Bausteine für agile und nachhaltige Business Intelligence

Ein Reifegradmodell für Agile BI

„Wir machen jetzt auch Agile BI“ – diese und ähnliche Aussagen hört man auch in der deutschsprachigen BI-Gemeinschaft immer häufiger. Doch ist Agilität in Business-Intelligence-Vorhaben wirklich so direkt machbar? Reicht es, zweiwöchige Iterationen einzuführen und die Mitarbeiter das Agile BI Memorandum [BIM] lesen zu lassen? Dieser Artikel zeigt auf, mit welchen Bausteinen und grob in welcher Reihenfolge eine Organisation sich ihren Weg zu nachhaltiger Agilität in BI-Vorhaben erarbeiten sollte.

In der Erfahrung des Autors greifen Maßnahmen wie die Einführung zweiwöchiger Iterationen und etwas Ausbildung zu agilen Prinzipien zu kurz. Sie führen häufig zu einem Anstieg technischer Schulden und hohen Änderungsaufwänden. Wie aber lässt sich Agile BI *nachhaltig* in einer Organisation einführen und etablieren? Ein möglicher Ansatz ist das vom Autor entwickelte Reifegradmodell, das Agile BI Maturity Model (ABIMM). Es ist in Abbildung 1 dargestellt. Das ABIMM identifiziert Bausteine für die Etablierung und Weiterentwicklung von Agile BI in einer Organisation. Der Fokus liegt dabei auf Situationen, in denen neben BI-Applikationen (Berichte, Dashboards etc.) immer auch die Datengrundlage, das sogenannte Data Warehouse (DWH) mitentwickelt und betrieben werden muss.

Es gibt bereits zahlreiche Reifegradmodelle im Bereich der agilen Softwareentwicklung (vgl. [Sch13] für eine Übersicht sowie [HuR09] für ein konkretes Beispiel). Es fehlt dabei aber ein Ansatz, der das Thema aus der BI-/DWH-Perspektive beleuchtet. Diese Lücke soll das ABIMM schließen.

Von der Idee her orientiert es sich an [Bel14]. In seinem Agile Engineering Fluency Model zeigt er, wie verschiedene agile Methoden und Konzepte aufeinander aufbauen und daher in einer sachlogischen Reihenfolge stehen. Inhaltlich stützt sich das Modell primär auf die Bausteine für Agile BI, zum Beispiel die Verwendung von User Stories, selbstorganisierten Teams, Testautomation oder das automatisierte Integrieren und Verteilen von BI-Lösungen [Col12]. Im Bereich Data Modeling stellt das ABIMM zudem auf den Ansatz nach [CoS11] ab, bei dem Fachanwender aktiv in die Datenmodellierung miteinbezogen werden. Die Farblegende der Bausteine orientiert sich an den Kategorien, wie sie auch in [Kra14] verwendet werden. Hier werden agile Methoden und Konzepte eingeteilt in Werte und Prinzipien, Vorgehensmodelle, Entwicklungsmethoden sowie Technologien. Das Modell wird komplementiert mit Roadmap-Aspekten, um Agilität nach und nach in kleinen Etappen zu erreichen [Hug12], und Lebenszyklus- (Englisch: Lifecycle-) Ansätzen. [AmL12] beschreiben unterschiedliche Lebenszyklen für IT-Projekte, zum Beispiel einen Zyklus auf Basis fixer Iterationen oder einen Zyklus, der auf kontinuierliche Auslieferung neuer und angepasster Funktionalitäten ausgerichtet ist.

Zielsetzung: Mehr Agilität durch weniger Voraus-Design

Das entwickelte Reifegradmodell (siehe Abbildung 1) unterscheidet vier aufeinander aufbauende Maturitätsstufen von links nach rechts. Zuoberst im Modell sind zwei gegenläufige Größen abgebildet: die Menge an notwendigem Voraus-Design (Englisch: Upfront Design) einerseits und die Agilität andererseits. Auch wenn in der agilen Gemeinschaft Voraus-Design oft verpönt ist, so hat dieses doch auch seine Vorteile: (Umfassendes) Voraus-Design ermöglicht es, das große Ganze in Ruhe zu konzipieren und durchzudenken. Damit lassen sich Abhängigkeiten frühzeitig erkennen und können in die Konzeption miteinfließen.

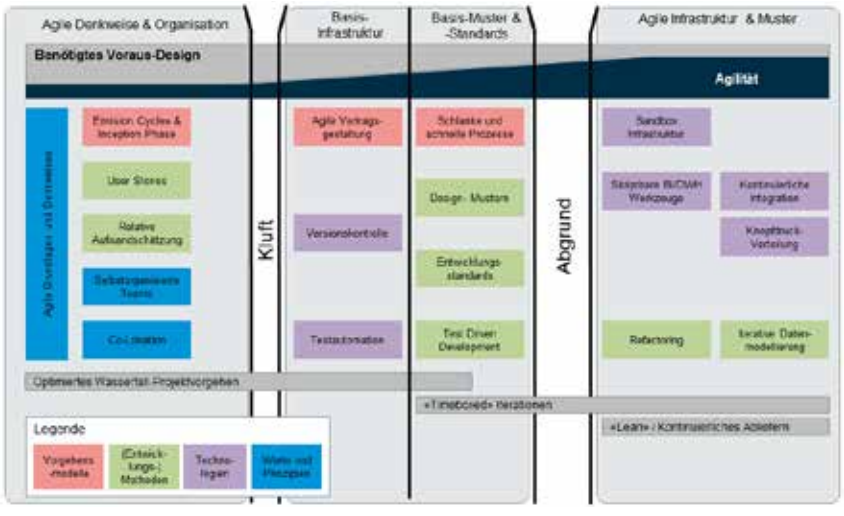

Abb. 1: Das Agile BI Maturity Model verknüpft BI-Bausteine mit Überlegungen zum Projektvorgehensmodell

Man kann es mit dem Hausbau vergleichen: Es ist schlicht günstiger, wenn ein Architekt erst das ganze Haus skizziert, bevor die Baufirma mit dem Bau des Fundaments startet. Grundsätzlich ist es auch möglich, ein bestehendes Gebäude zu erweitern (Anbau) oder gar in seiner Grundstruktur zu verändern (Umbau). Solche Vorhaben können jedoch aufwendig sein und erfordern typischerweise zusätzliches Know-how und eine erweiterte Infrastruktur (zum Beispiel andere Maschinen) verglichen mit einem

Neubau. Der Preis eines umfassenden Voraus-Designs ist jedoch eine geringere Agilität hinsichtlich sich ändernder Anforderungen.

Analog zu diesem Beispiel kann sich eine Organisation die Transformation zu Agile BI vorstellen: Es ist eine (einfachere) Sache, ein DWH mit viel Vorab-Design neu zu bauen. Es ist eine andere (anspruchsvollere) Sache, ein DWH iterativ in kleinen Inkrementen zu bauen und das Vorab-Design zu minimieren. Agilität bekommt man nicht umsonst.

Viele der nachfolgend vorgestellten Bausteine sind bei einem klassischen Projektvorgehen optional. Sie sind jedoch notwendige Voraussetzungen für das Gelingen einer inkrementellen Vorgehensweise.

Stufe Agile Denkweise & Organisation

Auf der Stufe „Agile Denkweise & Organisation" finden sich alle Bausteine, die sich ausschließlich auf Basis organisatorischer Maßnahmen (und des dafür notwendigen Willens) realisieren lassen:

- Es beginnt mit der Vermittlung von Grundlagen hinsichtlich Agile BI.
- Im Baustein „Envision Cycle & Inception Phase" geht es darum, eine Art Studienprojekt durchzuführen. In diesem Studienprojekt soll der initiale und daher nur grobe Umfang eines anstehenden BI-Vorhabens abgeschätzt und eine allfällige Finanzierung skizziert werden.
- Der Baustein „User Stories" adressiert die Anforderungsanalyse. Anstelle der detaillierten Anforderung werden (initiale) Anforderungen erst mal nur als Platzhalter in Form einer User Story erfasst. Zu einem späteren Zeitpunkt können nach Bedarf weitere Details erhoben oder direkt mit den Entwicklern besprochen werden.
- Der Baustein „Relative Aufwandschätzung" ist eng verwandt mit den User Stories. Es wird dabei versucht, die Komplexität verschiedener User Stories relativ zueinander statt in absoluten Aufwandswerten zu schätzen.
- Der Baustein „Selbstorganisierte Teams" regt an, dass sich Teams vermehrt selber organisieren und dadurch auch stärker engagiert sind. Der Projektmanager wird mehr und mehr zum Unterstützer (Englisch: Facilitator).
- Dazu gehört auch das Prinzip der „Co-Lokation". Im Idealfall arbeitet das Projektteam inkl. Personen aus den beauftragenden Fachbereichen in einem Raum zusammen. Das fördert die direkte, interpersonale Kommunikation, was ebenfalls ein Grundpfeiler agilen Denkens ist.

Kluft zur Basis-Infrastruktur

Auf der Stufe „Basis-Infrastruktur" finden grundlegende Elemente wie Testautomatisierung oder eine integrierte Versionskontrolle statt. Neben organisatorischen Aspekten geht es hier um den technischen Aufbau dieser Komponenten. Die Kluft soll dieses nicht triviale Unterfangen verdeutlichen: Häufig bestehen DWH-Landschaften ja aus einer Vielzahl von Tools unterschiedlicher Hersteller. Mit diesem Umstand müssen auch die besagten Lösungen erst einmal zurechtkommen. Auch nicht zu unterschätzen ist der Baustein „Agile Vertragsgestaltung", also die Vertragsgestaltung mit internen oder externen Dienstleistern.

Stufe Basis-Muster & -Standards

Während sich Testautomatisierung und Versionskontrolle primär durch Zukauf entsprechender Konzepte und Technologien realisieren lassen, erfordert die Stufe „Basis-Muster & -Standards" den Aufbau und die standardisierte Anwendung von Wissen, und zwar in Form von Design-Mustern sowie einheitlichen Entwicklungsstandards. Dieser Prozess ist typischerweise anspruchsvoll, nicht zuletzt weil Mitarbeiter mit entsprechendem Wissen rar am Markt sind. Parallel dazu lassen sich Prinzipien wie Test-Driven Development (vgl. [AmL12]) einführen.

Abgrund vor der Stufe Agile Infrastruktur & Muster

Die bisher eingeführten Bausteine sind einer größeren Agilität primär auf der Ebene Professionalisierung und Standardisierung zuträglich. Die nächste Reifegradstufe fokussiert auf die Automatisierung vor allem des Verteilungsprozesses (Englisch: Deployment), das heißt des Prozesses, wie Lösungen vom Entwickler in die Produktionsumgebung gelangen. Knackpunkt ist der Baustein „Kontinuierliche Integration", um Änderungen von parallel arbeitenden Entwicklern (automatisiert) in das Gesamtsystem zu integrieren. Spätestens an dieser Stelle muss eine Organisation das verwendete BI-/DWH-Toolset aus der Perspektive beurteilen, ob bzw. wie gut sich Verteilungs- und andere Aufgaben skripten und damit automatisieren lassen. Zudem müssen Versionskontrolle, Verteilung und Testautomatisierung über alle Technologie-Ebenen hinweg miteinander verknüpft werden. Erst dann kann man tatsächlich von einer Druckknopf-Verteilung (Englisch: Push-Button Deployment) sprechen.

Auf Ebene der Daten- und ETL-Modellierung muss sich ein Team zudem überlegen, wie es nachträglich kleine Änderungen möglichst automatisiert über verschiedene Ebenen hinweg umsetzen kann (Stichwort „Refactoring", vgl. dazu auch [AmS06]). Erst wenn eine Organisation diese Herausforderungen adressiert hat, ist auch eine iterative Datenmodellierung, wie sie [Col12] beschreibt, überhaupt nachhaltig möglich. Der „Abgrund" vor dieser Stufe soll dabei ausdrücken, dass die Erreichung dieses Reifegrades einen gewaltigen Komplexitätssprung gegenüber dem vorherigen Reifegrad darstellt.

Projektführungsansätze im Kontext der Reifegradstufen

Wie bereits in der Einleitung angesprochen, darf ein iteratives Projektvorgehen nicht zu einem Aufbau technischer Schulden führen. Der Autor hat sich deswegen vor allem mit der Frage beschäftigt, welche Grundvoraussetzungen erfüllt sein müssen, damit ein solches iteratives Vorgehen auch nachhaltig ist. Der Balken „Optimiertes Wasserfall-Projektvorgehen" in der Modelldarstellung repräsentiert die Hypothese, dass eine Organisation auf den ersten zwei Reifegradstufen noch auf die Einführung eines iterativen Vorgehens verzichten sollte. Die Chancen für einen Misserfolg sind hoch, weil die (noch) ungenügende Automatisierung

und der fehlende Einsatz von Design-Mustern die Qualität des Endprodukts beeinträchtigt. Folglich ist ein umfassendes Voraus-Design hier das kleinere Übel.

Unabhängig davon lassen sich dennoch einige Aspekte aus dem agilen Gedankengut in bestehende, wasserfallartige Vorgehen übernehmen. Ein solches Schlüsselelement ist zum Beispiel die aus Scrum bekannte Rolle des „Product Owner“: Als vollwertiges Mitglied des Projektteams übernimmt ein Vertreter der Auftraggeberseite die Pflicht, Anforderungen abschließend zu spezifizieren und bei Kapazitätsengpässen zu priorisieren.

„Timeboxed“ Iterationen entsprechen einem Vorgehen, bei dem neue oder angepasste Funktionalitäten in Iterationen gleicher Länge entwickelt werden. Voraus-Design (inklusive der Skizzierung initialer Anforderungen) wird so viel wie nötig, aber stets so wenig wie möglich in ein oder ein paar wenigen, einleitenden Iterationen betrieben (vgl. dazu „Inception-Phase“ in [AmL12]). Anschließend folgt eine beliebige Anzahl Konstruktions-Iterationen.

Zu Beginn jeder dieser Iterationen werden die zu entwickelnden Aspekte im Team bestimmt. Zusammen mit dem oben erwähnten Product Owner erarbeiten die Entwickler ein gemeinsames Verständnis der Anforderungen. Danach werden diese umgesetzt. Im Anschluss an die Konstruktions-Iterationen folgen einige abschließende Transitions-Iterationen. Dabei geht es darum, die erarbeiteten Artefakte sauber in den produktiven Betrieb zu überführen, das heißt, neue Features werden typischerweise nur jede dritte bis vierte Iteration für den produktiven Gebrauch freigegeben. Ein Verteilen neuer Funktionalität kann daher auch nur halbautomatisch oder gar manuell abgewickelt werden.

Befindet sich eine Organisation reifegradmäßig auf dem höchsten Stand, sind auch weitere Vorgehen denkbar, die [AmL12] mit „Lean“ und „Kontinuierlich“ (Englisch: Continuous) beschreiben. Wesentlichster Unterschied zum vorgängig beschriebenen Vorgehen ist der Wegfall der regelmäßigen Iterationsdauer. Im Falle von „Lean“ hat die Entwicklung neuer Funktionalitäten weiterhin Projektcharakter. Ein Minimum an Voraus-Design geschieht weiterhin in der „Inception“-Phase, die Konstruktionsphase gestaltet sich jedoch losgelöst von fixen Iterationen.

Ein Team muss dabei seinen eigenen, individuellen und auf die organisatorischen Rahmenbedingungen abgestimmten Rhythmus finden. Das ist erfahrungsgemäß anspruchsvoller, als mit fixen Iterationen zu arbeiten, bietet aber auch mehr Flexibilität: Neue Artefakte werden dann in die Produktionsumgebung übergeben, „wenn es sinnvoll“ ist, und nicht, wenn ein Iterationsplan es vorschreibt. Um dabei schneller auf die Bedürfnisse der Auftraggeberseite reagieren zu können, wird auch ein höherer Automationsgrad bei der Verteilung und in Bezug auf die Testung benötigt.

Im Falle eines kontinuierlichen Entwicklungsvorgehens verschwinden die „Inception“- und Transitions-Phasen mehr und mehr und machen einem kontinuierlichen Entwicklungs- und Verteilungsstrom Platz. Neue und geänderte Funktionalitäten werden fortlaufend entwickelt, getestet und in die Produktionsumgebung übergeben. Qualitätssichernde Maßnahmen müssen dabei zu größtmöglichen Teilen automatisiert sein, damit das System nachhaltig entwickelt und betrieben werden kann.

Fazit

Agilität in BI-Vorhaben ist nicht kostenlos, sondern bedingt eine hohe Professionalisierung, Standardisierung und Automatisierung in möglichst vielen Entwicklungsschritten. Folglich ist es nicht ratsam, Agile BI mit der Einführung eines iterativen Projektvorgehensmodells zu starten. Im Gegenteil, eine Organisation tut gut daran, zuerst die organisatorischen, fachlichen und technischen Grundlagen für ein iteratives, inkrementelles Vorgehen zu schaffen. Das ABIMM identifiziert die dafür notwendigen Bausteine und ordnet diese anhand sachlogischer Abhängigkeiten.

[Literatur]

[AmL12] Ambler, S. W. / Lines, M.: Disciplined Agile Delivery: A Practitioner’s Guide to Agile Software Delivery in the Enterprise. IBM Press 2012

[AmS06] Ambler, S. W. / Sadalage, P. J.: Refactoring Databases: Evolutionary Database Design. Addison-Wesley Professional 2006

[Bel14] Belshee, A.: Agile Engineering Fluency. © 2014, http://arlobelshee.github.io/AgileEngineeringFluency/Stages_of_practice_map.html, abgerufen am 4.11.2015

[BIM] Memorandum für Agile Business Intelligence: www.tdwi.eu/wissen/agile-bi/memorandum/, abgerufen am 4.11.2015

[Col12] Collier, K.: Agile Analytics. Addison-Wesley 2012

[CoS11] Corr, L. / Stagnitto, J.: Agile Data Warehouse Design: Collaborative Dimensional Modeling, from Whiteboard to Star Schema. DecisionOne Press 2011

[Hug12] Hughes, R.: Agile Data Warehousing Project Management: Business Intelligence Systems Using Scrum. Morgan Kaufmann 2012

[HuR09] Humble, J. / Russell, R.: The Agile Maturity Model – Applied to Building and Releasing Software. 2009, http://info.thoughtworks.com/rs/thoughtworks2/images/agile_maturity_model.pdf, abgerufen am 4.11.2015

[Kra14] Krawatzeck, R. et al.: Agile BI ist in der Praxis angekommen. In: BI-SPEKTRUM 04/2014, S. 8–12

[Sch13] Schweigert, T. et al.: Agile maturity model: analysing agile maturity characteristics from the SPICE perspective. In: Journal of Software: Evolution and Process, 2013, online unter www.sqs.com/de/_download/agile_maturity_wiley_2013_final.pdf, abgerufen am 4.11.2015

Raphael Branger studierte Wirtschaftsinformatik und hält einen MA in Information Management. Heute arbeitet er als Chief Knowledge Officer und Senior Solution Architect bei der IT-Logix AG. Er hat über 12 Jahre Praxiserfahrung im Business-Intelligence- und Data-Warehousing-Umfeld. Sein aktueller Fokus liegt im Bereich der BI-spezifischen Anforderungsanalyse sowie der Adaption agiler Methoden im Umfeld von BI. **E-Mail: rbranger@it-logix.ch**