Sonderdruck aus
BI-SPEKTRUM 3/2018

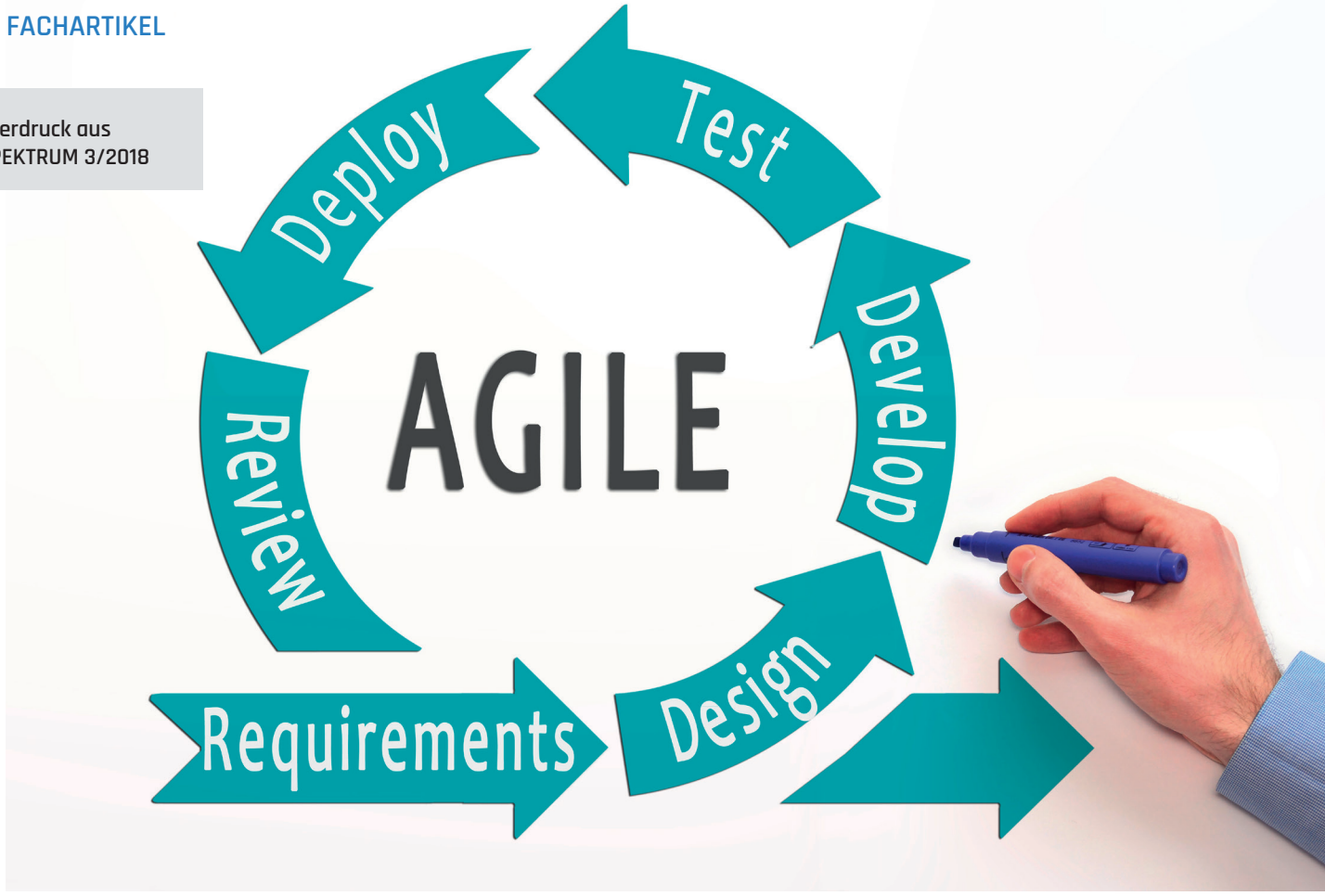

Bild: Shutterstock

Wie SAP HANA agile Methoden im Data Warehousing unterstützt

Agiles Data-Warehouse-Design mit SAP BW/4HANA

Ein Beitrag
von Matthias Gerisch
und Thomas Arnsberg

Seit dem „Manifest für agile Softwareentwicklung" haben agile Methoden einen Siegeszug durch die Softwareentwicklung angetreten. Anders sah es oft aus im Bereich Data Warehousing: Sind agile Methoden für Business-Intelligence-Projekte am Ende gar nicht geeignet? Dieser Beitrag argumentiert, dass mit dem Aufkommen moderner In-Memory-Datenbanken in den letzten Jahren technische und methodische Barrieren gefallen sind, die einer vermehrten Anwendung agiler Methoden in Business-Intelligence-Projekten traditionell im Wege standen. Am Beispiel von agilen Designmethoden wird aufgezeigt, wie agiles Denken wichtige Impulse für die Designpraxis im In-Memory Data Warehouse setzen kann. Dabei geht es für den Praktiker nicht nur darum, sich die agile Methodik zu erschließen, sondern auch darum, überholte Methoden der Business-Warehouse-Entwicklung bewusst und diszipliniert zu „entlernen".

„Agile" Werte, Grundsätze, Prozesse und Methoden haben die Softwareentwicklung seit 2001 fundamental verändert. Dem agilen Ansatz wird dabei in Abgrenzung zum herkömmlichen Wasserfallansatz zugeschrieben, durch einen iterativen Design- und Entwicklungsprozess Software von höherer Qualität und mit größerem Kundennutzen zu erzeugen – und dies auch noch mit einem geringeren Risiko des Scheiterns.

Die Business-Intelligence-Community war demgegenüber reservierter bei der Rezeption agiler Entwicklungsmethoden. Gegen die Anwendbarkeit agiler Methoden auf Data-Warehouse-Projekte wurde unter anderem argumentiert, dass sich datenzentrierte Integrationsprojekte methodisch fundamental von Softwareentwicklungsprojekten unterschieden. Schließlich fielen rund 80 Prozent des Entwicklungsaufwands im Backend-Datenmanagement an und nicht in der „Programmierung" von Berichten. Die deutsche Diskussion zum Thema Agile BI hat in den vergangenen Jahren einen Brückenschlag in Form eines domänenspezifisch

angepassten agilen Vorgehensmodells für die Business Intelligence unternommen [TZK16].

Mit dem Aufkommen von In-Memory-Datenbanken wie SAP HANA oder DB2 BLU in den vergangenen Jahren ist nunmehr technologisch ein günstiges Umfeld für eine vermehrte Anwendung agiler Methoden in der Data-Warehouse-Entwicklung entstanden. Die In-Memory-Technologie ermöglicht eine Reduktion von Persistenzebenen und deren Ablösung durch Echtzeit-Berechnungen von Dimensionsattributen und Key-Performance-Indikatoren. So bewegt sich das Data Warehouse weg vom traditionellen Paradigma der Extraktion-Transformation-Beladung hin zu logischen und föderierten Data-Warehouse-Architekturen [Mor15].

Leitplanken für das Agile DW

Ausgangspunkt jeder Betrachtung agiler Methoden sind die bekannten *Werte* und *Prinzipien* des „Agilen Manifests" [Bec01] (Abbildungen 1 und 2). Auf ihrer Homepage präsentiert die „Agile Allianz" [Agi18] eine Aufstellung agiler Methoden, visualisiert in Form eines U-Bahn-Netzplans (Abbildung 3). Die darin dargestellten agilen Praktiken lassen sich in die beiden Überkategorien Managementmethoden und technische Entwicklungsmethoden differenzieren [Col12]:

Agile Managementmethoden

- Agiles Projektmanagement
- Kollaboration
- User-Storys
- Selbstorganisierte Teams

Agile Entwicklungsmethoden

- Testgetriebene Entwicklung
- Versionsverwaltung
- Automation
- Evolutionäres gutes Design

Der letztgenannte Punkt der Aufzählung bildet den thematischen Schwerpunkt dieses Artikels: die Anwendung agiler Grundsätze auf das Data-Warehouse-Design im Kontext der neuen In-Memory-Datenbanktechnologien. Die Methodendiskussion orientiert sich dabei zwar am konkreten Beispiel von BW/4HANA, der In-Memory-Data-Warehouse-Plattform von SAP. Die dabei angestellten Überlegungen sind jedoch in der Regel auf die Technologien anderer Hersteller übertragbar.

Evolutionäres gutes Design

Dieser Aspekt setzt sich definitorisch aus zwei Komponenten zusammen: aus einem „Was" (gutes Design) und einem „Wie" (evolutionär). Gutes Data-Warehouse-Design wird Eigenschaften aufweisen wie:

- einen kohärenten technischen Rahmen,
- Offenheit gegenüber neuen und sich wandelnden Anforderungen,
- weitgehende technische Schuldenfreiheit,
- Fokus,
- Einfachheit.

Ein Punkt ist in dieser Aufstellung nicht erwähnt: der „Single Point of Truth". Er wird bewusst ausgelassen: Die einheitliche Definition eines Datenelements oder einer Führungskennzahl ist kein *notwendig* anzustrebendes Designprinzip. Das Streben nach „einer Wahrheit" ist zwar nützlich als Richtschnur oder Vision, darf aber nicht im Wege stehen, wenn es darum geht, konkreten Anwendernutzen zu erzeugen. Ansonsten besteht die Gefahr, dass die Suche nach dem heiligen Gral des Single Point of Truth Designentscheidungen auch dann noch prägt, wenn längst die Schwelle zum abnehmenden Grenznutzen überschritten worden ist.

„Evolutionär" verweist auf ein Verständnis von Design als etwas, das nicht ingenieurmäßig in einem großen Entwurf initial erstellt, geprüft, offiziell besiegelt und schlussendlich buchstabengetreu umgesetzt wird. Aus agiler Perspektive sind Design und Architektur dynamische und inkrementelle Lernprozesse. In dem Maße, wie das Domänenwissen im Team in Breite und Tiefe wächst, ist es erlaubt – und sogar ausdrücklich erwünscht –, getroffene Designentscheidungen zu hinterfragen, zu modifizieren oder auch aufzuheben. Der Punkt

MATTHIAS GERISCH, Senior Consultant movisco AG, studierte Verwaltungswissenschaft an der Universität Konstanz. Als Quereinsteiger in die Informatik ist er seit 2000 in der SAP-Beratung tätig. Bei der movisco AG ist er verantwortlich für die Themen Digitale Transformation, Agile Entwicklung und In-Memory-Datenbanken.

THOMAS ARNSBERG, Vorstand movisco AG, Dipl.-Kfm. und Dipl.-Wirt.-Inf., absolvierte sein Studium an der Universität Duisburg-Essen. 2012 gründete er mit einem Partner die movisco GmbH; seit 2017 ist er als Vorstand für die movisco AG tätig und betreut Innovations- und regulatorische Projekte.

E-Mail: info@movisco.com

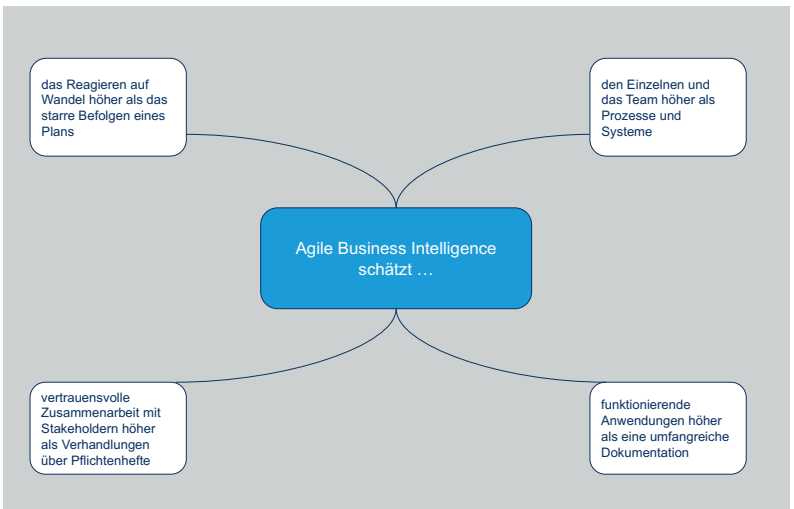

Abb. 1: Agile Werte (in Anlehnung an [Col12])

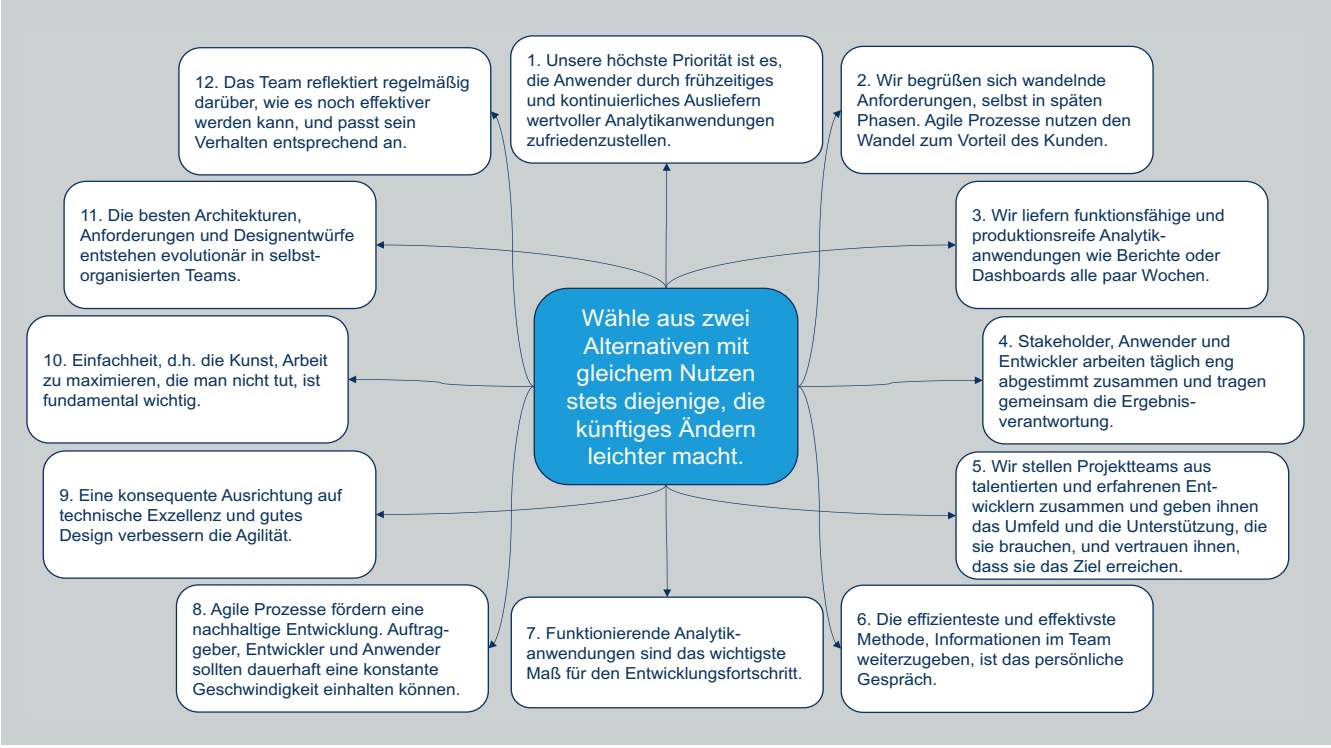

Abb. 2: Agile Prinzipien (in Anlehnung an [Col12])

ist, dass das Entwicklerteam eine echte Validierung der getroffenen Designentscheidungen erst erhält, wenn sich diese in der Anwenderpraxis bewährt haben [Col12]. Da jedoch auch diese einem kontinuierlichen Wandel unterliegt, kann nur ein evolutionärer Designansatz zu einer anwenderfreundlichen Applikation führen.

Aber wie kann man sicherstellen, dass das entstehende Design erstklassig ist und nicht anarchisch? Die Antwort liegt in der disziplinierten Anwendung agiler Praktiken.

Just-in-time-Modellierung

Auch bei einem evolutionären Architekturansatz gibt es Designaktivitäten, die zwangsläufig als Vorarbeit in der Anfangsphase erfolgen müssen. Jedoch beschränkt sich das Team in dieser Iteration 0 auf die Skizzierung einer noch wenig detaillierten Architekturvision. Diese weist nur so viel an Detaillierungsgrad auf, dass die Entwickler eine Grundlage haben, um mit der ersten Umsetzungsiteration zu beginnen. Der technische Detailentwurf erfolgt dann in „Model Storming"-Manier in den einzelnen Umsetzungsiterationen. So werden Designprobleme immer erst dann (just in time) behandelt, wenn dies aktuell für den weiteren Entwicklungsfortschritt unumgänglich ist. Auf diese Weise profitiert der finale Designentwurf von dem stetig gewachsenen Domänenwissen des Teams, ebenso wie von Designvalidierungen aus vorherigen Iterationen. Als Detaillierungsgrad wird im iterativen Designprozess jeweils ein „gerade so ausreichend" angestrebt, kein „umfassend und genau".

Jede Entwurfsaktivität innerhalb einer Iteration hat die konkreten Anwenderfunktionen zum Gegenstand, an denen gerade gearbeitet wird. Die vorausschauende Vorbereitung künftiger Funktionen ist nur ein sekundär verfolgtes Ziel. Gerade diese Fokussierung verlangt vom Entwicklungsteam ein großes Maß an Disziplin.

Das Entwicklungsteam setzt Funktionen mit hohem Wert für die Anwender frühzeitig um, ebenso wie Funktionen, die mit einem hohen Umsetzungsrisiko behaftet sind [Col12]. Somit wird sich das Design der Applikation ausgehend von diesem Kern kritischer Funktionen schrittweise entwickeln. Damit profitiert der Designentwurf von frühzeitigen und kontinuierlichen Modellvalidierungen gerade dieser kritischen Funktionen im Projekt.

Verschwendung vermeiden

Eine weitere zentrale Leitlinie für Designentscheidungen besagt, dass das Entwicklungsteam nur umsetzt, was für die geforderten Anwenderfunktionen auch *notwendig* ist. Diese Leitlinie ist einigen über die Jahre liebgewonnenen Gewohnheiten vieler SAP-Business-Warehouse-Entwickler allerdings diametral entgegengesetzt: Eine weit verbreitete Praxis ist es, gebrauchsfertigen SAP Business Content (wie beispielsweise den Geschäftspartner mit 100+ Attributen) zu aktivieren und zu verwenden, unabhängig davon, dass häufig nur ein kleiner Teil dieser Attribute tatsächlich zu den angeforderten Anwenderfunktionen beiträgt.

Das üblicherweise an dieser Stelle vorgetragene Argument lautet, schließlich baue man ja ein unternehmensweites Data Warehouse und müsse daher auch für künftige Anforderungen gewappnet sein. Historisch betrachtet war diese Haltung vollkommen vernünftig. Das Hinzufügen neuer Dimensionen und Attribute in reifen BW-Systemen konnte ein sehr aufwendiger Prozess sein. Mit SAP BW/4HANA sollte diese Gewohnheit nun jedoch

abgelegt werden. Wir implementieren im Core Data Warehouse nichts, was nicht notwendig ist, um die Arbeit an den aktuell angeforderten Benutzerfunktionen voranzubringen. Eine Abweichung von diesem Grundsatz mag im Einzelfall harmlos aussehen, kumulativ jedoch verschlingen derartige Abweichungen teure Ressourcen, die besser eingesetzt wären für den Bau und die Weiterentwicklung von Funktionen, die konkreten Anwendernutzen aufweisen. Bauen auf Vorrat ist schlechte Praxis, zehrt an Ressourcen und führt auf direktem Wege in die Komplexitätsfalle.

Änderungskosten minimieren

Die Agilität von Data-Warehouse-Systemen zu maximieren ist gleichbedeutend mit der Minimierung (latenter) Änderungskosten. SAP BW/4HANA stellt Entwicklern eine Reihe neuer Werkzeuge zur Verfügung, um die Agilität des DW-Systems zu erhöhen, wie zum Beispiel:

- Feldbasierte Modellierung
- Schichten-Virtualisierung
- Data-Vault-Modellierung

In dem Maße, wie das Business-Warehouse-Entwicklungsteam dadurch flexibler und schneller auf Änderungen reagieren kann, werden neue und modifizierte Anforderungen seitens der Fachanwender vom Entwicklungsteam nicht mehr als abzuwehrende Zumutung verstanden, sondern begrüßt und aktiv eingefordert.

Feldbasierte Modellierung

Traditionell stehen in SAP Business Warehouse Infoobjekte am Beginn von allem. Sie bilden die Bausteine für Führungskennzahlen, „Conformed Dimensions“, Infoprovider und alles andere. Die Mächtigkeit und Funktionsfülle des Infoobjekts war gleichzeitig auch seine Achillesferse: Gerade weil das Infoobjekt so zentral für das Business-Warehouse-Modell ist, war es auch eine Hauptquelle von „Systemsklerose“. Die feldbasierte Modellierung in BW/4HANA erlaubt es den Entwicklern nun, Infoobjekte zum spätmöglichsten Zeitpunkt in die Schichtenarchitektur einzuführen. So können wir dem Modell graduell Infoobjektassoziationen hinzufügen in dem Maße, wie sich die Annahmen über die zugrunde liegende Semantik bestätigen und festigen. Da wir vor dem Aufbau von Datenstrukturen und Queries nicht zunächst Hunderte von Infoobjekten anlegen müssen, ist es zudem nun möglich, Kunden im Projekt frühzeitig einen ersten Blick auf die Daten zu ermöglichen, Berichtsprototypen zu präsentieren und Anwender-Feedbacks für die Umsetzungsiterationen zu erhalten [Sau17].

Schichten-Virtualisierung

Die Schichten der skalierbaren Layer-Architektur für SAP BW/4HANA (LSA++) – Open Operational Data Store, EDW Core und Architected Data Marts – werden nicht mehr strikt als Persistenzebenen gedacht, sondern als offen, durchlässig und flexibel. In einem vollständig vom Quellsystem gemanagten Datenszenario könnte das Business Warehouse so im Extremfall gar keinen eigenen „Daten-Fußabdruck“ mehr aufweisen und alle Daten in Echtzeit aus dem operationalen Quellsystem beziehen. Aber auch in konventionellen Szenarien mit eigener Datenhaltung in SAP Business Warehouse reduzieren sich die notwendigen Persistenzebenen mit SAP BW/4HANA erheblich.

Dies minimiert Änderungskosten, weil Anpassungen an virtuellen Strukturen wie HANA Calculation Views, an Composite-Providern oder Queries immer weniger aufwendig sind als Änderungen an physischen Datenbankstrukturen. Damit zeichnet sich eine neue Best Practice für das Data-Warehouse-Design ab: „Erzeuge keine neue Datenpersistenz, es sei denn, es ist unvermeidbar.“

Data-Vault-Modellierung

Die Data-Vault-Modellierung ist ein Modellierungsansatz, der Schlüsselbegriffe zu Entitäten (Hubs), Assoziationen zwischen Entitäten (Links) sowie

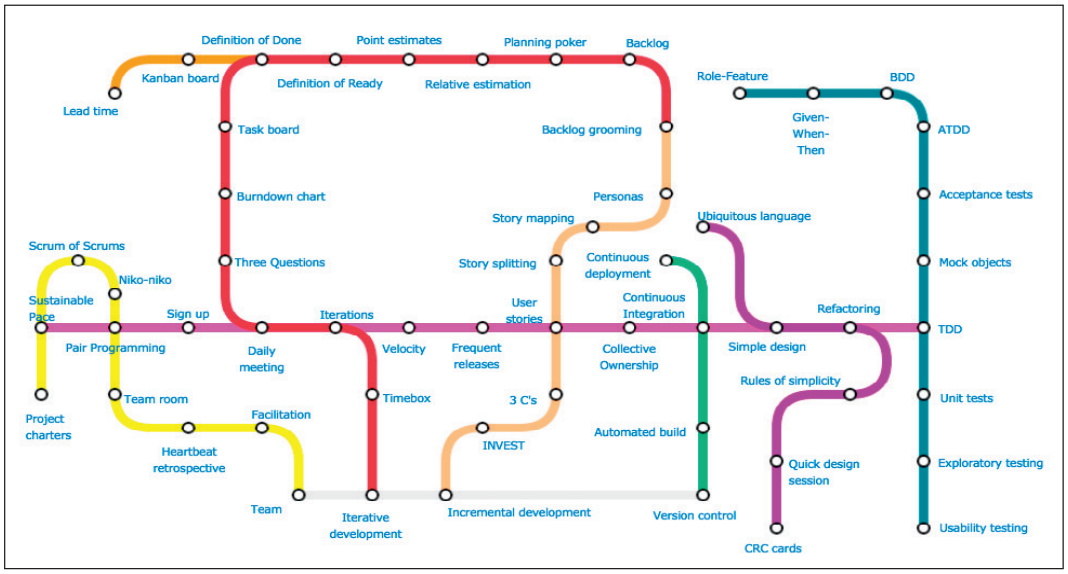

Abb. 3: Agile Praktiken und Methoden als U-Bahn-Netzplan (Agile Alliance, Screenshot [Agi18])

beschreibende Attribute von Entitäten (Satelliten) im Datenbankdesign physisch separiert. Die große Nützlichkeit des Ansatzes für das agile Data Warehouse liegt unter anderem darin begründet, dass sich mit seiner Hilfe einerseits Datenredundanzen, beispielsweise bei der Modellierung zeitabhängiger Dimensionen, systematisch eliminieren lassen und er andererseits eine enorme Offenheit für strukturelle Änderungen aufweist, die im Data Vault grundsätzlich durch Anlegen neuer Datenbank-Tabellen umgesetzt werden. Ein einmal implementierter Content mitsamt seinen zugehörigen ETL-Prozessen ist damit hochgradig robust gegenüber Änderungen. Aus agiler Sicht ist dies ein großer Vorteil im Hinblick auf die Skalierbarkeit und Flexibilität des Datenmodells und hilft, den Testaufwand zu reduzieren.

Im BW/4HANA-Umfeld legt der Ansatz unter anderem eine vermehrte Umstellung auf virtualisierte Infoobjekte nahe, die ihre Daten zum Beispiel aus mehreren (Satelliten-)Objekten des Core EDW Layer zusammenziehen oder auch im Direktdurchgriff zurück in das operative Quellsystem mit Hilfe von Smart Data Access [SAP15].

Refaktorierung

„Code Refactoring" ist eine agile Methode zur Tilgung technischer Schulden. Als „technische Schuld" bezeichnet man die Summe aller suboptimalen Design- und Umsetzungsentscheidungen während der Entwicklungsiterationen. In dem Maße, wie sich das Team mit jeder Iteration mehr Domänenwissen aneignet, können sich frühere Entscheidungen beispielsweise als technische Hypothek herausstellen. Technische Schulden machen es latent teurer, Artefakte zu ändern – daher die „Schulden"-Metapher. Unter „Refaktorierung" wird im Softwaredesign der Prozess der Verbesserung der internen Struktur von Programmkomponenten verstanden, ohne das nach außen wahrnehmbare Verhalten der Softwarekomponenten zu verändern. Im Datenbank-Design hat Refaktorierung die Veränderung von Struktur- (Views, Tabellen) und Funktionselementen (Prozeduren, Methoden) in Datenbank-Schemata unter Beibehaltung der Semantik und des externen Verhaltens zum Gegenstand.

Alle strukturellen Änderungen, die an der analytischen Datenbank vorgenommen werden sollen, sind konzeptionell als Refaktorierungen zu sehen, unabhängig davon, ob wir neue Datenfelder oder -tabellen einführen oder existierende modifizieren bzw. löschen. Dies ist *präskriptiv* gemeint: Gutes Design muss es ermöglichen, Datenstrukturen zu modifizieren, ohne dass dies eine unmittelbare Auswirkung auf das externe Verhalten der Applikation hat.

Dieses Prinzip unterstreicht die Bedeutung von Designpraktiken wie Data-Vault-Modellierung oder Schichten-Virtualisierung in BW/4HANA im Hinblick auf die Refaktorierungseffizienz. SAP Business Warehouse hat traditionell Refaktorierungsbemühungen nicht wirksam unterstützt (Ausnahme: Infocube-Remodellierung). SAP HANA hingegen beinhaltet eine Reihe nützlicher Funktionen, die eine Refaktorierung von Information Views erleichtern, wie zum Beispiel Verwendungsnachweis, „Column Lineage", Ersetzung von View-Knoten oder Aktualisierung von Referenzen [Jen15].

Schlussfolgerung

Unzureichende Ausrichtung an Anwenderbedürfnissen, lange und unflexible Entwicklungszyklen sowie hohe Entwicklungskosten zählen zu den Kritikpunkten, die Business-Intelligence-Abteilungen traditionell entgegengebracht wurden. Agile Methodik und In-Memory-Technologie bieten heute ein mächtiges Instrumentarium, diesen Problemen entgegenzuwirken.

Dabei sollte man sich vor Augen halten, dass in jeder Organisation die BI-Anwendungsentwicklung stets „agil" sein wird: Je weniger dies im Einzelfall für die zentrale, IT-gemanagte Business Intelligence zutrifft, umso mehr wird es für die „Schatten-BI" zutreffend sein.

Literatur

[Agi18] Agile Alliance: Subway Map to Agile Practices. 2018, https://www.agilealliance.org/agile101/subway-map-to-agile-practices/, abgerufen am 16.7.2018

[Bec01] Beck, K. et al: Manifesto for Agile Software Development. Twelve Principles of Agile Software. Snowbird 2001, http://agilemanifesto.org/, abgerufen am 16.7.2018

[Col12] Collier, K.: Agile Analytics. A value-driven Approach to Business Intelligence and Data Warehousing. Boston 2012

[Jen15] Jensen, M.: Hana Modeler: Impact Analysis and Refactoring of Calculation Views. Oktober 2015, https://blogs.sap.com/2015/10/23/hana-modeler-impact-analysis-and-refactoring-of-calculation-views/, abgerufen am 16.7.2018

[Mor15] Moria, I.: Using SAP HANA as a Logical Data Warehouse Platform. März 2015, https://blogs.sap.com/2015/03/31/using-sap-hana-as-a-logical-data-warehouse-platform/, abgerufen am 16.7.2018

[SAP15] SAP: Influence of SAP HANA on Data Warehouse Architecture. 2015

[Sau17] Sauer, K.-P.: How SAP BW/4HANA Revolutionizes the BW Modeling You Know. Juni 2017, https://blogs.saphana.com/2017/06/23/how-sap-bw4hana-revolutionizes-the-bw-modeling-you-know/, abgerufen am 16.7.2018

[TZK16] Trahasch, S. / Zimmer, M. / Krawatzeck, R.: Agile Business Intelligence als Beispiel für ein domänenspezifisch angepasstes Vorgehensmodell. In: Engstler, M. (Hrsg.): Projektmanagement und Vorgehensmodelle. Bonn 2016