Sonderdruck aus OBJEKTspektrum

Ausgabe 01|2021

Deutschland € 12,90 Österreich € 13,90 Schweiz sfr 22,20

4 194164 112900 01

OBJEKTspektrum

Software-Architektur und IT für Profis

www.OBJEKTspektrum.de

Architekturtrends – Die Must Dos von morgen

Interview mit Malte Fiala

Der Architekt mit Cloud-Spezialisierung hilft dabei, die Österreichische Bundesbahn ins Cloud-Zeitalter zu bringen. Im Gespräch berichtet er, wieso die ÖBB dabei mit verschiedenen Geschwindigkeiten fährt

- Burn-out-Systeme in der Softwareentwicklung
- Praktiken für gute Entscheidungen in IT-Projekten
- 2021: Die Prognosen und Wünsche der OS-Redaktion für das neue Jahr

G 6540F

Moderne Unternehmensarchitektur

Fractal Architecture for IT – Balance von Autonomie UND Alignment

Microservice-Architekturen und Team-Agilität sind an Autonomiezielen ausgerichtet. In IT-Landschaften führt das zu einer Vielzahl von kleinen Softwareeinheiten, die flexibel weiterentwickelbar und austauschbar sind – für eine kurze „time to market". Spätestens zum Endnutzer hin braucht es Alignment, ebenso für die Geschäftsprozessintegration. Der hier vorgestellte Ansatz „Fractal Architecture for IT" basiert auf einer realen Architekturexploration. Zentral ist ein bestimmtes Integrationsmuster.

Wenn man so will, waren die 2010er-Jahre von Microservices geprägt. Der Vordenker Martin Fowler berichtete im März 2014 von „einer bestimmten Art, Softwareanwendungen als Zusammenstellung von unabhängig deploybaren Diensten zu entwerfen" [Fowl]. Der Grundstein für agile Teams wurde schon 2001 mit dem Agilen Manifest gelegt.

Im letzten Jahrzehnt verbanden sich beide Ansätze zu einer Art Autonomiewunderwaffe: Im Gegensatz zur früher etablierten Alignment-Waffe Enterprise Architecture Management (EAM) galt es, Teams interdisziplinär zu besetzen, Entscheidungen dezentral zu fällen und Änderungskaskaden in IT-Systemen zu begrenzen. „Kurze ‚time to market' statt einer Ewigkeit für allgemeinen Konsens" könnte das Motto lauten. Heutzutage beherbergen IT-Landschaften eine Vielzahl von verteilten Softwareeinheiten, implementiert von agilen Teams, die autonom eine abgegrenzte Fachlichkeit verantworten. Dies ist eine Errungenschaft. Und zugleich ein hausgemachtes Problem der kommenden Jahre, wenn Unternehmen kein Alignment darüber hinaus schaffen. Ziel ist letztlich die Ausrichtung am Geschäftszweck des Unternehmens, der sich in Vision, Mission und strategischen Zielen des Unternehmens ausdrückt. Keller [Kel17] spricht in diesem Zusammenhang von strategischem IT-Alignment.

Greifbarer ist meist das architektonische Alignment von IT-Systemen an Geschäftsprozessen. Darauf fokussiert sich dieser Artikel. Anhand einer realen Architekturexploration rund um die Marktzulassung von Produkten eines Unternehmens beschreibt er die Herausforderungen und die erarbeitete Lösung, in deren Zentrum ein bestimmtes Integrationsmuster steht.

Eine fraktale IT-Landschaft entsteht dann, wenn dieses Muster rekursiv auf mehreren Ebenen einer IT-Landschaft wiederholt wird. Das darin liegende Potenzial zur Balance von Autonomie und Alignment ist im Rahmenwerk „Fractal Architecture for IT" [Fract] beschrieben, auf das der Artikel im letzten Abschnitt

Domäne	Im Kontext dieses Artikels: fachlich abgegrenzter Teilbereich einer IT-Landschaft; genauer: ein thematischer Cluster von Geschäftsfähigkeiten
EAM	Methoden und Strukturen, um die IT-Fähigkeiten eines Unternehmens im Kontext der Geschäftsstrategie (weiter) zu entwickeln. Wesentliche Artefakte sind Ist-, Soll- und Migrationsbeschreibungen. Zur Kommunikation und Planung dienen oft visualisierte Capability Maps und IT-Landschaften (angelehnt an [TOG18] und [Kel17])
Geschäftsfähigkeit (Business Capability)	Geschäftsfähigkeit innerhalb eines Unternehmens, das heißt, *etwas*, das Menschen oder technische Systeme im Unternehmen tun – unabhängig davon, *wie* sie es tun. Zum Beispiel wiederkehrende Tätigkeiten innerhalb von Geschäftsprozessen (angelehnt an [BAG20])

Tabelle 1: Grundbegriffe des Enterprise Architecture Management (EAM)

eingeht. Doch zunächst gibt **Tabelle 1** einen Überblick über die verwendeten Grundbegriffe aus dem EAM.

Flexibilität durch Autonomie UND Nachverfolgbarkeit durch Alignment?

Die eingangs skizzierte Problemstellung taucht in verschiedenen Ausprägungen immer wieder an der Schnittstelle von IT-Umsetzung und Management auf. Die Architekturexploration, die als Basis für diesen Artikel dient, hat mit behördlichen Marktzulassungen für die Produkte eines Unternehmens zu tun. Es war bereits eine IT-Landschaft mit fachlichen Domänen visualisiert. Nun wollte die auftraggebende Fachabteilung für Marktzulassungen die Landschaft mit bestehenden und neuen IT-Anwendungen weiterentwickeln. Ziel war es, eine Architektur zu explorieren, die den domänenübergreifenden Geschäftsprozess integriert, ohne die Autonomie der Teams in den einzelnen Domänen zu stark einzuschränken.

Abbildung 1 zeigt die IT-Landschaft zu Beginn der Architekturexploration in einer vereinfachten Form. Sie beschreibt die fachlichen Domänen, die IT-Anwendungen und den grundlegenden Geschäftsprozess. Da die Marktzulassung eines neuen oder geänderten Produkts sehr komplex ist, sind all jene Geschäftsfähigkeiten in der Domäne „Projektplanung" gebündelt. Hierfür war bereits eine microservicebasierte Anwendung „Projektplaner" in Entwicklung. In diese gibt ein Projektleiter die Rahmendaten eines neuen Marktzulassungsprojekts ein, insbesondere den Umfang der einzuholenden Genehmigungen (1). Letztlich geht es darum, ein Antragsdokument mit aufbereiteten Produktdaten zu generieren, das den Behörden zur Genehmigung vorgelegt werden kann. Dies ist der wichtigste Meilenstein, den der Projektleiter auch im Projektplaner nachhält (7).

Bis es so weit ist, muss zunächst ein Datenverantwortlicher offiziell freigegebene Produktdaten aus der Produktentwicklung abrufen (2) und gemäß der einschlägigen Rechtsnormen aufbereiten (3). In der zugehörigen Domäne „Informationsmanagement" gab es zu Beginn der Exploration allerdings keine IT-Unterstützung, weswegen noch Office-Dateien wie Spreadsheets zum Einsatz kamen. Letztlich landen die aufbereiteten Daten aber im „Dokument-Generator", einem Kaufprodukt, das auf rechtskonforme Antragsdokumente spezialisiert ist (4). In der zugehörigen Domäne „Dokumenten-Management" gab es bereits ein weiteres Kaufprodukt, nämlich ein Dokument-Management-System „DMS". In dieses

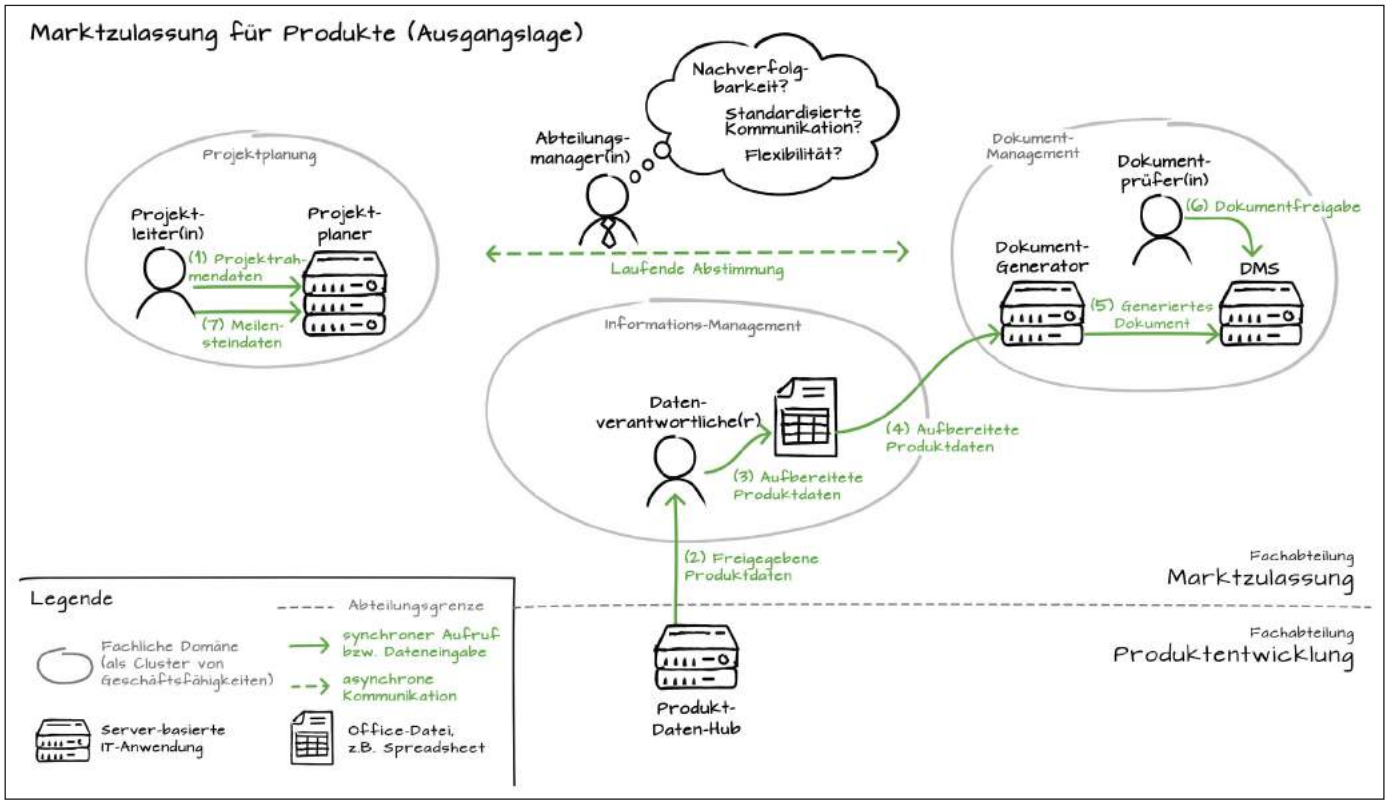

Abb. 1: IT-Landschaft für Marktzulassungen (Ausgangslage)

legt der Generator ein neues Dokument automatisiert ab (5).
Nach einer Sichtprüfung und Bündelung mit weiteren Dokumenten wie Messergebnissen gibt ein Dokumentprüfer das Antragsdokument für die Behörden offiziell frei (6). An dieser Stelle gibt er die Information über den erreichten Meilenstein an den Projektleiter (7). Wie allerdings das Dilemma der Abteilungsleitung zeigt, braucht es generell eine laufende Abstimmung zwischen den Projektbeteiligten, damit der Geschäftsprozess funktioniert.
Damit verbunden wollte die Abteilungsleitung die folgenden Ziele in der künftigen IT-Landschaft erreichen:

- *Nachverfolgbarkeit:* Um den Behörden gegenüber die gesetzeskonforme Erhebung der Produktdaten in einem Antragsdokument nachweisen zu können, sollten alle offiziellen Schritte im jeweiligen Marktzulassungsprojekt nachverfolgbar sein.
- *Standardisierte Kommunikation:* Als Mittel zum Zweck sollte die offizielle Kommunikation in Marktzulassungsprojekten standardisiert werden. Dies betraf vor allem eine gemeinsame Fachsprache und die formalisierte Datenaufbereitung für Produktdaten und Messergebnisse.
- *Flexibilität:* Jene Ziele sollten für ein gewisses Alignment der bisher autonom agierenden Teams in den Domänen sorgen. Dennoch sollte dies nicht zu einer allzu zentralisierten und inflexiblen IT-Landschaft führen. Schließlich sorgen die unternehmensinterne Produktentwicklung und der externe Gesetzgeber immer wieder für notwendige Veränderungen in der IT-Landschaft.

Um eine Lösung zu entwerfen, führte das Explorations-Team über einen Zeitraum von fünf Monaten mehrere Workshops durch und erstellte einen lauffähigen Softwareprototypen. Ergebnisartefakte waren zudem zahlreiche Visualisierungen und textuelle Beschreibungen. Konkret verfeinerte und operationalisierte das Explorations-Team zunächst die genannten Ziele. Es wurde klar, wie sie sich in die Strategie des Gesamtunternehmens einfügen und welchen zusätzlichen Nutzen die künftige IT-Landschaft erzeugen könnte. Ein Audit-Dashboard zur Nachverfolgbarkeit von Marktzulassungsprojekten könnte beispielsweise zusätzliche Informationen zur Prozessoptimierung liefern.
Zur Vorbereitung der Prototypentwicklung erarbeitete das Explorations-Team mit Fachexperten stark vereinfachte, aber realistische Szenarien. Auf diese Weise wurden sowohl die Fachexperten als auch die Softwareentwickler früh einbezogen. Letztere implementierten schließlich innerhalb eines knappen Zweimonatszeitraums den Prototypen auf einer kubernetesbasierten Cloud-Plattform mit vorausgewählten Technologien. Diese waren im Wesentlichen RabbitMQ, Kafka, Postgres und MongoDB. Letztlich gab das Explorations-Team eine Architekturempfehlung an die Abteilungsleitung und erstellte ein Scrum-Backlog für ein Minimum Viable Product (MVP).

Geht. Es wird durch ein bestimmtes Integrationsmuster erleichtert

Die künftige IT-Landschaft, wie sie das Explorations-Team erarbeitete, ist vereinfacht in **Abbildung 2** dargestellt. Die neu eingeführte Domäne der „Nachverfolgung“ steht unmittelbar für das Ziel der Nachverfolgbarkeit. Bisher konnte die Abteilungsleitung die erfolgten Schritte nur mittelbar und verteilt nachverfolgen – im Produkt-Daten-Hub, in Office-Dateien, im Dokument-Generator und im DMS. Die erreichten Meilensteine wiederum musste sie im Projektplaner nachschlagen. Da der bestehende Domänenschnitt nicht mehr geändert werden sollte, lag die Einführung einer zusätzlichen integrativen Domäne nahe.
Ihr sind die Softwareeinheiten zugeordnet, die dem Ziel der Nachverfolgbarkeit dienen. Die wichtigste davon ist ein Nachrichtenkanal, der auf dem asynchronen Publish-/Subscribe-Kommunikationsmuster basiert. Eine fundierte Einführung gibt [Hoh03], als Kurzreferenz eignet

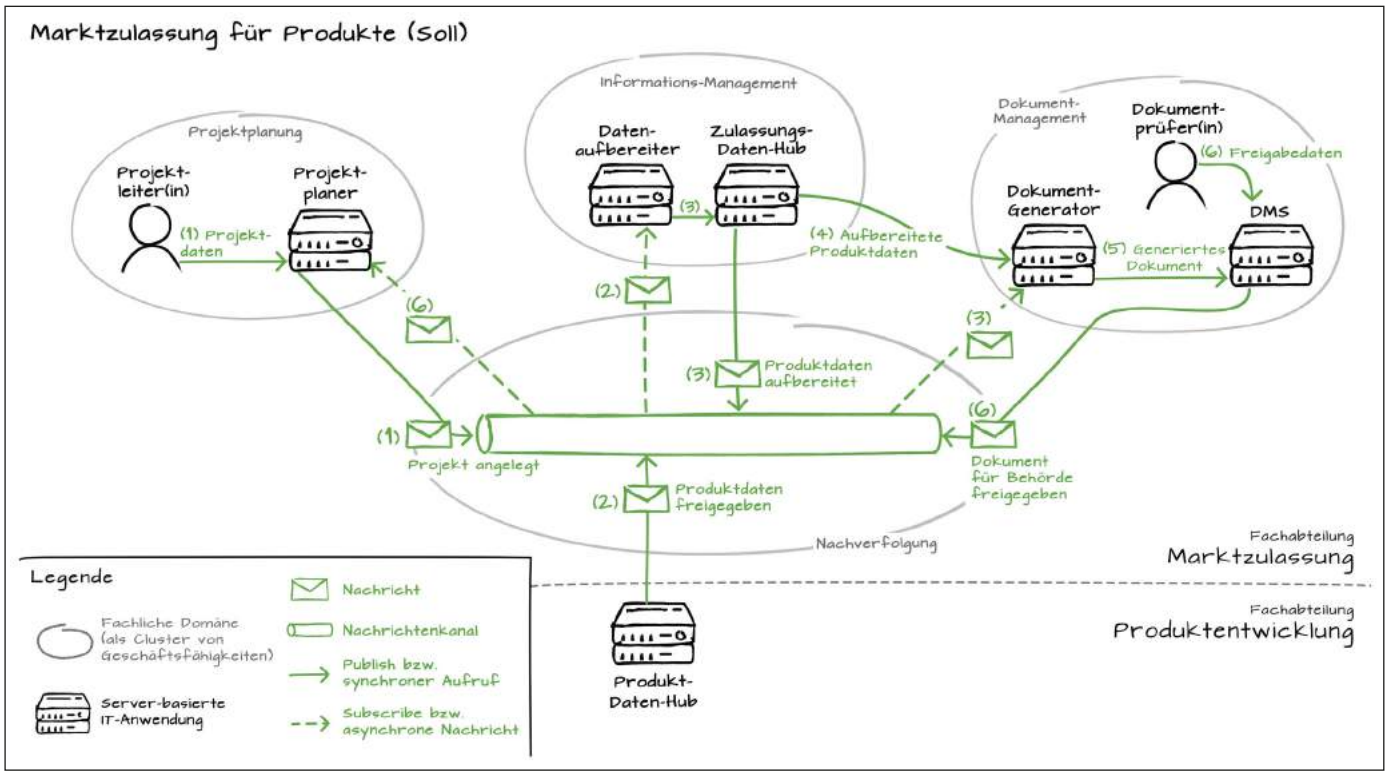

Abb. 2: IT-Landschaft für Marktzulassungen (Soll)

sich jedoch auch [Ente] desselben Autors. Der wesentliche Vorteil des Publish-/Subscribe-Musters ist es, dass die sendende die empfangende IT-Anwendung nur indirekt über das Netzwerk aufruft. Es kommt ein ereignisgetriebener Ansatz hinzu, wie er auch in vielen Architekturstilen wie Event Sourcing und SOA 2.0 prominent ist. Daher sind die Nachrichten, die über das Publish-/Subscribe-Muster ausgetauscht werden, als fachliche Ereignisse formuliert.

Beispielsweise veröffentlicht der Projektplaner eine Nachricht „Marktzulassungsprojekt angelegt“ (1), das DMS eine Nachricht „Dokument für Behörde freigegeben“ (6). Welche IT-Anwendungen jene Nachrichten vom Kanal abonnieren, ist für den Projektplaner beziehungsweise das DMS weitgehend unerheblich. So koppelt der Nachrichtenkanal die IT-Anwendungen nicht zu stark aneinander – trotz der Zentralisierung, die er in die IT-Landschaft einführt. Domäneninterne Ereignisse wie „Benutzer hinzugefügt“ bleiben unpubliziert, da sie für die Nachverfolgbarkeit des Geschäftsprozesses irrelevant sind.

Das grundlegende Integrationsmuster ist in **Abbildung 3** abstrahiert. In dessen Mitte befindet sich der Nachrichtenkanal, der eine integrative Domäne begründet (A). Damit verknüpft ist ein standardisiertes Format für Ereignisse, das auf diese Domäne zugeschnitten ist, wie hier die Nachverfolgbarkeit von Marktzulassungsprojekten (B). Das Ereignisformat steht unmittelbar für das Ziel der standardisierten Kommunikation und definiert Mindestangaben, die eine IT-Anwendung in einem Ereignis publizieren muss. Neben einem Zeitstempel und dem Kürzel der publizierenden Anwendung sind das eine Korrelations-ID sowie Revisions-IDs von historisierten Detaildaten (C). So sind umfangreiche Detaildaten, wie die Struktur von Baugruppen, in den Ereignissen selbst nicht vorgesehen. Stattdessen ist es IT-Anwendungen nach wie vor erlaubt, Detaildaten synchron und Punkt zu Punkt abzurufen (D). Dies wirkt einer zu starken Zentralisierung der IT-Landschaft entgegen.

Was das Ziel der Flexibilität angeht, herrschte durch den bestehenden Domänenschnitt und vorhandene Microservice-Konzepte eine gute Ausgangslage. Allerdings sollte die angelegte Flexibilität nicht zu stark eingeschränkt werden.

Wie Microservice-Architekturen – oder allgemein: verteilte modulare Architekturen – die Flexibilität einer IT-Landschaft unterstützen, ist in [Dow18] gut erklärt. Eine Modularisierung nach klar abgegrenzter Fachlichkeit ist dabei in aller Regel zu empfehlen. Hierbei wiederum hilft das Domain-Driven Design (DDD), das ursprünglich auf [Eva03] zurückgeht und sich zwischenzeitlich stark weiterentwickelt hat. Auch wenn dieser Artikel Gedankengut des DDD aufgreift, ist sein Domänenbegriff nicht mit dem von DDD identisch. Nähere Auskunft gibt [Fract]. Konkret behalten die Domänen im Integrationsmuster die Hoheit über „ihre“ Fachlichkeit und die IT-Anwendungen die Hoheit über „ihre“ Daten (C und D). Einen Überblick über die vorgestellten Merkmale des Integrationsmusters gibt **Tabelle 2**.

Wie kann man das vorgestellte Integrationsmuster nun bewerten? Es impliziert, dass Flexibilität durch Autonomie der „Default“ bleibt. Sie wird unterstützt durch fachlich abgegrenzte Domänen, in denen autonome Teams für modularisierte IT-Anwendungen verantwortlich sind. Alignment-Ziele wie die Nachverfolgbarkeit eines Geschäftsprozesses erfordern jedoch Zugeständnisse. Im Integrationsmuster sind das zunächst die fachlichen Ereignisse, welche die beteiligten IT-Anwendungen an eine zentrale, integrative Domäne publizieren müssen. Allerdings besteht diese Pflicht nur für Ereignisse, die dem Ziel und der Fachlichkeit der integrativen Domäne dienen (A). Neben einem Nachrichtenkanal beinhaltet eine solche Domäne weitere Softwareeinheiten wie einen persistenten Ereignisspeicher sowie mindestens eine IT-Anwendung zur Auswertung der Ereignisse. In **Abbildung 3** ist

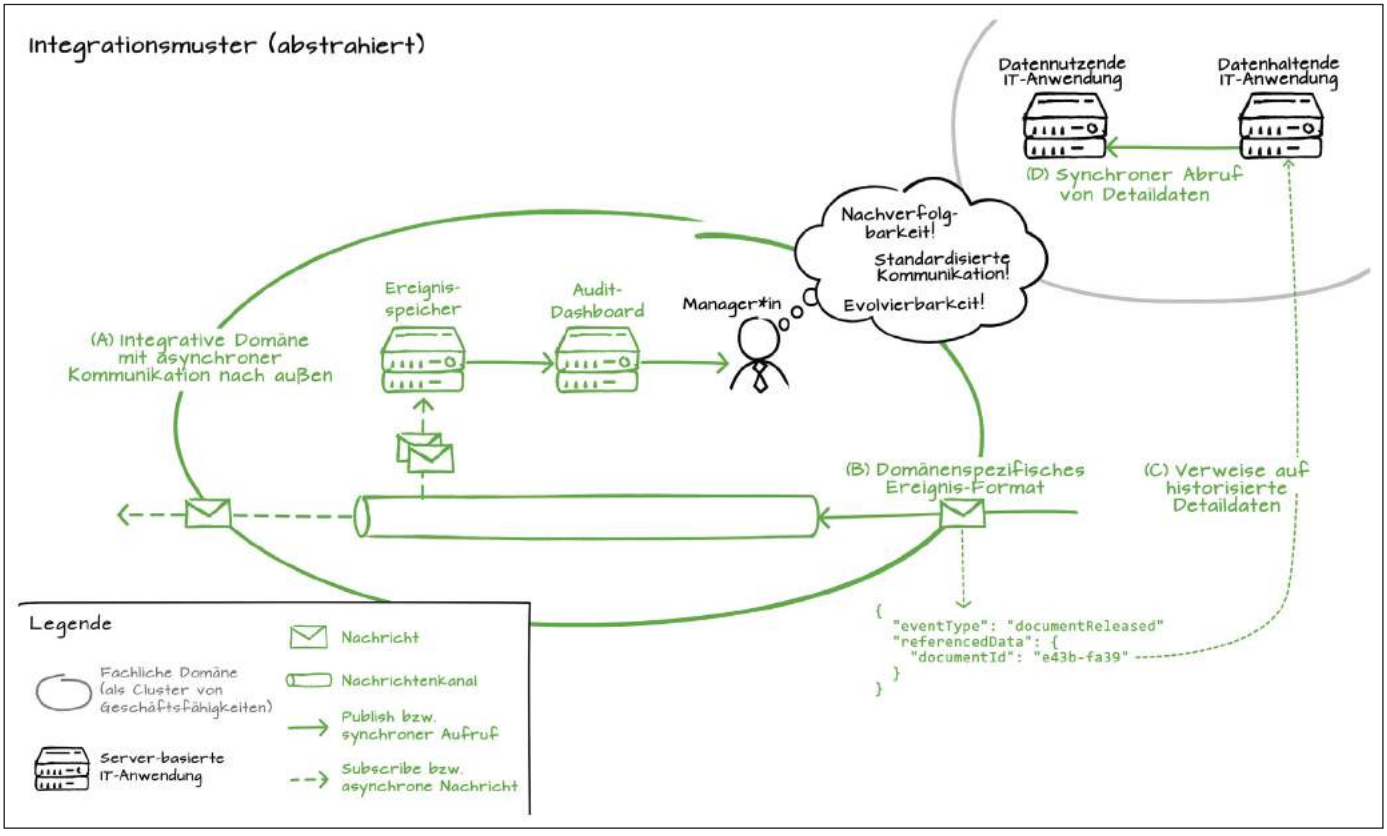

Abb. 3: Das abstrahierte Integrationsmuster

(A) Integrative Domäne	Ist mit einem konkreten Alignment-Ziel verknüpft, wie die Nachverfolgbarkeit eines Geschäftsprozesses. Bündelt typischerweise Geschäftsfähigkeiten, die stark auf die Kommunikation mit anderen Domänen angewiesen sind und so den „integrativen" Charakter der Domäne ausmachen.
(B) Domänenspezifisches Ereignisformat	Ein standardisiertes Format für fachliche Ereignisse, das speziell auf eine integrative Domäne zugeschnitten ist. Es besitzt eine technische Repräsentation zum Beispiel in XML oder JSON.
(C) Verweise auf historisierte Detaildaten	Revisions-IDs innerhalb des Ereignisformats (B), die auf die Detaildaten des jeweiligen Ereignisses verweisen. Diese befinden sich in einer datenhaltenden IT-Anwendung, die zumeist in einer separaten Domäne angesiedelt ist.
(D) Synchroner Abruf von Detaildaten	Abruf eines bestimmten historisierten Datenstandes, auf den eine Revisions-ID in einem bestimmten Ereignis verweist (C). Dieser Abruf erfolgt in der Regel synchron, um die asynchron kommunizierten Ereignisse leichtgewichtig zu halten.

Tabelle 2: Merkmale des Integrationsmusters

ein Audit-Dashboard dargestellt, welches beispielsweise Informationen zur Prozessoptimierung liefern kann.

Das zweite Zugeständnis ist ein standardisiertes, aber domänenspezifisches Ereignis-Format (B). Es ist auf die Fachlichkeit der integrativen Domäne zugeschnitten, damit diese überhaupt ihrem Ziel und ihrer Fachlichkeit dienen kann. Eine allgemeingültige Balance von Autonomie- und Alignment-Zielen stellt das Integrationsmuster jedoch nicht her. Vielmehr ermöglicht es eine fallspezifische Balance. Beispielsweise kann eine IT-Anwendung ihre Detaildaten in kleinem Umfang auch in einem fachlichen Ereignis mitschicken, um einen zusätzlichen synchronen Aufruf einzusparen. Abwägungen dieser Art sind jedoch nur fall-spezifisch in konkreten IT-Projekten möglich.

Fraktale Architektur für IT – Eine neue Sau durchs Dorf?

Was passiert, wenn man das grundlegende Integrationsmuster auch *innerhalb* der einzelnen Domänen anwendet? Schließlich müssen die Softwareeinheiten, die zusammen eine IT-Anwendung bilden, auch miteinander kommunizieren! Dies ist der Grundgedanke der „Fractal Architecture for IT". Es handelt sich um ein konzeptionelles Rahmenwerk, welches auf der rekursiven Wiederholung des Integrationsmusters auf mehreren Ebenen einer IT-Landschaft basiert. Einen visuellen Eindruck gibt **Abbildung 4**.

Wie auch das grundlegende Integrationsmuster geht das Rahmenwerk von Autonomie-Zielen als „Default" aus, allen voran die flexible Weiterentwicklung einer IT-Landschaft und eine kurze „time to market" für neue Produkte und Features. Sein Anspruch ist es, die Balance mit Alignment-Zielen zu erleichtern. Neben der Nachverfolgbarkeit von Geschäftsprozessen zählen dazu eine integrierte User Experience für Endnutzer und eine holistische Skalierung in der Cloud, die an Use Cases statt an einzelnen Softwareeinheiten ausgerichtet ist. Dazu vereint die „Fractal Architecture for IT" Konzepte aus dem EAM mit Konzepten aus Microservice-Architekturen. So soll es ein gemeinsames Verständnis zwischen Umsetzungs-Teams und Managern fördern. [Fract] beschreibt das Rahmenwerk näher und behandelt dabei dessen Grundlagen aus dem EAM, der Service-Oriented Architecture (SOA), aus Microservice-Architekturen und dem DDD sowie aus

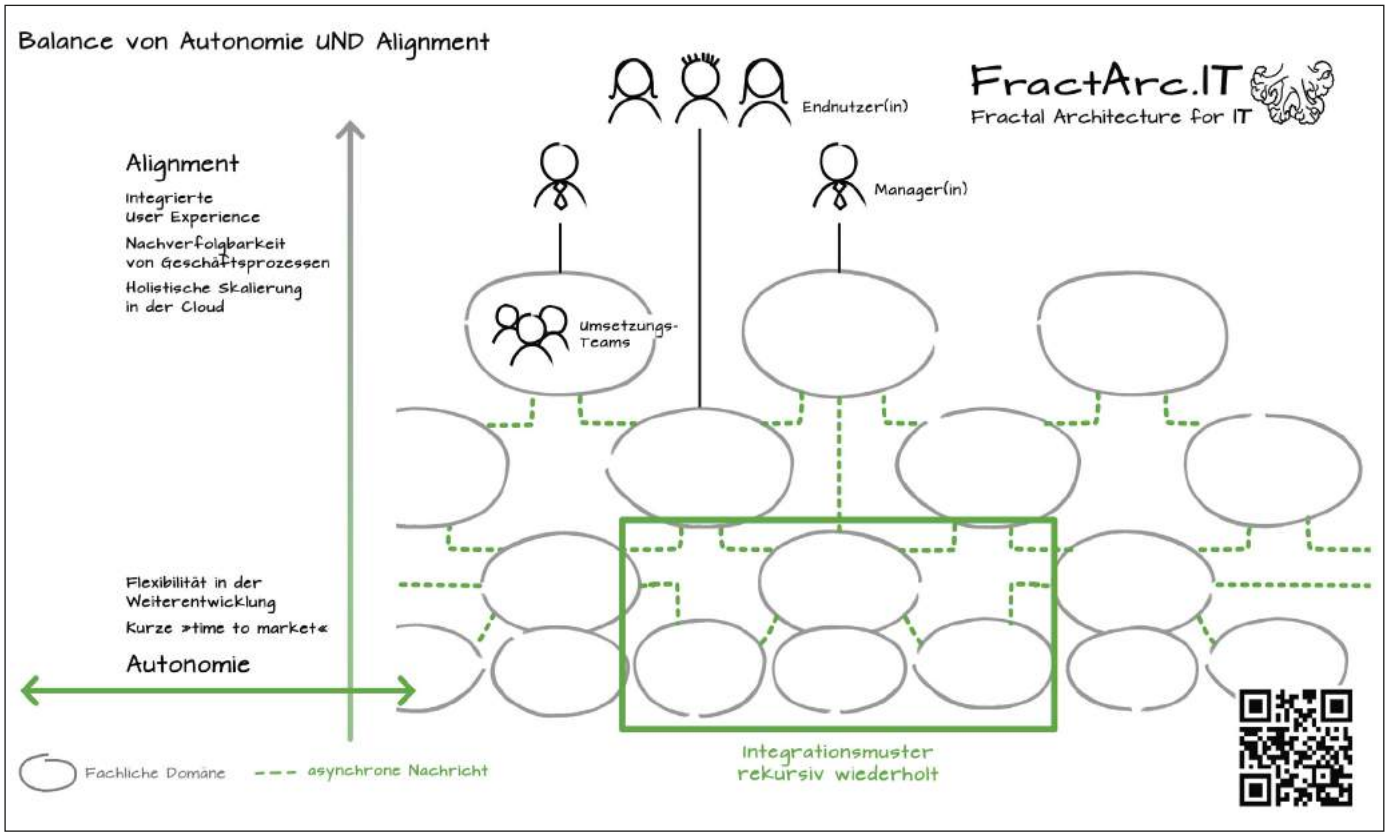

Abb. 4: Eine fraktale IT-Landschaft (schematisch)

der allgemeinen Systemtheorie. Im Unterschied zum hier vorgestellten Integrationsmuster bezieht sich die „Fractal Architecture for IT" auf Bounded Contexts statt auf Domänen, da jenes Konzept aus DDD spezifischer und in Umsetzungs-Teams bekannter ist.

Namensgebend ist die fraktale Geometrie, die mit der Systemtheorie verwandt ist. Ein fraktales Gebilde wie ein Brokkoli-Kopf ist selbstähnlich: Ein einzelnes Brokkoli-Röschen ähnelt in seiner Struktur dem übergeordneten Strunk und dieser wiederum dem gesamten Kopf. Erkennen Sie eine ähnliche Struktur der IT-Landschaft in **Abbildung 4**?

Fazit

Auch wenn das Rahmenwerk in seiner Gänze noch nicht in der Praxis eingesetzt wurde, mag es bereits ein gemeinsames Verständnis zwischen Umsetzungs-Teams und Managern fördern. Setzen Sie gerne die Ideen und Grafiken von [Fract] in Ihren IT-Projekten ein! Sie sind am Beispiel einer E-Commerce-Firma veranschaulicht, um konkrete Diskussionen zu ermöglichen. Wenn Sie auch nur einzelne Ideen und Grafiken in Ihrem Projekt nützlich finden, hat das Rahmenwerk seinen Anspruch zum Teil schon erfüllt. Kontaktieren Sie gerne den Autor, wenn Sie Feedback und Impulse haben! ||

Der Autor

Matthias Ostermaier

(matthias.ostermaier@maibornwolff.de)

ist ein leidenschaftlicher Integrator der IT-Architekturebenen, von einzelnen Softwareanwendungen bis zur strategischen Unternehmensebene. Hierzu baut er auf Domain-Driven Design (DDD), Microservice-Architekturen und Enterprise Architecture Management (EAM).

Literatur & Links

[BAG20] Business Architecture Guild®: A Guide to the Business Architecture Body of Knowledge® (BIZBOK® Guide), 2020, siehe: **view.businessarchitectureguild.org** (abgerufen am 13.07.2020)

[Dow18] H. Dowalil, Grundlagen des modularen Softwareentwurfs – Der Bau langlebiger Mikro- und Makro-Architekturen wie Microservices und SOA 2.0, Hanser, 2018

[Ente] G. Hohpe, Publish-Subscribe Channel, siehe: **www.enterpriseintegrationpatterns.com/patterns/messaging/PublishSubscribeChannel.html**

[Eva03] E. Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software, Addison-Wesley, 2003

[Fowl] M. Fowler, J. Lewis, Microservices – a definition of this new architectural term, siehe: **martinfow-ler.com/articles/microservices.html** (abgerufen am 16.09.2020)

[Fract] M. Ostermaier, Fractal Architecture for IT, siehe: **fractarc.it**

[Hoh03] G. Hohpe, B. Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, Addison-Wesley, 2003

[Kel17] W. Keller, IT-Unternehmensarchitektur – Von der Geschäftsstrategie zur optimalen IT-Unterstützung, dpunkt, 2017

[TOG18] The Open Group, The TOGAF® Standard, Version 9.2, 2018