# ALM-Datenmigration

## Mit systematischem und iterativem Vorgehen zum Erfolg

Eine Datenmigration durchzuführen, kann eine Sache weniger Klicks sein oder aber Zusatzaufwand von mehreren Monaten bedeuten. Es ist nicht leicht, herauszufinden, wo in diesem Zeitspektrum man sich befindet. Möchte man ALM-Daten migrieren, ist eine hohe Komplexität der Aufgabe recht wahrscheinlich. Der Artikel benennt die Herausforderungen dabei und stellt ein iteratives Vorgehensmodell vor, um das Datenmigrationsprojekt zum Erfolg zu bringen.

Jeder Hersteller betreibt ALM in irgendeiner Form. Wenn sein Produkt – und sei es nur teilweise – aus Software besteht, befasst er sich freiwillig oder unfreiwillig, bewusst oder unbewusst mit ALM.
Unter *Application Lifecycle Management (ALM)* versteht man die Überwachung und Verwaltung von Software über ihren gesamten Lebenszyklus, also von der Planung über die Entwicklung bis zur Wartung, Support und End-of-Life. ALM umfasst typischerweise die Domänen Requirements-Management, Testmanagement, Architektur-Management, Change- und Configuration-Management sowie Projektmanagement. Aber auch weitere Domänen wie Variantenmanagement oder Release-Management und DevOps können dazugehören.
Dabei werden in der Regel mehr oder weniger geeignete Tools zur Unterstützung des ALM verwendet. Hierbei reicht das Spektrum vom Einsatz generischer Tools (z. B. Excel), über spezialisierte Tools (z. B. Jira) bis hin zu integrierten Tools (z. B. IBM Collaborative Lifecycle Management). Auch in der Ausbaustufe des ALM-Toolings gibt es große Unterschiede: Existiert eine Traceability zwischen den Requirements? Gibt es ein Variantenmanagement? Ein Konfigurationsmanagement?

### Trend zu integrierten ALM-Tools

Branchen wie Medical und Automotive oder global agierende Unternehmen setzen bereits seit Längerem auf ein umfassendes, durch spezialisierte Tools gestütztes ALM. Inzwischen geht der Trend aber auch in vielen anderen Bereichen in Richtung eines systematischeren ALM. Für den klassischen Maschinenbauer mag bisher eine Excel-„Tapete“ ausreichend erscheinen. In Zeiten der Digitalisierung und dem damit einhergehenden massiv steigenden Anteil von Software in der Wertschöpfungskette reicht das nicht mehr aus. Auch die zunehmenden Anforderungen an Softwarequalität, Time-to-Market und Variantenvielfalt fordern von vielen Herstellern eine Verbesserung ihres ALM. Gleichzeitig ist eine Entwicklung weg von fragmentierten und generischen Tool-Landschaften hin zu integrierten ALM-Tools zu beobachten.
Aufgrund dessen stehen immer mehr Firmen vor der Herausforderung, ein neues ALM-Tool beziehungsweise eine Tool-Landschaft einzuführen. Daraus kann ein Projekt mit enormem Aufwand entstehen. Weite Teile der Belegschaft und IT-Landschaft einer Firma sind betroffen und werden vor allem auch beansprucht. Allein einzelne dafür notwendige Teilprojekte, wie etwa die Tool-Evaluierung, das Aufsetzen der Infrastruktur oder die Schulung von Mitarbeitern, sind oft komplex und aufwendig.

### Migration von Daten

Die Datenmigration kann ein eigenes Teilprojekt darstellen, wenn bereits umfangreiches ALM-Tooling vorhanden ist und die bestehenden Projekte in einem neuen Tooling weitergeführt werden sollen. Betrachtet man die Stacey-Matrix (siehe **Abbildung 1** und [Method]), wird das Teilprojekt der Datenmigration auf beiden Achsen vermutlich als schlimmstenfalls mittelmäßig schwer bewertet.
Dadurch wird das Gesamtprojekt im ungünstigsten Fall als kompliziert beurteilt und dessen Aufwand fälschlicherweise unterschätzt. Denn sowohl der Bekanntheitsgrad der Technologie als auch der

- *Migrationskonzept:* Liste der zu migrierenden Artefakte und Attribute inkl. deren Mapping
- *Liste der Entscheidungen:* Dokumentation von Entscheidungen (Warum ist das Migrationskonzept so, wie es ist?)
- *Liste offener Punkte (LoP):* Entscheidungen bzgl. des Migrationskonzepts, die noch ausstehen
- *Abnahmekriterien:* Kriterien, die erfüllt sein müssen, damit die Datenmigration als erfolgreich und abgeschlossen erklärt werden kann
- *Backlog:* Priorisierte Liste von Anforderungen an das Migrationsskript, die implementiert werden müssen
- *Demo-Projekt:* Kleines Projekt, mit dem die Funktionsweise des Migrationsskripts demonstriert werden kann
- *Sprint:* Ein Zeitraum von zwei Wochen, in dem die zu Sprint-Beginn festgelegten Backlog-Items umgesetzt werden
- *Sprint-Demo:* Hier wird alle zwei Wochen der aktuelle Stand des Migrationsskripts gezeigt (anhand des Demo-Projektes)
- *Fachlicher Workshop:* Hier werden die Workflows im alten System analysiert und die Abbildung im neuen System beschlossen
- *Key-User:* Erfahrener Anwender des alten ALM-Tools
- *Product Owner (PO):* Verantwortlich für die technische Umsetzung der Datenmigration

*Kasten 1: Glossar zum Vorgehensmodell bei der Datenmigration*

aller Anforderungen kann sich in der Realität sehr schnell als deutlich schlechter erweisen. Dadurch kann das Projekt rasch komplexe oder gar chaotische Züge annehmen. Warum ist das so? Wie beherrscht man diese Komplexität?

## Herausforderungen und Risiken

**Ursache: Stakeholder**

Bei der Migration von Daten begegnet man verschiedenen Herausforderungen und Risiken. Diese Herausforderungen sind häufig eng mit den Stakeholdern verknüpft. Zu den Stakeholdern gehören die User, insbesondere Key-User, aus verschiedenen Domänen, IT-, Budget- und Prozess-Verantwortliche, Tool-Experten, Berater und Projektleiter.

Ein Teil der Komplexität entsteht beispielsweise allein durch die hohe Anzahl an Stakeholdern. Je mehr Stakeholder eingebunden sind, desto größer ist der Kommunikationsaufwand im Projekt. Denn die Anforderungen an die Datenmigration lassen sich nur umfassend definieren, indem sich alle Stakeholder über Inhalte und Umfang abstimmen. Unter Umständen unterscheiden oder widersprechen sich die Anforderungen verschiedener Unternehmensbereiche an das Tool.

Auch die Verfügbarkeit von Stakeholdern, beispielsweise von Key-Usern oder Tool-Experten, kann sich negativ auf den Projektverlauf auswirken. Das ist dann der Fall, wenn mangels Verfügbarkeit Entscheidungen unnötig hinausgezögert werden.

Mitunter müssen Key-User erst von einem neuen Tool überzeugt werden. Gegebenenfalls haben sie Vorbehalte, falls sie sich nicht ausreichend über die Bedienung und Möglichkeiten des einzuführenden Tools informiert fühlen.

**Ursache: Technik**

Aber auch diverse technische Risiken gefährden das Projekt. Viele Details rücken erst im Verlauf des Projektes in den Mittelpunkt und offenbaren neue oder falsche Anforderungen (sog. „moving targets“).

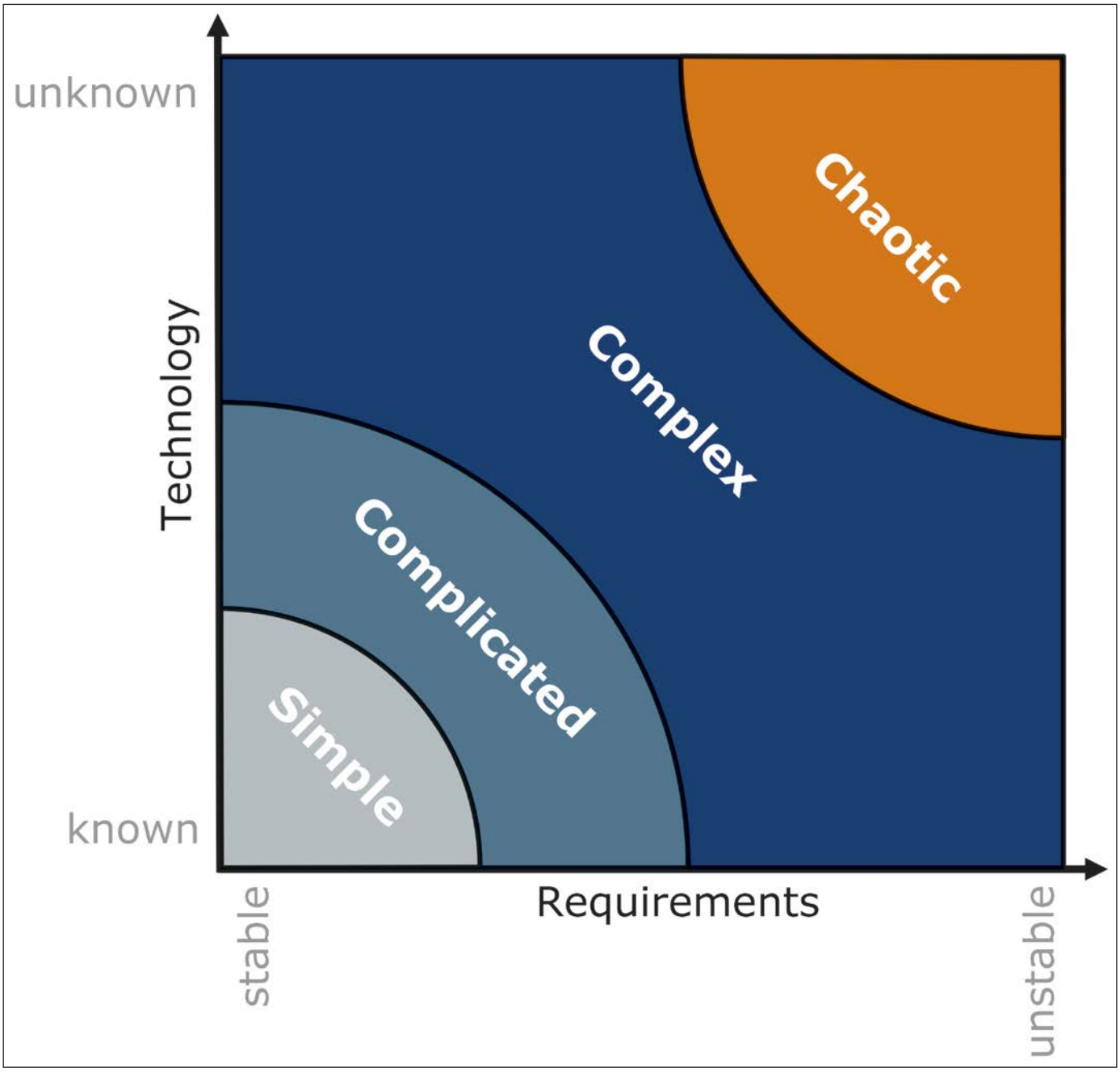


*Abb. 1: Klassifikation von Aufgaben mithilfe der Stacey-Matrix, © Method Park*

Beispielsweise können Bugs oder fehlende Features im Application Programming Interface (API) des neuen Tools eine Anpassung des Konzepts oder zumindest eine Projektverzögerung erzwingen. Ein Beispiel ist die Verwendung des Attributs „Created by“, um Zuständigkeiten abzubilden. Lässt sich dieses nicht frei über das API des ALM-Tools setzen, muss ein neues Attribut eingeführt oder direkt in die Datenbank geschrieben werden. Allgemein kann das Schreiben von User-Attributen technisch aufwendig oder sogar unmöglich sein.

Ebenfalls nicht zu unterschätzen: die Abbildung des Datenmodells des bestehenden Tools auf das Modell des neuen Tools. Die Abbildung ist in der Regel nicht trivial. Sie ist zu Projektbeginn kaum vollständig definiert und ihre Auswirkung auf die Umsetzung der Migration schwer abzuschätzen. Das gilt insbesondere, wenn die Projektmitarbeiter keine ausreichende Erfahrung mit dem neuen Tool haben.

Lässt sich die Formatierung von Freitextfeldern im neuen Tool eins zu eins abbilden? Müssen mehrere verschiedene Datenmodelle zu einem konsolidiert werden, weil ein Tool, wie etwa DOORS Next Generation, das Datenmodell auf einer höheren Ebene definiert als beispielsweise DOORS 9? Lassen sich OLE-Objekte inline oder nur als Anhang abbilden? Kann das neue Tool Funktionen nativ abbilden, die im alten Tool mittels Attributen umgesetzt wurden, etwa Varianten und Versionen? Haben Kommentare in beiden Tools dieselben Metadaten?

Die Verknüpfung der Daten untereinander hat einen erheblichen Einfluss auf die Migrationsstrategie. Gibt es viele Verknüpfungen zwischen Artefakten, lassen sich diese vermutlich nicht segmentieren. Damit ist eine Big-Bang-Migration unvermeidlich. Das ist zum Beispiel dann der Fall, wenn Artefakte aus einem Projekt in einem anderen wiederverwendet werden.

Gibt es im alten Tooling ein umfangreiches Customizing, vor allem durch Skripte, kann dies enorme Aufwände bei der Datenmigration verursachen. Zunächst muss das Projektteam das Customizing analysieren und bewerten. Dann muss es entscheiden, inwiefern Anpassungen am neuen Tooling nötig sind und wie dies bei der Datenmigration zu berücksichtigen ist. Custom-Skripte, die für das neue Tool erstellt werden, müssen auch langfristig gewartet werden. Hier kann das Unternehmen notwendiges Know-how entweder im eigenen Haus aufbauen oder das Scripting inklusive Wartungsauftrag an eine externe Firma übergeben.

**Ursache: Projektumfeld**

Abhängigkeiten zu Nachbarprojekten erzeugen zusätzliche Komplexität. Beispielsweise kann ein Probe-Import in das neue ALM-Tool nur erfolgen, wenn die Infrastruktur, als Ergebnis des „Tool Infrastructure Deployment“-Projektes (mit dem Ziel der initialen Bereitstellung des Tools), dafür schon bereitsteht.

Im Rahmen der ALM-Einführung passt das Projektteam oft auch Engineering-Prozesse an oder erstellt diese neu, etwa bei der Adaption von Automotive SPICE®. Dann ist die Konfiguration des neuen Tools und damit die Abbildung der zu migrierenden Daten abhängig von Entscheidungen, die das Team für diese Prozesse fällt. Daher ist ein Management der Abhängigkeiten im übergeordneten Gesamtprojekt unbedingt nötig.

Durch die vielen Abhängigkeiten des Migrationsprojektes können einzelne offene Entscheidungen zu weitreichenden Verzögerungen führen. Müssen Testergebnisse migriert werden? Wie werden Versionen abgebildet? Erst wenn solche und ähnliche Fragen geklärt sind, kann das Team an anderer Stelle im Projekt weiterarbei-

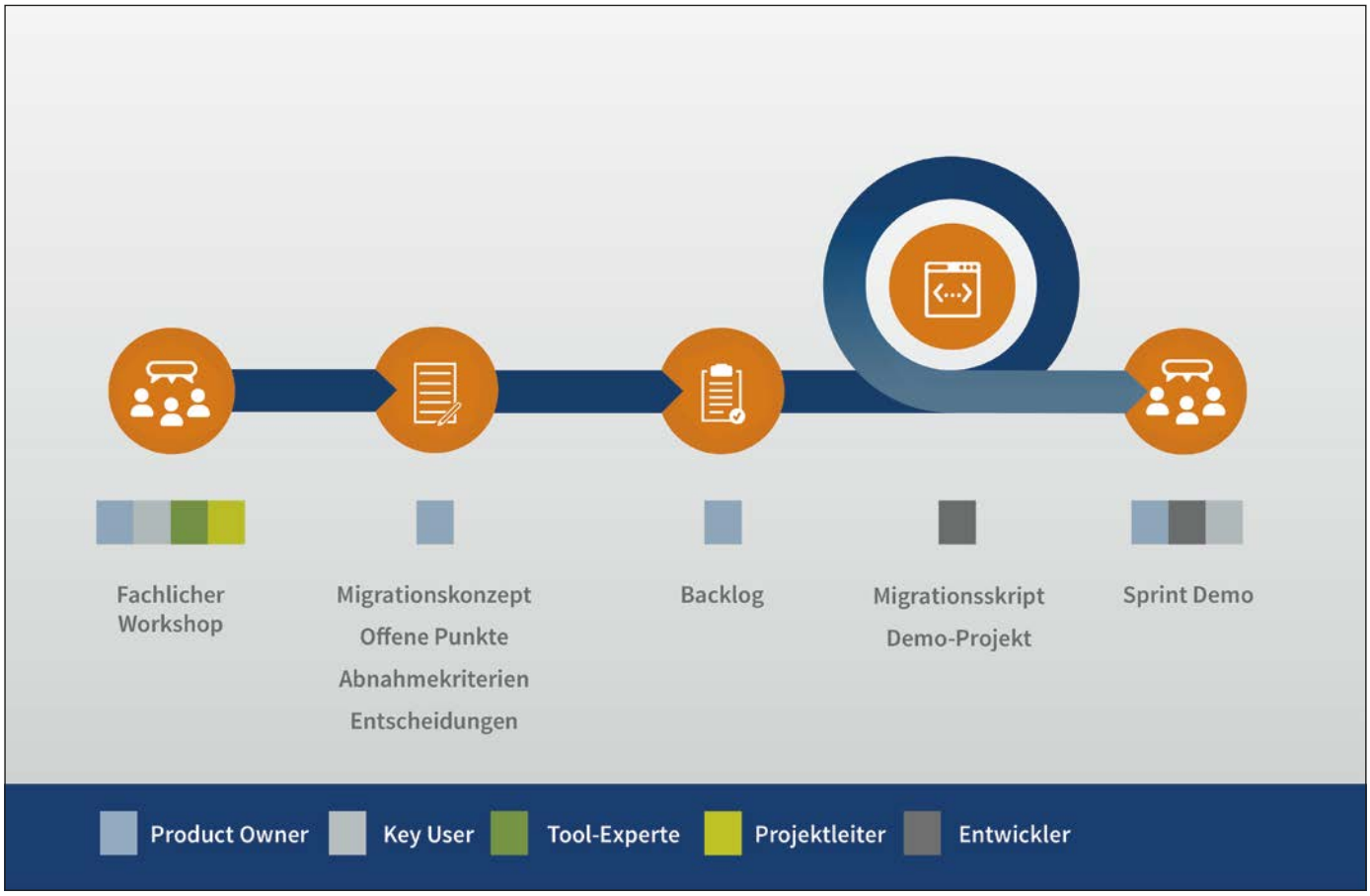


*Abb. 2: Iteratives Vorgehensmodell für die Zuordnung der Rollen zu den Artefakten und Ritualen, © Method Park*

*DOORS: 1 Datenmodell pro Anforderungsdokument*
DNG (Doors Next Generation): 1 Datenmodell pro Component (1 Component enthält mehrere Dokumente)

DOORS: keine expliziten Typen, oft über Enum-Attribut gelöst
DNG: explizite Typen (z. B. Customer Requirement)

DOORS: Text-Markup in allen Textfeldern möglich
DNG: Text-Markup nur im Beschreibungsfeld möglich

Jira: geschachtelter Enum-Datentyp möglich
DNG: geschachtelter Enum-Datentyp nicht vorhanden

Jira: dynamische Dokumente und Dokument-im-Dokument möglich (via Structure Plug-in)
DNG: beide nicht möglich

Jira: eingebettete Artefakt-Links im Beschreibungsfeld möglich
RQM (Rational Quality Manager): eingebettete Artefakt-Links nicht möglich

Kasten 2: Praktische Beispiele für inkompatible Datenmodelle

ten. Es ist also besonders wichtig, offene Entscheidungen zu dokumentieren und aktiv zu managen.
Wird eine externe Firma mit der Datenmigration beauftragt, was wahrscheinlich ist, muss zusätzlicher Aufwand berücksichtigt werden. Der externe Dienstleister braucht Zeit für die Einarbeitung; auch die eigenen Mitarbeiter müssen für den Wissenstransfer zur Verfügung stehen. Von einem Remote-Arbeiten ist hier abzuraten.

## DOs & DON'Ts

Um die Daten erfolgreich von einem Tool in das andere zu migrieren, lassen sich aus der Erfahrung mit solchen Projekten DOs und DON'Ts ableiten. Hier einige Tipps für Ihr nächstes Projekt.

**Technische Ebene (Skript)**
Technisch sollte eine Datenmigration, soweit es geht, automatisiert werden. Das grundlegende Vorgehen folgt dabei dem ETL-Prinzip: Extract – Transform – Load: Es werden ein oder mehrere Skripte erstellt, die die Daten aus dem bestehenden Tool exportieren, dann einer Transformation unterziehen und schließlich in das neue Tool importieren. Diese Skripte sollten die anschließende Datenvalidierung ebenfalls weitgehend erledigen.
Es ist sinnvoll, die Skripte von Anfang an so modular zu gestalten, dass sich Teile davon direkt für die Datenvalidierung verwenden lassen. Auch sollte berücksichtigt werden, dass das Skript bei der Migration großer Datenmengen unbetreut laufen wird und dazu sowohl so robust wie möglich als auch pausier- und wiederaufsetzbar sein muss. Hierzu empfiehlt es sich, die aus dem bestehenden Tool extrahierten Daten zu cachen und auch nach der Transformation zu persistieren. Damit vermeidet man während der Entwicklung unnötige Last auf dem bestehenden Tool.
Die Wahl der Schnittstelle sollte im Vorfeld wohl überlegt sein. Für die Migration von Requirements scheint beispielsweise das ReqIF-Format auf den ersten Blick attraktiv. Allerdings beschreibt dieses kein einheitliches Datenmodell, es handelt sich lediglich um ein Meta-Modell. Die verschiedenen und vermutlich inkompatiblen internen Datenmodelle der Tools lassen sich damit nicht automatisch vereinen, geschweige denn abbilden. Dadurch ist ein reibungsloser Datenaustausch mittels ReqIF relativ unwahrscheinlich. Ein REST-API ist in der Regel deutlich besser geeignet. Vorsicht jedoch bei OSLC-APIs: Hier sollte geprüft werden, ob das API alle benötigten Teile des Datenmodells abbilden kann.
Nicht nur die Daten selbst, sondern auch das Datenmodell muss migriert werden. Diese Migration kann gegebenenfalls automatisiert erfolgen. Das vereinfacht während der Skript-Entwicklung auch die nötigen Tests.

**Projektebene**
Zunächst sollte geklärt werden, ob eine Big-Bang-Strategie, also eine Migration des gesamten Datenbestands auf einmal, vermeidbar ist. Der größte Nachteil einer Big-Bang-Migration ist, dass diese gegebenenfalls nicht an einem Wochenende erfolgen kann. Außerdem sind Validierungen und unter Umständen ein Rollback bei einem Big Bang deutlich aufwendiger. Man kann einen Big Bang umgehen, wenn sich die Daten segmentieren lassen, beispielsweise, wenn es mehrere unabhängige Projekte gibt.
Bedenken Sie die Schnittstellen des bestehenden Toolings zu externen Tools. Müssen hier Verknüpfungen erhalten bleiben? Legen Sie außerdem Abnahmekriterien zur Datenvalidierung fest und automatisieren Sie soweit möglich deren Prüfung. Erstellen Sie alle Migrationsskripte von Beginn an inkrementell; schieben Sie dies nicht in den späteren Projektverlauf.
Sie können Entscheidungen im Projekt beschleunigen, indem Sie Statistiken generieren, beispielsweise über die Nutzung von Attributen und Artefakt-Typen oder über die Abhängigkeiten zwischen Daten verschiedener Projekte.
Entscheidend ist außerdem die ständige Zusammenarbeit mit den Key-Usern (hier unbedingt den Aufwand bei den Key-Usern einplanen) und den Tool-Experten. Wecken Sie jedoch bei den Key-Usern keine unrealistischen Erwartungen an das neue Tooling. Machen Sie beispielsweise deutlich, dass inkonsistente Daten auch in einem neuen Tool nicht automatisch konsistent werden, sondern im Vorfeld bereinigt werden müssen.

## Vorgehensmodell

Aus diesen Empfehlungen wurde ein Vorgehensmodell entwickelt (**siehe Abbildung 2**), das auf einem inkrementellen Vorgehen beruht. Dieses Modell beschreibt Projektrollen und ihre Interaktionen sowie zu entwickelnde Artefakte. Hierbei werden verschiedene agile Praktiken aus dem bekannten Scrum-Framework verwendet und an den Kontext angepasst. Die entsprechenden Begrifflichkeiten werden übernommen.
Zunächst wird ein Projektmitarbeiter für die Rolle des Product Owners (PO) des Migrationsskripts benannt, der das Scripting betreut oder selbst durchführt. Der PO ist verantwortlich für die Identifikation, Konsolidierung und Umsetzung der Anforderungen an die Datenmigration.
Daneben empfiehlt es sich, einen Projektleiter Datenmigration festzulegen. Dieser organisiert die im Weiteren beschriebenen Workshops und Meetings und verfolgt und kommuniziert den Projektfortschritt. Um den Erfolg seines Projektes zu sichern, muss der Projektleiter die Liste offener Entscheidungen im Blick haben und deren Abarbeitung vorantreiben. Die Rollen von Projektleiter und PO können durchaus von derselben Person ausgeübt werden.

Ein Entwickler ist für die Implementierung des Migrationsskripts und des Skripts zur Datenvalidierung zuständig. Wenn möglich, sollte der PO diese Tätigkeit übernehmen.

Es ist wahrscheinlich, dass die eigenen Mitarbeiter nicht über ausreichende Expertise in dem einzuführenden Tool verfügen. Daher werden externe Tool-Experten zur Beratung und gegebenenfalls Tool-Konfiguration herangezogen.

Neben einem initialen Kick-off-Workshop mit allen Projektbeteiligten sollten nach Bedarf regelmäßig fachliche Workshops stattfinden. Diese können entweder domänenspezifisch oder -übergreifend sein. Ein fachlicher Workshop will ein gemeinsames Verständnis des Istzustands erreichen, Entscheidungen bezüglich des Sollzustands treffen und Abnahmekriterien definieren.

Der PO dokumentiert in einem Migrationskonzept, welche Artefakte und Attribute auf welche Weise migriert werden. Entscheidungen hierzu sollte er ebenfalls dokumentieren. Auch die Abnahmekriterien hält der PO fest. Wichtig ist ebenso eine Liste offener Punkte (LoP), in der der PO erfasst, welche Entscheidungen noch getroffen werden müssen, damit das Migrationskonzept vollständig ist. Diese Artefakte entstehen iterativ und inkrementell unter anderem im Rahmen der fachlichen Workshops, parallel zur Entwicklung des Migrationsskripts.

Eine Checkliste (siehe unten) kann als Startpunkt für eine Anforderungsanalyse und Planung einer ALM-Datenmigration dienen. Für alle Artefakte, Attribute und Metadaten ist jeweils zu prüfen, ob das Datum migriert werden muss, und falls ja, wie es automatisiert aus dem alten Tool ausgelesen wird. Darüber hinaus ist zu klären, wie das Datum im Datenmodell des Ziel-Tools abgebildet (Mapping) und wie es automatisiert in das neue Tool eingespielt wird.

Die Anforderungen an das Migrationsskript leitet der PO aus dem Migrationskonzept ab und dokumentiert sie im Backlog. Dort liegen diese Anforderungen für den Entwickler in der gewünschten Abarbeitungsreihenfolge. In mehrwöchigen Iterationen (ca. zwei bis sechs Wochen) werden nach agiler Vorgehensweise diese Anforderungen implementiert und anschließend den relevanten Key-Usern im Rahmen der Sprint Demo präsentiert. Dort gewonnenes Feedback pflegt der PO in Migrationskonzept und Backlog ein.

Um den aktuellen Stand des Migrationsskripts demonstrieren zu können, wählt der PO ein repräsentatives Projekt im bestehenden ALM-Tool aus. Dieses Projekt migriert der Entwickler testweise für die Sprint-Demo mit dem aktuellen Stand des Skripts und zeigt das Ergebnis im neuen Tool vor.

Tools weisen zum Teil grundlegende Unterschiede im Datenmodell auf, die in der Migrationsstrategie berücksichtigt werden müssen. **Kasten 2** nennt einige praktische Beispiele.

## Fazit

Der Aufwand und die Komplexität eines ALM-Datenmigrationsprojektes sollten nicht unterschätzt werden. Zur Komplexität tragen technische Risiken bei, sowie die notwendige Abstimmung der Stakeholder. Wie so oft ist die Kommunikation zwischen den Projektbeteiligten von entscheidender Bedeutung. Dem trägt das beschriebene Vorgehensmodell Rechnung, indem es die wichtigsten Kommunikationswege und Transparenz-schaffende Maßnahmen vorgibt. Dieses Modell, die geschilderten Stolperfallen und Strategien helfen dabei, den Überblick zu behalten und zielgerichtet vorzugehen, damit das Projekt zum Erfolg führt.

## Datenmigration: Artefakt-Checkliste für Requirements- & Test-Management

*Requirements-Management*

Stakeholder Requirements, System Requirements, Component Requirements

Anforderungsdokumente
ggf. Include-Beziehungen beachten

Ziele z. B. Vision

Trace von Requirement zu Requirement(s)
z. B. System R. „derived from“ Stakeholder R.

Traces von Dokument zu Dokument

Trace von Requirement zu Test Case(s)
z. B. „tested by“

Trace von Component Requirement zu Code z. B. via VCS

Link von Requirement zu Requirement(s)
z. B. System Requirement X „variant of“ System Requirement Y

*Testmanagement*

Test Case

Test Suite ggf. weitere Sammlungen (Sessions, Collections, Cycles usw.)

Test Plan

Test Script

Test Execution Record, Test Result
ggf. weitere Artefakte der Ausführung

Trace zu Requirement(s) z. B. „validates“

Links zwischen Test-Artefakten
z. B. „related to“

**Domänen übergreifend**

*Artefakt-Attribute*

Lebenszyklus-Attribute z. B. Status

Titel und Beschreibung

Referenz auf User
z. B. „assigned to“, „watched by“

Referenz auf andere Artefakte
z. B. „Release“, „Build“, „Project“

Unveränderliche Attribute
z. B. „created by“, „created on“

Custom Attribute z. B. „Subsystem“, „Priority“

*Weitere Daten und Meta-Daten*

Traces/Links zu weiteren, externen Artefakten z. B. Architektur- oder Change-Management

Hyperlinks in Freitextfeldern zu anderen Artefakten
interne sowie externe Artefakte berücksichtigen

Datei-Anhänge und eingebettete Objekte
z. B. PDF, JPG, OLE

Kommentare incl. deren Metadaten

Verlauf/Historie des Artefakts

Text-Formatierung z. B. HTML, Wiki Markup

Dynamische Strukturen und Abfragen
z. B. Query, Jira Structure Generator

Aufwandstracking z. B. Arbeitsstunden

### Literatur & Links

**[Method]** Method Park Experten-Blog zum Software & Systems Engineering, siehe:
**https://www.methodpark.de/blog/scrum-und-kanban-wann-nutze-ich-was/**

### Der Autor

Benjamin Wiese
(benjamin.wiese@methodpark.de)
ist bei Method Park als Software Engineer und Berater tätig. Er leitet Projekte zur Datenmigration im ALM und führt Schulungen zu Clean Code und Requirements Engineering durch. Zudem gilt sein Interesse den agilen Methoden und dem Software Craftsmanship.