Voll-Integrierte DG
(Aspekt 3)

Data Governance

Thilo Riegel, München

Zusammenfassung

Data-Governance-Serie, Teil 6.3: Wir haben in uns den letzten Teilen mit verschiedenen Dimensionen der DG befasst: der Informations-Dimensions, der horizontalen, der vertikalen sowie mit dem DQ-Management. In diesem Teil sehen wir uns an, wie die geschickte Kombination diese Dimensionen zu einem effektiven und ganzheitlichen Lösungsansatz für einige der Probleme führt, die wir im ersten Teil aufgezählt haben.

Szenarien

Wenn man ein großes System (technisch und organisatorisch) aufbaut, dann hat man natürlich auch ein paar Szenarien im Kopf.[1] Szenarien, die nicht nur spezifizieren, welche Prozesse unterstützt werden sollen, sondern auch darlegen, worin der Mehrwert dessen besteht, das man aufbaut.

Hier meine „Lieblings“-Szenarien (es gibt natürlich noch mehr):

Szenario: „Forschung“ / Dokumentation über „Navigation“

Es kommt wohl häufiger vor, als man anfangs so denkt: Man geht von einem „x-beliebigen“ Element im Datenmodell aus, weil es in einer bestimmten Situation (z.B. Fehler oder auch Feature Request) sinnfällig geworden ist. Dieses wird vstl. von einer Änderung betroffen sein, und es gilt nun, herauszufinden, ob die von ihm abhängigen Elemente es ebenfalls sind – und wenn ja, wie genau.

Ohne Repository

Der Data Governor beauftragt ein „Forschungs-Team“, die Abhängigkeiten zu erforschen. Ein Analyst und ein technischer Modellierer legen los und schauen in der Doku nach. Die ist leider, wie befürchtet, hoffnungslos veraltet. Daher muss der Status Quo in mühevoller Kleinarbeit festgestellt werden. Nach einem halben Tag hat man den Ansprechpartner gefunden. Dieser arbeitet einen weiteren halben Tag im Team mit, um die Grundinformationen zu finden. Nun haben wir zumindest die nötigen statischen Informationen zum Datenmodell und zu den direkten ETL-Abhängigkeiten. Der ETL-Job ist leider auch nicht mehr so gut dokumentiert, denn daran wurde seit Go-Live einiges gemacht. Daher muss der nochmal grob re-engineert werden. Das dauert nochmal einen halben Tag (grob reicht in diesem Fall). Alles zusammen zu fassen und Management-tauglich aufzubereiten, dauert dann nochmal einen halben Tag. 2 Tage später stellt man fest, dass man noch etwas Wichtiges übersehen hat...

1 Oder auch „Use Cases“ oder „User Stories“ oder wie auch immer das in der Methode Ihrer Wahl heißt.

Mit Repository

Anruf bei einem Analysten. Der weiß es zwar gerade nicht auswendig, und das Element ist auch nicht in „seiner Ecke“ des Datenmodells, aber er kann es trotzdem herausfinden. 10 Minuten, um das Element durch Rückfragen nochmal genau zu spezifizieren (zur Sicherheit), und dann die Auswertung dazu machen (10 Sekunden). Diese dann nochmal auf Aktualität und Plausibilität überprüfen (30 Minuten), Management Summary schreiben (15 Minuten, denn i.W. ist die Auswertung bereits das Summary), und der Fall ist erledigt.

Szenario: Testplanung

Eine neue Fachfunktion ist beauftragt, und diese benötigt eine Erweiterung des Datenmodells. Das Design ist i.W. durch, und nun fragt man sich, wie der Test gestaltet werden soll.

Ohne Repository

Ein ganz Schlauer behauptet, die Fachfunktion habe keine Abhängigkeiten, und daher müsse sie nicht intensiv getestet werden – wenn überhaupt, dann nur für sich allein. Niemand kann das widerlegen Man definiert ein paar „typische“ (so denkt man) Eingangsdaten, und diese werden nach einem Lauf, der nicht sofort „Fehler“ schreit, auf Plausibilität überprüft (leider nicht auf mehr). Die Erweiterung geht online, und beim nächsten Monatswechsel merkt man, dass man leider ein paar Abhängigkeiten übersehen hat. Feuerwehr-Einsatz.

Mit Repository

Ein ganz Schlauer behauptet, die Fachfunktion habe keine Abhängigkeiten, und daher müsse sie nicht intensiv getestet werden – wenn überhaupt, dann nur für sich allein. Ein kurzer Blick ins Repository genügt jedoch, um das zu widerlegen. Der Data Governor besteht darauf, dass die Erweiterung nicht nur Modul- sondern auch Integrations-Tests durchläuft (was eigentlich ohnehin selbstverständlich sein sollte). Eingangsdaten für die Tests können sehr gezielt, Fall-spezifisch und Kritikalitäts-angemessen definiert werden, weil man die Abhängigkeit zu den zwei anderen „fiesen“ Fachfunktionen gleich erkannt hat, und weil man auch sofort bis ins Quellsystem rückverfolgen kann. Weil das so ähnlich abläuft wie bei der Einführung vor 2 Jahren, ist das auch gar nicht so schlimm, wie es zunächst klingt. Der Governor hat jederzeit Überblick, weil die Testfälle auch im Repository hinterlegt werden (nur per Referenz, weil fürs Test-Management ein anderes System führend ist) und er somit bereits grob im Bilde ist. Die Tests fördern noch eine weitere „Gemeinheit“ zu Tage, die noch anderthalb Tage brauchen, bis sie behoben sind. Die Auslieferung und der nächste Monatswechsel laufen reibungslos.

Szenario: Projektleitung und Controlling

Ohne Repository

Der Projektleiter führt eine Excel-Datei mit allen Fachfunktionen und Datenelementen. Diese lässt er kursieren, damit jeder „seinen Input“ liefern kann. 3 Wochen, 50 e-Mails und 200 Kopien später weiß niemand mehr, was die Master-Datei ist – es gibt schätzungsweise 5 „relevante“ Dateien, die alle „ein bisschen“ aktuell sind. Außerdem kann das sowieso niemand pflegen, weil die Liste erstens viel zu umfangreich und detailliert ist, und weil zweitens bei den meisten Einträgen gar nicht klar ist, was eigentlich genau damit gemeint ist, von wem das kommt, in welchem Zusammenhang das steht usw.

Der Projektleiter braucht aber Auskunft über Status. So läuft er jede Woche von Modul-Verantwortlichem zu Modul-Verantwortlichem und lässt ihn sich geben. Leider sprechen die Leute alle eine

andere Sprache… Weitere 3 Wochen später stellt er fest, dass zwei Teams an derselben Sache arbeiten, und dass eine andere liegen geblieben ist.

Mit Repository

Der Projektleiter lässt alle Fachfunktionen und Datenlemente ins Repository eintragen. Das ist zunächst mal nur ein bisschen „schicker“ als die Excel-Datei (und somit zunächst auch *scheinbar* genauso wenig zielführend), aber es ist nur ein Datenbestand, und der ist zentral, jeder hat lesenden Zugriff, und der schreibende Zugriff wird sinnvoll beschränkt. Bereits beim Eintragen werden Entitäten, höhere Fachfunktionen und weitere Abhängigkeiten mit erfasst, womit sich langsam, aber sicher der Nebel lichtet. Jeden Tag, jede Woche entsteht mehr Struktur. Durch High-Level-Auswertungen und Data Lineage werden die größeren Zusammenhänge klar gemacht. Die Detail-Zusammenhänge werden für die Analysten auf der BDE-, und für die Entwickler auf der Ebene der technischen Felder dargestellt. Für die nötigen Konsistenz-Abgleiche und die Suche nach Lücken gibt es eigene Check-Funktionen im Repository: Binnen Sekunden weiß man jederzeit, ob das Modell vollständig und konsistent ist, und wo es noch hakt. Im Lauf der nächsten 2 Wochen wird das ganze Modell in mehreren Iterationen aus mehreren Richtungen und Perspektiven kommend (top-down, bottom-up, links-rechts und rechts-links) immer wieder „durchkämmt“. Ein nicht unerheblicher Teil davon läuft halb-automatisch.

Jede Woche zieht sich der Projektleiter einen Bericht und Statistiken aus dem System. In Woche 4 sieht er, dass der Status weiter ist als ihm ein Modul-Verantwortlicher in der Kantine gesagt hatte. Er fragt nach, und der Modul-Verantwortliche sagt: „Nanu? Tatsächlich, das letzte Mal hatte ich gestern Abend ins System gesehen...“

Szenario: Datenanforderungen verwalten

Ohne Repository

Der Projektleiter muss die Anforderungen der unterschiedlichen Fachbereiche koordinieren, die künftig mit Berichten und Datenschnittstellen beliefert werden sollen. Er bittet jeden Bereich, auf Basis des Fachkonzepts die Datenanforderungen in einer Liste zu übergeben.

- Bereich 1 liefert eine Liste von technischen Feldern im Quellsystem (!) A. Unkommentiert, denn die Feldnamen sind „selbsterklärend“.
- Bereich 2 liefert eine Liste von Feldern in ihrem Mart – so wie sie ihn sich vorstellen. Einigermaßen gut kommentiert und auf so etwas ähnlichem wie der BDE-Ebene, aber sie sagen nicht dazu (können sie auch gar nicht), woher es letztlich kommen soll. V.a. über das Feld xyz, welches sich aus anderen berechnet, sagen sie wenig...
- Bereich 3 zeigt ins High-Level-Fachkonzept und sagt: „alles, was diese Fachfunktion liefert“ und meint, damit seine Pflicht mehr als erfüllt zu haben.

Der Projektleiter hat zwei Wutanfälle unterdrückt und kratzt sich am Kopf. Irgendwie wird’s schon gehen. Er sammelt die Listen ein und stellt sie aufs Laufwerk. Leider kursieren noch jeweils 12 Kopien und Varianten der Listen… Und leider kann man kaum / nur in einzelnen Fällen, aber nicht systematisch, feststellen, welche Anforderungen doppelt und dreifach gestellt wurden, und es sind auch vermutlich zu viele, um sie termingerecht liefern zu können. Anfragen bei den Fachbereichen, ob denn wirklich alles sein müsse, werden empört zurückgewiesen. Ein paar „interessante“ Fälle gibt es auch: Zwei Anforderungen, die inhaltlich auffällig ähnlich aussehen, werden angeblich, lt. Einschätzung der beiden zuständigen Fachbereichs-Mitarbeiter, auf unterschiedlichen Wegen / aus unterschiedlichen Quellfeldern befüllt. Der Projektleiter fragt sich, woher die das eigentlich bereits jetzt, in einem solch

frühen Stadium, so genau zu wissen glauben… Nach einem 3-Stunden-Workshop ist dieser Punkt geklärt. Für *eine* Datenanforderung.

Mit Repository

Der Projektleiter verweist auf die Eingabefunktion für Datenanforderungen.

Alle Datenanforderungen müssen letztlich, so fordert es der Workflow, in 3 Ebenen vorliegen: High-Level (Fachfunktion, Entität), Mid-Level (BDE) und Low-Level (technisch). Ob jemand von unten nach oben geht oder von oben nach unten, bleibt im Prinzip ihm/ihr überlassen, aber die 3 Ebenen müssen ausgefüllt sein. Empfohlen wird ein Top-Down-Vorgehen. Ferner müssen alle Anforderungen „rechts“ im Modell aufgehängt sein, also bei den Marts / Export-Schnittstellen.

Es gibt zwar Anlaufschwierigkeiten, weil es das erste Mal ist, dass die Fachbereiche gezwungen werden, in einem solch formalisierten Verfahren und auf einheitlicher Fachebene Datenanforderungen zu stellen, aber nach 2 Wochen hat es sich einigermaßen eingespielt, und weitere 2 Wochen später ist die anfängliche Skepsis verflogen – die meisten kommen sogar ziemlich gut damit zurecht.

Aus welchen Feldern / über welche ETL-Strecken diese angeforderten Felder dann befüllt werden, ist wieder eine andere Frage, denn das wird im Design-Prozess später festgelegt. Aber zuvor kommt noch eine Auswahl- und „Bereinigungs“-Phase: Erstens kommen natürlich zu viele Wünsche, die man nicht alle auf einmal erfüllen kann. U.a. aufgrund der Abhängigkeits-Auswertungen kommt man zu einer vernünftigen Einschätzung, was wirklich nötig ist. Ein paar Mini-Workshops von je einer halben bis ganzen Stunde klären die wichtigsten Inkonsistenzen. Agiles Arbeiten wird mit dem Repository enorm vereinfacht. Offensichtliche und auch nicht so offensichtliche Dubletten bei den Anforderungen werden über eine entspr. Funktion im Repository zu einer einzigen Anforderung zusammengefasst.

Schöne neue Welt?

Male ich im vorigen Abschnitt die „Welt ohne Repository“ schwarz und „die Welt mit Repository“ rosarot? Ja, ein bisschen schon, natürlich, zugegeben…

Aber ich traue mich trotzdem, mich hinzustellen und zu behaupten: So weit weg von der Realität ist das alles nicht! In mindestens 4 Vorhaben ähnlicher Größenordnung bei 4 verschiedenen Häusern konnte ich Beobachtungen der Art „ohne Repository“ machen. Und in dreien dieser Häuser konnte ich die positiven Auswirkungen des Ansatzes „mit Repository“ wenigstens grundlegend und teilweise auch in überraschender Klarheit beobachten. Ich habe nie die „Reinform“ im Einsatz erlebt, und vielleicht wird es die auch nie geben, aber ich bin überzeugt, dass wir da noch wesentlich weiter gehen könnten – und sollten. Ich spreche jedenfalls aus praktischer Erfahrung, und die Szenarien von oben basieren im Kern auf echte Erlebnissen (ein bisschen ausgeschmückt, natürlich).

Ja, ich habe natürlich ein bisschen vereinfacht und übertrieben. Aber wer das mit „wieder so ein Wunder-Tool“ oder „das gibt es so nicht!“ oder „wir brauchen pragmatische Lösungen“ wegwischen will, den möchte ich meinerseits auffordern, sich doch bitte erst einmal über die Tatsachen zu informieren und seine belastbaren (!) Argumente zu nennen!

Die Szenarien „mit Repository“ haben jedenfalls nichts mit Science-Fiction-Romantik zu tun! Es überrascht mich immer wieder, wie man in vielen Häusern an solche Themen herangeht: Zuerst sind alle ganz Feuer und Flamme fürs Vorhaben und naiver Vorfreude, und wenn es dann daran geht, es

umzusetzen (und weil es dabei natürlich auch ein paar kleinere Anlaufschwierigkeiten gibt, wie bei jeder größeren Sachen), werden plötzlich alle ganz überskeptisch und engstirnig.