

Business Intelligence im KMU - Agile Konzeption und Implementierung eines BI-Systems

Bachelorarbeit

Eingereicht von: Johannes Schilling
Matrikelnummer: 22444627
Studiengang: B.Sc. Wirtschaftsinformatik
Referent: Prof. Dr. Michael Amberg
Betreuer: Martin Enders
Bearbeitungszeit: 25.01.2021 - 15.03.2021



Abstract

Der Begriff der Business Intelligence (BI) ist bereits seit mehr als zwei Jahrzehnten in Literatur und Praxis etabliert. Die Bedeutung jener datengetriebener Geschäftsentscheidungen ist unumstritten, was nicht nur für Großkonzerne gilt, sondern insbesondere auch für kleine und mittlere Unternehmen (KMU). Letztere sind jedoch aufgrund ihrer schwächeren Positionierung im Markt und ihrer eingeschränkteren finanziellen Spielräume auf leichtgewichtige, flexible und schnell umsetzbare BI-Lösungen angewiesen.

Ziel dieser Arbeit ist daher die agile Konzeption und entsprechende Implementierung eines BI-Systems für ein KMU. Hierzu wird eingangs die Forschungsfrage „Wie kann ein BI-System in einem KMU gestaltet und agil umgesetzt sein?“ gestellt. Zur Beantwortung jener ist ein Vorgehen per Design Science Research-Methode gewählt. Vor der eigentlichen Instanziierung des Artefakts wird eine begriffliche und methodische Grundlage geschaffen, indem auf Konzepte wie Data Warehousing, Extract-Transform-Load (ETL) und benötigte IT-Architektur eingegangen wird. Zur praktischen zügigen Umsetzung des BI-Systems findet eine agile Projektmanagementmethode nach Scrum Anwendung. Dies bietet den Vorteil, dass Ergebnisse der Implementierung nach dem Design Science Research-Gedanken direkt mit dem Umfeld des Projektes evaluiert und gewonnene Erkenntnisse in der anknüpfenden Inkrement-Stufe mitberücksichtigt werden können. Nach einer abschließenden Diskussion wird gefolgert, dass ein BI-System mittels einer agilen Vorgehensweise und einer auf den Anwendungsfall konkretisierten Data Warehouse-Lösung auch im KMU ressourcenschonend umgesetzt werden kann. Die Vorgehensweise, kennzahlenorientiert Anforderungen an das BI-System abzuleiten, sowie sprintweise Kennzahlen und dafür benötigte ETL-Strecken umzusetzen, erweist sich als zielführend.

Keywords: Business Intelligence, Data Warehouse, Extract-Transform-Load, KMU, Mittelstand, agil

Inhaltsverzeichnis

Abstract	
Inhaltsverzeichnis.....	III
Abbildungsverzeichnis.....	V
Tabellenverzeichnis	VI
Quellcodeverzeichnis.....	VII
Abkürzungsverzeichnis	VIII
1. Einleitung	1
1.1 Motivation, Problemstellung und Zielsetzung	2
1.2 Aufbau der Arbeit.....	3
2. Theoretischer Hintergrund	4
2.1 Business Intelligence und Business Intelligence-Systeme	4
2.2 Einordnung im Kontext	8
2.2.1 Data Science	8
2.2.2 Data Analytics.....	9
2.2.3 Big Data Analytics und Big Data	9
2.2.4 Business Analytics	11
2.3 Technologische Perspektive eines Business Intelligence & Analytics Systems.....	12
2.3.1 Acquisition Layer - Extract, Transform, Load.....	14
2.3.2 Integration Layer - Core Data Warehouse	15
2.3.3 Reporting Layer - Data Mart	19
2.3.4 Visualization Layer	19
3. Forschungsmethode	20
3.1 Design Science Research.....	20
3.2 Agiles Projektmanagement	22
4. Artefakt	24
4.1 Ausgangssituation und vorbereitende Maßnahmen	24
4.1.1 Organisation und Personen.....	25
4.1.2 Technologien	27
4.2 Agiles Projektmanagement mittels Scrum	29
4.2.1 Product Backlog Items und Userstories	29
4.2.2 Sprint Planning, Sprint und Sprint Review.....	30
4.2.3 Limitationen.....	30
4.2.4 Projektverlauf.....	30
4.3 Artefaktbeschreibung	33
4.3.1 Core Data Warehouse	33
4.3.2 Initialisierung und Aufbau der Extract-Transform-Load-Strecken	36
4.3.3 Dashboard	44

5. Evaluation	48
6. Diskussion	51
7. Fazit	54
Literaturverzeichnis	55
Anhang	59

Abbildungsverzeichnis

Abbildung 1: Schematischer Business Intelligence Zyklus	7
Abbildung 2: Einordnung und Abgrenzung von Business Intelligence	8
Abbildung 3: Generische Business Intelligence & Analytics-Technologielandschaft.....	13
Abbildung 4: Vergleich zweier Datenmodelle derselben Datenbasis, (a) Operatives System, (b) Data Warehouse	17
Abbildung 5: Adaption des Design Science Research Framework auf angewendeten Kontext	21
Abbildung 6: Schematischer Aufbau und Ablauf eines Projektes nach Scrum	22
Abbildung 7: Komponenten des BI-Systems im Schichtenmodell	29
Abbildung 8: ER-Diagramm des Core Data Warehouse	35
Abbildung 9: ER-Diagramm des Staging-Bereichs (Ausschnitt).....	36
Abbildung 10: ETL-Strecke der Faktentabelle anhand des Ablaufmodells im SSIS-Paket....	39
Abbildung 11: Datenflusstask 4 für inkrementellen Load der Faktentabelle	41
Abbildung 12: Umgesetztes Dashboard in verschiedenen Filterungen, (a) Nach Monaten, (b) Nach Regionen, (c) Nach Kostenstellen	45
Abbildung 13: Händisch erstellter Vertriebsbericht der Vertriebsleitung.....	59
Abbildung 14: Angelegte Tabellen auf dem SQL-Server des DWHs.....	59
Abbildung 15: Bereitgestellte SSIS-Pakete auf dem SQL-Server des DWHs	60
Abbildung 16: Auszug aus der Changetracking-Versionstabelle	60
Abbildung 17: Bedienoberfläche in MS Visual Studio und Erweiterung Data Tools	61

Tabellenverzeichnis

Tabelle 1: Umfeld des IT-Artefakts nach Organisation, Personen/Rollen und Technologien.....	24
Tabelle 2: Vertriebliche Organisationsstruktur der SanData IT-Gruppe.....	26
Tabelle 3: Userstories zur Dashboardvision der Vertriebsleitung	31
Tabelle 4: Einzelkostenübersicht von Server, Storage, Lizenzen & Software und Personal.....	49
Tabelle 5: Gesamtkosten des umgesetzten BI-Systems	50

Quellcodeverzeichnis

Quellcode 1: Initialisierung der Dimension „Date“	37
Quellcode 2: SQL-Task 1, Lookup Changetracking-Versionsnummer im DWH	40
Quellcode 3: SQL-Task 2, Lookup Changetracking-Versionsnummer in Quelle	40
Quellcode 4: SQL-Task 3, Leeren der benötigten Staging-Tabelle	40
Quellcode 5: SQL-Task A, Abruf der zuletzt geänderten Datensätze	42
Quellcode 6: SQL-Task 5, Update- und Soft-Delete-Verarbeitung der Staging-Tabelle	43
Quellcode 7: SQL-Task 6, Speichern der aktuell gültigen Changetracking- Versionssnummer.....	44
Quellcode 8: DAX-Formeln zur Berechnung der Measures für ACT-, YTD- und %-Beträge ...	46
Quellcode 9: DAX-Formeln zur Berechnung der Measures für YOY-Vergleiche	47
Quellcode 10: create_DWH.sql, Skript zur Initialisierung des Data Warehouse.....	67
Quellcode 11: setup_ETL.sql, Übersicht der in den SSIS-Paketen verwendeten SQL- Fragmente	74

Abkürzungsverzeichnis

Allgemeine Abkürzungen

BI	Business Intelligence
BI & A	Business Intelligence & Analytics
CAx	Computer Aided Everything
DWH	Data Warehouse
ERP	Enterprise Resource Planning
ETL	Extract, Transform, Load
FK	Fremdschlüssel
IT	Informationstechnologie
KMU	Kleine und mittlere Unternehmen
MA	Mitarbeitende
MS	Microsoft
NF	Normalform
OLAP	Online Analytical Processing
PK	Primärschlüssel
SK	Surrogatschlüssel
SSIS	Microsoft SQL-Server Integration Services
VM	Virtuelle Maschine

Kennzahlenbezogene Abkürzungen

ACT	Betrachtungsperiode aktueller Monat
YOY	Vergleichsperiode Year-over-year
YTD	Betrachtungsperiode Year-to-date
YTP	Vergleichsperiode Year-to-plan

1. Einleitung

„Wenn uns die Welt um etwas beneidet, dann ist es der deutsche Mittelstand. Er ist Innovations-, Technologie- und Wirtschaftsmotor Deutschlands. Er erfindet sich ständig neu, steht für das internationale Qualitätsmerkmal „Made in Germany“ und ist Garant für die stabile Stellung Deutschlands. Regional, national und rund um den Globus.“

(Bundesverband mittelständische Wirtschaft, 2020)

Mit diesen Worten bewirbt der Bundesverband der mittelständischen Wirtschaft auf seiner offiziellen Website die Mitgliedschaft in selbigem. In der Tat zählen im Jahre 2020 mehr als 99% sämtlicher Unternehmen in Deutschland zum Bereich der kleinen und mittleren Unternehmen (KMU), erwirtschaften Kraft ihrer selbst rund 34,4% (~2 Billionen Euro) des gesamtheitlichen deutschen Umsatzes und beschäftigen mehr als 58% (~18 Mio. Personen) aller sozialversicherungspflichtigen Angestellten (Institut für Mittelstandsforschung (IfM) Bonn, 2020). Somit ist der Mittelstand Deutschlands berechtigterweise als Wirtschafts- und Beschäftigungsmotor zu bezeichnen.

In Deutschland sind vornehmlich zwei Charakterisierungen für den Begriff der KMU etabliert. Die Europäische Kommission zieht hierbei die quantitativen Merkmale Mitarbeiter (MA) und Jahresumsatz heran und trifft eine Unterteilung in Kleinstunternehmen (≤ 9 MA; ≤ 2 Mio. Euro Umsatz), kleine Unternehmen (≤ 49 MA; ≤ 10 Mio. Euro Umsatz) und mittlere Unternehmen (≤ 249 MA; ≤ 50 Mio. Euro Umsatz). Jenseits der genannten Schranken wird von großen Unternehmen gesprochen (Kaschny et al., 2015, S. 11). Das Institut für Mittelstandsforschung erweitert diesen Ansatz um den qualitativen Aspekt der Einheit von Eigentum und Leitung. In deren Charakteristik müssen bis zu zwei natürliche Personen sowohl der Geschäftsführung des Unternehmens angehören, als auch mindestens 50% der Unternehmensanteile halten (Welter et al., 2014).

Im Vergleich zu Großunternehmen ist es gerade für KMU eine Herausforderung, aufgrund beschränkter finanzieller Ressourcen dem anhaltenden, globalen Wettbewerb standzuhalten (Kaschny et al., 2015, S. 15). Daher ist es von essentieller Bedeutung, dass KMU fähig sind ihre Geschäftstätigkeit kontinuierlich zu überwachen und zu optimieren, um möglichst effizient mit vorhandenen Ressourcen umzugehen (Llave, 2019). In Unternehmen eingesetzte IT-Systeme generieren täglich ein enormes Volumen an Daten, welche als wertvolle Basis für datengetriebene Entscheidungsfindung zur Ressourcenoptimierung genutzt werden sollten (El-Adaileh & Foster, 2019). An dieser Stelle setzt der in der Literatur etablierte Begriff der Business Intelligence (BI) an. BI

im klassischen Sinne kann hierbei als Vorgehen verstanden werden, Strategien, Prozesse, Anwendungen, Daten, Technologien und Architekturen zu vereinen, um unternehmerische Entscheidungstragende durch das Sammeln, Aufbereiten und Visualisieren relevanter, betrieblicher Informationen zu unterstützen (Dedić & Stanier, 2017). Was als essentieller Bestandteil der Entscheidungsgrundlage in größeren Unternehmen bereits seit geraumer Zeit Standard ist, stellt KMU oftmals vor die Herausforderung der Konzeption und Implementierung eines solchen BI-Systems. Ursächlich sind beispielsweise kostenintensive BI-Software oder Hardware zum Betrieb der benötigten IT-Infrastruktur und insbesondere der Komplexitätsgrad der Problemstellung an sich (Becerra-Godinez et al., 2020; Llave, 2019). Der Einsatz von BI trägt nicht nur zu einem besseren Verständnis der Organisation bei, sondern ermöglicht Entscheidungstragenden dadurch auch fundiertere Entscheidungen, sowie eine frühzeitige Berücksichtigung von Entwicklungen im Unternehmen (Dedić & Stanier, 2017). Diese Chance sollten KMU zur Steigerung der Unternehmensperformance in einem äußerst dynamischen Wettbewerbsumfeld nutzen, um einen Anschluss an den Markt zu wahren (Popović et al., 2019).

1.1 Motivation, Problemstellung und Zielsetzung

Die SanData EDV-Systemhaus GmbH mit Hauptsitz in Nürnberg ist Bestandteil und Muttergesellschaft einer mittelständischen Unternehmensgruppierung, die unter dem Namen „SanData IT-Gruppe“ in Deutschland und Österreich auftritt. Seit der Gründung im Jahre 1981 durch Heinrich Straub werden mittleren, sowie großen Unternehmen maßgeschneiderte IT-Lösungen angeboten, wobei auf die umfängliche Expertise in sieben Geschäftsfeldern zurückgegriffen werden kann. Neben dem klassischen Systemhausgeschäft zählt auch die Erbringung von sogenannten Managed-Services, bei dem SanData als Dienstleister IT-Infrastruktur für Kunden betreibt, zu den Kernkompetenzen. Im Jahr 2020 beschäftigt die familiengeführte IT-Gruppe mehr als 460 Mitarbeitende und erzielt einen gruppenweiten Umsatz von ca. 118 Mio. Euro.

Durch das starke Wachstum des Unternehmens, sieht sich SanData 2018 gezwungen, die heterogen gewachsene interne IT-Landschaft aus mehrheitlich autarken IT-Systemen zu vereinheitlichen. Im Zuge dessen wird bis 2019 das auf Microsoft Dynamics basierende integrierte Enterprise-Resource-Planning (ERP)-System SITE eingeführt und alle Geschäftsprozesse entsprechend angepasst. Aufgrund fehlender personeller Kapazitäten werden allerdings bis heute Auswertungen, Berichte und Geschäftsanalysen mehrheitlich direkt aus der Datenbank des Live-Systems abgefragt und in Form von statischen Excel-Tabellen an das Management ausgewiesen. Um eine globale Sicht auf Quelldaten und damit übergreifende Auswertungen und selbstgesteuerte, systematische, dynamische Analysen zu ermöglichen, ohne die Performance des Live-Systems

durch zusätzliche umfangreiche Abfragen einzuschränken, soll im Rahmen der vorliegenden Arbeit ein BI-System, bestehend aus drei Kernkomponenten konzeptioniert und implementiert werden: Ein dediziertes Data Warehouse, zur langfristigen konsistenten Bereitstellung einer einheitlichen Datenbasis, entsprechende Extract-Transform-Load-Prozesse, die als Schnittstelle Daten aus Quellsystemen sammeln, aufbereiten und an das Data Warehouse übergeben und einer anschaulichen Visualisierung. Um möglichst zügig und damit kostensparend, valide Ergebnisse zu liefern, wird das BI-System mittels eines agilen Projektmanagementframeworks nach Schwaber & Sutherland (2017) namens Scrum umgesetzt.

Die vorliegende Arbeit beantwortet somit die Forschungsfrage „Wie kann ein BI-System in einem KMU gestaltet und agil umgesetzt sein?“. Die Fragestellung ist insofern relevant, da kontemporär keine vergleichbare Ausarbeitung existiert, die jene Thematik anhand eines konkreten Beispiels aus dem Mittelstand veranschaulicht. Insofern wird ein Beitrag zur - unter anderem von Trieu (2017) - identifizierten Forschungslücke geleistet.

1.2 Aufbau der Arbeit

In Kapitel 2 wird vorab eine begriffliche Ausgangsbasis geschaffen, indem BI in den Kontext von Data Science, Data Analytics, Big Data Analytics, Big Data und Business Analytics eingeordnet wird. Weiter werden anhand einer exemplarischen Business Intelligence- & Analytics-Technologielandschaft (BI & A) relevante Konzepte rund um den Begriff des Data Warehouses (DWH), Extract-Transform-Load (ETL) und der Datenvisualisierung eingeführt. Kapitel 3 beschreibt die in der Arbeit verwendete Forschungsmethode, den Design Science Research-Ansatz nach Hevner et al. (2004), näher. In Kapitel 4 wird die Vorgehensweise bei der Instanziierung des IT-Artefaktes mittels agilem Projektmanagement nach Scrum und das Artefakt selbst - das BI-System - beschrieben. Die Ergebnisse jener Instanziierung werden in Kapitel 5 evaluiert und in Kapitel 6 diskutiert. Abschließend fasst Kapitel 7 die in den vorherigen Abschnitten gewonnenen Erkenntnisse zusammen und gibt einen Ausblick auf weitere interessante Fragestellungen im Zusammenhang mit BI-Systemen in KMU.

2. Theoretischer Hintergrund

*„Today, it is difficult to find a successful enterprise that has not leveraged BI technology for its business.“
(Chaudhuri et al., 2011, S. 88)*

Mit dieser Aussage macht die Autorenschaft in ihrem von mehr als 1000 Veröffentlichungen zitierten Artikel „An overview of business intelligence technology“ in der renommierten Fachzeitschrift „Communications of the Association for Computing Machinery“ auf die enorme Bedeutung des Einsatzes von BI durch Unternehmen aufmerksam. Jedoch ist BI selbst zum Veröffentlichungszeitpunkt des Beitrages im Jahre 2011 lange kein neuartiger technologischer Trend mehr. Die Begrifflichkeit ist im Verständnis historisch gewachsen.

Im Folgenden wird in Kapitel 2.1 die historische Entwicklung des Begriffs BI kurz dargestellt, um anhand dessen eine aktuelle Definition leisten zu können. Hieran anknüpfend findet in Kapitel 2.2 eine Einordnung der Begrifflichkeit BI im Kontext aktueller Ausprägungen von Data Science statt, um abschließend in Kapitel 2.3 technologisch relevante Konzepte einzuführen.

2.1 Business Intelligence und Business Intelligence-Systeme

Erstmalig erwähnt wird der Begriff BI im Jahre 1958 durch den Informatiker Peter Luhn, als er ein automatisiertes System zur Datenanalyse von Textdokumenten beschreibt, das eine unternehmensinterne Entschlussfassung unterstützen soll (Luhn, 1958). Darauf aufbauend werden ab den 1970er Jahren sogenannte Decision Support-Systeme und Management Support-Systeme diskutiert (Ereth & Kemper, 2016; Z. Sun et al., 2015). Jene Systeme werten automatisiert, jedoch voneinander separiert, transaktionale Datenquellen im Unternehmen aus, um Entscheidungsprozesse im Management der Organisation mit Modellen, Methoden und problembezogenen Auswertungen zu erleichtern (Power, 2007).

Fortan erweitert sich das Verständnis von BI um die Integration mehrerer, unterschiedlicher Datenquellen. BI nimmt hierbei die Rolle einer Schnittstelle ein, die diverse Komponenten einer betrieblichen IT-Infrastruktur verbindet, um deren Informationen dediziert zu sammeln, aufzubereiten und zu visualisieren (Olszak & Ziembka, 2012). In diesem Kontext lassen sich mehrheitlich zwei Ausprägungen bei einer Definition von BI feststellen:

(1) BI *als Sammlung* von Technologien, Architekturen, Werkzeugen und Prozessen zur Umwandlung von Daten aus verschiedenen betrieblichen IT-Systemen in deskriptives und diagnostisches Wissen zur Unterstützung von Geschäftsentscheidungen (Ain et al.,

2019; Becerra-Godinez et al., 2020; Fischer et al., 2020, S. 148; Larson & Chang, 2016; Muntean & Surcel, 2013; Olszak & Ziembka, 2012).

(2) BI als Prozess des Extrahierens, Aufbereitens, nutzerzentrierten Speicherns und Visualisierens strukturierter Daten im betrieblichen Kontext, um operative und dispositive Entscheidungen in Form von deskriptiven und diagnostischen Analysen, Ad-hoc-Abfragen, Berichten und Prognosen zu unterstützen (Dedić & Stanier, 2017; Ereth & Kemper, 2016; Llave, 2019; Obeidat, 2015; Popovič et al., 2019; Z. Sun et al., 2018).

Ausprägung (1) kennzeichnet sich hierbei insofern, als das von einem Verständnis als Sammlung von Technologien anstatt einem Verständnis als Prozess an sich ausgegangen wird (Jourdan et al., 2008). Beide Ausprägungen haben den an Decision- und Management Support-Systeme angelehnten unterstützenden Charakter gemeinsam. Durch die Abdeckung sowohl operativer als auch dispositiver Entscheidungen ist die adressierte Zielgruppe jedoch breiter gefächert.

Ferner ist erwähnenswert, dass bei der Übersetzung von „Intelligence“ vom Englischen ins Deutsche ein differenzierteres Verständnis geboten ist. Bei einer wörtlichen Übersetzung von BI als „Geschäftsintelligenz“ ginge man terminologisch von einer Interpretation im Sinne von menschlicher Intelligenz aus. Dieser läge ein selbstständiges Lernen, Verstehen und Interpretieren von Zusammenhängen zugrunde. Im Rahmen von BI beschränkt sich Intelligenz allerdings auf die einfache, intuitive Anwendbarkeit eines Systems. Dementsprechend ist eine Übersetzung und Interpretation von „Intelligence“ im Sinne von „Einblick“ oder „Einsicht“ vorzugswürdiger, als die wörtliche Übersetzung (Gluchowski, 2016; Z. Sun et al., 2018).

Setzt man BI im Unternehmen ein, ist ein sogenanntes BI-System nicht als alleinstehendes IT-System zu sehen (Larson & Chang, 2016). Vielmehr kombiniert ein BI-System alle Systemkomponenten, die eben jene entscheidungsrelevanten Daten zusammentragen, aufbereiten, dauerhaft nutzerorientiert speichern und visuell veranschaulichen (Becerra-Godinez et al., 2020). BI-Systeme lassen sich nach organisatorischen und technologischen Gesichtspunkten unterscheiden. In technologischer Hinsicht werden zur Konsolidierung und dauerhaften Speicherung von Daten aus verschiedenen Quellsystemen Datenbanken, je nach Architektur in Form eines oder mehrerer DWHs, eingesetzt. ETL-Prozesse verantworten hierbei die automatisierte Transferierung und entsprechende Vorbereitung und Harmonisierung der Daten aus Live-Systemen im Unternehmen. Hierzu zählen beispielsweise das ERP-System oder das Customer-Relationship-Management-System. Die Speicherung folgt einem vorab festgelegtem vereinheitlichenden Schema, um die Daten - ähnlich dem Prinzip einer Lagerhalle - zur weiteren Verarbeitung durch anknüpfende Komponenten anzubieten (Ain et al., 2019).

Technologien zur Datenabfrage und -analyse ermöglichen eine umfangreiche Interpretation der gesammelten Daten. So erlaubt beispielsweise Online-Analytical-Processing (OLAP) eine multidimensionale Datenanalyse in Echtzeit mit Operationen wie „Rollup“, „Drillthrough“ oder pivotisieren der Daten anhand verschiedener Dimensionen, wie Kunde, Zeit oder Lokalität (Gluchowski, 2016).

Im letzten Schritt werden im Rahmen eines Reportings Entscheidungstragenden die gewonnenen Einblicke zur Verfügung gestellt. Im klassischen Sinne wird hierbei auf druckoptimierte, statische Berichte zurückgegriffen (Hoffjan & Rohe, 2018). Da aber der Informationsbedarf in den Unternehmen in den letzten Jahren enorm gestiegen ist, wird immer mehr auf interaktive Lösungen der Datenvisualisierung in Form von Dashboards zurückgegriffen, die auf diversen Endgeräten ausgespielt werden können (Obeidat, 2015). Gerade der Bereich Visual Analytics beziehungsweise Visual Business Intelligence, also eine explorative Datenanalyse durch aktive Nutzerinteraktion, erfährt immer größerer Beliebtheit (Kohlhammer et al., 2016, S. 311). Hierzu generieren dedizierte Report-Server ebendiese Dashboards, welche dem Endnutzer mithilfe von interaktiven Graphiken, Charts und Widgets die Möglichkeit bieten, diverse Kennzahlen des jeweiligen Unternehmens selbstgesteuert zu analysieren oder Ad-hoc-Abfragen zu erstellen (Ain et al., 2019). Zur weiteren Veranschaulichung ist auf Kapitel 2.3 zu verweisen, in welchem ein BI & A-System technologisch beispielhaft dargestellt und weiter erläutert wird.

Der derzeitige Trend entwickelt sich hin zu einer durch den Endanwendenden selbstgesteuerten Analyse und Auswertung, was als Self-BI bezeichnet wird. Dabei ist das BI-System so gestaltet, dass der Anwendende durch den Zugriff auf relevante Unternehmensinformationen - losgelöst von der IT-Abteilung - eigene Berichte oder Dashboards erstellen kann (Obeidat, 2015). Dies erhöht die Reaktionsfähigkeit auf kurzfristige Entwicklungen im Unternehmen enorm und verkürzt somit den Prozess der Informationsbereitstellung. Schlussendlich bleibt dadurch mehr Zeit für eine Validierung der Ergebnisse durch den Entscheidungstragenden (Hoffjan & Rohe, 2018).

Unter organisatorischer Perspektive kommen Aspekte einer einheitlich kommunizierten und gelebten Entscheidungskultur hinzu. Als Anwendungsfall zeichnet sich beispielsweise die Entwicklung einer Unternehmensstrategie ab (Obeidat, 2015). BI-Systeme sollten unternehmensweit als Lösungen verstanden werden, die Daten in wertvolle Informationen und Wissen transformieren und somit eine fundierte Grundlage für datengetriebene Entscheidungen bilden. Der tatsächliche Wert eines BI-Systems entsteht dann bei einer aktiven Adaption der durch BI gewonnenen Erkenntnisse bei der tatsächlichen Strategiumsetzung (Olszak & Ziemba, 2012).

BI ist weiter als iterativer Prozess zu verstehen. Abbildung 1 veranschaulicht einen typischen BI-Zyklus. In Phase 1 sind Erkenntnisbedarfe und -ziele der jeweilig adressierten Zielgruppe festzustellen. Anschließend wird die vorhandene Datenbasis geprüft, benötigte Informationen gesammelt und gegebenenfalls in das BI-System integriert, um in Phase 3 verarbeitet, transformiert und weiter analysiert zu werden. Phase 4 bereitet sodann die aggregierten Informationen auf und verteilt diese an die Adressaten. Abschließend wird in Phase 5 Feedback eingeholt, um die Ergebnisse in der nächsten Iteration kontinuierlich zu verbessern (El-Adaileh & Foster, 2019; Qamar & Raza, 2020, S. 12).

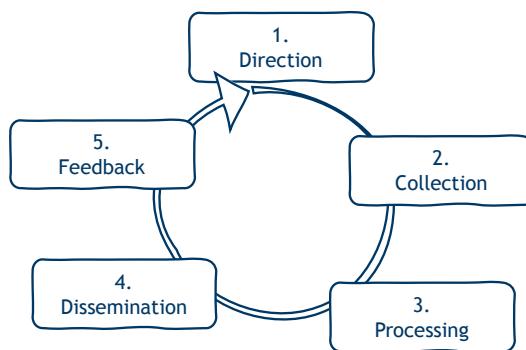
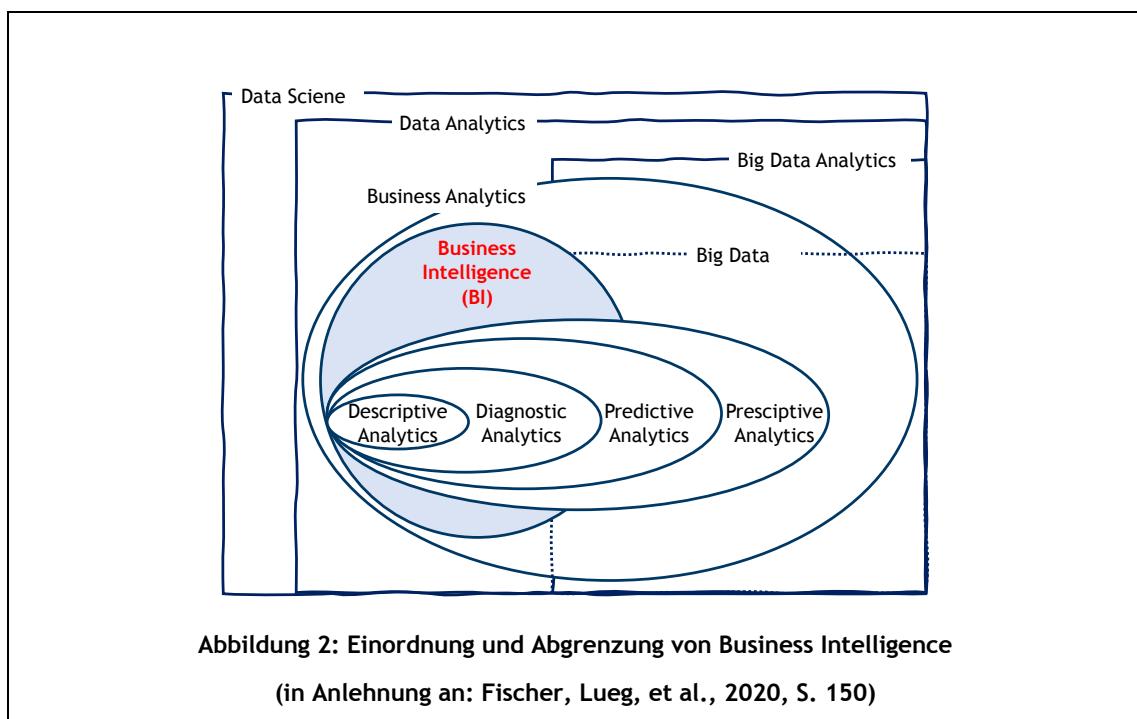


Abbildung 1: Schematischer Business Intelligence Zyklus

(in Anlehnung an: Qamar & Raza, 2020, S. 12)

2.2 Einordnung im Kontext

In der Praxis sind oftmals dahingehend Unklarheiten vorzufinden, inwiefern sich BI von Business Analytics unterscheidet, welche Rolle das Trendwort „Big Data“ spielt oder welchen Rahmen Data Science bildet (Dedić & Stanier, 2017), weshalb eine Veranschaulichung der begrifflichen Konzepte als Venn-Diagramm in Abbildung 2 erfolgt. In den nachfolgenden Kapiteln wird eine Einordnung der Begrifflichkeiten, anhand des Diagramms von außen nach innen vorgenommen. Hierzu wird jeder Begriff zuerst erläutert und darauffolgend anhand seiner Schnittmengen, also seines Kontextes, analysiert.



2.2.1 Data Science

Als konzeptioneller Rahmen steht Data Science für den interdisziplinären Ansatz aus Gebieten der Mathematik, Informatik, Informationstechnologie, Soziologie sowie des Managements, mit dem Ziel, aus Daten einen Mehrwert zu gewinnen. Dieser Ansatz deckt alle Aspekte, beginnend bei der Generierung von Daten aus Zeichen und Syntax, über deren Bedeutung und Verarbeitung, bis hin zur Konvertierung der Informationen in Wissen - durch Kontextualisierung - ab (Qamar & Raza, 2020, S. 20). Der Begriff der Daten ist zunächst nicht näher spezifiziert. Diese können daher strukturiert sowie semistrukturiert oder unstrukturiert, in größerem oder kleinerem Umfang, im Stream oder eben statisch vorliegen (Obeidat, 2015; van der Aalst, 2016, S. 10). Demnach stellt Data Science einen übergeordneten Begriff dar, der eine Reihe an Konzepten und Methoden, die einen Mehrwert aus Daten zu erzeugen, vereint (vgl. Abbildung 2).

2.2.2 Data Analytics

Allerdings ist hervorzuheben, dass Daten an sich noch keinen nennenswerten Mehrwert darstellen. Erst mit der Anwendung konkreter Konzepte und Methoden kann aus Daten ein Mehrwert extrahiert werden (Chamoni & Gluchowski, 2017). Data Analytics ist dementsprechend als Prozess der Wissensextraktion aus Rohdaten zu verstehen und umfasst somit alle weiteren Konzepte und Begrifflichkeiten wie BI, Business Analytics und Big Data (vgl. Abbildung 2, S. 8). Bereits triviale statistische Konzepte, wie beispielsweise eine Durchschnittsberechnung oder Varianzbestimmung können als Data Analytics bezeichnet werden (Qamar & Raza, 2020, S. 2).

Weitergeführt unterscheidet Data Analytics in der Perspektive der Analyse: Steht eine Beschreibung und Aggregation relevanter Daten im Vordergrund, wird von deskriptiver oder auch diagnostischer Analyse gesprochen (Fischer et al., 2020, S. 149). Bewegt man sich weg von einer historischen Analyse hin zu Kausalanalysen oder abgeleiteten Handlungsempfehlungen inklusive der zugehörigen Konsequenzen (prognostizierenden Verfahren) kommt der Bereich der Advanced Analytics ins Spiel. Dieser gliedert sich in prädiktive Analyse (Was wird geschehen?) und präskriptive Analyse (Was ist zu tun?) (Chamoni & Gluchowski, 2017). Mithilfe statistischer Verfahren und maschinellen Lernens können Zusammenhänge und Gesetzmäßigkeiten in vorliegenden Daten nutzbar gemacht werden. Machine Learning ermöglicht hierbei, ausgehend von einer Trainingsdatenmenge, Eingabedaten gleichen Kontexts auf einen Zielwert abzubilden (Japkowicz & Shah, 2011, S. 24). Hierzu werden beispielsweise Entscheidungsbäume, neuronale Netze oder Support Vector Machines genutzt, um Zusammenhänge und Muster in Daten zu erkennen (Ereth & Kemper, 2016). Exemplarische Anwendungsfälle sind medizinische Diagnosen auf Basis zahlreicher Muster in Patientendaten, Erkennung von Öllecken im Meer anhand von Satellitenbildern, Kreditkartenbetrug oder Risikomanagement bei Anlagestrategien im Wertpapierbereich (Y. Sun et al., 2009).

Der Bereich der Data Analytics unterteilt sich weiter anhand der Kriterien Umfang und Kontext der zu analysierenden Daten, in Big Data Analytics und Business Analytics (vgl. Abbildung 2, S. 8).

2.2.3 Big Data Analytics und Big Data

Die Beschaffenheit und das Volumen der zu analysierenden Daten werden als wesentliches Alleinstellungsmerkmal von Big Data genannt. Als Teilbereich von Data Analytics ist Big Data Analytics eine Spezifizierung selbiger, bei welcher Data Analytics-Methoden auf den Anwendungsfall zur Verarbeitung von „Big Data“ angepasst werden (Duan & Xiong, 2015). Bei der Betrachtung der Struktur und Beschaffenheit ist auffällig, dass sowohl strukturierte, semistrukturierte als auch unstrukturierte Daten zu Big Data gezählt werden können. Konträr hierzu steht BI, die sich auf die Analyse strukturierter

Daten beschränkt (Dedić & Stanier, 2017). Im Vergleich zu „small“ Data ist Big Data durch vier Eigenschaften charakterisiert (Qamar & Raza, 2020, S. 4):

- Volume (Datenmengen im Petabyte-, Zettabyte- und Exabytebereich)
- Velocity (Geschwindigkeit der Datengenerierung und -verarbeitung möglichst in Echtzeit)
- Veracity (Validität der Daten)
- Variety (Vielfalt der Datentypen und -quellen)

Während konventionelle BI auf strukturierte Speicheransätze wie die eines DWHs zurückgreift, setzt Big Data an dem Punkt an, an welchem traditionelle, dem relationalen Ansatz folgende Datenbanken, an ihre Grenzen kommen (Chen et al., 2014). Wenn Datenmengen im Petabyte-, Zettabyte- und Exabyte-Bereichpersistiert und verarbeitet werden müssen, haben sich NoSQL-Speicheransätze („Not only SQL“) als leistungsfähiger gegenüber konventionellen relationalen SQL-Ansätzen erwiesen, da diese effizienter mit verteilten Systemen und horizontaler Skalierung umgehen können. Im Gegensatz zum relationalen Ansatzes wird NoSQL-Technologie in vier Speicherstrategien differenziert (Duan & Xiong, 2015):

(1) *Document-Stores* (z.B. MongoDB oder Amazon DynamoDB) speichern Dokumente mit einem eindeutigen Identifikator ab. Hierbei spielt es keine Rolle, inwiefern ein Datensatz strukturiert ist. Es können alle Formate an Dateien verarbeitet werden, beispielsweise sowohl eine Bytefolge eines Videos als auch semistrukturierte Dateiformate wie XML oder JSON (Chen et al., 2014).

(2) *Graph-Datenbanken* (z.B. Neo4JS oder Microsoft Azure Cosmos DB) speichern Daten in Form von Knoten und Kanten. Dies eignet sich besonders gut, um netzwerkartige Strukturen zur Repräsentation eines Wissensgraphen oder eines sozialen Netzwerkes darzustellen (Angles & Gutierrez, 2008).

(3) *Key-Value-Stores* (z.B. Redis oder Memcached) gehören zu den trivialeren Formen der Datenspeicherung, da lediglich Schlüssel-Wert-Paare abgespeichert werden. Ein Wert kann nach Abspeicherung anhand seines Schlüssels zurückgegeben werden. Zwar eignet sich diese Form der Speicherung weniger für komplexe Anwendungsfälle, dafür skaliert jene aber aufgrund des trivialen Aufbaus besonders schnell (Chen et al., 2014).

(4) *Spaltenorientierte Datenbanken* (z.B. Cassandra oder HBase) verwahren Daten in Form dynamischer Spalten und können funktional als zweidimensionale Key-Value-Stores gesehen werden (Saecker & Markl, 2013).

Konkretisiert man nun den Kontext der Daten auf einen Anwendungsfall im Unternehmensumfeld, ergeben sich in der Betrachtung zwei weitere Teilmengen: Business Analytics und BI (vgl. Abbildung 2, S. 8).

2.2.4 Business Analytics

Während Data Analytics nicht auf einen konkreten Anlass beschränkt ist, bezeichnet Business Analytics die Anwendung von Analytics-Methoden im Kontext von Unternehmensdaten (Duan & Xiong, 2015). Vor diesem Hintergrund kann Business Analytics nach Fischer et al. (2020, S. 149) als Sammeln, Aufbereiten und Analysieren von großen, sowohl internen als auch externen Datenbeständen aus dem Unternehmenskontext, durch vergangenheits- und zukunftsorientierte statistische, qualitative sowie quantitative Verfahren, beschrieben werden.

In Abgrenzung zur vergangenheitsorientierten BI ist hervorzuheben, dass der Fokus von Business Analytics in der Nutzung von Zusammenhangsanalysen, Prognosen, Simulationen und Optimierungsverfahren liegt (Gronau et al., 2016). Entsprechend der Definition umfasst Business Analytics damit sowohl vergangenheits- als auch zukunftsbezogene Analysen und verbindet somit Methoden aus dem Bereich von BI und Big Data Analytics (vgl. Abbildung 2, S. 8). Eine Differenzierung von Business Analytics wird anhand der vier bereits in Kapitel 2.2.2 vorgestellten Perspektiven der Analysen getroffen, jedoch jeweils auf den unternehmerischen Kontext konkretisiert:

(1) *Deskriptive Business Analytics* beantwortet die Frage „Wie haben wir uns entwickelt?“. Hierzu erfolgt eine historische und gegenwärtige Beschreibung von Unternehmensdaten aus unterschiedlichen Quellen. Es werden einfache statistische Berechnungen, wie beispielsweise Durchschnitt, Median oder Standardverteilung, angewendet (Duan & Xiong, 2015). Als Darstellungsart bieten sich diverse Linien, Balken- oder Kreisdiagramme an. Beispielsweise können damit Forderungsausfälle im Vergleich zur Vorperiode oder Plan-Ist-Abweichungen einer Geschäftseinheit analysiert werden. Deskriptive Business Analytics wird auch als Synonym für BI verwendet (Fischer et al., 2020, S. 151).

(2) *Diagnostische Business Analytics* beantwortet die Frage „Weshalb ist ein Ereignis eingetreten beziehungsweise tritt gerade ein?“. Zur Identifikation der Ursachen sind Annahmen anhand von Abhängigkeiten oder Erfahrungen notwendig (Fischer et al., 2020, S. 152). Im Vergleich zu (1) werden statt trivialen statistischen Werkzeugen meist komplexere Methoden, etwa aus dem Bereich der Mustererkennung, angewendet (Duan & Xiong, 2015). Allerdings sind vollautomatisierte Analysen im Kontext von explorativen Fragestellungen oftmals nicht zielführend, da vorab meist nicht klar ist, nach was überhaupt gesucht werden soll. Als sinnvoller Kompromiss setzt an dieser Stelle Visual Analytics an, bei der beispielsweise ein Klassifizierungsalgorithmus Muster in Daten analysiert und mithilfe eines Diagramms - etwa einer Heatmap - visualisiert. Da jene visuelle Zusammenhänge für einen Computer schwer erkennbar sind, werden Schlussfolgerungen menschlicher Intelligenz überlassen (Ereth & Kemper, 2016).

(3) *Prädiktive Business Analytics* beantwortet die Frage „Was wird geschehen?“ (Fischer et al., 2020, S. 152). Hierzu werden mathematisch statistische und Machine Learning-Verfahren, wie Regression, Bayessche Statistik, Entscheidungsbäume, neuronale Netze, Support Vector Machines oder Nearest Neighbour-Verfahren angewendet, um Trends, Beziehungen und Affinitäten zu erkennen (Duan & Xiong, 2015). Ein beispielhafter Anwendungsfall hierfür wäre, im Vertragswesen Verhaltensmuster abgekündigter Kunden zu untersuchen, um bei aktiven Kunden eine Kündigungswahrscheinlichkeit aufzustellen (Gluchowski, 2016).

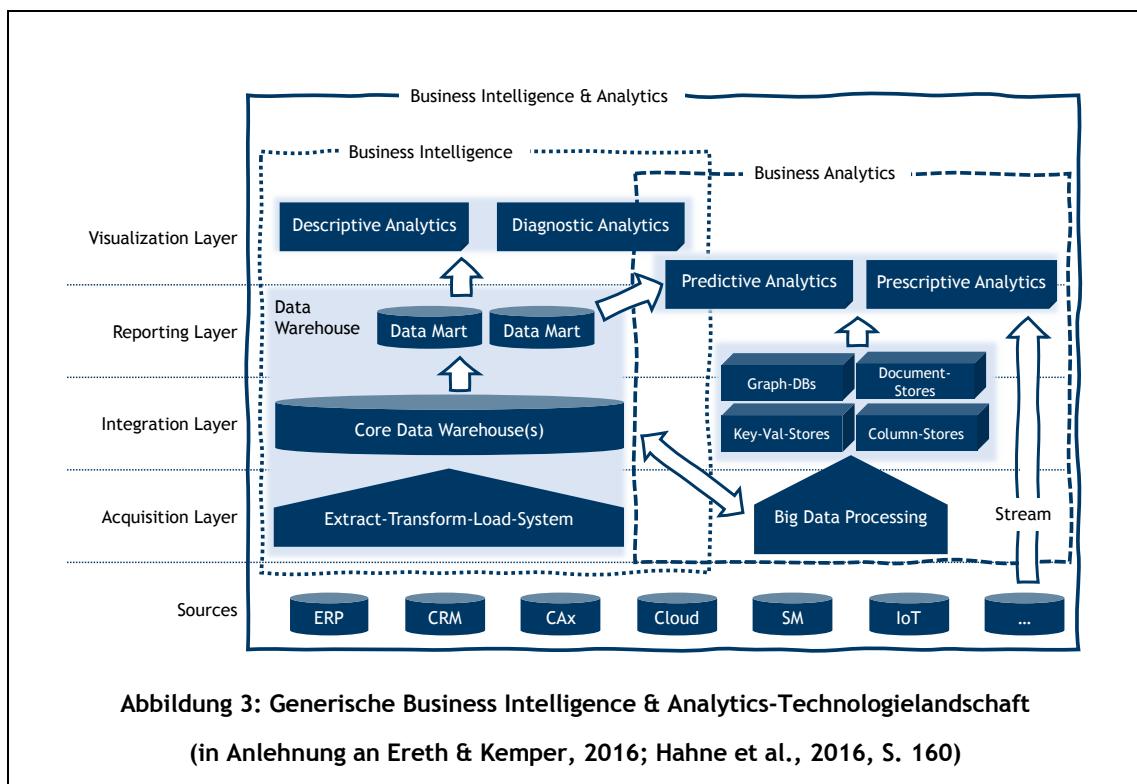
(4) *Präskriptive Business Analytics* beantwortet die Frage „Was ist zu tun?“. Hierbei werden Vorhersagemodelle mit semantischen Regeln und Simulations- und Optimierungsverfahren kombiniert (Ereth & Kemper, 2016). Anknüpfend an das Beispiel aus (3) würde dem betreuenden Mitarbeiter im Hinblick einer voraussichtlichen Kündigung eines Kunden entsprechend eine aktive Handlungsempfehlung zur zielführendsten Kundenrückgewinnungsaktion gegeben.

Zusammenfassend lässt sich also festhalten: Klassische BI, wie in Kapitel 2.1 definiert, ist als Teilbereich innerhalb von Business Analytics einzuordnen, der vornehmlich Methoden aus dem Bereich der deskriptiven und diagnostischen Business Analytics anhand strukturierter historischer Daten anwendet.

2.3 Technologische Perspektive eines Business Intelligence & Analytics Systems
Im Hinblick auf die beschränkten Ressourcen eines KMU werden aktuelle System-Architekturen in mehrfacher Hinsicht mit neuen Anforderungen konfrontiert. Gerade KMU sind volatilen Märkten gegenüber empfindlicher und ein schnelles Reagieren auf ein dynamisches Umfeld ist unumgänglich (Llave, 2019). Nebst klassischen Anforderungen an die Kostenstruktur einer initialen Konzeption und Implementierung eines BI & A-Systems steht also insbesondere eine agile, dynamische und kontinuierliche Anpassungs- und Weiterentwicklungsmöglichkeit im Fokus. Benötigt und erforderlich sind Architekturen und Konzepte, die ein schnelles Reagieren gestatten und somit die Zeitspanne bis zum produktiven Einsatz einer BI & A-Anwendung drastisch verringern. Eine konzeptionelle Antwort auf entsprechende Anforderungen sind Architekturen, die nach einem Mehrschichtenprinzip aufgebaut sind (Hahne, 2014, S. 32). Zudem ist auch die Entwicklung von BI hin zu Business Analytics in der Architektur zu berücksichtigen. Hierbei spielt die Erweiterung etablierter BI-Systeme um zahlreiche Datenquellen und -formen sowie analyseorientierter Datenbereitstellungssysteme eine wichtige Rolle (Ereth & Kemper, 2016).

In Abbildung 3 ist zur Veranschaulichung eine generische BI & A-Landschaft, wie sie in vielen Unternehmen vorgefunden werden kann, aufgezeigt. Im Acquisition Layer werden Rohdaten aus verschiedenen Quellen aufgenommen, transformiert und harmonisiert, um sie im Integration Layer zu persistieren. Im Reporting Layer erfolgt eine anwendungsfallspezifische, häufig auf konkrete betriebswirtschaftliche Anforderungen abgestimmte Aggregation der Daten. Ergebnisse können in dedizierten Data Marts gespeichert werden, um diese abschließend über das Visualization Layer an den Endanwendenden auszuspielen. Je nach konkreter Unternehmensdomäne kommen durchaus unterschiedliche Schichtenaufteilungen zum Einsatz. So könnte der Visualization Layer und Reporting Layer durch ein IT-System vereint sein. Das Schichtenprinzip an sich ist also als logische Klammer zu sehen, was unternehmensindividuell ausgeprägt ist (Hahne, 2014, S. 32). Im Bereich Business Analytics verschwimmt die klare Schichtenaufteilung etwas. So werden durch Big Data Processing-Systeme, wie beispielsweise Hadoop, heterogene Daten im Big Data-Umfeld durch hoch parallelisierte Systeme verarbeitet. Jene Systeme sind jedoch nicht als Substitute des DWHs zu betrachten, sondern als Ergänzung für spezielle Problemstellungen (Ereth & Kemper, 2016).

Da sich hierbei BI als fundamental zu Business Analytics verhält und erstere ausschlaggebend zur Beantwortung der Forschungsfrage ist, werden im Folgenden nun die Konzepte der einzelnen Schichten im Bereich BI näher erläutert. Für Ausführungen des Business Analytics-Bereichs ist auf Chen et al. (2012), Duan & Xiong (2015), Ereth & Kemper (2016), Gökalp et al. (2017) und Z. Sun et al. (2018) zu verweisen.



2.3.1 Acquisition Layer - Extract, Transform, Load

Im Allgemeinen besteht das Acquisition Layer aus einem sogenannten ETL-System, in welchem nacheinander die im Namen enthaltenen Vorgänge als Extract-Transform-Load-Prozess in regelmäßigen Zeitintervallen durchgeführt werden. Dieses agiert als Schnittstelle zwischen den Quellsystemen und dem DWH (S. Chaudhuri & Dayal, 1997). Das ETL-System besteht in der Regel wiederum aus einer Staging Area, instanzierten Datenstrukturen, einer Abfolge an Prozessen, die auf den importierten Daten ausgeführt werden und einem sogenannten Corporate Staging Memory (Hahne, 2014, S. 34).

(1) *Extraktion:* Im ersten Schritt werden relevante Daten aus den Quellsystemen gelesen und in der Staging Area gespeichert. Ab diesem Zeitpunkt sind diese nun Teil des DWHs, benötigen aber noch entsprechende Transformationen und Harmonisierungen, um an das nächste Layer weitergegeben werden zu können (Kimball & Ross, 2013, S. 19). Die Datenrepräsentationsform der Staging Area ist im Allgemeinen relationaler Art, bedarf also einer normalisierten Modellierung auf konzeptioneller Ebene beispielsweise anhand eines Entity-Relationship-Modells. Datenquellen werden weiter in logische Liefereinheiten unterteilt, wobei jede Lieferstruktur eine gleichartige Repräsentation in der Staging Area vorzuweisen hat. Lieferungen werden mit einem eindeutig identifizierbaren Schlüssel, um Metadaten wie etwa Zeitpunkt, Quelle, Ziel oder Inhaltskategorisierung erweitert und in der korrespondierenden Relation der Staging Area gespeichert (Hahne, 2014, S. 35).

(2) *Transformation:* Nach dem Staging werden auf Grundlage der Daten diverse Transformationen mit dem Ziel einer generellen Harmonisierung der extrahierten Daten ausgeführt. Hierbei werden unter anderem Domänenkonflikte behandelt, Rechtschreibfehler korrigiert, doppelte oder fehlende Werte eliminiert, redundante Quellen zusammengeführt oder bei numerischen Werten Konvertierungen in Standardformate vorgenommen. Dieser Vorgang wird auch Cleansing genannt und sorgt für eine einheitliche Datenqualität. Etwaige Änderungen können für diagnostische Zwecke weiterhin als Metadaten mitgegeben werden (Kimball & Ross, 2013, S. 20).

(3) *Laden:* Der letzte Schritt des ETL-Prozesses besteht aus der Übergabe der vorbereiteten Daten an das Core DWH. Hierbei herrscht Uneinigkeit, inwiefern die Relationen während des Ladens in eine niedrigere Normalform (NF) überführt werden sollten (Kimball & Ross, 2013, S. 20). Eine Abwägung der Möglichkeiten findet im anknüpfenden Kapitel statt, da es vorab noch einer Einführung in Grundlagen des DWH-Designs bedarf.

Um eine erneute Extraktion aus den Quellsystemen zu vermeiden, kann im Corporate Staging Memory eine Extraktionshistorie aufgebaut werden, in der die harmonisierten Daten selbst und sämtliche Metadaten persistiert werden. Teilweise ist ein Zugriff auf

Quelldaten nicht mehr möglich, da gegebenenfalls bereits eine Löschung stattgefunden hat oder ein solcher Zugriff nicht zeitnah umzusetzen wäre (Hahne et al., 2016, S. 159). Das Corporate Staging Memory muss dabei nicht als dedizierte Speicherlösung implementiert sein, sondern kann durchaus mittels der ursprünglichen Staging-Tabellen erfolgen (Hahne, 2014, S. 34).

Ein weiterer wichtiger Punkt und zugleich eine große Herausforderung im ETL-Prozess ist die Umsetzung eines inkrementellen Load-Ansatzes. Neben dem sogenannten Full Load, welcher bei jeder Ausführung des ETL-Prozesses alle Daten aus der Quelle erneut verarbeitet und vorhandene Daten im DWH ersetzt, wird mittels eines Incremental Loads nur der Unterschied zwischen dem Datenstand des Ziels und der Quelle verarbeitet und geladen. Dies spart nicht nur wertvolle Verarbeitungszeit, sowohl auf der Quell- als auch auf der Zielseite, sondern ermöglicht zusätzlich noch eine Historisierung von Änderungen, auf welche im nachfolgenden Kapitel näher eingegangen wird. Als Nachteil des Incremental Loads ist die Komplexität des ETL-Prozesses zu sehen, was ebenfalls auch den nachfolgenden Wartungsaufwand erhöht. Im Gegensatz hierzu gestaltet sich der Full Load in der Umsetzung recht trivial, da lediglich eine Kopie des aktuellen Datenstandes verarbeitet werden muss. Bei der Entscheidung, welcher ETL-Ansatz gewählt wird, ist zwischen Historisierungsbedarf, Performancekosten und Komplexität abzuwägen (Kimball & Ross, 2013, S. 512-518).

Im Regelfall werden ETL-Prozesse außerhalb der Hauptbenutzungszeit der operativen Systeme durchgeführt, um diese nicht durch zusätzliche umfangreiche Datenabfragen auszubremsen. Jene Verzögerung in der Aktualität der Daten des DWHs nennt sich Datenlatenz. Neuere Ansätze, wie die des sogenannten Realtime DWH setzen auf Stream-Technologien, um die Datenlatenz zu verringern (Wani & Raina, 2019).

2.3.2 *Integration Layer - Core Data Warehouse*

Die zentrale Aufgabe des Core DWHs als sogenannter Single Point of Truth besteht aus dem Persistieren gesammelter relevanter Unternehmensdaten auf einem festgelegten Niveau der Granularität und der erläuterten Gewährleistung von Datenqualität (Hahne et al., 2016, S. 162). Somit agiert es als zentrales Repozitorium, welches geschäftsprozessrelevante Daten eines Unternehmens für anknüpfende Schichten bereitstellt (March & Hevner, 2007). In der Praxis hat sich eine Granularität auf Transaktionsebene etabliert, da dies die größtmögliche Flexibilität bei der Auswertung im Nachgang bietet (Hahne et al., 2016, S. 162).

Im Schichtenmodell kommen unterschiedliche Datenmodellierungsparadigmen zum Tragen. Diese divergieren von Layer zu Layer. Im Integration Layer hat sich insbesondere die sogenannte Star-Schema-Modellierung durchgesetzt (Hahne, 2014, S. 251). In

der Literatur werden zwar auch andere Core DWH-Modellierungsansätze, wie beispielsweise Modellierung nach der dritten NF, diskutiert, allerdings verringert dieser die Agilität bei nachträglichen Änderungen am Datenmodell, was sich negativ auf die Dynamik des BI & A-Systems auswirkt (Hahne et al., 2016, S. 171; Kimball & Ross, 2013, S. 16).

Datenmodelle in operativen Systemen sind auf das zügige Einfügen und Ändern von Werten hin optimiert. Dies wird durch die Verwendung von Normalisierung - mit dem Ziel die Datenredundanz zu minimieren - erreicht. Hierbei werden funktional abhängige Attribute auf separate Tabellen aufgeteilt, was die Anzahl der referenzierten Tabellen vervielfacht. Dies erhöht die Performance beim Einfügen und Ändern von Werten, da eine Änderung entsprechend nur an einer Stelle erfolgen muss. Je höher die NF eines Datenmodells ist, desto weniger Daten werden kohärent, sondern elementar, gespeichert. Um die eingangs an die Architektur gestellten Anforderungen im Hinblick auf Agilität, Flexibilität und Dynamik zu erfüllen, erweist sich eine Einfachheit im Datenmodell als kritischer Erfolgsfaktor (Moody, 2000). Dadurch kann der Anwendende, aber auch Software aus anknüpfenden Schichten intuitiv in den Daten navigieren, um schneller und effizienter Ergebnisse zu liefern. Das Star-Schema entwirft das Core DWH anhand zweier Arten von Tabellen:

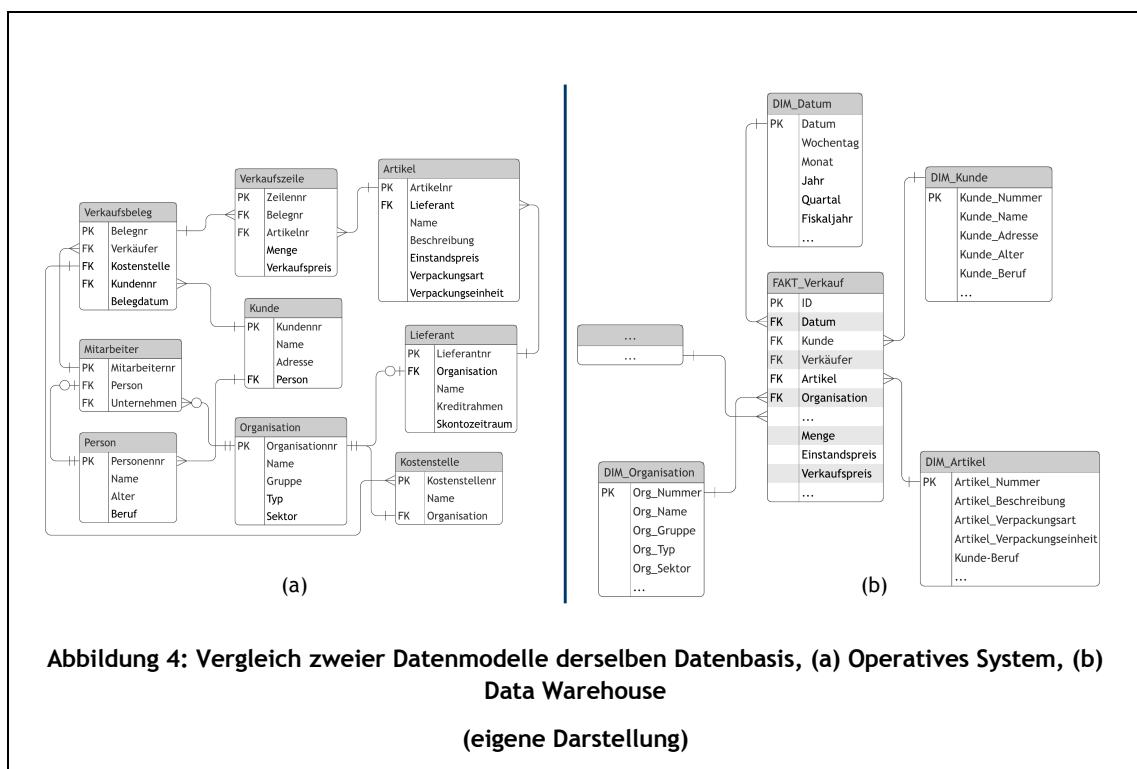
- (1) Fakten-Tabellen
- (2) Dimensions-Tabellen

Erstere beinhalten je Datensatz alle Informationen, die maßgeblich für eine konkrete Transaktion sind. Des Weiteren sind Fremdschlüssel (FK), referenzierend auf verschiedene Dimensionen der Transaktion, aufgeführt. Der Datensatz des „Faktes“, identifiziert über einen eindeutigen Primärschlüssel (PK), bezieht sich hierbei immer auf eine messbare Größe einer Transaktion innerhalb eines Geschäftsprozesses. Hierzu zählen beispielsweise die Menge oder der Preis einer verkauften Einheit eines Produktes. Da Transaktionen den Großteil aller erfassten Daten im Unternehmen ausmachen, ist es von Bedeutung, dass ein Fakt möglichst nur einmal im Core DWH gespeichert ist, sodass nicht mehrere Fakten-Tabellen für dieselbe Transaktion existieren. Weiter ist eine einheitliche, möglichst minimale Granularität der jeweiligen Faktentabelle wichtig. Ist dies gewährleistet, sind bereits einige Fehlerquellen - verursacht durch verfrühte Aggregation - von Vornherein ausgeschlossen. Fakten können additiv, semiadditiv oder nichtadditiv sein. Bei additiven Fakten, beispielsweise „Verkaufsbetrag in Euro“, kann unabhängig vom jeweiligen Granularitätsgrad oder der aktuellen Gruppierung der Daten anhand der Dimensionen eine Summe oder ein Durchschnitt gebildet werden. Semiadditive Fakten, beispielsweise Kontostände, lassen sich nicht anhand jeder Dimension aufaddieren. So würde eine zeitliche Addition eines Kontostandes keine sinnvolle

Summe ergeben, wohingegen eine Durchschnittsbildung weiter möglich ist. Nichtadditive Fakten, etwa Prozentangaben, lassen keinerlei Addition zu. Daher bietet es sich an, mögliche additive Komponenten der Prozentangabe weiterhin in der Faktentabelle abzuspeichern und Berechnungen erst im Nachgang durch anknüpfende Schichten durchzuführen (Kimball & Ross, 2013, S. 10-12; Moody, 2000).

Letztere, also Dimensions-Tabellen, beinhalten kontextuale Bestandteile einer konkreten Transaktion aus der Fakten-Tabelle und sollten möglichst die Fragen „Wer?“, „Wo?“, „Wann?“, „Was?“ und „Warum?“ beantworten. Weitergehende Referenzen werden durch die Transformation in eine niedrigere NF in die entsprechende Dimensionstabelle integriert. Hierbei ist es nicht ungewöhnlich, dass Dimensionstabellen eine hohe Anzahl an Attributen aufweisen. Diese Attribute sind essentiell für Auswertungen, da anhand jener sämtliche Gruppierungen, Kennzeichnungen oder andere Einschränkungen der Faktentabelle getroffen werden. Bezeichnungen sollten dabei stets einheitlich und selbsterklärend sein. Dies ermöglicht eine einfache Navigation durch den Endanwendenden und erspart ein nachträgliches Umbenennen im Visualization Layer (Kimball & Ross, 2013, S. 13-16; Moody, 2000).

Abbildung 4 veranschaulicht den Unterschied der Datenmodelle eines operativen Systems (a) im Vergleich zur Informationsrepräsentation innerhalb eines DWH (b) anhand einer exemplarischen Verkaufstransaktion. Gleichwohl auffällig ist der hohe Normalisierungsgrad des operativen Systems. Zwischen den Tabellen bestehen teils mehrstellige zyklische Beziehungen und Hierarchiestufen. Im Gegensatz hierzu steht die Fak-



tentabelle bei (b) mit maximal einer weiteren Hierarchie in Beziehung. Mehrfachverschachtelungen wie in (a) existieren hingegen nicht. Des Weiteren fällt die eindeutige und intuitive Bezeichnung der jeweiligen Attribute bei (b) auf, um eine erleichterte Navigation ohne weitere Umbenennungsschritte der im DWH anknüpfenden Schicht zu ermöglichen. Eine Überführung der Dimensionstabellen in eine niedrigere NF erfolgt anhand des Ansatzes nach Husemann et al. (2000), bei welchem in einem iterativen Prozess Komplexitäten aus der Datenhierarchie entfernt werden.

Attribute in Dimensionstabellen sind in den meisten Fällen von statischem Charakter. Es existieren aber auch Szenarien, in welchen sich diese im zeitlichen Verlauf verändern können. Beim Umzug eines Kunden ändert sich entsprechend auch seine Anschrift, was es gegebenenfalls im DWH zu erfassen und zu historisieren gilt. Dies könnte etwa für Auswertungen im Sinne von „Umsätze nach Regionen“ relevant sein, bei welcher die Umsätze des Kunden korrekt zugeordnet werden sollten. Das erläuterte Phänomen wird in der Literatur als sogenannte „Slowly Changing Dimension“ bezeichnet. Im Wesentlichen werden nach Kimball & Ross (2013) vier Unterscheidungen des Phänomens getroffen. Attribute von Typ 0 ändern sich niemals und werden auch als Originale bezeichnet. Als Beispiel aufzuführen wären hierbei das Geburtsdatum eines Kunden und generell sämtliche die Datumstabelle referenzierende Attribute. Bei Typ 1 werden Änderungen von Attributen im DWH trivialerweise überschrieben. Entsprechend findet keine Historisierung der Änderungen statt. Im Gegensatz zu Typ 1 historisiert Typ 2 Änderungen über den Verlauf der Zeit mittels einer neu einzufügenden Zeile. Über zusätzliche Attribute könnten dann Informationen über den Gültigkeitszeitraum des Eintrages geführt oder ein Aktivstatus signalisiert werden. Da in der Dimensionstabelle nun aber nicht mehr auf einen eindeutigen PK referenziert werden kann, ist neben diesem auch ein synthetischer sogenannter Surrogateschlüssel (SK) einzuführen, welcher ebenfalls in der Faktentabelle aufzuführen ist. So kann mittels des SKs präzise nach einzelnen Gültigkeitszeiträumen gefiltert werden. Typ 3 folgt einem ähnlichen Ansatz, allerdings werden hier Änderungen über ein weiteres Attribut nachverfolgt. Übertragen auf das hiesige Kundenbeispiel würden die Attribute „Adresse_Alt“ und „Adresse_Aktuell“ existieren. In der Praxis hat sich Typ 3 allerdings aufgrund der Steigerung der Attributanzahl und der damit einhergehenden Unübersichtlichkeit nicht weiter etabliert.

Der Integration Layer kombiniert also durch die Bereitstellung der gesamten relevanten Unternehmensdaten - transaktionaler und kontextueller Art, in detaillierter und integrierter Form - sowohl eine Sammel- und Integrationsfunktion, als auch eine Distributionsfunktion für anknüpfende Schichten.

2.3.3 *Reporting Layer - Data Mart*

Durch die Einführung eines Reporting Layers und dazugehörigen Data Marts, welche unabhängig vom Core DWH sind und mit Kopien der Daten des Core DWH arbeiten, wächst die Flexibilität der Architektur enorm (Moody, 2000). Da eine persistierte Historisierung der Daten bereits im Integration Layer erfolgt, können Data Marts flexibel an die jeweiligen domänenspezifischen Anforderungen des Endanwendenden angepasst werden. Das Reporting Layer kapselt somit domänenspezifische Logiken (Hahne et al., 2016, S. 162).

2.3.4 *Visualization Layer*

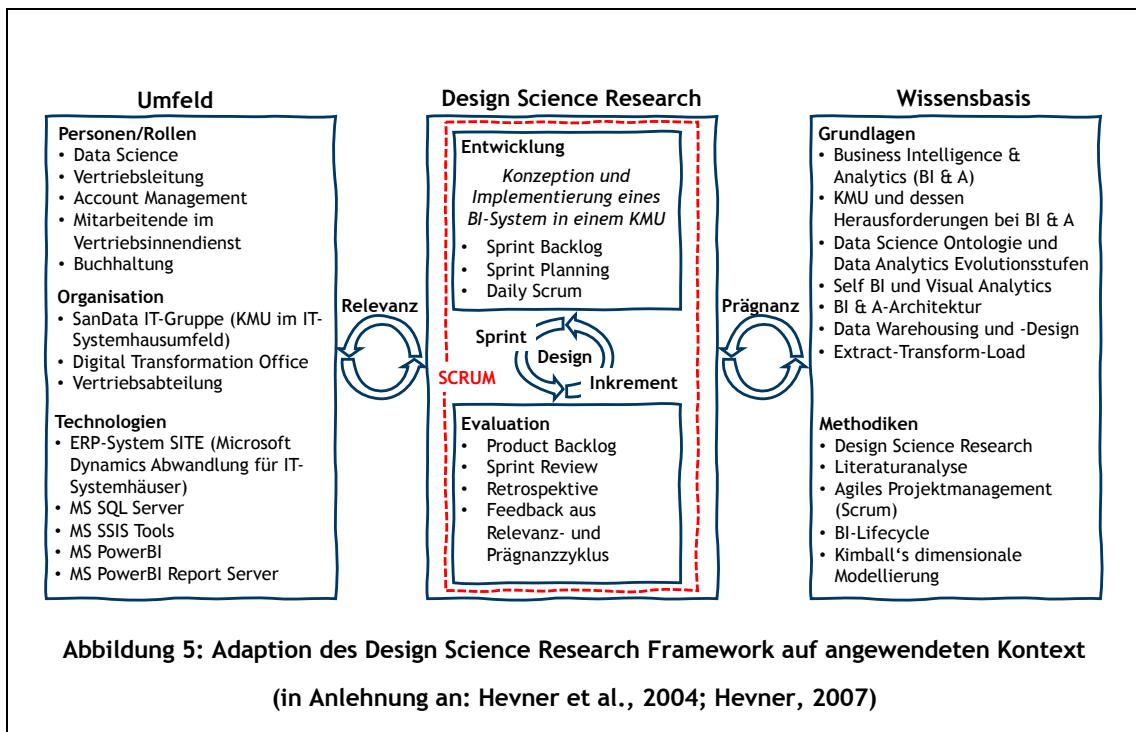
Die letzte Schicht agiert als Frontend, um Daten für den Endanwendenden anschaulich und leicht interpretierbar darzustellen. Während klassische BI den Fokus eher auf ETL-Prozesse und statische Berichte legt, ist ein Trend hin zur selbstgesteuerten, interaktiven, visuellen Datenerkundung zu beobachten (Obeidat, 2015; Stodder, 2015). Bis heute haben zahlreiche Unternehmen, darunter insbesondere KMU, die Möglichkeiten der Datenanalyse aufgrund beschränkter Ressourcen längst nicht ausgeschöpft. Die Herausforderung besteht nicht mehr in der Beschaffung auszuwertender Daten. Sie besteht vielmehr in der Beantwortung der Frage, wie in möglichst kurzer Zeit, entscheidungsrelevante Rohdaten für ein breites Adressatenspektrum von BI-Anwendenden und Entscheidungstragenden aufbereitet werden können (Kohlhammer et al., 2016, S. 304-307). Als Antwort kommt Self-Service-BI, auch Self-BI genannt, in Betracht. Demnach besteht das Ziel darin, dem Endanwendenden mehr Rechte, Möglichkeiten und Verantwortung einzuräumen, indem in Sachen Auswertung die Abhängigkeiten von der IT-Abteilung bei der Datenbeschaffung reduziert werden (Hoffjan & Rohe, 2018). Hierdurch hat der Endanwendende die Perspektive, selbstgesteuert Dashboards, Performanceübersichten oder Auswertungen zu erstellen (Stodder, 2015).

3. Forschungsmethode

Die vorliegende Arbeit folgt dem Ansatz der Design Science Research nach Hevner et al. (2004), erweitert um die drei Forschungszyklen Relevanz, Design und Prägnanz aus Hevner (2007). Bei Design Science handelt es sich hierbei um einen problemlösungsorientierten Ansatz mit Ursprung in der Ingenieurswissenschaft. Im Fokus stehen die Erstellung von Prototypen oder Konzepten und deren Einfluss auf Prozesse oder Organisationen (Lindner, 2020). Das primäre Ziel der Arbeit ist die Konzeption und Implementierung eines BI-Systems in einem mittelständischen Unternehmen, was ebenfalls das instanzierte Artefakt darstellt.

3.1 Design Science Research

Ursprünglich repräsentiert Design Science Research das strukturierte Vorgehen, ein Artefakt anhand einer konkreten Problemstellung zu konstruieren, um eine bestehende Forschungslücke zu schließen. Im Entstehungsprozess der vorliegenden Arbeit wechseln sich kontinuierlich die Instanziierung des implementierten IT-Artefakts mit einer Evaluation desselbigen - einerseits in der betreffenden Umwelt, andererseits mit der wissenschaftlichen Seite - ab. Hierbei entwickelt sich das IT-Artefakt inkrementell (Gregor & Hevner, 2013). Mit der Erweiterung um die drei Zyklen in Hevner (2007) verdeutlicht der Autor den inkrementellen beziehungsweise iterativen Charakter des Rahmenwerks. Zusätzlich wird der enge Zusammenhang zwischen den Bereichen Umfeld, Design Science Research-Aktivität und Wissensbasis betont. Der Relevanzzzyklus tauscht bezüglich der Design Science Research-Aktivität Feedback mit dem betreffenden Umfeld aus und prüft damit, ob Ergebnisse des IT-Artefaktes einen relevanten Einfluss auf das definierte Umfeld haben. Im Prägnanzzzyklus findet ein kontinuierlicher Abgleich mit der wissenschaftlichen Basis statt, während im Designzyklus das IT-Artefakt inkrementell instanziert und verbessert wird. Abbildung 5 auf folgender Seite stellt die inhaltliche Einordnung der vorliegenden Arbeit in das Design Science Research-Framework dar. Hierzu wird in Kapitel 4 ein BI-System für ein KMU konzeptioniert und implementiert. Darunter fallen der Entwurf eines dedizierten DWHs, ETL-Prozesses und die intuitive Visualisierung der Ergebnisse. Aufgrund des limitierten Umfangs der Arbeit beschränkt sich die Umsetzung auf einen konkreten Anwendungsfall aus dem Verkaufsbereich. Die Herausforderung besteht in einer möglichst flexiblen Gestaltung des DWH, um im Nachgang für anknüpfende Anwendungsfälle leicht erweiterbar zu bleiben und somit den Dynamiken des Umfelds im Bereich KMU gewachsen zu sein.



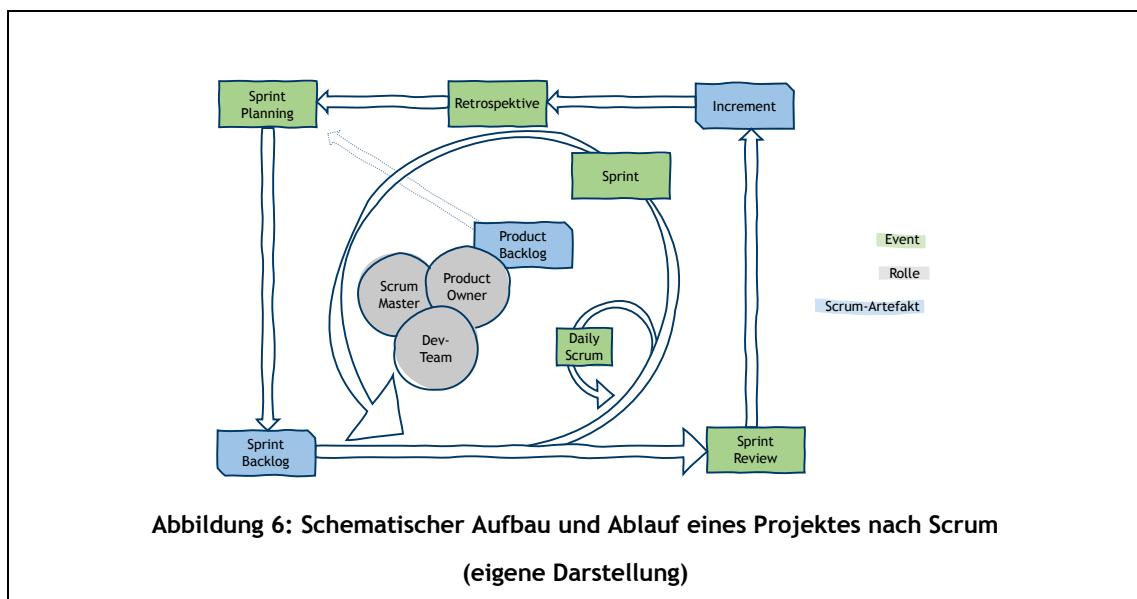
Zur Schaffung einer theoretischen Grundlage, werden in Kapitel 2 Konzepte und Entwicklungen von BI aufgezeigt, sowie technologische Ansätze wie Data Warehousing, ETL und dimensionale Modellierung nach Kimball & Ross (2013) erläutert. Hierzu sind primär die Datenbanken „Scopus“ und „Web of Science“ mit den Suchkombinationen

- ((Business AND (Intelligence OR Analytics)) AND (Data AND (Warehouse OR Warehousing)))
- "Business Intelligence" AND ("small and medium (enterprises OR enterprise)" OR SME OR SMEs)

abgefragt worden. Im Anschluss wurden die Suchergebnisse jeweils nach deren Zitierhäufigkeit sortiert, um möglichst fundiertes und in der Forschung etabliertes Wissen abzuschöpfen. In einem weiteren Schritt wurde die Auswahl anhand der jeweiligen Titel und Schlüsselwörter thematisch eingeschränkt. Zur fortführenden Analyse ist jene Auswahl nun auf die Frei-Text-Verfügbarkeit begrenzt worden. Im letzten Schritt wurden mittels einer Rückwärts- und anschließender Vorwärtssuche die Inhaltsverzeichnisse der Auswahl, respektive die Artikel, welche die Auswahl zitieren, untersucht. Dies führte wiederum zu einer Erweiterung an zu analysierender Literatur. Im Fortgang der Arbeit wurden aus den Ergebnissen nun die Charakteristiken von BI, die Ontologie im Bereich Data Science und Grundlagen zu DWH, ETL und Modellierung extrahiert.

3.2 Agiles Projektmanagement

Die praktische Umsetzung im Unternehmen erfolgt mittels des agilen Projektmanagement-Frameworks Scrum. Hierbei handelt es sich um ein prozessuales, agiles Framework, mit welchem komplexe Produkte effizient entwickelt, ausgeliefert und betreut werden können. Mittels des Frameworks lassen sich Vorteile in zweierlei Hinsicht ausschöpfen. Zum einen werden nach kurzer Zeit valide Ergebnisse geliefert, was unter anderem eine deutliche Kostenersparnis für das betreffende KMU bedeutet. Zum anderen integriert sich die agile Vorgehensweise nahtlos in den Designzyklus der verwendeten Forschungsmethode (vgl. roten Rahmen in Abbildung 5, S. 21). Um ein Verständnis für die Vorgehensweise der inkrementellen Instanziierung des IT-Artefakts zu schaffen, ist im Folgenden die Grundidee des Scrum-Frameworks kurz erläutert. Für weiterführende Erklärungen und detaillierte Beschreibungen bezüglich der Rollen, Artefakte und Events des Frameworks ist auf Schwaber & Sutherland (2017) zu verweisen. In Abbildung 6 ist der Ablauf eines Projektes nach Scrum schematisch veranschaulicht.



Das Framework besteht aus drei Rollen, vier formalen Veranstaltungen und drei Artefakten. Als Rolle ist der Product Owner zum einen für die Maximierung des Nutzens des Inkrement, welches das Development Team entwickelt, verantwortlich, zum anderen verwaltet er das Product Backlog, welches mehrere Product Backlog Items enthält. In ihrer Gesamtheit repräsentieren Product Backlog Items alle Anforderungen und Funktionen des zu entwickelnden Produktes, im Product Backlog sortiert nach Kundenpriorität. Dem Scrum Master kommt die Aufgabe zu, den agilen Projektgedanken nach Scrum, wie ihn der Scrum-Guide nach Schwaber & Sutherland (2017) empfiehlt, in der Organisation zu vertreten und zu gewährleisten, so dass jeder Beteiligte die Theorien, Praktiken, Regeln und Werte versteht und befolgt. Für jeden Sprint entwickelt das Development Team eine lauffähige Version des Produktes, das sogenannte Inkrement.

Im Falle der vorliegenden Arbeit ist das Inkrement mit dem IT-Artefakt nach Design Science Research gleichzusetzen. Das Projekt an sich ist in mehrere Sprints unterteilt. Zu Beginn jedes Sprints findet ein Sprint Planning statt, in dem das Development Team mehrere Product Backlog Items auswählt, die als Sprintziel, nach gemeinsamer Entscheidung von Product Owner und Development Team, im anknüpfenden Sprint zu erreichen sind. Die ausgewählten Product Backlog Items werden während des Sprints vom Development Team im Sprint Backlog verwaltet. Im Sprint setzt das Development Team die gewählten Product Backlog Items selbstverantwortlich um und spricht sich im täglichen Daily Scrum, mit einem maximalen Zeitrahmen von 15 Minuten, intern ab. Am Ende eines Sprints wird im Sprint Review vom Scrum Team und dem Auftraggebenden überprüft, ob die Anforderungen an das jeweilige Product Backlog Item, nach der gemeinsam festgelegten „Definition of Done“ erfüllt wurden. In der Retrospektive werden anschließend zwischenmenschliche Aspekte diskutiert, um die Zusammenarbeit im Team zu verbessern. Dieser Prozess wiederholt sich iterativ und sorgt damit für ein stetiges Wachstum des Inkrements von Sprint zu Sprint.

Die agile Vorgehensweise nach Scrum lässt sich in der praktischen Umsetzung ideal in den Designzyklus des Design Science Research-Frameworks integrieren, indem ein Rahmen um den mittleren Design Science Research-Bereich gebildet wird. Dies ist in Abbildung 5 auf Seite 21 durch die rote Hervorhebung veranschaulicht. Ziel des Designzyklus‘ ist eine kontinuierliche Entwicklung des Artefakts, stetig verbessert durch die Adaption des Feedbacks aus der Evaluationsphase. Jener Kreislauf aus Entwicklung, Evaluation und kontinuierlicher Verbesserung wird durch die agile Projektmanagementmethode, wie erläutert, verwirklicht. Somit findet der Designzyklus in Form der praktischen Umsetzung mittels Scrum Anwendung.

4. Artefakt

Im Folgenden wird nun die Vorgehensweise bei der Instanzierung des IT-Artefaktes näher beschrieben. Das IT-Artefakt ist hierbei das zu konzeptionierende und zu implementierende BI-System. Zur Einordnung des Projekts in den betrieblichen Kontext San-Datas wird vorab in Kapitel 4.1 die Ausgangssituation, sowohl technisch als auch perspektivisch dargelegt. Hierzu ist eine nähere Betrachtung des Umfelds (vgl. Abbildung 5, S. 21) erforderlich. Daran anknüpfend wird in Kapitel 4.2 auf die aus dem Umfeld abgeleiteten Ausprägungen des Scrum-Projektes und deren zeitliche Entwicklung während des Projektverlaufes eingegangen. Kapitel 4.3 beschreibt abschließend das finale Inkrement des IT-Artefaktes. Eine besondere Herausforderung stellt die Vorgabe dar, dass für die Umsetzung des BI-Systems möglichst bereits vorhandene Ressourcen, in Form von Werkzeugen, Programmen und Infrastruktur genutzt werden sollten, um die Kosten für jenes überschaubar zu halten.

4.1 Ausgangssituation und vorbereitende Maßnahmen

Initialen und damit maßgebenden Einfluss auf das zu entwickelnde IT-Artefakt hat das Umfeld, aus und mit welchem essentielle Anforderungen abgeleitet, respektive ermittelt werden. Nach dem Design Science Research-Framework, wie in Kapitel 3.1 erläutert, ist das Umfeld des IT-Artefakts in Organisation, Personen und Technologie zu unterteilen. In Tabelle 1 ist das konkrete Umfeld des Projektes weiter aufgeschlüsselt. Als übergeordnete Organisation wird bereits in Kapitel 1.2 die SanData IT-Gruppe identifiziert und vorgestellt.

	<i>Organisation¹</i>	<i>Personen/Rollen¹</i>	<i>Technologien¹</i>
SanData IT-Gruppe	Vertriebsabteilung	Vertriebsleitung	ERP-System SITE MS SQL Server MS SSIS Tools MS PowerBI MS Report Server
		Regionalleitung	
		Account Management	
		Mitarbeitende im Vertriebs- innendienst	
	Digital Transformation Office	Data Science	
		Application Management	
		Prozess-Management	
		Finanzabteilung	
		Buchhaltung	

¹Redundante Werte sind zur besseren Übersichtlichkeit gruppiert dargestellt.

Tabelle 1: Umfeld des IT-Artefakts nach Organisation, Personen/Rollen und Technologien
(eigene Darstellung)

Eine weitere Aufschlüsselung in teilhabende Abteilungen, Personen beziehungsweise Rollen und verwendete Technologien ergibt sich aus den vom internen Auftraggeber den gestellten Anforderungen an das BI-System. Konkreter Anlass des Projektes ist ein

von der Vertriebsleitung angemeldeter Bedarf an einer automatisierten Auswertungsmöglichkeit diverser vertriebsorientierter Kennzahlen. In monatlichen Geschäftsleitungsterminen kommt der Vertriebsleitung die Aufgabe zu, über die vertriebliche Entwicklung in der vergangenen Periode zu berichten und auf etwaige Trends einzugehen. Zum aktuellen Zeitpunkt werden hierzu seitens der Vertriebsleitung händisch Umsatzdaten aus mehreren Quellen zusammengetragen, in einem Tabellenkalkulationsprogramm konsolidiert, ausgewertet und statisch in Form einer Präsentationsfolie dargestellt. Dies bedeutet nicht nur einen ständigen hohen manuellen Aufwand für eigentliche Routineaufgaben, sondern birgt auch ein gewisses Fehler- und Unschärfepotential in sich. Ziel des BI-Systems ist somit die automatisierte Bereitstellung korrekter vertrieblicher Kennzahlen. Hierzu sind im Rahmen eines Vorgesprächs seitens der Vertriebsleitung folgende zur selbstgesteuerten und dynamischen Auswertung bereitzustellende Kennzahlen festgelegt worden:

1. Umsatz und Deckungsbeitrag, je Monat (ACT) und kumuliert im laufenden Jahr (YTD)
2. Umsatz und Deckungsbeitrag, Year-over-year (YOY)
3. Umsatz und Deckungsbeitrag, Year-to-plan (YTP)
4. Auftragsbestand, aktueller Monat (ACT)
5. Angebotsbestand, aktueller Monat (ACT)
6. Verkaufschancenbestand, aktueller Monat (ACT)

Es gilt nun, sowohl die Anwendungsdomeine zu analysieren als auch die zur Umsetzung benötigte IT-Infrastruktur und Technologien zu prüfen, um diese gegebenenfalls anzupassen. Dies wird in den nachfolgenden Kapiteln dargelegt.

4.1.1 Organisation und Personen

Für eine Auswertung im Sinne des angemeldeten Bedarfs ist die vertriebliche Struktur des Unternehmens grundlegend, da sich hieraus unteranderem für eine spätere Analyse relevante Dimensionen ableiten. Die Bezeichnung SanData *IT-Gruppe* beruht auf keinem rechtlichen Konstrukt, sondern dient der einheitlichen Außendarstellung als Unternehmensgruppe mehrerer Gesellschaften in Deutschland und Österreich. Trotz der Gliederung in einzelne Gesellschaften obliegt die gesamt vertriebliche Verantwortung aller Gesellschaften der Vertriebsleitung. Die Gruppe bildet also eine Klammer über alle Gesellschaften hinweg.

Aus vertrieblicher Perspektive ist die SanData IT-Gruppe in mehrere Regionen unterteilt. Eine Region besteht dabei aus verschiedenen Standorten. Auf regionaler Ebene trägt eine Regionalleitung die Verantwortung, welche wiederum ein Team mehrerer Mitarbeitender im Account Management leitet. Administrative Tätigkeiten werden durch einen Vertriebsinnendienst je Standort übernommen.

Buchhalterisch entspricht ein Standort einer Kostenstelle, wobei jede Kostenstelle wiederum zu einer Gesellschaft gehört. Aus der dargelegten Struktur ergibt sich Tabelle 2, in welcher die vertriebliche Organisationsstruktur veranschaulicht ist. Die generischen Regionen „CCC“ und „SIT“ (abgeleitet vom jeweiligen Unternehmensnamen) sind hierbei außen vorgenommen, da deren Umsätze separat ausgewertet werden sollen.

<i>Land¹</i>	<i>Region¹</i>	<i>Standort¹</i>	<i>Kostenstelle¹</i>	<i>Gesellschaft¹</i>
AT	Austria Mitte	Linz	503 SDT KG LIN	SD Technology KG
		Klagenfurt	502 SDT KG KLG	
	Austria Ost	Wien	501 SDT KG W	SD Technology GmbH
			511 SDT GMBH W	
DE	Austria West	Innsbruck	504 SDT KG INB	SD Technology KG
	Baden-Württemberg	Heilbronn	061 PROTEAM HEB	PT Business Solutions GmbH
			016 SD EDV HEB	
	Bayern Nord	Nürnberg	011 SD EDV NBG	SD EDV-Systemhaus GmbH
		Ingolstadt	013 SD EDV ING	
	Bayern Ost	Regensburg	012 SD EDV RGB	
		Bayreuth	015 SD EDV BTH	
	Bayern Süd	Garching	021 SDS MUC	SD Solutions GmbH
	CCC		031 CCC MUC	
	Sachsen	Dresden	014 SD EDV DRS	SD EDV-Systemhaus GmbH
	SIT	Nürnberg	051 SIT NBG	SD IT-Trainingszentrum GmbH
AT	Data Technology	Garching	611 DATA TEC D MUC	Data Technology GmbH
		Wien	601 DATA TEC AT WIE	

¹Redundante Werte sind zur besseren Übersichtlichkeit gruppiert dargestellt.

Tabelle 2: Vertriebliche Organisationsstruktur der SanData IT-Gruppe

(eigene Darstellung)

Als weitere Stakeholder sind für das Projekt im Bereich der Organisation noch die Abteilungen Digital Transformation Office und die Finanzabteilung relevant.

Erstere ist im Unternehmen für zwei Kernaspekte zuständig. Zum einen wird seitens des Application Managements im Digital Transformation Office das ERP-System SITE betrieben, betreut und gewartet, zum anderen nimmt die Abteilung eine Rolle als Competence-Center für jegliche Form von Prozessoptimierungen und Auswertungen, SITE betreffend, ein. Hierzu leisten Fachkräfte, beispielsweise aus dem Bereich Prozess-Management oder Data Science, ihre Beiträge in interdisziplinären Teams. Hinsichtlich des Projektes ist das Digital Transformation Office folglich als umsetzende Abteilung zu sehen.

Letztere Abteilung übernimmt im Wesentlichen zwei Rollen. Auf der einen Seite ist die Finanzabteilung für eine Validierung der Auswertungsergebnisse aus fachlicher Perspektive zuständig („Sind die ermittelten Umsätze und Deckungsbeiträge korrekt?“). Anhand des Feedbacks wird das Inkrement qualitativ auf dessen Validität geprüft und

somit von Sprint zu Sprint verbessert. Auf der anderen Seite fungiert die Finanzabteilung auch als Datenquelle für das DWH. Dies hängt mit der Datenqualität von SITE zusammen. Zum aktuellen Zeitpunkt werden in SITE keine Korrekturbuchungen erfasst. Aufgrund dieser Tatsache führt die Finanzabteilung eine manuell gepflegte Liste an Korrekturbuchungen, um buchhalterisch die Synchronizität zwischen Monatsabschlüssen und Systemdaten zu gewährleisten. Die Berücksichtigung der Korrekturdaten im DWH ist also für die Gewinnung valider Auswertungsergebnisse von Umsätzen und Deckungsbeiträgen bedeutend. Jener doppelten manuellen Pflege soll zwar in Zukunft durch Prozessanpassungen und regulatorische Maßnahmen entgegengewirkt werden. Hiervon kann aber zum Projektstart noch nicht ausgegangen werden, sodass dies bei der Konzeption des DWH zu berücksichtigen ist.

4.1.2 *Technologien*

Als Microsoft (MS) Gold Partner ist die IT-Umgebung von SanData größtenteils auf Produkten des US-amerikanischen Software-Konzerns aufgebaut. So existieren Volumen-Lizenzverträge etwa im Rahmen von MS Office 365 oder MS Server. Der Gold-Partner-Status bietet außerdem die Möglichkeit eines teilweise kostenfreien Bezugs von Lizenzen. Zur Wahrung sensibler Kundendaten verfolgt SanData eine Strategie, die benötigte IT-Infrastruktur selber und auf eigenen Servern - auch „on-premise“ genannt - zu betreiben.

Für die zentrale Abwicklung und Administration von Geschäftsprozessen wird bei SanData gruppenweit das integrierte ERP-System SITE eingesetzt. Hierbei handelt es sich um eine speziell auf IT-Unternehmen angepasste Adaption des von MS entwickelten ERP-Systems „Dynamics NAV“. In SITE werden alle unternehmerischen Anwendungsberiche, angefangen von Kundenbeziehungen (z.B. Customer-Relationship-Management, Dienstleistungs-Management und Helpdesk-Support), über geschäftliche Prozesse (z.B. Rechnungswesen, Vertrags- oder Projektmanagement) bis hin zu warenwirtschaftlichen Aspekten (z.B. Einkauf, Verkauf und Lagerhaltung), abgebildet. Das System verwaltet alle Informationen an einem Ort, über im Hintergrund eingesetzte Datenbanken.

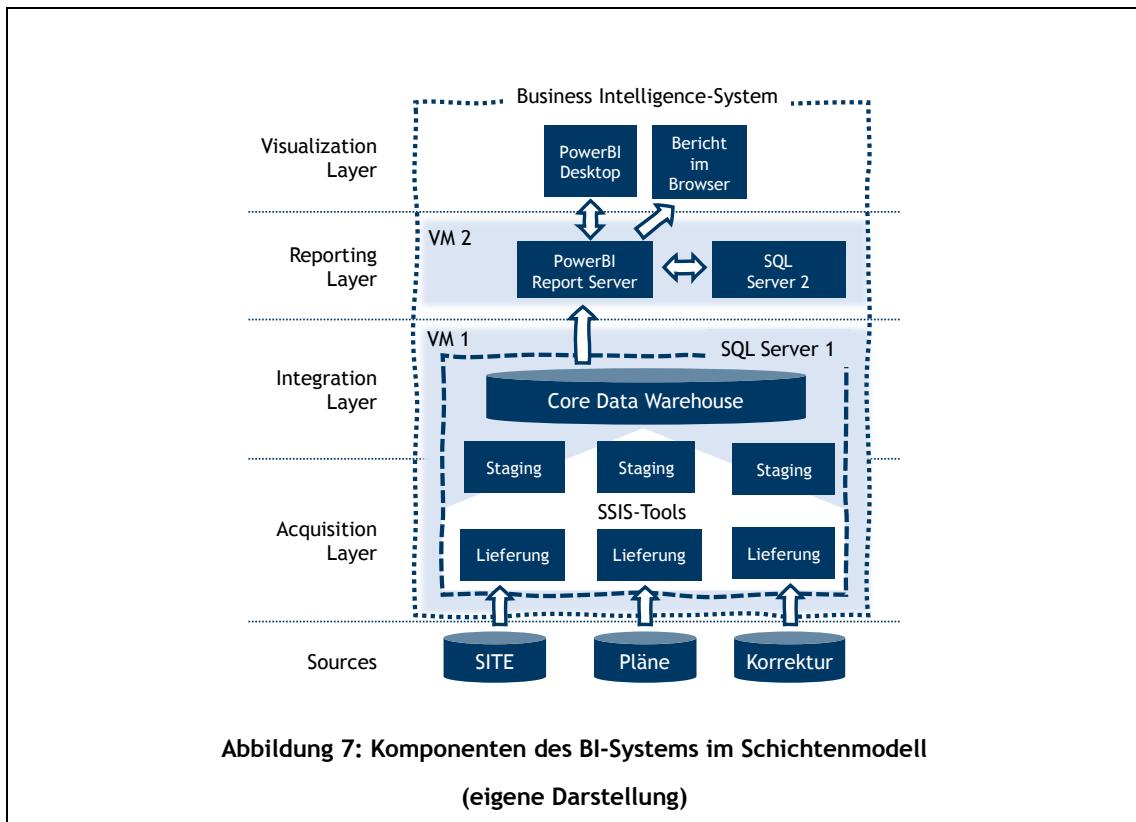
Aufgrund kostenfreier Kapazitäten einer Volumenlicenzierung für MS Server und MS SQL Server, ist eine Realisierung des DWHs in Form einer weiteren virtuellen Maschine (VM) naheliegend. Die Konfigurationen der Datenflüsse zwischen Live-System und DWH im Rahmen des zu erstellenden ETL-Prozesses werden mithilfe der MS SQL Server Integration Services (SSIS) umgesetzt. Hierbei handelt es sich um eine wesentliche Komponente des MS SQL Servers zur Erstellung von DWH-Anwendungen. Zudem arbeitet MS SQL Server mit der proprietären Erweiterung des SQL-Standards „Transact-SQL“, welche prozedurale Programmierung, Variablen und Funktionen in einer SQL-nahen Syntax

ermöglicht. Ein ETL-Prozess wird in der Entwicklungsumgebung MS Visual Studio und deren Erweiterung namens MS SQL Server Data Tools definiert. Diese ermöglicht eine intuitive grafische Modellierung der ETL-Strecken zwischen Live-System und DWH. Im Anhang ist zur Veranschaulichung ein Ausschnitt der Bedienoberfläche der Entwicklungsumgebung abgebildet (vgl. Anhang, Abbildung 17, S. 61). Anschließend werden einzelne modellierte Strecken als ausführbare Pakete - welche vereinfacht ausgedrückt eine Abfolge an SQL-Operationen kapseln - auf den SQL-Servern bereitgestellt und mittels einer Routine periodisch ausgeführt.

Weiter ist naheliegend, dass zur interaktiven und selbstgesteuerten Visualisierung die Entscheidung auf MS PowerBI gefallen ist. Mit PowerBI können Dashboards auf Basis verschiedener Datenquellen, hier im konkreten Falle des DWHs entwickelt werden. Neben PowerBI existiert auch alternative BI-Software, beispielsweise von Tableau oder Looker. Diese werden jedoch aufgrund des beschränkten Umfangs der Arbeit nicht näher behandelt. Die Konfiguration der Dashboards ist in erster Instanz kostenfrei über das Programm MS PowerBI Desktop möglich. Bezugnehmend auf die Schichtenstruktur aus Kapitel 2.3 ist PowerBI selbst aber als schichtenübergreifendes Werkzeug zu betrachten. Es dient nicht nur der reinen Visualisierung von Daten, sondern erfüllt auch Dienste des Reporting Layers. So können über das Programm auch Daten weiter separiert, transformiert und aggregiert werden. Ebenfalls besteht die Möglichkeit neue, speziell auf den jeweiligen Anwendungsfall des Dashboards optimierte, Datenmodelle zu definieren. Folglich kann die Funktionsweise eines Data Martes mittels PowerBI realisiert werden. Allerdings fehlen bei der reinen Desktop-Variante zentrale Verteilungsmöglichkeiten der erstellten Datenmodelle und der daraus entwickelten Dashboards. Bezuglich einer Bereitstellung der Datenmodelle und Dashboards für wiederkehrende Verwendung im Unternehmen ist seitens des Herstellers der Weg über die hauseigenen Cloud-Services (Azure) angedacht. Durch Azure werden zum einen sämtliche per PowerBI definierten Datenmodelle optimiert, berechnet und für wiederkehrende Verwendung zentral persistiert. Zum anderen werden über ein Webinterface einheitlich abrufbare interaktive Dashboards erzeugt. Allerdings würde die Nutzung des Azure-Dienstes im einfachen Falle für jeden Betrachtenden eine weitere monatliche Pauschale nach sich ziehen, außerdem müssten sensible Daten aus den Händen gegeben werden. Dies widerspräche der aktuellen IT-Strategie SanDatas, Infrastruktur und Daten möglichst intern zu halten.

Als Alternative bietet MS über einen sogenannten PowerBI Report Server auch die Möglichkeit, die Cloudfunktionalität über einen eigens betriebenen Server abzubilden. Der Report Server nutzt dabei einen weiteren SQL-Server zur Verwaltung der mit PowerBI erstellten Datenmodelle. Umso erfreulicher ist, dass etwaige Lizenziierungskosten des

Report Server und des SQL-Servers bereits durch den aktuellen Software-Assurance-Vertrag abgegolten sind. Es bietet sich also eine Erweiterung des BI-Systems um eine virtuelle Maschine, die den PowerBI Report Server und den dazugehörigen SQL Server betreibt, an. Aus den erläuterten Rahmenbedingungen ergeben sich die in Abbildung 7 veranschaulichten Komponenten des BI-Systems.



4.2 Agiles Projektmanagement mittels Scrum

Die tatsächliche Implementierung des BI-Systems wird mittels einer agilen Projektmanagementmethode nach Scrum umgesetzt. Vor Projektstart ist eine Initialisierung des Produkt Backlogs mit entsprechenden Product Backlog Items erforderlich. Die folgenden Kapitel beschreiben chronologisch das Vorgehen bei der Umsetzung des Artefakts.

4.2.1 Product Backlog Items und Userstories

Zur bestmöglichen Umsetzung im Sinne des Auftraggebenden ist zur Erstellung und Definition der Produkt Backlog Items von der Perspektive der Vertriebsleitung auszugehen. Die bereitzustellenden Kennzahlen aus Kapitel 4.1 dienen hierbei als maßgebende Orientierung für den Umfang eines einzelnen Sprints. Ziel ist somit - je Sprint - die selbstgesteuerte und dynamische Auswertung jeweils einer der Kennzahlen über ein PowerBI-Dashboard zu ermöglichen. Folglich beinhaltet jeder Sprint sämtliche der fünf in Kapitel 2 aufgeführten Schritte des BI-Lifecycles. Dies umfasst sowohl die Erstellung beziehungsweise Anpassung der ETL-Strecken und somit auch des Core DWHs, als auch die Visualisierung des Datenmodells in PowerBI. Hierzu soll vorab in einem initialen

Workshop mit der Vertriebsleitung eine Sammlung an detaillierten Userstories erarbeitet werden. Eine Userstory ist hierbei nach Diehl (2019) wie folgt aufgebaut: „*Als [Anwendender] möchte ich [Funktionalität], damit / um / weil [Mehrwert]*“. Jede Userstory beschreibt eine konkrete Funktionalität des Dashboards, welche die Vertriebsleitung als notwendig erachtet. Die Summe der Userstories pro Kennzahl ergibt somit ein genaues Bild des zu entwickelnden Dashboards.

4.2.2 Sprint Planning, Sprint und Sprint Review

Im anschließenden Sprint Planning findet je Userstory eine Extraktion konkreter Aufgaben zur Umsetzung im Sprint statt. Im Zuge dessen entsteht das später im Sprint umzusetzende Sprint Backlog. Nach der Durchführung des Sprints werden im Sprint Review die Ergebnisse der Vertriebsleitung und einer fachkundigen Person aus der Finanzabteilung vorgestellt, validiert sowie etwaiges Feedback und Korrekturhinweise aufgenommen, um diese im darauffolgenden Sprint zu berücksichtigen. Jeder Sprint hat einen zeitlichen Umfang von mindestens einer Woche (5 Arbeitstagen entsprechend). Die Funktionalität des IT-Artefakts wächst somit inkrementell von Woche zu Woche.

4.2.3 Limitationen

Da der Umfang der vorliegenden Arbeit begrenzt ist, müssen bei der Ausführung der Ergebnisse Limitationen in mehrerlei Hinsicht getroffen werden. Die Scrum-Rollen werden in der Projektbeschreibung nicht weiter personifiziert, da die vollständige Umsetzung des BI-Systems allein durch den Autor im Rahmen der vorliegenden Arbeit erfolgt. Dementsprechend bilden auch die konkrete Beschreibung der Scrum-Artefakte Retrospektive und Daily Scrum keinen Bestandteil der Arbeit, obwohl diese in der Realität wahrgenommen werden. Abschließend wird die Umsetzung innerhalb der Arbeit auf Kennzahlen im Bereich Umsatz und Deckungsbeitrag beschränkt, weil diese anhand einer gemeinsamen Datenquelle bestimmt werden können. Eine Kennzahl erfordert einen Sprint, sodass sich insgesamt drei durchzuführende Sprints ergeben.

4.2.4 Projektverlauf

Im initialen Workshop mit der Vertriebsleitung stellt sich heraus, dass seitens jener bereits eine konkrete Vision für das Dashboard des erstens Sprints besteht: Zusammengefasst sollen in erster Instanz allein die Informationen aus der händisch erstellten Excel-Matrix (vgl. Anhang, Abbildung 13, S. 59) automatisiert auf dem Dashboard abgebildet werden. Von weiteren Visualisierungen oder Darstellungsformen wird vorerst abgesehen. Dies hängt mit der aktuellen Vorgehensweise bezüglich der Erstellung des Vertriebsberichtes für das regelmäßige Geschäftsleitungsmeeting zusammen. Da in der monatlichen Absprache strategische Entscheidungen anhand aktueller vertrieblicher Zahlen getroffen werden müssen, ist jene Auswertung ebenfalls im selben Rhythmus

vorab zu erstellen. Der Aufwand hinter einer manuellen Konsolidierung der Umsatzdaten eines Unternehmens in der Größenordnung SanDatas ist intuitiv nachvollziehbar. Dementsprechend bedeutet eine automatisierte Erstellung der Übersicht eine enorme Arbeitserleichterung für die Vertriebsleitung, da händische Konsolidierungen und Korrekturen zukünftig wegfallen.

Nr.	Als...	möchte ich...	damit / um / weil...	Prio.	Sprint
1	Vertriebsleitung	den gruppenweiten Umsatz, Deckungsbeitrag und die daraus ermittelte Marge automatisiert dargestellt bekommen (analoge Informationen wie aus der bisher verwendeten Matrix)	damit jene Übersicht nicht jedes mal vor dem monatlichen Geschäftsleitungsmeeting händisch zusammengestellt, konsolidiert und aufbereitet werden muss	10	1
2	Vertriebsleitung	dass die ermittelten Kennzahlen/Beträge je Monat (ACT) und rückwärts zum Jahresanfang (YTD) gerechnet werden	um eine intuitiv verständliche Auswertungsperspektive in Form der aktuellen und kumulierten Beträge zu haben	8	1
3	Vertriebsleitung	dass die ermittelten Kennzahlen/Beträge bereits um die händischen Korrekturen der Buchhaltung korrigiert sind	damit die Kennzahlen/Beträge möglichst exakt sind und somit verlässliche Informationen liefern	7	1
4	Vertriebsleitung	die ermittelten Kennzahlen/Beträge jeweils nach Land, Region, und Kostenstelle gruppiert dargestellt haben (analog zur bisher verwendeten Matrix)	um auf verschiedener Granularität Entscheidungen treffen zu können	7	1
5	Vertriebsleitung	ein fixes Berichtslayout haben	um mich schnell im Bericht zurechtzufinden und um eine einheitliche Anlaufstelle für einen Navigations- beziehungsweise Filterbereich zu haben	5	2
6	Vertriebsleitung	zu jeder Kennzahl die prozentuale Abweichung im Vergleich zur Vorjahresperiode dargestellt bekommen (YOY)	um auf einen Blick zu sehen, ob ein Wachstum stattgefunden hat	4	2
7	Vertriebsleitung	im Kopfbereich informiert werden, wie aktuell der zugrundeliegende Datenstand ist	damit die Kennzahlen/Beträge in Relation zur Aktualität des Datenstand gesetzt werden können	3	2
8	Vertriebsleitung	zu jeder Kennzahl die prozentuale Abweichung zu dem im Jahresplan gesetzten Ziel dargestellt bekommen (YTP)	um auf einen Blick zu sehen, ob die aktuelle Entwicklung nach Plan verläuft	3	3
9	Vertriebsleitung	möglichst aktuelle Daten auf dem Dashboard angezeigt bekommen	damit mindestens tagesaktuelle Entscheidungen getroffen werden können	2	3
10	Vertriebsleitung	im Kopfbereich eine Filtermöglichkeit nach Ländern, Regionen und Kostenstellen haben	um die Kennzahlen/Beträge dediziert auswerten zu können	1	3
11	Vertriebsleitung	im Kopfbereich eine Filtermöglichkeit nach Verkäufer haben	um die Kennzahlen/Beträge einzelner Mitarbeitenden auswerten zu können	1	3
12	Vertriebsleitung	im Kopfbereich eine Filtermöglichkeit nach Kunden haben	um die Kennzahlen/Beträge einzelner Kunden auswerten zu können	1	3

Tabelle 3: Userstories zur Dashboardvision der Vertriebsleitung
(eigene Darstellung)

Um die Anforderungen an das Ergebnis der Automatisierung korrekt umzusetzen, gilt es nun, aussagekräftige Userstories anhand der vorliegenden Excel-Matrix auszuformulieren. In Tabelle 3 auf der vorherigen Seite werden die ermittelten Userstories - sortiert nach absteigender Priorität - aufgelistet. Diese sind hierbei schon nach der jeweiligen zu visualisierenden Kennzahl separiert. So umfasst Sprint 1 die Userstories zu „Umsatz und Deckungsbeitrag, ACT und YTD“, Sprint 2 die Userstories zu „Umsatz und Deckungsbeitrag, YOY“ et cetera. Anhand der Userstories werden anschließend explizite Produktbacklog Items in Form von Subtasks für die jeweiligen Userstories abgeleitet, deren Aufwand im jeweiligen Sprint Planning seitens der Entwicklung abgeschätzt werden. Im Rahmen des ersten Sprints muss zusätzlich zum Dashboard eine Datengrundlage in Form des DWH und entsprechender ETL-Strecken umgesetzt werden. In den anknüpfenden Sprints fällt dieser Arbeitsschritt entsprechend weg, da bereits die Datenbasis geschaffen ist. Aufgrund des Arbeitsumfangs, der für das Erstellen des DWH inklusive der ETL-Strecken zu erwarten ist, wird der erste Sprint mit einem Umfang von zwei Wochen, also 10 Werktagen, veranschlagt. Alle weiteren Sprints haben hingegen einen zeitlichen Umfang von 5 Werktagen. Unterschiedliche Sprintlängen innerhalb eines Projekts widersprechen zwar der Empfehlung des Scrum-Guides, sind jedoch aufgrund des Framework-Charakters desselbigen zulässig. Ebendieser Punkt wird auch nochmals in der Diskussion aufgegriffen (vgl. Kapitel 6).

4.3 Artefaktbeschreibung

Als instanziertes Artefakt ist das BI-System zunächst als ganzheitlicher Komplex zu sehen. Anhand des Schichtenmodells separiert, kann dieser weiter untergliedert werden (vgl. Abbildung 7, S. 29). Als zentrale Bestandteile gelten hierbei drei Komponenten. Im Integration Layer repräsentiert das Core DWH den Single Point of Truth, welcher die auszuwertenden Daten bereitstellt. Im Acquisition Layer kommt den ETL-Strecken die Aufgabe zu, Daten aus unterschiedlichen Quellen zu konsolidieren und aufzubereiten, damit diese im Core DWH persistiert werden können. Das Dashboard im Visualization Layer dient als Schnittstelle zwischen Persistierungsebene und Anwendenden. Diese drei genannten Komponenten sind im Rahmen dieser Arbeit inkrementell instanziert worden und bilden zusammen mit der betreibenden IT-Infrastruktur das BI-System ab. Im Folgenden wird nun die jeweils finale Version der genannten Komponenten beschrieben.

4.3.1 *Core Data Warehouse*

Nach Kimball & Ross (2013) sollte die Granularität der im Core DWH persistierten Daten möglichst auf Transaktionsebene sein. Einzelne Transaktionen, im vorliegenden Falle Verkaufstransaktionen, werden als zentrale Faktentabelle abgebildet, welche auf unterschiedliche Dimensionen referenziert. Um das Schema der Faktentabelle für die angeforderte Auswertung von Umsätzen und Deckungsbeiträgen entwerfen zu können, ist ein grundlegendes Verständnis für die Abwicklung von ergebnisbeeinflussenden Tätigkeiten in SITE notwendig. Im System werden diese gegenüber dem Kunden über sogenannte Belege abgewickelt. Je nach Geschäftsvorfall existieren etwa Verkaufsaufträge, Dienstleistungsaufträge oder Ticketaufträge. Jeder dieser Belege besteht aus einer oder mehrerer Zeilen. Auf Zeilenebene wird zwischen Zeilenarten unterschieden. Am konkreten Beispiel eines Verkaufsauftrag kann eine Zeile aus einem Artikel, dessen Verkaufs- beziehungsweise Bestellmenge sowie dessen Einkaufs- und Verkaufspreis bestehen. Buchhalterische Wirksamkeit entfalten Belege durch den sogenannten Buchungsvorgang. Übertragen auf das hiesige Beispiel würde bei Fakturierung an den Kunden anhand des Verkaufsauftrag eine gebuchte Rechnung erzeugt. Aus den Zeilen der gebuchten Rechnung entstehen sogenannte Verkaufsposten, welche separat in einer gleichnamigen Relation im System erfasst werden. Dieser Prozess fällt nicht nur beim Buchen von Verkaufsaufträgen, sondern bei allen weiteren erlösrelevanten Belegen analog an. Somit stellt die gesamte Relation „Verkaufsposten“ die zur Auswertung grundzulegende Faktentabelle dar. Summiert man hier beispielsweise das Attribut „Zeilenbeträge“ erhält man dadurch, dass alle ergebnisrelevanten gebuchten Zeilen in der Tabelle aufgeführt sind, den zum Zeitpunkt der Summierung erwirtschafteten Umsatz der IT-Gruppe in SITE. Zur Transformation der Relation „Verkaufsposten“ in das für das Core DWH benötigte Star-Schema ist eine tiefere Analyse jener notwendig. Die

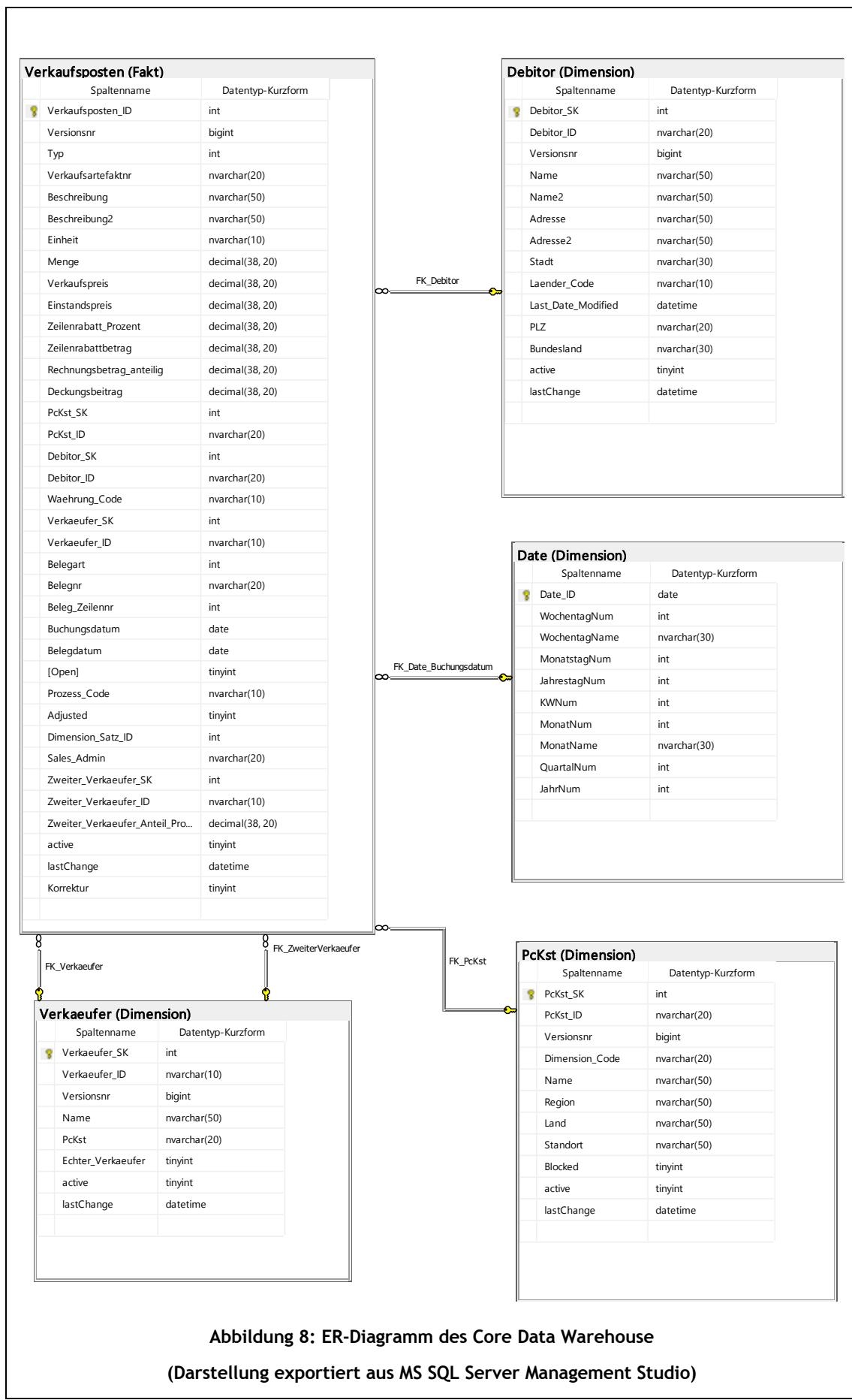
Relation besteht in SITE aus 58 Attributen, die zum einen Fakten wie Menge, Einkaufspreis und Verkaufspreis enthalten, zum anderen aber auch Dimensionen wie Kunde, Kostenstellen, Buchungsdatum oder Verkäufer, welche somit die Transaktion in der jeweiligen Zeile in einen Kontext setzen. Allerdings ist, wie in ERP-Systemen üblich, von einem hohen Normalisierungsgrad bei referenzierten Relationen auszugehen.

Es gilt nun, die für die Auswertung relevanten Dimensionen anhand der verfügbaren Attribute der Relation „Verkaufsposten“ zu wählen und etwaige anfallende Indirektionen auf maximal eine Stufe zu entzerren, damit die Fakten- und Dimensionstabellen im Core DWH dem Star-Schema entsprechen. Nach eingehender Analyse und Transformation der beteiligten Relationen ergibt sich das in Abbildung 8 auf der folgenden Seite veranschaulichte Star-Schema des Core DWs.

Da anhand der Dimensionen der Faktentabelle spätere Aggregationen vorgenommen werden, ist es hilfreich, möglichst aussagekräftige Bezeichnungen der Dimensionen und deren Attributen zu wählen. Die in Kapitel 2.3.2 erwähnte intuitive Benennung der Attribute und Tabellen wird durch die Einführung von Schemata unterstützt. So sind datenbankseitig die Schemata „Dimension“, „Fakt“, „Staging“ und „ETL“ eingerichtet worden, mittels derer sich die Tabellen nach dem Muster „[Schemaname].[Tabellename]“ direkt adressieren lassen. Über das jeweilige Schema der Tabelle wird deutlich, um welche Art von Tabelle es sich handelt.

Die seitens der Vertriebsleitung geforderte Gruppierung der Auswertung nach Land, Region und Kostenstelle (vgl. Userstory 4, Tabelle 3, S. 31) wird über die Dimension „PcKst“ realisiert. Die Bezeichnung „PcKst“ entstammt der systemseitigen Abkürzung für Profitcenter / Kostenstelle und wird ohne Umbenennung übernommen, da diese dem Anwendenden geläufig ist. Durch deren Attribute „Land“, „Region“ und „PcKst_ID“ können die Fakten entsprechend nach Land, Region und Kostenstelle gruppiert werden. Da SITE-systemseitig keine Regionen und Länder gepflegt sind, muss diese Zuordnung vorab im Rahmen des zugehörigen ETL Prozess berücksichtigt werden.

Wie in Kapitel 4.1.1 ausführlich beschrieben, werden Korrekturen von Verkaufstransaktionen zum Zeitpunkt der Umsetzung des Projektes außerhalb von SITE händisch erfasst. Dies hat zur Folge, dass anhand der SITE-Daten ermittelte Umsätze und Deckungsbeiträge um die händisch erfassten Korrekturdaten korrigiert werden müssen. Die händische Korrektur findet seitens der Buchhaltung monatsbasiert je Verkäufer und Kostenstelle statt. Um dieses Problem (vgl. Userstory 3, Tabelle 3, S. 31), zu lösen, werden einzelne Korrekturzeilen mit positiven oder negativen Beträgen je „Verkäufer_ID“, je Kostenstelle und je Monat in der Faktentabelle ergänzt. Hierzu ist im Core DWH die Dimension „Verkäufer“ erfasst, die jedem Fakt einen Verkäufer, also einem Mitarbeitenden im Account Management, zuordnet.

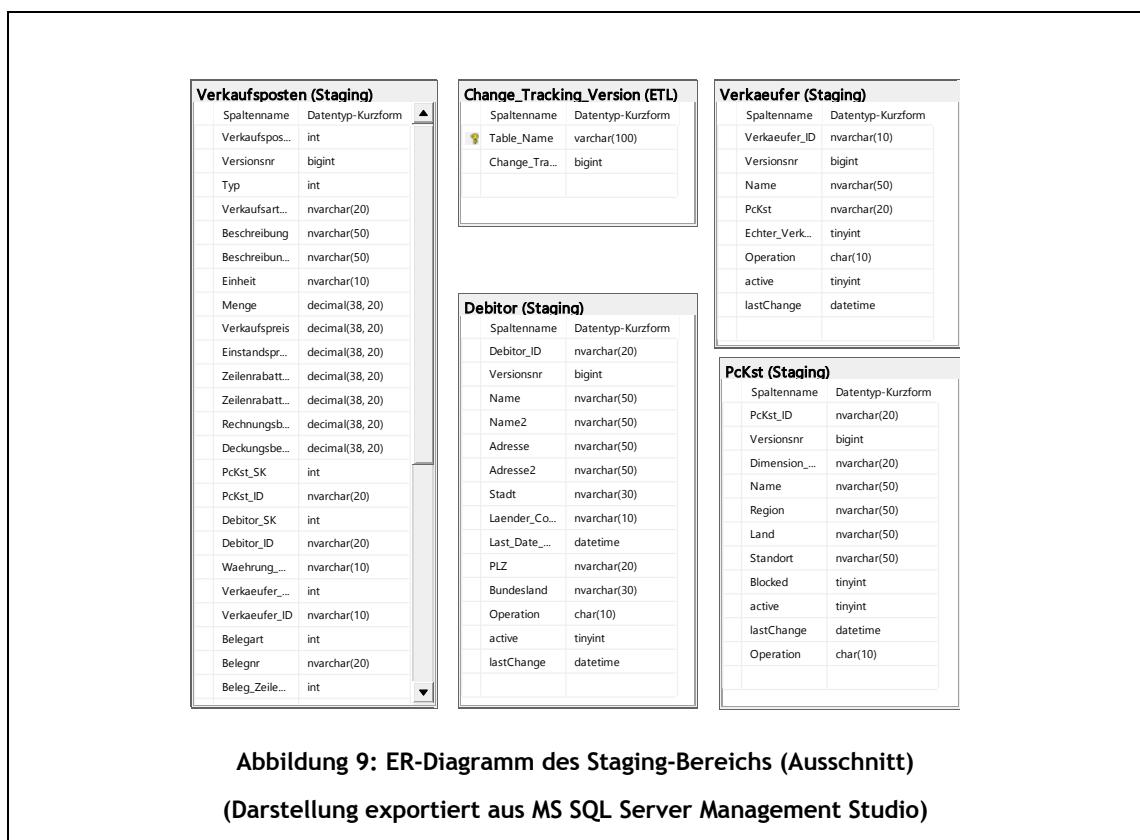


Zur zeitlichen Analyse der Beträge wird eine zentrale Dimension „Date“ angelegt. Jene beinhaltet für jedes Datum, beginnend ab dem ältesten Datum des DWHs, einen eindeutigen Eintrag, um Informationen wie die Quartalsnummer oder Wochenummer erweitert.

Für die Umsetzung eines inkrementellen Loads und einer Historisierung von Änderungen werden weiter in jeder Tabelle des Core DWH die Attribute „active“ und „last-Change“ aufgeführt. Ersteres signalisiert, ob es sich um eine aktive oder deaktivierte (also gelöschte) Zeile handelt, letzteres Attribut führt Buch über den Zeitpunkt der letzten durch den ETL-Prozess bedingten Änderung am Datensatz. Außerdem ist die Funktionalität von SKs, zusätzlich zum eigentlichen PK der Quellrelation, implementiert. Dies erleichtert eine etwaige spätere Erweiterung um detailliertere Historisierungsansätze. Die referentielle Integrität der Faktentabelle in Bezug auf die entsprechenden Dimensionstabellen wird jeweils mittels eines Fremdschlüssel-Constraints auf den SK gewährleistet.

4.3.2 Initialisierung und Aufbau der Extract-Transform-Load-Strecken

Für den ETL-Prozess benötigt jede Tabelle im Core DWH eine gleichartige Repräsentation als Staging-Tabelle (ohne Attribute mit referentieller Integrität). In Abbildung 9 ist das ER-Diagramm des Staging-Bereichs dargestellt. Zusätzlich wird jeweils das Attribut „Operation“ mitgeführt, welches einer Datensatzversion eine CRUD-Änderung (Create, Read, Update, Delete) zuordnet.



Darüber hinaus ist der SK in den Staging-Tabellen nicht vorhanden, da dieser erst beim Einfügen in das Core DWH generiert, respektive mittels eines Lookups ermittelt wird. Im Staging-Bereich wird außerdem noch eine Tabelle, um die jeweils jüngste Change-tracking-Versionsnummer je Tabelle zu speichern, aufgeführt.

Das DWH und entsprechende, in den abgebildeten ER-Diagrammen enthaltene, Relationen werden mittels eines SQL-Skriptes erzeugt. Dies hat den Vorteil, dass auf beliebigen MS SQL-Servern die Strukturen des DWH (vgl. Anhang, Abbildung 14, S. 59) durch Ausführen des Skriptes automatisiert erzeugt werden können. Hierbei ist zu beachten, dass zuerst die Dimensionen und im Anschluss die Faktentabelle erzeugt werden, da diese auf die Dimensionen referenziert. Das vollständige Skript zur Erzeugung der Datenstrukturen ist aus Gründen der Übersichtlichkeit dem Anhang zu entnehmen (vgl. Quellcode 10, S. 62).

Die Initialisierung der Dimension „Date“ erfolgt ebenfalls mittels einer TSQL-Prozedur. Das entsprechende Code-Fragment ist Quellcode 1 zu entnehmen. Zur besseren Übersichtlichkeit sind im Fragment redundante Zeilen durch „...“ ersetzt. Als untere Datumsgrenze der Dimension ist der 01.01.2019 gewählt, da SITE ab diesem Zeitpunkt aktiv im Unternehmen genutzt wird und somit keine älteren Einträge existieren.

```

01  SET NOCOUNT ON
02  SET LANGUAGE German;
03  DECLARE @minDate date = '01.01.2019'
04      , @maxDate date = '31.12.2022'
05      , @loadDate date;
06  SET @loadDate = @minDate;
07
08 WHILE @loadDate <= @maxDate
09 BEGIN
10     if not exists (SELECT Date_ID from Dimension.Date WHERE Date_ID =
@loadDate)
11         INSERT INTO Dimension.Date
12             SELECT
13                 @loadDate as calendarDate
14                 , DATEPART(dw, @loadDate)
15                 , DATENAME(dw, @loadDate)
...
22                 , DATEPART(yy, @loadDate)
23     SET @loadDate = DATEADD(dd, 1, @loadDate);
24 END

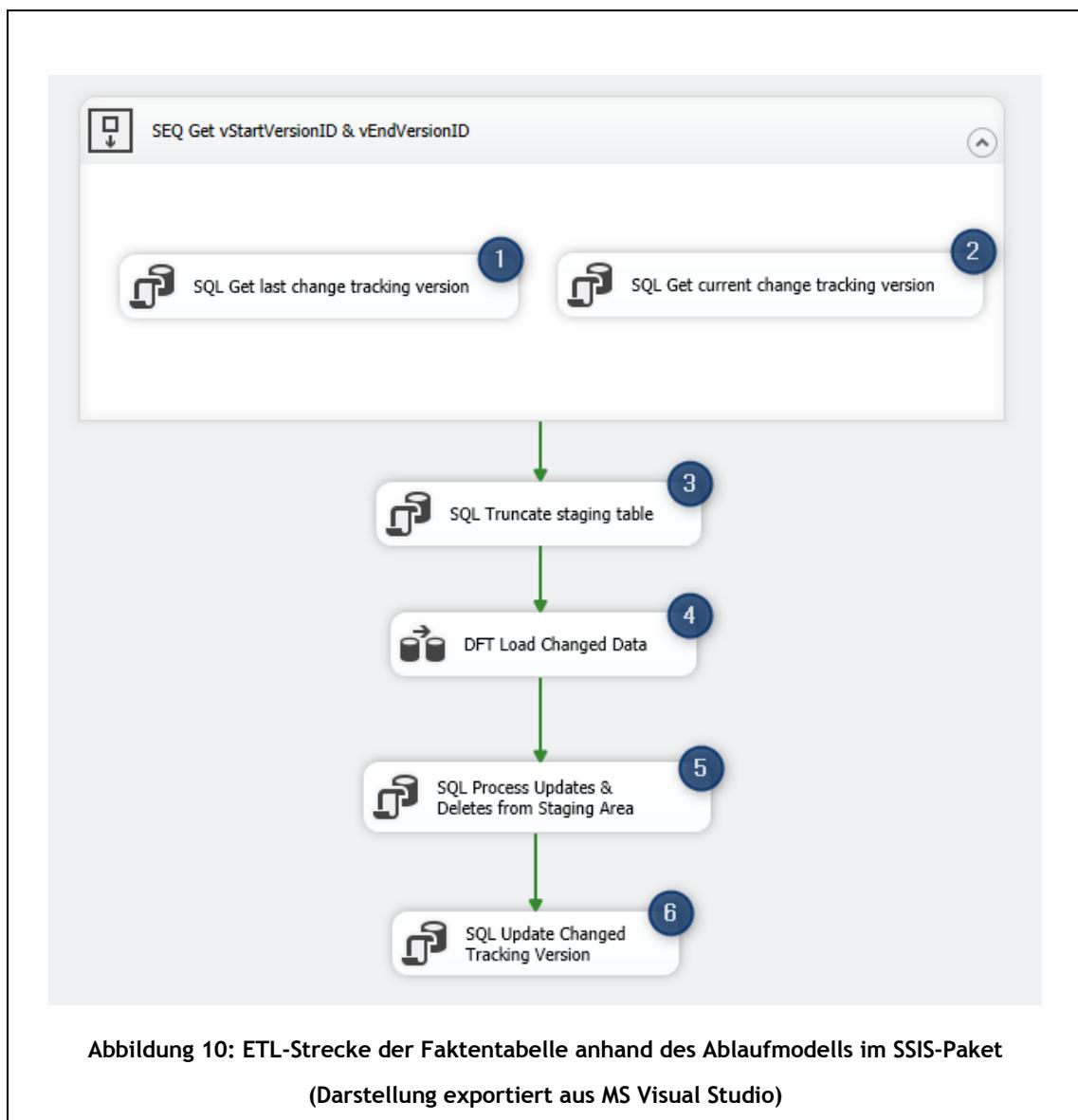
```

Quellcode 1: Initialisierung der Dimension „Date“

<create_DWH.sql> [88-111]

Mittels einer while-Schleife wird über alle Daten zwischen dem definierten Start- und Enddatum iteriert, um jeweils ein Tupel der Form `<Datum, WochentagNummer, WochentagName, MonatstagNummer, JahrestagNummer, KalenderwocheNummer, MonatNummer, MonatName, QuartalNummer, JahrNummer>` in die entsprechende Relation einzufügen. Die Variable `@loadDate` entspricht hierbei dem jeweiligen Datum, auf dessen Anteile mittels der TSQL-Funktionen `DATEPART()` und `DATENAME()` zugegriffen werden kann.

Im Anschluss sind für jede Tabelle zwei SSIS-Pakete erstellt worden. Das jeweils erste Paket ist zur einmaligen Ausführung gedacht und führt einen initialen Full-Load der entsprechenden Daten aus SITE in das Core DWH durch. Im jeweils zweiten Paket ist ein zur periodischen Ausführung entwickelter inkrementeller ETL-Prozess umgesetzt. Zur effizienten Umsetzung einer inkrementellen Variante wird auf eine Funktionalität des MS SQL Servers namens Changetracking zurückgegriffen. Vereinfacht ausgedrückt, notiert der Quell-SQL-Server bei einer CRUD-Operation auf einem Datensatz in einer Systemtabelle eine globale Versionsnummer und die auf dem Datensatz ausgeführte Operation mit. Wird beispielsweise in der Quelle ein Datensatz geändert, bekommt dieser, über den PK des Datensatzes identifiziert, in der Changetracking-Systemtabelle der Quelle einen Eintrag mit der aktuellen um 1 inkrementierten Versionsnummer und der entsprechenden Operation „U“ (für Update). Für einen inkrementellen Übertrag der Daten ins DWH muss sich also die höchste Versionsnummer innerhalb der jeweiligen Tabelle des DWH gemerkt werden, sodass bei erneuter Ausführung des ETL-Prozesses nur die Datensätze der Quelle mit einer noch höheren Versionsnummer geladen werden. Dies bietet den entscheidenden Vorteil, dass auf eine einfache Art und Weise ein inkrementeller Load umgesetzt werden kann. Da beim Changetracking von der Quelle selbst auch nur die CRUD-Operation je Versionsnummer gepflegt wird, ist dieser Ansatz ebenfalls ressourcenschonender als bei einer Speicherung des tatsächlich geänderten Inhalts. Zwar ergibt sich hieraus im Falle mehrerer Änderungen in kurzen Zeitabschnitten auf dem Quelldatensatz die Einschränkung, dass jeweils nur die jüngste Änderung erfasst werden kann, und somit keine vollständige Historisierung möglich ist. Dies ist jedoch seitens des Auftraggebenden nicht gefordert und kann vernachlässigt werden.



Im Folgenden wird nun exemplarisch die periodische ETL-Strecke der Faktentabelle „Verkaufsposten“ näher erläutert. Hierzu ist in Abbildung 10 das Ablaufmodell des entsprechenden SSIS-Paktes veranschaulicht, wobei jedem Task eine eindeutige Nummer zur späteren Beschreibung zugeordnet ist. Im Ablaufmodell selbst wird der Tasktyp „SQL“, welcher ein SQL-Skript auf einer zu definierenden Datenverbindung ausführt (vgl. Tasks 1, 2, 3, 5, 6 in Abbildung 10) und der Tasktyp „Datenfluss“, zur Kapslung komplexerer Datenströme (vgl. Task 4 in Abbildung 10), unterschieden. Ein Pfeil zwischen den jeweiligen Tasks beziehungsweise Objekten definiert die Abfolge der Ausführung. Beginnend mit Task 1 und 2 folgt Task 3 et cetera. Des Weiteren stellt die Ausführungsumgebung die Möglichkeit bereit, Laufzeitvariablen zu benutzen, um beispielsweise Zwischenergebnisse zu speichern. Dies ist anhand des reinen Ablaufmodels nicht erkennbar, da eine Konfiguration über diverse Menüdialoge in MS Visual Studio erfolgt. Fortlaufend wird außerdem auf verwendete SQL-Fragmente je Task eingegangen, um deren Funktionsweise darzustellen.

Task 1: Zuletzt gespeicherte Changetracking-Versionsnummer abrufen (Datenverbindung DWH)

Im ersten Schritt wird die bei der letzten Ausführung des Paketes gespeicherte Changetracking-Versionsnummer abgerufen und in der Laufzeitvariable `vStartVersionID` gespeichert. Die Laufzeitumgebung weist bei Ausführung dem Platzhalter „?“ im SQL-Fragment die Referenz auf die Laufzeitvariable zu. Dies wird im Eigenschaftsmenü des SQL-Tasks definiert und ist daher nicht im Code ersichtlich.

```
1  SELECT ? = (SELECT [Change_Tracking_Version]
2    FROM [ETL].[Change_Tracking_Version]
3   WHERE Table_Name = 'Verkaufsposten')
```

Quellcode 2: SQL-Task 1, Lookup Changetracking-Versionsnummer im DWH

<setup_ETL.sql> [218-220]

Task 2: Aktuelle Changetracking-Versionsnummer aus der Quelle abrufen (Datenverbindung SITE)

Parallel zur Ausführung von Task 1 wird in Task 2 die aktuelle Changetracking-Versionsnummer aus der Quelle, hier der ursprünglichen Verkaufsposten-Tabelle in SITE, abgerufen und in `vEndVersionID` gespeichert. Task 1 und 2 sind hierbei innerhalb einer Sequenz gekapselt, was eine parallele Ausführung der Tasks ermöglicht. Die Parallelität stellt auch kein Problem dar, da hierzu unterschiedliche Datenverbindungen genutzt werden.

```
1  SELECT ? = CHANGE_TRACKING_CURRENT_VERSION()
```

Quellcode 3: SQL-Task 2, Lookup Changetracking-Versionsnummer in Quelle

<setup_ETL.sql> [223]

Task 3: Staging Tabelle leeren (Datenverbindung DWH)

Im Anschluss wird die benötigte Staging-Tabelle geleert, um die zu verarbeitenden Datensätze aus dem vorherigen Durchlauf zu entfernen. An dieser Stelle könnte man bei Bedarf das in Kapitel 2.3.1 erwähnte Corporate Staging Memory aufbauen. Dies würde insofern realisiert werden können, indem nicht jedes Mal die Staging-Tabelle verworfen, sondern eine entsprechende Historisierung in ebenjener aufbaut wird. In der hiesigen Umsetzung ist davon abgesehen, da seitens des Auftraggebenden dahingehend keine Anforderung besteht.

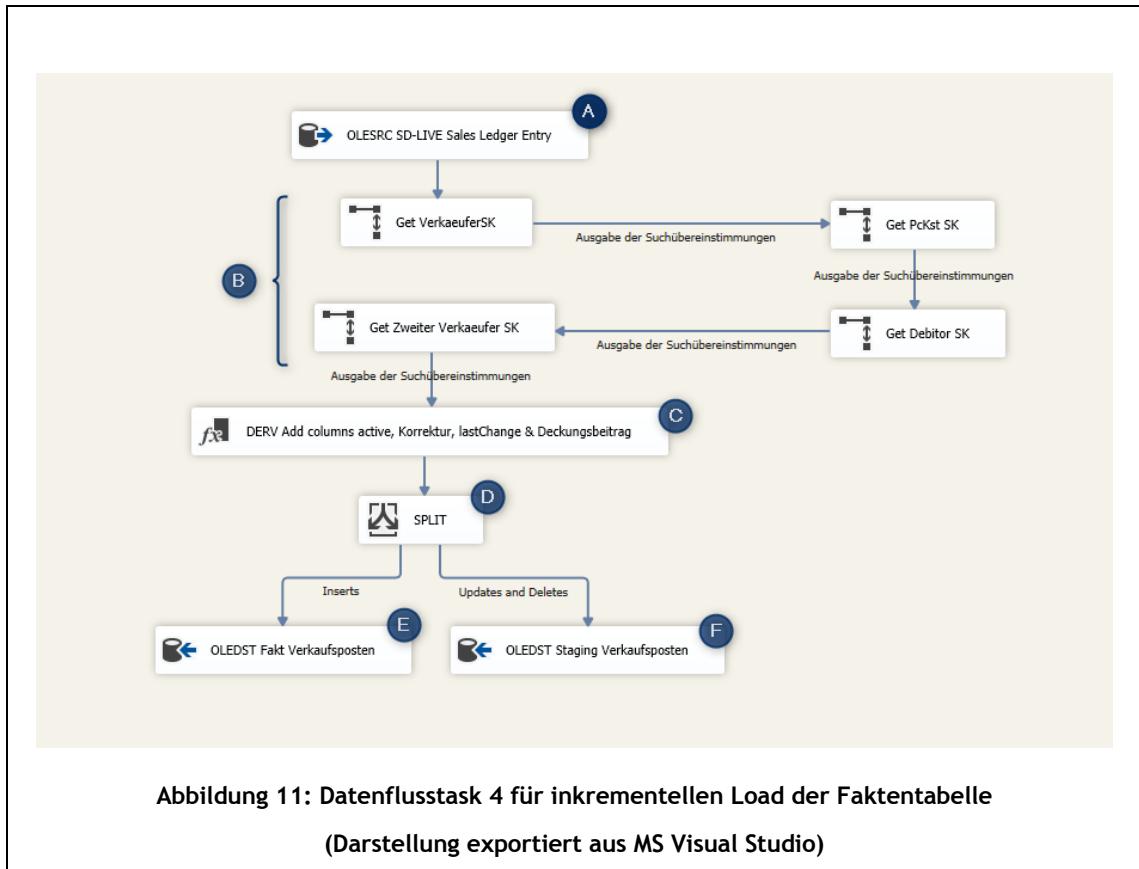
```
1  TRUNCATE TABLE [Staging].[Verkaufsposten]
```

Quellcode 4: SQL-Task 3, Leeren der benötigten Staging-Tabelle

<setup_ETL.sql> [226]

Task 4: Datenfluss der zu persistierenden Daten (Datenverbindung DWH und SITE)

Nun erfolgt mittels des Datenflusstasks 4 der tatsächliche Abruf der geänderten Daten. In Abbildung 11 ist ebendieser Task weiter aufgeschlüsselt. Es erfolgt ein Datenfluss von Quelle A zu den Zielen E und F. Unter A werden nun alle Datensätze der Quelle selektiert, die eine höhere Changetracking-Versionsnummer als bereits persistierte Datensätze haben.



In Quellcode 5 auf der folgenden Seite ist das entsprechende Code-Fragment zu A aufgeführt. Zur Ermittlung der jeweiligen Changetracking-Versionsnummer wird die Quell-tabelle auf die entsprechende Changetracking-Systemtabelle gejoint, um zum einen alle benötigten Attribute der Quelle, zum anderen, um die durch den Join ermittelbare Operation der Version zu projizieren. Zur besseren Übersichtlichkeit innerhalb des abgebildeten Code-Fragments ist die redundante Auflistung benötigter Attribute der Verkaufsposten ebenfalls wieder durch „...“ ersetzt. Eine vollständige Übersicht ist dem Anhang zu entnehmen (vgl. Quellcode 11, S.62).

```

01   SELECT ct.[Entry No_] AS Verkaufsposten_ID
02   ,s.[timestamp] AS Versionsnr
...
29   ,CAST(ct.SYS_CHANGE_OPERATION AS CHAR(1)) [Operation]
30   FROM CHANGETABLE(CHANGES [dbo].[00 SD Gruppe$Sales Ledger Entry], ?) ct
31   LEFT JOIN [dbo].[00 SD Gruppe$Sales Ledger Entry] s
32   ON s.[Entry No_] = ct.[Entry No_]
33   WHERE (
34       SELECT MAX(v)
35       FROM (VALUES(ct.SYS_CHANGE_VERSION),
36               (ct.SYS_CHANGE_CREATION_VERSION)) AS VALUE(v) ) <= ?

```

Quellcode 5: SQL-Task A, Abruf der zuletzt geänderten Datensätze

<setup_ETL.sql> [229-264]

Sollte ein Datensatz im DWH vorhanden sein, ist diesem bereits ein eindeutiger SK zugeordnet. Da im DWH Updates überschrieben werden, benötigt die neuere Version des Datensatzes die entsprechenden SKs der Dimensionen des älteren Datensatzes im DWH. Hierzu wird durch die gruppierten Tasks in B ein Lookup für etwaige SKs der ausgelesenen Zeilen durchgeführt. Sollte es sich um neu einzufügende Datensätze handeln, also kein Lookup-Ergebnis auffindbar sein, werden die Datensätze mit temporären NULL-Werten im SK weitergereicht. Zum aktuellen Zeitpunkt ist eine Historisierung aufgrund der Anforderungen des Auftraggebenden nur im Rahmen von Delete-Operationen umgesetzt. In der Quelle gelöschte Datensätze werden im DWH als inaktiv markiert, folglich findet also nur ein sogenannter Soft-Delete statt. Die Implementierung mittels zusätzlichen SKs für die Dimensionen bietet für das aktuelle Projekt keinen direkten Vorteil, bedeutet jedoch für die Implementierung selbst auch keinen nennenswerten Mehraufwand. Als Vorteil der hiesigen Implementierung, in der bereits SKs gepflegt werden, ist die Möglichkeit einer einfachen Einführung vollständiger Historisierung zu sehen. Die Nichtberücksichtigung von SKs würde bei einer nachträglichen Einführung von Historisierung einen erheblichen Mehraufwand bedeuten, da der aktuelle Datenbestand verworfen und bestehende Tabellenstrukturen weitestgehend geändert werden müssten.

Task C fügt dem Datensatz die Spalten „active“ (Standardwert 0 oder 1, abhängig von der jeweiligen Operation), „Korrektur“ (Standardwert 0), „lastChange“ (Standardwert „[aktuelle Systemzeit]“) und „Deckungsbeitrag“ hinzu. Der Deckungsbeitrag eines Datensatzes ermittelt sich anhand einer berechneten Spalte und ergibt sich aus der folgenden Formel:

$$\text{Deckungsbeitrag} = \text{Rechnungsbetrag}_{\text{anteilig}} - (\text{Einstandspreis} * \text{Menge})$$

Wobei der *Rechnungsbetrag anteilig* bereits durch SITE festgelegt ist und dem an den Kunden verrechneten Zeilenbetrag (inklusive etwaiger Rabatte) entspricht.

Task D teilt nachfolgend den Datenfluss anhand des Wertes in der Spalte „Operation“ auf, welcher durch den vorherigen JOIN mit der Changetracking-Systemtabelle mitgeführt wird. Im Falle eines Inserts werden die Datensätze direkt in das Core DWH eingefügt, im Falle einer Update- respektive Delete-Operation werden die Datensätze in die Staging-Tabelle geschrieben. An dieser Stelle ist Datenflusstask 4 beendet und die Ausführung kann auf höherer Ebene im Ablaufmodell fortgesetzt werden.

Task 5: Datensätze der Staging-Tabelle verarbeiten (Datenverbindung DWH)

Nun werden zum einen alle zu aktualisierenden Datensätze der Staging-Tabelle erneut mit der Faktentabelle gejoint, um die Aktualisierungen entsprechend der Operation „U“ durchzuführen. Zum anderen werden Datensätze, die in der Quelle gelöscht wurden, als inaktiv markiert.

```

01 UPDATE f
02 SET (
03     f.[Versionsnr]=stage.[Versionsnr]
04     , f.[Typ] = stage.[Typ]
...
38 FROM [Fakt].Verkaufsposten f
39 INNER JOIN [Staging].Verkaufsposten stage
40     ON f.Verkaufsposten_ID = stage.Verkaufsposten_ID
41 WHERE stage.Operation = 'U'
42
43 UPDATE f
44 SET f.active = stage.active
45     , f.[lastChange] = stage.[lastChange]
46 FROM [Fakt].Verkaufsposten f
47 INNER JOIN [Staging].Verkaufsposten stage
48     ON f.Verkaufsposten_ID = stage.Verkaufsposten_ID
49 WHERE stage.Operation = 'D'
```

Quellcode 6: SQL-Task 5, Update- und Soft-Delete-Verarbeitung der Staging-Tabelle

<setup_ETL.sql> [267-316]

Task 6: Speichern der aktuellen Changetracking-Versionsnummer (Datenverbindung DWH)

Zuletzt wird noch der Wert der Variable `vEndVersionID` in die Changetracking-Tabelle des Staging-Bereichs zurückgeschrieben, damit bei erneuter Paketausführung, wie vorab dargestellt, nur die geänderten Datensätze geladen werden können. Ein Auszug der entsprechenden Tabelle ist dem Anhang zu entnehmen (vgl. Anhang, Abbildung 16, S. 60). Somit schließt sich der Kreislauf und das SSIS-Paket ist bereit zum periodischen inkrementellen Laden von Datensätzen.

```

01  MERGE [ETL].[Change_Tracking_Version] AS target
02  USING (SELECT 'Verkaufsposten' [Table_Name], ? [Version_ID]
03  ) AS source
04  ON target.Table_Name = source.Table_Name
05  WHEN MATCHED
06  THEN UPDATE
07  SET target.Change_Tracking_Version = source.Version_ID
08  WHEN NOT MATCHED
09  THEN INSERT (Table_Name, Change_Tracking_Version)
10  VALUES (source.Table_Name, source.Version_ID);

```

Quellcode 7: SQL-Task 6, Speichern der aktuell gültigen Changetracking-Versionssnummer

`<setup_ETL.sql> [319-328]`

Nach erfolgter Modellierung und Entwicklung der jeweiligen SSIS-Pakete werden diese über einen Deploy-Vorgang in MS Visual Studio auf dem entsprechenden SQL-Server des DWH bereitgestellt (vgl. Anhang, Abbildung 15, S.60) und dort anhand eines festzulegenden Zeitplans periodisch ausgeführt. Alle weiteren periodischen ETL-Strecken gestalten sich analog zur erläuterten, entsprechend der unterschiedlichen Attribute und ohne Lookup-Routinen für die SKs angepasst. Jene Code-Fragmente sind ebenfalls aus Gründen der Übersichtlichkeit dem Anhang zu entnehmen (vgl. Quellcode 11, S.67).

4.3.3 Dashboard

Zur Auswertung und Visualisierung der Daten mittels MS PowerBI werden die Relationen des DWH als zugrundeliegendes Datenmodell importiert und als periodische Datenquelle verknüpft. Als zentrales Analyseelement im Dashboard ist die Darstellungsform einer Matrix gewählt. Diese bildet auf Zeilenebene die Hierarchie Jahr → Monat → Region → Kostenstelle ab, wobei das Symbol „→“ einer Drilldown-Operation entspricht. In Abbildung 12 auf folgender Seite ist das Dashboard in unterschiedlichen Filterstufen abgebildet. Die konkreten Beträge sind aus datenschutzrechtlichen Gründen unkenntlich gemacht.



Auf Spaltenebene werden die Beträge des aktuellen Umsatzes und Deckungsbeitrages („Umsatz ACT“ und „DB ACT“), die daraus resultierende Marge („Marge ACT“), die ab Jahresanfang kumulierten Beträge des Umsatzes und Deckungsbeitrages („Umsatz YTD“ und „DB YTD“), die prozentuale Veränderung zur Vorjahresperiode („Umsatz ACT YoY“ und „DB ACT YoY“) und das Verhältnis zu den Planzahlen („Umsatz VS Plan %“ und „DB VS Plan %“) dargestellt. Visuell wird eine eingehende Analyse durch die integrierten Datenbalken unterstützt, welche den jeweils angezeigten Betrag optisch ins Verhältnis zu den anderen Beträgen setzt, wie von einem klassischen Säulendiagramm bekannt. Hierbei sind positive Zahlen blau hervorgehoben, negative hingegen rot, was ebenfalls die Veranschaulichung fördert. Der markante Kopfbereich im oberen Part des Dashboards ermöglicht dem Anwendenden, die Matrix nach Land, Region und Kostenstelle zu filtern. Des Weiteren wird, wie in einer der Userstories gefordert, über die Aktualität der Daten informiert.

Sämtliche Beträge und Anteile im Dashboard sind mittels der von Microsoft entwickelten Formel- und Abfragesprache DAX (Data Analysis Expressions) umgesetzt. DAX bietet die Möglichkeit, programmiersprachenähnlich mittels einer umfangreichen Bibliothek relationale Daten dynamisch zu aggregieren. Hierzu werden sogenannte Measures auf den benötigten Tabellen erstellt. In Quellcode 8 sind die Formelfragmente zur Berechnung der Umsätze und Deckungsbeiträge dargestellt.

```

1 #Umsatz ACT = SUM(Fakt_Verkaufsposten[Rechnungsbetrag_anteilig])
2 #DB ACT = SUM(Fakt_Verkaufsposten[Deckungsbeitrag])
3 #Marge ACT% = DIVIDE([#DB ACT],[#Umsatz ACT],0)
4 #Umsatz YTD = TOTALYTD([#Umsatz ACT],DIM_Date[Date_ID])
5 #DB YTD = TOTALYTD([#DB ACT],DIM_Date[Date_ID])
6 #Umsatz VS Plan % = DIVIDE([#Umsatz ACT],[#Umsatz Plan],0)
7 #DB VS Plan % = DIVIDE([#DB ACT],[#DB Plan],0)

```

Quellcode 8: DAX-Formeln zur Berechnung der Measures für ACT-, YTD- und %-Beträge

Hervorzuheben ist, dass Measures in DAX im aktuellen Filterkontext ausgewertet werden. So genügt beispielsweise in Quellcode 8 die simple Summierung aller Rechnungsbeträge zur Berechnung des Umsatzes, da beim Einfügen des Measures in die Matrix der Kontext durch die Zeilenebene (Jahr, Monat, Region, Kostenstelle) gegeben ist.

Analog hierzu verhält sich auch die Berechnung der YOY-Abweichung (vgl. Quellcode 9, S. 47). Kontextabhängig wird zur prozentualen Bestimmung des Anteils die korrekte Vorjahresperiode als Vergleichsbasis herangezogen. Damit YOY-Vergleiche nur ausgeführt werden, wenn auch tatsächlich Umsätze in der betrachteten Periode erzielt wurden, wird vor Berechnung noch das Datum der letzten Transaktion bestimmt und in einer Variable gespeichert. So werden im laufenden Jahr YOY-Berechnung nur bis zur

aktuellen Transaktion durchgeführt. Für Zeilen über das Datum der letzten Transaktion hinweg wird statt einer Abweichung in Höhe von -100% ein leerer Spalteneintrag angezeigt, entsprechend entsteht auch kein negativer roter Balken im Rahmen der Visualisierung als Datenbalken. Dies lässt die Ansicht für das laufende Jahr übersichtlicher wirken.

```

01  #infoHighestDate(Umsatz ACT) = (
02      CALCULATE(
03          MAX(DIM_Date[Date_ID]),
04          FILTER(Fakt_Verkaufsposten, Fakt_Verkaufsposten[#Umsatz ACT]<>0))
05  )
06
07 #DB ACT YoY% = (
08     var highestYear = [infoHighestDate(Umsatz ACT)]
09     var prevYear =
10        CALCULATE(
11            [#DB ACT],
12            DATEADD('DIM_Date'[Date_ID], -1, YEAR),
13            DIM_Date[Date_ID]<highestYear
14        )
15     RETURN DIVIDE([#DB ACT] - prevYear, prevYear,0)
16  )
17
18 #Umsatz ACT YoY% = (
19     var highestYear = [infoHighestDate(Umsatz ACT)]
20     var prevYear =
21        CALCULATE(
22            [#Umsatz ACT],
23            DATEADD('DIM_Date'[Date_ID], -1, YEAR),
24            DIM_Date[Date_ID]<highestYear
25        )
26     RETURN DIVIDE([#Umsatz ACT] - prevYear, prevYear,0)
27  )

```

Quellcode 9: DAX-Formeln zur Berechnung der Measures für YOY-Vergleiche

5. Evaluation

Das umgesetzte Artefakt wird nun im folgenden Abschnitt anhand angefallener Kosten, Feedback des Anwendenden, aus technischer und aus Projektmanagement-Perspektive evaluiert.

In Kapitel 4 wurde die Herausforderung identifiziert, dass das BI-System möglichst kosteneffizient umgesetzt und betrieben werden können sollte. Entsprechend wird im ersten Schritt der Evaluation eine kalkulatorische Rechnung im Kontext SanDatas direkt identifizierbarer Kosten aufgestellt. Im Wesentlichen lassen sich folgende zwei Kostentreiber im Projekt feststellen:

- (1) Hardware- und Software-/Lizenzkosten, Personalaufwand zum Setup und Betrieb der Infrastruktur
- (2) Personalaufwand zur Umsetzung des Artefakts

Das BI-System wird auf zwei virtuelle Maschinen (VM) aufgeteilt, um zum einen den Reportserver und zum anderen das DWH zu betreiben (vgl. Abbildung 7, S. 29). Ausschlaggebend für die Kosten einer VM sind die ihr zugeteilten Ressourcen in Form von virtuellem Arbeitsspeicher, Hintergrundspeicher und CPU-Kernen. Da sämtliche IT-Infrastruktur von SanData betrieben wird, gilt es nun, einmalige Investitionskosten für Hardware und Lizenzen auf die Konfiguration der jeweiligen VM umzulegen. Um einen Anhaltspunkt für Einzelkosten hinsichtlich (1) zu erhalten, wurde im Rahmen der Arbeit ein Interview mit der IT-Leitung durchgeführt. Die VMs werden auf einem Datacenter, bestehend aus einer Servereinheit und einem All-Flash-Storagesystem, betrieben. Anhand von für das Virtualisierungspotential der Hardware maßgebenden Virtualisierungsfaktoren können die Hardwarekosten des Datacenters auf einen einzelnen virtuellen Gigabyte Arbeitsspeicher, einen Gigabyte Hintergrundspeicher und einen Prozessorkern heruntergerechnet werden. Generell wird Hardware bei SanData auf einen Zeitraum von 5 Jahren abgeschrieben, was ebenfalls in der Kalkulation zu berücksichtigen ist. Mittels der aus dem Interview extrahierten Informationen entsteht die in Tabelle 4 auf der Folgeseite dargestellte Einzelkostenübersicht. Anhand dieser können im Anschluss die Kosten der beiden VMs inklusive deren Betriebskosten auf ein Jahr gerechnet und der Personalaufwand hinsichtlich der Umsetzung des Artefakts bestimmt werden (vgl. Tabelle 5, S. 50). Die für die Gesamtkosten relevanten anteiligen Kosten sind in Tabelle 4 jeweils grau hervorgehoben. Demzufolge belaufen sich die kalkulatorischen Gesamtkosten des Projektes auf 8.252,77 Euro, wovon der Personalaufwand zur Umsetzung des Artefakts mehr als 90% ausmacht. Entsprechend gering ist der Anteil, welcher der Anschaffung und dem Betrieb der IT-Infrastruktur zukommt (10%).

Art	Bereich	Beschreibung	Preis / Anzahl	Formel / Herkunft
Server	Allgemein	Beschaffungs-preis	12.251,33 €	Interview
		Abschreibungs-zeitraum	5 Jahre	Interview
		Abschreibung pro Jahr	2.450,27 €	$\frac{\text{Beschaffungspreis}}{\text{Abschreibungszeitraum}}$
		Kostenanteil CPU	0,33	Interview
		Kostenanteil RAM	0,66	Interview
	CPU	CPUs physisch	20	Interview
		Virtualisierungs-faktor	8	Interview
		vCPU (virtuell)	160	$\text{Virt. faktor} * \text{CPUs physisch}$
		Kosten je vCPU pro Jahr	5,10 €	$\frac{\text{Abschr. pro Jahr} * \text{Kostenanteil CPU}}{\text{vCPU}}$
	RAM	RAM physisch	765 GB	Interview
		Virtualisierungs-faktor	0,8	Interview
		vRAM (virtuell)	614,4 GB	$\text{Virt. faktor} * \text{RAM physisch}$
		Kosten je GB vRAM pro Jahr	2,66 €	$\frac{\text{Abschr. pro Jahr} * \text{Kostenanteil RAM}}{\text{vRAM}}$
Storage-system	Allgemein	Beschaffungs-preis	59.110,85 €	Interview
		Abschreibungs-zeitraum	5 Jahre	Interview
		Abschreibung pro Jahr	11.822,17€	$\frac{\text{Beschaffungspreis}}{\text{Abschreibungszeitraum}}$
	Speicher	Speicher physisch	42240 GB	Interview
		Virtualisierungs-faktor	0,85	Interview
		Speicher virtuell	35.904 GB	$\text{Virt. faktor} * \text{Speicher physisch}$
		Kosten je GB virt. Speicher pro Jahr	0,33 €	$\frac{\text{Abschr. pro Jahr}}{\text{Speicher virtuell}}$
	ESXi Li-zenz	Lizenzkosten je vCPU pro Jahr	7 €	Interview
Lizenzen & Soft-ware	Microsoft	Windows Server 2019	kostenfrei, da Microsoftpartner	
		SQL Server	kostenfrei, da Microsoftpartner	
		PowerBI Report Server	kostenfrei, da Microsoftpartner	
		PowerBI Desktop	grundätzlich kostenfrei	
Personal	Systemad-min	Setupaufwand je VM	2 h	Interview
		Wartungsauf-wand pro Monat	0,5 h	Interview
		Wartungsauf-wand pro Jahr	6 h	$\text{Wartungsaufwand pro Monat} * 12$
		Systemadmin-Aufwand gesamt	8 h	$\text{Setupaufwand je VM} + \text{Wartungsaufwand pro Jahr}$
		Stundenkosten	33 €	Interview
		Systemadmin-Kosten gesamt	264 €	$\text{Systemadminaufwand gesamt} * \text{Stundenkosten}$
	Umset-zung Arte-fakt	Sprints	3	
		Arbeitstage ge-samt	20 Tage	$10 \text{ Tage} + 5 \text{ Tage} + 5 \text{ Tage}$
		Stunden je Tag	8 h	
		Umsetzung Auf-wand gesamt	160 h	$\text{Arbeitstage gesamt} * \text{Stunden je Tag}$
		Stundenkosten	46,50 €	Interview
		Umsetzung Kos-tten gesamt	7.440,00 €	$\text{Umsetzung Aufwand gesamt} * \text{Stundenkosten}$

Tabelle 4: Einzelkostenübersicht von Server, Storage, Lizenzen & Software und Personal

(eigene Darstellung)

Kostentreiber	Beschreibung	Konfiguration / Menge	Kosten
(1)	VM1 Data Warehouse	4 vCPUs, 12GB VRAM, 180GB Speicher virt.	403,09 €
	VM2 PowerBI-Reportserver	4 vCPUs, 12GB VRAM, 200GB Speicher virt.	409,68 €
(2)	Umsetzung des Artefakts	160 h	7.440,00€
SUMME			8.252,77 €

Tabelle 5: Gesamtkosten des umgesetzten BI-Systems

(eigene Darstellung)

Aus Nutzerperspektive evaluiert, bringt das BI-System überwiegend Vorteile mit sich. Durch die Automatisierung der vormals händischen Konsolidierung und Aufbereitung unternehmerischer Vertriebsdaten wird zunächst wertvolle Zeit eingespart, die seitens der Vertriebsleitung stattdessen etwa in strategische Arbeit investiert werden kann. Darüber hinaus werden durch die Automatisierung auch Fehler vermieden, Korrekturen sind nachvollziehbar im DWH hinterlegt und das Dashboard zeigt stets aktuelle Zahlen an. Gerade letzteres bietet einen maßgeblichen Vorteil, da bisweilen vor Umsetzung des Projektes jeweils nur einmal im Monat ein entsprechender Bericht ausgewertet werden konnte. Tagesaktuelle Auswertungen gewähren demgegenüber die Möglichkeit, frühzeitig Entwicklungen zu erkennen und entsprechend zu reagieren. Eine weitere Optimierung ist in der dynamisch und intuitiv verwendbaren Auswertung über den gesamten Datenbestand seit SITE-Start zu sehen. Ausgehend von einem statischen Bericht je Geschäftsleitungs-Termin mit einer Auswertung anhand eines fixen Zeitpunkts kann nunmehr anhand des PowerBI-Dashboards flexibel durch die Daten navigiert werden, etwa in Form von Rollup-und Drillthrough-Operationen bis auf Kostenstelleebene und vieles mehr. Außerdem werden Anwendende nun auch durch visuelle Elemente in der Matrix unterstützt, was sich als besonders benutzerfreundlich erweist. Dies ermöglicht einen völlig neuen Ansatz der Analyse und anknüpfender Diskussion von Ergebnissen.

Aus technischer Perspektive ist zu erwähnen, dass sich die Beanspruchung des Quellsystems durch die inkrementellen Loads in Grenzen hält. Die Faktentabelle umfasst zum Zeitpunkt der Umsetzung 851.403 Datensätze. Müssten diese bei jeder Datenaktualisierung erneut vollständig geladen werden, könnte sich dies negativ auf die Performance im Quellsystem auswirken. Zwar bestünde die Möglichkeit, Load-Vorgänge zum Zeitpunkt einer geringen Nutzung im Quellsystem durchzuführen, allerdings würde sich dies wiederum auf die Datenaktualität auswirken. Mittels des inkrementellen Ansatzes ist es problemfrei möglich, beispielsweise im Stundentakt die Einträge im DWH zu aktualisieren, um möglichst aktuelle Daten im Dashboard vorzuhalten. Des Weiteren

ist das DWH so konstruiert, dass es in der Zukunft leicht um eine vollständige Historisierung erweitert werden kann. Dadurch ist seitens der technischen Architektur Flexibilität für weitere Anwendungsfälle gewährleistet.

Aus projektseitiger Perspektive erweist sich das agile Vorgehen nach Scrum als besonders nützlich. Da das Projekt innerhalb von drei Sprints umgesetzt wurde, ergibt sich auch dreimal die Möglichkeit eines direkten Feedbacks seitens des Anwendenden und der Fachabteilung, um dieses im jeweils anknüpfenden Inkrement zu berücksichtigen. Somit ist von Beginn an eine zielgerichtete Entwicklung gewährleistet. Dem Anwendenden bietet sich dadurch auch die Möglichkeit, sich zeitnah mit den neuen Begebenheiten vertraut zu machen und aktiv mitgestalten zu können. Dies erhöht die Akzeptanz des Artefakts im Nachgang enorm und trägt maßgeblich zum Projekterfolg bei. Außerdem ist ein Projekt nach Scrum-Gedanken nicht abgeschlossen, solange das zugehörige Produkt betrieben wird. Entsprechend besteht auch das Product Backlog während des gesamten Lebenszyklus des Produktes. Übertragen auf das Dashboard-Szenario ist ebenso das Ziel, jenes langfristig, mit größtmöglichem Nutzen für den Anwendenden zu betreiben und flexibel den Funktionsumfang zu ergänzen, respektive zu erweitern. Mittels des Scrum-Prozesses kann somit problemlos nach Veröffentlichung des Dashboards weitere Funktionalität ergänzt oder inkrementell an dessen Qualität gearbeitet werden.

6. Diskussion

Zur Interpretation der Evaluationsergebnisse wird im Folgenden diskutiert, welche Bedeutung den einzelnen Ergebnissen zukommt und wie sich diese zur eingangs gestellten Forschungsfrage „Wie kann ein BI-System in einem KMU gestaltet und agil umgesetzt sein?“ verhalten.

Die Kosten, insbesondere der zeitliche Aufwand zur Umsetzung des Projekts, mögen auf den ersten Blick verhältnismäßig hoch erscheinen. Dies ist auf zweierlei Umstände zurückzuführen: Zum einen bestehen noch keine Erfahrungswerte innerhalb des Unternehmens bei der Umsetzung des DWHs, der ETL-Strecke und des Dashboards, auf die zurückgegriffen werden könnte. Daher ist ein bedeutender Anteil des Aufwandes konzeptioneller Art, auf welchen in nachfolgenden Projekten aufgebaut werden kann. Zum anderen ist das Potential der aufbereiteten Datenbasis durch die aktuell dargestellten Kennzahlen längst nicht ausgeschöpft. Durch die Etablierung einer umfangreichen Faktentabelle anhand der Verkaufsposten sind in Zukunft verschiedenste Dashboards möglich, die beispielsweise Auskünfte über Kunden oder Hersteller visualisieren könnten und dadurch bereits weitere Anwendungsfälle abdecken. Bei einer Steigerung der Anwendungsfälle würde der höhere initiale Aufwand entsprechend relativiert. Zudem

überwiegt mittelfristig die Zeitersparnis durch Automatisierung gegenüber den einmaligen Projektkosten, da die bislang regelmäßig stattfindende händische Konsolidierung und Aufbereitung entfällt. Bei einem fiktiven Stundensatz der Vertriebsleitung in Höhe von 100 Euro und 3 Tagen Aufwand bezüglich der Berichtserstellung je Monat amortisieren sich die Projektkosten innerhalb von 3,44 Monaten. Einhergehend ist zu erwähnen, dass in dieser Betrachtung noch nicht der Mehrwert berücksichtigt ist, welcher der Vertriebsleitung durch die gewonnene Zeit zukommt. Somit ist der initial hohe Zeitaufwand zur Projektumsetzung als notwendiges Fundament einer längerfristigen Investition zu sehen.

Für einen Kulturwandel, weg von statischen Berichten, hin zu Vorteilen eines dynamischen Reporting in Form von Dashboards werden ganzheitliche Ansätze benötigt. Die Umsetzung des vorliegenden Projektes hat anhand eines konkreten Anwendungsfalles aufgezeigt, wie ein BI-System in einem KMU gestaltet sein kann. Übertragen auf einen unternehmensweiten Kontext, ersetzt das Dashboard allerdings nur einen von vielen existierenden statischen Berichten. Um langfristig unternehmensweit von einer Nutzung von BI-Systemen zu profitieren, gilt es nun, Schritt für Schritt vom oberen Management ausgehend, Maßnahmen im Hinblick einer kontinuierlichen Weiterverfolgung des aufgezeigten Ansatzes zu treffen. Um dies zukünftig kostenschonend weiter verfolgen zu können, empfiehlt sich nach Möglichkeit der Aufbau eines internen Kompetenzteams im Bereich Data Science / Reporting. Zwar könnte eine entsprechende Kompetenz auch extern eingekauft werden, allerdings wäre diese wiederum nur zeitlich begrenzt und widersprüche somit einem nachhaltigen Ansatz. Diese Erkenntnis deckt sich auch mit den in Kapitel 2 analysierten Quellen (Becerra-Godinez et al., 2020; Eret & Kemper, 2016; Llave, 2019; Olszak & Ziembka, 2012; Popović et al., 2019).

Als Limitation für die Anwendbarkeit des beschriebenen Leitfadens zur agilen Umsetzung eines BI-Systems im KMU, sind die Rahmenbedingungen des Projektes zu sehen. Dass SanData als IT-Systemhaus, aufgrund der Natur der Geschäftstätigkeit, bereits eine gewisse Nähe und Affinität zu Technologie und IT besitzt, ist intuitiv nachvollziehbar. Außerdem ergeben sich durch den Microsoft-Partnerstatus auch gewisse Kostenvorteile im Bereich Softwarelizenierung, was wiederum Einschränkungen in der Technologiewahl bedeutet. So wurde das BI-System, den Umständen geschuldet, zum Großteil mittels Software des Herstellers Microsoft umgesetzt.

Herausfordernd im Projektmanagement gestaltet sich die gleichmäßige Verteilung der Arbeitslast auf die einzelnen Sprints. Im Verhältnis zum DWH ist das Dashboard aufgrund der aufbereiteten Datenbasis zügiger umsetzbar. Im Umkehrschluss bedeutet dies, dass mehr Zeit für das DWH und entsprechende ETL-Strecken eingeplant werden sollte als für die Umsetzung des Dashboards. Für den konkreten Fall zeigt sich dies

anhand der Sprintdauer. Der erste Sprint, bei dem auch der initiale Aufbau des DWH und der ETL-Strecken stattgefunden hat, nahm zwei Wochen in Anspruch, wohingegen die restlichen beiden Sprints jeweils mit einer Woche veranschlagt wurden. Um den Aufwand innerhalb der Sprints nach ursprünglichem Scrum-Gedanken gleichmäßig zu gewichten, bestünde die Möglichkeit, einen dedizierten Sprint zur Umsetzung des DWH zu planen. Allerdings widerspräche dieser Ansatz - trotz resultierender gleichmäßiger zeitlicher Gewichtung der Sprints - ebenfalls wieder dem Scrum-Gedanken, da nach jedem Sprint eine potentiell veröffentlichtbare Inkrementversion bereitzustellen sei. In jenem Falle würde das für den Anwendenden benutzbare Dashboard nach dem ersten Sprint nicht existieren. Da der Scrum-Guide nur ein Rahmenwerk darstellt, wäre eine Abänderung im erläuterten Sinne durchaus möglich, wobei es abzuwägen gilt, ob jenes Vorgehen bestmöglich die Ziele des Auftraggebers erfüllen kann. Außerdem ist im Kontext von Scrum erwähnenswert, dass innerhalb einer Organisation, die nicht agil aufgestellt ist, anfangs Hürden bezüglich der Akzeptanz einer neuen Projektmanagementmethode nicht unterschätzt werden dürfen. Entsprechend empfehlenswert ist, zeitliche Puffer bei der Sprintplanung einzukalkulieren.

Insgesamt schafft die Arbeit sowohl für die Forschung als auch für die Praxis einen Mehrwert. Für Erstere, durch drei wesentliche Kernaspekte: Es wird ein Überblick zu aktueller Literatur im Bereich BI, auch im Kontext KMU, geschaffen, es wird aufgezeigt, wie Design Science Research mit einer agilen Projektmanagementmethode kombiniert werden kann und es wird anhand eines Leitfadens dargelegt, wie ein BI-System in einem KMU effizient gestaltet werden könnte. Für Letztere entsteht der Mehrwert durch die tatsächliche Umsetzung des BI-Systems im mittelständischen Umfeld. Somit wird die eingangs identifizierte Forschungslücke geschlossen.

7. Fazit

Business Intelligence ist einerseits als Sammlung von Technologien und Werkzeugen, andererseits als Prozess zur Unterstützung von Geschäftsentscheidungen, bei welchem betriebliche Daten in Wissen umgewandelt werden, zu verstehen. Ein BI-System kombiniert hierbei mehrere IT-Komponenten, die eine automatisierte Bereitstellung des Wissens ermöglichen. Im begrifflichen Kontext ist Business Intelligence als Teilbereich von Business Analytics einzuordnen, welcher wiederum innerhalb von Data Analytics und Big Data Analytics zu verorten ist. Data Science bildet hierbei einen begrifflichen Rahmen. Auf technologischer Seite haben sich BI-Systeme in Form mehrschichtiger Architekturen, bestehend aus einem Acquisition Layer (Extract-Transform-Load-Komponenten), einem Integration Layer (Core Data Warehouse), einem Reporting Layer (Data Marts) und einem Visualization Layer, durchgesetzt.

Weiter wird aufgezeigt, dass ein BI-System mittels eines agilen Ansatzes durchaus ressourcenschonend in einem KMU umsetzt werden kann. Eine Gliederung der Sprints nach umzusetzenden Kennzahlen erweist sich als zielführend und regelmäßige Feedbackrunden mit dem Auftraggebenden während des Entwicklungsprozesses tragen maßgeblich zum Projekterfolg und der Akzeptanz des Artefakts bei. Der Aufwand für die Erstellung des zugrundeliegenden Data Warehouse darf jedoch nicht unterschätzt werden, da dieser in zeitlicher Hinsicht einen Großteil des ersten Sprints einnimmt. Es wird gefolgert, dass ein initiales BI-Projekt im Unternehmen nicht als alleinstehende, gekapselte Einheit beurteilt werden sollte, sondern stattdessen eine Betrachtung als Fundament für anknüpfende Projekte zielführender ist. Der deutlich kostendominierende Anteil eines intern umgesetzten BI-Projektes beläuft sich mit 90% im hiesigen Fall auf personelle Aufwände, die in die Konzeption und Umsetzung des Artefaktes fließen. Demgegenüber stehen verhältnismäßig geringe Kosten für die reine Technologie (10% Kostenanteil am Gesamtprojekt). Der hohe initiale zeitliche Aufwand relativiert sich allerdings nach kurzer Zeit, wenn anhand der geschaffenen Datenbasis weitere Auswertungsperspektiven entstehen und Automatisierungseffekte genutzt werden.

Für anknüpfende Forschungsarbeiten bliebe unter anderem zu prüfen, ob ein etwaiger Open-Source-Ansatz zu anderen Ergebnissen führt. Eine Abschätzung der einhergehenden potenziellen Möglichkeiten oder Limitationen ist wünschenswert. Weiterführende Ausarbeitungen wären außerdem im Kontext der kleinen Unternehmen oder in Form eines Branchenvergleichs zu identifizieren.

Literaturverzeichnis

- Ain, N., Vaia, G., DeLone, W. H., & Waheed, M. (2019). Two decades of research on business intelligence system adoption, utilization and success – A systematic literature review. *Decision Support Systems*, 125, 113113. <https://doi.org/10.1016/j.dss.2019.113113>
- Angles, R., & Gutierrez, C. (2008). Survey of graph database models. *ACM Computing Surveys*, 40(1), 1–39. <https://doi.org/10.1145/1322432.1322433>
- Becerra-Godinez, J. A., Serralde-Coloapa, J. L., Ulloa-Marquez, M. S., Gordillo-Mejia, A., & Acosta-Gonzaga, E. (2020). Identifying the main factors involved in business intelligence implementation in SMEs. *Bulletin of Electrical Engineering and Informatics*, 9(1), 304–310. <https://doi.org/10.11591/eei.v9i1.1459>
- Bundesverband mittelständische Wirtschaft. (2020). *Der Mittelstand ist Garant für Stabilität und Fortschritt*. BVMW. <https://www.bvmw.de/themen/mittelstand/zahlen-fakten/>, aufgerufen am 13.12.2020
- Chamoni, P., & Gluchowski, P. (2017). Business Analytics—State of the Art. *Controlling & Management Review*, 61(4), 8–17. <https://doi.org/10.1007/s12176-017-0030-6>
- Chaudhuri, S., & Dayal, U. (1997). An Overview of Data Warehousing and OLAP Technology. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 26(1), 65–74. Scopus. <https://doi.org/10.1145/248603.248616>
- Chaudhuri, Surajit, Dayal, U., & Narasayya, V. (2011). An overview of business intelligence technology. *Communications of the ACM*, 54(8), 88–98. <https://doi.org/10.1145/1978542.1978562>
- Chen, Chiang, & Storey. (2012). Business Intelligence and Analytics: From Big Data to Big Impact. *MIS Quarterly*, 36(4), 1165. <https://doi.org/10.2307/41703503>
- Chen, M., Mao, S., & Liu, Y. (2014). Big Data: A Survey. *Mobile Networks and Applications*, 19(2), 171–209. <https://doi.org/10.1007/s11036-013-0489-0>
- Dedić, N., & Stanier, C. (2017). Towards Differentiating Business Intelligence, Big Data, Data Analytics and Knowledge Discovery. In F. Piazolo, V. Geist, L. Brehm, & R. Schmidt (Hrsg.), *Innovations in Enterprise Information Systems Management and Engineering* (Bd. 285, S. 114–122). Springer International Publishing. https://doi.org/10.1007/978-3-319-58801-8_10
- Diehl, A. (2019, März 25). User Stories—Definition, Struktur und Beispiele. *Andreas Diehl (#DNO)*. <https://digitaleneuordnung.de/blog/user-stories/>, aufgerufen am 25.01.2021
- Duan, L., & Xiong, Y. (2015). Big data analytics and business analytics. *Journal of Management Analytics*, 2(1), 1–21. <https://doi.org/10.1080/23270012.2015.1020891>

- El-Adaileh, N. A., & Foster, S. (2019). Successful business intelligence implementation: A systematic literature review. *Journal of Work-Applied Management*, 11(2), 121–132. <https://doi.org/10.1108/JWAM-09-2019-0027>
- Ereth, J., & Kemper, H.-G. (2016). Business Analytics und Business Intelligence. *Controlling*, 28(8–9), 458–464. <https://doi.org/10.15358/0935-0381-2016-8-9-458>
- Fischer, T. M., Lueg, K.-E., Steuernagel, M., Mauch-Maier, B., Schüler, F., Hofbeck, D., & Schneck, L. (2020). Business Analytics in Shared Service Organisationen. In K.-E. Lueg & T. M. Fischer (Hrsg.), *Erfolgreiche Digitale Transformation von Shared Services* (S. 147–187). Springer Fachmedien Wiesbaden. <https://doi.org/10.1007/978-3-658-30484-3>
- Gluchowski, P. (2016). Business Analytics – Grundlagen, Methoden und Einsatzpotenziale. *HMD Praxis der Wirtschaftsinformatik*, 53(3), 273–286. <https://doi.org/10.1365/s40702-015-0206-5>
- Gökalp, M. O., Kayabay, K., Zaki, M., & Koçyi, A. (2017). *Big-Data Analytics Architecture for Businesses: A comprehensive review on new open-source big-data tools*. 35.
- Gregor, S., & Hevner, A. R. (2013). Positioning and Presenting Design Science Research for Maximum Impact. *MIS Quarterly*, 37(2), 337–355. <https://doi.org/10.25300/MISQ/2013/37.2.01>
- Gronau, N., Thim, C., & Fohrholz, C. (2016). Business Analytics in der deutschen Praxis. *Controlling*, 28(8–9), 472–479. <https://doi.org/10.15358/0935-0381-2016-8-9-472>
- Hahne, M. (2014). In *Modellierung von Business-Intelligence-Systemen: Leitfaden für erfolgreiche Projekte auf Basis flexibler Data-Warehouse-Architekturen*. dpunkt.verlag. <http://ebookcentral.proquest.com/lib/erlangen/detail.action?docID=1734322>
- Hahne, M., Wiener, A., & Proff, D. U. (2016). Architekturkonzepte und Modellierungsverfahren für BI-Systeme. In P. Gluchowski & P. Chamoni (Hrsg.), *Analytische Informationssysteme* (S. 147–185). Springer Berlin Heidelberg. <https://doi.org/10.1007/978-3-662-47763-2>
- Hevner, A. R. (2007). A Three Cycle View of Design Science Research. *Scandinavian Journal of Information Systems*, 19, 7.
- Hevner, March, Park, & Ram. (2004). Design Science in Information Systems Research. *MIS Quarterly*, 28(1), 75. <https://doi.org/10.2307/25148625>
- Hoffjan, A., & Rohe, M. (2018). Konzeptionelle Analyse von Self-Service Business Intelligence und deren Gestaltungsmöglichkeiten. In A. Wieseckahn & M. Kißler (Hrsg.), *Erfolgreiches Controlling: Theorie, Praxis und Perspektiven* (1. Aufl., S. 99–112). Nomos Verlagsgesellschaft mbH & Co. KG. <https://doi.org/10.5771/9783845288741-99>
- Husemann, B., Lechtenborger, J., & Vossen, G. (2000). *Conceptual Data Warehouse Design*. 12.
- Institut für Mittelstandsforchung (IfM) Bonn. (2020). *Informationen zum Mittelstand aus erster*

Hand.

- Japkowicz, N., & Shah, M. (2011). *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press. <https://doi.org/10.1017/CBO9780511921803>
- Jourdan, Z., Rainer, R. K., & Marshall, T. E. (2008). Business Intelligence: An Analysis of the Literature. *Information Systems Management*, 25(2), 121–131. <https://doi.org/10.1080/10580530801941512>
- Kaschny, M., Nolden, M., & Schreuder, S. (2015). *Innovationsmanagement im Mittelstand*. Springer Fachmedien Wiesbaden. <https://doi.org/10.1007/978-3-658-02545-8>
- Kimball, R., & Ross, M. (2013). *The data warehouse toolkit: The definitive guide to dimensional modeling* (Third edition). John Wiley & Sons, Inc.
- Kohlhammer, J., Proff, D. U., & Wiener, A. (2016). Der Markt für Visual Business Analytics. In P. Gluchowski & P. Chamoni (Hrsg.), *Analytische Informationssysteme* (S. 303–323). Springer Berlin Heidelberg. <https://doi.org/10.1007/978-3-662-47763-2>
- Larson, D., & Chang, V. (2016). A review and future direction of agile, business intelligence, analytics and data science. *International Journal of Information Management*, 36(5), 700–710. <https://doi.org/10.1016/j.ijinfomgt.2016.04.013>
- Lindner, D. (2020). *Forschungsdesigns der Wirtschaftsinformatik Empfehlungen für die Bachelor- und Masterarbeit*.
- Llave, M. R. (2019). A Review of Business Intelligence and Analytics in Small and Medium-Sized Enterprises: *International Journal of Business Intelligence Research*, 10(1), 19–41. <https://doi.org/10.4018/IJBIR.2019010102>
- Luhn, H. P. (1958). A Business Intelligence System. *IBM Journal of Research and Development*, 2(4), 314–319. <https://doi.org/10.1147/rd.24.0314>
- March, S. T., & Hevner, A. R. (2007). Integrated decision support systems: A data warehousing perspective. *Decision Support Systems*, 43(3), 1031–1043. <https://doi.org/10.1016/j.dss.2005.05.029>
- Moody, D. L. (2000). *From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design*. 12.
- Muntean, M., & Surcel, T. (2013). Agile BI □ The Future of BI. *Informatica Economica*, 17(3/2013), 114–124. <https://doi.org/10.12948/issn14531305/17.3.2013.10>
- Obeidat, M. (2015). *Business Intelligence Technology, Applications, and Trends*. 11(2), 11.
- Olszak, C., & Ziembra, E. (2012). Critical Success Factors for Implementing Business Intelligence Systems in Small and Medium Enterprises on the Example of Upper Silesia, Poland. *Interdisciplinary Journal of Information, Knowledge, and Management*, 7, 129–150.

<https://doi.org/10.28945/1584>

Popovič, A., Puklavec, B., & Oliveira, T. (2019). Justifying business intelligence systems adoption in SMEs: Impact of systems use on firm performance. *Industrial Management & Data Systems*, 119(1), 210–228. <https://doi.org/10.1108/IMDS-02-2018-0085>

Power, D. J. (2007). *A Brief History of Decision Support Systems*. <http://dssresources.com/history/dsshistory.html>

Qamar, U., & Raza, M. S. (2020). *Data Science Concepts and Techniques with Applications*. Springer Singapore. <https://doi.org/10.1007/978-981-15-6133-7>

Saecker, M., & Markl, V. (2013). Big Data Analytics on Modern Hardware Architectures: A Technology Survey. In M.-A. Aufaure & E. Zimányi (Hrsg.), *Business Intelligence* (Bd. 138, S. 125–149). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-36318-4_6

Schwaber, K., & Sutherland, J. (2017). *The Scrum Guide—The Definitive Guide to Scrum: The Rules of the Game*.

Stodder, D. (2015). *Visual Analytics for Making Smarter Decisions Faster*. 44.

Sun, Y., Wong, A. K. C., & Kamel, M. S. (2009). CLASSIFICATION OF IMBALANCED DATA: A REVIEW. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(04), 687–719. <https://doi.org/10.1142/S0218001409007326>

Sun, Z., Sun, L., & Strang, K. (2018). Big Data Analytics Services for Enhancing Business Intelligence. *Journal of Computer Information Systems*, 58(2), 162–169. <https://doi.org/10.1080/08874417.2016.1220239>

Sun, Z., Zou, H., & Strang, K. (2015). Big Data Analytics as a Service for Business Intelligence. In M. Janssen, M. Mäntymäki, J. Hidders, B. Klievink, W. Lamersdorf, B. van Loenen, & A. Zuiderwijk (Hrsg.), *Open and Big Data Management and Innovation* (Bd. 9373, S. 200–211). Springer International Publishing. https://doi.org/10.1007/978-3-319-25013-7_16

Trieu, V.-H. (2017). Getting value from Business Intelligence systems: A review and research agenda. *Decision Support Systems*, 93, 111–124. <https://doi.org/10.1016/j.dss.2016.09.019>

van der Aalst, W. (2016). *Process Mining—Data Science in Action*. Springer Berlin Heidelberg. <http://link.springer.com/10.1007/978-3-662-49851-4>

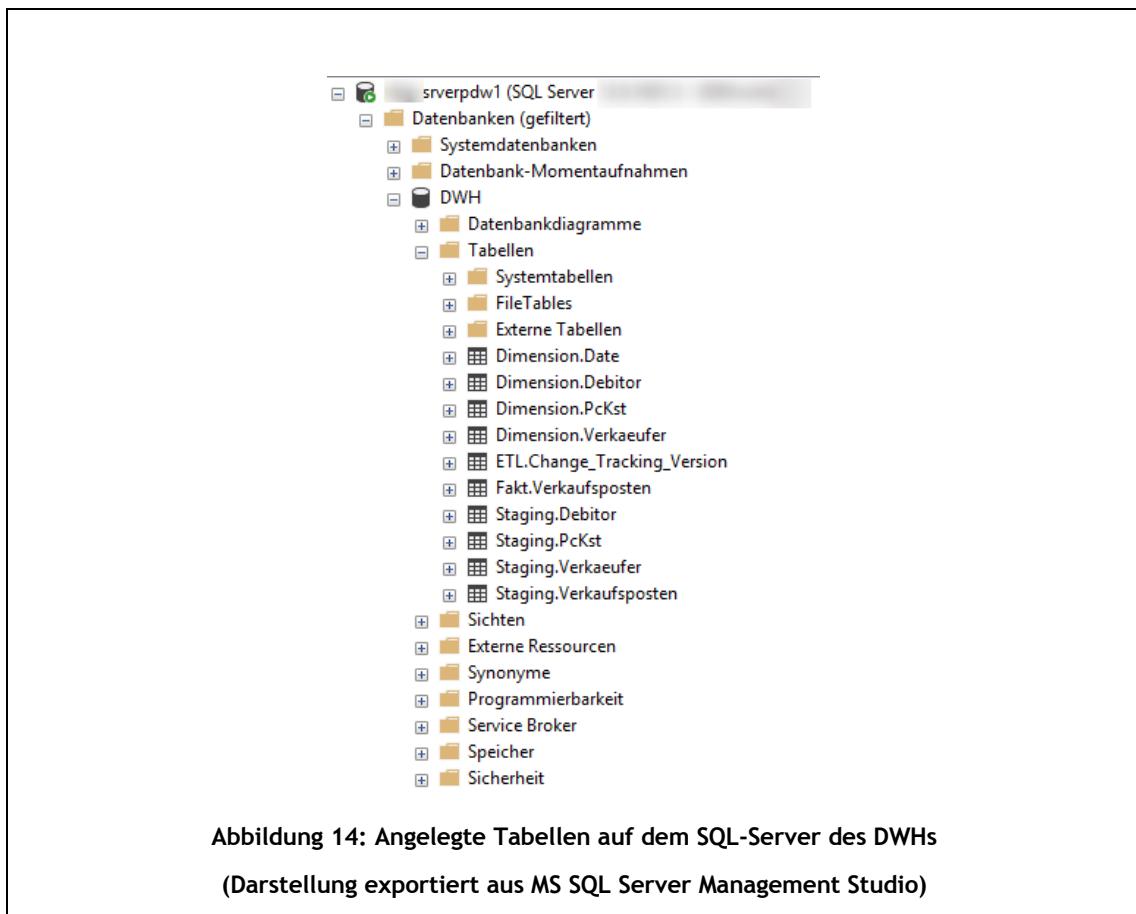
Wani, A. A., & Raina, B. L. (2019). Issues and handy solutions addressed at every stage in real time data warehousing, I.E. ETL (Extraction, transformation & loading). *International Journal of Engineering and Advanced Technology*, 8(5 Special Issue 3), 344–348. Scopus. <https://doi.org/10.35940/ijeat.E1100.0785S319>

Welter, Friederike, May-Strobl, E., Wolter, H.-J., & Günterberg, B. (2014). *Mittelstand im Wandel* (Research Report No. 232; IfM-Materialien). Institut für Mittelstandsforschung (IfM) Bonn.

Anhang

Umsatz Regionen Österreich	Standort	Umsatz			DB			Marge in %					
		YTD 01 20	YTD 01 21	Differenz	+/- %	YTD 01 20	YTD 01 21	Differenz	+/- %	YTD 01 20	YTD 01 21	Differenz	+/- %
Austria Ost	501 SDT KG W												
Austria Mitte	502 SDT KG KLG												
	503 SDT KG LIN												
Austria West	504 SDT KG INB												
Deutschland													
Baden-Wuerttemberg	016 SD EDV HEB												
	061 PROTEAM HEB												
Bayern Nord	011 SD EDV NBG												
	013 SD EDV ING												
Bayern Ost	012 SD EDV RGB												
	015 SD EDV BTW												
Bayern Sued	021 SDS MUC												
CCC	031 CCC MUC												
Sachsen	014 SD EDV DRS												
Gesamtergebnis													

Abbildung 13: Händisch erstellter Vertriebsbericht der Vertriebsleitung
(Darstellung exportiert aus MS Excel)



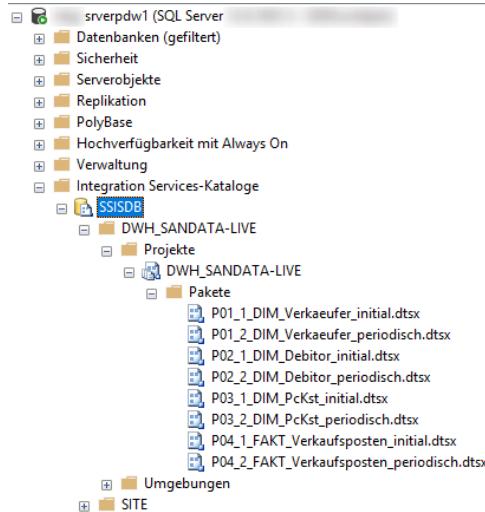
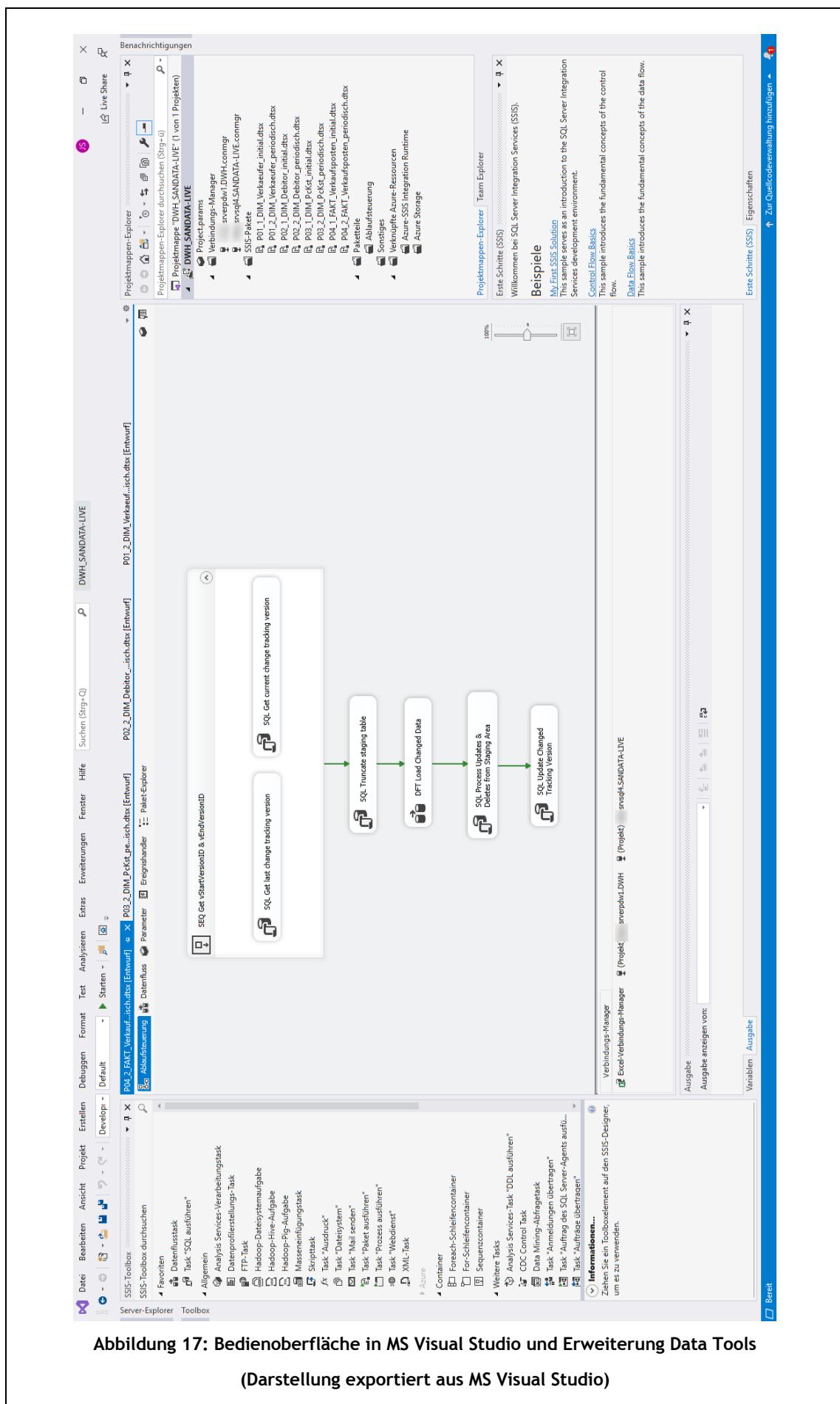


Abbildung 15: Bereitgestellte SSIS-Pakete auf dem SQL-Server des DWHs
(Darstellung exportiert aus MS SQL Server Management Studio)

	Table_Name	Change_Tracking_Version
1	Debitor	2278
2	PcKst	2118
3	Verkaeufer	2278
4	Verkaufsposten	2278

Abbildung 16: Auszug aus der Changetracking-Versionstabelle
(Darstellung exportiert aus MS SQL Server Management Studio)



SQL-Skripte

Im Folgenden ist zum einen das SQL-Skript zur Initialisierung des DWH (Quellcode 10, S.62 bis S. 67) und eine Übersicht zu den in den jeweiligen SSIS-Paketen verwendeten SQL-Fragmenten (Quellcode 11, S.67 bis S. 74) aufgeführt. Abschnitte sind durch entsprechende Kommentierungen im Code getrennt.

```

001 -----CREATE Database-----
002 CREATE DATABASE DWH;
003 GO
004 -----CREATE Schemas-----
005 USE [DWH]
006 GO
007 CREATE SCHEMA [Dimension]
008 GO
009 CREATE SCHEMA [Fakt]
010 GO
011 CREATE SCHEMA [Staging]
012 GO
013 CREATE SCHEMA [ETL]
014 GO
015 -----CREATE Dimensions - und Fakttabellen-----
016 SET ANSI_NULLS ON
017 GO
018 SET QUOTED_IDENTIFIER ON
019 GO
020 USE [DWH]
021 GO
022
023 -- CREATE DIM Debitor
024 CREATE TABLE Dimension.Debitor(
025     [Debitor_SK] [int] IDENTITY(1,1) NOT NULL,
026     [Debitor_ID] [nvarchar](20) NOT NULL,
027     [Versionsnr] [bigint] NOT NULL, --timestamppt
028     [Name] [nvarchar](50) NOT NULL, --name
029     [Name2] [nvarchar](50) NOT NULL, --name2
030     [Adresse] [nvarchar](50) NOT NULL,
031     [Adresse2] [nvarchar](50) NOT NULL,
032     [Stadt] [nvarchar](30) NOT NULL,
033     [Laender_Code] [nvarchar](10) NOT NULL, --countrycode
034     [Last_Date_Modified] [datetime] NOT NULL,
035     [PLZ] [nvarchar](20) NOT NULL,
036     [Bundesland] [nvarchar](30) NOT NULL,
037     [active] [tinyint] NOT NULL,
038     [lastChange] datetime NOT NULL
039 CONSTRAINT PK_Debitor_SK PRIMARY KEY([Debitor_SK])

```

```

040      )
041
042      -- CREATE DIM Verkaeufer
043      CREATE TABLE Dimension.[Verkaeufer] (
044          [Verkaeufer_SK] [int] IDENTITY(1,1) NOT NULL,
045          [Verkaeufer_ID] [nvarchar](10) NOT NULL,
046          [Versionsnr] [bigint] NOT NULL,
047          [Name] [nvarchar](50) NOT NULL,
048          [PcKst] [nvarchar](20) NOT NULL,
049          [Echter_Verkaeufer] [tinyint] NOT NULL,
050          [active] [tinyint] NOT NULL,
051          [lastChange] datetime NOT NULL
052      CONSTRAINT PK_Verkaeufer_SK PRIMARY KEY ([Verkaeufer_SK] ASC)
053      )
054
055      -- CREATE DIM PcKst
056      CREATE TABLE Dimension.[PcKst] (
057          [PcKst_SK] [int] IDENTITY(1,1) NOT NULL,
058          [PcKst_ID] [nvarchar](20) NOT NULL,
059          [Versionsnr] [bigint] NOT NULL,
060          [Dimension_Code] [nvarchar](20) NOT NULL,
061          [Name] [nvarchar](50) NOT NULL,
062          [Region] [nvarchar](50) NULL,
063          [Land] [nvarchar](50) NULL,
064          [Standort] [nvarchar](50) NULL,
065          [Blocked] [tinyint] NOT NULL,
066          [active] [tinyint] NOT NULL,
067          [lastChange] datetime NOT NULL
068      CONSTRAINT PK_PcKst_SK PRIMARY KEY ([PcKst_SK])
069      )
070
071      --CREATE DIM Date
072      if exists (SELECT object_id FROM sys.tables WHERE [name] = 'Dimension.Date') DROP TABLE Dimension.Date;
073      CREATE TABLE Dimension.Date (
074          Date_ID date NOT NULL
075          , WochentagNum int NULL -- 1 bis 7
076          , WochentagName nvarchar(30) NULL -- Montag...
077          , MonatstagNum int NULL -- 1,2 bis 31
078          , JahrestagNum int NULL -- 1,2 bis 365
079          , KWNum int NULL -- 1,2 bis 54
080          , MonatNum int NULL -- 1,2 bis 12
081          , MonatName nvarchar(30) NULL -- Januar...
082          , QuartalNum int NULL -- 1,2 bis 4
083          , JahrNum int NULL -- 2019...
084          ,CONSTRAINT PK_Date_ID PRIMARY KEY (Date_ID)
085      )
086

```

```

087  --INIT DIM Date
088  SET NOCOUNT ON
089  SET LANGUAGE German;
090  DECLARE @minDate date = '01.01.2019'
091  , @maxDate date = '31.12.2022'
092  , @loadDate date;
093  SET @loadDate = @minDate;
094
095  WHILE @loadDate <= @maxDate
096  BEGIN
097      if not exists (SELECT Date_ID from Dimension.Date WHERE Date_ID = @loadDate)
098          INSERT INTO Dimension.Date
099          SELECT
100              @loadDate as calendarDate
101              , DATEPART(dw, @loadDate)
102              , DATENAME(dw, @loadDate)
103              , DATEPART(dd, @loadDate)
104              , DATEPART(dy, @loadDate)
105              , DATEPART(wk, @loadDate)
106              , DATEPART(m, @loadDate)
107              , DATENAME(m, @loadDate)
108              , DATEPART(q, @loadDate)
109              , DATEPART(yy, @loadDate)
110      SET @loadDate = DATEADD(dd, 1, @loadDate);
111  END
112
113  -- CREATE Fakt Verkaufsposten
114  CREATE TABLE Fakt.[Verkaufsposten](
115      [Verkaufsposten_ID] [int] NOT NULL,
116      [Versionsnr] [bigint] NOT NULL,
117      [Typ] [int] NOT NULL,
118      [Verkaufsartefaktnr] [nvarchar](20) NOT NULL,
119      [Beschreibung] [nvarchar](50) NOT NULL,
120      [Beschreibung2] [nvarchar](50) NOT NULL,
121      [Einheit] [nvarchar](10) NOT NULL,
122      [Menge] [decimal](38, 20) NOT NULL, --Quantity
123      [Verkaufspreis] [decimal](38, 20) NOT NULL,
124      [Einstandspreis] [decimal](38, 20) NOT NULL, --Unit Cost (LCY)
125      [Zeilenrabatt_Prozent] [decimal](38, 20) NOT NULL,
126      [Zeilenrabattbetrag] [decimal](38, 20) NOT NULL,
127      [Rechnungsbetrag_anteilig] [decimal](38, 20) NOT NULL,
128      [Deckungsbeitrag] [decimal](38, 20) NOT NULL,
129      [PcKst_SK] int NULL,
130      [PcKst_ID] [nvarchar](20) NOT NULL,
131      [Debitor_SK] int NULL,
132      [Debitor_ID] [nvarchar](20) NOT NULL,
133      [Waehrung_Code] [nvarchar](10) NOT NULL, --Ggf auch Usd
134      [Verkaeufer_SK] int NULL,

```

```

135      [Verkaeufer_ID] [nvarchar](10) NOT NULL,
136      [Belegart] [int] NOT NULL,
137      [Belegnr] [nvarchar](20) NOT NULL,
138      [Beleg_Zeilennr] [int] NOT NULL,
139      [Buchungsdatum] [date] NOT NULL,
140      [Belegdatum] [date] NOT NULL,
141      [Open] [tinyint] NOT NULL,
142      [Prozess_Code] [nvarchar](10) NOT NULL,
143      [Adjusted] [tinyint] NOT NULL,
144      [Dimension_Satz_ID] [int] NOT NULL,
145      [Sales_Admin] [nvarchar](20) NOT NULL,
146      [Zweiter_Verkaeufer_SK] int NULL,
147      [Zweiter_Verkaeufer_ID] [nvarchar](10) NULL,
148      [Zweiter_Verkaeufer_Anteil_Prozent] [decimal](38, 20) NULL,
149      [active] [tinyint] NOT NULL,
150      [lastChange] datetime NOT NULL,
151      [Korrektur] tinyint NOT NULL
152      CONSTRAINT PK_Verkaufsposten_ID PRIMARY KEY CLUSTERED (Verkaufsposten_ID ASC),
153          CONSTRAINT FK_Date_Belegdatum FOREIGN KEY (Belegdatum) REFERENCES Dimension.Date(Date_ID),
154          CONSTRAINT FK_Date_Buchungsdatum FOREIGN KEY (Buchungsdatum) REFERENCES Dimension.Date(Date_ID),
155          CONSTRAINT FK_Debitor FOREIGN KEY (Debitor_SK) REFERENCES Dimension.Debitor(Debitor_SK),
156          CONSTRAINT FK_PcKst FOREIGN KEY (PcKst_SK) REFERENCES Dimension.PcKst(PcKst_SK),
157          CONSTRAINT FK_Verkaeufer FOREIGN KEY (Verkaeufer_SK) REFERENCES Dimension.Verkaeufer(Verkaeufer_SK),
158          CONSTRAINT FK_ZweiterVerkaeufer FOREIGN KEY (Zweiter_Verkaeufer_SK) REFERENCES Dimension.Verkaeufer(Verkaeufer_SK),
159      )
160
161      -- CREATE ETL Changetracking-Version
162      CREATE TABLE [etl].[Change_Tracking_Version]
163      (
164          [Table_Name] VARCHAR(100) NOT NULL PRIMARY KEY
165          , [Change_Tracking_Version] [BIGINT] NULL
166      )
167
168      -----CREATE Staging Tables-----
169      --CREATE Staging Table Verkaeufer
170      CREATE TABLE Staging.[Verkaeufer](
171          [Verkaeufer_ID] [nvarchar](10) NOT NULL,
172          [Versionsnr] [bigint] NULL,
173          [Name] [nvarchar](50) NULL,
174          [PcKst] [nvarchar](20) NULL,
175          [Echter_Verkaeufer] [tinyint] NULL,
176          [Operation] [CHAR](10) NOT NULL, --CRUD
177          [active] [tinyint] NULL,
178          [lastChange] datetime NULL
179      )

```

```

180
181    --CREATE Staging Table Debitor
182    CREATE TABLE Staging.[Debitor](
183        [Debitor_ID] [nvarchar](20) NOT NULL,
184        [Versionsnr] [bigint] NULL,      --timestamppt
185        [Name] [nvarchar](50) NULL,     --name
186        [Name2] [nvarchar](50) NULL,    --name2
187        [Adresse] [nvarchar](50) NULL,
188        [Adresse2] [nvarchar](50) NULL,
189        [Stadt] [nvarchar](30) NULL,
190        [Laender_Code] [nvarchar](10) NULL, --countrycode
191        [Last_Date_Modified] [datetime] NULL,
192        [PLZ] [nvarchar](20) NULL,
193        [Bundesland] [nvarchar](30) NULL,
194        [Operation] [CHAR](10) NULL, --CRUD
195        [active] [tinyint] NULL,
196        [lastChange] datetime NULL
197    )
198
199    --CREATE Staging Table PCKSt
200    CREATE TABLE Staging.[PcKst](
201        [PcKst_ID] [nvarchar](20) NOT NULL,
202        [Versionsnr] [bigint] NULL,
203        [Dimension_Code] [nvarchar](20) NULL,
204        [Name] [nvarchar](50) NULL,
205        [Region] [nvarchar](50) NULL,
206        [Land] [nvarchar](50) NULL,
207        [Standort] [nvarchar](50) NULL,
208        [Blocked] [tinyint] NULL,
209        [active] [tinyint] NULL,
210        [lastChange] datetime NULL,
211        [Operation] [CHAR](10) NULL, --CRUD
212    )
213
214    --CREATE Staging Table Verkaufsposten
215    CREATE TABLE Staging.Verkaufsposten(
216        [Verkaufsposten_ID] [int] NOT NULL,
217        [Versionsnr] [bigint] NULL,
218        [Typ] [int] NULL,
219        [Verkaufsartefaktnr] [nvarchar](20) NULL,
220        [Beschreibung] [nvarchar](50) NULL,
221        [Beschreibung2] [nvarchar](50) NULL,
222        [Einheit] [nvarchar](10) NULL,
223        [Menge] [decimal](38, 20) NULL, --Quantity
224        [Verkaufspreis] [decimal](38, 20) NULL,
225        [Einstandspreis] [decimal](38, 20) NULL, --Unit Cost (LCY)
226        [Zeilenrabatt_Prozent] [decimal](38, 20) NULL,
227        [Zeilenrabattbetrag] [decimal](38, 20) NULL,

```

```

228     [Rechnungsbetrag_anteilig] [decimal](38, 20) NOT NULL,
229     [Deckungsbeitrag] [decimal](38, 20) NOT NULL,
230     [PcKst_SK] int NULL,
231     [PcKst_ID] [nvarchar](20) NULL,
232     [Debitor_SK] int NULL,
233     [Debitor_ID] [nvarchar](20) NULL,
234     [Waehrung_Code] [nvarchar](10) NULL, --Ggf auch Usd
235     [Verkaeufer_SK] int NULL,
236     [Verkaeufer_ID] [nvarchar](10) NULL,
237     [Belegart] [int] NULL,
238     [Belegnr] [nvarchar](20) NULL,
239     [Beleg_Zeilennr] [int] NULL,
240     [Buchungsdatum] [date] NULL,
241     [Belegdatum] [date] NULL,
242     [Open] [tinyint] NULL,
243     [Prozess_Code] [nvarchar](10) NULL,
244     [Adjusted] [tinyint] NULL,
245     [Dimension_Satz_ID] [int] NULL,
246     [Sales_Admin] [nvarchar](20) NULL,
247     [Zweiter_Verkaeufer_SK] int NULL,
248     [Zweiter_Verkaeufer_ID] [nvarchar](10) NULL,
249     [Zweiter_Verkaeufer_Anteil_Prozent] [decimal](38, 20) NULL,
250     [active] [tinyint] NULL,
251     [lastChange] datetime NULL,
252     Korrektur tinyint NULL,
253     Operation char(10) NULL
254   )

```

Quellcode 10: create_DWH.sql, Skript zur Initialisierung des Data Warehouse

```

001 -----Activate Changetracking-----
002 ALTER DATABASE [ChangeTracking] SET change_tracking = ON (change_retention = 14
days)
003 GO
004 ALTER TABLE [dbo].[Salesperson_Purchaser]
005 ENABLE change_tracking WITH (track_columns_updated = ON)
006 GO
007 ALTER TABLE [dbo].Customer
008 ENABLE change_tracking WITH (track_columns_updated = ON)
009 GO
010 ALTER TABLE [dbo].[Dimension Value]
011 ENABLE change_tracking WITH (track_columns_updated = ON)
012 GO
013 ALTER TABLE [dbo].[00 SD Gruppe$Sales Ledger Entry]
014 ENABLE change_tracking WITH (track_columns_updated = ON)
015 GO
016 ALTER TABLE [dbo].[00 SD Gruppe$Sales Ledger Entry]
017 DISABLE CHANGE_TRACKING;

```

```

018
019  -----SSIS-Pakete periodisch-----
020  -----Verkaeufer -----
021  --Get last change tracking version
022  SELECT ? = (SELECT [Change_Tracking_Version]
023  FROM [ETL].[Change_Tracking_Version]
024  WHERE Table_Name = 'Verkaeufer')
025
026  --Get current change tracking Version
027  SELECT ? = CHANGE_TRACKING_CURRENT_VERSION()
028
029  --SQL Truncate staging table
030  TRUNCATE TABLE [Staging].[Verkaeufer]
031
032  -- GET Changed Data
033  SELECT ct.[Code]
034  , s.[timestamp]
035  , s.[Name]
036  , s.[Global Dimension 1 Code]
037  , s.[Real Salesperson]
038  , CAST(ct.SYS_CHANGE_OPERATION AS CHAR(1)) [Operation]
039  FROM CHANGETABLE(CHANGES [dbo].[Salesperson_Purchaser], ?) ct
040  LEFT JOIN [dbo].[Salesperson_Purchaser] s
041  ON s.[Code] = ct.[Code]
042  WHERE (
043    SELECT MAX(v)
044    FROM (VALUES(ct.SYS_CHANGE_VERSION),
045           (ct.SYS_CHANGE_CREATION_VERSION)) AS VALUE(v)) <= ?
046
047  -- UPDATE
048  UPDATE dim
049  SET dim.[Versionsnr] = stage.[Versionsnr]
050  , dim.[Name] = stage.[Name]
051  , dim.[PcKst] = stage.[PcKst]
052  , dim.[Echter_Verkaeufer] = stage.[Echter_Verkaeufer]
053  , dim.[lastChange] = stage.[lastChange]
054  FROM [Dimension].[Verkaeufer] dim
055  INNER JOIN [Staging].[Verkaeufer] stage
056  ON dim.[Verkaeufer_ID] = stage.[Verkaeufer_ID]
057  WHERE stage.Operation = 'U'
058
059  --DELETES
060  UPDATE dim
061  SET dim.active = stage.active
062  , dim.[lastChange] = stage.[lastChange]
063  FROM [Dimension].[Verkaeufer] dim
064  INNER JOIN [Staging].[Verkaeufer] stage
065  ON dim.[Verkaeufer_ID] = stage.[Verkaeufer_ID]

```

```

066 WHERE stage.Operation = 'D'
067
068 --Update ETL Change Tracking Version
069 MERGE [ETL].[Change_Tracking_Version] AS target
070 USING (SELECT 'Verkaeufer' [Table_Name], ? [Version_ID]
071 ) AS source
072 ON target.Table_Name = source.Table_Name
073 WHEN MATCHED
074 THEN UPDATE
075 SET target.Change_Tracking_Version = source.Version_ID
076 WHEN NOT MATCHED
077 THEN INSERT (Table_Name, Change_Tracking_Version)
078 VALUES (source.Table_Name, source.Version_ID)
079
080 -----Debitor-----
081 --Get last change tracking version
082 SELECT ? = (SELECT [Change_Tracking_Version]
083 FROM [ETL].[Change_Tracking_Version]
084 WHERE Table_Name = 'Debitor')
085
086 --Get current change tracking Version
087 SELECT ? = CHANGE_TRACKING_CURRENT_VERSION()
088
089 --SQL Truncate staging table
090 TRUNCATE TABLE [Staging].[Debitor]
091
092 -- GET Changed Data
093 SELECT ct.[No_]
094 , c.[timestamp]
095 , c.[Name]
096 , c.[Name 2]
097 , c.[Address]
098 , c.[Address 2]
099 ,c.[City]
100 , c.[Country_Region Code]
101 ,c.[Last Date Modified]
102 ,c.[Post Code]
103 ,c.[County]
104 , CAST(ct.SYS_CHANGE_OPERATION AS CHAR(1)) [Operation]
105 FROM CHANGETABLE(CHANGES [dbo].Customer, ?) ct
106 LEFT JOIN [dbo].Customer c
107 ON c.[No_] = ct.[No_]
108 WHERE (
109     SELECT MAX(v)
110     FROM (VALUES(ct.SYS_CHANGE_VERSION),
111                 (ct.SYS_CHANGE_CREATION_VERSION)) AS VALUE(v) ) <= ?
112
113 -- UPDATE

```

```

114  UPDATE dim
115  SET dim.[Versionsnr] = stage.[Versionsnr]
116  , dim.[Name] = stage.[Name]
117  , dim.[Name2] = stage.[Name2]
118  , dim.Adresse = stage.Adresse
119  , dim.Adresse2 = stage.Adresse2
120  , dim.Stadt = stage.Stadt
121  , dim.[Laender_Code] = stage.[Laender_Code]
122  , dim.[Last_Date_Modified] = stage.[Last_Date_Modified]
123  , dim.[PLZ] = stage.[PLZ]
124  , dim.[Bundesland] = stage.[Bundesland]
125  , dim.[lastChange] = stage.[lastChange]
126  FROM [Dimension].[Debitor] dim
127  INNER JOIN [Staging].[Debitor] stage
128  ON dim.[Debitor_ID] = stage.[Debitor_ID]
129  WHERE stage.Operation = 'U'
130
131  --DELETES
132  UPDATE dim
133  SET dim.active = stage.active
134  , dim.[lastChange] = stage.[lastChange]
135  FROM [Dimension].[Debitor] dim
136  INNER JOIN [Staging].[Debitor] stage
137  ON dim.[Debitor_ID] = stage.[Debitor_ID]
138  WHERE stage.Operation = 'D'
139
140  --Update ETL Change Tracking Version
141  MERGE [ETL].[Change_Tracking_Version] AS target
142  USING (SELECT 'Debitor' [Table_Name], ? [Version_ID]
143  ) AS source
144  ON target.Table_Name = source.Table_Name
145  WHEN MATCHED
146  THEN UPDATE
147  SET target.Change_Tracking_Version = source.Version_ID
148  WHEN NOT MATCHED
149  THEN INSERT (Table_Name, Change_Tracking_Version)
150  VALUES (source.Table_Name, source.Version_ID)
151
152  -----PcKst-----
153  --Get last change tracking version
154  SELECT ? = (SELECT [Change_Tracking_Version]
155  FROM [ETL].[Change_Tracking_Version]
156  WHERE Table_Name = 'PcKst')
157
158  --Get current change tracking Version
159  SELECT ? = CHANGE_TRACKING_CURRENT_VERSION()
160
161  --SQL Truncate staging table

```

```

162  TRUNCATE TABLE [Staging].[PcKst]
163
164  -- GET Changed Data
165  SELECT ct.[Dimension Code]
166  ,ct.Code
167  , p.[timestamp]
168  , p.[Name]
169  , p.[Blocked]
170  , CAST(ct.SYS_CHANGE_OPERATION AS CHAR(1)) [Operation]
171  FROM CHANGETABLE(CHANGES [dbo].[Dimension Value], ?) ct
172  LEFT JOIN [dbo].[Dimension Value] p
173  ON p.[Dimension Code] = ct.[Dimension Code] AND p.Code = ct.Code
174  WHERE (
175      SELECT MAX(v)
176      FROM (VALUES(ct.SYS_CHANGE_VERSION),
177                  (ct.SYS_CHANGE_CREATION_VERSION)) AS VALUE(v) ) <= ?
178
179  -- UPDATE
180  UPDATE dim
181  SET dim.[Versionsnr] = stage.[Versionsnr]
182  , dim.[Name] = stage.[Name]
183  , dim.[Dimension_Code] = stage.[Dimension_Code]
184  , dim.[Region] = stage.[Region]
185  , dim.[Standort] = stage.Standort
186  , dim.Land = stage.Land
187  , dim.[Blocked] = stage.[Blocked]
188  , dim.[lastChange] = stage.[lastChange]
189  FROM [Dimension].PcKst dim
190  INNER JOIN [Staging].PcKst stage
191  ON dim.[PcKst_ID] = stage.[PcKst_ID]
192  WHERE stage.Operation = 'U'
193
194  --DELETES
195  UPDATE dim
196  SET dim.active = stage.active
197  , dim.[lastChange] = stage.[lastChange]
198  FROM [Dimension].PcKst dim
199  INNER JOIN [Staging].PcKst stage
200  ON dim.[PcKst_ID] = stage.[PcKst_ID]
201  WHERE stage.Operation = 'D'
202
203  --Update ETL Change Tracking Version
204  MERGE [ETL].[Change_Tracking_Version] AS target
205  USING (SELECT 'PcKst' [Table_Name], ? [Version_ID]
206  ) AS source
207  ON target.Table_Name = source.Table_Name
208  WHEN MATCHED
209  THEN UPDATE

```

```

210  SET target.Change_Tracking_Version = source.Version_ID
211  WHEN NOT MATCHED
212  THEN INSERT (Table_Name, Change_Tracking_Version)
213  VALUES (source.Table_Name, source.Version_ID);
214
215
216  -----Verkaufsposten-----
217  --Get last change tracking version
218  SELECT ? = (SELECT [Change_Tracking_Version]
219  FROM [ETL].[Change_Tracking_Version]
220  WHERE Table_Name = 'Verkaufsposten')
221
222  --Get current change tracking Version
223  SELECT ? = CHANGE_TRACKING_CURRENT_VERSION()
224
225  --SQL Truncate staging table
226  TRUNCATE TABLE [Staging].[Verkaufsposten]
227
228  -- GET Changed Data
229  SELECT ct.[Entry No_] AS Verkaufsposten_ID
230  ,s.[timestamp] AS Versionsnr
231  ,s.Type AS Typ
232  ,s.[No_] AS VerkaufsartefaktNr
233  ,s.Description AS Beschreibung
234  ,s.[Description 2] AS Beschreibung2
235  ,s.[Unit of Measure] AS Einheit
236  ,s.Quantity AS Menge
237  ,s.[Unit Price] AS Verkaufspreis
238  ,s.[Unit Cost (LCY)] AS Einstandspreis
239  ,s.[Line Discount _] AS Zeilenrabatt_Prozent
240  ,s.[Line Discount Amount] AS Zeilenrabattbetrag
241  ,s.[Invoice Amount] AS Rechnungsbetrag_anteilig
242  ,s.[Allow Invoice Disc_]
243  ,s.[Global Dimension 1 Code] AS Pkst_ID
244  ,s.[Sell-to Customer No_] AS Debitor_ID
245  ,s.[Salesperson Code] AS Verkaeufer_ID
246  ,s.[Document Type] AS Belegart
247  ,s.[Document No_] AS Belegnr
248  ,s.[Document Line No_] AS Beleg_Zeilennr
249  ,CAST(s.[Posting Date] AS date) AS Buchungsdatum
250  ,CAST(s.[Document Date] AS date) AS Belegdatum
251  ,s.[Open], [Process Code] AS Prozess_Code, Adjusted
252  ,s.[Dimension Set ID] AS Dimension_Satz_ID
253  ,s.[Sales Admin] AS Sales_Admin
254  ,s.[Second Salesperson Code] AS Zweiter_Verkaeufer_ID
255  ,s.[Second Salesperson Comm_Share] AS Zweiter_Verkaeufer_Anteil_Prozent
256  ,s.[Currency Code] AS Waehrung_Code
257  ,CAST(ct.SYS_CHANGE_OPERATION AS CHAR(1)) [Operation]

```

```

258   FROM CHANGETABLE(CHANGES [dbo].[00 SD Gruppe$Sales Ledger Entry], ?) ct
259   LEFT JOIN [dbo].[00 SD Gruppe$Sales Ledger Entry] s
260   ON s.[Entry No_] = ct.[Entry No_]
261   WHERE (
262     SELECT MAX(v)
263     FROM (VALUES(ct.SYS_CHANGE_VERSION),
264             (ct.SYS_CHANGE_CREATION_VERSION)) AS VALUE(v)) <= ?
265
266   -- UPDATE
267   UPDATE f
268   SET (
269     f.[Versionsnr]=stage.[Versionsnr]
270     ,f.[Typ] = stage.[Typ]
271     ,f.[Verkaufsartefaktnr]=stage.[Verkaufsartefaktnr]
272     ,f.[Beschreibung]=stage.[Beschreibung]
273     ,f.[Beschreibung2]=stage.[Beschreibung2]
274     ,f.[Einheit]=stage.[Einheit]
275     ,f.[Menge]=stage.[Menge]
276     ,f.[Verkaufspreis]=stage.[Verkaufspreis]
277     ,f.[Einstandspreis]=stage.[Einstandspreis]
278     ,f.[Zeilenrabatt_Prozent] =stage.[Zeilenrabatt_Prozent]
279     ,f.[Zeilenrabattbetrag]=stage.[Zeilenrabattbetrag]
280     ,f.[PcKst_SK]=stage.[PcKst_SK]
281     ,f.[PcKst_ID]=stage.[PcKst_ID]
282     ,f.[Debitor_SK]=stage.[Debitor_SK]
283     ,f.[Debitor_ID]=stage.[Debitor_ID]
284     ,f.[Waehrung_Code] =stage.[Waehrung_Code]
285     ,f.[Verkaeufer_SK]=stage.[Verkaeufer_SK]
286     ,f.[Verkaeufer_ID]=stage.[Verkaeufer_ID]
287     ,f.[Belegart]=stage.[Belegart]
288     ,f.[Belegnr]=stage.[Belegnr]
289     ,f.[Beleg_Zeilennr]=stage.[Beleg_Zeilennr]
290     ,f.[Buchungsdatum]=stage.[Buchungsdatum]
291     ,f.[Belegdatum]=stage.[Belegdatum]
292     ,f.[Open]=stage.[Open]
293     ,f.[Prozess_Code] =stage.[Prozess_Code]
294     ,f.[Adjusted]=stage.[Adjusted]
295     ,f.[Dimension_Satz_ID] =stage.[Dimension_Satz_ID]
296     ,f.[Sales_Admin]=stage.[Sales_Admin]
297     ,f.[Zweiter_Verkaeufer_SK]=stage.[Zweiter_Verkaeufer_SK]
298     ,f.[Zweiter_Verkaeufer_ID]=stage.[Zweiter_Verkaeufer_ID]
299     ,f.[Zweiter_Verkaeufer_Anteil_Prozent] =stage.[Zweiter_Verkaeufer_Anteil_Prozent]
300     ,f.[lastChange]=stage.[lastChange]
301     ,f.[Korrektur]=stage.[Korrektur]
302     ,f.[Deckungsbeitrag]=stage.[Deckungsbeitrag]
303     ,f.[Rechnungsbetrag_anteilig] =stage.[Rechnungsbetrag_anteilig]
304   FROM [Fakt].Verkaufsposten f

```

```
305  INNER JOIN [Staging].Verkaufsposten stage
306    ON f.Verkaufsposten_ID = stage.Verkaufsposten_ID
307  WHERE stage.Operation = 'U'
308
309  --DELETES
310  UPDATE f
311  SET f.active = stage.active
312    , f.[lastChange] = stage.[lastChange]
313  FROM [Fakt].Verkaufsposten f
314  INNER JOIN [Staging].Verkaufsposten stage
315    ON f.Verkaufsposten_ID = stage.Verkaufsposten_ID
316  WHERE stage.Operation = 'D'
317
318  --Update ETL Change Tracking Version
319  MERGE [ETL].[Change_Tracking_Version] AS target
320  USING (SELECT 'Verkaufsposten' [Table_Name], ? [Version_ID]
321 ) AS source
322  ON target.Table_Name = source.Table_Name
323 WHEN MATCHED
324 THEN UPDATE
325 SET target.Change_Tracking_Version = source.Version_ID
326 WHEN NOT MATCHED
327 THEN INSERT (Table_Name, Change_Tracking_Version)
328 VALUES (source.Table_Name, source.Version_ID);
```

Quellcode 11: setup_ETL.sql, Übersicht der in den SSIS-Paketen verwendeten SQL-Fragmente

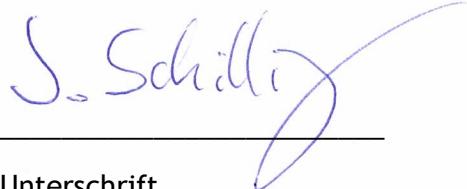
Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit "**Business Intelligence im KMU - Agile Konzeption und Implementierung eines BI-Systems**" selbstständig und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch an keiner anderen Prüfungsbehörde vorgelegen hat.

Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Nürnberg, 15.03.2021

Ort, Datum



A handwritten signature in blue ink, appearing to read "S. Schillig".

Unterschrift