Data Lineage goes Traceability - oder was Requirements Engineering von Business Intelligence lernen kann

Andreas Ditze

MID GmbH
Kressengartenstraße 10
90402 Nürnberg
a.ditze@mid.de

Abstract: Data Lineage ist als Konzept in Business Intelligence Systemen unabdingbar, um z.B. die Verwendung bzw. Verfolgung von Attributen über die verschiedenen Ebenen einer Data Warehouse Infrastruktur zu ermöglichen.

Ausgehend von den Attributen des fachlichen Datenmodells über die Spalten des physischen Datenbankmodells der operativen Systeme werden die Informationen über die Data Warehouse Schicht bis zur Data Mart Schicht per Data Lineage verfolgt.

Diese Konzepte, Mechanismen und entsprechende Werkzeugunterstützung lassen sich nahezu 1:1 auf die Traceability-Anforderungen eines Requirements Engineering & Management anwenden.

Der Beitrag zeigt anhand von kleinen Praxisbeispielen die Vergleichbarkeit der beiden Konzepte auf und stellt eine entsprechende Werkzeugunterstützung vor.

Erschienen in:
GI-Edition, Lecture Notes in Informatics, Stefan Wagner, Horst Lichter (Hrsg.)
Software Engineering 2013 Workshopband 26. Februar – 1. März 2013, Aachen Proceedings
http://www.se2013.rwth-aachen.de/downloads/proceedings/SE2013WS.pdf

1 Einleitung

Traceability ist extrem hilfreich, aber auch aufwendig aufzubauen und damit teuer in der Erstellung und im Unterhalt. Deshalb müssen effektive und effiziente Methoden und Werkzeuge zum Aufbau und Pflege der für eine Traceability notwendigen Abhängigkeiten zwischen den Ergebnisartefakten aufgebaut und genutzt werden. Dabei ist es besonders wichtig, dass die Abhängigkeiten möglichst automatisch während des Entwicklungsprozesses erstellt bzw. abgeleitet und gepflegt werden. Dieser Beitrag wird anhand einer modellbasierten Vorgehensweise und einer Anlehnung an das Konzept des Data Lineage aus dem Business Intelligence Umfeld aufzeigen, wie in der Praxis und unter Verwendung eines entsprechenden Modellierungswerkzeugs Traceability genutzt wird.
Data Lineage (oder auch Datenherkunft) ist sinngemäß die Fragestellung, zu gegebenen (meist) aggregierten Datensätzen die ursprünglichen Datensätze zu bestimmen, aus denen sie entstanden sind.
Für Ergebnisartefakte in Softwareentwicklungsprozessen gilt ähnliches. Auch hier möchte man wissen, aufgrund welcher Anforderung eine bestimmte Softwarekomponente oder –funktion entstanden ist, bzw. welcher Testfall sich auf welche Anforderung von welchem Stakeholder bezieht. Warum also nicht die gleichen Mechanismen und Techniken dafür einsetzen?

2 Data Lineage als Grundprinzip

Zur Verfolgung der Datensätze über die verschiedenen Ebenen einer Data-Warehouse-Architektur werden die jeweiligen Attribute auf den unterschiedlichen Ebenen mit Abhängigkeiten verbunden. Zusätzlich kann die Abhängigkeit noch mit einer Transformationsregel erweitert werden. Mit einem einfachen Abhängigkeitsgraphen kann so die Datenherkunft jedes einzelnen Attributs eines Datensatzes bestimmt werden.

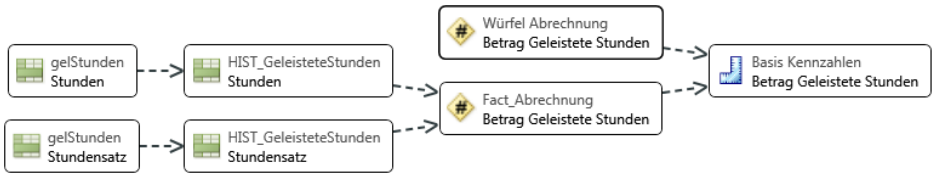

Abbildung 1: Data Lineage für die Basis-Kennzahl "Betrag Geleistete Stunden"

Die Abbildung 1 zeigt die Verfolgung der Zusammensetzung und Herkunft einer Basis-Kennzahl „Betrag Geleistete Stunden“. Ausgehend von den links dargestellten Attributen „Stunden“ und „Stundenstatz“ aus der Ladetabelle „gelStunden“ über die historisierten Tabellen der Data-Warehouse-Schicht bis zur Faktentabelle und der Kennzahl aus der Data-Mart-Schicht.

Erschienen in:
GI-Edition, Lecture Notes in Informatics, Stefan Wagner, Horst Lichter (Hrsg.)
Software Engineering 2013 Workshopband 26. Februar – 1. März 2013, Aachen Proceedings
http://www.se2013.rwth-aachen.de/downloads/proceedings/SE2013WS.pdf

Dieses Grundprinzip lässt sich auch in der Softwareentwicklung anwenden, nur das hier die Art der Elemente vielfältiger ist. Es handelt sich nicht nur um Attribut von Entitäten, Spalten von Tabellen oder Würfeln, sondern um unterschiedlichste Instanzen der verschiedenen Metamodelle (u.a. UML 2, BPMN 2). Meist sogar noch ergänzt um textuelle Artefakte wie funktionale Anforderungen, Abnahmekriterien, Risiken und Testfälle. Dadurch ergeben sich eine Reihe von möglichen Beziehungen und Abhängigkeiten, die entweder von Hand im Rahmen der Modellierung explizit erzeugt oder implizit durch die Metamodell-Beziehungen hergestellt werden.
Zwei Beispiele für implizite Metamodell-Beziehungen sind die „Owned Element" Beziehung (Ownership) eines UML-Attributes zur UML-Klasse oder einem BPMN-Task zum BPMN-Prozess. Diese Beziehungen werden nicht durch Abhängigkeiten modelliert sondern werden im Rahmen der Modellbildung erstellt.
Beispiele für explizite Abhängigkeiten sind die manuelle Zuordnung von Anforderungen zu Anwendungsfällen, UML-Klassen zu BPMN-Datenobjekten oder Rollen bzw. Organisationseinheiten zu BPMN-Lanes bzw. BPMN-Pools.
Beispielhaft soll dies an einem Übersichtsschaubild (sog. Whiteboard-Diagramm) für ein kleines Modellierungsprojekt gezeigt werden:

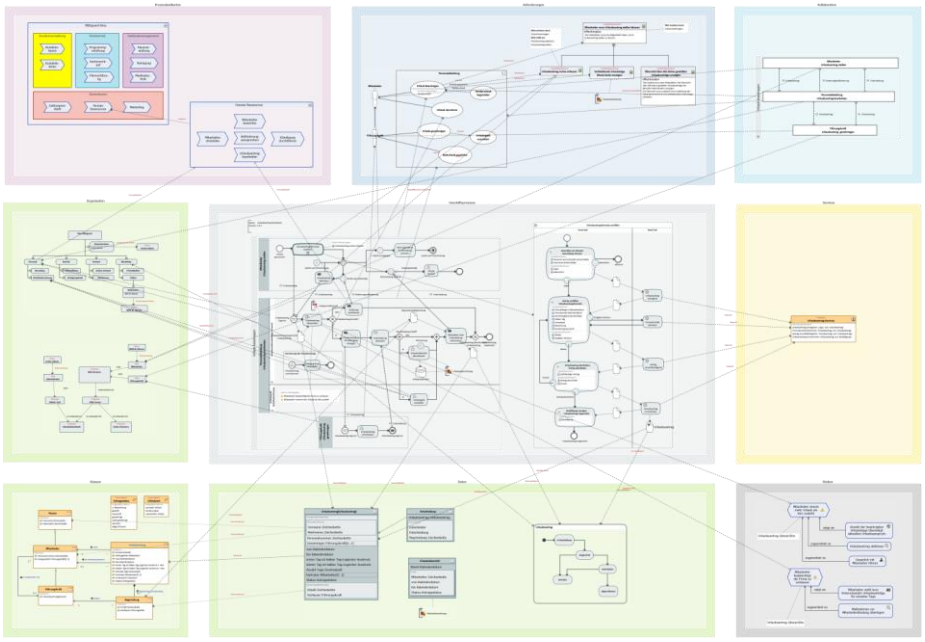

Abbildung 2: Übersichtsschaubild mit allen Abhängigkeiten zwischen den Modellelementen

Die Abbildung 2 zeigt die unterschiedlichen Modellierungsartefakte und die verschiedene Beziehungen (explizit oder implizit erstellt) untereinander. Jede Beziehung (Abhängigkeit) kann in einer Traceability-Analyse zur Verfolgung über die verschiedenen Modellierungsebenen genutzt werden.
So entsteht z.B. ein Traceability-Pfad von einer funktionalen Anforderung → UML-Anwendungsfall → BPMN-Kollaboration → BPMN-Prozess → BPMN-Task

Erschienen in:
GI-Edition, Lecture Notes in Informatics, Stefan Wagner, Horst Lichter (Hrsg.)
Software Engineering 2013 Workshopband 26. Februar – 1. März 2013, Aachen Proceedings
http://www.se2013.rwth-aachen.de/downloads/proceedings/SE2013WS.pdf

→ BPMN-Datenassoziation → BPMN-Datenobjekt → Geschäftsobjekt → UML-Klasse → ER-Entität → DB-Tabelle → Datenbank.
Über diesen Pfad kann direkt die Frage beantwortet werden, welche Datenbank von der Änderung einer Anforderung betroffen ist (in Pfeilrichtung) bzw. welcher Geschäftsprozess vom Ausfall einer Datenbank betroffen ist und welcher Stakeholder informiert werden muss (entgegen der Pfeilrichtung).
Neben der Darstellung in Abbildung 2 mit vollständigen Diagrammen in der jeweiligen Modellierungsnotation ist auch eine Darstellung als einfacher Abhängigkeitsgraph möglich. Dieser Graph stellt nur die einzelnen Modellelemente (z.B. Anforderung, Geschäftsobjekt, Klassenattribut, Entitätsattribut) mit ihren Abhängigkeiten dar und ermöglich damit eine schnelle Verfolgung der Auswirkung einer Änderung.

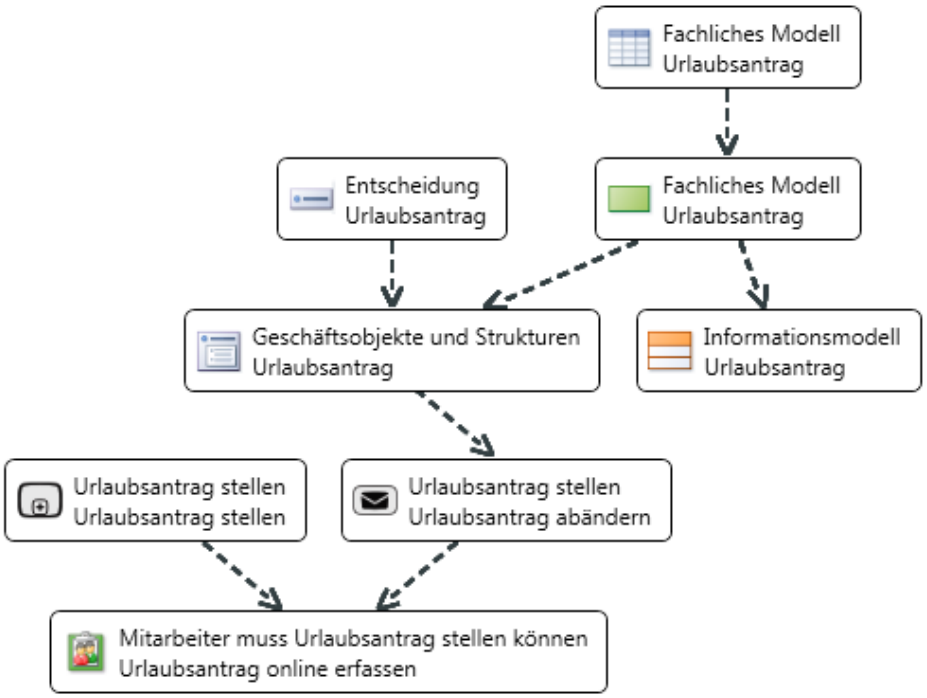

Abbildung 3: Dynamischer Abhängigkeitsgraph für beliebige Modellelemente

Mit Hilfe der in Abbildung 3 gezeigten Darstellungsmöglichkeit wird Traceability sowohl beim Erstellen der Abhängigkeiten als auch bei der Pflege und Auswertung beherrschbar. Viele der Abhängigkeiten entstehen implizit beim Modellieren oder durch einfache Drag-&-Drop-Aktionen bzw. automatisierte Modell-zu-Modell-Transformationen.
Die nachfolgende Abbildung 4 zeigt eine spezielle Applikation („Beamer“), die zur interaktiven Erstellung und Verwaltung von Abhängigkeiten zwischen unterschiedlichen Quellen und Zielen geeignet ist. Dabei können verschiedene sog. Abbildungsthemen definiert werden, bei denen die Auswahl von Quell- und Zielelementen und der dazwischen anzulegenden Abhängigkeit vordefiniert wird.

Erschienen in:
GI-Edition, Lecture Notes in Informatics, Stefan Wagner, Horst Lichter (Hrsg.)
Software Engineering 2013 Workshopband 26. Februar – 1. März 2013, Aachen Proceedings
http://www.se2013.rwth-aachen.de/downloads/proceedings/SE2013WS.pdf

Ein beispielhaftes Abbildungsthema ist das Mapping von Attributen der Entitäten des konzeptionellen Datenmodells auf Spalten von Tabellen eines physischen Datenbankmodells. Dabei werden als Quelle die Attribute und Entitäten des konzeptionellen Datenmodells angezeigt und als mögliche Ziele die bereits vorhandenen Spalten und Tabellen des physischen Datenbankmodell. Nun kann entweder eine Verbindung (Abhängigkeit) zwischen bereits vorhandenen Modellelementen hergestellt werden oder man erzeugt aus einem Quellelement ein neues Zielelement und verbindet beide gleichzeitig mit einer entsprechenden Abhängigkeit.
Vorhandene Abhängigkeiten werden durch eine grüne Verbindungslinie zwischen dem Quell- und Zielelement im oberen Bereich der Applikation angezeigt und zusätzlich in der Bearbeitungsliste (unterer Bereich) aufgelistet.

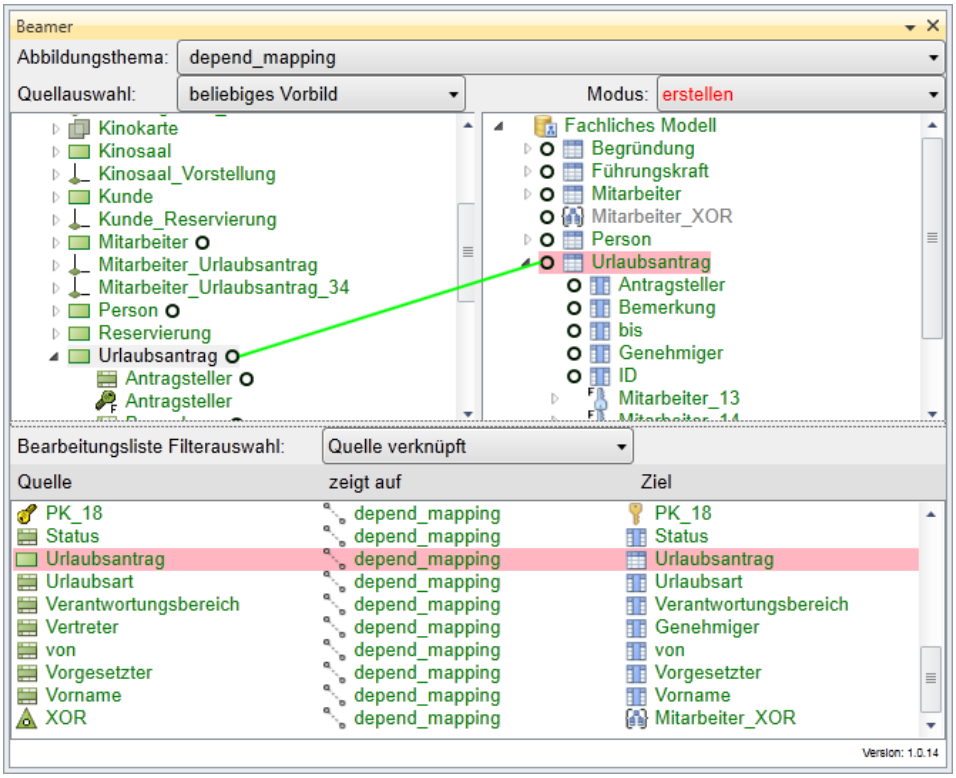

Abbildung 4: Der "Beamer" - eine interaktive Applikation für Abhängigkeiten

In der Bearbeitungsliste der „Beamer“-Applikation (unterer Bereich in Abbildung 4) wird durch eine entsprechende Einfärbung der Einträge deren Aktualität gekennzeichnet. So werden Einträge, bei denen das Modellelement der Quelle ein jüngeres Änderungsdatum hat als das Ziel-Modellelement rot eingefärbt. Dadurch können auch bei einer großen Anzahl an vorhandenen Abhängigkeiten die notwendigen Anpassungen nach einer Änderung des Quellmodells schnell ermittelt werden.

Erschienen in:
GI-Edition, Lecture Notes in Informatics, Stefan Wagner, Horst Lichter (Hrsg.)
Software Engineering 2013 Workshopband 26. Februar – 1. März 2013, Aachen Proceedings
http://www.se2013.rwth-aachen.de/downloads/proceedings/SE2013WS.pdf

Hierzu kann die Bearbeitungsliste, wie in Abbildung 5 dargestellt, auch entsprechend gefiltert werden.

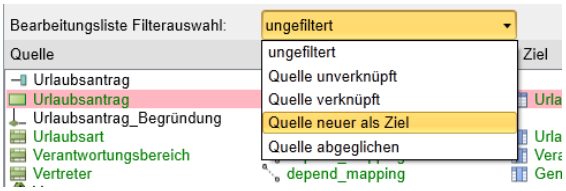

Abbildung 5: Filterauswahl der Bearbeitungsliste

Mit den im Beitrag gezeigten Beispielen sollen die verschiedenen Unterstützungsmöglichkeiten eines modernen Modellierungswerkzeugs beim Aufbau und Pflege der Abhängigkeiten zwischen Modellelementen vorgestellt werden.

Der Anwender erstellt im Rahmen des Modellierungsprozess durch verschiedenste manuelle, teilautomatische oder vollautomatische Verfahren entweder implizit oder explizit die unterschiedlichen Abhängigkeitsbeziehungen zwischen den Quell- und Zielmodellelemente. Diese Abhängigkeiten können mit den dargestellten Funktionalitäten ausgewertet und gepflegt und damit nutzbar gemacht werden.

3 Fazit

Traceability ist in einer modellbasierten Vorgehensweise beim Einsatz von modernen Modellierungswerkzeugen mit unterschiedlichen Darstellungs- und Auswertungsfunktionalitäten effizient und effektiv einsetzbar.

Quelle für alle Abbildungen: Modellierungsplattform Innovator®

Erschienen in:
GI-Edition, Lecture Notes in Informatics, Stefan Wagner, Horst Lichter (Hrsg.)
Software Engineering 2013 Workshopband 26. Februar – 1. März 2013, Aachen Proceedings
http://www.se2013.rwth-aachen.de/downloads/proceedings/SE2013WS.pdf