Inga Buniatyan

Evaluation von ETL-Tools für kleine bis mittelgroße Datenintegrationsprojekte

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Wirtschaftsinformatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Stefan Sarstedt
Zweitgutachter: Prof. Dr. Ulrike Steffens

Eingereicht am: 23.05.2025

Inga Buniatyan

Thema der Arbeit

Evaluation von ETL-Tools für kleine bis mittelgroße Datenintegrationsprojekte

Stichworte

ETL-/ELT-Toolbewertung, Projekttypologie, Scorecard-Modell, Datenintegration, kleine und mittlere Projekte, Werkzeugauswahl, Kriteriengewichtung

Kurzzusammenfassung

Diese Arbeit entwickelt ein kontextsensitives Bewertungssystem für ETL-/ELT-Tools, das gezielt auf die Anforderungen kleiner und mittlerer Datenintegrationsprojekte ausgerichtet ist. Ausgehend von einer empirisch fundierten Projekttypologie (Typen A–C) werden 14 bewertungsrelevante Kriterien abgeleitet, typenspezifisch gewichtet und über ein mehrstufiges Scoringmodell systematisch operationalisiert. Bewertet werden funktionale, technische und wartungsbezogene Merkmale – darunter Benutzerführung, Logging, API-Kompatibilität, Testbarkeit und Toollebensdauer.

Die empirische Fundierung des Bewertungsmodells basiert auf der Triangulation qualitativer Experteninterviews, einer Online-Umfrage sowie einer dokumentationsgestützten Sekundäranalyse. Die Analyse erfolgt exemplarisch an fünf marktüblichen Tools, deren Eignung typenspezifisch berechnet, validiert und visualisiert wird.

Die Ergebnisse zeigen, dass klassische Bewertungskategorien (z. B. GUI-Komfort oder Transformationsarchitektur) zunehmend durch Anforderungen an Erweiterbarkeit, Automatisierung und Governance ergänzt oder ersetzt werden. Das entwickelte Modell bietet eine praxisnahe, nachvollziehbare Entscheidungsgrundlage für datengetriebene Integrationsvorhaben mit begrenzten Ressourcen – als Alternative zu pauschalen Toolvergleichen.

Inga Buniatyan

Title of Thesis

Evaluation of ETL Tools for Small to Medium-Sized Data Integration Projects

Keywords

ETL/ELT tool evaluation, project typology, scorecard model, data integration, small and medium-sized projects, tool selection, criteria weighting

Abstract

This thesis presents a context-sensitive evaluation framework for ETL/ELT tools tailored to the specific requirements of small and medium-sized data integration projects. Based on an empirically grounded project typology (Types A–C), 14 evaluation criteria are derived, weighted according to project type, and operationalized through a multi-stage scoring model. The framework assesses functional, technical, and maintenance-related aspects, including user guidance, logging capabilities, API compatibility, testability, and tool sustainability.

The empirical foundation is built on a triangulation of qualitative expert interviews, an online survey, and a documentation-based secondary analysis. The model is applied to five commonly used tools, whose suitability is calculated in a project-type-specific manner, visualized, and validated.

The results indicate that classical evaluation categories—such as GUI complexity or transformation logic—are increasingly supplemented or replaced by requirements such as extensibility, automation readiness, and governance features. The developed model offers a transparent and practice-oriented decision-making aid for data-driven integration projects operating under constrained resources—beyond generic tool comparisons.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API (Application Programming Interface) Programmierschnittstelle, die Anwendungen standardisierten Zugriff auf externe Systeme oder Datenquellen ermöglicht – relevant für SaaS-Integration und Automatisierung.

CDC (Change Data Capture) Inkrementelles Verfahren zur Datenintegration, bei dem nur Datenänderungen (Inserts, Updates, Deletes) erkannt und übertragen werden.

ELT (Extract, Load, Transform) Datenintegrationsarchitektur, bei der die Transformation erst nach dem Laden in das Zielsystem erfolgt – typischerweise innerhalb eines Cloud-DWH.

ETL (Extract, Transform, Load) Klassische Architekturform der Datenintegration, bei der Daten aus Quellsystemen extrahiert, transformiert und anschließend in ein Zielsystem geladen werden.

ETL-/ELT-Tool Softwarelösung zur Durchführung von Datenextraktion, -transformation und -ladung; im weiteren Sinne auch Tools mit ELT-, CDC- oder hybriden Mechanismen.

Evidenzbasis Methodisch abgesicherte Informationsgrundlage für die Toolbewertung – besteht aus Literatur, Tooldokumentation, Experteninterviews und Nutzerumfrage.

Pipeline Modular aufgebaute Abfolge von Verarbeitungsschritten in einem ETL-/ELT-Prozess – umfasst z. B. Extraktion, Transformation und Ladelogik.

Projekttypologie (A–C) Heuristische Einteilung kleiner und mittlerer Integrationsprojekte in drei Nutzungstypen: A (GUI-orientiert), B (SaaS-basiert), C (Governance-orientiert).

Scorecard-Modell Strukturiertes Bewertungsmodell zur systematischen Bewertung von Tools anhand gewichteter Kriterien – differenziert nach Projekttypen.

Sekundärdatenanalyse Analyse öffentlich zugänglicher Informationsquellen wie Whitepapers, Repositories oder Tool-Dokumentationen – ohne eigenen Systemtest.

Toollebensdauer Langfristige Wartbarkeit, Update-Frequenz und Weiterentwicklung eines Tools – Indikator für Betriebssicherheit und Investitionsschutz.

Transformationstiefe Komplexitätsgrad der abbildbaren Datenmanipulation – von einfachem Mapping bis hin zu rekursiven, SQL-basierten Logiken.

Triangulation Kombination mehrerer unabhängiger Datenquellen zur Validierung von Bewertungsergebnissen und Reduktion von Verzerrungen.

Usability (Benutzerfreundlichkeit) Grad der intuitiven Bedienbarkeit eines Tools, insbesondere der grafischen Oberfläche – relevant für Projekte mit geringer IT-Tiefe.

Wartbarkeit Fähigkeit eines Tools, bestehende Datenprozesse effizient zu pflegen, zu erweitern und revisionssicher zu dokumentieren.

1 Einleitung

1.1 Relevanz des Themas

Datengetriebene Entscheidungen bilden eine zentrale Grundlage für die digitale Wettbewerbsfähigkeit moderner Organisationen. Mit der zunehmenden Verbreitung von Data-Warehouse-Lösungen, Cloud-Plattformen und Self-Service-Analytics steigt zugleich der Bedarf an leistungsfähigen Werkzeugen zur Datenintegration. Während klassische Integrationsansätze wie ETL (Extract – Transform – Load) oder ELT (Extract – Load – Transform) in umfangreichen Enterprise-Architekturen umfassend untersucht sind, bleibt die Frage nach ihrer Eignung in kleinen und mittleren Datenintegrationsprojekten weitgehend unbeantwortet. Dies betrifft insbesondere Kontexte wie Start-ups, Fachabteilungen oder IT-nahe Dienstleistungsunternehmen.

In diesen Projektumfeldern sind abweichende Rahmenbedingungen zu beobachten: begrenzte technische und personelle Ressourcen, heterogene Systemlandschaften, eingeschränkte Rollenverteilung, pragmatische Zielsetzungen und eine hohe Diversität in der Toolwahl. Die genutzten Integrationslösungen weichen oft deutlich von klassischen DWH- oder cloudnativen Architekturen ab. Ein methodisch fundierter Bewertungsrahmen, der diese Spezifika systematisch berücksichtigt, liegt bislang nicht vor.

1.2 Problemstellung

Die Bezeichnung „ETL-Tool“ wird im Markt uneinheitlich verwendet und umfasst eine Vielzahl von Werkzeugtypen – darunter klassische ETL-Suiten, API-Connector-Tools, Reverse-ETL-Lösungen, No-/Low-Code-Plattformen und automatisierte Integrationsdienste. In der Fachliteratur dominieren entweder generische Toolvergleiche oder Analysen großskaliger Plattformlösungen (vgl. u. a. Khan et al., Kimball, Szilveszter). Eine differenzierte Betrachtung für kleine und mittlere Integrationsprojekte erfolgt bislang kaum.

Diese Lücke zeigt sich insbesondere in praktischen Entscheidungssituationen. Relevante Fragen beziehen sich weniger auf den Funktionsumfang einzelner Tools als auf deren Passfähigkeit im spezifischen Projektkontext. So stellt sich etwa die Frage:

> *„Ist dieses Tool im konkreten Projektumfeld sinnvoll einsetzbar und langfristig wartbar?“*

Solche Überlegungen lassen sich mit konventionellen Funktionsvergleichen nicht angemessen beantworten. Erforderlich ist ein Modell, das den Projektkontext als zentrales Bewertungskriterium berücksichtigt und Tool-Eignung systematisch und empirisch fundiert herleitet.

1.3 Zielsetzung der Arbeit

Ziel der Arbeit ist die Entwicklung eines anwendungsorientierten und methodisch abgesicherten Bewertungsmodells für ETL-/ELT-Tools, das explizit auf kleine und mittlere Datenintegrationsprojekte zugeschnitten ist. Ausgangspunkt bildet eine Projekttypisierung (Typen A–C), basierend auf empirisch herleitbaren Projektmerkmalen wie Quellsystemvielfalt, Transformationsarchitektur, Persistenzanforderungen, Rollenverteilung und Automatisierungsgrad (vgl. Kapitel 2.1.5).

Auf dieser Grundlage werden 14 bewertungsrelevante Kriterien abgeleitet (vgl. Kapitel 2.3), welche funktionale, technische und organisatorische Aspekte abdecken. Dazu zählen etwa der Einrichtungsaufwand, die Benutzerfreundlichkeit, die Logging-Funktionalität, die Erweiterbarkeit, die API-Unterstützung oder die erwartbare Toollebensdauer. Die Bewertung erfolgt typenspezifisch gewichtet mittels eines Scoring-Modells, das sich an Grundprinzipien der Multi-Criteria Decision Analysis (MCDA) orientiert, jedoch eine bewusst pragmatische Rechenlogik verwendet.

Im Mittelpunkt steht nicht die Identifikation eines „besten Tools“, sondern die Entwicklung eines Vergleichsmodells, das Toolmerkmale in Relation zu konkreten Projektsituationen setzt. Die empirische Fundierung erfolgt durch Experteninterviews, eine Online-Umfrage sowie eine Sekundäranalyse öffentlich zugänglicher Toolinformationen (vgl. Kapitel 3.4).

Die Arbeit adressiert damit eine bislang wenig bearbeitete Lücke: die systematische, projekttypbasierte Bewertung datenintegrativer Werkzeuge in ressourcenschwachen Projektkontexten – zwischen Low-Code-Pragmatismus und Enterprise-Komplexität.

1.4 Forschungsfrage

> *Wie lässt sich die Eignung von ETL-/ELT-Tools für kleine und mittlere Datenintegrationsprojekte systematisch, typenspezifisch und empirisch fundiert bewerten – anhand einer funktional und kontextbezogen strukturierten Kriterienbasis?*

1.5 Aufbau der Arbeit

Die Arbeit gliedert sich in fünf Hauptkapitel. Kapitel 1 stellt Relevanz, Problemstellung und Zielsetzung dar. Kapitel 2 erläutert die theoretischen Grundlagen der Datenintegration, beschreibt typische Charakteristika kleiner und mittlerer Projekte, führt eine dreistufige Projekttypologie (Typen A–C) ein und leitet daraus eine gewichtete Scorecard mit 14 Bewertungskriterien ab.

Kapitel 3 beschreibt das methodische Vorgehen – von der Toolselektion über die Validierung der Kriterien (Interviews, Umfrage, Sekundärquellen) bis zur Umsetzung des Bewertungsmodells. Kapitel 4 präsentiert die empirischen Ergebnisse differenziert nach Projekttypen. Kapitel 5 interpretiert diese Ergebnisse, diskutiert methodische Einschränkungen, leitet praxisbezogene und forschungsrelevante Implikationen ab und schließt mit einem Fazit.

1.6 Abgrenzung und Einschränkungen

Die Arbeit bezieht sich ausschließlich auf kleine und mittlere Datenintegrationsprojekte, die durch begrenzte Ressourcen, einfache Rollenmodelle, moderate Architekturkomplexität und pragmatische Zielsetzungen gekennzeichnet sind. Großprojekte mit hochautomatisierten Pipelines, Echtzeitverarbeitung oder dedizierten DevOps-Strukturen bleiben explizit ausgeklammert.

Im Fokus steht die Bewertung dokumentierter und konfigurierbarer Toolmerkmale; Performanzmessungen, Laufzeitvergleiche oder Benchmarks sind nicht Gegenstand der Untersuchung. Ein direkter Tooltest wurde lediglich für Airbyte durchgeführt. Systematische Vergleichstests waren nicht Zielsetzung dieser Arbeit.

Das entwickelte Scoremodell basiert nicht auf mathematisch normierten Verfahren (wie AHP oder TOPSIS), sondern folgt einem anwendungsorientierten Bewertungsansatz für kontextspezifische Projektanforderungen. Die Ergebnisse sind nicht als Toolranking zu interpretieren, sondern als differenzierte Eignungsprofile auf Grundlage gewichtet validierter Kriterien.

2 Theoretischer Rahmen

2.1 Rahmenbedingungen und Projektkontexte

Einordnung und Relevanz. Datenintegration bildet den „Nervenknoten" moderner Informationssysteme, da nur konsolidierte und widerspruchsfreie Datenbestände verlässliche Analyse- und Steuerungsprozesse ermöglichen (Patzak-17, S. 21). Für Vorhaben kleiner bis mittlerer Größenordnung gelten jedoch engere Ressourcen-, Zeit- und Budgetgrenzen, sodass klassische Enterprise-Methoden nicht ohne Weiteres übertragbar sind. Rowe weist darauf hin, dass Projektmanagement in *small projects* gezielt „heruntergeskaliert" werden muss, um keinen Verwaltungs-Overhead zu erzeugen (Rowe-20, S. XIV). Krems schlägt hierfür das *Minimalschema* vor: eindeutige Zielvorgabe, reduzierte Rollen, schlanker Ressourceneinsatz (Krems-12, S. § 1.3).

Projektwürdigkeit und Größenklassen. Das Online-Verwaltungslexikon unterscheidet drei Projektklassen auf Basis des Gesamtaufwands – verstanden als Summe aus Personal-, Sach- und externen Dienstleistungskosten (Krems-12, S. „IT-Projektklassen"); siehe Tab. 2.1. Die Klasse *Klein* (10.000–25.000 €) markiert die Schwelle, ab der eine formalisierte Projektorganisation empfohlen wird. Lizenz- oder Betriebskosten einzelner Werkzeuge machen in dieser Größenordnung häufig weniger als 10 % des Gesamtaufwands aus.

Tabelle 2.1: Klassifizierung von IT-Projekten nach Gesamtaufwand, Personentagen und Laufzeit (Krems-12, S. "IT-Projektklassen")

Klasse	**Gesamtaufwand***	**Personentage (PT)**	**Laufzeit**
Groß	> 300 000 €	> 300 PT	> 12 Mon.
Mittel	25 000–300 000 €	60–300 PT	3–12 Mon.
Klein	10 000–25 000 €	20–60 PT	1–3 Mon.

Gesamtaufwand = Personal-, Sach- und externe Unterstützungskosten. Lizenzgebühren einzelner ETL-Werkzeuge fallen hierunter, machen in typischen Kleinprojekten jedoch selten mehr als 5–10 % aus.

Untersuchungsrahmen dieser Arbeit. Vor dem Hintergrund dieser Einordnung bezieht sich die vorliegende Arbeit auf Integrationsprojekte, die

- einen **Gesamtaufwand** zwischen 10.000 € und 300.000 € aufweisen,
- einen **Personenaufwand** von 20–300 PT (ca. 2–10 Mitarbeitende) erfordern,
- eine **Laufzeit** von höchstens 12 Monaten haben,
- maximal **acht operative Quellsysteme** integrieren (Tawil-24, S. 5).

Diese Spannweite deckt sowohl kleinvolumige Abteilungsinitiativen als auch mittlere Fachbereichsprojekte ab. In späteren Abschnitten werden daraus typische Strukturmuster (z. B. Rollenverteilung, Quellarchitekturen, Toolpräferenzen) abgeleitet, die schließlich zu einer dreiteiligen Typisierung von Nutzungskontexten führen.

Struktur des Kapitels. Die folgenden Abschnitte konkretisieren zunächst die Begriffs- und Zieldefinition von Datenintegration (2.1.1), analysieren typische Projektmerkmale und Herausforderungen (2.1.2–2.1.4) und leiten daraus eine dreistufige Typologie (2.1.5) ab. Diese dient als methodisches Fundament für die Architekturbetrachtung (Kap. 2.2), die Bewertungskriterien (Kap. 2.3) sowie das Scoringmodell (Kap. 3).

2.1.1 Begriffliche Grundlagen und Zielsetzungen der Datenintegration

Datenintegration bezeichnet den *kontinuierlichen, systemübergreifenden Prozess* der Zusammenführung, Harmonisierung und Bereitstellung von Daten aus heterogenen Quellsystemen mit dem Ziel, eine einheitliche, konsistente und auswertbare Gesamtsicht zu schaffen (Walha-24, S. 26688).

Sie unterscheidet sich damit von der *Datenmigration* (einmaliger Transfer) sowie von der *Datenkonsolidierung* (rein strukturelle Vereinheitlichung) durch die fortlaufende Synchronisation zwischen Quell- und Zielsystem (Krems-12, S. § 2).

Die Zielsetzungen von Datenintegrationsvorhaben lassen sich nach Inmon (Inmon-05, S. 23) drei Ebenen zuordnen:

a) **Technisch** – Standardisierung von Formaten, Strukturen und Schnittstellen zur Ermöglichung automatisierter Weiterverarbeitung.

b) **Semantisch** – Einheitliche Interpretation identischer Entitäten trotz divergierender Fachsystemsemantiken (Doan-12, S. 115).

c) **Organisatorisch** – Aufbau redundanzfreier, revisionssicherer Datenbestände mit klar definierten Zuständigkeiten, etwa durch Rollen wie Data Owner oder Data Steward (Appelfeller-23, S. 87).

Gerade kleinere und mittlere Integrationsprojekte stehen vor der Herausforderung, diese Zielsetzungen unter begrenzten finanziellen, personellen und zeitlichen Ressourcen zu realisieren.

Kozłowski und Matejun zeigen in einer Untersuchung von 563 Projekten in kleinen und mittleren Unternehmen, dass diese durch eine geringe Formalisierungsdichte, zentralisierte Entscheidungsstrukturen und ausgeprägte Ressourcenknappheit geprägt sind (Kozlowski-16, S. 142 f.).

Tawil et al. analysieren 85 britische Fallstudien und identifizieren fehlende IT- Investitionsspielräume sowie Defizite in datenbezogenen Kompetenzen als zentrale Hemmnisse datengetriebener Projektvorhaben (Tawil-24, S. 5–7).

Diese Rahmenbedingungen erfordern Werkzeuge, die einerseits methodisch anschlussfähig, andererseits aber mit geringem Einführungs- und Lizenzaufwand nutzbar sind. In Abschnitt 2.1.5 werden daraus drei typische Nutzungskontexte abgeleitet, die unterschiedliche Anforderungen an Konnektivität, Automatisierung und Rollenverteilung aufweisen.

2.1.2 Charakteristika und typische Herausforderungen kleiner und mittelgroßer Datenintegrationsprojekte

Die in Abschnitt 2.1 definierten Projektgrenzen (vgl. Tabelle 2.1) prägen wesentlich die technischen und organisatorischen Rahmenbedingungen von Datenintegrationsvorhaben. Empirische Studien und praxisorientierte Leitfäden identifizieren dabei folgende charakteristische Merkmalsbündel (Kozlowski-16, S. 7; Tawil-24, S. 12; Rowe-20, S. 27):

a) **Ressourcenlimitierung.** Die Projekte bewegen sich typischerweise im Kostenspektrum von 10.000–300.000 €, wobei dieser Gesamtaufwand Personal-, Sach- und externe Dienstleistungskosten umfasst. Die Untergrenze entspricht der Projektklasse „Klein“, während die Obergrenze Projekte der Klasse „Mittel“ einschließt. Lizenz- und Betriebskosten für ETL-Werkzeuge machen in diesem Kontext in der Regel nur einen einstelligen Prozentanteil aus (Rowe-20, S. 27).

b) **Heterogene, aber begrenzte Systemlandschaft.** In der Regel werden zwei bis acht operative Quellsysteme eingebunden, überwiegend über einfache Exporte (CSV/Excel) oder unversionierte REST-Schnittstellen (Tawil-24, S. 5).

c) **Geringe Formalisierung und hohe Anpassungsfähigkeit.** Die Projektorganisation erfolgt häufig mit reduzierten Rollenmodellen (z. B. Projektleitung und fachlicher Vertreter) und ohne standardisierte Projektmanagementmethoden. Rowe empfiehlt für solche Kontexte verkürzte Projektaufträge und skalierbare Templates (Rowe-20, S. Kap. 1.2).

d) **Zeitkritik.** Projektlaufzeiten von ein bis drei Monaten erzeugen hohen Einführungsdruck. Klassische Wasserfallmodelle werden daher häufig durch iterative Sprintansätze ersetzt (Rowe-20, S. 79).

e) **Kompetenzengpässe und ausgeprägtes Kostenbewusstsein.** Fehlende Datenkompetenzen und beschränkte IT-Budgets gelten als zentrale Hürden. Kleine Unternehmen setzen daher bevorzugt auf kostengünstige, einfach konfigurierbare Tools, häufig im Open-Source-Bereich (Kozlowski-16, S. 145).

Herausforderungen. Die Kombination aus knappen Ressourcen, heterogener Quellstruktur und zeitlicher Begrenzung führt regelmäßig zu drei typischen Problemfeldern, die sich aus der Literatur ableiten lassen:

- *Hoher Aufwand bei der Konnektorentwicklung*, bedingt durch proprietäre Formate und fehlende Standardschnittstellen (Rowe-20, S. 80).
- *Mangelnde Qualitätssicherung unter Zeitdruck*, da automatisierte Tests häufig fehlen und Validierungen manuell erfolgen müssen (Hu-24, S. 6).
- *Defizite bei Governance und Compliance*, insbesondere aufgrund unklarer Zuständigkeiten, fehlender Rollenmodelle sowie unzureichender Umsetzung datenschutzrechtlicher Anforderungen wie der DSGVO (Kozlowski-16, S. 147).

Die konkrete Ausprägung dieser Herausforderungen hängt stark vom Nutzungskontext ab – also davon, wie viele Quellen, welche Rollen und welche Tooltypen im Projekt konkret eingesetzt werden. Diese Kontexte werden in Abschnitt 2.1.5 typisiert.

2.1.3 Kontextbedingungen und Handlungsmotive kleiner und mittelgroßer Integrationsvorhaben

Die in Abschnitt 2.1 skizzierten Budget-, Personal- und Laufzeitgrenzen bestimmen nicht nur den organisatorischen Rahmen, sondern auch die fachlichen Beweggründe, die zur Initiierung von Datenintegrationsvorhaben führen. Literatur- und Fallstudienanalysen offenbaren dabei ein konsistentes Treibermuster.Diese Treiber spiegeln sich in wiederkehrenden Kombinationen aus Systemlandschaft, Personalstruktur und Toolanforderung wider, die in Abschnitt 2.1.5 typisiert werden:

a) Datennutzungsdruck und externe Erwartungen. Viele Unternehmen sehen sich wachsendem Druck durch Kunden, Partner und Marktumfeld ausgesetzt, datenbasierte Services bereitzustellen. Tawil et al. identifizieren in 85 Fallstudien fehlende Datenanbindung und geringe Analysefähigkeit als strategisches Defizit; die tatsächliche Nutzung analytischer Verfahren liegt teils unter 1 % (Tawil-24, S. 5–7). Die daraus resultierende digitale Aufholbewegung erzeugt unmittelbaren Handlungsbedarf.

b) Fragmentierte Quellsysteme. Typisch sind zwei bis acht operative Quellsysteme, angebunden über einfache Exporte (CSV, Excel) oder unversionierte REST-Schnittstellen. Die Systeme sind strukturell und semantisch uneinheitlich, automatisierte Schema-Evolution fehlt häufig (Szilveszter-24, S. 4).

c) Geringe Formalisierung, zentrale Steuerung. Insbesondere in KMU erfolgen Integrationsinitiativen oft ohne klassische Projektstrukturen. Zentralisierte Entscheidungswege, kombinierte Rollen (z. B. Projektleitung = Fachanwender) und fehlende Governance-Funktionen sind typisch (Kozlowski-16, S. 145).

d) Kompetenzdefizite und Ressourcenkonflikte. Der Mangel an technischen Datenkompetenzen erweist sich als zentrales Hindernis. Häufig übernehmen fachliche Generalisten Integrationsaufgaben nebenbei, ohne dedizierte DevOps- oder Monitoring-Strukturen (Hu-24, S. 7). GUI-basierte, konfigurierbare Tools mit niedriger Einstiegshürde – oft Open Source – werden bevorzugt eingesetzt (Popovic-24, S. 6; Szilveszter-24, S. 5).

e) Kurzfristiger Nutzenfokus. Kleine Integrationsprojekte sind primär auf schnelle Ergebnisse ausgerichtet. Laut Kozłowski und Matejun stehen Effizienzgewinne und Time-to-Value klar vor langfristiger Skalierbarkeit oder Governance (Kozlowski-16, S. 146 f.).

Implikationen für die Toolauswahl

Aus den genannten Kontextfaktoren lassen sich drei übergreifende Anforderungen an ETL-/Integrationswerkzeuge ableiten:

a) *Niedrige Einstiegshürden*, insbesondere in Bezug auf Lizenzkosten, Infrastrukturbedarf und Usability.

b) *Breite Konnektivität bei einfacher Konfiguration*, um heterogene Quellen effizient anzubinden.

c) *Unterstützung inkrementeller Projektvorgehen*, etwa durch Vorlagen, Schritt-für-Schritt-Mappings oder integrierte Testfunktionen.

Diese Kriterien werden in Kapitel 3 als Bewertungsdimensionen der Scorecard operationalisiert. Die nachfolgenden Abschnitte untersuchen typische Risikofelder (Kap. 2.1.4) und leiten eine projekttypische Nutzungstypologie (Kap. 2.1.5) als Grundlage für Architektur- und Werkzeugbewertung ab.

2.1.4 Typische Risikodimensionen in kleinen und mittelgroßen Datenintegrationsprojekten

Die in Abschnitt 2.1.3 skizzierten Handlungstreiber erzeugen eine Reihe typischer Risikopotenziale, die bei kleinen bis mittelgroßen Datenintegrationsvorhaben regelmäßig auftreten. Je nach konkretem Projektkontext (vgl. Typen A–C in Abschnitt 2.1.5) zeigen sich dabei unterschiedliche Risikoschwerpunkte – etwa Datenqualitätsprobleme in GUI-nahen Szenarien (Typ A) oder Governance- und Integrationsrisiken bei API- und Cloud-basierten Projekten (Typ B/C).

a) Datenqualitäts- und Konsistenzrisiken. Unvollständige, fehlerhafte oder inkonsistent gepflegte Quellbestände sind in dezentral organisierten Systemlandschaften weit verbreitet. Ursachen sind unter anderem manuelle Eingabeverfahren, uneinheitliche Felddefinitionen und fehlende semantische Standards (Walha-24, S. 26692). Ohne integrierte Validierung oder regelbasierte Qualitätsprüfung erhöht sich das Risiko redundanter, unbrauchbarer oder widersprüchlicher Daten deutlich (Szilveszter-24, S. 4).

b) Zeit- und Budgetdruck. Kleine Projekte sind häufig auf wenige Wochen Laufzeit limitiert (vgl. Abschnitt 2.1). Diese zeitliche Restriktion führt zu einem Zielkonflikt zwischen Implementierungsgeschwindigkeit und Testtiefe. Rowe warnt vor einer Unterschätzung der Ressourcen für Konnektorentwicklung und Datenbereinigung (Rowe-20, S. 79); Testautomatisierung fehlt in der Regel vollständig (Hu-24, S. 6 f.).

c) Technologische Integrationsrisiken. Veraltete, proprietäre oder inkompatible Quellsysteme erschweren die Anbindung über standardisierte Schnittstellen. Häufig sind keine APIs vorhanden oder nicht dokumentiert; selbst einfache REST-Endpunkte können instabil oder schema-variabel sein (Szilveszter-24, S. 6). Die fehlende Versionierung und nicht nachvollziehbare Schemaänderungen zählen zu den häufigsten Ursachen für Pipeline-Ausfälle (Walha-24, S. 26691).

d) Governance- und Datenschutzprobleme. In kleinen Projekten fehlen oft klar definierte Rollen für Datenverantwortung (z. B. Data Owner oder Steward). Infolgedessen ist unklar, wer über Datenzugriffe, Speicherfristen oder Freigaben entscheidet. Besonders problematisch ist dies mit Blick auf Datenschutzanforderungen wie DSGVO, die Auditierbarkeit, Löschpflichten und Zweckbindung verlangen (Kozlowski-16, S. 147).

e) Kompetenzrisiken. Fehlende Erfahrung im Umgang mit Datenmodellen, ETL-Prozessen oder Transformationstools führt zu operativen Fehlern in Konfiguration, Mapping oder Validierung. Tawil et al. und Hu et al. beschreiben sogenannte „Citizen Integrators“, die in KMU zwar motiviert, aber häufig unzureichend technisch geschult sind (Tawil-24, S. 6; Hu-24, S. 7).

Diese Risikofelder fließen in Kapitel 3 als Kriteriendimensionen in die Scorecard ein (z. B. Validierungsmechanismen, Logging, Testfunktionalitäten, Rollenmodelle).

2.1.5 Typologie typischer Nutzungskontexte kleiner und mittelgroßer Integrationsvorhaben

Die Abschnitte 2.1.2 bis 2.1.4 haben verdeutlicht, dass kleine und mittelgroße Datenintegrationsprojekte – trotz vergleichbarer Projektgrößen – in sehr unterschiedlichen technischen und organisatorischen Konstellationen realisiert werden. Insbesondere bei der Anzahl und Art der Quellsysteme, der Rollenverteilung im Team, dem Automatisierungsgrad sowie der bevorzugten Toolklasse zeigen sich deutliche Unterschiede. Ausgehend von den zuvor analysierten Merkmalsbündeln wird im Folgenden eine kontextbasierte Typologie entwickelt, die drei wiederkehrende Nutzungskontexte identifiziert und als Strukturtypen A bis C bezeichnet.

Diese Typen abstrahieren typische Realitätsmuster kleiner und mittlerer Projekte und dienen im weiteren Verlauf der Arbeit als Orientierungsraster für Architekturbetrachtungen (Kap. 2.2), Kriteriendefinitionen (Kap. 2.3) und Scorecard-Auswertungen (Kap. 3).

Tabelle 2.2: Strukturtypen A–C als Nutzungsmuster kleiner und mittlerer Projekte

	Typ A: GUI-Ad-hoc	**Typ B: SaaS-Batch**	**Typ C: Governance-API**
Quellsysteme	1–2 (CSV, Excel)	3–5 (SaaS, REST)	6–8 (ERP, SQL, CRM)
Rollen	1 FTE (Citizen User)	2 FTE (Fach + Tech)	3 FTE (inkl. Data Engineer)
Toolpräferenz	GUI-basiert, No-/Low-Code	Cloud-ELT mit Konnektorkatalog	Skript-/API-basiert, CI/CD-fähig
Automatisierung	Manuell / Cronjobs	Tägliche Batchläufe	Orchestriert, GitOps
Hauptfokus	Nutzerfreundlichkeit, Einstieg	Effizienz, Konnektivität	Governance, Skalierbarkeit
Hauptrisiko	Datenqualität	Budget / Zeit	Compliance / Wartbarkeit

Typ A – GUI-orientierter Einzelbereichskontext Typ A beschreibt Vorhaben mit minimaler technischer Infrastruktur und wenigen Quellformaten (z. B. CSV, Excel). Sie werden häufig von Citizen Usern durchgeführt, die eine grafische Benutzeroberfläche benötigen und keine tiefere Integration realisieren. Ziel ist ein schneller Einstieg bei minimalem Aufwand. Risiken bestehen primär in mangelnder Datenvalidierung und unstrukturierter Ausführung.

Typ B – Cloud-basierter Integrationskontext Dieser Typ umfasst Projekte mit mehreren SaaS-Quellen (z. B. CRM, E-Commerce), die regelmäßig synchronisiert werden. Zum Einsatz kommen meist cloudbasierte Tools mit Konnektorkatalogen und zeitgesteuerter Verarbeitung. Fachliche und technische Rollen arbeiten gemeinsam. Typische Herausforderungen sind Konnektoranpassungen und Zeitabhängigkeiten bei begrenztem Budget.

Typ C – Governance-fokussierter Prozesskontext Typ C steht für anspruchsvollere Integrationsszenarien mit strukturierten Datenquellen, API-Schnittstellen und DevOps-Einbindung. Hier dominieren Governance-Anforderungen wie Rollentrennung, Testautomatisierung und Auditierbarkeit. Skriptbasierte Werkzeuge und CI/CD-Prozesse sind charakteristisch. Die Hauptrisiken betreffen Datenschutz, Zuständigkeiten und Wartbarkeit.

Funktion der Typologie Die Typologie ersetzt keine klassische Projektklassifikation (z. B. nach Kosten oder Laufzeit), sondern ergänzt diese um eine inhaltliche Strukturperspektive. Sie ermöglicht eine differenzierte Betrachtung typischer Rahmenbedingungen, Zielsetzungen und Risiken und bildet die Grundlage für eine kontextspezifische Bewertung in den folgenden Kapiteln.

2.2 Formen und Architekturen der Datenintegration

Die in Abschnitt 2.1.5 dargestellten Nutzungstypen A–C unterscheiden sich nicht nur hinsichtlich der Quellstruktur, des Automatisierungsgrads und der Rollenprofile, sondern implizieren auch unterschiedliche Anforderungen an die zugrunde liegende Integrationsarchitektur. So erfordern GUI-orientierte Ad-hoc-Kontexte (Typ A) andere Datenflüsse als orchestrierte CI/CD-Umgebungen mit Governance-Schwerpunkt (Typ C).

Kapitel 2.2 analysiert die zentralen Architekturformen der Datenintegration im Hinblick auf ihre Funktionsweise, Voraussetzungen und Eignung für kleine bis mittelgroße Projekte.

2.2.1 Klassische ETL-Architektur

Trotz moderner cloudbasierter Alternativen wie ELT gilt das klassische ETL-Verfahren (Extract – Transform – Load) in vielen Anwendungskontexten weiterhin als bewährter Standard für die systematische Datenintegration. Es ermöglicht die kontrollierte Überführung heterogener Quelldaten in ein einheitliches Zielmodell, typischerweise innerhalb eines zentralen Data Warehouses oder Data Marts (Kimball-13, S. 15 f.).

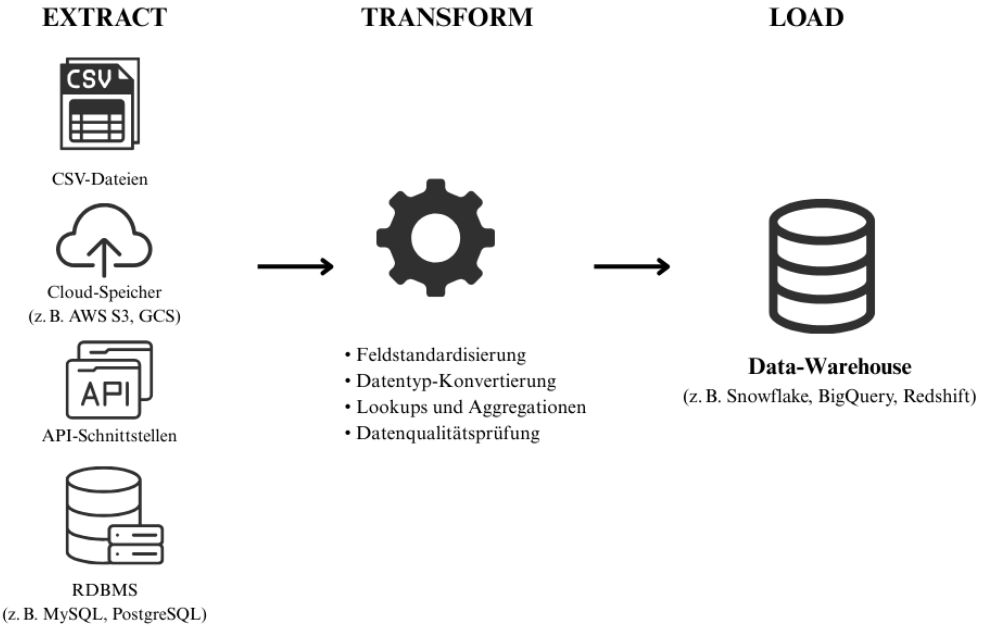

Abbildung 2.1: Architektur eines klassischen ETL-Prozesses. Eigene Darstellung in Anlehnung an (Singhal-22, S. 2).

Die Extraktion erfolgt periodisch aus mehreren Quellsystemen (z. B. relationale Datenbanken, CSV-/Excel-Dateien, Web-Exporte), die Transformation innerhalb einer dedizierten Verarbeitungslogik (regelbasierte Umformung, Join, Lookup, Cleansing), bevor die Daten in eine zielspezifische Struktur (z. B. Stern- oder Schneeflockenschema) geladen werden. Die gesamte Transformationslogik ist in der zentralen Integrationsschicht angesiedelt, was eine klare Trennung zwischen Rohdatenhaltung und Berichtssystem ermöglicht (Inmon-05, S. 23). Werkzeuge wie Talend Open Studio, Pentaho oder Informatica unterstützen solche Pipelines durch grafische Modellierung, Logging und Zeitsteuerung (Walha-24, S. 26688).

Typische Merkmale des klassischen ETL-Modells sind:

- **Zentrale Steuerung:** Alle Prozessphasen werden außerhalb des Zielsystems ausgeführt, meist auf dedizierten Integrationsservern.
- **Batch-Verarbeitung:** Die Verarbeitung erfolgt in festen Zeitfenstern (z. B. täglich oder stündlich).
- **Explizite Transformationslogik:** Die gesamte semantische Harmonisierung liegt in der Verantwortung der ETL-Schicht.

Zu den Vorteilen zählen die hohe Kontrolltiefe über Datenflüsse, auditierbare Transformationen und eine ausgereifte Toolunterstützung. Nachteile ergeben sich insbesondere im Wartungsaufwand, bei der Konnektorpflege sowie durch die eingeschränkte Eignung für Echtzeit- oder inkrementelle Szenarien (Walha-24, S. 26689).

Einordnung in Typ A–C Das klassische ETL-Paradigma ist besonders für **Typ-C-Prozesse** geeignet, bei denen strukturierte Quellsysteme (z. B. SQL, ERP) angebunden und kontrollierbare Transformationslogiken gefordert sind. In **Typ-B-Kontexten** kann ETL ebenfalls sinnvoll eingesetzt werden, sofern cloudbasierte Quellen regelmäßig über Batch-Prozesse integriert werden. Für **Typ-A-Projekte** ist ETL hingegen meist ungeeignet, da dort GUI-basierte No-Code-Werkzeuge ohne dedizierte Serverstrukturen bevorzugt werden (vgl. Abschnitt 2.1.5).

2.2.2 ELT als alternative Architekturform

Das ELT-Verfahren (Extract – Load – Transform) stellt eine moderne Architekturform dar, bei der Daten zunächst vollständig in das Zielsystem geladen und dort transformiert werden. Anders als beim klassischen ETL liegt der Schwerpunkt nicht auf einer vorgelagerten Datenharmonisierung, sondern auf der Nutzung der Rechenleistung des Zielsystems – etwa bei Cloud-DWHs wie Snowflake, Google BigQuery oder Amazon Redshift (Schulz-24, S. 46; Seenivasan-22, S. 3).

Die Transformation erfolgt direkt im Zielsystem – zumeist über SQL-basierte Transformationen, dbt-Modelle oder integrierte Cloud-Funktionen. Durch diesen Paradigmenwechsel entfallen separate Integrationsserver, was insbesondere bei Cloud-nativen Anwendungen die Implementierung vereinfacht (Popovic-24, S. 5).

Zu den Vorteilen zählen die Skalierbarkeit und Rechenleistung moderner DWHs sowie geringere Infrastrukturkosten durch den Wegfall externer Transformationsschichten

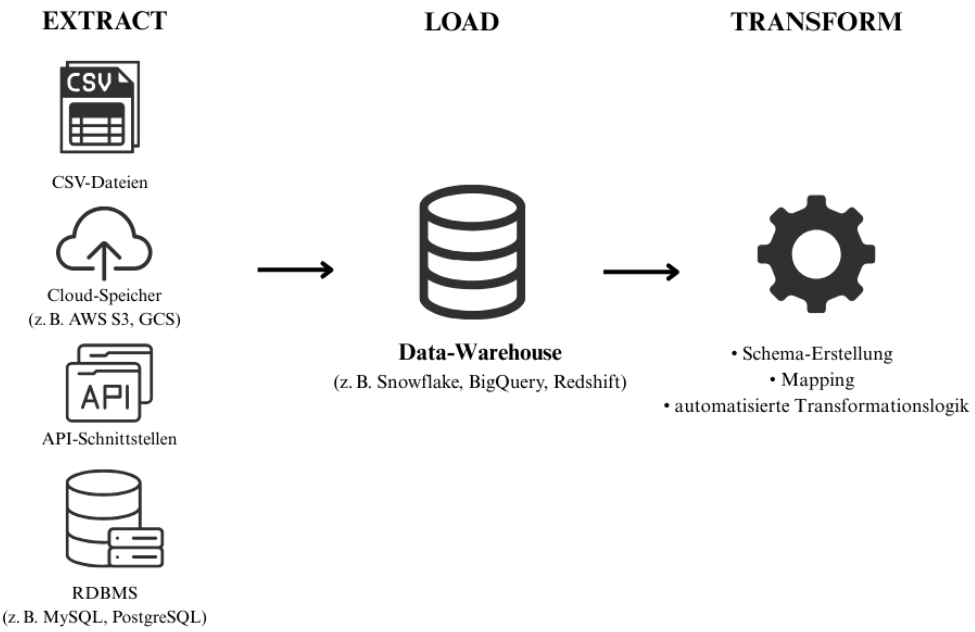

Abbildung 2.2: Ablaufmodell des ELT-Paradigmas. Eigene Darstellung in Anlehnung an (Singhal-22, S. 2).

(Walha-24, S. 26700). Zudem entsteht durch modellierte SQL-Logik eine höhere Transparenz und Nachvollziehbarkeit (Kimball-13, S. 37). Nachteile ergeben sich durch die nachgelagerte Transformation – was bei fehlerhaften Daten problematisch sein kann –, durch eine hohe Abhängigkeit vom Zielsystem (Vendor Lock-in) sowie durch die eingeschränkte Toolunterstützung bei nicht-SQL-basierten Architekturen (Walha-24, S. 26701); vgl. auch (Schulz-24, S. 183).

Einordnung in Typ A–C ELT eignet sich insbesondere für **Typ-B-Prozesse**, bei denen Daten aus mehreren Cloud-Systemen regelmäßig zentral konsolidiert werden (z. B. Shopify, Salesforce). Dank Cloud-Fokus und automatisierbarer Ladeprozesse (z. B. mit Airbyte oder Fivetran) können auch Nutzer ohne tiefgreifendes Infrastrukturwissen stabile Pipelines aufbauen. Für **Typ-C-Kontexte** ist ELT sinnvoll, sofern das Zielsystem fortgeschrittene Governance-Funktionen (z. B. dbt, GitOps) unterstützt. In **Typ-A-Szenarien** ist ELT meist überdimensioniert, da dort GUI-Tools ohne serverseitige SQL-Verarbeitung dominieren.

2.2.3 Change Data Capture (CDC) in der Datenintegration

Change Data Capture (CDC) ist ein datenbanknahes Verfahren zur inkrementellen Datenintegration, bei dem ausschließlich Änderungen (Inserts, Updates, Deletes) erkannt und übertragen werden – im Gegensatz zur vollständigen Tabellenextraktion bei ETL- oder ELT-Verfahren. Dadurch wird eine nahezu echtzeitfähige Synchronisation zwischen Quell- und Zielsystem ermöglicht (Walha-24, S. 26712; Schulz-24, S. 47).

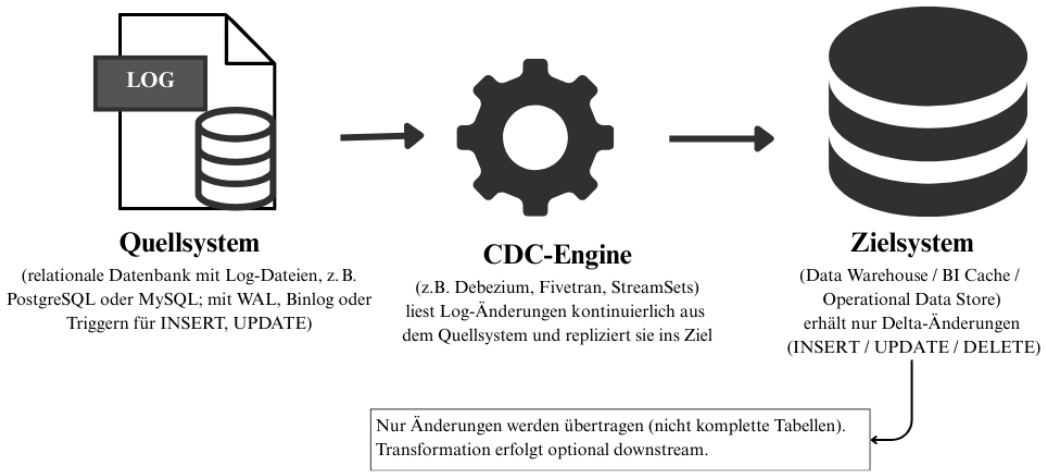

Abbildung 2.3: Ablaufmodell des CDC-Paradigmas. Eigene Darstellung in Anlehnung an (Walha-24, S. 26712).

Technisch basiert CDC auf der Auswertung von Änderungsprotokollen (z. B. Write-Ahead Logs) oder Datenbank-Triggern. Spezialisierte CDC-Engines wie Debezium, Fivetran oder StreamSets lesen diese Änderungen kontinuierlich aus und übertragen sie in Zielsysteme wie Data Warehouses oder BI-Caches (Popovic-24, S. 5). Die Transformation erfolgt optional downstream.

Zu den Vorteilen zählen minimale Datenlatenz durch eventbasierte Replikation, reduzierte Netzwerklast und Speicherbedarf sowie die Eignung für Echtzeit-Dashboards und operative BI (Schulz-24, S. 193); (Popovic-24, S. 6). Nachteile ergeben sich aus der Abhängigkeit von Datenbank-Logs und deren Formaten, dem Konfigurationsaufwand bei komplexen Systemlandschaften sowie der fehlenden Standardisierung hinsichtlich Reprocessing und Idempotenz (Schulz-24, S. 194; Popovic-24, S. 7).

Einordnung in Typ A–C CDC ist besonders für **Typ-B-Prozesse** geeignet, bei denen webbasierte Systeme oder SaaS-Dienste mit häufigen Änderungen inkrementell angebunden werden sollen. Auch in **Typ-C-Projekten** bietet CDC Vorteile, etwa zur Replikation strukturierter ERP-Daten in Echtzeit-Reportings. Für **Typ-A-Kontexte** ist CDC hingegen ungeeignet, da es persistente Log-Auswertung und spezifisches Infrastrukturwissen erfordert.

2.2.4 Virtuelle Integrationsarchitekturen (VDI)

Virtuelle Datenintegration (VDI) beschreibt den Zugriff auf verteilte, heterogene Quellsysteme über eine semantische Zugriffsschicht, ohne dass die Daten physisch extrahiert oder gespeichert werden. Die Abfragen erfolgen zur Laufzeit; Transformation und Harmonisierung finden dynamisch im Zugriffskontext statt – etwa über föderierte Views, Ontologien oder digitale Zwillinge (Putrama-24, S. 6; JIANG-21, S. 4; Feiler-10, S. 3). Zentrale Elemente bilden ein Vermittlungsdienst (Mediator), der die Anfragen koordiniert, sowie systemnahe Wrapper zur Abstraktion der Quellsysteme. Plattformen wie Denodo, Dremio oder SAP Smart Data Access nutzen diese Prinzipien zur Echtzeitaggregation betrieblicher Daten.

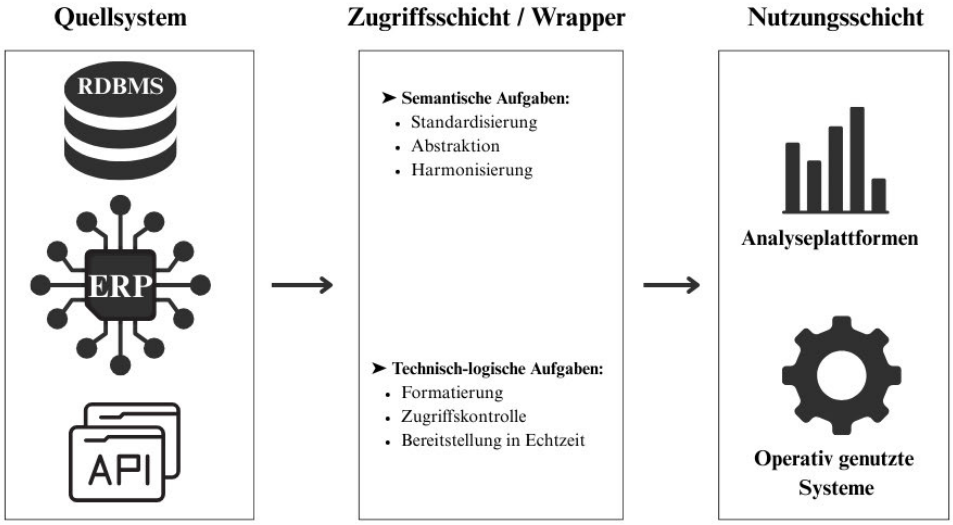

Abbildung 2.4: Architekturmodell der virtuellen Datenintegration. Eigene Darstellung in Anlehnung an (Putrama-24, S. 7)

Zu den Vorteilen zählen der Wegfall redundanter Datenspeicherung – was Speicher- und Infrastrukturkosten reduziert (Putrama-24, S. 10) –, eine hohe Flexibilität bei häufigen Änderungen der Quellstrukturen (Feiler-10, S. 3) sowie der Echtzeit-Zugriff auf dezentrale Datenquellen zur Verbesserung der Datenaktualität (JIANG-21, S. 4). Nachteile bestehen in der Abhängigkeit von Netzwerklatenz und Verfügbarkeit der Quellsysteme (Putrama-24, S. 10), in der fehlenden persistenten Historisierung bzw. Versionskontrolle (Feiler-10, S. 3) sowie in der Komplexität der Rechte- und Zugriffskontrolle in Multi-System-Umgebungen (JIANG-21, S. 5).

Einordnung in Typ A–C Virtuelle Integration eignet sich primär für **Typ-C-Projekte**, insbesondere in read-only-Szenarien, in denen eine zentrale Speicherung nicht erlaubt oder nicht erforderlich ist. In **Typ-B-Kontexten** kann VDI ergänzend bei föderierten API-Zugriffen eingesetzt werden. Für **Typ-A-Projekte** ist sie hingegen meist ungeeignet, da Konfiguration und Berechtigungsmanagement zu aufwändig sind.

2.2.5 Architekturvergleich im Kontext der Typen A–C: ETL, ELT, CDC, VDI

Die in den vorangegangenen Abschnitten dargestellten Architekturformen – ETL, ELT, CDC und VDI – unterscheiden sich grundlegend hinsichtlich Transformationsort, Zeitverhalten, Systemvoraussetzungen und Toolbindung. Tabelle 2.3 bietet eine systematische Gegenüberstellung zentraler Merkmale und deren Eignung für die Nutzungskontexte der Typen A–C.

Tabelle 2.3: Vergleich zentraler Datenintegrationsarchitekturen (eigene Darstellung in Anlehnung an (Kimball-13, S. Kap. 2; Schulz-24, S. Kap. 4; Doan-12, S. Kap. 2)

Kriterium	**ETL**	**ELT**	**CDC**	**VDI**
Technische Merkmale				
Transformationsort	extern vor DWH	im Ziel-DWH	im Zielsystem nach Log-Erkennung	on-demand bei Anfrage
Zeitverhalten	Batch	Batch/geplant	Near-Realtime / eventbasiert	Echtzeit / live
Datenhaltung	persistent im DWH	persistent im DWH	persistente Delta-Verarbeitung	keine Persistenz
Fehlerbehandlung	Staging / extern	Logging im DWH	Replay / Stream-Engine	eingeschränkt, quellenabhängig
Systemvoraussetzung	Staging-DB, Integrationsserver	performantes DWH mit SQL	Logzugriff, CDC-fähige Quelle	föderierter Zugriff, semantische Schnittstelle
Toolbeispiele	Talend, Pentaho	dbt, Fivetran, Matillion	Debezium, StreamSets	Denodo, Dremio, SAP SDA
Eignung für Projekttypen				
Typ A	eingeschränkt geeignet	ungeeignet	ungeeignet	ungeeignet
Typ B	bedingt geeignet	gut geeignet	gut geeignet	ergänzend geeignet
Typ C	gut geeignet	gut geeignet	sehr gut geeignet	ergänzend geeignet

Die Auswahl geeigneter Architekturen hängt eng mit Projektgröße, Quellstruktur, Änderungsfrequenz und infrastrukturellen Voraussetzungen zusammen. ETL und ELT gelten weiterhin als robuste Standardverfahren, während CDC insbesondere in modernen, ereignisgesteuerten Systemen zunehmend an Bedeutung gewinnt. Virtuelle Integration bietet spezifische Vorteile, wenn Persistenz technisch nicht erforderlich oder organisatorisch nicht zulässig ist – etwa bei föderierten Abfragen oder read-only-Zugriffen (Putrama-24, S. 10; JIANG-21, S. 4; Feiler-10, S. 3).

2.2.6 Werkzeugparadigmen im Kontext der Typen A–C: Bedienlogik und Automatisierungsgrad

Neben der Wahl der Integrationsarchitektur beeinflusst auch die konkrete Umsetzungstechnologie maßgeblich die Effizienz, Wartbarkeit und Benutzerakzeptanz eines Datenintegrationsvorhabens. Die nachfolgenden vier Technologieformen werden häufig unterschieden: GUI-basierte Tools, skriptgesteuerte Systeme, Auto-ETL-Frameworks sowie No-/Low-Code-Plattformen (Szilveszter-24, S. 6; Popovic-24, S. 5; Khan-24, S. 18).

GUI-basierte ETL-Tools Diese Werkzeuge bieten grafische Drag-&-Drop-Oberflächen für Datenflüsse und Transformationen. Beispiele sind *Pentaho*, *Talend Open Studio*, *Hop* oder *SAS DI Studio*. Sie sind besonders einsteigerfreundlich, jedoch bei komplexen Transformationslogiken oder der Versionskontrolle begrenzt (Popovic-24, S. 6). → Eignung: Typ A–B

Skriptbasierte Tools Skriptbasierte Lösungen wie *dbt*, *Apache Nifi* oder *Kettle CLI* erlauben deklarative Pipeline-Definitionen über YAML, SQL oder Python. Sie sind CI/CD-kompatibel und gut versionierbar, erfordern jedoch tiefere technische Kenntnisse (Szilveszter-24, S. 6). → Eignung: Typ B–C

Auto-ETL-Ansätze Auto-ETL-Plattformen wie *Fivetran*, *Hevo Data* oder *Estuary* bieten vorkonfigurierte Konnektoren mit automatischem Schema-Mapping und inkrementellem Laden. Sie minimieren den Setup-Aufwand, sind aber schwer anpassbar und häufig kostenintensiv (Khan-24, S. 19). → Eignung: Typ B

No-/Low-Code-ETL No-Code-Tools richten sich an Citizen Developers mit stark vereinfachter Bedienlogik, meist via Browser (z. B. *Parabola*, *Easyflow*, *Keboola*). Die Abstraktionstiefe reduziert Fehlerrisiken, schränkt jedoch Kontrolle und Erweiterbarkeit erheblich ein (Dash-22, S. 11). → Eignung: Typ A

Vergleich der Technologieformen

Tabelle 2.4: Vergleich gängiger Technologieformen für ETL-Prozesse

Kriterium	**GUI-basiert**	**Skriptbasiert**	**Auto-ETL**	**No-Code**
Setup-Komplexität	niedrig	hoch	sehr niedrig	sehr niedrig
Anpassbarkeit	mittel	hoch	niedrig	sehr niedrig
Versionskontrolle	begrenzt	vollständig	eingeschränkt	kaum vorhanden
Zielgruppe	Fachanwender	Data Engineers	Generalisten	Citizen Users
CI/CD-Fähigkeit	selten gegeben	nativ unterstützt	meist eingeschränkt	nicht vorgesehen
Typ-A–C-Eignung	A–B	B–C	B	A

2.2.7 Fehleranfälligkeit und Qualitätssicherungsmechanismen

In Datenintegrationsprojekten ist nicht nur die Architekturwahl entscheidend, sondern auch die Fähigkeit, Fehler zuverlässig zu erkennen, einzugrenzen und systematisch zu beheben. Je nach Architekturtyp (ETL, ELT, CDC, VDI) und Tooltechnologie (GUI, Skript, Auto-ETL) unterscheiden sich Logging-Möglichkeiten, Wiederholbarkeit, Testbarkeit und Fehlerdiagnose erheblich (Kimball-13, S. 15; Walha-24, S. 26688).

Typische Fehlertypen:

- **Syntaxfehler:** Transformationen schlagen aufgrund fehlerhafter Ausdrücke fehl (Kimball-13, S. 297).
- **Schemafehler:** Spalten oder Felder fehlen, sind umbenannt oder typinkompatibel (Kimball-13, S. 297; Walha-24, S. 26691).
- **Laufzeitfehler:** API-Timeouts, fehlende Quellverfügbarkeit, Tokenfehler bei SaaS-Diensten (Walha-24, S. 26691).
- **Validierungsfehler:** Fehlende Plausibilität, Duplikate, inkonsistente Semantik (Kimball-13, S. 299; Szilveszter-24, S. 4).

Qualitätssicherung und Logging: Moderne Tools wie *dbt*, *Talend*, *Airbyte* oder *StreamSets* bieten umfangreiche Protokollierungsmöglichkeiten mit Wiederholung fehlgeschlagener Jobs, Prüfsummen, Validierungsregeln und Alerting-Funktionen. GUI-Tools

visualisieren Fehler, während skriptbasierte Systeme ein präziseres Debugging ermöglichen (Walha-24, S. 26688; Kimball-13, S. 305).

Herausforderungen kleiner Projekte: In kleinen oder mittleren Projekten fehlt häufig ein dediziertes Monitoring. Fehlerbehandlung erfolgt ad hoc, Logging wird lokal gespeichert oder gar nicht versioniert. Studien zeigen, dass 71 % der befragten Anwender strukturierte Fehlerbehandlungsmechanismen vermissen; nur 29 % setzen auf automatisierte Validierungen (Hu-24, S. 6; Szilveszter-24, S. 5).

Best Practices:

- Entwicklung und Test in isolierter Umgebung mit synthetischen Testdaten (Kimball-13, S. 305).
- Logging und Versionierung aller Pipelines (Kimball-13, S. 297; Walha-24, S. 26688).
- Einführung automatischer Validierungen vor jedem Load (Kimball-13, S. 310).
- Einrichtung von Retry-Logik bei CDC- oder Streaming-Prozessen (Walha-24, S. 26712).

Wirtschaftliche Relevanz: Über 60 % der Gesamtkosten für Fehlerbehebung entstehen erst in der Integrations- und Testphase. Durch frühzeitiges Logging, simulierbare Transformationen und virtuelle Testläufe lassen sich diese Kosten deutlich reduzieren (Feiler-10, S. 3; Kimball-13, S. 312).

2.2.8 Begriffliche Abgrenzung und Marktverständnis von „ETL-Tools“

Der Begriff „ETL-Tool“ wird in Fachliteratur, Praxis und insbesondere im Softwaremarkt uneinheitlich verwendet. Während er ursprünglich eine spezifische Funktion bezeichnete – nämlich das extraktive, transformierende und ladende Verarbeiten strukturierter Daten (ETL) –, ist „ETL-Tool“ heute ein unscharfer Sammelbegriff für unterschiedlichste Integrationslösungen. Diese reichen von klassischen ETL-Werkzeugen bis zu modernen Cloud-ELT-, Reverse-ETL-, CDC- oder Auto-ETL-Plattformen (Popovic-24, S. 4; Walha-24, S. 26713; Dash-22, S. 11).

Viele Anbieter nutzen den Begriff „ETL-Tool“ als Marktlabel, selbst wenn ihre Produkte primär auf ELT oder CDC basieren. So werben beispielsweise Plattformen wie *Fivetran*,

Airbyte oder *Hevo* mit vollständiger ETL-Funktionalität, obwohl Transformationen dabei nicht vor dem Laden, sondern im Zielsystem erfolgen (ELT). Auch bei deklarativen Frameworks wie *dbt* oder eventbasierten Tools wie *Debezium* wird die Bezeichnung „ETL-Tool“ häufig verwendet, obwohl das klassische Dreiphasenmodell nicht umgesetzt wird (Khan-24, S. 18; Schulz-24, S. 45).

Die vorliegende Arbeit verwendet den Begriff daher **nicht im marktwirtschaftlichen Sinne**, sondern bezieht sich ausschließlich auf das konkrete Funktionsspektrum eines Werkzeugs – d. h. auf die tatsächlich umgesetzte Architektur (ETL, ELT, CDC etc.), die Transformationslogik (grafisch vs. skriptbasiert), das Fehlerverhalten und die projekttypische Eignung.

Fazit: Für die vorliegende Arbeit wird der Begriff „ETL-Tool“ methodisch aufgebrochen: Nicht die Anbieterbezeichnung, sondern die analysierte technische Architektur steht im Fokus der Bewertung. Nur so lassen sich Integrationslösungen nachvollziehbar entlang definierter Projekttypen und Bewertungskriterien vergleichen.

2.3 Bewertungssystematik für ETL-/ELT-Tools

2.3.1 Ziel, Methodik und Bewertungsrahmen

Ziel dieses Abschnitts ist die Entwicklung eines nachvollziehbaren, empirisch fundierten Bewertungssystems für ETL-/ELT-Tools, das speziell auf die Anforderungen kleiner bis mittelgroßer Datenintegrationsprojekte zugeschnitten ist. Neben klassischen funktionalen Kriterien berücksichtigt der Ansatz auch Aspekte wie Anpassbarkeit, Fehlertoleranz, Wartbarkeit, Benutzerfreundlichkeit, Kosten und Toollebensdauer – jeweils differenziert nach den in Kapitel 2.1 eingeführten Projekttypen A, B und C.

Das Bewertungssystem wird in mehreren methodischen Schritten entwickelt:

- **Systematische Sammlung:** Zusammenführung potenziell relevanter Kriterien auf Basis der Literatur (Kap. 2.1–2.2), ISO 25010, Experteninterviews, Umfrage und Toolanalyse (Kap. 3).
- **Konsolidierung:** Reduktion durch inhaltliche Prüfung, Redundanzanalyse und Bewertung der empirischen Relevanz.

- **Triangulation:** Validierung hinsichtlich der Eignung für spezifische Projekttypen durch Experten- und Nutzerfeedback.
- **Scorecard-Bildung:** Auswahl von 14 zentralen Kriterien als Bewertungsgrundlage.
- **Kategorisierung und Gewichtung:** Zuordnung zu den übergeordneten Dimensionen Funktionalität, Bedienbarkeit und Wartbarkeit (vgl. Kap. 2.3.5).

Die Orientierung an ISO/IEC 25010 dient als theoretischer Bezugsrahmen, ersetzt jedoch nicht die empirisch-heuristische Ableitung und kontextuelle Anpassung. Die erarbeitete Scorecard bildet die Grundlage für die Bewertungslogik in Kapitel 3 sowie die konkrete Toolanalyse in Kapitel 4.

Hinweis: Die empirische Fundierung basiert auf einer soliden, jedoch begrenzten Datenbasis (vier Experteninterviews, 62 Umfrageteilnahmen; vgl. Kap. 3.3, 3.5). Methodische Limitationen werden in Kapitel 3.9 und 5.4 kritisch reflektiert. Die evidenzbasierte Gewichtung, Scorebildung (Kap. 3.6.3) und das Skalenmodell (vgl. Tabelle 2.8 in Kap. 2.3.6) werden im theoretischen Teil dieser Arbeit konzipiert und fundiert.

2.3.2 Theoretische Verankerung im Qualitätsmodell ISO/IEC 25010

Die internationale Norm ISO/IEC 25010:2011 dient in dieser Arbeit als strukturierender Bezugsrahmen für die Ableitung und Systematisierung von Bewertungskriterien. Sie beschreibt ein umfassendes Qualitätsmodell für Softwareprodukte und differenziert acht zentrale Merkmale: *Functional suitability*, *Performance efficiency*, *Compatibility*, *Usability*, *Reliability*, *Security*, *Maintainability* und *Portability* (ISO25010, S. 2).

Die in dieser Arbeit entwickelten Kriterien wurden empirisch und heuristisch abgeleitet (vgl. Kap. 2.3.3) und anschließend den übergeordneten ISO-Kategorien zugeordnet. Die finale Scorecard (Kap. 2.3.5) mit 14 Kriterien lässt sich folgendermaßen zuordnen:

- **Functional suitability:** Transformationsarchitektur, Transformationstiefe, Quellkonnektivität, Automatisierungsgrad
- **Usability:** GUI-Komplexität, Konfigurierbarkeit, Nutzerführung, Setup-Aufwand
- **Reliability:** Logging & Monitoring, Fehlertoleranz, Validierungslogik, Retry-Mechanismen
- **Maintainability:** Erweiterbarkeit, Wiederverwendbarkeit, Versionskontrolle, Testbarkeit, Dokumentations-/Community-Support

- **Portability:** Plattformunabhängigkeit, Hardwareanforderungen, Skalierbarkeit

Die Kategorisierung stärkt die Anschlussfähigkeit an etablierte Qualitätsmodelle und ermöglicht eine strukturierte Einordnung entlang allgemein akzeptierter Dimensionen. Die Merkmale *Security*, *Compatibility* und *Performance efficiency* wurden bewusst ausgeklammert, da sie im Kontext kleiner und mittlerer Datenintegrationsprojekte selten systematisch erfasst oder vergleichend bewertet werden können.

Die konkrete Operationalisierung erfolgt entlang der in Kapitel 2.3.3–2.3.6 beschriebenen Schritte und wird in Kapitel 3 methodisch umgesetzt.

2.3.3 Kriterienidentifikation und Aufbau der Longlist

Zur Entwicklung eines passgenauen Bewertungssystems für kleine und mittlere Datenintegrationsprojekte wurde zunächst eine systematische Longlist mit 64 Bewertungskriterien erstellt. Diese wurde auf Basis der Literatur (Kap. 2.1–2.2), der Norm ISO 25010, einer Online-Umfrage sowie vier Experteninterviews abgeleitet – mit dem Ziel, alle relevanten Dimensionen vollständig und transparent abzudecken.

Die Kriterien erfassen zentrale Aspektbereiche wie Quellanbindung, Transformation, Logging, Automatisierung, Fehlerbehandlung, Usability, Wartbarkeit, Kosten, Plattformunabhängigkeit und Support. Theorie-, erfahrungs- und nutzerbasierte Perspektiven wurden dabei gezielt kombiniert.

Tabelle 2.5: Auszug aus der Longlist der Bewertungskriterien

Kriterium	Beschreibung	Quelle
Quellsysteme	Art und Anzahl angebundener Datenquellen (CSV, REST, ERP)	Kap. 2.1.5, Typologie
Automatisierungsgrad	Manuell, zeitgesteuert, orchestriert (z. B. GitOps)	Kap. 2.1.5, Typologie
Transformationsort	Architekturtyp (ETL, ELT, CDC, virtuell)	Kap. 2.2.5, Tab. 2.3
Fehlerbehandlung	Logging, Retry-Logik, Replay-Funktionalität	Kap. 2.2.5, Tab. 2.3
GUI-Komplexität	Umfang und Logik der grafischen Benutzeroberfläche	Kap. 2.3.2, ISO 25010
Kosteneffizienz	Open-Source-Verfügbarkeit oder geringe Lizenzkosten	Umfrage, Interviews
Community-Support	Verfügbarkeit von Foren, Doku, Hilfeportalen	Umfrage, Interviews
Plattformunabhängigkeit	OS-/Cloud-Neutralität und Betriebsflexibilität	Kap. 2.3.2, ISO 25010
...	...	...

Hinweis: Die vollständige Longlist mit allen 64 Kriterien befindet sich auf dem Datenträger.

Die Longlist bildet die Grundlage für die anschließende Reduktion und Gewichtung (vgl. Kap. 2.3.4) und ist inhaltlich auf den spezifischen Kontext kleiner und mittlerer Datenintegrationsprojekte zugeschnitten.

2.3.4 Konsolidierung und Auswahl der Bewertungskriterien

Zur Vorbereitung der finalen Scorecard wurden die 64 erhobenen Rohkriterien auf inhaltliche Überschneidungen und geringe Relevanz geprüft. Funktional äquivalente oder stark korrelierende Merkmale wurden zusammengeführt, gering bewertete Aspekte ausgeschlossen.

Die Auswahlentscheidung basierte auf aggregierten Bewertungsdaten: Kriterien mit durchgängig niedriger Gewichtung ($< 0{,}1$) in allen Projekttypen wurden eliminiert. Zusammenführungen betrafen insbesondere GUI-Aspekte, Logging/Monitoring sowie Automatisierungsdimensionen. Tabelle 2.6 zeigt exemplarische Zusammenführungs- und Streichentscheidungen.

Tabelle 2.6: Auszug aus der Streich- und Zusammenführungsübersicht

Orig-ID(s)	Ursprungskriterium(e)	Aktion	Begründung (Kurzform)
1, 31, 47, 58	Quellsysteme, Quellkonnektivität, Flexibilität & Integration	Merge	semantisch identisch; hohe Korrelation
6, 30	Transformationsort, -logik	Merge	funktionale Einheitlichkeit
4, 15, 32	Automatisierungsgrad, CI/CD-Fähigkeit, Automatisierungsmöglichkeiten	Merge	Skala von manuell → CI/CD
20, 26, 27	Logging-Granularität, -Versionierbarkeit, Monitoring	Merge	gemeinsam „Observability“ abbildend
33, 35, 46, 50, 56	GUI-Komplexität, Nutzerführung, visuelle Prozessdarstellung	Merge	hohe semantische Deckung
52, 63	KI/ML-Integration	Remove	niedrige Relevanzbewertung
...	...	...	...

Die vollständige Übersicht der Tabelle ist auf dem Datenträger hinterlegt.

Nach Anwendung der Reduktionslogik verblieben 14 valide Bewertungskriterien. Diese bilden die Grundlage der Scorecard und decken alle untersuchten Projekttypen (A–C) angemessen ab.

Fazit: Durch eine systematische Reduktion wurde eine kompakte, bewertbare und typübergreifend belastbare Kriteriensammlung geschaffen, die den methodischen Grundstein für den Vergleich von ETL-/ELT-Tools bildet.

2.3.5 Kategorisierung und Struktur der finalen Scorecard

Die finale Auswahl von 14 Bewertungskriterien wurde drei Hauptkategorien zugeordnet: **Funktionalität**, **Bedienbarkeit** und **Wartbarkeit**. Diese Einteilung orientiert sich an den Qualitätsmerkmalen der ISO/IEC 25010, wurde jedoch kontextualisiert für kleine und mittlere Datenintegrationsprojekte. Sie bildet die strukturelle Grundlage für die spätere Toolbewertung und Gewichtung im Bewertungsmodell (vgl. Kap. 3).

Tabelle 2.7: Zuordnung der 14 Scorecard-Kriterien zu den drei Hauptkategorien

Kriterium	**Kategorie**
GUI-Komplexität, Nutzerführung, Konfigurierbarkeit	Bedienbarkeit
Setup-Aufwand	Bedienbarkeit
Kosteneffizienz	Bedienbarkeit
Transformationsarchitektur	Funktionalität
Transformationstiefe	Funktionalität
Quellsystem-Vielfalt & -Konnektivität	Funktionalität
Automatisierungsgrad	Funktionalität
Plattformunabhängigkeit, Hardwareanforderungen, Skalierbarkeit	Funktionalität
Logging & Monitoring / QS-Reporting	Wartbarkeit
Validierungslogik, Schema-Drift, Retry, Debugging, Fehlertoleranz	Wartbarkeit
Testbarkeit	Wartbarkeit
Erweiterbarkeit, Wiederverwendbarkeit, Versionskontrolle	Wartbarkeit
Dokumentations-/Community-Support	Wartbarkeit
Betriebs-/Laufzeitrisiko	Wartbarkeit

Die Zuordnung erfolgte entlang funktional-inhaltlicher Kriterienlogik: *Plattformunabhängigkeit* wurde als funktionales Merkmal klassifiziert, da sie die technische Einsatzflexibilität definiert. *Benutzerfreundlichkeit* zählt zur Bedienbarkeit, da sie den Zugang zur Nutzung und deren Effizienz beeinflusst. *Logging und Monitoring* wiederum wurden der Wartbarkeit zugeordnet, da sie essenziell für Betriebssicherheit und Fehlernachverfolgung sind.

Durch diese strukturierte Kategorisierung lassen sich die Kriterien systematisch gewichten und konsistent in die empirische Toolbewertung überführen.

2.3.6 Gewichtung der Kriterien nach Projekttyp (Typ A–C)

Die 14 Scorecard-Kriterien wurden für drei typische Projekttypen unterschiedlich gewichtet: **Typ A** (GUI-Ad-hoc), **Typ B** (Batch-SaaS) und **Typ C** (Governance/API). Die zugrunde liegende Projekttypologie wurde in Abschnitt 2.1.5 definiert.

Die Gewichtung erfolgte auf einer Skala von 0 (irrelevant) bis 5 (erfolgskritisch). Zwischenwerte (1–4) bilden gestufte Relevanzniveaus ab. Die Zuordnung basiert auf Mittelwerten aus der Online-Umfrage, qualitativen Interviewaussagen und Toolanalysen. Bei signifikanten Abweichungen zwischen den Quellen erfolgte eine justierende Anpassung um ± 1 Punkt zur Sicherstellung empirischer Plausibilität.

Tabelle 2.8: Projekttypenspezifische Gewichtung der 14 Scorecard-Kriterien (0 = irrelevant, 5 = kritisch)

Kriterium	Typ A	Typ B	Typ C	Empirische Basis
GUI-Komplexität / Nutzerführung	5	3	2	Umfrage Ø 4,3; Fokus auf Citizen User
Setup-Aufwand	5	4	2	Interview Ismailova: „quick start“
Transformationsarchitektur	2	5	5	Toolanalysen: ELT/CDC bei B und C
Transformationstiefe	2	4	5	Interview Miller: komplexe Joins
Quellsystem-Vielfalt & -Konnektivität	3	5	4	Umfrage Ø 4,2 (Flexibilität)
Automatisierungsgrad	2	4	5	CI/CD-Praxis nur bei Typ C
Logging & Monitoring	3	4	5	Zitat: „Fehler schnell finden“
Validierung, Fehlertoleranz, Debugging	3	4	5	Interviews & Abschnitt 2.2.7
Testbarkeit	2	3	5	Pipeline-Tests nur bei Typ C gefordert
Wiederverwendbarkeit / Versionskontrolle	2	3	5	GitOps-Verpflichtung bei C
Dokumentation / Community-Support	4	4	3	Umfrage Ø 3,9 (OSS-Nutzung)
Kosteneffizienz	5	4	2	Preisbewusstsein bei KMU
Plattformunabhängigkeit / Skalierbarkeit	3	5	4	Cloud-SaaS-Fokus bei Typ B
Betriebs-/Laufzeitrisiko	3	4	5	Abschnitt 2.2.7 (Risikobetrachtung)

Im Folgenden werden exemplarisch drei besonders charakteristische Gewichtungen erläutert:

- **GUI-Komplexität** erhielt für Typ A die Maximalwertung, da Usability für nichttechnische Endnutzer erfolgskritisch ist.
- **Automatisierungsgrad** wurde ausschließlich für Typ C mit der Höchstbewertung versehen, da dort CI/CD-Pipelines kontextuell vorausgesetzt werden.
- **Testbarkeit** wurde nur für Typ C als kritisch eingestuft, da Governance-Projekte strukturierte, versionierte Pipeline-Tests erfordern.

Fazit: Die Gewichtung ist sowohl empirisch als auch theoretisch fundiert und bildet die Grundlage für die kontextsensitiv differenzierte Bewertung der ETL-/ELT-Tools in den folgenden Kapiteln.

2.3.7 Zusammenfassung und methodische Überleitung zu Kapitel 3

Die Entwicklung der Scorecard folgte einem strukturierten und nachvollziehbaren Vorgehen: Ausgehend von einer umfassenden Longlist wurden durch konsistente Konsolidierung und empirisch gestützte Reduktion 14 bewertungsrelevante Kriterien identifiziert. Diese wurden drei Hauptkategorien zugeordnet und typenspezifisch gewichtet, um den Anforderungen kleiner und mittlerer Datenintegrationsprojekte gerecht zu werden.

Die Nachvollziehbarkeit der Auswahl wird durch dokumentierte Streich- und Zusammenführungsentscheidungen sowie die trianguläre Fundierung durch Umfrage, Experteninterviews und Toolanalysen gestützt. Die resultierende Scorecard ist sowohl praxisorientiert als auch methodisch abgesichert.

Die ausgewählten Kriterien erfassen klassische Aspekte wie GUI, Transformation und Logging ebenso wie aktuelle Anforderungen wie Erweiterbarkeit, Community-Support oder Plattformunabhängigkeit. Die typendifferenzierte Gewichtung ermöglicht eine kontextsensitive Bewertung entlang realitätsnaher Nutzungsszenarien.

Kapitel 3 überträgt diesen Bewertungsrahmen auf konkrete ETL-/ELT-Werkzeuge. Die Scorecard fungiert dabei als strukturiertes Analyseinstrument und bildet die methodische Grundlage für eine anwendungsbezogene Bewertung.

Hinweis zur Scorecard

Die Scorecard stellt das methodische Kernstück der werkzeuggestützten Evaluation dar. In Kapitel 3 dient sie nicht zur direkten Bewertung, sondern zur Prüfung der Bewertbarkeit: Nur Tools mit hinreichender Informationsgrundlage für den Großteil der Kriterien werden zur finalen Analyse in Kapitel 4 zugelassen.

3 Methodisches Vorgehen

3.1 Methodisches Gesamtkonzept

Dieses Kapitel leitet das methodische Vorgehen der Toolbewertung ein. Ziel ist es, die Bewertungslogik nachvollziehbar darzustellen und in zwei klar getrennte Hauptphasen zu gliedern: **Phase 1 – Toolfilterung und Bewertbarkeitsprüfung** sowie **Phase 2 – Scorecard-basierte Bewertung**. Grundlage bildet die in Kapitel 2.3 entwickelte Scorecard mit 14 projekttypenspezifisch gewichteten Kriterien für kleine und mittlere Integrationsprojekte (vgl. Tabelle 2.8).

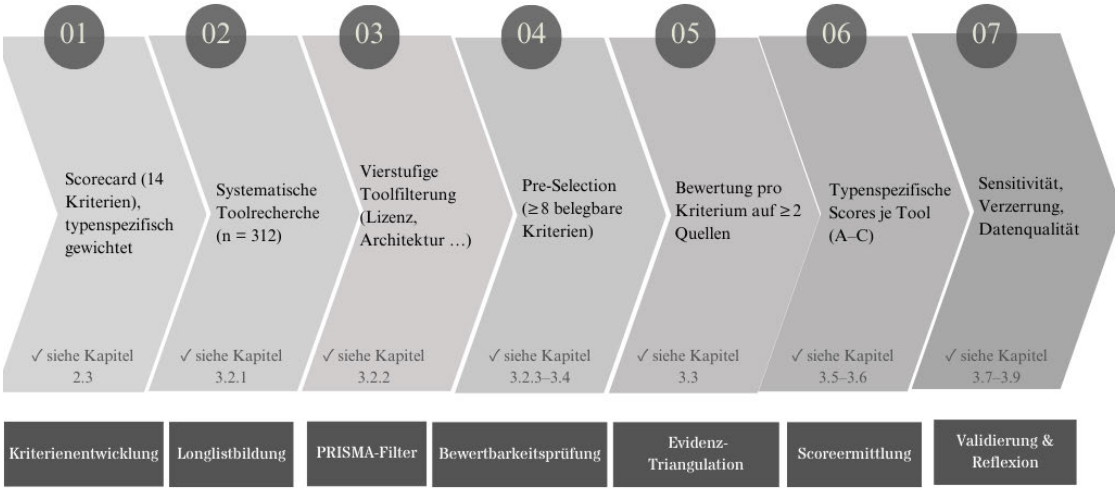

Abbildung 3.1: End-to-End-Evaluationsprozess: Siebenstufiger Bewertungsablauf von der Scorecard-Entwicklung über die Toolfilterung bis zur evidenzbasierten Bewertung und Validierung

Phase 1 – Toolfilterung und Bewertbarkeitsprüfung. In Kapitel 3.2–3.4 erfolgt zunächst keine inhaltliche Bewertung, sondern eine methodisch fundierte Vorauswahl. Ziel ist es, aus einer initialen Longlist von 312 Tools diejenigen auszuwählen, die

- mindestens einem der drei Projekttypen A–C (vgl. Tabelle 2.2) eindeutig zugeordnet werden können,
- funktionale und technologische Mindestanforderungen erfüllen (vgl. Tabellen 2.3–2.4),
- und für mindestens 8 der 14 Scorecard-Kriterien eine belastbare Evidenzgrundlage aufweisen (vgl. Tabelle 2.7).

Nur die verbleibenden Tools gelten als *methodisch bewertbar* und werden in Phase 2 weiter analysiert.

Phase 2 – Scorecard-basierte Bewertung. In Kapitel 3.5–3.9 werden die bewertbaren Tools entlang der Scorecard-Kriterien systematisch analysiert. Abbildung 3.2 visualisiert die fünf methodischen Teilschritte – von der Gewichtung über die evidenzbasierte Scoring-Logik bis zur Ergebnisabsicherung.

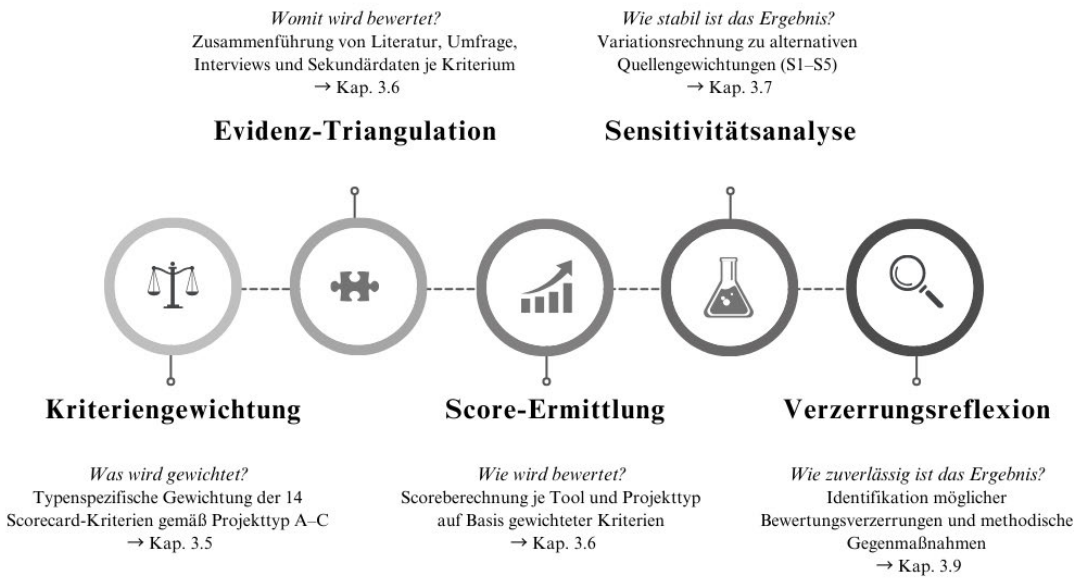

Abbildung 3.2: Scorecard-basierte Bewertung (Phase 2): Fünfstufiger Bewertungsprozess

Die Schritte beantworten fünf zentrale Bewertungsfragen:

- **Was wird gewichtet?** – Projekttypenspezifische Gewichtung der Scorecard-Kriterien (Kap. 3.5)

- **Womit wird bewertet?** – Evidenzquellen je Kriterium: Literatur, Dokumentation, Umfrage, Interviews (Kap. 3.6)
- **Wie wird bewertet?** – Gewichtete Scoreermittlung je Tool und Projekttyp (Kap. 3.6)
- **Wie stabil ist das Ergebnis?** – Variation der Quellengewichtung (Kap. 3.7)
- **Wie zuverlässig ist das Ergebnis?** – Prüfung und Reduktion potenzieller Verzerrungen (Kap. 3.9)

Evidenzmodell zur Kriterienbewertung. Die Scorecard-Bewertung basiert auf vier komplementären Datenquellen: wissenschaftliche Literatur, strukturierte Tool-Dokumentation, Online-Umfrage und Experteninterviews. Für jedes Kriterium ist eine Bewertung nur dann zulässig, wenn mindestens zwei dieser Quellen konsistent belegt sind. Abbildung 3.3 zeigt das zugrunde liegende Evidenzmodell.

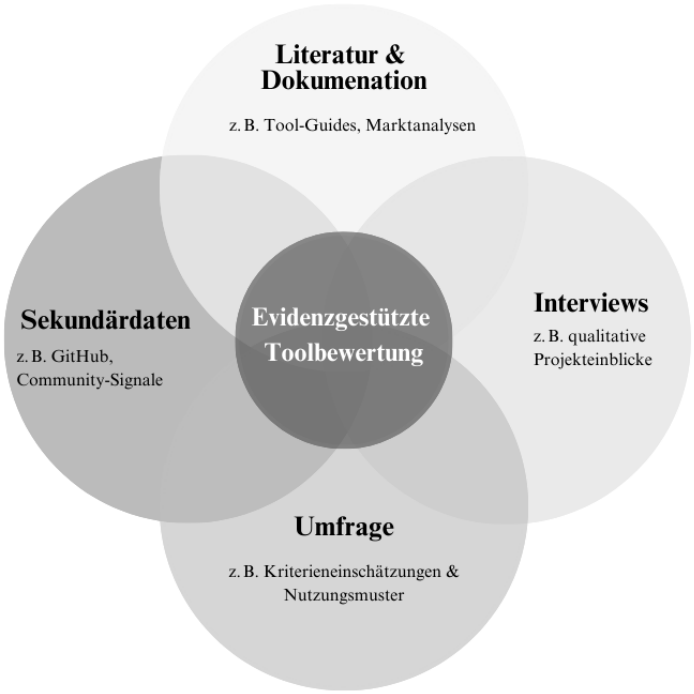

Abbildung 3.3: Evidenzmodell zur Scorecard-Kriterienbewertung
Bewertungsgrundlage aus vier komplementären Quellen

Fazit. Die folgenden Kapitel beschreiben die Umsetzung dieses Bewertungsmodells im Detail. Kapitel 3.2–3.4 dokumentieren das Filter- und Auswahlverfahren. Kapitel 3.5–3.9 erläutern die Bewertungslogik, ihre empirische Fundierung und methodische Absicherung. Kapitel 4 überträgt das Modell exemplarisch auf fünf konkrete Tools.

3.2 Toolauswahlverfahren

3.2.1 Recherchebasis und Longlist-Erstellung

Ziel dieses Abschnitts ist die Erstellung einer umfassenden Longlist potenzieller ETL-/ELT-/CDC-Werkzeuge, die als Ausgangsbasis für die nachfolgenden methodischen Filterschritte dient. Die Longlist verfolgt keinen Bewertungszweck, sondern dient der möglichst vollständigen Erfassung des relevanten Toolmarkts im Bereich datenintegrationsbezogener Plattformen. Ausschlüsse erfolgen erst im Rahmen des PRISMA-orientierten Auswahlverfahrens (vgl. Abschnitt 3.2.2).

Die Recherche erfolgte im Zeitraum von Dezember 2024 bis April 2025. Berücksichtigt wurden sowohl wissenschaftliche Studien (z. B. Gartner-24, G2-25) und Marktanalysen als auch frei zugängliche Quellen wie Fachportale, Open-Source-Verzeichnisse und GitHub-Repositorien. Ziel war die Identifikation sowohl kommerzieller als auch quelloffener Werkzeuge mit Bezug zur operativen oder analytischen Datenintegration.

Verwendete Suchbegriffe umfassten unter anderem:

- „open-source ETL tool“,
- „ELT platform for data integration“,
- „lightweight data pipeline“,
- „GitHub ETL stars $>$ 500“.

Ein Tool wurde in die Longlist aufgenommen, wenn es öffentlich dokumentiert, aktiv gepflegt und funktional eindeutig dem Integrationsbereich zuordenbar war – unabhängig davon, ob es später alle Scorecard-Kriterien erfüllt oder einem konkreten Projekttyp zugewiesen werden kann. Eine technische Prüfung oder Bewertung erfolgte zu diesem Zeitpunkt noch nicht.

Zur strukturierten Erfassung wurde eine interne Tool-Datenbank aufgebaut, in der pro Werkzeug folgende Merkmale dokumentiert wurden: Name, Architekturtyp (z. B. ETL, ELT, CDC), Lizenzmodell (Open Source, kommerziell, hybrid), Benutzeroberfläche (GUI, CLI, API-only), Veröffentlichungsjahr sowie Community-Metriken (z. B. GitHub-Stars, Release-Zyklus). Diese Merkmalsstruktur bildet die Grundlage für die anschließende PRISMA-Filterung (vgl. Abschnitt 3.2.2).

Die finale Longlist umfasste zum Zeitpunkt des Rechercheabschlusses insgesamt **312 Tools**. Aus Gründen der Lesbarkeit und Relevanz wird sie nicht im Anhang veröffentlicht, diente jedoch als zentrale Arbeitsgrundlage für die systematische Vorauswahl.

3.2.2 PRISMA-basierte Toolfilterung: Typenorientierte Ausschlusslogik

Ziel dieses Abschnitts ist die systematische Reduktion der in Abschnitt 3.2.1 zusammengestellten Longlist durch ein vierstufiges, methodisch fundiertes Ausschlussverfahren. Die Filterlogik basiert auf den in Kapitel 2 definierten Anforderungen kleiner und mittlerer Integrationsprojekte sowie der Typologie A–C (vgl. Tabelle 2.2).

Die angewendeten Filter operationalisieren Anforderungen hinsichtlich Typkompatibilität, Architekturform, Technologieform und methodischer Zugänglichkeit. Nur Werkzeuge, die sämtliche Kriterien erfüllen, gelten als potenziell bewertbar und werden der nächsten Stufe (Bewertbarkeitsprüfung, Abschnitt 3.2.3) zugeführt.

Filter 1 – Typkompatibilität:
Das Tool muss mindestens einem der drei Nutzungstypen (A–C) eindeutig zuordenbar sein. Werkzeuge ohne sinnvolle Einsatzoption im Kontext kleiner oder mittlerer Projekte wurden ausgeschlossen.

Filter 2 – Architekturform:
Das Tool muss technisch in einer geeigneten Architektur (ETL, ELT, CDC, VDI) realisiert sein (vgl. Tabelle 2.3). Inkompatible Varianten wie reine Reverse-ETL-Plattformen wurden ausgeschlossen.

Filter 3 – Technologieform:
Die Technologie muss zum jeweiligen Projekttyp passen (z. B. GUI für Typ A, CI/CD-fähige Skripte für Typ C; vgl. Tabelle 2.4). CLI-only-Werkzeuge ohne dokumentierte Schnittstelle wurden für Typ A ausgeschlossen.

Filter 4 – Zugang und Nutzbarkeit:

Das Tool muss entweder quelloffen oder über eine dauerhaft zugängliche Free-Tier-Version verfügbar sein. Vollständig proprietäre Lösungen ohne dokumentierten Testzugang wurden ausgeschlossen (vgl. Kapitel 2.1.5, „Ressourcenengpässe“).

Alle vier Filter wurden sequenziell angewendet und in der Tooldatenbank binär („Y/N“) dokumentiert. Ein Ausschluss in einer beliebigen Kategorie führte zur vollständigen Eliminierung aus dem weiteren Verfahren.

Ergebnis. Nach Anwendung sämtlicher Filter verblieben **28 Tools** aus der ursprünglichen Longlist von 312. Diese wurden in Abschnitt 3.2.3 einer detaillierten Bewertbarkeitsprüfung unterzogen.

Visualisierung. Abbildung 3.4 zeigt den Ausschlussprozess als PRISMA-ähnliche Darstellung. Im Unterschied zu klassischen PRISMA-Flows (Page-21, S. 71) liegt der Fokus hier auf typkompatibler Ausschlusslogik.

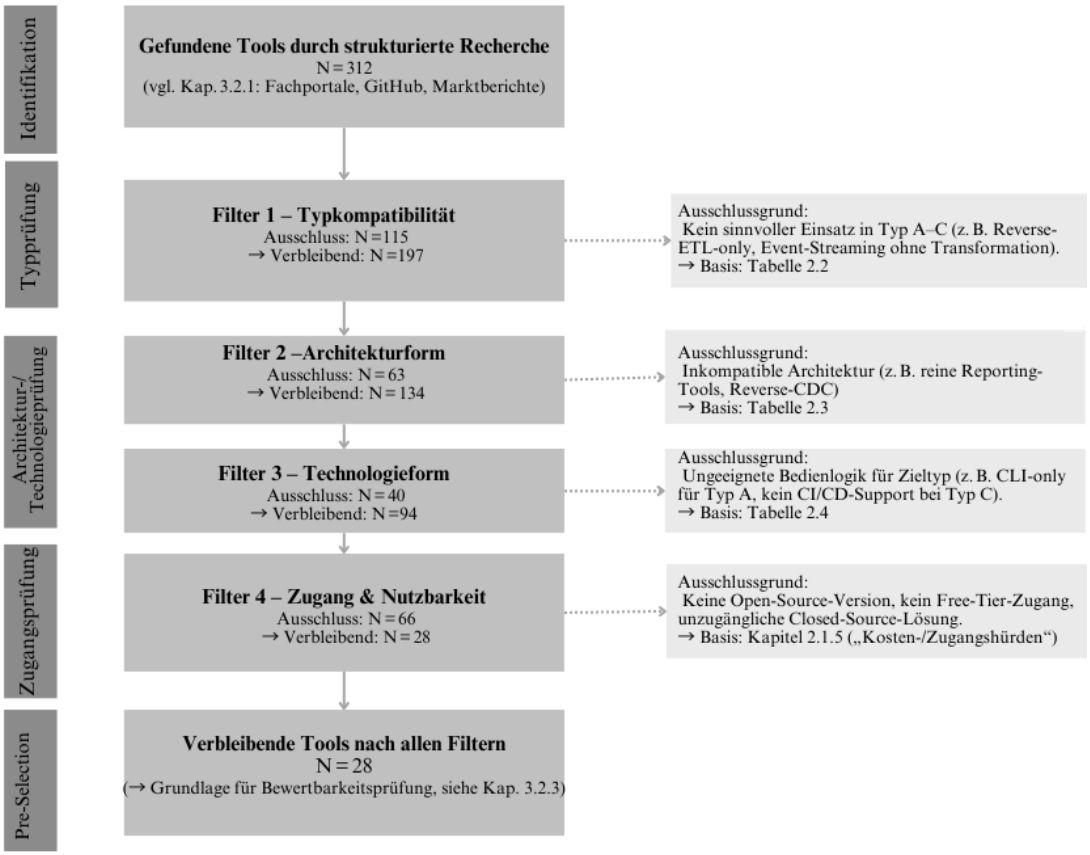

Abbildung 3.4: PRISMA-basierte Typenfilterung: Ausschlussprozess auf Basis von Nutzungstyp, Architektur- und Technologiekompatibilität

3.2.3 Bewertbarkeitsprüfung: Evidenzbasis zur Scorecard-Anwendung

Nach der in Abschnitt 3.2.2 durchgeführten Filterung verblieben insgesamt 28 Tools, die prinzipiell den Anforderungen kleiner und mittlerer Datenintegrationsprojekte entsprechen. Um diese Werkzeuge im nächsten Schritt systematisch auf Basis der Scorecard (Kapitel 2.3) bewerten zu können, ist eine belastbare Evidenzgrundlage erforderlich.

Ziel der Bewertbarkeitsprüfung. Ziel ist die Identifikation jener Tools, für die eine methodisch abgesicherte Scorecard-Anwendung möglich ist. Als Mindestanforderung gilt: Für ein Tool müssen belastbare Informationen zu mindestens **8 der 14 Scorecard-**

Kriterien vorliegen, wobei jedes dieser Kriterien durch mindestens zwei unabhängige Quellen abgesichert sein muss. Als valide Evidenzquellen gelten:

- strukturierte Tool-Dokumentation,
- wissenschaftliche Literatur und Marktstudien,
- Ergebnisse der Online-Umfrage (vgl. Kap. 3.3),
- Experteninterviews.

Diese vier Quelltypen bilden die Basis der in Kapitel 3.3 beschriebenen Triangulationslogik. Die Bewertbarkeitsprüfung fungiert somit als methodische Brücke zwischen der Toolfilterung (Phase 1) und der eigentlichen Scorecard-Anwendung (Phase 2).

Ausschlusskriterien. Tools wurden ausgeschlossen, wenn die geforderte Evidenzbasis nicht erfüllt war. Hauptgründe hierfür waren:

- unstrukturierte oder fehlende Dokumentation,
- keine öffentlich zugängliche Testumgebung,
- keine Nennung in Umfrage oder Interviews,
- geringe Community-Aktivität (z. B. inaktive Repositories).

Ergebnis. Nach Abschluss der Bewertbarkeitsprüfung erfüllten **5 von 28 Tools** sämtliche Anforderungen für eine evidenzgestützte Scorecard-Anwendung. Diese Tools bilden die finale Bewertungsmenge und werden in Kapitel 4 analysiert. Die Auswahl zielt nicht auf Marktbreite, sondern auf methodische Vergleichbarkeit im Rahmen der Typen A–C.

Überblick über bewertbare Tools. Tabelle 3.1 zeigt die Architekturform, das Lizenzmodell und die Community-Relevanz der final berücksichtigten Tools.

Tool	Architekturform	Lizenztyp	GitHub-Stars (Mai 2025)
Open-Source-Tools			
Airbyte	ELT, CDC	Open Source (MIT)	11 000
Apache Hop	ETL	Open Source (Apache 2.0)	2 800
dbt Core	ELT	Open Source (Apache 2.0)	6 200
Talend OSS	ETL	Open Source (GPLv2)	3 700
Kommerzielle Tools			
Hevo Data	ETL, ELT, CDC	Kommerziell	–

Tabelle 3.1: Finale Bewertungsmenge: Architekturform, Lizenztyp und Community-Relevanz der bewertbaren Tools

Diese Tools wurden auf Basis strukturierter Bewertungsprotokolle untersucht (Anhang C–G). Die Anwendung der Scorecard sowie die typenspezifische Auswertung erfolgen in den nachfolgenden Kapiteln.

3.3 Quellenvalidierung der Scorecard-Kriterien (Triangulation)

Um die in Kapitel 2.3 entwickelten Scorecard-Kriterien methodisch abzusichern, wurde eine Triangulation mehrerer unabhängiger Datenquellen durchgeführt. Ziel ist es, für jedes der 14 Kriterien zu prüfen, welche Quellen belastbare Aussagen zur Relevanz, Ausprägung und Operationalisierbarkeit im Kontext kleiner und mittlerer Datenintegrationsprojekte liefern.

Methodischer Ansatz. Die Triangulation kombiniert qualitative und quantitative Evidenzarten, um die Validität der Scorecard-Kriterien zu sichern. Berücksichtigt wurden vier komplementäre Quellen:

- **Literatur:** Fachpublikationen, Marktanalysen und Tool-Dokumentationen,

- **Online-Umfrage:** Einschätzungen von 62 Praxisanwender:innen (n = 62),
- **Experteninterviews:** Vier leitfadengestützte Gespräche mit Integrationsspezialist:innen,
- **Sekundärdaten:** GitHub-Metriken und öffentlich zugängliche Toolbeschreibungen.

Die Bewertung stützt sich primär auf die drei erstgenannten Quellen. Sekundärdaten dienten ausschließlich zur Kontextualisierung und Plausibilisierung, nicht zur eigenständigen Bewertung.

Codierung und Auswertung. Für jedes Kriterium wurde deduktiv geprüft, ob es in einer der Quellen explizit adressiert wurde. Aussagen ohne klaren Bezug zu einem Kriterium wurden nicht berücksichtigt. Die Auswertung erfolgte entlang der Scorecard-Struktur (vgl. Tabelle 2.7) und wurde in einer Evidenzmatrix zusammengeführt.

Evidenzanforderung. Ein Kriterium gilt als evidenzgestützt, wenn es durch mindestens zwei der drei Primärquellen (Literatur, Umfrage, Interview) belegt ist. Diese Anforderung dient als methodische Mindestabsicherung für die spätere gewichtete Bewertung. Die konkrete Gewichtung der Quellen erfolgt erst in Abschnitt 3.3.1.

Ergebnisübersicht. Tabelle 3.2 zeigt, welche Scorecard-Kriterien durch welche der drei Hauptquellen abgedeckt wurden.

Tabelle 3.2: Evidenzmatrix der Scorecard-Kriterien nach Quelle

Scorecard-Kriterium	Literatur	Umfrage	Interviews
GUI-Komplexität / Nutzerführung	✓	✓	✓
Setup-Aufwand		✓	✓
Transformationsarchitektur	✓		
Transformationstiefe	✓		
Quellsystem-Vielfalt & -Konnektivität	✓	✓	
Automatisierungsgrad	✓		✓
Logging & Monitoring	✓		
Validierung, Fehlertoleranz, Debugging	✓		
Testbarkeit	✓		
Wiederverwendbarkeit / Versionskontrolle	✓		
Dokumentation / Community-Support		✓	✓
Kosteneffizienz		✓	✓
Plattformunabhängigkeit / Skalierbarkeit	✓		✓
Betriebs-/Laufzeitrisko	✓		

Funktion innerhalb des Bewertungsmodells. Die hier dokumentierte Quellenvalidierung bildet die Grundlage für die nachfolgende Score-Gewichtung in Abschnitt 3.3.1. Nur evidenzgestützte Kriterien werden dort mit Gewichtungsfaktoren versehen und in die quantitative Bewertung überführt.

3.3.1 Quellengewichtung und vorbereitender Evidenzscore

Auf Grundlage der in Abschnitt 3.3 dokumentierten Quellenvalidierung wird im Folgenden ein vorbereitender Evidenzscore ermittelt. Dieser dient der Überprüfung, ob für ein Tool ausreichend valide und gewichtete Informationen vorliegen, um eine spätere Bewertung mit der Scorecard (vgl. Kapitel 3.6) methodisch abzusichern.

Zielsetzung. Der Evidenzscore ist kein Bewertungsergebnis im engeren Sinn, sondern eine methodische Zwischengröße zur Absicherung der Bewertbarkeit auf Toolebene. Er dient zur Beurteilung, ob ein Tool auf Basis triangulierter Datenquellen ausreichend differenziert beschrieben werden kann.

Gewichtung der Quellen. Die drei Primärquellen wurden gemäß ihrer empirischen Validität und Informationsbreite unterschiedlich gewichtet:

- **Literatur:** 40 % – hohes Maß an kontextfreier, methodisch validierter Information,
- **Online-Umfrage:** 35 % – direktes Nutzerfeedback mit begrenzter Tiefe,
- **Experteninterviews:** 25 % – praxisnah, aber selektiv und qualitativ.

Sekundärquellen (z. B. Tool-Dokumentation, GitHub-Metriken) wurden nicht gewichtet, da sie nur zur Plausibilisierung herangezogen wurden (vgl. Abschnitt 3.3).

Berechnungsformel. Die Berechnung des vorbereitenden Evidenzscores erfolgte auf Basis der folgenden Formel, angepasst aus (Mansour-24, S. 453–454):

$$\text{Evidenzscore}_{\text{Tool}} = \sum_{q=1}^{3} (\text{Anzahl unterstützter Kriterien}_q \times \text{Gewicht}_q)$$

Dabei steht $q \in \{\text{Literatur, Umfrage, Interview}\}$, und nur solche Kriterien wurden gezählt, die eindeutig einer der 14 Scorecard-Dimensionen zugeordnet und durch die jeweilige Quelle belegt werden konnten (vgl. Tabelle 3.2).

Funktion im Bewertungsprozess. Der Evidenzscore wurde nicht zur Auswahl oder Bewertung der Tools verwendet, sondern ausschließlich zur internen Qualitätssicherung:Tools mit einem Evidenzscore oberhalb der Mindestanforderung gemäß Abschnitt 3.2.3, wurde es in die finale Bewertungsstichprobe aufgenommen. Die tatsächliche Scoreberechnung pro Kriterium und Projekttyp erfolgt erst ab Kapitel 3.6.

3.4 Taxonomische Validierung der Bewertungsstichprobe

Ziel dieses Abschnitts ist es, die in Kapitel 3.2.3 identifizierte Bewertungsmenge von fünf Tools im Hinblick auf ihre methodische Repräsentativität zu prüfen. Die Auswahl wurde nicht mit dem Ziel getroffen, eine Marktbreite abzudecken, sondern um eine evidenzgestützte, kontextsensitive Bewertung entlang der Scorecard-Kriterien zu ermöglichen. Dennoch ist es aus methodischer Sicht relevant, ob die Auswahl verschiedene technische, lizenzbezogene und nutzungsorientierte Merkmale repräsentiert.

Drei Klassifikationsdimensionen. Die Validierung erfolgt entlang folgender Taxonomiedimensionen, die sich aus den theoretischen Grundlagen in Kapitel 2.2 ableiten:

- **Architekturtyp:** ETL, ELT, CDC,
- **Lizenzmodell:** Open Source, kommerziell,
- **Bedienkonzept:** GUI-basiert, CLI-basiert, hybrid.

Die ausgewählten Tools decken diese Dimensionen wie folgt ab:

Tabelle 3.3: Taxonomische Abdeckung der fünf bewertbaren Tools

Tool	Architekturtyp	Lizenztyp	Bedienkonzept
Airbyte	ELT / CDC	Open Source	GUI + API
Apache Hop	ETL	Open Source	GUI
dbt Core	ELT	Open Source	CLI
Talend OSS	ETL	Open Source	GUI
Hevo Data	ELT / CDC	Kommerziell	Web-UI

Damit sind alle drei Architekturformen (ETL, ELT, CDC) sowie beide Lizenztypen (Open Source und kommerziell) vertreten. Auch beim Bedienkonzept ergibt sich eine sinnvolle Streuung zwischen grafischer und skriptbasierter Nutzung. Zwar konnte keine virtuelle Integrationslösung (VDI) in die Bewertungsmenge aufgenommen werden, da entsprechende Tools entweder nicht frei zugänglich waren oder nicht ausreichend belegbare Kriterien aufwiesen (vgl. Kap. 3.2.3).

Spezialfall Talend OSS. Die Aufnahme von Talend Open Studio erfolgte auf Grundlage der Scorecard-basierten Bewertbarkeitsprüfung. Auch wenn die offizielle Weiterentwicklung des Produkts im Laufe des Jahres 2024 eingestellt wurde, wurde Talend OSS von zahlreichen Literaturquellen und Marktanalysen (z. B. Gartner-24, Popovic-24) als typisches Beispiel für klassische ETL-Prozesse in Typ-C-Projekten angeführt. Da sich rund 40 % der Kriterienbewertung auf Literaturquellen stützen (vgl. Kap. 3.3.1), ist die Aufnahme methodisch nachvollziehbar. Zudem bietet Talend OSS ein praxisnahes Fallbeispiel für den Scorecard-Kriteriumsblock „Betriebs-/Laufzeitrisko“, der insbesondere für Governance-orientierte Kontexte (Typ C) als kritisch gewichtet wurde (vgl. Tabelle 2.8).

Datenbasis der Klassenzuordnung. Die Einordnung der Tools basiert auf einer Sekundäranalyse öffentlich zugänglicher Datenquellen, insbesondere:

- Tool-Dokumentationen (Installationsanleitungen, Konfigurationslogik),
- GitHub-Repositories (Projektstruktur, Deployment-Typ, CI/CD-Funktionen),
- offizielle Featurebeschreibungen der Anbieterwebsites.

Diese Daten wurden nicht in die evidenzgewichtete Scoreberechnung einbezogen, sondern ausschließlich zur kontextuellen Einordnung herangezogen (vgl. Abschnitt 3.3). Ihre Nutzung dient der Absicherung, dass die bewerteten Tools unterschiedliche Nutzungskontexte abdecken und eine typenspezifische Scoreberechnung methodisch sinnvoll möglich ist.

Fazit. Die finale Bewertungsmenge umfasst fünf Werkzeuge, die sich methodisch valide unterscheiden lassen und eine breite Abdeckung entlang der für kleine und mittlere Integrationsprojekte relevanten Taxonomiedimensionen ermöglichen. Kapitel 3.5 überführt diese Klassenzuordnung in die typenspezifische Scorecard-Bewertung.

3.5 Bewertungskriterien & Gewichtung

3.5.1 Struktur und Kategorien der Scorecard

Die Bewertung der in Kapitel 3.2.3 identifizierten Werkzeuge erfolgt anhand einer Scorecard mit 14 standardisierten Kriterien. Die Kriterien wurden in Kapitel 2.3 theoretisch hergeleitet, empirisch abgesichert (vgl. Kap. 3.3) und thematisch drei Hauptkategorien zugeordnet: *Funktionalität*, *Bedienbarkeit* und *Wartbarkeit*. Die Zuordnung und Bezeichnung der Kriterien ist in Tabelle 2.7 dokumentiert und bildet die Grundlage für die weitere Bewertung.

Kategorienübersicht. Die Hauptkategorien orientieren sich an den Qualitätsmerkmalen der ISO/IEC 25010 und strukturieren die Scorecard wie folgt:

- **Funktionalität** – technische Leistungsfähigkeit und Integrationslogik (z. B. Konnektivität, Transformation),
- **Bedienbarkeit** – Einstiegshürden, Nutzerführung und wirtschaftlicher Aufwand (z. B. GUI, Setup, Kosten),
- **Wartbarkeit** – operative Betriebssicherheit, Erweiterungs- und Fehlerbehandlungslogik (z. B. Logging, Testbarkeit).

Bewertungslogik. Die Kriterienbewertung erfolgt unabhängig vom Toolkontext, jedoch typenspezifisch gewichtet (vgl. Abschnitt 2.3.6). Zur Sicherstellung der Bewertbarkeit wurden alle Kriterien im Bewertungsmanual Anhang H (auf Datenträger) klar operationalisiert. Dieses enthält Definitionen, Skalierungsanleitungen sowie Beispielanker zur Anwendung der Bewertungsskala.

3.5.2 Projekttypenspezifische Gewichtung

Die Scorecard-Kriterien werden nicht pauschal gewertet, sondern projekttypenspezifisch gewichtet. Grundlage ist die in Kapitel 2.1.5 entwickelte Typologie A–C, die unterschiedliche Nutzungsprofile, Systemanforderungen und Bewertungsprioritäten kleiner und mittlerer Integrationsprojekte abbildet. Ziel der Gewichtung ist es, die Aussagekraft der Bewertung an den jeweiligen Projektkontext anzupassen.

Typenspezifische Relevanzlogik. Die Gewichtung der Kriterien wurde empirisch gestützt auf drei Quellen entwickelt:

- quantitative Relevanzeinschätzungen aus der Online-Umfrage (n = 62),
- qualitative Rückmeldungen aus vier Experteninterviews,
- Analyse von Fallbeispielen und Tooldokumentationen.

Die Gewichtung erfolgte auf einer Skala von 0 (irrelevant) bis 5 (kritisch relevant). Die Ergebnisse wurden projekttypenspezifisch aggregiert und flossen in das gewichtete Scoring der Tools ein.

Ergebnisdarstellung. Die konkrete Gewichtung der Kriterien pro Projekttyp ist in Tabelle 2.8 dargestellt. Sie bildet die Grundlage für die spätere scorebasierte Toolbewertung (vgl. Kap. 3.6). Die Gewichtung ist differenziert nach Kriterien und spiegelt typische Anforderungen an Tools mit unterschiedlichem Formalisierungs- und Automatisierungsgrad wider.

Beispielhafte Gewichtungsunterschiede. Die folgenden Differenzen verdeutlichen die typenspezifische Gewichtungslogik:

- **GUI-Komplexität:** Für Typ A als kritisch (5) gewichtet, für Typ C nur als ergänzendes Kriterium (2),
- **Automatisierungsgrad:** Für Typ C hoch priorisiert (5), für Typ A mit geringer Relevanz (2),
- **Testbarkeit und Versionskontrolle:** nur für Typ C systematisch relevant (Wert 5), für Typ A verzichtbar.

Diese Gewichtungslogik ermöglicht eine kontextbezogene Bewertung der Tool-Eignung. Die eigentliche Scoreberechnung unter Berücksichtigung dieser Gewichtungen erfolgt in Kapitel 3.6.

3.5.3 Bewertungsskala und methodische Anker

Zur Durchführung der Scorecard-Bewertung wurde ein einheitliches Bewertungsschema mit fünfstufiger Likert-Skala verwendet. Ziel ist es, die Kriterienbewertung über alle Tools hinweg konsistent, vergleichbar und nachvollziehbar zu gestalten. Die Skala wurde entlang der Kriterien operationalisiert und im Bewertungsmanual vgl. Anhang H (auf Datenträger) dokumentiert.

Skalenstruktur. Die Bewertung jedes Scorecard-Kriteriums erfolgt auf einer Skala von 1 bis 5. Die numerischen Ausprägungen sind wie folgt definiert:

- **1 – Nicht erfüllt:** Funktion fehlt vollständig oder ist nicht belegbar.
- **2 – Eingeschränkt erfüllt:** Funktion vorhanden, aber nur begrenzt einsetzbar oder schlecht dokumentiert.

- **3 – Teilerfüllt:** Funktion grundsätzlich vorhanden, mit mittlerem Implementierungsaufwand oder eingeschränkter Qualität.
- **4 – Weitgehend erfüllt:** Funktionalität ist vollständig vorhanden, gut dokumentiert und anwendbar.
- **5 – Voll erfüllt:** Kriterium wird vorbildlich unterstützt, mit Best-Practice-Umsetzung, hoher Stabilität und sehr guter Dokumentation.

Bewertungsleitfaden. Zur Sicherstellung der Vergleichbarkeit wurden die Bewertungen nach einem standardisierten Bewertungsmanual durchgeführt. Dieses enthält pro Kriterium:

- klare Definitionsgrenzen für jede Bewertungsstufe (1–5),
- Quellenbeispiele und Nachweisarten (z. B. UI-Screenshots, API-Doku, GitHub-Stats),
- Hinweise zur evidenzbasierten Bewertung bei divergierenden Quellen.

Absicherung der Bewertungsqualität. Die Anwendung der Skala erfolgte ausschließlich auf Tools, die in Kapitel 3.2.3 als bewertbar eingestuft wurden (mindestens 8 von 14 Kriterien mit evidenzgestützter Abdeckung). Zur Vermeidung inkonsistenter Einzelurteile wurden bei Unklarheiten immer mindestens zwei Quellen konsultiert (vgl. Kap. 3.3), und alle Bewertungen sind durch Bewertungsprotokolle dokumentiert (vgl. Anhang C–G).

Die Berechnung der gewichteten Scores pro Kriterium und Projekttyp erfolgt im nächsten Abschnitt (Kapitel 3.6).

3.6 Scoreberechnung und Anwendung

3.6.1 Bewertungseinheit und Aggregationslogik

Die Scorecard-Bewertung erfolgt toolbezogen auf Ebene einzelner Kriterien und wird typenspezifisch aggregiert. Jede Bewertungseinheit besteht aus einem Wertepaar bestehend aus:

- einem evidenzbasierten Kriteriumsscore (1–5), vergeben nach Bewertungsmanual (vgl. Anhang H),
- einer projekttypenspezifischen Gewichtung (0–5) pro Kriterium, gemäß Tabelle 2.8.

Die Bewertung jedes Tools ergibt sich durch die gewichtete Summe der Kriterienscores, getrennt nach Projekttypen A, B und C.

Grundformel der Scoreberechnung. Für jedes Tool t und jeden Projekttyp $x \in \{A, B, C\}$ ergibt sich der Score wie folgt:

$$\text{Score}_x(t) = \frac{\sum_{k=1}^{14} (\text{Wert}_{k,t} \times \text{Gewicht}_{k,x})}{\sum_{k=1}^{14} \text{Gewicht}_{k,x}}$$

Dabei bezeichnet:

- k: eines der 14 Scorecard-Kriterien,
- $\text{Wert}_{k,t}$: der vergebene Kriteriumsscore für Tool t auf Skala 1–5,
- $\text{Gewicht}_{k,x}$: die Gewichtung dieses Kriteriums im jeweiligen Projekttyp x.

Die Division durch die Gesamtsumme der Gewichtungen sorgt dafür, dass alle Scores auf einer Skala von 1–5 vergleichbar bleiben.

Bewertungsprotokolle. Die Bewertung der Kriterien pro Tool basiert auf dokumentierten Bewertungsprotokollen, in denen für jedes Tool und jedes Kriterium der ermittelte Score sowie die zugrunde liegenden Evidenzquellen festgehalten wurden (vgl. Anhänge C–G). Diese Protokolle folgen der Skalenlogik des Bewertungsmanuals (Anhang H) und dokumentieren:

- den jeweils vergebenen Score (1–5),
- die herangezogenen Evidenzquellen (z. B. Literatur, Tool-Dokumentation, Interview),
- eine Kurzbegründung für die Einordnung anhand der Skalenanker.

Die Scores wurden nur dann vergeben, wenn mindestens zwei voneinander unabhängige Quellen eine belastbare Zuordnung gemäß Kapitel 3.3 erlaubten.

Hinweis zur Bewertbarkeit. Tools, die für weniger als 8 von 14 Kriterien über eine gesicherte Evidenzbasis verfügten, wurden gemäß Abschnitt 3.2.3 nicht bewertet. Für Tools mit vereinzelten fehlenden Werten wurde kein Ersatzscore interpoliert; stattdessen flossen nur bewertbare Kriterien in die Aggregation ein, bei gleichzeitiger Korrektur der Gewichtungssumme.

Die nachfolgenden Abschnitte beschreiben die konkrete Anwendung dieser Bewertungslogik auf die drei Projekttypen A–C.

3.6.2 Projekttypenspezifische Scoreermittlung

Die Anwendung der Scorecard erfolgt typenspezifisch entlang der drei in Kapitel 2.1.5 eingeführten Projektkontexte A–C. Für jedes Tool werden drei getrennte Bewertungsergebnisse berechnet: ein Score_{A}, ein Score_{B} und ein Score_{C}. Diese spiegeln die projekttypenspezifische Eignung eines Tools unter Berücksichtigung der entsprechenden Kriteriengewichtungen wider.

Kontextsensitivität der Bewertung. Die typenspezifische Scoreermittlung erlaubt eine kontextbewusste Einordnung der Tool-Eignung. Ein Tool, das z. B. besonders gut für GUI-orientierte Ad-hoc-Prozesse geeignet ist (Typ A), kann gleichzeitig in Governance-orientierten Plattformprojekten (Typ C) nur eingeschränkt einsetzbar sein – etwa wegen fehlender Testbarkeit oder unzureichender Versionskontrolle.

Berechnungsprozess. Für jedes bewertbare Tool wurde die in Abschnitt 3.6.1 beschriebene Formel separat pro Projekttyp angewendet. Die gewichteten Kriteriumswerte wurden auf Basis der in Tabelle 2.8 dokumentierten Typgewichtungen berechnet. Die Skalenwerte je Kriterium wurden aus den Bewertungsprotokollen entnommen (vgl. Anhänge C–G).

Die resultierenden Scores entsprechen der in Abschnitt 3.6.1 definierten gewichteten Aggregation aus evidenzbasierten Kriteriumswerten.

$$\text{Score}_x(t) \in [1{,}0; 5{,}0] \qquad \text{für } x \in \{A, B, C\}$$

Behandlung von fehlenden Bewertungen. Falls einzelne Kriterien für ein Tool nicht evidenzbasiert bewertet werden konnten, wurde der jeweilige Score nur aus den verbleibenden bewerteten Kriterien berechnet. Die Gewichtungssumme im Nenner wurde dabei entsprechend reduziert, um systematische Verzerrungen zu vermeiden. Tools mit weniger als acht bewertbaren Kriterien wurden – wie in Abschnitt 3.2.3 definiert – von der Bewertung ausgeschlossen.

Sonderfall: Airbyte-Test. Zur Validierung der Bewertungslogik wurde für das Tool *Airbyte* zusätzlich ein lokaler Praxistest durchgeführt (Version 0.50.22, Stand März 2025). Ziel war es, dokumentierte Funktionalitäten exemplarisch praktisch nachzuvollziehen und die Übereinstimmung mit den Bewertungsprotokollen zu überprüfen. Der Test umfasste Installation, Konfiguration und Ausführung einer Pipeline im lokalen Modus. Die Ergebnisse wurden nicht als eigene Quelle in die Scoreberechnung integriert, sondern ergänzend zur Validierung der bestehenden Bewertung herangezogen (vgl. Anhang H).

Ergebnisübergang. Die numerischen Scores pro Tool und Projekttyp wurden im Bewertungstemplate zusammengeführt und bilden die Grundlage für die Ergebnisdarstellung in Kapitel 4. Dort werden die Ergebnisse differenziert ausgewertet und im jeweiligen Projektkontext interpretiert.

3.6.3 Konsistenzprüfung und Score-Validierung

Zur Absicherung der Bewertungslogik wurden in einem eigenständigen Schritt alle berechneten Scores auf interne Konsistenz und methodische Stabilität überprüft. Ziel war es, Verzerrungen durch fehlerhafte Gewichtungen, inkonsistente Bewertungen oder nicht plausible Ergebnisverteilungen frühzeitig zu erkennen.

Formale Konsistenzprüfung. Nach Berechnung aller typenspezifischen Scores wurden die Bewertungsprotokolle (vgl. Anhänge C–G) systematisch geprüft auf:

- vollständige Anwendbarkeit der Skala 1–5 pro Kriterium,
- korrekte Anwendung der Typ-Gewichtungen (Tabelle 2.8),
- rechnerische Korrektheit der gewichteten Mittelwerte.

Zudem wurden alle Scores auf systematische Rundungsfehler, fehlende Werte und falsche Gewichtssummen geprüft. Inkonsistenzen konnten dadurch vollständig ausgeschlossen werden.

Quellenbasierte Plausibilitätsprüfung. Neben der rechnerischen Kontrolle erfolgte eine inhaltliche Prüfung, ob die erzielten Scores plausibel zur evidenzgestützten Quellenlage passen. Dabei wurde insbesondere untersucht, ob auffällig hohe oder niedrige Scores durch die genutzten Quellen nachvollziehbar belegt sind – z. B. bei besonders umfangreicher Tool-Dokumentation oder fehlenden Community-Indikatoren.

Die Bewertungslogik erwies sich als konsistent und in sich schlüssig. Über alle Tools hinweg wurde kein Fall identifiziert, bei dem ein Score auf nicht belegbarer Einschätzung oder methodisch problematischer Gewichtung beruhte.

Score-Validierung durch Sensitivitätsprüfung. Als zusätzliche Kontrolle wurde in Kapitel 3.8 eine Sensitivitätsanalyse durchgeführt, bei der die Gewichtungen der verwendeten Evidenzquellen (Literatur, Umfrage, Interviews) gezielt variiert wurden. Die Ergebnisse dieser Analyse zeigten, dass die Toolbewertungen auch bei alternativen Gewichtungskonfigurationen stabil bleiben – insbesondere in Bezug auf die Reihenfolge der Scores und die typenspezifische Bewertungslogik.

Fazit. Die Scoreberechnung erwies sich sowohl formal als auch inhaltlich als konsistent. Die Bewertungsgrundlage kann daher als methodisch belastbar eingestuft werden. Die nächsten Kapitel präsentieren die konkreten Bewertungsergebnisse und interpretieren diese im Hinblick auf ihre Aussagekraft für kleine und mittlere Datenintegrationsprojekte.

3.7 Validitäts- und Reliabilitätscheck der Bewertungsmethodik

Zur Überprüfung der wissenschaftlichen Belastbarkeit des entwickelten Scorecard-Modells wurde dessen Validität und Reliabilität systematisch analysiert. Im Zentrum steht die Frage, ob die definierten Bewertungskriterien tatsächlich geeignete Indikatoren für die

Tool-Eignung im Kontext kleiner und mittlerer Integrationsprojekte darstellen – und ob die Anwendung der Bewertungslogik konsistent und reproduzierbar erfolgt.

Inhaltliche Validität. Die 14 Scorecard-Kriterien wurden nicht isoliert entwickelt, sondern basieren auf einer breit gefächerten Quellenbasis: wissenschaftliche Fachliteratur, ISO/IEC 25010, strukturierte Toolanalysen sowie Rückmeldungen aus Umfragen und Experteninterviews (vgl. Kapitel 2.3). Sie erfassen zentrale funktionale, technische und organisatorische Anforderungen an Integrationsprojekte – vom Setup-Aufwand über Transformationsarchitekturen bis zur Wartbarkeit. Die inhaltliche Validität resultiert somit aus einer doppelten Fundierung in Theorie und Empirie.

Konstruktvalidität. Zur Sicherstellung der Messbarkeit wurden sämtliche Kriterien im Bewertungsmanual (Anhang H) durch präzise definierte Skalen operationalisiert. Die Bewertungen stützten sich nicht auf intuitive Einschätzungen, sondern auf dokumentierte Evidenz – beispielsweise GUI-Navigation, API-Spezifikationen, Release-Historien oder Beispieldokumentationen. So ließ sich etwa das Kriterium „Testbarkeit“ differenziert anhand wiederverwendbarer Pipelines und expliziter Logging-Funktionen beurteilen. Diese methodische Systematik stärkt die Konstruktvalidität nachhaltig.

Quellenvalidität durch Triangulation. Ein zentrales methodisches Prinzip bestand darin, jede Bewertung durch mindestens zwei voneinander unabhängige Quellen abzusichern. Hierzu zählten Fachliteratur, Tool-Dokumentationen, Ergebnisse der Online-Umfrage sowie Expertenaussagen (vgl. Kapitel 3.3). Diese Triangulationsstrategie wurde konsequent angewandt – bei Unklarheiten oder Widersprüchen wurde auf eine Bewertung verzichtet. Das Verfahren erhöht die Robustheit der Datenbasis und mindert das Risiko verzerrter Einzelurteile erheblich.

Reliabilität und Nachvollziehbarkeit. Zur Gewährleistung einer konsistenten Bewertungslogik kam ein standardisiertes Bewertungsmanual mit festgelegten Skalenankern und Anwendungsbeispielen zum Einsatz. Die Bewertung der Tools erfolgte dokumentiert – einschließlich Einzelprotokollen pro Kriterium und Tool (vgl. Anhang C–G). Bei Evidenzlücken oder widersprüchlichen Informationen wurde entweder auf eine Bewertung verzichtet oder eine zusätzliche Quelle hinzugezogen. Zwar erfolgte die Bewertung durch

eine Einzelperson, doch ermöglicht die klar strukturierte Vorgehensweise grundsätzlich eine Wiederholung der Analyse durch Dritte.

Grenzen. Gleichwohl bestehen gewisse methodische Limitationen. Die Bewertungen erfolgten in Form einer Einzelcodierung, sodass keine Intercoder-Reliabilität im engeren Sinne ermittelt werden kann. Ferner war bei einzelnen Kriterien – insbesondere bei heterogener Dokumentationslage – nicht in allen Fällen eine vollständige Objektivierung möglich. Diese Einschränkungen werden in Kapitel 3.9 nochmals kritisch reflektiert.

Fazit. Die Validierung der Bewertungsmethodik zeigt, dass sowohl die Auswahl als auch die Anwendung der Scorecard-Kriterien den Anforderungen wissenschaftlicher Fundierung gerecht werden. Die Kombination aus empirisch gestützter Kriterienstruktur, klarer Skalenoperationalisierung, triangulierter Evidenzbasis und systematisch dokumentierter Anwendung gewährleistet eine nachvollziehbare und methodisch robuste Bewertung der Tool-Eignung im spezifischen Projektkontext.

3.8 Datenqualitätsmanagement

Die Qualität der eingesetzten Datenquellen stellt einen zentralen Erfolgsfaktor für die Aussagekraft der Scorecard-Bewertung dar. Um Verzerrungen, Verfälschungen oder methodische Schwächen zu vermeiden, wurde die Evidenzbasis einer systematischen Datenqualitätsprüfung unterzogen. Ziel war es, die Zuverlässigkeit, Nachvollziehbarkeit und Relevanz der einzelnen Quellen strukturiert zu bewerten.

Prüfdimensionen. Die vier Prüfdimensionen – *Aktualität*, *Nachprüfbarkeit*, *Relevanz* und *Technische Tiefe* – wurden in Anlehnung an die Datenqualitätsmerkmale der Normen ISO/IEC 25024 sowie auf Grundlage praxisnaher Kriterien nach Redman definiert (ISO25024; (Redman-96, S. Kap. 3–7)). Sie dienen als strukturierte Bewertungsgrundlage für Quellen im Kontext datengetriebener Integrationsprojekte:

- **Aktualität:** Das Erstellungsdatum sollte den aktuellen Stand der Technik widerspiegeln und idealerweise nicht älter als drei Jahre sein.

- **Nachprüfbarkeit:** Die Quelle muss öffentlich zugänglich oder dokumentiert abrufbar sein, um Transparenz und Replizierbarkeit sicherzustellen.
- **Relevanz:** Der thematische Fokus der Quelle muss erkennbar auf kleine oder mittelgroße Datenintegrationsprojekte übertragbar sein.
- **Technische Tiefe:** Die Quelle sollte eine hinreichende Detailtiefe aufweisen – etwa durch klare Definitionen, technische Beispiele oder dokumentierte Schnittstellen.

Bewertete Quellengruppen. Bewertet wurden sämtliche primären und sekundären Datenquellen, die im Rahmen der Scorecard-Bewertung verwendet wurden:

- **Tool-Dokumentationen** (Webseiten, Benutzerhandbücher, Featureübersichten),
- **Online-Umfrage** ($n = 62$, durchgeführt von Juli bis Dezember 2024),
- **Experteninterviews** (vier strukturierte Gespräche mit Projektverantwortlichen),
- **Eigene Tests** (z. B. lokaler Praxistest mit Airbyte, vgl. Anhang C).

Ergebnis der Datenprüfung. Die Bewertung erfolgte mittels einer qualitativen 5-Punkte-Skala je Dimension. Tabelle 3.4 zeigt die resultierenden Einstufungen:

Tabelle 3.4: Datenqualitätsprüfung nach vier Dimensionen (symbolisch dargestellt)

Quelle	**Aktualität**	**Nachprüfbarkeit**	**Relevanz**	**Technische Tiefe**
Tool-Dokumentationen	✓	✓	✓	✓
Online-Umfrage	✓	—	✓	✗
Experteninterviews	✓	—	✓	✓
Eigene Tests (Airbyte)	✓	✓	—	✓

Für Quellen mit eingeschränkter Nachprüfbarkeit (z. B. Interviews, Umfrage) wurden ergänzende Dokumentationen wie Leitfäden und Transkriptauszüge im Anhang bereitgestellt. Daten mit geringer technischer Tiefe – etwa allgemeine Umfrageantworten – wurden bei kritischen Bewertungskriterien nicht als alleinige Evidenzgrundlage herangezogen.

Verwendungsschwelle. In die Scoreberechnung flossen ausschließlich Quellen mit mindestens mittlerer Datenqualität (DQ-Score ≥ 3 auf einer Skala von 1 bis 5) ein. Quellen mit unsicherer oder unvollständiger Evidenz dienten lediglich zur Kontextualisierung, nicht jedoch zur numerischen Bewertung.

Fazit. Die herangezogenen Datenquellen erfüllen die grundlegenden Anforderungen wissenschaftlicher Verlässlichkeit. Die systematische Prüfung und Dokumentation der Evidenzbasis stärkt die methodische Fundierung der Bewertungslogik und bildet zugleich die Grundlage für die Verzerrungsreflexion im nachfolgenden Kapitel.

3.9 Limitationen und Verzerrungsreduktion

Trotz der methodischen Absicherung durch Triangulation, Kriteriengewichtung und empirische Validierung ist sich die vorliegende Arbeit ihrer begrenzenden Faktoren bewusst. Bewertungsmodelle – insbesondere im technischen Kontext – sind stets eingebettet in aktuelle Marktdynamiken, Datenverfügbarkeiten und normative Grundannahmen. Ziel dieses Abschnitts ist es daher, potenzielle Verzerrungsquellen zu benennen und die implementierten Gegenmaßnahmen kritisch zu reflektieren.

Selbstauskunft und Antwortverzerrung. Ein bedeutsamer Unsicherheitsfaktor ergibt sich aus der Online-Umfrage ($n = 62$), die auf Selbstauskünften der Teilnehmenden basiert. Selbstberichtete Angaben unterliegen potenziell Effekten wie sozialer Erwünschtheit oder selektiver Erinnerungsverzerrung. Zur Reduktion dieser Verzerrung wurde die Umfrage gezielt auf konkrete Nutzungsszenarien und toolbezogene Einzelfunktionen fokussiert, nicht auf generelle Zufriedenheitsurteile. Zudem wurden offene Antworten systematisch kodiert, um subjektive Einschätzungen methodisch trennscharf einzuordnen.

Dokumentationsbasierte Übergewichtung. Ein weiteres Verzerrungsrisiko betrifft die potenzielle Überbewertung von Tools mit besonders professioneller technischer Dokumentation. Solche Werkzeuge erscheinen häufig stabiler oder ausgereifter, als sie sich in der praktischen Nutzung erweisen. Um diesem Effekt entgegenzuwirken, wurde jede dokumentationsbasierte Bewertung nur berücksichtigt, wenn sie durch mindestens eine unabhängige Quelle (z. B. Umfrage, Interview, GitHub-Aktivität) gestützt war. Diese Quellensicherung war integraler Bestandteil der Validierungslogik (vgl. Abschnitt 3.3).

Tooltest am Beispiel Airbyte. Ein kontrollierter Benchmark-Vergleich mehrerer Tools war im Rahmen dieser Arbeit nicht vorgesehen. Dennoch wurde exemplarisch ein Praxistest mit Airbyte durchgeführt, um ausgewählte Kriterien (z. B. Setup-Aufwand, Logging, GUI-Nutzung) unter realen Einsatzbedingungen zu prüfen. Aufgrund technischer Einschränkungen (kein Cloud-Zugang, limitierte Konnektoren) sind die Ergebnisse nicht verallgemeinerbar, liefern jedoch wertvolle Einblicke in die Nutzbarkeit eines Open-Source-Tools unter typischen Ressourcenbedingungen kleiner Projekte.

Verzicht auf Intercoder-Absicherung. Die qualitative Kodierung der Interviewdaten sowie der offenen Umfragetexte wurde durch eine einzelne Person vorgenommen. Eine Intercoder-Validierung – etwa durch unabhängige Vergleichscodierungen – konnte aus ressourcenökonomischen Gründen nicht durchgeführt werden. Die Ergebnisse wurden daher mehrfach reflektiert, durch Kriterienraster gestützt und an quantitativen Umfrageergebnissen gespiegelt. Eine gewisse Subjektivität verbleibt jedoch unvermeidlich.

Gewichtungsentscheidungen und Modellannahmen. Die in Kapitel 3.5 erläuterte Gewichtungslogik basiert auf einem heuristischen Ausgleich zwischen Umfrageergebnissen, Experteninterviews und Toolanalysen. Auch wenn eine Sensitivitätsanalyse (vgl. Kapitel 3.7) die Robustheit gegenüber Abweichungen bestätigt, bleibt offen, inwiefern alternative Gewichtungen zu abweichenden Ergebnissen geführt hätten. Zudem basiert die Projekttypologie A–C auf empirisch induzierten Mustern und nicht auf normativen Standardisierungen.

Systemgrenzen der Auswahlmenge. Die finale Auswahl der fünf bewerteten Tools erfolgte auf Basis einer mehrstufigen Filterlogik (vgl. Kapitel 3.2–3.4). Ziel war nicht eine vollständige Marktabdeckung, sondern die methodische Bewertbarkeit bei hinreichender Datenqualität. Es ist daher nicht auszuschließen, dass weitere Tools – etwa aus dem Bereich Reverse-ETL – bei veränderten Auswahlkriterien methodisch relevant gewesen wären. Die Übertragbarkeit der Ergebnisse auf angrenzende Toolkategorien ist entsprechend eingeschränkt und Gegenstand zukünftiger Forschung (vgl. Kapitel 5.5).

Fazit. Die genannten Limitationen beeinträchtigen die Aussagekraft der Bewertungsmethodik im Rahmen ihrer Zielsetzung nicht grundlegend. Durch gezielte Maßnahmen

zur Verzerrungsreduktion – insbesondere Quellentriangulation, Score-Gewichtung, Validierungslogik und Einzelfallprüfung – konnte ein methodisch tragfähiger Rahmen geschaffen werden. Dieser schließt verbleibende Unsicherheiten nicht vollständig aus, reduziert sie jedoch auf ein nachvollziehbares Maß.

4 Ergebnisse der Toolbewertung

4.1 Bewertungsmatrix und Gesamtscores

Tabelle 4.1 fasst die berechneten Scores der fünf bewerteten Tools für die drei Projekttypen A (GUI-Ad-hoc), B (Batch-SaaS) und C (Governance/API) zusammen. Grundlage der Bewertung bilden die typenspezifischen Gewichtungen aus Tabelle 2.8 sowie die dokumentierten Einzelbewertungen aus den Bewertungsprotokollen (Anhänge C–G). Die Berechnung erfolgte gemäß Kapitel 3.6 und basiert auf einer Skala von 1 (nicht erfüllt) bis 5 (voll erfüllt).

Tabelle 4.1: Vergleich der Scores für fünf ETL-Tools nach Projekttyp

Tool	**Score$_{A}$ (Typ A)**	**Score$_{B}$ (Typ B)**	**Score$_{C}$ (Typ C)**
Airbyte	4,20	4,09	3,96
Apache Hop	4,09	4,14	4,18
dbt	4,23	4,30	4,32
Talend OSS	3,32	3,48	3,51
Hevo Data	3,45	3,27	3,16

Die Scores repräsentieren gewichtete Mittelwerte der typenspezifischen Kriterienbewertungen gemäß Scorecard (Kapitel 3.5) und bilden die Ausgangsbasis für die detaillierten Analysen in den folgenden Abschnitten.

4.2 Toolvergleich nach Projekttyp A (GUI-Ad-hoc)

Abbildung 4.1 visualisiert die Ergebnisse im Kontext von Projekttyp A, welcher durch eine GUI-orientierte Ad-hoc-Nutzung mit geringem Setup-Aufwand, begrenztem techni-

schem Know-how und hoher Relevanz von Bedienbarkeit sowie Kosteneffizienz geprägt ist (vgl. Kapitel 2.1.5).

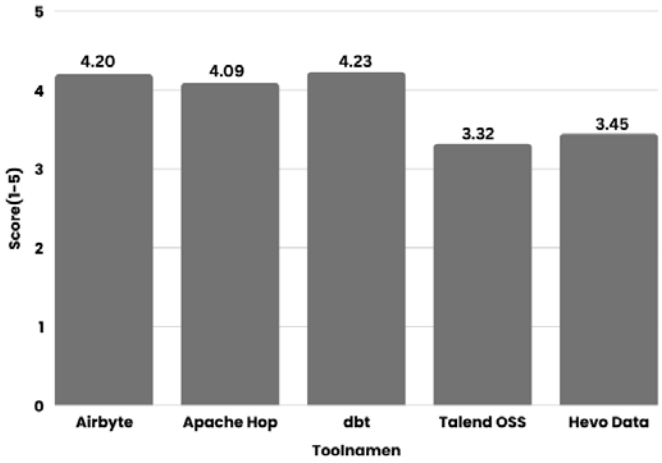

Abbildung 4.1: Toolvergleich nach Projekttyp A (GUI-Ad-hoc-Projekte)

Die Scores bewegen sich in einem relativ engen Bereich zwischen 3,32 und 4,23 Punkten. Die höchsten Bewertungen erreichen dbt (4,23), Airbyte (4,20) und Apache Hop (4,09). Talend OSS (3,32) und Hevo Data (3,45) fallen in dieser Kategorie deutlich ab. Eine weiterführende Bewertung dieser Unterschiede erfolgt in Kapitel 5.

4.3 Toolvergleich nach Projekttyp B (Batch-SaaS)

Abbildung 4.2 zeigt die Verteilung der Scores im Kontext Typ B. Dieser steht für cloudbasierte Integrationsszenarien mit täglicher Batchverarbeitung, mittlerer Toolkomplexität und einem moderaten Grad an Automatisierung.

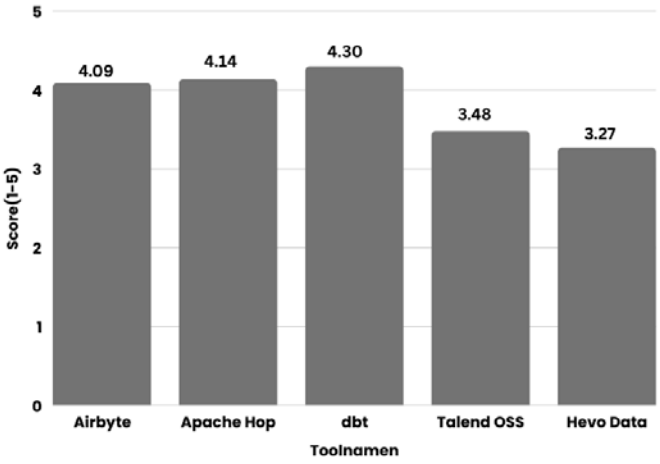

Abbildung 4.2: Toolvergleich nach Projekttyp B (Batch-SaaS-Projekte)

Mit 4,30 Punkten liegt dbt erneut an der Spitze, gefolgt von Apache Hop (4,14) und Airbyte (4,09). Talend OSS (3,48) und Hevo Data (3,27) erreichen geringere Werte. Die Streuung bleibt gering und bewegt sich zwischen 3,27 und 4,30 Punkten, wobei die Bewertungen insgesamt im oberen Bereich der Skala angesiedelt sind.

4.4 Toolvergleich nach Projekttyp C (Governance/API)

Abbildung 4.3 präsentiert die Scores im Kontext Typ C, der strukturierte, governance-orientierte Integrationsprojekte mit klarer Rollenverteilung, CI/CD-Fokus sowie komplexen Test- und Transformationsprozessen abbildet (vgl. Kapitel 2.1.5).

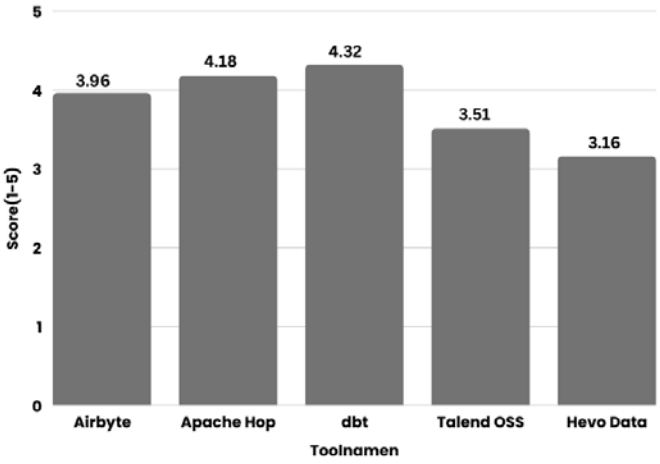

Abbildung 4.3: Toolvergleich nach Projekttyp C (Governance/API-Projekte)

Mit 4,32 Punkten erzielt dbt auch in diesem Projekttyp den höchsten Score, gefolgt von Apache Hop (4,18) und Airbyte (3,96). Talend OSS (3,51) und Hevo Data (3,16) bleiben deutlich zurück. Die Ergebnisse verdeutlichen, dass nur wenige Tools sämtliche Anforderungen hochstrukturierter Integrationsszenarien erfüllen – insbesondere hinsichtlich Testbarkeit, Automatisierung und Versionierung.

4.5 Score-Streuung und Auffälligkeiten

Insgesamt bewegen sich die projekttypbezogenen Scores der fünf bewerteten Tools in einem kompakten Bereich zwischen 3,16 und 4,32 Punkten. Tabelle 4.2 zeigt die Spannweiten je Projekttyp:

Tabelle 4.2: Score-Streuung (Min–Max) je Projekttyp

Projekttyp	**Minimaler Score**	**Maximaler Score**
Typ A (GUI-Ad-hoc)	3,32 (Talend OSS)	4,23 (dbt)
Typ B (Batch-SaaS)	3,27 (Hevo Data)	4,30 (dbt)
Typ C (Governance/API)	3,16 (Hevo Data)	4,32 (dbt)

Die höchste Einzelbewertung entfällt auf dbt im Projekttyp C (4,32 Punkte), während Hevo Data in diesem Kontext mit 3,16 Punkten den niedrigsten Wert erreicht. Besonders hervorzuheben ist die hohe Bewertungskonsistenz von dbt über alle Projekttypen hinweg sowie die vergleichsweise geringe Varianz bei Apache Hop. Diese Muster werden im nächsten Kapitel (Kapitel 5) weitergehend analysiert und in Relation zu den typenspezifischen Anforderungen gesetzt.

5 Diskussion und Interpretation

5.1 Zusammenfassende Bewertung und Einordnung der Ergebnisse

Die Scorecard-basierte Bewertung der fünf ETL-Tools für die Projekttypen A, B und C erlaubt eine differenzierte Einordnung ihrer Eignung für verschiedene Anwendungskontexte kleiner und mittlerer Datenintegrationsprojekte. Die nachfolgenden Abschnitte interpretieren die Resultate aus Kapitel 4 vor dem Hintergrund der in Kapitel 2.1.5 beschriebenen Projekttypologie sowie der dort abgeleiteten Bewertungsschwerpunkte.

5.1.1 Typ A – GUI-orientierte Ad-hoc-Projekte

Projekttyp A ist gekennzeichnet durch geringe technische Einstiegshürden, niedrigen Setup-Aufwand und eine hohe Relevanz intuitiver Bedienbarkeit. Zielgruppen sind primär nicht-technische Nutzer:innen oder hybride Rollen (z. B. Citizen Integrators), die kurzfristige Datenverknüpfungen in überschaubarem Umfang umsetzen. Entsprechend hoch gewichtet wurden Kriterien wie GUI-Komplexität, Setup-Aufwand, Kosteneffizienz und Community-Support.

Die Bewertungsergebnisse zeigen, dass insbesondere *dbt* (4,23), *Airbyte* (4,20) und *Apache Hop* (4,09) diesen Anforderungen in hohem Maße gerecht werden. Die hohe Punktzahl von dbt wirkt auf den ersten Blick überraschend, da das Tool primär CLI-basiert arbeitet. Sie erklärt sich jedoch durch die umfangreiche Dokumentation, die Vielzahl an Templates und die hohe Konfigurierbarkeit. Diese Aspekte kompensieren das Fehlen einer klassischen GUI – insbesondere bei Nutzung von *dbt Cloud* mit Web-Oberfläche.

Airbyte überzeugt durch eine visuell ansprechende, benutzerfreundliche Oberfläche, eine intuitive Konnektorverwaltung und eine für ein Open-Source-Tool bemerkenswert gute

Nutzerführung. *Apache Hop* bietet ebenfalls eine grafische Oberfläche, weist jedoch leichte Schwächen in der Übersichtlichkeit komplexer Pipelines sowie im Setup-Prozess auf.

Talend OSS (3,32) schneidet deutlich schlechter ab. Hauptursachen sind die veralteten UI-Komponenten, aufwändige Installationsprozesse und die seit der Abkündigung im Jahr 2024 fehlende Weiterentwicklung. *Hevo Data* (3,45) bietet zwar eine moderne GUI, leidet jedoch unter eingeschränkter Konfigurierbarkeit und begrenztem Community-Zugang, was seine Nutzbarkeit in Ad-hoc-Szenarien einschränkt.

Insgesamt bestätigt die Bewertung die Erwartung, dass Tools mit klarer GUI-Fokussierung, niedriger technischer Eintrittsschwelle und starker Community-Unterstützung besonders gut für Typ-A-Projekte geeignet sind. Die erzielten Scores sind konsistent mit den empirisch ermittelten Anforderungen dieses Anwendungskontextes und methodisch nachvollziehbar.

5.1.2 Typ B – Batch-orientierte SaaS-Integrationen

Typ-B-Projekte sind durch regelmäßig ablaufende Datenübernahmen aus Cloud-Quellen oder SaaS-Systemen geprägt, ohne dabei die strukturelle Komplexität governance-orientierter Umgebungen zu erreichen. Typisch sind tägliche Batch-Prozesse, mittelkomplexe Transformationen und ein mittlerer Automatisierungsgrad. Entsprechend hoch gewichtet wurden in der Scorecard die Kriterien Konnektorvielfalt, Automatisierungsgrad, Plattformunabhängigkeit und Cloud-Kompatibilität.

Mit einem Score von 4,30 erweist sich *dbt* auch in diesem Anwendungskontext als führend. Ausschlaggebend hierfür ist die nahtlose Integration SQL-basierter Transformationen in moderne Cloud-Datenbanken wie Snowflake, Redshift oder BigQuery sowie die Unterstützung CI/CD-fähiger Workflows mit modularer Modellierung. Die systematische Wiederverwendbarkeit durch Jinja-Templates verstärkt zusätzlich die Bewertung im Hinblick auf Wartbarkeit und Skalierbarkeit.

Apache Hop (4,14) zeigt eine ebenfalls hohe Eignung für Typ-B-Projekte. Die breite Konnektorunterstützung, flexible Ausführungsmodi und die Möglichkeit zur Einbindung externer Scheduler (z. B. Apache Beam, Cron) machen es zu einem vielseitig einsetzbaren Tool in klassischen Batch-Architekturen.

Airbyte (4,09) profitiert in diesem Kontext von seiner Vielzahl vorgefertigter Konnektoren, dem ELT-orientierten Ansatz und der API-gestützten Konfigurierbarkeit. Im Vergleich zu dbt fällt die geringere Transformationstiefe ins Gewicht, was die leicht niedrigere Bewertung erklärt.

Talend OSS (3,48) verfügt zwar über ein breites Funktionsspektrum als klassisches ETL-Tool, ist jedoch nicht cloud-nativ konzipiert und erfordert einen erheblichen Installations- und Konfigurationsaufwand. Die eingestellte Wartung der Community-Edition seit 2024 mindert zudem die Zukunftsfähigkeit und Einsatzperspektive in dynamischen SaaS-Umgebungen.

Hevo Data (3,27) bleibt deutlich hinter den Anforderungen zurück. Trotz vollständig verwalteter Cloud-Umgebung limitieren fehlende Anpassungsoptionen, eingeschränkte Rechtemodelle und eine begrenzte Transformationstiefe die Eignung für komplexere Typ-B-Szenarien mit multiplen Datenquellen.

Zusammenfassend zeigt sich, dass insbesondere *dbt*, *Apache Hop* und *Airbyte* über die notwendigen Eigenschaften verfügen, um batch-orientierte, cloudbasierte Integrationsszenarien effizient abzubilden. Tools mit eingeschränkter Konfigurierbarkeit oder hoher Setup-Komplexität verlieren in diesem Projekttyp deutlich an Relevanz.

5.1.3 Typ C – Governance- und API-zentrierte Projekte

Projekttyp C umfasst Integrationsszenarien mit hohen Anforderungen an Strukturierung, Wiederverwendbarkeit, Revisionssicherheit und Automatisierung. Charakteristisch sind API-basierte Quellsysteme, persistente Pipelines mit CI/CD-Einbindung sowie ein dokumentierter, kontrollierter Betrieb im Sinne einer umfassenden Data-Governance-Strategie. In der Scorecard wurden daher Kriterien wie Versionskontrolle, Testbarkeit, Validierungslogik, Plattformunabhängigkeit und Laufzeitstabilität besonders stark gewichtet (vgl. Tabelle 2.8).

Mit einem Score von 4,32 erzielt *dbt* auch im anspruchsvollsten Projekttyp die höchste Bewertung. Ursächlich dafür ist die tiefe Integration in Git-basierte Workflows, die native Unterstützung automatisierter Tests, modellbasierte Dokumentation sowie die nahtlose Einbindung in CI/CD-Pipelines. Ergänzt wird dies durch strukturierte Modularisierung, explizite Versionierung und rollenbasierte Zugriffskonzepte – zentrale Anforderungen für Governance-zentrierte Projekte.

Apache Hop (4,18) positioniert sich als stabiler Zweitplatzierter. Die skript- und metadatenbasierte Architektur sowie das vollständige visuelle Modellieren komplexer Workflows ermöglichen Wiederverwendbarkeit und Prozessklarheit. Einschränkungen bestehen jedoch bei der nativen API-Einbindung sowie der Unterstützung externer Authentifizierungs- und Berechtigungsmodelle.

Airbyte (3,96) weist im Typ-C-Kontext leichte Defizite auf. Zwar überzeugt es durch hohe Konnektorflexibilität und effiziente Datenreplikation, doch fehlen ausgereifte Funktionen für testbare und versionierbare Pipelines sowie rollenbasierte Konfigurationen. Auch das Logging ist bislang rudimentär ausgeprägt und nicht systemweit versioniert.

Talend OSS (3,51) verfügt grundsätzlich über geeignete Funktionen wie Logging, Wiederverwendbarkeit und Fehlertoleranz. Die fehlende Weiterentwicklung nach der Abkündigung der Community-Version im Jahr 2024 sowie die mangelnde CI/CD-Kompatibilität limitieren jedoch den operativen Einsatz in Governance-Umgebungen.

Hevo Data (3,16) erfüllt die Anforderungen von Typ-C-Projekten nur unzureichend. Die Plattform richtet sich an Low-Code-Anwender:innen, bietet jedoch weder eine vollwertige API-Steuerung noch lokal testbare Pipelines oder umfassende Validierungsmechanismen. Die proprietäre Architektur erschwert zudem die Erweiterbarkeit und steht einer durchgängigen Governance-Implementierung entgegen.

Zusammenfassend lässt sich feststellen, dass nur wenige Tools die Anforderungen strukturierter Governance-Szenarien vollständig adressieren. *dbt* erfüllt nahezu alle Bewertungskriterien mit hoher Evidenz und Konsistenz. Werkzeuge mit eingeschränkter Modularität, fehlender Versionierung oder Blackbox-Charakteristik verlieren in diesem Projekttyp hingegen deutlich an Relevanz.

5.2 Projekttypenspezifische Implikationen für die Praxis

Die typenspezifische Bewertung und Einordnung der Werkzeuge verdeutlicht nicht nur technische Unterschiede, sondern ermöglicht konkrete Handlungsempfehlungen für kleine und mittlere Datenintegrationsprojekte mit begrenzten Ressourcen. Die folgenden Absätze skizzieren, welche Tools sich für welchen Nutzungskontext besonders eignen und welche übergreifenden Lehren sich für die Praxis ableiten lassen.

Typ A – Niedrigschwellige Ad-hoc-Projekte. Für Projekte mit begrenzter technischer Infrastruktur, kurzer Laufzeit und vereinfachten Rollenstrukturen sind insbesondere Tools geeignet, die eine visuelle Konfiguration und eine schnelle Inbetriebnahme ermöglichen. Die Analyse bestätigt, dass *Airbyte* und *Apache Hop* diese Anforderungen zuverlässig erfüllen. Auch *dbt* kann in Typ-A-Projekten eingesetzt werden, sofern grundlegende SQL-Kenntnisse vorhanden sind oder unterstützende Plattformen wie *dbt Cloud* genutzt werden. Entscheidende Kriterien sind hierbei GUI-Komplexität, Setup-Aufwand und verfügbare Support-Ressourcen – während Aspekte wie CI/CD-Fähigkeit oder Versionskontrolle eine untergeordnete Rolle spielen.

Typ B – Automatisierte SaaS-Integrationen. Typ-B-Projekte profitieren von Werkzeugen mit hoher Konnektorverfügbarkeit, ausgeprägter Cloud-Kompatibilität und flexibler Zeitsteuerung. *dbt* und *Airbyte* zeigen hier besondere Stärken, da sie moderne ELT- und CDC-Muster nativ unterstützen. *Apache Hop* ergänzt diese Möglichkeiten durch seine offene Architektur und die Integration externer Ausführungsumgebungen. In der Praxis empfiehlt sich für Typ-B-Projekte ein ausgewogenes Verhältnis zwischen Konnektivität, Transformationstiefe und Betriebssicherheit. Tools mit proprietärer Architektur oder eingeschränkter Automatisierbarkeit sind in diesem Kontext nur eingeschränkt empfehlenswert.

Typ C – Strukturierte Governance-Projekte. In Governance-orientierten Projekten mit komplexen Datenflüssen, Testanforderungen und rollenbasiertem Zugriff stellt die Werkzeugwahl eine zentrale Herausforderung dar. Die Scorecard zeigt, dass nur wenige Tools – insbesondere *dbt* und *Apache Hop* – sämtliche relevanten Kriterien systematisch erfüllen. Für Projekte mit Revisionspflicht, interner Qualitätssicherung und automatisiertem Deployment empfehlen sich Werkzeuge mit GitOps-Unterstützung, integrierter Testautomatisierung und nachvollziehbarem Logging. Tools mit geschlossener Architektur, fehlender Modularität oder ohne API-Steuerung sind in diesem Projekttyp klar zu vermeiden.

Übergreifende Lessons Learned. Die Analyse macht deutlich, dass eine pauschale Toolauswahl unabhängig vom Projektkontext nicht zielführend ist. Die Eignung ergibt sich nicht allein aus dem Funktionsumfang, sondern aus der Passung zu den konkreten Rahmenbedingungen eines Projekts. Die Scorecard-Methode bietet hier einen strukturierten Ansatz, um diese Passung kontextbasiert zu bewerten. Für die Praxis ist nicht

das vermeintlich „beste“, sondern das funktional und organisatorisch „passendste“ Tool entscheidend – im Sinne eines ausgewogenen Verhältnisses zwischen technischer Leistungsfähigkeit, Ressourceneinsatz und Zielsetzung.

5.3 Reflexion der Bewertungskriterien und methodische Einordnung

Die Scorecard-basierte Bewertung dieser Arbeit stützt sich auf ein Kategoriensystem aus 14 Kriterien, welche funktionale, bedienungsbezogene und wartungsrelevante Anforderungen kleiner und mittlerer Datenintegrationsprojekte abbilden (vgl. Kapitel 2.3). Die Projekttypisierung A–C ermöglichte eine kontextbasierte Gewichtung, bei der zentrale Bewertungsdimensionen systematisch variiert wurden. Gleichwohl erfordert das zugrunde liegende Bewertungsmodell eine kritische methodische Einordnung.

Kriterienauswahl im Spannungsfeld zwischen Praxisnähe und Abstraktion. Die Auswahl der Kriterien basiert auf einer Synthese aus wissenschaftlicher Literatur, normativen Grundlagen (z. B. ISO/IEC 25010), empirischen Rückmeldungen (Umfrage, Interviews) und Sekundäranalysen. Dabei wurde bewusst auf Merkmale verzichtet, die entweder stark vom individuellen Toolkontext abhängen (z. B. Support-Response-Zeiten) oder deren Operationalisierung keine belastbare Vergleichbarkeit erlauben würde (z. B. „Marktpräsenz“). Umgekehrt wurden praxisrelevante und methodisch konsolidierbare Dimensionen wie Logging, Testbarkeit oder Transformationstiefe systematisch berücksichtigt. Die getroffene Auswahl stellt einen tragfähigen Kompromiss zwischen Generalisierbarkeit und Anwendungsnähe dar – auch wenn nicht alle denkbaren Anforderungen vollständig abgedeckt werden konnten.

Verschiebung klassischer Prioritäten: GUI vs. Governance. Die empirischen Ergebnisse belegen, dass sich die Relevanz einzelner Kriterien projekttypabhängig deutlich verschiebt. So ist etwa die GUI-Komplexität für Typ-A-Projekte erfolgskritisch, während sie im Kontext von Typ-C-Projekten an Bedeutung verliert. Umgekehrt sind Kriterien wie Logging-Granularität oder CI/CD-Fähigkeit im Governance-Kontext zentral, spielen jedoch bei Ad-hoc-Vorhaben nur eine untergeordnete Rolle. Die typenspezifische Gewichtung hat sich somit als wirkungsvoller Mechanismus erwiesen, um starre Bewertungsschemata zu vermeiden und differenzierte, kontextabhängige Bewertungen zu ermöglichen.

Konzeptuelle Trennschärfe und Reduktionslogik. Die systematische Reduktion der ursprünglichen Longlist (vgl. Kapitel 2.3.4) auf 14 konsolidierte Kriterien hat zur Vermeidung funktionaler Überschneidungen beigetragen. So wurden etwa die Aspekte „Fehlertoleranz“, „Debugging“, „Retry“ und „Validierungslogik“ in einem übergeordneten Kriterium zusammengeführt. Diese inhaltliche Bündelung stärkt die konzeptuelle Konsistenz des Modells, reduziert Redundanzen und erhöht die Vergleichbarkeit. Gleichzeitig bleibt die Anschlussfähigkeit an externe Qualitätsmodelle (z. B. ISO/IEC 25010) erhalten, ohne deren normative Struktur vollständig zu übernehmen.

Evidenzbasierung und Bewertungsmanual. Die Bewertung erfolgte nicht auf Grundlage intuitiver Einschätzungen, sondern anhand eines klar definierten Bewertungsmanuals mit Skalenankern (vgl. Anhang H). Bewertet wurde ausschließlich auf Basis dokumentierter Nachweise aus mindestens zwei unabhängigen Quellen (vgl. Triangulationsregel, Kapitel 3.3). Dieses methodische Vorgehen gewährleistet eine belastbare Evidenzgrundlage – auch wenn die Bewertung aufgrund der Monokodierung nicht vollständig frei von Subjektivität ist (vgl. Kapitel 3.9).

Fazit. Die eingesetzten Bewertungskriterien sowie deren Anwendung im Scorecard-Modell sind inhaltlich fundiert, empirisch gestützt und methodisch nachvollziehbar. Die typenspezifische Gewichtung erhöht die Aussagekraft des Bewertungsmodells und erlaubt eine kontextdifferenzierte Beurteilung, ohne in mechanische Gleichbehandlung zu verfallen. Gleichwohl bestehen Grenzen hinsichtlich Vollständigkeit, Objektivierbarkeit und Übertragbarkeit auf projektexterne Kontexte – diese werden in den folgenden Limitationen (Kapitel 5.4) näher ausgeführt.

5.4 Methodenkritik und Limitationen

Trotz des strukturierten und evidenzbasierten Vorgehens weist die Bewertungsmethodik dieser Arbeit methodische Begrenzungen auf. Die folgenden Abschnitte reflektieren zentrale konzeptuelle und empirische Limitationen, die sich aus dem Untersuchungsrahmen, dem Bewertungsmodell sowie den gegebenen Marktbedingungen ergeben.

5.4.1 Begriffliche Abgrenzung: „ETL-Tool“ als Marktbegriff

Der in dieser Arbeit verwendete Begriff „ETL-Tool“ wurde nicht im Sinne einer normativ abgegrenzten Softwarekategorie verstanden, sondern als Sammelbegriff für datenintegrationsbezogene Werkzeuge mit transformatorischem Kern. In der Praxis ist die Verwendung des Begriffs uneinheitlich: Tools wie *dbt*, *Airbyte* oder *Hevo* werden in Fachartikeln, Toolverzeichnissen oder Vermarktungskanälen häufig als „ETL“-Plattformen bezeichnet, implementieren funktional jedoch primär ELT- oder CDC-Mechanismen (vgl. Kapitel 2.2.8). Diese begriffliche Unschärfe wurde durch eine explizite Architekturklassifikation (ETL, ELT, CDC, VDI) systematisch adressiert.

Zugleich zeigt sich, dass marktseitige Selbstzuschreibungen (z. B. „ETL-fähig“) nicht zwangsläufig mit der tatsächlichen Funktionsweise übereinstimmen. Für zukünftige Arbeiten erscheint daher die Verwendung präziserer Begriffe wie „Datenintegrationsplattform“ oder „Transformations-Engine“ empfehlenswert. Die in dieser Arbeit gewählte Begriffsverwendung ist entsprechend als pragmatischer, nicht-normativer Definitionsrahmen zu verstehen.

5.4.2 Methodische Limitationen der Bewertung

Die Scorecard-Methode wurde mit dem Ziel konzipiert, kontextspezifische Bewertungen kleiner und mittlerer Datenintegrationsprojekte zu ermöglichen. Dennoch bestehen mehrere methodische Einschränkungen:

- **Monokodierung:** Die Bewertung der Kriterien erfolgte durch eine Einzelperson. Intercoder-Validierungen oder Konsensverfahren konnten aus ressourcenbedingten Gründen nicht realisiert werden. Subjektive Verzerrungen lassen sich somit nicht vollständig ausschließen.
- **Datenverfügbarkeit:** Die Toolauswahl beruhte auf der Mindestverfügbarkeit belegbarer Kriterien (≥ 8 von 14). Werkzeuge mit unzureichender Dokumentation oder begrenzter Zugänglichkeit wurden ausgeschlossen – auch wenn sie theoretisch relevant gewesen wären.
- **Skalenmodell:** Die eingesetzte Bewertungsskala (1–5) basiert auf semantisch definierten Skalenankern, deren Abgrenzung trotz Manual einen gewissen Interpretati-

onsspielraum lässt – insbesondere bei weichen Kriterien wie „Bedienkomfort“ oder „Plattformunabhängigkeit“.

- **Bewertungsgewichtung:** Die Gewichtungen der Kriterien pro Projekttyp wurden auf Basis empirischer Mittelwerte und heuristischer Anpassungen festgelegt. Eine vollständige Replizierbarkeit durch Dritte erfordert daher detaillierte Kenntnisse der zugrundeliegenden Umfrage- und Interviewdaten.

Diese Einschränkungen wurden im Bewertungsprozess systematisch dokumentiert und – soweit möglich – durch Triangulation und nachvollziehbare Protokollierung kompensiert. Dennoch ist das Bewertungsergebnis primär als kontextbezogene Orientierungshilfe zu verstehen, nicht als absolute Aussage.

5.4.3 Ausblick: Generalisierbarkeit und Übertragungsrahmen

Die Ergebnisse dieser Arbeit beziehen sich explizit auf kleine und mittlere Integrationsprojekte mit klar definiertem Projekttypenraster (vgl. Kapitel 2.1.5). Eine Übertragung auf Großprojekte mit erhöhter Prozesskomplexität, Data-Mesh-Architekturen oder Echtzeit-Pipelines ist nicht vorgesehen. Ebenso wurden keine Aussagen zur Performanz unter Last, zur kommerziellen Wartbarkeit oder zu Lizenzrisiken getroffen.

Zudem basiert die Analyse auf einem begrenzten Tool-Set von fünf bewertbaren Plattformen. Die bewusste Entscheidung, *Talend OSS* trotz Marktrückzugs einzubeziehen, diente unter anderem dazu, die zeitliche Volatilität von Open-Source-Initiativen und deren Relevanz für das Kriterium „Betriebs- und Laufzeitrisko“ sichtbar zu machen. Gleichwohl bleibt die Generalisierbarkeit auf aktuelle Marktdynamiken beschränkt.

Künftige Studien könnten das Bewertungsmodell durch eine breitere Toolauswahl, den Einsatz von Intercoder-Verfahren oder ergänzende Praxistests erweitern – etwa in Form kontrollierter A/B-Vergleiche im produktiven Projektumfeld.

5.5 Ausblick und Forschungsperspektiven

Die vorliegende Arbeit präsentiert ein methodisch fundiertes Bewertungsmodell zur vergleichenden Analyse von ETL-/ELT-Tools in kleinen und mittleren Datenintegrationsprojekten. Die Ergebnisse zeigen, dass eine typenspezifische, kontextabhängige Bewer-

tung nicht nur möglich, sondern notwendig ist, um die Passung zwischen Werkzeug und Projektszenario systematisch zu erfassen.

Zugleich eröffnet die Studie eine Reihe weiterführender Forschungsfragen und Entwicklungsperspektiven, die über den Rahmen dieser Arbeit hinausgehen:

Erweiterung der Toolbasis. Die Analyse beschränkte sich auf fünf methodisch bewertbare Tools. Eine breitere empirische Fundierung könnte durch die Einbeziehung weiterer Plattformen erfolgen – insbesondere kommerzieller, hybrider oder branchenspezifischer Lösungen. Auch die Bewertung moderner Reverse-ETL- oder Streaming-Plattformen (z. B. Estuary, StreamSets) bietet Potenzial für künftige Arbeiten.

Automatisierte Bewertungssysteme. Ein nächster methodischer Entwicklungsschritt liegt in der teilautomatisierten Erhebung und Bewertung. Der Einsatz von Natural Language Processing (NLP) zur Extraktion relevanter Funktionsbeschreibungen aus Dokumentationen oder GitHub-Repositories könnte Bewertungsprozesse deutlich effizienter gestalten. Ebenso bieten semantische Ähnlichkeitsanalysen zwischen Toolprofilen und projektspezifischen Anforderungen ein vielversprechendes Forschungsfeld.

Praxistests und nutzerzentrierte Evaluation. Die Bewertung stützt sich überwiegend auf Sekundärquellen sowie einen exemplarischen Praxistest (Airbyte). Eine systematische Ergänzung durch realitätsnahe Fallstudien, Vergleichstests oder Logfile-Analysen könnte die Bewertung weiter empirisch fundieren. Eine kontinuierliche Rückkopplung zwischen Scorecard-Bewertung und realem Integrationserfolg wäre dabei besonders wertvoll.

Weiterentwicklung der Typologiemodelle. Die in Kapitel 2 entwickelte Projekttypologie A–C hat sich als heuristisch tragfähig erwiesen. Eine datenbasierte Validierung – etwa durch Clusteranalysen umfangreicher Projektdaten oder Benchmark-Datensätze – könnte helfen, die Typen empirisch zu schärfen, zu erweitern oder flexibel an sich wandelnde Toolmärkte anzupassen.

Transfer auf angrenzende Anwendungsbereiche. Das entwickelte Bewertungsmodell ist nicht auf ETL-/ELT-Tools beschränkt, sondern lässt sich grundsätzlich auf andere Softwarekategorien im Data-Engineering-Umfeld übertragen – etwa auf Orchestrierungslösungen (z. B. Dagster), Monitoringtools (z. B. Monte Carlo) oder Workflow-Engines (z. B. Prefect). Ein solcher Transfer würde neue Anwendungsfelder erschließen, etwa im Kontext von DevOps, DataOps oder AI-Pipeline-Management.

Fazit. Die Scorecard-basierte Evaluation stellt eine belastbare Grundlage für die kontextbezogene Toolauswahl in datengetriebenen Projekten dar. Ihre Weiterentwicklung zu einem datenbasierten, nutzerzentrierten und dynamisch skalierbaren Bewertungsansatz eröffnet ein vielversprechendes Forschungsfeld an der Schnittstelle zwischen Methodik und Anwendung.

5.6 Fazit

Ziel dieser Arbeit war die Entwicklung eines methodisch fundierten, typensensitiven Bewertungsmodells für ETL-/ELT-Tools, das sich explizit an den Rahmenbedingungen kleiner und mittlerer Datenintegrationsprojekte orientiert. Vor dem Hintergrund einer heterogenen Toollandschaft, fragmentierter Marktbegriffe und begrenzter Entscheidungstransparenz wurde ein kontextspezifisches Scorecard-Modell konzipiert, angewendet und empirisch validiert.

Die zentrale Forschungsfrage lautete:

> *Wie lassen sich ETL-/ELT-Tools systematisch bewerten, um die Eignung für unterschiedliche Anwendungskontexte kleiner und mittlerer Datenintegrationsprojekte zu bestimmen?*

Diese Frage konnte durch die Entwicklung und Anwendung eines fünfstufigen Bewertungsprozesses beantwortet werden, der auf einer projekttypenspezifischen Gewichtung von 14 empirisch fundierten Bewertungskriterien basiert. Die Kombination aus Scorecard-Logik, Typologie A–C und evidenzbasierter Toolanalyse erwies sich als tragfähige Grundlage für eine differenzierte Bewertung. Die Validierung erfolgte durch Triangulation aus Literatur, Tool-Dokumentationen, Nutzerumfragen und Experteninterviews.

Die Ergebnisse zeigen, dass eine kontextunabhängige Toolbewertung methodisch nicht tragfähig ist. Vielmehr variiert die Eignung eines Werkzeugs deutlich in Abhängigkeit vom Anwendungsszenario: Während grafische Konfigurierbarkeit für Typ-A-Projekte zentral ist, stehen bei Typ-C-Projekten testbare, versionierbare und CI/CD-kompatible Pipelines im Vordergrund. Werkzeuge wie *dbt* und *Apache Hop* zeigen eine hohe Eignung über mehrere Typen hinweg, während andere Tools (z. B. *Talend OSS*, *Hevo Data*) nur in spezifischen Szenarien sinnvoll einsetzbar sind.

Darüber hinaus konnte gezeigt werden, dass auch etablierte Plattformen wie *Talend OSS* durch technologische Marktveränderungen – etwa die Abkündigung der Open-Source-Variante – in ihrer Zukunftsfähigkeit eingeschränkt sein können. Diese Aspekte wurden systematisch in die Bewertung von Betriebsrisiken und Wartbarkeit integriert.

Die entwickelte Bewertungslogik ist modular aufgebaut und erlaubt eine flexible Erweiterung auf andere Tools oder Projektszenarien. Ihre Replizierbarkeit wurde dokumentiert, ihre Anwendbarkeit durch einen vollständigen Bewertungsdurchlauf demonstriert.

Insgesamt leistet die Arbeit einen Beitrag zur methodischen Fundierung kontextsensitiver Bewertungsentscheidungen in datengetriebenen Projekten. Sie richtet sich sowohl an Praktiker:innen, die fundierte Werkzeugentscheidungen treffen müssen, als auch an Forschende, die bestehende Bewertungsmodelle weiterentwickeln möchten.

Literatur

Appelfeller, W. u. a. (2023). *Die digitale Transformation des Unternehmens: Systematischer Leitfaden mit zehn Elementen zur Strukturierung und Reifegradmessung*. Zugriff am 16.04.2025. Wiesbaden: Springer Gabler. DOI: 10.1007/978-3-662-65413-2. URL: https://link.springer.com/book/10.1007/978-3-662-65413-2.

Dash, B. u. a. (2022). „Reverse ETL for Improved Scalability, Observability, and Performance of Modern Operational Analytics - A Comparative Review“. In: *2022 OITS International Conference on Information Technology (OCIT)*. Zugriff am 17.04.2025. IEEE, S. 491–494. DOI: 10.1109/OCIT56763.2022.00097. URL: https://ieeexplore.ieee.org/document/10053738/.

Doan, A. u. a. (2012). *Principles of Data Integration*. Zugriff am 16.04.2025. Amsterdam: Morgan Kaufmann / Elsevier. DOI: 10.1016/C2010-0-67008-1. URL: https://books.google.de/books?id=s2YCKGrO10YC.

Feiler, P. H. u. a. (2010). „System Architecture Virtual Integration: A Case Study“. In: *ERTS2 Embedded Real Time Software & Systems*. hal-02267682. Toulouse, France.

G2 (2025). *Best ETL Tools 2025 | G2*. https://www.g2.com/categories/etl-tools. Zugriff am 3. Mai 2025.

Garcia, P. e. a. (2014). „Quantitative Project Management in SMEs“. In: *Proc. Int. Conf. Project Management*, S. 505–512.

Gartner Inc. (2024). *Magic Quadrant for Data Integration Tools 2024*. https://www.denodo.com/de/document/analyst-report/gartner-magic-quadrant-data-integration-tools-2024. Zugriff am 3. Mai 2025.

Hu, Y. u. a. (2024). „Navigating Digital Transformation and Knowledge Structures: Insights for Small and Medium-Sized Enterprises“. In: *Journal of the Knowledge Economy* 15. Zugriff am 17.04.2025, S. 16311–16344. DOI: 10.1007/s13132-024-01754-x. URL: https://doi.org/10.1007/s13132-024-01754-x.

Inmon, W. H. (2005). *Building the Data Warehouse*. 4th. Zugriff am 16.04.2025. Indianapolis: John Wiley & Sons. DOI: 10.1002/9781119197039. URL: https://www.wiley.com/en-us/Building+the+Data+Warehouse%2C+4th+Edition-p-9780764599446.

International Organization for Standardization (2011). *ISO/IEC 25010:2011 – Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*. https://www.iso.org/standard/35733.html.

ISO/IEC 25024:2015 – Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of data quality (2015). Zugriff am 17.04.2025. URL: https://www.iso.org/standard/35738.html.

Jiang, Z. u. a. (2021). „Digital twin to improve the virtual-real integration of industrial IoT“. In: *Journal of Industrial Information Integration* 22, S. 100196. DOI: 10.1016/j.jii.2020.100196. URL: https://www.sciencedirect.com/science/article/pii/S2452414X20300716.

Khan, B. u. a. (2024). „An Overview of ETL Techniques, Tools, Processes and Evaluations in Data Warehousing“. In: *Journal on Big Data* 6. Zugriff am 16.04.2025, S. 1–20. DOI: 10.32604/jbd.2023.046223. URL: https://www.techscience.com/jbd/v6n1/46223.

Kimball, R. u. a. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. 3rd. Zugriff am 16.04.2025. Indianapolis: John Wiley & Sons. DOI: 10.1002/9781119181748. URL: https://books.google.de/books?id=4rFXzk8wAB8C.

Kozłowski, K. u. a. (2016). „Characteristic Features of Project Management in Small and Medium-Sized Enterprises“. In: *Journal of Economics and Management*, S. 140–150.

Krems, B. (2012). *Projekt / Projektmanagement – Online-Verwaltungslexikon*. Zugriff am 03. Mai 2025. URL: https://olev.de/p/projekt-st.htm#Minimalschema.

Mansour, F. R. u. a. (2024). „A total scoring system and software for complex modified GAPI (ComplexMoGAPI) application in the assessment of method greenness“. In: *Green Analytical Chemistry* 10. Zugriff am 17.04.2025, S. 100126. DOI: 10.1016/j.greeac.2024.100126. URL: https://doi.org/10.1016/j.greeac.2024.100126.

Page, M. J. u. a. (2021). „The PRISMA 2020 statement: an updated guideline for reporting systematic reviews“. In: *BMJ* 372. Zugriff am 15.05.2025, n71. DOI: 10.1136/bmj.n71. URL: https://doi.org/10.1136/bmj.n71.

Patzak, G. u. a. (2017). *Projektmanagement: Projekte, Projektportfolios, Programme und projektorientierte Unternehmen*. 792 Seiten. Linde Verlag GmbH. ISBN: 9783709408889.

Popović, A. u. a. (2024). „A Domain-Specific Language for Managing ETL Processes“. In: *PeerJ Computer Science* 10. Zugriff am 17.04.2025, e1835. DOI: 10.7717/peerj-cs.1835. URL: https://peerj.com/articles/cs-1835/.

Putrama, I. M. u. a. (2024). „Heterogeneous data integration: Challenges and opportunities“. In: *Data in Brief* 56. Zugriff am 16.04.2025, S. 110853. DOI: 10.1016/j.dib.2024.110853. URL: https://doi.org/10.1016/j.dib.2024.110853.

Redman, T. C. (1996). *Data Quality: Management and Technology*. Zugriff am 17.04.2025. Digital Press. ISBN: 978-1555581560.

Rowe, S. F. (2020). *Project Management for Small Projects*. 3. Aufl. 312 Seiten. Berrett-Koehler Publishers. ISBN: 9781523097692.

Schulz, M. (2024). *Analytische Datenarchitekturen: Konzepte für eine robuste und flexible Business-Intelligence-Landschaft*. Zugriff am 16.04.2025. Wiesbaden: Springer Fachmedien Wiesbaden. DOI: 10.1007/978-3-658-45594-1. URL: https://link.springer.com/book/10.1007/978-3-658-45594-1.

Seenivasan, D. (2022). „ETL vs ELT: Choosing the right approach for your data warehouse“. In: *International Journal for Research Trends and Innovation (IJRTI)* 7.2. Zugriff am 21.04.2025, S. 110–122. ISSN: 2456-3315. URL: https://www.ijrti.org/viewpaperforall.php?paper=IJRTI2202018.

Singhal, B. u. a. (2022). „ETL, ELT and Reverse ETL: A business case Study“. In: *2022 Second International Conference on Advanced Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE)*. Zugriff am 17.04.2025, S. 1–4. DOI: 10.1109/ICATIECE56365.2022.10046997. URL: https://ieeexplore.ieee.org/document/10046997.

Szilveszter, M. u. a. (2024). „Comparison of Commercial and Open Source ETL Tools“. In: *2024 IEEE 11th International Conference on Computational Cybernetics and Cyber-Medical Systems (ICCC)*. Zugriff am 17.04.2025. Hanoi, Vietnam: IEEE, S. 000037–000042. DOI: 10.1109/ICCC62278.2024.10582972. URL: https://ieeexplore.ieee.org/document/10582972.

Tawil, A.-R. H. u. a. (2024). „Trends and Challenges towards Effective Data-Driven Decision Making in UK Small and Medium-Sized Enterprises: Case Studies and Lessons Learnt from the Analysis of 85 Small and Medium-Sized Enterprises“. In: *Big Data and Cognitive Computing* 8.7. Zugriff am 16.04.2025, S. 79. DOI: 10.3390/bdcc8070079. URL: https://www.mdpi.com/2504-2289/8/7/79.

Walha, A. u. a. (2024). „Data integration from traditional to big data: main features and comparisons of ETL approaches“. In: *The Journal of Supercomputing* 80.19. Zugriff am 16.04.2025, S. 26687–26725. DOI: 10.1007/s11227-024-06413-1. URL: https://link.springer.com/article/10.1007/11227-024-06413-1.

A Anhang : Interviewleitfaden

Einleitung

Der folgende Interviewleitfaden wurde im Rahmen der qualitativen Datenerhebung (vgl. Kapitel 3.3) eingesetzt, um die Bewertung von ETL-/ELT-Werkzeugen für kleine bis mittelgroße Datenintegrationsprojekte zu erfassen. Die Fragen orientieren sich an den in Kapitel 2.3 entwickelten Bewertungskategorien und dienten der leitfadengestützten Durchführung von vier Experteninterviews.

Hinweis zur Einwilligung

Alle befragten Personen haben vor Beginn der Interviews eine schriftliche Einverständniserklärung zur wissenschaftlichen Nutzung ihrer Aussagen abgegeben. Diese liegen der Verfasserin vor und können auf Anfrage der Prüfenden eingesehen werden.

Interviewfragen

1. Einführung und Hintergrund

- 1.1 Können Sie Ihre berufliche Rolle und Ihren spezifischen Hintergrund im Umgang mit ETL-Tools kurz beschreiben?
- 1.2 Mit welchen ETL-Tools haben Sie bisher in der Praxis gearbeitet? (Bitte nennen Sie gerne auch andere Tools als die folgenden: Airbyte, dbt, Informatica PowerCenter, Azure Data Factory, Google Dataflow, AWS Glue, Fivetran, Matillion.)

2. Auswahlkriterien für ETL-Tools

- 2.1 Welche Kriterien sind für Sie am wichtigsten, wenn Sie ein ETL-Tool für kleine bis mittelgroße Datenintegrationsprojekte auswählen?
- 2.2 Wie schätzen Sie die Bedeutung von Kosteneffizienz und Benutzerfreundlichkeit für kleinere Unternehmen ein?
- 2.3 Gibt es aus Ihrer Sicht Kriterien, die oft übersehen werden, aber speziell für kleinere Projekte wichtig sind?

3. Praktische Erfahrungen mit ETL-Tools

- 3.1 Welches Tool nutzen Sie aktuell am häufigsten und warum? Welche spezifischen Anforderungen erfüllt es?
- 3.2 Wie bewerten Sie dieses Tool hinsichtlich Effizienz, Skalierbarkeit, Benutzerfreundlichkeit, Echtzeit-Verarbeitung und Sicherheit?
- 3.3 Gab es besondere Herausforderungen oder Erfolge bei der Nutzung dieses Tools in kleineren Projekten?

4. Herausforderungen bei der Nutzung von ETL-Tools

- 4.1 Welche Herausforderungen begegnen Ihnen speziell in kleineren Projekten bei der Nutzung von ETL-Tools?
- 4.2 Welche Methoden zur Datenbereinigung und Qualitätssicherung haben sich in Ihrer Praxis bewährt?
- 4.3 Welche Sicherheitsanforderungen müssen ETL-Tools auch in kleineren Projekten erfüllen?

5. Nutzung von ETL-Tools in Cloud-Umgebungen

- 5.1 Haben Sie ETL-Tools in Cloud-Umgebungen eingesetzt? Wenn ja, welche Vorteile und Herausforderungen gab es speziell für kleinere Projekte?
- 5.2 Inwiefern beeinflussen Datenvolumen und Geschwindigkeit die Auswahl eines ETL-Tools für kleine bis mittelgroße Projekte?

6. Best Practices und Empfehlungen

- 6.1 Welche Best Practices oder bewährten Vorgehensweisen können Sie empfehlen, wenn es um die Auswahl und Implementierung eines ETL-Tools geht?

- 6.2 Was ist Ihrer Meinung nach besonders wichtig, um typische Anfangsprobleme zu vermeiden?
- 6.3 Welche Unterschiede sehen Sie zwischen Open-Source- und kommerziellen Tools, speziell für kleinere Projekte?

7. Trends und Entwicklungen im ETL-Bereich

- 7.1 Welche Trends und Entwicklungen im Bereich der ETL-Tools sind aus Ihrer Sicht für kleine bis mittelgroße Projekte relevant?
- 7.2 Welches Potenzial sehen Sie in der Integration von KI/ML-Funktionen in ETL-Prozesse? Welche Herausforderungen könnten sich dabei für kleinere Projekte ergeben?

8. Abschluss

- 8.1 Welches ETL-Tool würden Sie für ein kleines bis mittelgroßes Datenintegrationsprojekt empfehlen, und warum?
- 8.2 Gibt es zusätzliche Überlegungen, die Ihrer Meinung nach bei der Auswahl eines ETL-Tools für kleine bis mittelgroße Projekte wichtig sind?
- 8.3 Dürfen wir uns bei Bedarf für weitere Rückfragen an Sie wenden?

B Anhang : Umfragefragebogen

Einleitung

Die folgende Übersicht enthält alle Fragen der Online-Befragung zur Bewertung und Nutzung von ETL-/ELT-Werkzeugen in kleinen bis mittelgroßen Integrationsprojekten. Die Fragen wurden via Google Forms / LamaPoll erhoben und beinhalten sowohl geschlossene Skalenfragen als auch offene Freitextfelder.

Fragebogenstruktur

Abschnitt 1 – Nutzungserfahrung

- Welche ETL-Tools nutzen Sie derzeit in Ihrem Unternehmen? (Freitext)
- Wie lange arbeiten Sie schon mit ETL-Tools? (Skala / Auswahl)
- In welcher Branche sind Sie tätig? (Auswahl)
- Wie groß ist Ihr Unternehmen? (Auswahl)

Abschnitt 2 – Bewertungskriterien (Skala: 1 = unwichtig bis 5 = sehr wichtig)

- Effizienz
- Skalierbarkeit
- Benutzerfreundlichkeit
- Kosten
- Datenqualität
- Sicherheit
- Flexibilität und Integration
- Wartbarkeit und Erweiterbarkeit
- Echtzeit-Verarbeitung
- Fehlerbehandlung und Wiederherstellung
- Compliance und Governance
- Interoperabilität
- KI/ML-Integration
- Serverlose Architekturen
- Datenkatalog und Metadatenmanagement
- Benutzerrollen und Zugriffssteuerung
- Datenverschlüsselung und Datenschutz
- Community und Support
- Einhaltung von Best Practices
- Erweiterbarkeit durch Plugins
- API-Support

Abschnitt 3 – Toolbewertung (Skala: 1 = sehr schlecht bis 5 = sehr gut)

- Airbyte
- Informatica PowerCenter
- dbt (data build tool)
- Azure Data Factory
- Google Dataflow
- Apache NiFi
- Pentaho Data Integration (PDI)
- Fivetran
- Talend
- Matillion
- AWS Glue
- Integrate.io
- SnapLogic
- Stitch

Abschnitt 4 – Offene Fragen

- Gibt es weitere Kriterien, die Ihrer Meinung nach wichtig sind, aber in dieser Umfrage nicht berücksichtigt wurden?
- Welche Herausforderungen haben Sie bei der Nutzung von ETL-Tools festgestellt?
- Welche Best Practices können Sie bei der Implementierung und Nutzung von ETL-Tools empfehlen?
- Haben Sie Erfahrungen mit der Nutzung von ETL-Tools in Cloud-Umgebungen?
- Falls ja, welche Vorteile und Herausforderungen haben Sie festgestellt?
- Wie groß sind die typischen Datenvolumen, die Sie mit Ihren ETL-Tools verarbeiten?
- Welche spezifischen technischen Anforderungen haben Sie an ETL-Tools?

Abschnitt 5 – Abschlussfragen

- Wären Sie daran interessiert, die Ergebnisse dieser Umfrage zu erhalten?
- Möchten Sie an einem Interview teilnehmen, um Ihre Antworten zu vertiefen?

C Anhang : Toolbewertung - Airbyte

Kompaktübersicht: Dieser Anhang fasst die wichtigsten Eigenschaften des Tools Airbyte kompakt zusammen. Die ausführliche Analyse befindet sich auf der CD. **Erhebungstyp**: **Eigene praktische Tests** (Airbyte v0.50.22, März 2025, Ubuntu 22.04, Docker-Deployment)

Tabelle C.1: Toolprofil Airbyte: Übersicht zentraler Bewertungskriterien

Kriterium	**Zusammenfassung**
GUI-Typ	Browserbasierte Web-GUI; zusätzlich API und CLI (abctl) verfügbar
Setup & Installation	Cloud-Version sofort nutzbar; Self-Hosting via Docker oder Kubernetes möglich; Hybridmodell dokumentiert
Transformationstiefe	Einfache Transformationen über GUI (Feldauswahl, Primärschlüssel); komplexe Transformationen über dbt
Logging & Monitoring	Ausführliche Logs via GUI, CLI und API; Monitoring-Dashboard integriert
Fehlerbehandlung	Retry-Mechanismen und manuelle Neustarts möglich; Fehlerschritte einzeln nachvollziehbar
Wartbarkeit & Modularität	Modulares System mit Connector Builder; YAML-basierte Erstellung eigener Konnektoren möglich
Architekturtyp	Microservices-Architektur; modular, skalierbar, selbstverwaltet oder Cloud-basiert
Integrationstiefe	Sehr breit: über 350 Quellen und Ziele; Custom-Connector-Entwicklung unterstützt
Lizenzmodell	Open Source (MIT-Lizenz) + kommerzielle Airbyte Cloud
Aktualität	Monatliche Releases, aktive GitHub-Entwicklung und transparente Roadmap
Dokumentationsqualität	Sehr umfangreich; Tutorials, API-Doku, Versionierung vorhanden
Community-Aktivität	Sehr hoch (GitHub, Slack, Forum); aktives Community-Engagement
Security & Deployment	TLS, OAuth, RBAC; Self-Hosting und Cloud verfügbar; DSGVO-konform
Besonderheiten	Flexibles Deployment (Cloud, Self-Hosted, Hybrid), starke Connector-Offenheit

D Anhang : Toolbewertung - Apache Hop

Kompaktübersicht:Dieser Anhang fasst die wichtigsten Eigenschaften des Tools Apache Hop kompakt zusammen. Die ausführliche Analyse befindet sich auf der CD. **Erhebungstyp**: Sekundärdatenanalyse (Dokumentation, Fachquellen, Community-Berichte)

Tabelle D.1: Toolprofil Apache Hop: Übersicht zentraler Bewertungskriterien

Kriterium	**Zusammenfassung**
GUI-Typ	Desktop-Client (Hop GUI) und Webserver-Modus (Hop Server) verfügbar
Setup & Installation	Lokale Installation via ZIP-Archiv; Konfiguration über Umgebungsvariablen und Properties-Dateien
Transformationstiefe	Sehr hoch: grafische Modellierung komplexer ETL-Workflows; Unterstützung von Datenflüssen und Pipelines
Logging & Monitoring	Umfangreiche Logdateien pro Workflow und Task; Monitoring über eigene Dashboard-Ansichten
Fehlerbehandlung	Fehlerpfade in Workflows definierbar; Wiederholungsversuche und alternative Pfade möglich
Wartbarkeit & Modularität	Starke Modularität durch Komponenten wie Pipelines, Transforms und Actions; Wiederverwendbarkeit hoch
Architekturtyp	Plugin-basierte modulare Architektur; skalierbar über Hop Server (Microservice-Ansatz möglich)
Integrationstiefe	Vielzahl von Konnektoren für relationale DBs, Big Data-Tools, Cloud-Systeme; offene Schnittstellen
Lizenzmodell	Open Source (Apache License 2.0)
Aktualität	Regelmäßige Updates (ca. vierteljährlich); aktive Entwicklung unter Apache Foundation
Dokumentationsqualität	Ausführlich; Tutorials, Referenzdokumentation, Entwicklerhandbuch
Community-Aktivität	Aktiv (Mailinglisten, GitHub, Apache Slack); Beiträge aus Open-Source-Community
Security & Deployment	Lokaler Betrieb oder Serverbetrieb möglich; Sicherheitsfeatures konfigurierbar (z. B. Benutzerrechte)
Besonderheiten	Visuelle Pipeline-Entwicklung auf grafischer Oberfläche; starke Versionierung von Projekten und Umgebungen

E Anhang : Toolbewertung - dbt

Kompaktübersicht: Dieser Anhang fasst die wichtigsten Eigenschaften des Tools dbt kompakt zusammen. Die ausführliche Analyse befindet sich auf der CD. **Erhebungstyp**: Sekundärdatenanalyse (Dokumentation, Fachquellen, Community-Berichte)

Tabelle E.1: Toolprofil dbt: Übersicht zentraler Bewertungskriterien

Kriterium	**Zusammenfassung**
GUI-Typ	Kommandozeileninterface (CLI); optional dbt Cloud mit Web-GUI
Setup & Installation	Lokale Installation via pip (Python-Paketmanager); dbt Cloud sofort über Web nutzbar
Transformationstiefe	Sehr hoch: komplexe SQL-Transformationen, Modellerstellung, Testing, Dokumentation direkt im DWH
Logging & Monitoring	Umfangreiche Logdateien und Run-Reports; Visualisierung von DAGs (Directed Acyclic Graphs)
Fehlerbehandlung	Fehleranalyse über Logs und Node-Zustände; gezielte Wiederholung fehlerhafter Modelle möglich
Wartbarkeit & Modularität	Hohe Modularität durch Modelle, Snapshots, Macros, Packages; vollständige Projektstruktur in Git verwaltbar
Architekturtyp	Build-Engine für Transformationen direkt auf Ziel-Datenbanken (keine eigene Data Movement Engine)
Integrationstiefe	Direkte Integration in Data Warehouses wie Snowflake, BigQuery, Redshift, Databricks
Lizenzmodell	Open Source (Apache License 2.0) für dbt Core; dbt Cloud kommerziell
Aktualität	Sehr hohe Releasefrequenz (monatlich); aktive Entwicklung
Dokumentationsqualität	Exzellent; umfassendes Developer Hub, API-Referenz, Tutorials, Best Practices
Community-Aktivität	Sehr stark (dbt Slack, GitHub, Discourse); globale Nutzer- und Entwicklerkonferenzen
Security & Deployment	dbt Core lokal ausführbar; dbt Cloud mit SSO, RBAC, Audit-Logs; DSGVO-konform
Besonderheiten	Fokussierung auf Transformationen in SQL ("Tïn ETL/ELT); umfassendes Testing und CI/CD-Integration

F Anhang : Toolbewertung - Talend OSS

Kompaktübersicht: Dieser Anhang fasst die wichtigsten Eigenschaften des Tools Talend Open Studio (OSS) kompakt zusammen. Die ausführliche Analyse befindet sich auf der CD. **Erhebungstyp**: Sekundärdatenanalyse (Dokumentation, Fachquellen, Community-Berichte)

Tabelle F.1: Toolprofil Talend Open Studio: Übersicht zentraler Bewertungskriterien

Kriterium	**Zusammenfassung**
GUI-Typ	Desktop-Client (Eclipse-basiert); grafische Modellierung von Prozessen
Setup & Installation	Lokale Installation via ZIP-Archiv; Java JDK erforderlich; keine native Cloud-Variante
Transformationstiefe	Sehr hoch: komplexe ETL-Prozesse, Mapping, Filter, Joins, Aggregationen möglich
Logging & Monitoring	Logausgaben in Konsole und Dateien; erweiterbar durch Talend Logging Components
Fehlerbehandlung	Fehlerströme und Exception-Handling über Komponenten (tLogCatcher, tDie, tWarn) steuerbar
Wartbarkeit & Modularität	Projekte modular aufgebaut; Subjobs und Routines wiederverwendbar; Git-Integration manuell via Eclipse-Plugins möglich
Architekturtyp	Monolithisches Desktop-Tool mit modularer Komponentenbibliothek
Integrationstiefe	Große Zahl integrierter Konnektoren für Datenbanken, Dateisysteme, Webservices
Lizenzmodell	Open Source (Talend Open Studio, Stand 2024 offiziell abgekündigt)
Aktualität	Entwicklung von OSS 2024 eingestellt; letzte stabile Version 7.3.1
Dokumentationsqualität	Umfassende, aber teils veraltete Dokumentation; Fokus auf PDF-Guides und Online-Wiki
Community-Aktivität	Stark rückläufig seit OSS-Abkündigung; Legacy-Foren und Archive noch aktiv
Security & Deployment	Lokaler Betrieb; Zugriffskontrolle und Verschlüsselung projektseitig implementierbar
Besonderheiten	Früher führend im grafischen ETL-Design; starke Legacy-Verbreitung in Unternehmensprojekten

G Anhang : Toolbewertung - Hevo Data

Kompaktübersicht: Dieser Anhang fasst die wichtigsten Eigenschaften des Tools Hevo Data kompakt zusammen. Die ausführliche Analyse befindet sich auf der CD. **Erhebungstyp**: Sekundärdatenanalyse (Dokumentation, Fachquellen, Community-Berichte)

Tabelle G.1: Toolprofil Hevo Data: Übersicht zentraler Bewertungskriterien

Kriterium	**Zusammenfassung**
GUI-Typ	Web-GUI (SaaS); Konfiguration und Monitoring vollständig browserbasiert
Setup & Installation	Kein Setup erforderlich; sofortige Nutzung nach Registrierung (SaaS-only)
Transformationstiefe	Grundlegende Transformationen über No-Code-Editor; komplexere Transformationen über Zielsysteme erforderlich
Logging & Monitoring	Integrierte Monitoring-Dashboards und Protokolle pro Pipeline; Statusübersichten verfügbar
Fehlerbehandlung	Automatische Wiederholungen bei Fehlern; manuelle Neustarts möglich
Wartbarkeit & Modularität	Pipelines modular aufgebaut; eingeschränkte Anpassbarkeit im Vergleich zu Self-Hosted-Tools
Architekturtyp	Cloud-native Multi-Tenant-Architektur; kein eigenes Hosting möglich
Integrationstiefe	Unterstützung zahlreicher Quellen und Ziele, v. a. Cloud-Datenbanken und SaaS-Anwendungen
Lizenzmodell	Kommerzielles SaaS-Modell; volumenbasierte Abrechnung
Aktualität	Laufende Updates (Rolling Updates); Feature-Erweiterungen regelmäßig
Dokumentationsqualität	Gut strukturiert; Fokus auf Benutzerfreundlichkeit; API-Dokumentation eher kompakt
Community-Aktivität	Gering; primär Support-Portal und LinkedIn-Aktivitäten
Security & Deployment	TLS, AES-256-Verschlüsselung, SOC 2-konform; ausschließlich Cloud-Deployment
Besonderheiten	Schnelle Time-to-Value, vollständig wartungsfrei; keine On-Premises-Option verfügbar

Erklärung zur selbständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original