Agile Entwicklung und zentrales Data Warehouse

Widerspruch oder Ergänzung?

„Fachabteilung und Zentral-IT“ - oft bekannt als Antipoden im Unternehmen oder einfach nur zwei nicht versöhnlich Positionen unter einem Unternehmensdach?
Von außen betrachtet treibt es die einen zu immer neuen Ideen und Projekten, sie können neue Informationen und Fachanwendungen nicht schnell genug bekommen, während die anderen als „zentrale“ Dienstleister kaum hinterherkommen und (manchmal unfreiwillig) bremsen.
Bezogen auf zentrale Data Warehouse Systeme scheinen diese Gegensätze besonders krass. Anders als bei einzelnen IT-Fachanwendungen (OLTP-Anwendungen) sind zentrale Warehouse-Systeme und deren System-Verantwortliche gleich für mehrere Fachanwendungen und somit unterschiedliche Fachanwendergruppen zuständig. Die Distanz zum Fachanwender ist automatisch noch größer.
Agile Software-Entwicklungsmethoden haben auch in BI / DWH-Projekten Einzug gehalten, um zu einer schnelleren und flexibleren Weiterentwicklung der Systeme zu kommen. Doch diese Methoden alleine werden nicht ausreichen, um Warehouse-Systeme auf lange Sicht effizient und flexibel zu halten. Die agile Denkweise in den Projekten muss durch koordinierende Funktionen und pfiffige Architekturen eines nach wie vor notwendigen zentralen Data Warehouse ergänzt werden.

Projekt-Letargie?

Warehouse – Systeme in größeren Unternehmen haben den Ruf mit zu den komplexesten IT-Systemen zu gehören. Ein Grund dafür ist naturgegeben die Sachgebiets- und Bereichs-übergreifende Spannweite der Systeme. Die zu leistende Integrationsaufgabe ist nicht zu unterschätzen.
Ein weiterer Grund ist jedoch auch der oft planlose Ausbau der Systeme in der Vergangenheit, mit dem man immer neue Bereiche an das Warehouse angeschlossen hat. Das so entstandene Chaos in manchen Systemen (*„Keiner weiß, was wirklich wo und in welcher Form zu finden ist“*) hat den Ruf komplex zu sein noch verstärkt. Das Ergebnis: Bei einem Erneuerungsprojekt des Data Warehouse wuchs der Erwartungsdruck und die Teams mussten besonders gründlich vorgehen und im Vorfeld möglichst das komplette künftige System fehlerfrei spezifizieren.
Noch zu Beginn der 2000er Jahre nutzte man für solche gewaltigen Systementwicklungsprojekte Engineering-Methoden aus Zeiten der zentralen Mainframe IT-Systeme. Noch im Jahr 2012 konnte man aktenordnerweise Systemspezifikationen von mehreren 1000 Seiten in den Projekten finden, bevor auch nur ein Stück Lösung für die Endanwender entwickelt war.
Oft fühlten sich die zentralen IT-Abteilungen in der Pflicht für derart große Projekte, zumal Data Warehouse Systeme auch an zentraler Stelle, durch eine Zentral-IT, für das Unternehmen betrieben und weiter entwickelt werden sollten. Damit entstand neben der Größe und Komplexität ein weiterer Schwachpunkt in dem Umgang mit Data Warehouse – Systemen.

- Neue Projekte waren intensiv zu prüfen, neue Funktionalität war genau zu spezifizieren, Pflichtenhefte waren z. T. umfangreicher und detaillierter als der anschließend zu programmierende Code.
- Damit man bloß nichts Falsches im Data Warehouse tut (*„ist ja komplex“*), wurde der Informationsbedarf der Fachanwender in langen Interviewphasen genau analysiert, bevor auch nur ein Handschlag am Zielsystem geschah. Oft überlebte der ursprüngliche Bedarf das neue System nicht (moving target issue).
- Verfahrensregeln und Standards wurden in Konzepthandbücher geschrieben und dann nach mehreren Mitarbeiterwechseln in bürokratischer Manier leider nicht mehr hinterfragt und standen aktuellen Anforderungen und Wünschen immer wieder im Weg.
- Man legte Service Level Agreements mit den Fachabteilungen fest, die bald als Erklärung für alle Zeitverzögerungen herhalten mussten oder gern zitiert wurden, um Neues und Änderungen zu verhindern.
- War eine Spezifikation aufwendig erstellt, stand die Auswahl von Entwicklungs-Tools und Technologie auf der Agenda. Mitarbeiter – Teams, gestärkt durch externe Berater, verglichen über Monate (manchmal Quartale) hinweg, viele Anbieter. Ziel: Es sollte die Technologie gefunden werden, die man schließlich strategisch – meint mehrere Jahre – einsetzen will. *Und dann sollte man sich auch genügend Zeit für die Evaluierung nehmen.* Derart intensiv geprüft fällt es erst recht schwer, sich ad hoc auch mal für einen anderen Weg zu entscheiden. Manche Tool- / Technologie-Entscheidung stand echter Innovation schon mal im Weg.
- Kommt noch das „Experten-Problem“ hinzu: Löst man Aufgaben zentral, hofft man auf den Multiplikatoren-Effekt nachdem sich einzelne Mitarbeiter sich für bestimmte Aufgaben (und Technologien und Tools) zum Experten entwickelt haben. Experten sind einerseits sinnvoll, führen aber andererseits in Teams zu vielen Reibungsverlusten, weil ganzheitliches Denken (das Mitberücksichtigen fremder Aufgabenstellungen) fehlt („Insel-Wissen“).

Flexibilität und Schnelligkeit bei der Umsetzung neuer Aufgaben und Anforderungen sieht anders aus.

In einigen Unternehmen eskaliert die Situation, zumal heute Fachabteilungen umfangreichere und diversifiziertere Informationsbedarfe entwickeln. In immer kürzeren Zeitabständen werden neue Analysen gefertigt und neues Datenmaterial mit zusätzlichen Aspekten muss beschafft werden.

Hinzu kommen neue Randbedingungen innerhalb und außerhalb der Unternehmen durch neue technologische Möglichkeiten sowie ökonomische und gesellschaftliche Einflüsse.

- Änderungen von Datenformaten und Datenhandhabung

- Neuartige Endgeräte
- Geänderte Nutzungszeiten
- Ausweitung von geografischen Aktionsfeldern
- Neue Zielgruppe für Produkte und Dienstleistungen
- Weichere Datenqualitätsanforderungen
- ..

Agile BI/DWH-Entwicklung als „Heil-bringende“ Lösung

Agile Software Methoden (Scrum) sind angetreten, um hier Abhilfe zu schaffen. Sie adressieren vor allem die enge Einbindung der Fachanwender, für die letztlich eine Lösung zu erstellen ist und das permanente Überprüfen der Projektziele gegenüber sich verändernden Interessen in der Fachanwenderschaft.
Vertreter aus den Fachabteilungen werden als sog. *Product-Owner* in die Projektarbeit einbezogen. Sie übernehmen damit permanent Mitverantwortung für Projektverlauf und Dauer. Weil langfristige Projektplanung und detaillierte Spezifikationen den sich immer schneller ändernden betrieblichen Rahmenbedingungen kaum hinterherkommen, reduziert man die langfristige Planung der Funktionalität einer Business Intelligence / Warehouse-Lösung auf ein Minimum (sog. Epics). Als Funktionsbeschreibung ausformuliert werden lediglich überschaubare und abgrenzbare Funktionseinheiten (User Storys). Der *Product Owner* priorisiert diese *User Storys* und übernimmt sie in das *Product-Backlog* (Liste mit Beschreibungen von Zielfunktionen) mit dem Ziel, die beschriebene Funktionalität in den *Sprints* durch das Entwicklerteam umsetzen zu lassen. Sprints sind kurze, überschaubare Entwicklungszyklen, in denen funktionsfähige Komponenten entstehen. Regelmäßige Reviews der Sprints und der Entwicklungsergebnisse durch Endanwender als auch durch das Entwicklungsteam ermöglichen frühzeitige Korrekturen der Projektarbeit ohne, dass durch langfristige Planung und Prozesse bereits viel Arbeit investiert wurde.
Der kurze Einblick soll an dieser Stelle genügen. Über agile Software-Entwicklung (auch in BI / DWH-Projekten) ist in den vergangenen Jahren eine Fülle von Büchern geschrieben worden, über die man sich weiter informieren kann. (S. Scrum Guide als offizielle Literatur).

Agile Entwicklung im Data Warehouse?

Tatsächlich findet man mittlerweile ohne besonderen Suchaufwand offenbar erfolgreiche Beispiele für die Umsetzung agiler Software-Entwicklung auch in Data Warehouse-Projekten.
Doch es gibt auch Fragen, die bei aller Euphorie zu beantworten sind. Was in Einzelprojekten und abgrenzbaren Sachgebieten gut funktioniert, stößt bei unternehmensweit angelegten Systemen, wie bei einem Enterprise Data Warehouse an Grenzen und bedarf zumindest ergänzender Überlegungen und Vorkehrungen.

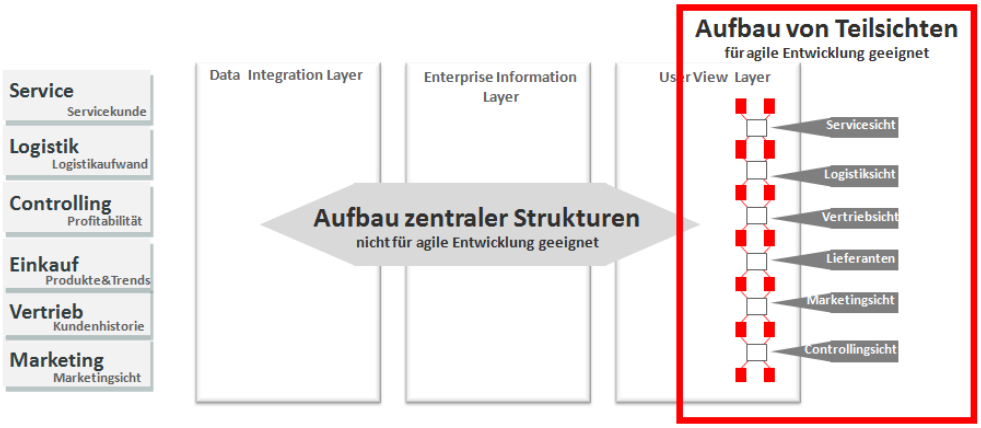
Gibt es globale Aufgabenstellungen in einem Data Warehouse, die über lange Phasen hinweg zu erfüllen sind, um ein koordiniertes, überschaubares und dennoch flexibles Warehouse-System am Leben zu erhalten?
Einerseits lassen sich globale Aufgaben in einem zentralen Data Warehouse schwer in einzelnen „Sprints“ abarbeiten. Andererseits müssen auch die globalen Warehouse Systeme flexibel genug sein, um agile Software – Entwicklung bei der Erstellung von Teillösungen für Endanwender zu unterstützen.
Während über agile Projektmethoden auch im BI-Umfeld seit Jahren berichtet wird, fehlt es immer noch an Ideen zur Flexibilität und Schnelligkeit in zentralen Warehouse-Systemen.

Ein Blick in das zentrale Data Warehouse

Kurz zusammengefasst erfüllt auch heute noch das klassische Data Warehouse mindestens 4 Kernfunktionen:

1. Es integriert i.d.R. unternehmensweit unterschiedliche Sachgebiete. Es stellt also eine einheitliche Sicht über nahezu alle relevanten geschäftlichen Aktivitäten da.
2. Es überführt betriebliche Kennzahlen in einen für jeden verstehbaren transparenten Kontext. Es erklärt die Zusammenhänge und ergänzt sie z. B. durch Referenzdaten.
3. Es zeigt den betrieblichen Werdegang in einer historischen Sicht.
4. Es neutralisiert die betrieblichen Informationen von ihrem ursprünglichen Kontext der operativen Vorsysteme und ermöglich damit z. B. Simulationen, mehrdimensionale Betrachtungen oder Datenpräparation für weitergehende Analysen.

Alle diese Punkte haben etwas Gemeinsames: Betriebliche Daten verlassen ihren Entstehungsort und werden in einer modifizierten Form an anderer Stelle, meist mit neuen Beziehungen und Beschreibungen neu abgelegt.
Für diesen Zweck zerlegt man die Daten der Vorsysteme in ihre Einzelinformationen.
Die so entstandenen granularen Informationsbausteine lassen sich jetzt leicht in beliebiger Kombination für beliebige Auswertezwecke neu zusammensetzen. Über dieses von Inmon in den 90er Jahren vorgeschlagene Verfahren lassen sich z. B. gleiche Daten aus verschiedenen Vorsystemen fachlich und semantisch in einer Kern-Warehouse-Schicht zusammenführen (Enterprise Information Layer).



Teilsichten und zentrale Aufgabenstellungen müssen koordiniert werden.

Warehouse-Systeme sind also, im Gegensatz zu einzelnen sachgebietsbezogenen Analyseanwendungen, Lösungen, die Daten nahezu unternehmensweit aus vielen Fachanwendungen einsammeln. Dieser Vorgang ist bei großen Unternehmen heute besonders wichtig: Einzelne Geschäftsfelder werden permanent beobachtet, auf Rentabilität hin bewertet und unternehmensweit der optimale Einsatz von Kapital und Ressourcen geprüft. Wozu ist ein Unternehmen insgesamt fähig? Wo lohnen sich Investitionen? In welchen Segmenten kann man schneller wachsen? Die Antworten auf solche Fragen liefert nur ein sachgebiets-übergreifendes Data Warehouse – System.

Verlässt man ein Sachgebiet und wagt sich an sachgebietsübergreifende Zusammenstellungen von Daten, muss man harmonisieren. Man muss gleiche Dinge in verschiedenen Sachgebieten als solche erkennen bzw. berücksichtigen, wie sich der Umgang mit gleichen Geschäftsobjekten in verschiedenen Sachgebieten unterschiedliche gestaltet.

Das Geschäftsobjekt *„Kunde“* wird in einer „Beschwerdeabteilung“ (Service-Abteilung) anders wahrgenommen, als in einer Vertriebsabteilung. Die Sichten aus beiden Abteilungen zusammen ergeben ein vollständigeres Kunden-Bild. In Zeiten von Big Data steuern zusätzlich auch außerhalb des Unternehmens liegende Vorgänge Sichten auf das Kunden-Geschäftsobjekt. Auch diese Sichten sind zu integrieren, um das Kundenbild zu vervollständigen.

Aufbau unternehmensweiter Sichten und Agilität - ein Widerspruch?

In manchen Artikeln und Vorträgen wird die Mehrschichtigkeit zentraler Warehouse-Systeme als unflexibel und zu starr für die heutigen Bedürfnisse abgetan. Der Aufbau der Kern-Warehouse-Schicht sei zu aufwendig und änderungsunfreundlich.

Zunächst: Bei entsprechender Vorgehensweise entsteht in der Kern-Warehouse-Schicht ein sachgebietsübergreifendes Datenmodell, das mit der Zeit alle wichtigen Geschäftsobjekte des Unternehmens umfasst und für Analysen zugänglich macht. Die

Tatsache, dass diese Daten meist in einer nahezu 3.Normalform vorliegen, ist dabei nur praktisch. Diese Modellform schafft erst die Voraussetzung für das Zusammenführen von Daten aus unterschiedlichen Vorsystemen, und man kann sehr schnell neue Auswertemodelle (Kennzahlen mit ihren beschreibenden Dimensionen) daraus ableiten.

Das Problem, dass man auch in einem normalisierten Datenmodell nicht ohne Weiteres doppelte oder unscharf abgegrenzte Informationen ausfindig machen kann, hängt mit dem altbekannten *Synonymen-/Homonymen-Aspekt* zusammen. Dieses lässt sich bekanntlich nicht (!) durch das Normalisierungsverfahren lösen. Der Umstand der fehlenden Eindeutigkeit von Information im Warehouse ist also *kein Datenmodell-Problem*, sondern tritt bei jeder Datenmodellform auf (s. weiter unten).

Ja, das Kerndatenmodell wächst, im Idealfall sogar bis hin zum Unternehmensdatenmodell. Hier entstehen weitere Irrtümer der aktuellen Warehouse-Diskussion. Man sollte an dieser Stelle *nicht* die Unternehmensdatenmodell-Diskussion führen. Das ist zu Beginn der 90er Jahre gemacht worden und die ist abgeschlossen! Auch ohne agile Methoden anzuwenden, wird heute niemand versuchen wollen, bei dem Neudesign eines Data Warehouse zunächst die Kern-Schicht komplett – sprich unternehmensweit - zu entwerfen, um dann im Nachgang sich mit den Data Marts bzw. den eigentlichen Endbenutzerfragen zu beschäftigen. Bereits seit über 15 Jahren werden gerade aufgrund der schlechten Erfahrungen mit monolithischen Riesenprojekten der 90er Jahre, besonders die unternehmensweiten Warehouse-Systeme *inkrementell* erstellt. Der erste Schritt bei dem Neu–Design eines Data Warehouse ist immer zuerst eine Analyse einer konkreten Anforderung aus einem Sachgebiet gewesen, das als Erstes mit einem Analysesystem versorgt werden musste. Bei konsequenter Anwendung entsprechender Methoden war das zentrale Kern-Warehouse-Datenmodell danach eine Art Abfallprodukt, das sukzessive nach jedem neuen Inkrement um die Geschäftsobjekte der jeweils neu entwickelten Analysemodelle erweitert wurden.

So gesehen steckt bereits in der oft schon praktizierten inkrementellen Vorgehensweise ein Stück Agilität. Das *Scrum* hat viele der Erfahrungen aus dieser Vorgehensweise standardisiert und eine Art Entwicklungs-Framework geschaffen. Hierin besteht der eigentliche Mehrwert von Scrum.

Damit sind wir aber bei einem Kernproblem angelangt, das bei jeder Data Warehouse-Weiterentwicklungsaktivität zu lösen ist:

- Wie kann Weiterentwicklung unter Zuhilfenahme bereits bestehender Informationen in dem Kern-Warehouse-Modell stattfinden?
- Wie kann das Informationsgebilde eines Analysesystems so weiterentwickelt werden, dass Eindeutigkeit über alle Sachgebiete hinweg garantiert ist?
- Wie kann ein bereits bestehendes Kern-Warehouse-Modell die Arbeit von Projekten beschleunigen, weil bestehende Geschäftsobjekte nicht mehr neu

entwickelt werden müssen? (Das Problem besteh darin, zu erkennen, dass es sie schon gibt).

Diese Fragen müssen auch Entwicklerteams beantworten, die agil unterwegs sind.

Gebraucht werden einheitliche Instrumentarien

Wir brauchen also zusammenhängende und sachgebietsübergreifende Warehouse – Systeme

- mit einheitlichen Sichten auf Geschäftsobjekte,
- mit standardisierten Verfahren bei der Herleitung und Berechnung von Datenobjekten und Kennzahlen,
- mit abgestimmten Datenqualitätsstandards,
- mit einer gleichen Art Geschäftsobjekte zu dokumentieren und
- mit einer unternehmensweit einheitlichen Sprache / Begrifflichkeit.

Walter Brenner stellte 1985 in seiner Dissertation („Entwurf betrieblicher Datenelemente“, Springer Verlag, 1988) Lösungsmöglichkeiten für das fachliche Sortieren und Pflegen von Informationen in einem Unternehmen dar. Auch nach fast 30 Jahren sind diese Hilfsmittel noch aktuell, denn das gelöste Problem ist nach wie vor vorhanden: Unternehmensweite Warehouse-Systeme leiden unter fehlender Stimmigkeit und Eindeutigkeit der enthaltenen Informationen.

Wer erkennt ob „Erlös“ dasselbe ist wie „Gewinn“, wann ist ein „Kunde“ ein „Kunde“. In einem Data Warehouse treffen Daten aus allen möglichen Geschäftsprozessen im Unternehmen zusammen und die Geschäftsobjekte sind oft nur aus dem Zusammenhang des konkreten Geschäftsprozesses heraus verstehbar. Ein Warehouse muss aber diese Geschäftsobjekte aus den jeweiligen Geschäftsprozessen herauslösen und sie mit Geschäftsobjekten anderer Geschäftsprozesse vereinigen, damit übergreifende Analysen möglich sind, und das, ohne dass ein Informationsverlust stattfindet.

Erschwert wird diese Situation, weil oft Daten einfach nur abgelegt werden, ohne sie nach inhaltlichen Kriterien mit den bereits in dem Warehouse bestehenden Datenbestand abzugleichen. Oft fehlt die Grundvoraussetzung für einen solchen Abgleich: eine nachvollziehbare Beschreibung der Inhalte der bestehenden Daten.

Dies scheint zu den Hauptproblemen vieler Auswertesysteme zu gehören. Manch ein Warehouse-Verantwortlicher hat sich schon oft gefragt, ob es nicht günstiger wäre, ein Warehouse komplett neu mit gültigen stimmigen Prinzipien zu bauen, anstatt ständig das bestehende Chaos weiterzuentwickeln.

Bei isolierten Data Marts, die sich nur um „ihr privates“ Sachgebiet bzw. ihre Fachabteilung kümmern, tritt das Problem nicht auf. Auch wenn man sich nur mit den sichtbaren Phänomenen aus der BI / DWH-Welt beschäftigt, den Dashboards und gedruckten Berichten, dem Variantenreichtum der Charts, Mobilität etc. verschwindet das Problem im Hintergrund.

Auch das hier beschriebene Phänomen ist ein Grund für die nach außen spürbare Schwerfälligkeit zentraler Warehouse-Systeme. Vielen DWH-Verantwortlichen ist die Herausforderung bewusst, und sie sind vorsichtig bei Weiterentwicklungsprojekten geworden. Deswegen sind sie jedoch keine noch keine „Bremser“.
Zurück zu Brenner 1985. Seine Lösung sieht Folgendes vor: Was wir heute Geschäftsobjekte nennen, sind aus Datenmodellsicht einfach nur Entitäten. Entitäten charakterisieren sich über ihre Attribute. Brenner-Terminologie aus früheren Jahren: Datenelemente. Das sind die kleinsten informatorischen Bausteine, die wir kennen. Wenn „Erlös“ und „Gewinn“ mit denselben Attributen beschrieben werden, handelt es sich mit einfacher Logik um das gleiche Geschäftsobjekt, auch wenn unterschiedliche Namen genutzt wurden.
Aber auch Datenelemente sind nicht eindeutig, d. h., das Synonymen-/Homonymenproblem steckt tiefer. Als Lösung werden mehrere Techniken vorgeschlagen. Zunächst geht man davon aus, dass alle Datenelemente auf einer Metadatenebene zu beschreiben sind. Weil das für ein durchschnittliches Unternehmen schon mal 10000 Elemente/Attribute werden können, bietet sich eine sog. Entwurfsdatenbank an, oder anderer Name: Metadaten-Repository. (Bevor man jetzt allerdings über die Einführung eines Repositories nachdenkt, und wegen des zu erwartenden Aufwands das Vorhaben beendet, reicht sicherlich eine einfache Datenbanktabelle für diesen Zweck.)
Datenelemente (Attribute) verfügen über 2 Merkmale:

- Sie haben einen Bezeichner (Name) und
- Sie verfügen über eine Semantik (Bedeutung), die man mit Sprachmitteln beschreibt.

Hat man es nicht mit einem Altsystem zu tun, wird man bei dem Entwurf eines Attributes Namenskonventionen oder sonstige Standards festlegen, an die sich jeder Entwickler zu halten hat, wenn Attribute für eine Entität zu definieren sind. Hier gibt es eine Reihe von Methoden wie *Wortstammanalysen* oder definierte *Unternehmensabkürzungsverzeichnisse*.

Aufwendiger wird die Beschreibung der Semantik der Elemente. Weil die Namen der Attribute selbst keinen Aufschluss auf deren korrekten Inhalt geben, sucht man nach Kriterien, die ein Attribut eindeutig beschreibbar machen. Man entwirft sog. Deskriptoren. Das sind 5 oder 10 Sichten, mit denen man eindeutig die Bedeutung eines Elements (Attributes) umschreibt. Die Deskriptoren selber müssen jedoch auch eindeutig sein, damit das funktioniert. Man legt dazu für jeden Deskriptor eine Reihe gültiger Werte fest, die man zur Beschreibung nutzt. Also keine freie Sprachwahl bei der Beschreibung der Inhalte sondern ein wohlüberlegtes und überprüfbares Vokabular. Damit ist dieser Entwurfsvorgang sogar durch ein kleines EDV-Programm automatisierbar und der zu Unrecht vermutete Aufwand bleibt gering.
Eine Sicht (Deskriptor) auf das Attribut „Wert“ der Entität „Erlös“ mag z. B. die Art der Entstehung sein: Das, was mit „Wert“ ausgedrückt wird, könnte durch einen

Rechenvorgang, durch einen Übertrag aus einem anderen Feld oder durch einen Definitionsvorgang entstehen. Für alle denkbaren Entstehungsarten wählt man ein festes Schlagwort, mit dem man die Entstehungsart immer wieder gleich beschreibt.
Der Begriff „Wert“ kann aber auch in einem völlig anderen Kontext genutzt werden: „Wert einer Produktlinie für das Gesamtportfolio“, „Wert einer Abteilung für das Gesamtunternehmen“ etc. Das würde man sicher nicht mit einem numerischen Inhalt beschreiben. Eine solche Bedeutung wird sicher mit anderen sprachlichen Mitteln ausgedrückt, als der zuvor beschriebene numerische Inhalt. Also können weitere Deskriptoren hilfreich sein, z. B. ist das Feld numerisch, alphanumerisch usw.

Ein solches Verfahren klingt aufwendig. Vermehrt Aufwand entsteht allerdings nur bei der Einführung und bei dem einmaligen Entwurf des Verfahrens. Wichtig ist, dass es gelebt wird, d. h. bei der Entwicklung von Geschäftsobjekten für ein Analysesystem die Stimmigkeit mit den bereits bestehenden Datenobjekten in dem zentralen Warehouse überprüft wird. Wenn das sogar maschinenunterstützt erfolgt, ist der Aufwand überschaubar. In vielen Fällen kann es die Entwicklungsarbeit sogar beschleunigen, weil vermeintlich neue Objekte bereits in dem zentralen Bestand existieren und deren Definitionen wiederverwendet werden können. Solche Objekte würde man nie in dem bestehenden Datenpool des Warehouse finden, wenn sie andere Namen trage.
Auch Projektteams, die mit agilen Methoden unterwegs sind, müssen diese Aufgabe über nehmen, oder sie zumindest bewusst an eine separate Gruppe delegieren.

Dokumentation im Warehouse muss trotz aller Agilität sein

Ist das vorher beschriebene Verfahren einmal eingeführt, so fällt es nicht schwer, diese Meta-Informationen ist einem separaten Verwaltungsbereich des Warehouse explizit zu erfassen. (Der Begriff Metadaten-Repository wird vermieden, weil dieser schon wieder sehr viel Aufwand suggeriert. Ein solches Repository wäre aber eine ideale Lösung für diesen Zweck.)
Dieses beinhaltet eine Liste aller in dem System erfassten Geschäftsobjekte. Da die Geschäftsobjekte über ihre Attribute mit ihren Deskriptoren bereits beschrieben sind, liegen im Prinzip schon alle Informationen für ein verlässliches Auskunftssystem vor. Sowohl für Endanwender bei Ihrer Suche nach sinnvollen Abfragen als auch für Entwickler bei der Suche nach bereits bestehenden Datenobjekten können solche Metadaten sehr hilfreich sein.
Für Praktiker in dem Oracle-Umfeld: Ein einfaches Tabellenwerk mit einer **APEX**-Oberfläche reicht völlig aus, ist schnell entwickelt und kostet nichts.

Domainen-Konzept

ETL-Prozesse sollten um eine Protokollfunktion erweitert werden, um die jeweiligen Lade- und Aktualitätsstände zu den einzelnen Geschäftsobjekten zu dokumentieren. Die Beschreibung der Entitäten kann für diesen Zweck um ein *Domainen-Konzept*

erweitert werden. Ein solches Domainen-Konzept beschreibt die vorkommenden Wertebereiche von Geschäftsobjekten. Bietet man in dem Data Warehouse z. B. Vertriebszahlen zu mehreren Vertriebsgebieten an, so kann es hilfreich sein, die regelmäßig geladenen Ladevolumen pro Vertriebsgebiet zu dokumentieren. An solchen Zahlen erkennt man die Nutzung und die Verteilung der Inhalte im Verlauf des Systemlebens. Man kann aktive von weniger aktiven Bereichen unterscheiden. Ohne ein solches Verfahren sieht man nur die nackten Warehouse-Tabellen in denen Sätze stehen.

Weitere Metadaten

Beispiele für weitere zentralen Metadaten-Beschreibungen sind:

- Glossare zur fachlichen Erklärung von Zusammenhängen und Begriffen
- Beschreibung der Inhalte externer Referenzdaten
- Beschreibung von Standardberichten
- Kennzahlen-Dokumentation
- Standardisiertes Abkürzungsverzeichnis

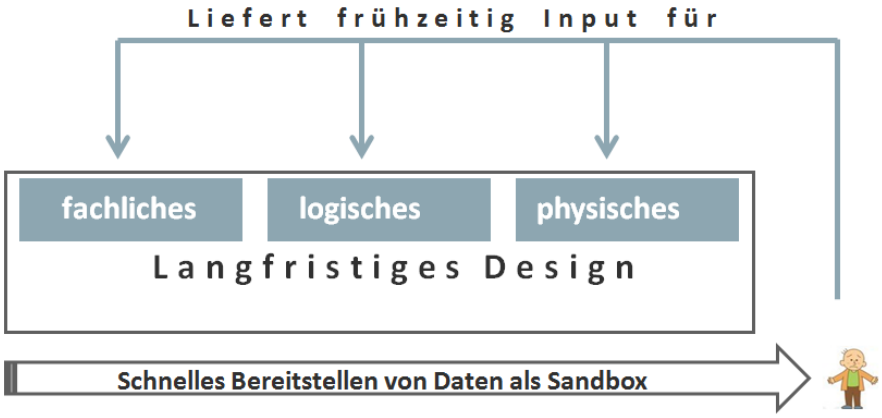
Auch die Pflege von solchen projekt- und sachgebietsübergreifenden Metadaten-Informtionen muss selbst bei agiler Vorgehensweise mitberücksichtigt werden. Auch dieses könnte man natürlich in ein separates Team für zentrale Aufgaben ausgliedern. Dann wäre eine Schnittstelle zu dem Agil-Team zu definieren.

Agilität à la Sandboxing mit Oracle

Ein Problem klassischer Software-Entwicklung ist der permanente Versuch, möglichst alle potenziellen Fehler eines neu zu entwickelnden Systems im Vorfeld zu erkennen und zu verhindern. Das kann dauern.
Agile Vorgehensweisen drehen die Reihenfolge um. Sie implementieren auf Verdacht quasi unreife Lösungen und sammeln bereits in dem Stadium der Entwicklung Praxiserfahrung mit einer potenziellen Lösung. Fehlentwicklungen lassen sich schneller korrigieren bzw. auch stoppen. Die klassische Methode entdeckte Schwachstellen oft erst zum Zeitpunkt des Go Live oder gar Monate später. Korrekturen waren teuer oder sogar kaum noch möglich.
Zu diesem Zweck stellt man entsprechende Datenmengen als *Sandbox* zur Verfügung und lässt z. B. Prototypen erstellen.
Das Verfahren hat mehrere Vorteile:

- Es wird zunächst keine ausformulierte und alles berücksichtigende Spezifikation benötigt.
- Es sind nur die offensichtlich relevanten Benutzergruppen mit ihrem Informationsbedarf einzubeziehen. Eine potenzielle Integrationsaufgabe entfällt.

- Datenqualitätsanforderungen (oft das Teuerste an Data Warehouse Projekten) sind zunächst noch ignorierbar. Probleme werden auf kurzem Dienstweg über schnelle Work arounds gelöst.
- Neue Datenmodelle müssen nur rudimentär vorhanden sein, eben soweit, dass der Prototyp funktioniert. Verfeinerungen bei den Attributen oder Ergänzungen mit weniger wichtigen Entitäten lassen sich in einer späteren Praxisphase nachholen, wenn wirklich klar ist, dass man sie benötigt.
- In dieser lockeren Form können Mitarbeiter noch flexibel Zuarbeit zu einem Projekt leisten, ohne zu 100 % einem Projekt zugeordnet zu sein.



Frühzeitiges Korrigieren des langfristig angedachten Systemdesigns durch frühzeitiges Erfahrungsfeld als Sandbox

Zentrale Warehouse-Lösungen und Sandboxing sind kein Widerspruch, sie ergänzen sich und mildern den Druck auf die Verantwortlichen der zentralen Systeme. Mit der Unterstützung des vorher beschriebenen Auskunftssystems fällt es besonders leicht in dem unternehmensweiten Datenpool, diejenigen Datenobjekte und sogar spezielle Domains innerhalb der Datenobjekt-Daten als Datenmenge zu separieren. Gebraucht wird dann natürlich eine stabile Datenbankfunktionalität, mit der das Herauslösen und separate Verwalten von Sandbox-Datenbeständen möglich ist. Die technischen Hilfsmittel der Oracle-Datenbank hierfür sind:

- **Partitioning**, zum schnellen Separieren, Hinzufügen und Löschen von oft zeitabhängigen Transaktionsdaten (Bewegungsdaten, Fakten-Informationen). Verteilen von Daten auf unterschiedliche physikalische Medien.
- **Multitenant-Database** zum einfachen Bereitstellen kompletter Datenbereiche mit eigenen Datenmodellen, in denen man losgelöst von dem eigentlichen Data Warehouse seine eigene Spielwiese erhält. Bestehende Datenbestände lassen sich leicht zu Sandboxes clonen und auch wieder schnell entfernen. Die technische Verwaltung der Datenbank erfolgt weiterhin durch die zentrale DB-Administration, ohne dass sich der Anwender, oder das Projektteam, das die Sandbox nutzt, darum kümmern muss.

- **Datapump** zum schnellen Extrahieren und Verteilen von gezielt ausgesuchten Datenbeständen. Möglich ist nicht nur die Auswahl und Bereitstellung einzelner Tabellen, sondern auch von Teilmengen innerhalb der Tabellen. Datapump-Jobs lassen sich automatisieren, und als Hintergrundprozesse sehr flexibel bei Bedarf abrufen.
- Die **Flashback Database** – Funktion der Datenbank kann ebenfalls hilfreich sein, um einen ursprünglichen Datenzustand wieder herzustellen, nachdem eine Simulation z. B. mit massiven Datenmanipulationen schief gelaufen ist. Flashback in Verbindung mit Multitenant ist ein unschlagbares Werkzeug, um wiederholte Versuchsreihen immer wieder auf null zurückzusetzen.
- Auch **Materialized Views** (MAV) sind hilfreiche Mittel, weil man damit einerseits eine gewünschte Datenmenge mit SQL genau beschreiben kann und auf der anderen Seite durch die MAV-Mechanik ein separater Datenbestand entsteht. Materialized Views lassen sich wieder mit Datapump verteilen und in eine Multitenant-Database überführen.

Das sind nur einige Beispiele für die technische Unterstützung von Sandboxing in der Oracle Datenbank. (Weitere Ideen liefert dieser Link: https://blogs.oracle.com/datawarehousing/entry/x_charging_for_sandboxes).

Teamarbeit ist Rollenarbeit

Die nächste kritische Stelle für agiles Handeln bei der Data Warehouse Entwicklung ist das Projektteam selber. Das agile Methodenkonzept schlägt für die Projektteams solche Teammitglieder vor, so sowohl Experten auf einem bestimmten Gebiet sind, aber auch zugleich Gesamtzusammenhänge verstehen und einschätzen können (T-Typen).
Warehouse-Projekte sind bzgl. der Teamzusammensetzung besonders herausfordernd, denn hier treffen neben der jeweiligen betriebswirtschaftlichen Fachkenntnis auch mehrere andere Disziplinen zusammen.
Einer der schädlichste Fehler, der fast in allen Projekten immer wieder gemacht wird, ist die Trennung zwischen *Technik* und *Business*. Indirekt verhindert man damit ganzheitliches, lösungsorientiertes Denken und Handeln aller Projektmitarbeiter.
Auch wenn das viele glauben, eine solche Trennung ist nicht naturgegeben, sondern oft durch dritte Faktoren festgeschrieben. Ein großer Nutzen von Scrum ist, dass sie helfen, solche Fehler zu verhindern.

Logische und technische Architekturen und die Aufgabenverteilung in Projektteams hängen bislang eng miteinander zusammen. Leider bedingt das eine manchmal das andere, d. h., das Abteilungs-Organigramm wird zur Blaupause einer Architektur und umgekehrt, die Aufgabenverteilung im Organigramm ist an einer in der Vergangenheit einmal festgelegten System-/Anwendungsarchitektur ausgerichtet.

Einer der beiden Pole dieses Duos wird Agilität schon verhindern, denn Veränderung bedeutet unbequemer Aufwand. Es ist offensichtlich, dass eventuell sinnvolle Architekturänderungen an Stellenbeschreibungen scheitern können, die sich nur über eine bestimmte Aufgabe in einer mal irgendwann in der Vergangenheit festgelegten Architektur definieren.
Ein anderes Beispiel sind Technik- bzw. Tool-Experten. Diese sehen i. d. R. nur den Horizont ihres Tools oder der Technologie, die sie gerade beherrschten. Sie stellen die technischen Möglichkeiten ihrer Instrumente in den Vordergrund. Die tatsächlich zu lösende Aufgabenstellung ist jedoch eine andere, nämlich Informationen passgenau und ökonomisch zur Verfügung zu stellen *mit allen (!) Mitteln, die passen,* also nicht nur mit dem persönlichen –technischen –Steckenpferd.

Zur Entwicklung von Warehouse-Systemen benötigt man heute stattdessen Teams, die über ein Rollenkonzept zusammengesetzt sind, wobei sich die Rollen an den zu leistenden Aufgaben in einem Data Warehouse orientieren. Rollen und Personen sind nicht identisch. Innerhalb einer Rolle werden eine oder mehrere Aufgaben beschrieben. Diese Rollen müssen nicht zwangsläufig 1:1 auf Personen zugeschnitten sein, sondern verteilen sich idealerweise auf mehrere Personen. Personen können mehrere Rollen einnehmen. Ziel dieser Verteilung der Rollen auf mehrere Personen ist das Schaffen eins Aufgaben-übergreifenden und damit ganzheitlichen Lösungsverständnisses. Wenn man z. B. die optimale Position für die Bildung einer Kennzahl ermitteln will, muss man auch alle potenziellen Stellen kennen, an denen die Bildung einer Kennzahl möglich ist.
Die Praxis in den Projekten sieht heute anders aus: Kennzahlen sind Sache der BI-Tools und damit automatisch Aufgabe des BI-Tool-Experten. Dass eine gleiche Kennzahl in mehreren Projekten gebraucht wird und eventuell an anderer Stelle bereits für eine andere Auswertung für eine andere Abteilung entwickelt wurden, entgeht dem Tool-Experten und er entwickelt die Kennzahl neu. Ähnliches gilt für den ETL-Tool-Experten. Weil er nicht weiß, welche Daten in letzter Konsequenz für die Bildung von Berichten und Kennzahlen benötigt werden, greift er ins Volle und lädt schon mal Spalten und Tabellen aus den Vorsystemen auf Vorrat oder sogar doppelt unter einem anderen Namen.
Ergebnis ist ein ungeordnetes, wenig abgestimmtes und synchronisiertes Sammelsurium mit unkontrollierten synonymen und homonymen Datenobjekten, das Agilität, also rasches zielgenaues Handeln, zumindest erschwert.
Zur möglichen Rollenbeschreibung in Warehouse – Projekten kann die folgende Tabelle dienen. Sie ist sicher nicht vollständig, zeigt aber doch auf, wie breit die Kompetenzen gestreut sind, und hilft bei der späteren Zuordnung zu Personen. Der eine oder andere mag jetzt diskutieren, ob einzelne der unten aufgelisteten Aufgaben in einem Data Warehouse – Team zu lösen sind, z. B. wenn es um Hardware geht. Gerne verlässt man sich bei bestimmten Aufgaben auf andere Personen im Unternehmen, die solche Aufgaben auch schon für andere Anwendungen

übernehmen. Das kann man machen, man sollte die Aufgabenstellung dennoch mit planen, um für den Fall der Fälle die Konsequenzen einer solchen Aufgabenauslagerung bewerten und überprüfen zu können.

Rolle	Aufgaben	Kompetenz
Rolle A: Projektmanagement / Koordination	• Koordination, Steuerung und Monitoring Gesamtprojekt • (Formale) Schnittstelle zum Anwender • Schnittstelle zum Management • Zeitplanung und Projektfokus • Implementierung Kommunikationsstruktur (Zeitraum Projektdauer) • Bereitstellen der jeweiligen Informationen • Bereitstellen der jeweiligen Ressourcen • Festlegen und Kontrollieren von Dokumentationsregeln	• Kommunikation • Warehouse – Projekterfahrung • Erfahrung in Prozessanalyse • Erfahrung in Datenmodellierung • Allgemeine Warehouse – Kenntnisse • Überblick Warehouse – Technik
Rolle B: Architekturdesign	• Funktionale Beschreibung der einzelnen Lösungskomponenten • Schichtenmodell • Acting-Konzept des Systems (Durchlaufzeiten und Reaktionen) • Schnittstellenbeschreibung • Securitykonzept	• Warehouse-Architekturen • IT-Background • Übersicht über Möglichkeiten der IT-Infrastruktur
Rolle C: Informations-modellierung	• Aufnahme Informationsbedarf Endbenutzer • Erfassung bestehender Informationsobjekte (Daten und Berichte) • Erstellung Kennzahlenliste (Standard-Kennzahlen) • Festlegen Namenskonventionen • Modellierung von Auswertemodellen • Erstellung von Verbund-Modellen • Erstellung einer Referenzdaten-Struktur (extern / intern) • Selektion Stammdaten • Dokumentation des Informationsbedarfs • Pflege der Kennzahlenliste • Kennzahlenkonzept • Festlegen von Datenqualitätsstandards	• Techniken zur Erhebung von Informationsbedarfen auf der Fachebene • Kommunikations- und Fragetechniken Methoden zur Erstellung von Analyse- und Objektmodellen • Multidimensionale Modellierungsverfahren • Datenmodellierung • Metadatenkonzept
Rolle D Infrastruktur-technik	• Planung und Sizing Datenhaltung • Einsatz der relevanten DB-	• Oracle Datenbank-Konzept

	Techniken • Koordination aller Tätigkeiten rund um die Datenbank • Caching- und In-Memory-Konzept • Schlüssel- und Index-Planung • Securitykonzept • Partitioning-Konzept • Compression-Konzept • Aufbau und Pflege Kennzahlenobjekte • Metadatenverwaltung • Unterstützung ETL-Prozess	• Tiefer gehende Datenbank-Kenntnisse • Data Warehouse Konzepte • Datenmodellierung
Rolle E: Entwicklung Lade- und Updateprozess	• Erstellung übergreifendes ETL-Konzept • Entwurf der Schnittstellen des Systems • Bedienen von Schnittstellen des Systems • Erstellen Laderoutinen / Workflowprozess • Erstellen von Datenbank-Routinen	• Infrastruktur- und Schnittstellenkenntnis • Datenmodellierung • Oracle Datenbank-Konzept • Oracle Datenbank Features • ETL-Tool-Experte • SQL • PL/SQL
Rolle F: Aufbereitung Berichte / Dashboard	• Erstellung Zugriffs- und Verwendungskonzept • Festlegung von Standard Style Guide Elementen und Bedien-Systematik • Definieren von Benutzergruppen • Erstellen Standardberichte • Entwurf Dashbords	• Visualisierungs-Know how • GUI-Design • Kenntnis von Analyse-Abläufen • Tool-Wissen • Datenmodellierung
Rolle G: Weiterentwicklungskonzept	• Implementierung Kommunikationsstruktur zu den Nutzern • Versionierungskonzept • Umgang mit dem genutzten Datenmodell • Planung von Umgebungen und Übergabeverfahren • Inventarisierung von Informationen und Metadaten	• Softwareentwicklungsmethoden • Übersicht über die eingesetzte Infrastruktur
Rolle H: Betriebsplanung	• Implementierung Kommunikationsstruktur zu den Nutzern • Inbetriebnahme-Plan • Trainingsplan für Benutzer • Trainingsplan für Betrieb und Administratoren	

	• Monitoring Datenbanknutzung • Allgemeine Datenbankverwaltung (Statistiken, Sizing) • Optimierung Datenbanksystem (Indexpflege) • Pflege Kennzahlensystem • Verwaltung Security-System • Backup-Konzept • Test Recovery-Verfahren • Festlegen Patching-Verfahren	
Rolle I: Hardware-Systeme	• Koordination Rechenzentrumsabläufe • Bereitstellungsprozess-Hardware • Storage-Konzept • Server-Architektur • Netz-Verwaltung • Verfügbarkeitsplanung	• Kenntnis Hardwaresysteme • Anforderungen von Warehouse-Systeme • Betriebssystem-Knowhow (Linux)

Diese Aufgabenübersicht ist jetzt nur der erste Schritt. Es folgt noch die Zuordnung zu natürlichen Personen. Die Größe des Teams ist zunächst variabel aber eher klein zu wählen. Sie hängt natürlich sehr stark von der personellen Situation in den Abteilungen und der Komplexität der zu lösenden Aufgabenstellung ab.
Der Umfang der Gesamtaufgabenstellung spielt nur indirekt eine Rolle, wenn wir von einer inkrementellen Software-Entwicklung reden und davon, dass die Größe der Teilprojekte zugeschnitten wird. Man arbeitet überschaubare Teilschritte ab und hat automatisch kleinere Teams.

Hardware im Data Warehouse „agil“ auswählen

Wie bringt man das denn jetzt zusammen. Wir sprechen von agilen Software – Entwicklungsmethoden im Data Warehouse und was hat das jetzt mit Hardware zu tun?
Ja natürlich hängt auch das zusammen. Im Grunde sprechen wir über agile und *flexible Warehouse-Systeme* und da gehört Hardware sehr wohl mit zu der Diskussion.
Warehouse-Systeme gehören zu den Ressourcen-intensivsten und damit auch teuersten Anwendungen in der Unternehmens-IT. Vor dieser Aussage fürchten sich viele Systemplaner derart, dass sie sehr viel Energie in die möglichst exakte und lange vorausschauende Hardware-Planung investieren. In fast allen Planungsdokumenten für neue Systeme gibt es eine „Sizing-Ecke“. Es werden Initial- und Inkremental-Load-Zahlen für Tage, Monate und Jahre aufaddiert. Es sind schon Planungen gesichtet worden, in denen man den Hardware-Bedarf sogar für die nächsten 10 Jahre hochgerechnet hat. Die meisten planen in jedem Fall für mindestens 3-5 Jahre.
Aber die Hardware-Planung muss ähnlich kritisch betrachtet werden, wie der Versuch die Abfragebedürfnisse der Endanwender für die nächsten 5 Jahre vorherzusagen. Auf dem Weg dort hin wird es genügend viele Unwegsamkeiten und Veränderungen

geben, sodass dies kaum möglich sein wird. Wir haben ja gerade erst gelernt, dass das Finden neuer Geschäftsideen und das permanente Optimieren von Geschäftsprozessen massiv zur Verbesserung der Wettbewerbsfähigkeit beitragen.
Bzgl. Hardware muss dies bedeuten, dass man Systeme mit möglichst wenig Verwaltungsaufwand wählt und die flexibel skalierbar sind, ohne dass man von Beginn an in teuere Hardware investieren muss.
Wenig Verwaltungsaufwand deswegen, weil man Hardware eher als zeitlich begrenzte Verbrauchs-Ressource sehen muss und nicht als langfristige Investition. Die amerikanische Philosophie, für ein bestimmtes Projekt, *Einmal–Handware-Resourcen* einzuplanen, die nach Projektende ohne Verlust „entsorgt" werden kann, setzt sich zunehmend durch. Hat man im Rechenzentrum spezialisierte Ressourcen nur zur Verwaltung einer bestimmten Hardware implementiert, ist man mit Folgekosten daran gebunden.
Sinnvolle Konzepte für diese Vorgehensweise ist der Appliance-Gedanke, der auch in der **Exadata-Maschine** steckt. Es ist kein explizites Storage-Know how und es sind auch keine Netzexperten nötig. Zu Projektbeginn wählt man eine kleine Ausbaustufe, und wenn der Bedarf wächst, baut man das System aus. Das macht es auch leichter den Planungshorizont für die Systeme kurz zu wählen. Üblicherweise versucht man Hardware über einen Investitionszeitraum von 3 – 5 Jahren zu kalkulieren. Bei der hohen Innovationsgeschwindigkeit der Hardware-Komponenten und dem parallel stattfindenden Preisverfall sind diese Zyklen aber zu lang. Ein Zeitraum von 2 Jahren ist heute schon oft eine sinnvolle Planungsgrenze.

Pfiffige Architekturen

Was haben Architekturen mit Agilität zu tun? Die Frage bräuchte eigentlich nicht gestellt zu werden, aber viele aktuelle Diskussionen verstellen den Blick einfache und klare Dinge. Zunächst der Begriff „Architektur". Hiermit sind ein Plan und die Mittel gemeint, Informationen für einen bestimmten Zweck (hier Analyse) zu erstellen und vorzuhalten. Der Einsatz von Technologie ist also nur ein Teil, Daten- und Schichtenmodelle sind andere Teile. Eine geschickte Architektur versucht praktisch zu sein, sie

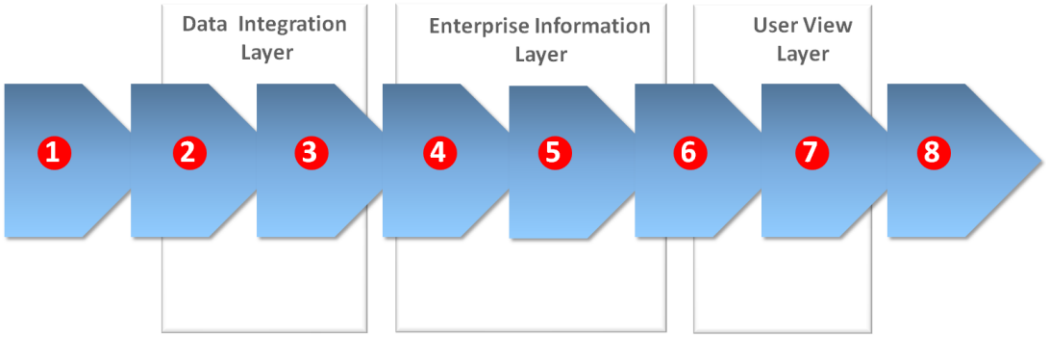
- definiert den richtigen Platz für Maßnahmen und
- verhindert vor allem Redundanzen,
- schafft Wiederverwendungsmöglichkeiten,
- legt Synchronisationspunkte fest,
- ermöglicht Standardisierung.

Im Ergebnis ist weniger Aufwand nötig, was ein System an sich agiler macht und agile Projektvorgehensweise fördert.
Wie sich z. B. *Alt-Hergebrachtes* unüberlegt verfestigt hat, erkennt man oft an den „Zwangs-„Warehouse-Architekturen und den *Regelwerken* drum herum. Aussagen wie *„Das haben wir immer schon so gemacht*" oder „*Das ist gängige Praxis bei*" lassen

aufhorchen. Also immer dann, wenn etwas als *gesetzt bezeichnet wird und nicht begründet* werden kann, hat man einen Punkt zum Anfassen gefunden.

Schichten-Architekturen erfüllen ihren Zweck in der Möglichkeit verteilte und aus unterschiedlichem Kontext stammende Informationen zusammenzubringen, zu harmonisieren, um sie dann brauchbar zu machen für eine weitere analysegerechte Modellform. Um das Ziel der Flexibilität zu erreichen, darf es hier keine Dogmen, also keine unveränderbaren Regeln geben. Es müssen Fragen erlaubt sein wie:

- Wann ist der Zustand einer analysegerechten Modellform erreicht?
- Benötigt man multidimensionale Auswertemodelle (Star Schema)?
- Ist es sinnvoll, ein solches Auswertemodell aus einem eher normalisierten Zwischenmodell, einer Kern-Warehouse – Schicht (Enterprise Layer), abzuleiten?
- Wann und wo bereitet man bekannte fixe Kennzahlen-Objekte auf?
- Können diese auch schon in der Kern-Warehouse-Schicht entstehen?



Aktionsstellen in einem Data Warehouse - Schichtenmodell

Hier spielen sicherlich weitere Faktoren eine Rolle, die allerdings hinterfragbar sein müssen. Für flexible Architekturen gelten u. a. folgende Empfehlungen (vgl. Grafik mit Positionsangaben):

- Prüfungen, Datenqualitäts-Checks etc. sollten an der frühestmöglichen Stelle in dem Schichtenmodell erfolgen. Wenn möglich bereits *vor dem Eintreten* der Daten in das Data Warehouse (Position 1-3)
- Ein pauschales 1:1 Laden von Daten aus den Vorsystemen ist zu vermeiden. Hier sollte man grundsätzlich selektieren (Position 1-2).
- Bekannte Kennzahlen sollten bereits in dem Data Warehouse an der frühestmöglichen Stelle berechnet werden. Diese Kennzahlen sind zu dokumentieren und über Kennzahlensysteme nachgelagerten Analyse-Prozessen bereitzustellen.
- Die Bildung fixer Kennzahlen-Objekte (Views, Materialized Views oder Tabellen, die aggregierte oder aggregierbare Werte in einem bekannten Format bzw. Struktur enthalten) erfolgt so früh wie möglich. D. h. unmittelbar

nach Abschluss der Harmonisierungsarbeiten des ETL-Prozesses und bevor Daten für die Bildung von Modellen für unterschiedliche Analysezwecke erstellt werden (Position 5-7).

- Daten sollten nur dann bewegt werden, mit ihnen etwas geschieht, z. B.,
    1. wenn man sie ergänzt,
    2. sie in eine andere Modellform überführt
    3. oder sie an einer funktionsbestimmten Position des Schichtenmodells ablegen will. Der letzte Punkt gilt besonders für die Ablage von Stamm- und Referenzdaten in dem Enterprise Layer.
- Daten sollten vom Enterprise Layer zum User View Layer nur dann bewegt werden, wenn sich Änderungen z. B. in der Modellform ergeben, meist, wenn multidimensionale Modelle (Star Schemen) gebildet werden.
- Die Verteilung von Daten auf mehrere Datenhaltungssysteme ist zu vermeiden. Wenn Analysen nur in einem bestimmten Datenhaltungssystem möglich sind, sollten alternative Analyseverfahren gewählt werden, die mit Daten aus dem zentrale Warehouse-System zurechtkommen. Generell sind Verfahren zu bevorzugen, bei denen die Analyse-Methoden zu den Daten gebracht werden und nicht die Daten zu einem Analyse-Tool oder –Plattform.

Einige dieser Punkte führen sicher zu Diskussionen mit den technisch Verantwortlichen für die Warehouse–Datenbanken. Hier wird das Schichtenmodell gerne als Ordnungskriterium missbraucht, um die Zugriffe der Benutzer zu steuern. Man lässt Endbenutzer nicht gerne auf die Kern-Warehouse-Schicht (Enterprise Layer) zugreifen und findet eine Reihe von Gründen dafür, die in der Regel und vor dem Hintergrund heutiger technischer Möglichkeiten nicht stichhaltig sind.

Mit In-Memory die Architektur noch flexibler gestalten

Die **In-Memory-Fähigkeit** der Oracle-Datenbank ragt mit ihren Möglichkeiten aus den Datenbanktechnologien heraus, denn sie hat das Potenzial direkt auf das Schichtenmodell Einfluss zu nehmen.

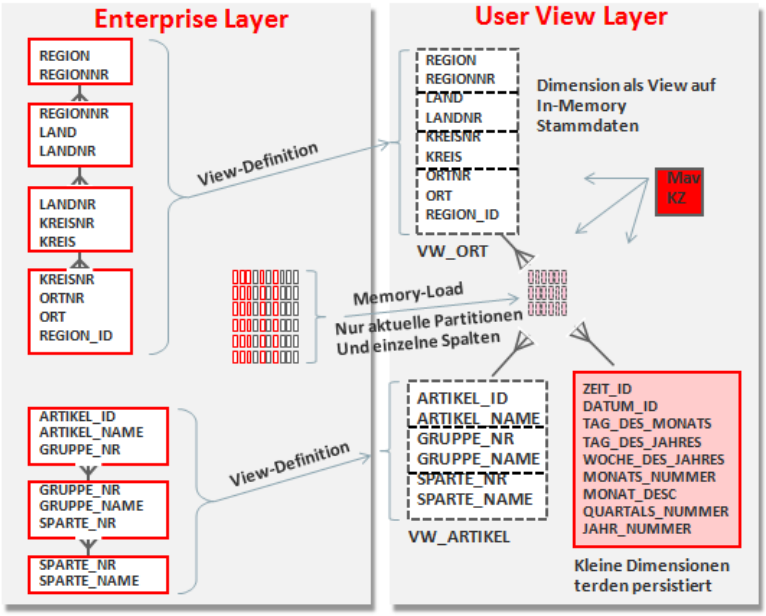
Mit der In-Memory Option können wir die 3. Schicht des Inmon'Schen Konzeptes, die Data Marts (User View Layer), in Teilen auflösen (virtualisieren). Z. B. können wir Star-Schemen, die bisher aus physischen Fakten- und Dimensionstabellen bestanden, durch In-Memory-Daten ersetzen. Für multidimensional angelegte Abfragen ändert sich nichts. Aber es gibt keine physikalischen Tabellen mehr für ein Star Schema und damit auch keine ETL-Prozesse und keinen Zeitverzug zur Beschaffung der Daten. Die Daten des Data Marts lassen sich ad hoc verändern, nachdem sich die Basis in dem Kern-Data Warehouse (Enterprise Layer) geändert hat. Auch die Strukturen eines Star Schemas lassen sich schnell ändern, indem man geänderte Objekte in den Column-Store lädt.

Die Bedeutung der zentralen Warehouse-Schicht steigt:

- Sie ist die Basis für alle virtuell konzipierten multidimensionalen Sichten,

- sie stellt weiterhin eine Data Mart-übergreifende zentrale Synchronisation aller (!) verteilte Sichten dar.
- Durch die granulare Struktur des Datenmodells (3. NF) lassen sich neue Sichten auf Geschäftsobjekte in kürzester Zeit neu erstellen, ohne in aufwendige Programmierarbeit (ETL) investieren zu müssen.

Voraussetzung für diese Vorgehensweise ist wieder eine gute Dokumentation der gespeicherten Geschäftsobjektdaten in dem Kern-Warehouse. Diesen Punkt haben wir oben bereits besprochen. Die Meta-Informationen zu den Geschäftsobjekten müssten jedoch um potenzielle Beziehungseigenschaften zwischen den Geschäftsobjekten erweitert werden, die man schließlich über Views abbildet und daraus Dimensionen mit ihren Hierarchien nachempfinden kann.



Virtualisierung eines multidimensionalen Stars durch In-Memory-Technik

Ausblick: logisches Data Warehouse

Die Virtualisierung von Data Marts durch In-Memory zeigt bereits die aufkommende Tendenz auf, Daten möglichst wenig zu bewegen. Ein großer Applikations-Software-Hersteller hat bereits das Ende des klassischen Data Warehouse ausgerufen und glaubt, Unternehmensdaten am Entstehungsort, also in den operativen Systemen mit Analysemitteln auswerten zu können. Der Gedanke ist sicher nicht zu Ende gedacht oder wird nur aus Marketinggründen formuliert. Als Gegenargument sei hier wird nur an die weiter oben bereits beschriebenen 4 Hauptaufgaben eines Data Warehouse verwiesen.

Betrachtet man jedoch *Big Data Konzepte*, mit denen zusätzliche Datenquellen für Analysen erschlossen werden oder die immer stärkere Kopplung von operativen und

dispositiven Systemen (Realtime Monitoring und Close Loop – Konzepte), so eröffnet sich die Frage, ob all diese Daten zum Zwecke der Analyse überhaupt noch in ein Data Warehouse zu laden sind. Bzgl. der Big Data Daten hat Oracle mit der Funktionalität **Big Data SQL** eine Lösung bereitgestellt, mit der Big Data-Daten in dem Kontext eines für Analysezwecke konzipierten Data Warehouse auswertbar sind, ohne diese Daten selbst in das Warehouse laden zu müssen. Mit SQL – Mitteln kann man beiden Welten, Big Data Strukturen und relational organisiertes Data Warehouse zusammenhängend abfragen.
Die technischen Möglichkeiten sind damit gegeben. Bei der Vielfältigkeit all dieser Daten wird es jedoch immer wichtiger, die Zusammenhänge zwischen den verschiedenen Datenbereichen einheitlich zu dokumentieren. Bei noch mehr unterschiedlichen Datentöpfen, Formaten, Quellen und letztlich immer neuen Geschäftsobjekten ist eine zentrale Koordination immer wichtiger, sonst *„analysiert man sich irgendwann im Kreis“*, weil die Übersicht verloren ging. Daran ändert auch eine agile Methode nichts.

Alfred Schlaucher Januar 2015