Modellorientierte DWH-Entwicklung mit SparxSystems Enterprise Architect

White Paper der solvistas GmbH

Thomas Neuböck
Melissa Schmidt
Andreas Weißenböck

Linz, Dezember 2021

1 Einleitung

Eine agile modellorientierte Vorgehensweise zur Entwicklung von BI- und DWH-Systemen wurde bereits in (Neuböck T., Raab J., Weißenböck A., März 2014) vorgestellt. In (Hiebl J., Hörak K., Linner K., Neuböck T., Raab J., Weißenböck A., Juli 2014) wurde dieser Ansatz auf einer konkreten DWH-Plattform (IBM Netezza und IBM InfoSphere) demonstriert. Auf Basis dieser Erkenntnisse wurde das Vorgehen in (Neuböck T., Raab J., Weißenböck A., Dezember 2014) konsolidiert dargestellt, mit dem Schwerpunkt auf die Modell- und Architektursicht. Die Darstellung in (Neuböck T., Pommerenke R., Raab J., Schmidt M., Weißenböck A., März 2017) bezieht sich auf eine neuere und aktualisierte Version dieser agilen modellorientierten Vorgehensweise (dort nun auch als *solvistas modellorientierte agile Vorgehensweise bei BI-/DWH-Projekten* bezeichnet). Zudem wurde diese Vorgehensweise darin auf ein konkretes Projekt zur Entwicklung einer *Enterprise Business Intelligence (EBI)* in einem österreichischen öffentlichen Bereich bezogen und auf einer konkreten DWH-Plattform (Oracle, IBM InfoSphere und Cognos) demonstriert. Im vorliegenden White Paper wird diese Vorgehensweise auf Basis weiterer österreichischer Projekte im öffentlichen Bereich (und somit bezogen auf konkrete DWH-Plattformen) betrachtet. Der Fokus liegt diesmal auf der Modellierung mit einem konkreten Modellierungswerkzeug: Enterprise Architect von SparxSystems.

Ein einführender Überblick aus früheren Darstellungen wird in Abschnitt 2 gegeben (Dimensionen eines BI-/DWH-Projektes, Modellebenen). Im Abschnitt 3 werden die Architektur, die Modelle der verschiedenen Ebenen und die iterative Vorgehensweise (ohne Bezug auf agile Methoden) vorgestellt. Hinsichtlich Architektur wird der in früheren Darstellungen verwendete Begriff der historisierten Datenschicht (HDS) in der vorliegenden

Präsentation als Ladeschicht (LDS) bezeichnet. Die Abschnitte 4 und 5 demonstrieren die fachliche Modellierung (Domänenmodell und dimensionales Faktenmodell) und logische Modellierung (Modellierung der LDS, Modellierung der IDS in Data Vault und dimensionale Modellierung der DMS) mit Fokus auf das Modellierungswerkzeug Enterprise Architect von SparxSystems. In Abschnitt 6 folgt eine Zusammenfassung als Fazit.

2 Überblick

Business Intelligence (BI) und Data Warehouse (DWH) Projekte weisen in vielerlei Hinsicht Besonderheiten auf. Insbesondere ist eine BI-/DWH-Entwicklung ein kontinuierlicher Prozess, der, verglichen mit anderen IT-Projekten, keinen so klar definierten Endzeitpunkt aufweist. Wurden Anforderungen umgesetzt, entstehen durch laufende Datenanalysearbeit der Endanwender neue Wünsche, die möglichst unverzüglich im BI-/DWH-System umgesetzt werden sollen. Bezüglich einer eingesetzten Vorgehensweise für die Entwicklung in diesem Bereich sind daher folgende Fragen zu stellen:

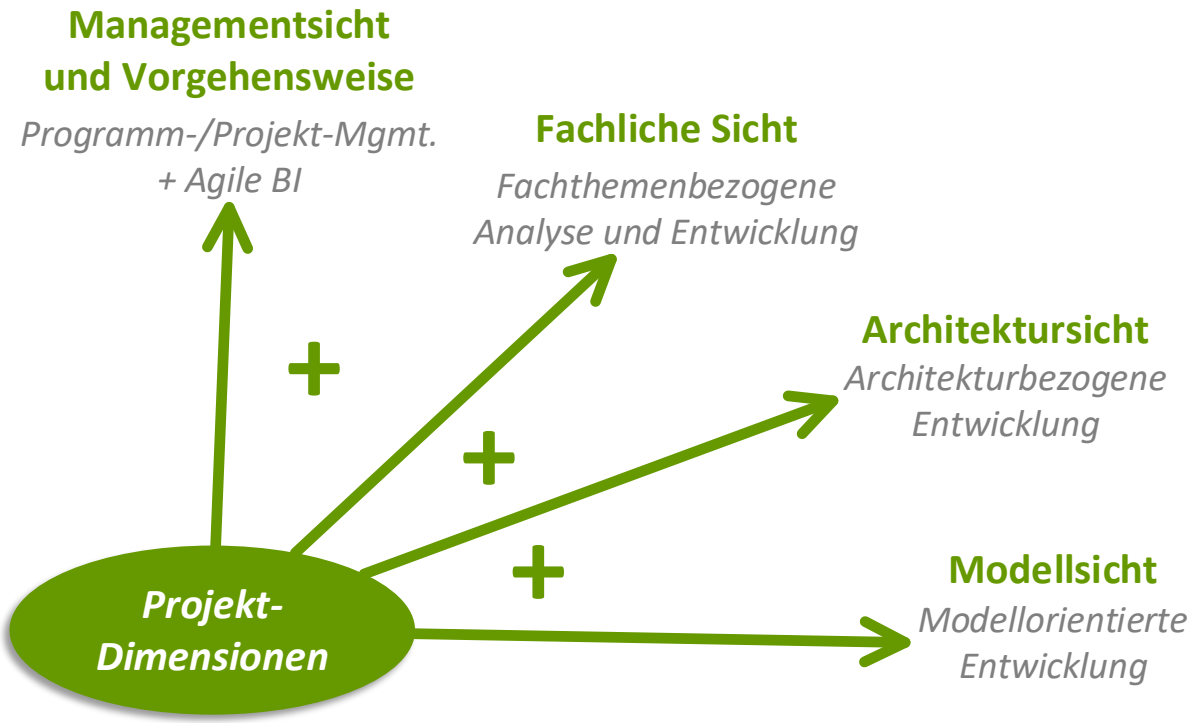
- Wie zufrieden sind unsere Kundinnen und Kunden?
- Setzen wir das um und verstehen wir, was die Kundinnen und Kunden wollen?
- Wie bewältigen wir fachliche und technische Komplexität?
- Wie sieht es mit der Transparenz und Verfügbarkeit von fachlichem und technischem Wissen aus?
- Haben wir eine Architektur (und eine Umsetzung), die einfach gewartet und erweitert werden kann?
- Haben wir ausreichende Richtlinien und Standards zur Qualitätssicherung und werden diese auch gelebt?
- Wie gehen wir bei geänderten Anforderungen oder aufgetretenen Hindernissen um?
- Wie hoch sind die Eigenständigkeit, das Verantwortungsbewusstsein und der Motivationsgrad im Entwicklungsteam?

Aus der Beantwortung dieser Fragen ergeben sich folgende Forderungen an ein effektives und effizientes Vorgehensmodell bei der Entwicklung von BI-/DWH-Systemen:

- Kundenorientiert Umsetzung von Fachthemen
- Verringerung der Kommunikations- und Verständnislücke zwischen Fachlichkeit und Technik
- Verbesserter Umgang mit der fachlichen und technischen Komplexität durch Anforderungszerlegung und kurze Umsetzungszyklen
- Breite Wissensstreuung in der Abteilung
- Einführung und Umsetzung einer leicht wartbaren und erweiterbaren Architektur
- Qualitätsgesicherte Lösungen
- Flexibilität bei sich ändernden Anforderungen oder auftretenden Hindernissen

- Motivationsförderung in den Projektteams

In der angewendeten Vorgehensweise werden die folgend dargestellten Sichtweisen kontinuierlich im Fokus gehalten:



Die Managementsicht und Projektvorgehensweise verlangt ein übergeordnetes Programm- und Projektmanagement. Darin eingebettet können Projekte und Programme auch in einer agilen Weise durchgeführt (agile BI). Das BI-System wird mit dem Grundsatz „Think Big, Act Small“ schrittweise erweitert. Bei komplexen Fachthemen sind Evaluierungsphasen oder prototypische Umsetzungen im Sinne einer evolutionären Vorgehensweise (oder gemäß „Spiral-Modell“) denkbar.

Die Tiefe von Fachthemen erfordert eine Zerlegungsstrategie, um Komplexität zu reduzieren. Verschiedene Themen verlangen auch unterschiedliche fachliche Sichtweisen. Mit dem Vorgehensmodell wird eine fachthemenbezogene BI-/DWH-Entwicklung erleichtert.

Die BI-/DWH-Architektur muss wichtige Anforderungen an ein zukunftsorientiertes BI-/DWH-System erfüllen, bspw. eine flexible Erweiterbarkeit, die Bereitstellung historisierter Informationen, Datenintegration und Definition eines „Single Point of Truth“ oder die Bereitstellung verschiedener fachlicher Sichtweisen.

Die Vorgehensweise in BI-/DWH-Projekten fordert eine modellorientierte Entwicklung. Modelle erlauben die Einbringung verschiedener Sichtweisen und ergeben eine formale Dokumentation. Drei Modellebenen werden unterschieden:

- Fachliche Modellebene (konzeptuelle Modelle)
- Logische Modellebene
- Physische Modellebene

In der fachlichen Ebene werden Anforderungen und Sichtweisen des Fachbereichs beschrieben und modelliert. In der logischen Modellebene werden die fachlichen

Anforderungen und Sichtweisen in der BI-/DWH-Architektur plattformunabhängig abgebildet. Die physische Modellebene definiert die Umsetzung der logischen Modellebene auf eine konkreten BI-/DWH-Plattform.

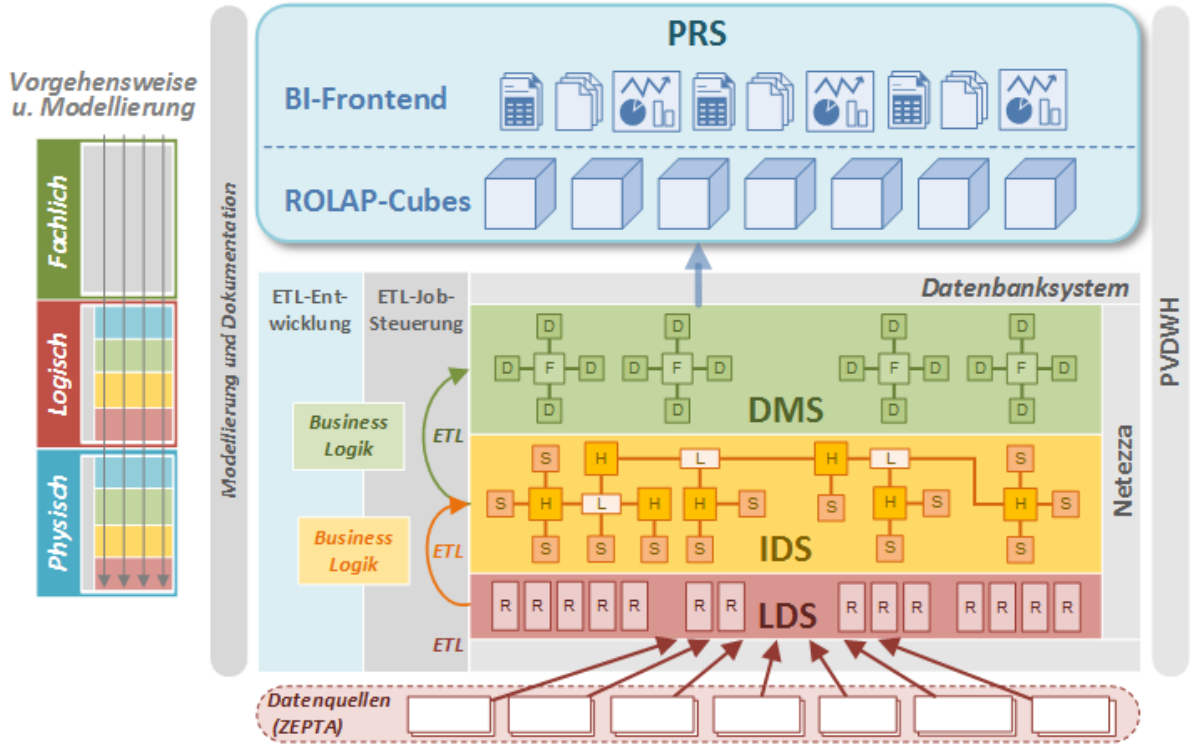
3 Architektur und Vorgehensweise

In diesem Abschnitt wird auf die zugrundeliegende generische BI- und DWH-Architektur eingegangen. Das Umfeld einer konkreten BI- und DWH-Plattform der zugrundeliegenden Projekte dient dabei als Basis für die Darstellung.

3.1 Architekturschichten

Die Architektur eines BI-/DWH-Systems muss ein festes und stabiles technisches Fundament für die Umsetzung fachlicher Anforderungen bilden. Die Änderbarkeit und Ausbaufähigkeit ist hinsichtlich sich ändernder und neuer Anforderungen zu gewährleisten. DWH-Schichten müssen um fachliche Themen einfach erweitert werden können.

Das gesamte Business Intelligence (BI) System gliedert sich in drei Datenschichten und einer Präsentationsschicht (PRS), die in den folgenden Unterabschnitten kurz erläutert werden:



3.1.1 Datenschichten

Die Datenschichten werden auf der DWH-Appliance IBM PureData for Analytics (Netezza) verwaltet. Es gibt drei Datenschichten, die jeweils eine separate Aufgabe wahrnehmen: Ladeschicht (LDS), Integrierte Datenschicht (IDS) und Data Mart Schicht (DMS). Alle Datenschichten werden in drei verschiedenen Umgebungsarten bereitgestellt: Entwicklungsumgebungen, Testumgebungen und Produktionsumgebungen.

Für die Benennung der Datenbanken können dazu Namenskonventionen eingeführt werden:

*DWH-Name_Umgebung*_D_*Schicht*, wobei

DWH-Name = Datenschichtenübergreifender Name für das gesamte DWH

Umgebung =

- D0, D1, D2, … für Entwicklungsumgebungen (falls es mehrere gibt mit Nummerierung)
- T0, T1, T2, ... für Testumgebungen (falls es mehrere gibt mit Nummerierung)
- P0, P1, P2, … für Produktionsumgebungen (falls es mehrere gibt mit Nummerierung)

Schicht =

- LDS für Ladeschicht
- IDS für integrierte Datenschicht
- DMS für Data Mart Schicht

Das Kürzel D drückt aus, dass es sich um eine Datenbank handelt. Davon unterschieden werden die Kürzel G und U für User Group und User.

Beispiele:

- PVDWH_D0_D_LDS: definiert im Data Warehouse PVDWH die Ladeschicht (LDS) der Entwicklungsumgebung D0 als Datenbank (Kürzel D)
- PVDWH_T1_D_IDS: definiert im Data Warehouse PVDWH die integrierte Datenschicht (IDS) der Testumgebung T1 als Datenbank (Kürzel D)
- PVDWH_P1_D_DMS: definiert im Data Warehouse PVDWH die Data Mart Schicht (DMS) der Produktionsumgebung P1 als Datenbank (Kürzel D)

3.1.1.1 Ladeschicht (LDS)

In die Ladeschicht werden die Daten aus den operativen Systemen möglichst 1:1 übernommen. Es findet maximal eine erforderliche Anpassung in den Zieldatentypen statt, falls die Quelldatentypen sinnvollerweise nicht 1:1 übernommen werden können. Zusätzlich werden die Daten noch mit Verwaltungsinformation angereichert (z.B. Ladezeitpunkt).

3.1.1.2 Integrierte Datenschicht (IDS)

In der integrierten Datenschicht (LDS) werden die Daten der LDS zusammengeführt und konsolidiert. Ziel ist es, eine verbesserte „Datenhygiene“ und eine unternehmensweite Sichtweise über alle Daten (aus verschiedenen Quellen) zu erlangen. Logisch werden die Daten der IDS als Data Vault Modell geführt.

3.1.1.3 Data Mart Schicht (DMS)

Die Data Mart Schicht (DMS) stellt die Auswerteschicht dar. Endbenutzerwerkzeuge greifen auf diese Datenschicht zu. Logisch werden die Daten in Star-Schemen aufgebaut.

3.1.2 ETL-Entwicklung und Job-Steuerung

ETL-Jobs zur Beladung aller Datenschichten werden mit Pentaho Data Integration (PDI) und SQL-Anweisungen entwickelt. Erforderliche Transformationen (Business Logik) vor allem von der LDS in die IDS und von der IDS in die DMS werden damit umgesetzt.

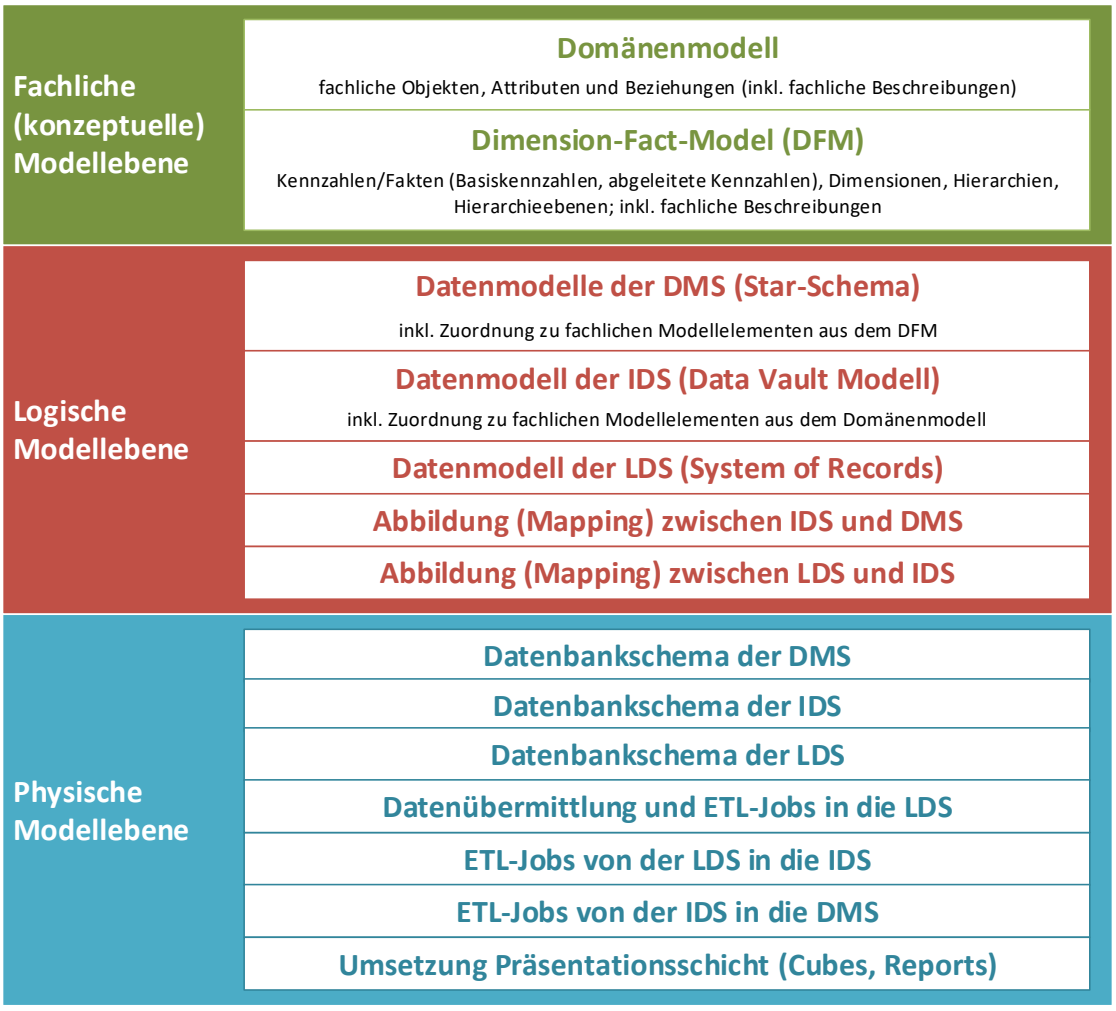
3.1.3 Präsentationsschicht

In der Präsentationsschicht greift der Endanwender über BI-Werkzeuge auf die Data Mart Schicht zu. Die Präsentation wird im Folgenden nicht weiter betrachtet.

3.2 Modellierung

Modellierung dient zum Erreichen eines gemeinsamen Verständnisses (fachlich und technisch) über das Umzusetzende, zur Dokumentation und zur Vorgabe für die Implementierung.

Die folgende Abbildung bietet einen Überblick über alle Modellebenen, die später im Detail beschrieben werden:

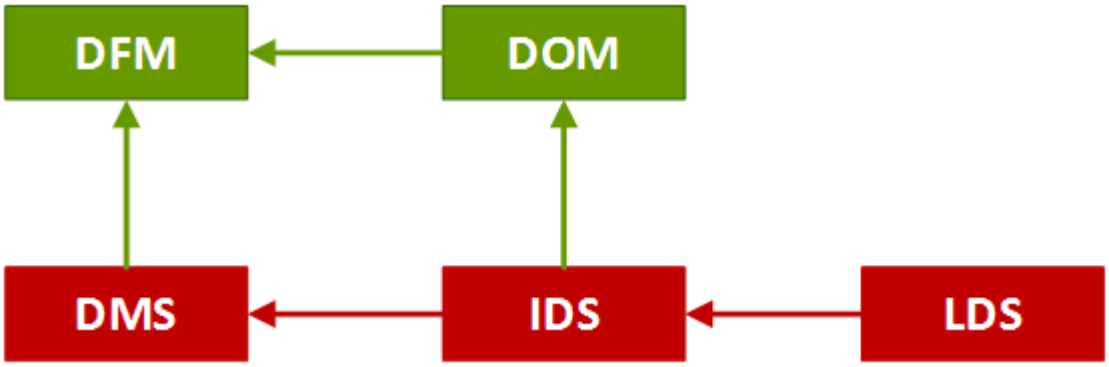


Auf der fachlichen oder konzeptuellen Modellebene werden Domänenmodelle (DOM) und Dimension-Fact-Models (DFM) erstellt. Das Domänenmodell bildet fachliche Objekte, Attribute und Beziehungen ab. Das DFM stellte eine fachliche Auswertungssicht zur Verfügung (Kennzahlen, Fakten, Auswertungsdimensionen, Auswertungshierarchien). Im DFM ist ein Bezug zum Domänenmodell herzustellen. Beide Modelltypen abstrahieren noch von der eigentlichen Tabellendefinition in der Datenbank. Die fachliche Modellebene soll auch fachliche Beschreibungen beinhalten.

Die logische Modellebene definiert gibt die Tabellen der drei Datenschichten (LDS, IDS, DMS) und beschreibt die Abbildungen zwischen den Schichten, d.h. wie werden die Attribute der IDS aus der LDS bzw. wie werden die Attribute der DMS aus der IDS ermittelt. Außerdem ist in der logischen Modellebene der Bezug zur fachlichen Modelleben herzustellen, d.h. welche Elemente der IDS werden durch welche Elemente des Domänenmodells bzw. welche Elemente der DMS werden durch welche Elemente des DFM fachlich beschrieben. Die fachliche Beschreibung der LDS müsste mit fachlichen Beschreibungen aus den operativen Datenquellen in Beziehung gesetzt werden – dies wird in der vorgeschlagenen Vorgehensweise nicht gefordert.

Die physische Modellebene stellt die eigentliche Umsetzung dar (Datenbankschemata, ETL-Jobs, Cubes, Reports, …).

Im Folgenden sind die Abbildungen zwischen den Modellebenen und Modellarten schematisch dargestellt:



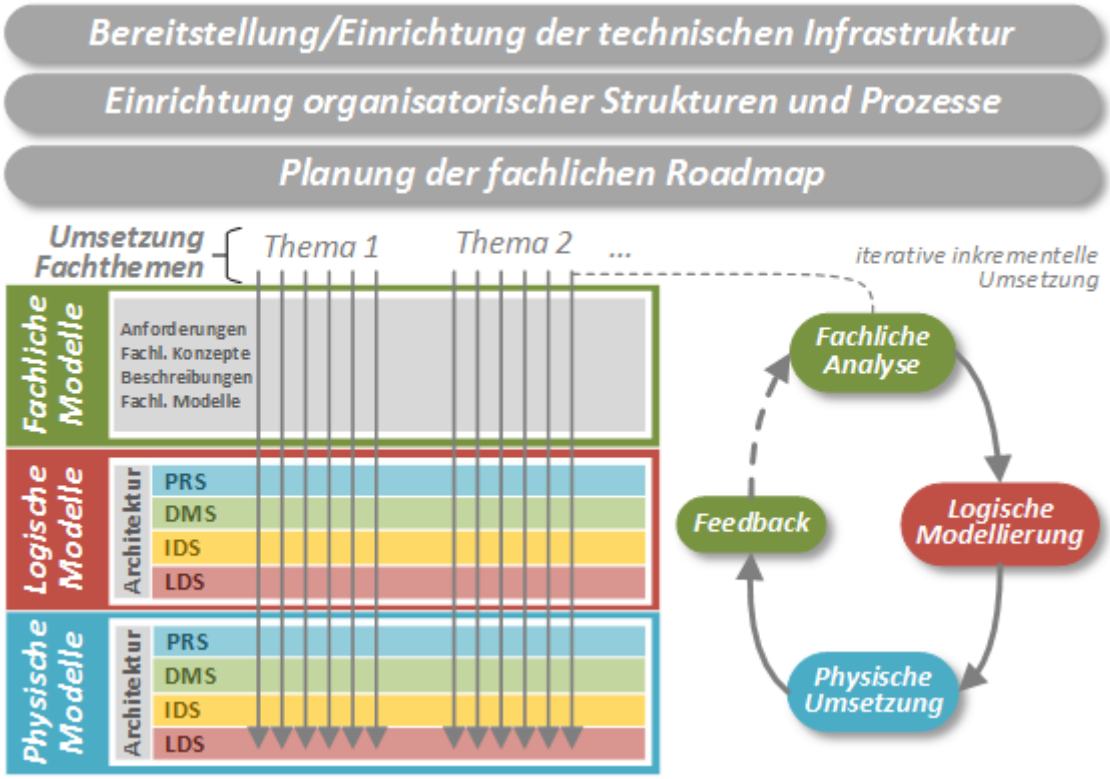
In der IDS und in der DMS existieren Verweise auf fachliche Informationen im Domänenenmodell (DOM) bzw. DFM. Elemente des Domänenmodells werden zur Definition des DFM herangezogen. Auf der logischen bzw. technischen Ebene sind die Mappings von der LDS in die IDS und von der IDS in die DMS beschrieben.

Im Enterprise Architect wird empfohlen die Modellebenen und Architekturschichten mit Packages in einer hierarchischen Form abzubilden:

- Fachliche Modelle
  - DFM
  - DOM
- Logische Modelle
  - DMS
  - IDS
  - LDS

3.3 Allgemeine Vorgehensweise

Die folgende Abbildung symbolisiert eine modellorientierte schrittweise Umsetzung eines DWHs:



In dieser Darstellung werden die Modellebenen und Architekturschichten als notwendig und gegeben vorausgesetzt. Neben allgemeinen Themen, die als „Querlieger“ betrachtet werden können (Bereitstellung und Einrichtung der technischen Infrastruktur, Einrichtung organisatorischer Strukturen und Prozesse und die Planung einer fachlichen Roadmap) wird die Umsetzung fachlicher Themen als ein iterativer, explorativer und inkrementeller Prozess betrachtet.

Ein Fachthema wird für die Umsetzung im DWH nicht vollständig bis ins letzte Detail durchgeplant. Dies ist in der Regel auch gar nicht möglich, da zum Beispiel bei der Anbindung einer neuen Datenquelle diese zuerst „erforscht“ werden muss, d.h. werden zum Beispiel nur Teile der neuen Daten ins DWH integriert (vielleicht zu Beginn nur bis in die LDS), dann werden diese genauer analysiert und interpretiert (Data Profiling und Data Assessment), was einem explorativen Vorgehen entspricht. Erst dann kann entschieden werden, inwiefern diese Teile zum Beispiel in Kennzahlen weiterverwendet werden können. Fachliche Anforderungen werden somit inkrementell umgesetzt. Dieser Vorgang findet wiederholt (iterativ) statt und schlägt sich auch im Modellierungsprozess nieder: fachliche Analyse (und Modellierung), logische Modellierung, physische Modellierung (und Umsetzung). Gewonnene Erkenntnisse führen zu einem Feedback, welches wieder weitere fachliche Analysen initiiert – eine neue Iteration startet.

Obwohl an dieser Stelle nicht explizit auf eine agile Vorgehensweise eingegangen wird, ist es nahe liegend (wenn auch nicht zwingend notwendig), dieses iterative Vorgehen agil abzuwickeln. Der Fokus in diesem White Paper liegt auf dem modellorientierten Vorgehen mit einer konkreten Werkzeugunterstützung (Enterprise Architect von SparxSystems) zur Modellierung. Im Folgenden werden die zur Modellierung relevanten Ebenen (fachliche und

logische Modellierung) auf Basis des Enterprise Achitect genauer erläutert. Die physische Modellebene kann, bezogen auf das Modellierungswerkzeug, als durch Code-Generierung entstehende Ansammlung von Artefakten wie zum Beispiel Data Definition Language Skripten (DDL-Skripten) verstanden werden.

4 Fachliche Datenmodelle

Fachliche bzw. konzeptuelle Datenmodelle liefern fachliche Informationen und abstrahieren noch von der technischen Umsetzung in Datenbanken. Wir stellen im Folgenden das Domänenmodell (DOM) und das Dimensional Fact Model (DFM) vor.

4.1 Domänenmodell (DOM)

Im Domänenmodell (DOM) werden die für das DWH relevanten Informationen aus einer fachlichen Sichtweise und möglichst noch unabhängig von Quellsystemen modelliert. Im Vordergrund steht dabei noch nicht die Auswertesicht sondern die Darstellung fachlicher Sachverhalte und Beziehungen. Die Modelle werden als UML-Klassendiagramme erstellt. Prinzipiell sind darin alle dem UML 2.0 Standard entsprechenden Modellelemente verwendbar.

Klassen, die **Geschäftsobjekten** entsprechen, werden mit dem Stereotyp <<*businessObject*>> gekennzeichnet. In einem Geschäftsobjekt sind ein oder ggf. auch mehrere Attribut(e) als **Business Key** zu identifizieren. Dieser ist mit dem Stereotyp <<*businessKey*>> zu kennzeichnen. Geschäftsobjekte und Business Key werden später für die Hub-Definition in der IDS herangezogen.

Das Modellieren **weiterer Klassen**, die keine Geschäftsobjekte darstellen, ist erlaubt und oft auch sinnvoll.

Eine Klasse im Domänenmodell besteht nur aus Attributen – Datentypen sind nicht erforderlich. Diese Attribute sind als rein **fachliche Attribute** zu betrachten (keine „technischen Attribute“). Methoden werden in einer Klasse des Domänenmodells nicht definiert. Im Minimalfall wird bei der Modellierung auch auf Attribute verzichtet, d.h. es werden nur Klassen (als Geschäftsobjekte) dargestellt.

Zwischen den Klassen werden fachlich relevante **Beziehungen** (Assoziation und Vererbung) angegeben. Eine Beziehung wird inkl. Leserichtung beschriftet. Kardinalitäten (Multiplizitäten) sollten angegeben werden. Falls sinnvoll sind auch Rollen an Beziehungsenden zu definieren.

Alle Modellelemente des Domänenmodells (Klasse, Attribute, Beziehungen, Rollen) sollten **fachliche Beschreibungen** aufweisen. Im Enterprise Architect kann dafür das „Notes-Textfeld“ zum jeweiligen Modellelement zu verwendet werden. Zusätzlich kann am Diagramm selbst das UML-Element „Note“ („Notizzettel“) eingesetzt werden. Verweise auf andere Modellelemente können und sollen über **Hyperlinks** im Beschreibungsfeld aufgenommen werden.

Beschreibungsfeld:

Fachliche Beschreibung des Elementes des Domänenmodells

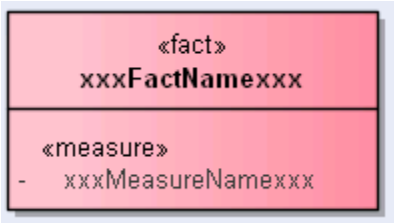
4.2 Dimensionales Faktenmodell (DFM)

Das Dimension-Fact-Model (DFM) stellt eine fachliche Auswertesicht zur Verfügung. Es wird als UML-Klassenmodell erstellt, wobei die hier beschriebenen Stereotypen (inkl. farblicher Kennzeichnung) verwendet werden könne. Im Enterprise Architect können dafür Patterns vorbereitet werden.

4.2.1 Fakten und Kennzahlen

Fakten (Stereotyp <<*fact*>>) repräsentieren Geschäftsereignisse (z.B. REHA-Aufenthalt) oder fachliche Zustände (z.B. Antragsstatistik per Monatsende), die ausgewertet werden können. Sie beinhalten (Basis-)**Kennzahlen** (Stereotyp <<*measure*>>), die in der Regel aggregiert werden können (Summierung, Durchschnittsbildung, Maximum-/Minimumermittlung) – z.B. Anzahl Aufenthaltstage oder Anzahl Anträge.

Im Enterprise Architect kann zum Beispiel folgendes **Pattern** vorbereitet und zur Modellierung wiederverwendet werden:



xxxFactNamexxx ist ein Platzhalter für den Klassennamen. Als Attribute können eine oder mehrere Kennzahlen definiert werden. *xxxMeasureNamexxx* steht exemplarisch als Platzhalter für eine Kennzahl.

4.2.2 Dimensionen

Dimensionen (Stereotyp <<*dimension*>>) repräsentieren fachliche Objektklassen nach denen ausgewertet werden kann (z.B. nach Versicherten oder Vertragspartnern). Die Zeit wird in der Regel als besondere Dimension immer mitgeführt.

Im Enterprise Architect kann folgendes **Pattern** vorbereitet und in der Modellierung wiederverwendet werden:

xxxDimNamexxx ist ein Platzhalter für den Klassennamen einer Dimension.

Eine Dimension wird mit einer oder mehreren Faktenklassen verbunden (über eine Assoziation). Es ist auch möglich, dass eine Dimension öfters mit einer Faktenklasse in Beziehung steht. In diesem Fall sollen unbedingt Rollen eingetragen werden (**Dimensionsrollen**). Z.B. kann die Zeitdimension öfters mit der Antragsstatistik verbunden werden (Antragsdatum oder Erledigungsdatum).

4.2.3 Dimensionsebenen und Dimensionshierarchien

Eine **Dimensionsebene** (Stereotyp <<*level*>>) repräsentiert eine fachliche Auswertungsebene einer Dimension. Beispielsweise könnte ein Versicherter auf Basis der Einzelperson (identifiziert über die Versicherungsnummer), hinsichtlich Geschlecht, Wohnbezirk oder Bundesland ausgewertet werden. Versicherungsnummer, Geschlecht, Bezirk und Bundesland stellen somit Dimensionsebenen dar. Die Zeitdimension könnte als Ebenen Datum, Monat (inkl. Jahr) und Jahr aufweisen.

Das folgende **Pattern** kann im Enterprise Architect vorbereitet und bei der Modellierung wiederverwendet werden:

xxxLevelNamexxx ist ein Platzhalter für den Klassennamen der Dimensionsebene.

Eine Dimension kann noch zusätzliche Eigenschaften besitzen, die als **beschreibende Attribute** (Stereotyp <<*descriptiveAttribute*>>) bezeichnet werden. Beispielsweise können zur Dimensionsebene Bundesland die beschreibenden Attribute Bevölkerungsanzahl und Fläche abgegeben werden.

Enthält eine Dimensionsebene eine oder mehrere beschreibende Attribute, kann folgendes **Pattern** verwendet werden:

xxxDescrAttrxxx steht exemplarisch als Platzhalter für ein beschreibendes Attribut.

Die Ebenen einer Dimension werden zu **Dimensionshierarchien** verbunden (über eine Assoziation mit 1:*-Multiplizität. Zum Beispiel: Ein Bundesland hat mehrere Bezirke und ein Bezirk sind mehrere Versicherungsnummern zugeordnet. Entlang einer Dimensionshierarchie können Kennzahlen **aggregiert** werden, z.B. Anzahl Anträge pro Versicherungsnummer, pro Bezirk oder pro Bundesland. Die feinste Dimensionsebene (z.B. Versicherungsnummer) wird über eine Assoziation mit der Dimensionsklasse verbunden.

4.2.4 Fachliche Beschreibungen

Alle Modellelemente des DFM sollten **fachliche Beschreibungen** aufweisen. Im Enterprise Architect kann dafür das „Notes-Textfeld“ zum jeweiligen Modellelement verwendet werden. Zusätzlich kann am Diagramm selbst das UML-Element „Note“ („Notizzettel“) eingesetzt werden.

Verweise auf andere Modellelemente können und sollen über **Hyperlinks** im Beschreibungsfeld aufgenommen werden. Insbesondere soll ein Modellelement des DFM immer einen Verweis auf ein Modellelement des Domänenmodells aufweisen.

Beispielhafte Strukturierung des Beschreibungsfeldes:

Fachliche Beschreibung des Elementes des DFM **DOM und Ermittlungslogik:** *Hyperlink zu Elementen im Domänenmodell und fachliche Ermittlungslogik*

5 Logische Datenmodelle

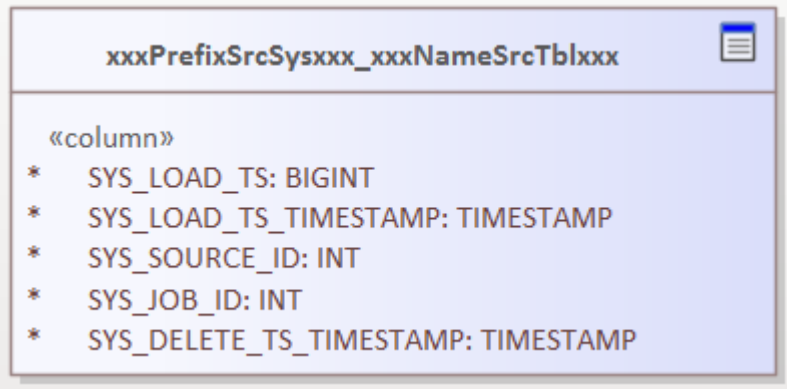
Logische Datenmodelle liefern die Vorgaben zur Implementierung der jeweiligen Datenschicht in der Datenbank. Jede Datenschicht erfüllt dabei einen bestimmten Zweck. Die logischen Datenmodelle spezifizieren die konkreten Datenbanktabellen. Die in den Datenbanken erstellten Tabellen bzw. die dazugehörenden DDL-Skripten stellen die physische Modellebene dar.

Allgemeine **Namenskonventionen**: Auf der logischen Datenmodellebene werden Tabellen- und Attributsnamen ausschließlich mit Großbuchstaben (ohne Umlaute), Ziffern (aber nicht beginnend) und/oder mit „_“ (underscore) geschrieben. Leerzeichen und weitere Sonderzeichen sind nicht erlaubt.

5.1 Ladeschicht (LDS)

In der Ladeschicht (LDS) werden Datenbanktabellen der operativen Quellsysteme soweit möglich unverändert abgebildet und zusätzlichen Metadateninformationen (SYS-Attribute) angereichert. Die LDS wird regelmäßig beladen. Häufig werden in den Zieltabellen immer nur die im operativen System neu hinzugekommenen und geänderten Datensätze hinzugefügt bzw. werden die im operativen System gelöschten Datensätze mit einem Löschzeitpunkt versehen.

Im Enterprise Architect könnte zum Beispiel folgendes **Pattern** definiert und bei der Modellierung wiederverwendet werden:



Mit entsprechenden Namenskonventionen soll der Bezug zur Quelltabelle (zum Beispiel mit einem Präfix) hergestellt werden können. Im Pattern werden insbesondere die zusätzlichen Metadaten-Attribute vorgegeben. Zudem sind eventuell Datentyp-Mappings vorzudefinieren.

Falls notwendig können die Modellelemente der LDS im Enterprise Architect („Notes-Textfeld“ bzw. „Notizzettel“) beschrieben werden. Da Beschreibungen von operativen

Quellen häufig an anderen Stellen existieren, wird eine Beschreibung der LDS nicht zwingend gefordert.

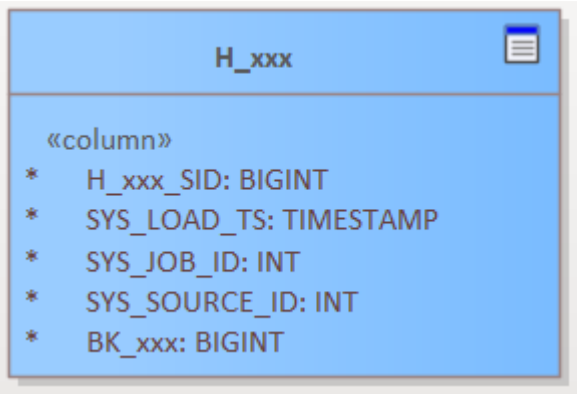
5.2 Integrierte Datenschicht (IDS)

In der integrierten Datenschicht (IDS) erfolgt eine Datenkonsolidierung aller Quellen, möglicherweise angereichert um zusätzliche Business Logik. Als Modellierungsmethode wird Data Vault verwendet. Data Vault definiert drei elementare Modellierungselemente: Hub, Satellit und Link.

5.2.1 Hub

Eine Hub-Tabelle repräsentiert eine Geschäftsobjekt (bzw. ein „fachliches Objekt“), wobei die Hub-Tabelle nur eine technische und fachliche Identifikation des Geschäftsobjekts (technischer und fachlicher Schlüssel) enthält. Weitere Informationen eines Geschäftsobjekts liegen in Satellit-Tabellen die zur Hub-Tabelle gehören.

Im Enterprise Architect kann zum Beispiel folgendes **Pattern** vorbereitet und zur Modellierung wiederverwendet werden:



Entsprechende Konventionen bei der Vorgabe eines Hub-Namens, die enthaltenen Metadateninformationen (SYS-Attribute), künstliche und fachliche Schlüssel (SID bzw. BK) können darin vorgegeben werden. Zusätzlich können bei der Modellierung Verweise ins Domänenmodell und Mapping-Vorschriften von der LDS in die IDS angegeben werden.

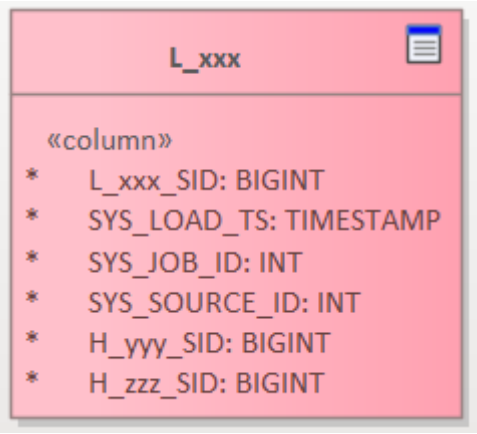
Beispielhafte Strukturierung des Beschreibungsfeldes:

Allgemeine Beschreibung des Elementes der Hub-Tabelle **DOM*:*** *Hyperlinks zu Elementen im Domänenmodell und, falls erforderlich, zusätzliche Beschreibungen* **LDS und Ermittlungslogik:** *Beschreibung der Ermittlungslogik inkl. Verweise in die LDS (über Hyperlinks)*

5.2.2 Link

Beziehungen zwischen mehreren Hubs werden über Link-Tabellen abgebildet. Ein Link enthält zwei oder mehrere Fremdschlüssel auf Hub-Tabellen, die über den vorliegenden Link in Beziehung gebracht werden.

Beispielhaftes **Pattern** im Enterprise Architect:



Im vorliegenden Beispiel werden ein Präfix zur Benennung eines Links und SYS-Attribute vorgegeben. Zudem sind ein künstlicher Schlüssel für einen Link-Eintrag (L_..._SID) und weitere künstliche Schlüsseln für die referenzierten Hub-Einträge (H_..._SID) enthalten. Zusätzliche Verweise ins Domänenmodell und Mapping-Vorschriften von der LDS in die IDS können ebenfalls angegeben werde.

Beispielhafte Strukturierung des Beschreibungsfeldes:

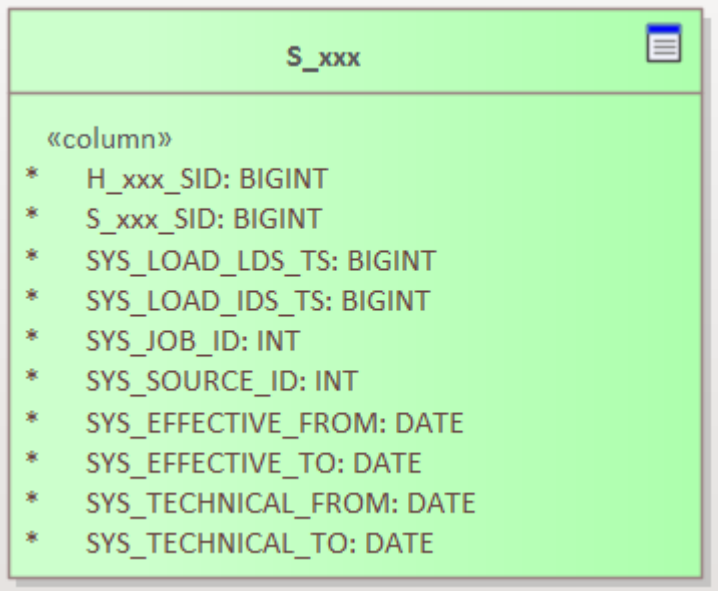
Allgemeine Beschreibung des Elementes der Link-Tabelle **DOM:** *Hyperlinks zu Elementen im Domänenmodell und, falls erforderlich, zusätzliche Beschreibungen* **LDS und Ermittlungslogik:** *Beschreibung der Ermittlungslogik inkl. Verweise in die LDS (über Hyperlinks)*

5.2.3 Satellit

Ein Satellit enthält alle zu einem Hub oder Link gehörenden Kontextinformationen. Er ist somit der eigentliche Träger der fachlichen Informationen. Zu einem Hub bzw. zu einem Link kann es mehrere Satelliten geben. Welche Satelliten definiert werden, ist von unterschiedlichen Faktoren abhängig (z.B. fachlicher Kontext, Historisierungsfrequenz).

In einem Satelliten sind Daten historisiert, d.h. zu einem Hub- bzw. Link-Eintrag kann es mehrere Satelliteneinträge geben. Normalerweise ist zu einem Zeitpunkt nur genau ein Satelliteneintrag zu einem Hub bzw. Link gültig. Gibt es mehrere gültige Satelliteneinträge, spricht man von multi-aktiven Satelliten (z.B. eine Kunde hat mehrere Adressen).

Beispielhaftes **Pattern** im Enterprise Architect:



Ein Satellit wird wiederum gemäß entsprechenden Namenskonventionen und Konventionen zu den Attributen modelliert. Zusätzlich enthält ein Satellit Datumswerte zur Historisierung. Verweise ins Domänenmodell und Mapping-Vorschriften von der LDS in die IDS können wiederum in einem Beschreibungsfeld angegeben werden.

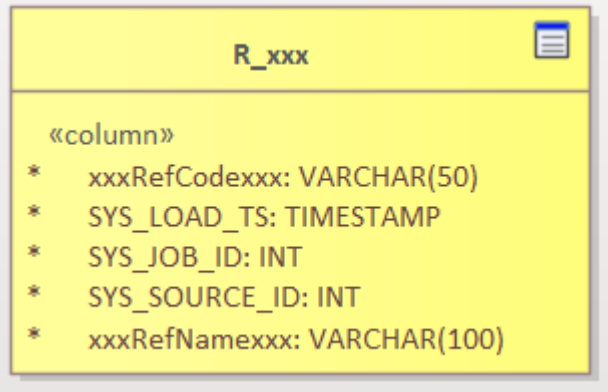
Beispielhafte Strukturierung des Beschreibungsfeldes:

Allgemeine Beschreibung des Elementes der Satellit-Tabelle **DOM*:*** *Hyperlinks zu Elementen im Domänenmodell und, falls erforderlich, zusätzliche Beschreibungen* **LDS und Ermittlungslogik:** *Beschreibung der Ermittlungslogik inkl. Verweise in die LDS (über Hyperlinks)*

5.2.4 Referenztabelle

Einfache Referenzdaten, die des Aufwands wegen nicht über Hub-/Satellitenkombinationen modelliert werden sollen, können als Referenztabelle spezifiziert werden.

Im Enterprise Architect kann folgendes **Pattern** zum Erstellen einer Referenztabelle verwendet werden:



Beispielhafte Strukturierung des Beschreibungsfeldes:

Allgemeine Beschreibung des Elementes der Referenztabelle **DOM:** *Hyperlinks zu Elementen im Domänenmodell und, falls erforderlich, zusätzliche Beschreibungen* **LDS und Ermittlungslogik:** *Beschreibung der Ermittlungslogik inkl. Verweise in die LDS (über Hyperlinks)*

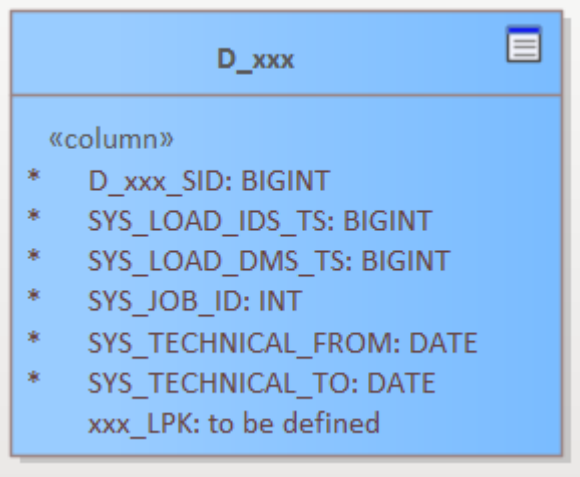
5.3 Data Mart Schicht (DMS)

In der Data Mart Schicht (DMS) erfolgt die relationale Umsetzung der fachlichen Auswertesicht. Als Grundlage dazu dienen die fachlichen (konzeptuellen) Modelle so wie sie im Dimensional Fact Model (DFM) dargestellt werden. Ein DMS-Modell wird in der Regel als Star-Schema (mit Dimensionen und Fakten) umgesetzt, dabei werden bewusst Redundanzen aufgenommen (zum Unterschied zur strikten Normalisierung in der Datenmodellierung von operativen Systemen). Auf der anderen Seite werden Konsolidierungen im Sinne von gemeinsamen Dimensionen forciert.

5.3.1 Dimensionstabelle

Eine Dimensionstabelle enthält alle im DFM fachlich spezifizierten Informationen zu Auswertungsdimensionen. Dimensionen können historisiert werden – müssen aber nicht – abhängig von den fachlichen Anforderungen und von Umsetzungsvarianten.

Im Enterprise Architect kann folgendes **Pattern** zum Erstellen einer Dimensionstabelle verwendet werden:



Zum Tabellenname und den technischen und fachlichen Attributen (z.B. SIDs oder fachliche Schlüssel wie LPKs = logischer Primary Key) können Verweise ins DFM und Mapping-Vorschriften von der IDS in die DMS angegeben werden.

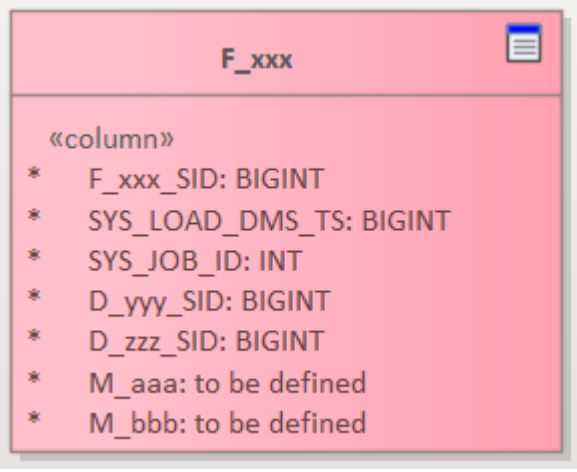
Beispielhafte Strukturierung des Beschreibungsfeldes:

Allgemeine Beschreibung des Elementes der Dimensionstabelle **DFM*:*** *Hyperlinks zu Elementen im DFM und, falls erforderlich, zusätzliche Beschreibungen* **IDS und Ermittlungslogik:** *Beschreibung der Ermittlungslogik inkl. Verweise in die IDS (über Hyperlinks)*

5.3.2 Faktentabelle

Eine Faktentabelle enthält alle im DFM fachlich spezifizierten Informationen zu den auszuwertenden Kennzahlen. Sie enthält die Kennzahlenwerte und die Verweise auf die Dimensionen, nach denen ausgewertet werden kann.

Im Enterprise Architect kann folgendes **Pattern** zum Erstellen einer Faktentabelle verwendet werden:



Zum Tabellenname und den technischen und fachlichen Attributen (z.B. SIDs als Verweise auf Dimensionen oder Measure-Attribute) können wiederum Verweise ins DFM und Mapping-Vorschriften von der IDS in die DMS angegeben werden.

Beispielhafte Strukturierung des Beschreibungsfeldes:

Allgemeine Beschreibung des Elementes der Faktentabelle
DFM*: Hyperlinks zu Elementen im DFM und, falls erforderlich, zusätzliche Beschreibungen*
IDS und Ermittlungslogik: *Beschreibung der Ermittlungslogik inkl. Verweise in die IDS (über Hyperlinks)*

6 Fazit

So wie in früheren Darstellungen wurde die *modellorientierte agile Vorgehensweise bei BI-/DWH-Projekten von solvistas* in einer aktualisierten Form präsentiert. Der Fokus lag dabei auf der Präsentation der Modellierung mit dem konkreten Modellierungswerkzeug Enterprise Architect von SparxSystems. Als Basis dienten verschiedene österreichische Projekte aus dem öffentlichen Bereich, die von solvistas durchgeführt wurden. Es wurde die Abbildung der Modellebenen (fachliche Modelle und logische Modelle) und deren Modellarten, wie Domänenmodelle und Dimensional Fact Models auf der fachlichen Modellebene bzw. LDS-Modelle (als System-of-Records), IDS-Modelle (in Data Vault) und DMS-Modell (im dimensionalen Star-Schema), hierarchisch im Modellierungswerkzeug in einer UML-Package-Struktur gegliedert. Für wichtige Modelelemente wurden Enterprise Architect Patterns zur Wiederverwendbarkeit definiert. Beziehungen zwischen Modellebenen (z.B. Bezug zwischen IDS- und Domänenmodell) und innerhalb einer Modellebene (z.B. Bezug zwischen IDS- und DMS-Modell) wurden durch Verlinkungen im Enterprise Architect hergestellt. Befinden sich auch die Modelle von operativen Quellsystemen können auch diese in der Verlinkung berücksichtigt werden.

7 Literatur

Hiebl J., Hörak K., Linner K., Neuböck T., Raab J., Weißenböck A. (Juli 2014). *Agile modellorientierte DWH-Entwicklung mit IBM InfoSphere.* Linz: solvistas GmbH.

Neuböck T., Pommerenke R., Raab J., Schmidt M., Weißenböck A. (März 2017). *Enterprise Business Intelligence – agile modellorientierte Entwicklung.* Linz: solvistas GmbH.

Neuböck T., Raab J., Weißenböck A. (Dezember 2014). *Agile modellorientierte DWH-Entwicklung – Modellebenen und Architektur.* Linz: solvistas GmbH.

Neuböck T., Raab J., Weißenböck A. (März 2014). *Eine modellgetriebene Vorgehensweise in der Data Warehouse Entwicklung.* Linz: solvistas GmbH.