Digital Commerce COOKBOOK

Autoren: Günter Heiß,
Joachim Uhrlaß

eCube

01 Einleitung 4

1. Replatforming 6

2. Gründe für Replatforming 7

3. Bedeutung des Replatformings für den Unternehmenserfolg 8

4. Grundlegende Fragestellungen 9

4.1 Startpunkt: Frontend oder Backend? 9
4.2 Frontend: Integriert oder Headless*? 11
4.3 Umfang: Replatforming oder Retrofitting? 13
4.4 System: Neu bauen oder transformieren? 15
4.5 Zielarchitektur: Monolith oder API-Plattform*? 17
4.6 Lösung: Best-of-breed oder Best-fit*? 19
4.7 Umsetzung: Agil oder Wasserfall? 20

02 Organisation und Vorbereitung 22

1. Zielsetzung 24

1.1 Smarte Ziele 24
1.2 Minimum Viable Product (MVP)* 26

2. Projekt-Team 27

2.1 Agiles Projekt-Team zusammenstellen 28
2.2 Nicht-agiles Projektteam zusammenstellen 29

3. Kernkompetenzen im Projektmanagement 30

4. Anforderungen 31

4.1 User Stories in agilen Projekten 31
4.2 Qualitätsszenarien 32
4.3 Kommunikation 33

5. Klassisches Lastenheft in nicht-agilen Projekten 34

6. Tools für die Organisation und Vorbereitung 37

03 Planung 38

1. Agil und Wasserfall sinnvoll verbinden 40

2. Business Owner* einbinden 42

3. Anforderungen strukturieren 43

3.1 Epics* entwickeln 43

3.2 Ressourcenbedarf und Projektdauer ermitteln 44

3.3 Projektkosten 44

3.4 Epics planen und priorisieren 45

3.5 Scope* anpassen 46

3.6 Fertigstellungstermin ermitteln 47

4. Kommunikation 48

5. Tools für die Planung 50

04 Umsetzung am Beispiel 51

1. Migration vom Monolithen zu Composable Commerce 52

1.1 Motivation für das Replatforming 53

1.2 Annahmen zum Status quo 53

2. Iterative Transformation 54

2.1 Transform Eliminate Pattern* 54

2.2 Strangler Fig Pattern* 54

3. Schritt für Schritt zum Composable Commerce 55

3.1 Composable-Commerce-System einrichten 55

3.2 Aggregator-Service* und Drittsysteme einrichten 56

3.3 Eigene Services implementieren 57

3.4 Frontend einrichten 58

3.5 Migration 59

4. Tools für die Umsetzung 60

05 Onboarding 61

06 eCube als Migrationspartner 63

07 Glossar 66

01 Einleitung

** Zu allen nachfolgend mit Stern gekennzeichneten Begriffen findet sich eine Erklärung im Glossar.*

Digitalisierung bedeutet heute, die Weichen für ein innovationsfähiges Business zu stellen. An die Stelle von monolithischen* Fertiglösungen treten flexibel orchestrierbare Software-Architekturen. Nicht Technologie bestimmt das Geschäft, sondern Technologie folgt den Anforderungen des Geschäfts.

Mit dem Composable Commerce* nehmen Unternehmen ihre Technologien selbst in die Hand. Fähigkeiten und Prozesse werden individuell zusammengestellt, um den größtmöglichen Geschäftswert zu schaffen. Das aktive Gestalten und Steuern bedeutet, Technologie neu zu denken und dynamisch zu entwickeln.

Klassische Vorgehensweisen im Projektmanagement wie die sogenannte Wasserfall-Methode stoßen in der neuen, dynamischen Welt des Composable Commerce an Grenzen. Hier können agile Methoden und Tools helfen, die Schwächen starrer, sequenzieller Planung und Umsetzung auszugleichen.

Dieser Leitfaden soll Projektleiter, Commerce-Verantwortliche und IT-Entscheider in die Lage versetzen, das Replatforming* eines monolithischen Systems hin zu einer flexibel orchestrierbare Software-Architektur mithilfe von agilen Methoden und Tools so zielgenau und gleichzeitig flexibel wie möglich zu steuern.

Nach einer kurzen Einführung in grundlegende Fragestellungen und Vorüberlegungen folgt ein Überblick über die Phasen eines Replatforming-Projekts – von der Organisation und Vorbereitung über die Planung bis hin zur Umsetzung. Dabei wird agile und Wasserfall-Methodik, wo möglich, sinnvoll kombiniert.

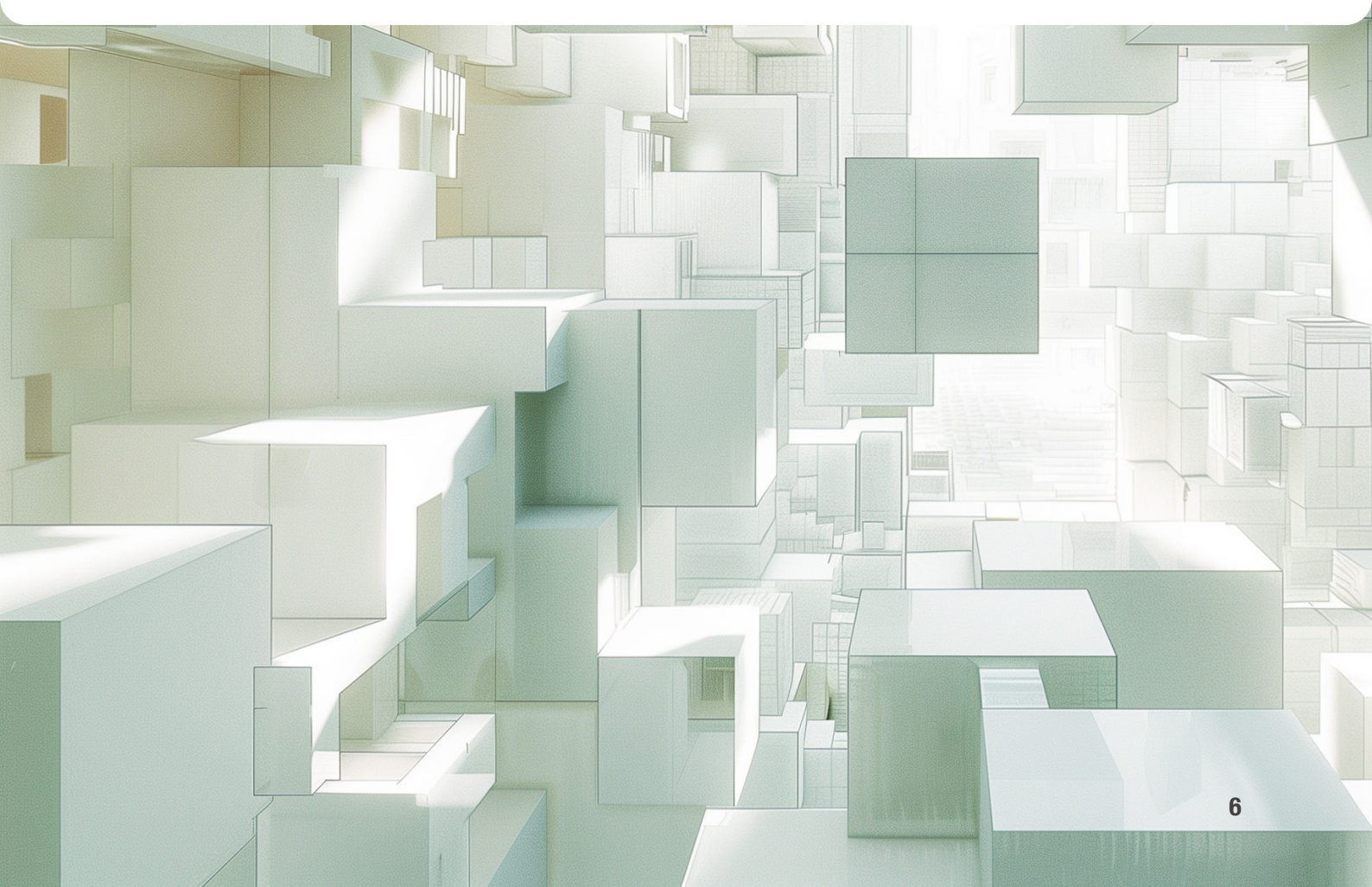
Wichtig: Dieser Leitfaden kann nur einen groben Überblick und Ideen für neue Arbeitsweisen geben. Bei Fragen zu Details unterstützen die Experten von eCube gerne in einem persönlichen Gespräch.

1. Replatforming

Der Begriff Replatforming bezeichnet das Überführen bestehender Software-Anwendungen und Funktionen von einem Altsystem zu einer zukunftssicheren (Cloud)-Plattform.

Vereinfachend kann man auch von einem Systemwechsel sprechen, wenn ein Unternehmen beispielsweise von einem monolithischen Komplettsystem zu einer modularen, flexibel orchestrierbaren Best-of-breed*-Architektur im Sinne des Composable Commerce migriert.

Wir verwenden die Begriffe Replatforming und Migration* im Folgenden synonym, um die Transformation von einer Systemarchitektur zu einer anderen zu beschreiben.



2. Gründe für Replatforming

Viele Unternehmen, die planen, ihr Altsystem abzulösen, wollen technische Schulden tilgen, die sich über Jahre angehäuft haben, besonders wenn diese Ressourcen binden und Innovation behindern.

Andere haben die strategische Notwendigkeit erkannt, jederzeit auf disruptive Krisen und Veränderungen im Markt reagieren und ihren E-Commerce inhaltlich und fachlich flexibel an neue Herausforderungen anpassen zu können.

Nicht zuletzt soll ein Replatforming Unternehmen dazu befähigen, sich aus Abhängigkeiten von einem Technologie-Anbieter (Vendor Lock-in*) zu lösen und nach dem Best-of-breed-Prinzip stets die besten Lösungen zu kombinieren.

3. Bedeutung des Replatformings für den Unternehmenserfolg

Wenn ein Altsystem den Anforderungen an den heutigen E-Commerce nicht mehr genügt, die Betriebskosten laufend steigen, ohne dass der geschaffene Mehrwert in gleichem Maß wächst oder das Risiko der Weiterentwicklung steigt, dann ist es Zeit für Veränderung.

Ein Replatforming bietet hier die Chance, die technologischen Weichen strategisch für ein zukunftssicheres Business zu stellen, statt nur von einem Komplettsystem zum anderen zu wechseln, ohne nachhaltig grundlegende, systematische Verbesserungen zu erreichen.

Ein Beispiel für eine solche nachhaltige Verbesserung ist die Migration von einem monolithischen Altsystem, etwa einer Software-Suite wie SAP Hybris oder Shopware, zu einer Composable-Commerce-Architektur auf Basis von commercetools oder Emporix.

Weitere Informationen im eCube-Magazin:
„Replatforming-Hub“

4. Grundlegende Fragestellungen

Das Ziel einer digitalen Transformation sollte es sein, mit möglichst geringem Zeit- und Kostenaufwand sowie möglichst geringem unternehmerischen Risiko die technologische Basis für die kontinuierliche Schaffung von messbarem Mehrwert zu legen.

Die Kernfrage lautet: Wie kann es gelingen, mit minimalem Aufwand maximalen Mehrwert zu generieren? Bezogen auf die technische Transformation des E-Commerce ergeben sich daraus folgende Fragestellungen, die in der Planungsphase beantwortet werden müssen.

4.1 Startpunkt: Frontend oder Backend?

Digitale Veränderungen sollten grundsätzlich darauf abzielen, mit möglichst geringem Aufwand größtmögliche Verbesserungen zu erreichen. Der Schlüssel dazu kann sowohl in der Präsentationsebene (Frontend) als auch der Funktionsebene (Backend) liegen.

Optimierungen des Nutzungserlebnisses (Customer Experience) betreffen primär das Frontend, im E-Commerce einen Online-Shop oder eine mobile App, wobei in der Regel auch Anpassungen im Backend, z. B. Content- und Datenmanagement nötig sind.

Ähnlich verhält es sich bei einer technischen Frontendoptimierung, um beispielsweise einen Online-Shop für Suchmaschinen zu optimieren. Neben Performance, Reaktionszeiten und Personalisierung im Frontend, spielen hier auch die Flexibilität und Struktur der Backend-Architektur eine Rolle. Das Thema Barrierefreiheit und andere gesetzgeberische Vorgaben wie Kennzeichnungspflichten müssen hier ebenfalls berücksichtigt werden.

Anpassungen der Systemarchitektur setzen dagegen hauptsächlich im Backend an, um beispielsweise funktionale Komponenten oder Systeme per API anzubinden. Im Einzelfall kann dies auch Implikationen für die Frontend-Architektur haben.

Je nach Zielsetzung und technischen Rahmenbedingungen muss im Einzelfall die günstige Einflugschneise – Frontend oder Backend – ermittelt werden, um die „Low Hanging Fruits“ der digitalen Veränderung zu ernten.

Entscheidungshilfe | Checkliste:

	FRONTEND	BACKEND
Marketing möchte schneller Aktionen durchführen können	☑	
Schlechter Suchmaschinen-Score (z. B. Core Web Vitals)	☑	
Zielgenauere Kundenansprache durch Personalisierung	☑	☑
Optimierung und Automatisierung von Prozessen		☑
Erweiterung um neue Funktionen und Prozesse		☑
Anbindung von SaaS-Services und Systemen per API		☑

Wichtig: Der Schlüssel zum Erfolg liegt in sehr vielen Fällen im Frontend eines Systems. Hier lässt sich unmittelbar und mit vergleichsweise geringem Entwicklungsaufwand viel Mehrwert für Kunden schaffen. Es lohnt sich also meist, hier anzusetzen.

Weitere Informationen im eCube-Magazin:
„Technische Frontendoptimierung ‚Perfect Storefront' (Teil 1)"
„Technische Frontendoptimierung ‚Perfect Storefront' (Teil 2)"

4.2 Frontend: Integriert oder Headless*?

In traditionellen Softwaresystemen sind häufig die Präsentationsebene und die Funktionsebene in einer monolithischen Architektur zusammengefasst. Sämtliche Geschäftsfunktionen spiegeln sich in einer gemeinsamen Code-Basis wider.

Das hat zur Folge, dass alle Softwarekomponenten in einem monolithischen System aufgrund der Datenaustauschmechanismen und der engen Verzahnung zwischen Frontend und Backend innerhalb des Systems voneinander abhängig sind.

Headless-Architekturen trennen Frontend und Backend voneinander. Beide Schichten kommunizieren über eine standardisierte Programmierschnittstelle (API), was die Entwicklung und Skalierung von Funktionen im Frondend vereinfacht und eine schnellere Time-to-market ermöglicht.

Der Headless-Ansatz bietet sich besonders bei Softwarearchitekturen an, bei denen das Frontend neu entwickelt, regelmäßig einfach und möglichst risikoarm (ohne Wechselwirkungen mit dem Backend) an neue Anforderungen angepasst oder weiterentwickelt werden muss.

Entscheidungshilfe | Checkliste:

	INTEGRIERT	HEADLESS
Anforderungen an das Frontend ändern sich nicht oder selten	☑	
Frontend wird vorerst nicht um weitere Kanäle erweitert	☑	
Ressourcen/Know-how für Frontend-Entwicklung fehlen	☑	
Anforderungen an das Frontend ändern sich häufig		☑
Verkaufskanäle sollen aktiv gemanaged und orchestriert werden		☑
Frontend soll flexibel erweitert und skaliert werden können		☑

Weitere Informationen im eCube-Magazin:
„Warum Headless Commerce für eine effiziente Omnichannel-Strategie unerlässlich ist“

4.3 Umfang: Replatforming oder Retrofitting?

Nicht jede Veränderung erfordert eine grundlegende Transformation der gesamten Software-Architektur oder -Infrastruktur. Eine gezielte Anpassung kann in vielen Fällen eine schnelle und risikoarme Alternative zum Replatforming sein.

Software-Retrofit* beschreibt ein Verfahren, bei dem ein System in Teilbereichen so modernisiert wird, dass es je nach Zielsetzung in kleinen, kontrollierten Schritten in ein besser wartbares oder flexibler anpassbares System überführt wird.

Auf diese Weise können Unternehmen ihre bestehenden Investitionen schützen und gleichzeitig die Leistungsfähigkeit und Nachhaltigkeit ihrer Technologien gezielt verbessern. Ein Beispiel für Retrofitting ist die Erweiterung um bestimmte Funktionen über die Grenzen des Systems hinaus.

Die Migration zu einer neuen Commerce-Architektur ist meist alternativlos, wenn die Möglichkeiten, notwendige Verbesserungen im Rahmen des bestehenden Systems oder in Form eines Retrofits vorzunehmen, begrenzt sind oder die Commerce-Technologie völlig neu ausgerichtet werden soll.

Entscheidungshilfe | Checkliste

	REPLATFORMING	RETROFIT
Bestehendes System beibehalten und optimieren		☑
Verbesserung per Versions-Update erreichbar		☑
Verbesserung per Funktions-Upgrade erreichbar		☑
Strategische Neuausrichtung der Architektur	☑	
Abhängigkeiten von Anbietern (Vendor Lock-in) beenden	☑	
Steigende Entwicklungs- und Betriebskosten (TCO)	☑	

Weitere Informationen im eCube-Magazin:

„Retrofit im E-Commerce: Wie Unternehmen ihre Technologien mit geringem Risiko und Aufwand nachrüsten“

„Systeme intelligent tunen, statt neu bauen: Beispiele für Software-Retrofitting im E-Commerce“

4.4 System: Neu bauen oder transformieren?

Veraltet, kompliziert, unflexibel – wenn monolithische Shopsysteme an ihre Grenzen stoßen, ist eine technologische Neuausrichtung nötig. Je nach Ausgangssituation und Zielsetzung kann dies durch kompletten Neubau („Big Bang”) oder durch Umwandlung erreicht werden.

1. Neuentwicklung einer neuen Commerce-Plattform und Ablösung der alten. Das ist die bevorzugte Option, wenn ein Umbau unwirtschaftlich oder technisch (nahezu) unrealistisch ist.

2. Schrittweise Transformation der bestehenden Plattform. Dazu wird das vorhandene, in der Regel monolithische System im laufenden Betrieb in eine modulare, Service-orientierte Architektur (SOA) umgebaut.

Bei beiden Lösungsansätzen erfolgt die Transformation schrittweise. Die Reihenfolge der einzelnen Entwicklungsschritte orientiert sich daran, welche Maßnahmen nötig und sinnvoll sind, schnell und mit geringem Risiko maximalen Geschäftswert zu schaffen.

Die Herausforderung besteht darin, kritisch zu analysieren, welcher Ansatz besser geeignet ist, um kurzfristige Projektziele wie Aufwand, Qualität und Dauer der Umsetzung (Time-to-market) bestmöglich mit strategischen Commerce-Zielen in Einklang zu bringen.

Entscheidungshilfe | Checkliste

	NEUENTWICKLUNG	TRANSFORMATION
Investitionen und Fachwissen sollen erhalten werden		☑
Systemwechsel soll im laufenden Betrieb erfolgen		☑
Verteilung von Kosten und Risiken auf kleine Veränderungen		☑
Schrittweise Transformation ist technisch unrealistisch	☑	
Technischer Neustart ohne Kompromisse	☑	
Grundlegende Veränderung des Geschäftsmodells	☑	

Weitere Informationen im eCube-Magazin:
„Monolithische Shopsysteme: Neu bauen oder schrittweise transformieren?“

4.5 Zielarchitektur: Monolith oder API-Plattform*?

Während der digitale Commerce bisher stark auf monolithischen Komplettsystemen basierte, setzen Unternehmen heute zunehmend auf modulare Plattformen, die sich flexibel nach individuellen Anforderungen ausrichten und modifizieren lassen.

Serviceorientierte Architekturen (SOA) sind kein neues Phänomen. Sie bilden jedoch zusammen mit Prinzipien wie API-first, cloud-native SaaS* und Headless das Fundament für neue Konzepte wie MACH* basierten Composable Commerce.

API-Plattformen wie commercetools oder Emporix ermöglichen es Unternehmen, digitale Fähigkeiten in Form von Services nach Bedarf flexibel zu orchestrieren und stehen damit im Gegensatz zu Komplettsystemen, die Funktionen in einem Monolithen vereinen.

API-first bedeutet, IT neu zu denken und in Besitz zu nehmen. Weg vom reinen Konsumieren, hin zum Gestalten von Lösungen. Weg von Best-of-suite hin zu Best-of-breed. Dafür müssen Unternehmen das nötige Know-how, Fähigkeiten und Ressourcen aufbauen.

Entscheidungshilfe | Checkliste

	MONOLITH	API-PLATTFORM
Anforderungen an den Digital Commerce ändern sich regelmäßig		☑
Integration von SaaS-Services und IT-Systemen per API		☑
Individuelle Anforderungen an den B2B-Commerce		☑
Standardanforderungen an den B2C-Commerce	☑	
Technologien müssen nicht individuell orchestriert werden	☑	
Ressourcen und Know-how für aktives Orchestrieren fehlen	☑	

Weitere Informationen im eCube-Magazin:
„Best-of-breed ist cool, aber nicht immer die beste Lösung“

4.6 Lösung: Best-of-breed oder Best-fit*?

Best-of-breed – das Kombinieren von bestmöglichen IT-Lösungen für bestimmte Geschäftszwecke – erlebt in Zeiten des Composable Commerce eine Renaissance. Falsch verstanden führt dieser Ansatz jedoch zu falschen Prämissen in der Technologie-Strategie.

Die Lösung, die als Best-of-breed im Markt gilt, ist nicht per se die beste Lösung, um spezifische Probleme in einem Unternehmen zu lösen oder strategische Ziele zu erreichen. Die Frage nach der besten Lösung ist immer aus Sicht des Unternehmens zu beantworten.

Dazu ist es zwingend notwendig, die konkrete Problemstellung, bzw. Zielsetzung in allen Facetten zu verstehen. Denn erst auf dieser Basis ist es möglich, die richtigen Anforderungen zu formulieren und die individuell passenden Technologien zu finden.

Checkliste für die systematische Bedarfsanalyse und Lösungssuche:

- Keine Lösung, ohne das Problem verstanden zu haben
- Verschiedene Lösungsvorschläge berücksichtigen
- Lösungskandidaten in die engere Wahl nehmen und prüfen
- Hintergründe und Motivation hinter einer Lösung verstehen
- Vorteile und Nachteile kritisch abwägen
- Favorisierte Lösung kritisch reflektieren

4.7 Umsetzung: Agil oder Wasserfall?

Die Wasserfall-Methode ist ein Klassiker im Projektmanagement und hat sich in vielen IT-Projekten bewährt. Bei komplexeren Projekten, in denen ein hohes Maß an Dynamik und Unsicherheit gemanagt werden müssen, stößt dieses Modell jedoch an Grenzen.

Anforderungen verändern sich regelmäßig im laufenden Prozess und Annahmen, die der Planung zugrunde liegen, stellen sich später in der Umsetzung als falsch oder als überholt heraus. Nicht selten fehlt von Beginn an ein gemeinsames Verständnis vom Endprodukt.

Das Risiko, dass sich die Entwicklung in die falsche Richtung bewegt, ist in komplexen Replatforming-Projekten groß. Deshalb bietet es sich an, solche Vorhaben nach agilen Prinzipien zu planen und zu steuern, statt statische Anforderungskataloge abzuarbeiten.

Sind agile Methoden unter bestimmten Bedingungen, z. B. in einem nicht-agilen Unternehmensumfeld, nicht möglich, bietet sich ggf. eine hybride Vorgehensweise an, bei der beide Konzepte – Wasserfall-Modell und agiles Projektmanagement – sinnvoll kombiniert werden.

Entscheidungshilfe | Checkliste

	AGIL	HYBRID	WASSERFALL
Komplexe Transformation in agilem Umfeld	☑		

Beispiel: Migration von einem monolithischen Altsystem zu Composable Commerce, Schneller Start mit MVP

Komplexe Neuentwicklung in nicht-agilem Umfeld		☑	

Beispiel: Agile Migration von einem integrierten Frontend zu Headless Commerce, Anforderungsmanagement per Lastenheft

Einfache Retrofit-Anpassungen am System			☑

Beispiel: System-Upgrade wegen Auslaufen des End-of-Life-Supports des Altsystems

Weitere Informationen im eCube-Magazin:
„Klassisches Projektmanagement ist nicht tot, aber unpopulär. Zu Recht?“

02 Organisation und Vorbereitung

Die Erfahrungen aus vielen IT-Projekten im Bereich Digital Commerce zeigen, dass Projekte besonders dann schieflaufen oder scheitern, wenn Projektziele, Anforderungen und Randbedingungen nicht klar definiert bzw. intern unzureichend kommuniziert werden.

Ein weiterer Stolperstein lauert in schwammig definierten Rollen und Zuständigkeiten im Projekt-Team – nicht selten fehlt ein Product Owner*, der den Hut aufhat und das undurchsichtige Gewirr an Fäden in komplexen Projekten zusammenhält.

Projektverantwortliche sind gefordert, im Rahmen der Planung maximale Klarheit bezüglich Zielsetzung, Randbedingungen und Ressourcenbedarf herzustellen, um sämtliche Herausforderungen und Risiken von Beginn an jederzeit im Blick zu haben.

1. Zielsetzung

Normalerweise starten klassisch sequenzielle Projekte mit einer Vereinbarung zwischen Sponsoren, Projektmanagement und bestimmten Interessengruppen über den Inhalt eines Projekts – die Projektcharta.

Darin werden das grobe Ziel und einige bereits bekannte Randbedingungen des Projekts festgehalten. Im agilen Umfeld wird statt der Projektcharta eine konkrete Vision für das Produkt bzw. Projektergebnis entworfen.

1.1 Smarte Ziele

Sowohl für die Projektcharta als auch für die Projektvision gilt, dass die definierten Ziele **SMART** – spezifisch (specific), messbar (measurable), erreichbar (achievable), relevant (relevant) und zeitgebunden (time-bound) – sein müssen.

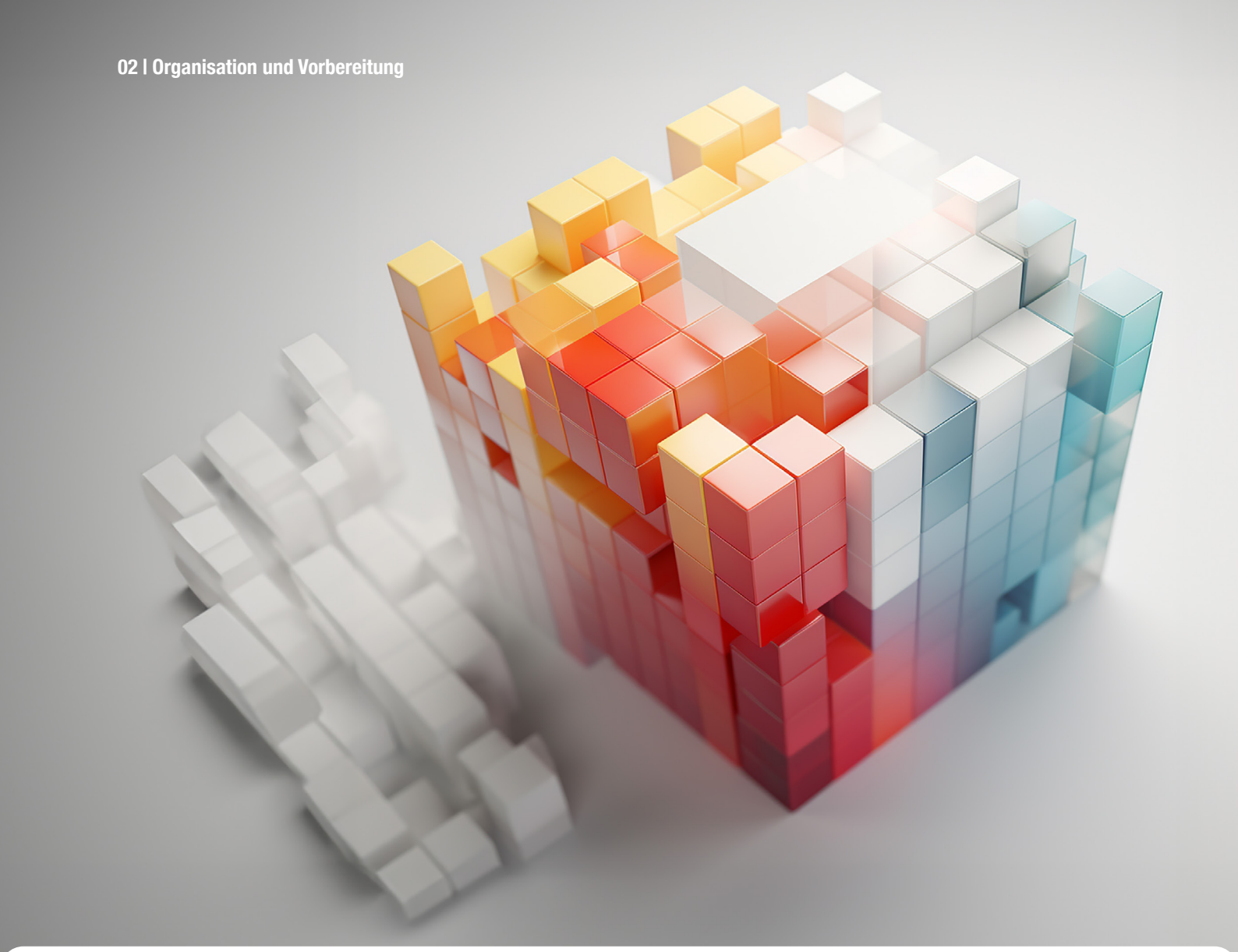
Die Zieldefinition sollte nicht nur relevante Anforderungen („Must-haves“) der Stakeholder enthalten, sondern auch Begeisterungsfaktoren. Diese werden in der Regel niedriger priorisiert, müssen aber ebenfalls den SMART-Kriterien ensprechen.

Beispiele für eine Projektcharta/Vision im E-Commerce

PROJEKT: ERNEUERUNG DES ONLINESHOPS

Sponsorin	Sabine Sponsor
Projektleiterin	Laura Leiter
Projektziel	Ablösung des alten Shopsystems durch ein neues System für: ■ technische Performancesteigerung um mindestens 70 % ■ Reduktion von Wartungskosten um mindestens 50 % ■ Mehrsprachigkeit im Shop ■ Verwendung von Wunschlisten
Bekannte Randbedingungen und Annahmen	■ Ein Dienstleister muss den Shop technisch einwandfrei halten. ■ Der Shop soll nicht von uns selbst gehostet werden. ■ Der Shop soll zu 99,98 % erreichbar sein. ■ Wir werden in den nächsten 5 Jahren um 300 % wachsen und brauchen entsprechende Prozesse innerhalb des neuen Shops.

Festzulegen, was erreicht werden soll, ist gerade zu Beginn eines Projektes nicht einfach. Die Herausforderung besteht darin, Ziele so SMART wie möglich zu formulieren und trotzdem genügend Spielraum für eine Konkretisierung im Projekt zu geben.



1.2 Minimum Viable Product (MVP)*

Grundsätzlich zielen agile Software-Projekte darauf ab, zunächst ein MVP zu entwickeln. Es hat einen minimalen Funktionsumfang und ermöglicht es, so schnell wie möglich einen messbaren Geschäftswert zu schaffen und für die Optimierung von Features und Prozessen zu lernen.

Anders als in Wasserfall-Projekten wird das Produkt ohne Meilensteine und ohne Projektphasen erreicht. Der erste „Meilenstein" ist quasi das Minimum Viable Product, das iterativ nach agilen Prinzipien bis zum gewünschten Ergebnis weiterentwickelt wird.

2. Projekt-Team

Der Projekterfolg steht und fällt mit der richtigen Zusammensetzung des Projekt-Teams. Diese in der Planungsphase zu ermitteln, ist eine zentrale Aufgabe der Projektleitung. Die Methodik ist je nach Vorgehensweise – agil oder klassisch sequenziell – unterschiedlich.

Typische Aufgaben und Fähigkeiten im Team:

- Entwicklung von Komponenten bzw. Features für die neue Plattform
- Redesign und Deaktivierung von Features im alten System
- Gängigen DevOps-Tätigkeiten, einschließlich Datenbank-Management, Monitoring, Alerting, Logging, Deployment, Servermanagement etc.
- Entwicklung einer eigenen API und Umgang mit APIs von Drittanbietern
- Dokumentation von Plattform-Modulen und Datenmodellen
- Systematische Migration von großen Datenmengen
- Releasemanagement
- Testing und Qualitätsmanagement

2.1 Agiles Projekt-Team zusammenstellen

Bei agiler Vorgehensweise empfiehlt es sich, für die ersten Entwicklungsstufen (Sprints) mit einem minimalen Team-Setup zu starten und die Zusammensetzung schrittweise zu erweitern bzw. zu verfeinern. Das Setup bleibt im gesamten Projektverlauf flexibel.

Beispiel: Projektteam für ein agiles Replatforming-Vorhaben

- **Product Owner:** Verantwortlich für die Definition, Priorisierung und kontinuierliche Weiterentwicklung der Produktanforderungen, um sicherzustellen, dass das Endprodukt den Anforderungen der Stakeholder entspricht.
- **Scrum Master*:** Verantwortlich für die Förderung und Umsetzung agiler Prinzipien und Praktiken im Scrum-Team, Beseitigung von Hindernissen, Erleichterung der Kommunikation und Voraussetzungen für effektive Arbeit im Team schaffen.
- **Development-Team mit 3–7 Mitgliedern:** Kollaborative Gruppe von Fachleuten, die flexibel und iterativ entwickeln, sich kontinuierlich an Veränderungen anpassen und schnell auf Anwender- oder Stakeholder-Feedback reagieren.

Die Zusammensetzung des Teams orientiert sich in agilen Projekten quasi in Echtzeit am Projektfortschritt und an möglichen Veränderungen hinsichtlich der Anforderungen und Rahmenbedingungen. Diese werden laufend analysiert und das Team-Setup angepasst.

Zu Beginn des Projekts lässt sich bereits nach wenigen Sprints per Release-Burndown erkennen, wie sich die Geschwindigkeit (Velocity) des Teams entwickelt und ob sich der sogenannte Product-Backlog* so entwickelt, wie es die Stakeholder erwarten.

2.2 Nicht-agiles Projektteam zusammenstellen

Die Bestimmung der Größe und der Fertigkeiten des Teams erfolgt im nicht-agilen Umfeld nicht im laufenden Projekt. Vielmehr werden die genauen Anforderungen des zu entwickelnden Produktes und damit das Team-Setup bereits vor dem Start festgelegt.

Die Anforderungen werden im Rahmen von Stakeholder-Interviews und Anforderungs-Workshops ermittelt und dokumentiert. Auf dieser Basis werden Projektmeilensteine in einer Work Breakdown Structure (WBS)* festgelegt.

In einem Netzplan* werden Abhängigkeiten dargestellt sowie die Projektdauer und der Ressourcenbedarf auf Basis von Erfahrungen aus anderen Projekten berechnet. Mithilfe eines GANTT-Diagramms* wird der geschätzte Ressourcenbedarf abschließend überprüft.

3. Kernkompetenzen im Projektmanagement

Ein erfahrenes Projektmanagement verfügt über einen umfassenden Baukasten an Methoden und Tools, die virtuos eingesetzt werden, um alle Fäden in der Produktentwicklung jederzeit zusammenzuhalten. Dabei spielen auch Soft Skills eine zentrale Rolle.

Wichtige Projektmanagement-Fähigkeiten:

1. Kommunikation:
Informationsfluss zwischen allen Projektbeteiligten sicherstellen

2. Empathie:
Bedürfnisse von allen Projektbeteiligten erkennen und verstehen

3. Erwartungsmanagement:
Erwartungen an das Produkt erfassen und aktiv steuern

4. Risikomanagement:
Projektrisiken identifizieren und bewerten

5. Geschwindigkeitsmanagement:
Tempo, Kosten und Qualität balancieren

6. Marktkenntnisse:
Entwicklungen und Trends im Markt und der IT-Welt kennen

Weitere Informationen im eCube-Magazin:
„Die 5 wichtigsten Projektmanagement-Skills im E-Commerce“

4. Anforderungen

Agile und nicht-agile Projekte unterscheiden sich in der Anforderungsanalyse erheblich voneinander.

Agiles Projekt: Das Scrum-Team startet mit der Umsetzung, sobald die erste User Story* von Product Owner, Entwickler, Tester und Benutzervertretern in enger Zusammenarbeit entwickelt und kommuniziert wurde.

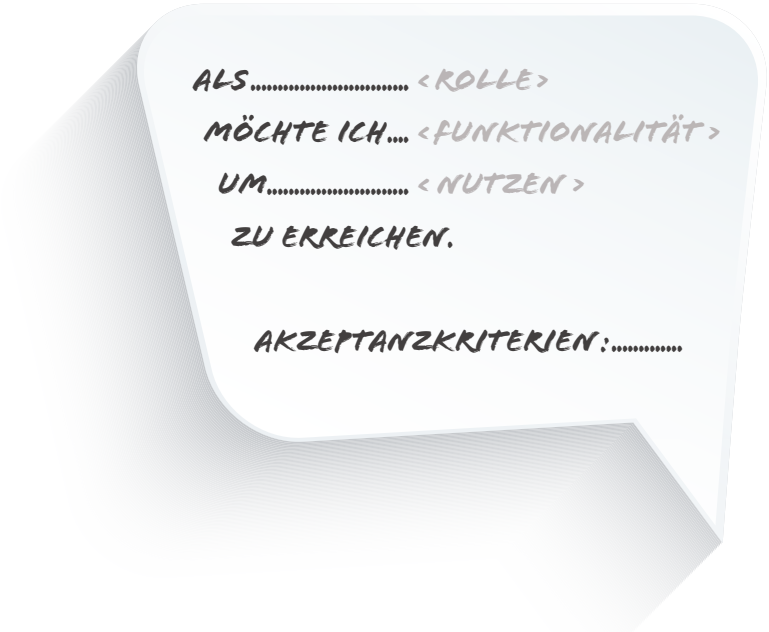
Wasserfall-Projekt: Die Projektleitung erfasst vor Beginn der Umsetzung alle Anforderungen an die Funktionalität und die Qualität so minutiös wie möglich, analysiert diese und stellt sie in einem Lastenheft zusammen.

4.1 User Stories in agilen Projekten

User Stories sind Software-Anforderungen, die aus der Sicht des Anwenders formuliert werden. Sie haben sich in der nutzerorientierten Softwareentwicklung als wirksames Werkzeug etabliert und schaffen wertvolle Klarheit in einem Replatforming-Projekt.

Wichtig: User Stories dienen nicht dazu, Randbedingungen und Qualitätsanforderungen, wie etwa die Geschwindigkeit der API oder die Sprache der Systeme, abzubilden. Solche Anforderungen werden in sogenannten Qualitätsszenarien* für das Projekt erfasst.

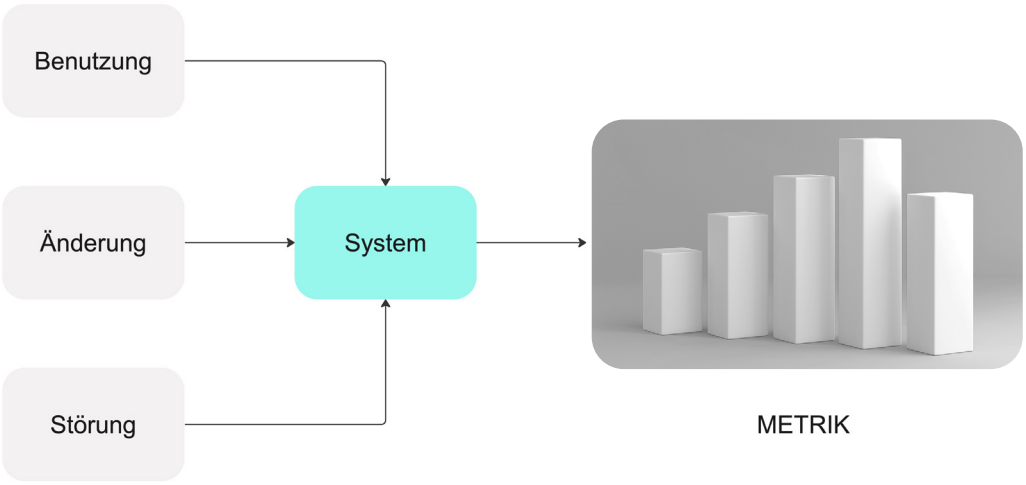
Vorlage für eine User Story



4.2 Qualitätsszenarien

Ein Qualitätsszenario in der agilen Softwareentwicklung ist eine präzise Beschreibung einer funktionalen oder nicht-funktionalen Anforderung an die Software, die die Erwartungen der Stakeholder widerspiegelt und als Grundlage für Tests und Bewertungen dient.

Grundprinzip von Qualitätsszenarien



Beispiele für Qualitätsszenarien im E-Commerce:

- Der Warenkorb soll immer innerhalb von 500 ms alle Produkte geladen haben.
- Der PageSpeed Score von Google muss über 90 Punkte sein.
- Das Ausrollen einer neuen Software-Version darf nicht länger als 5 Minuten dauern.
- Die Beschreibung aller Artikel muss mindestens 100 Zeichen lang sein.

4.3 Kommunikation

Wurden die ersten User Stories verfasst, die Qualitätsszenarien zusammengestellt und mit dem Entwicklungsteam besprochen, kann die Produktentwicklung starten. Entscheidend ist hier eine kontinuierliche, engmaschige Kommunikation mit allen Projektbeteiligten.

Nach der Sprintplanung werden die Stakeholder vom Umfang des ersten Sprints in Kenntnis gesetzt und können von nun an transparent Woche für Woche den Fortschritt, die Entwicklung in Review-Meetings, Planungsprotokollen und dem Produkt-Backlog verfolgen.

Wichtig ist, Stakeholder zum richtigen Zeitpunkt mit den richtigen Informationen zu versorgen. Die Kommunikation von Mensch zu Mensch oder ein gemeinsamer Chatroom bieten die Möglichkeit des schnellen Feedbacks und ist deshalb häufig effektiver als die Kommunikation per E-Mail.

Über die Abonnement-Funktion von Anforderungsmanagement-Tools wie Jira, Jama oder ReqSuite RM bleiben die Stakeholderinnen bequem über Änderungen im Sprint-Umfang oder im Produkt-Backlog auf dem Laufenden.

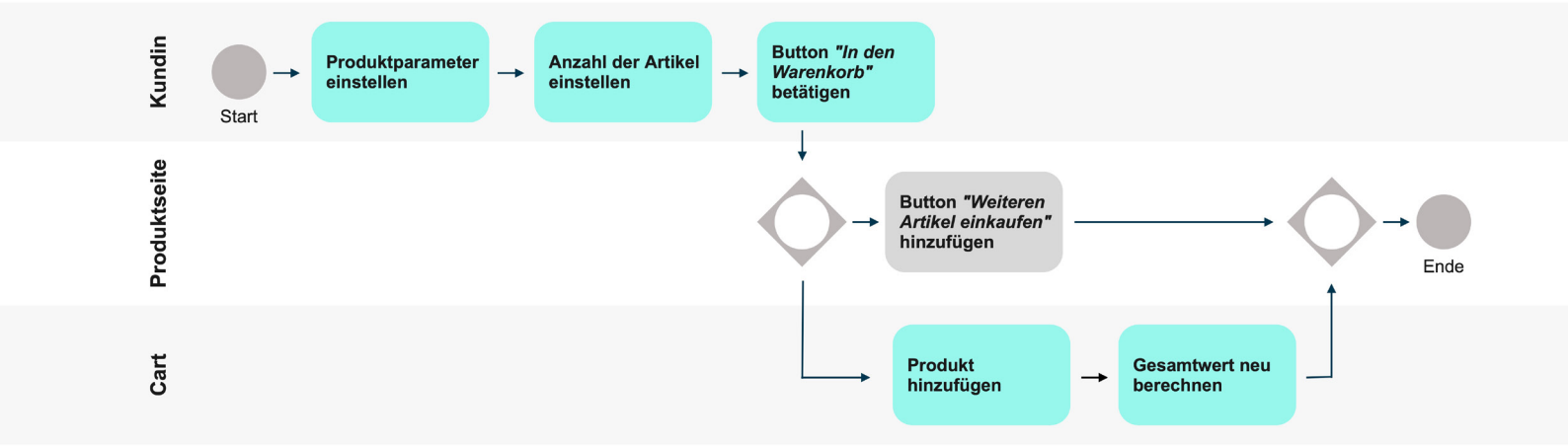
Umfangreiche Änderungen, die die Planung oder den Feature-Umfang betreffen, sollten mit individuell aufbereiteten Informationen separat kommuniziert werden. Ergänzend sollten Möglichkeiten zum Feedback und zum Dialog im Team geschaffen werden.

5. Klassisches Lastenheft in nicht-agilen Projekten

Auch Projekte in nicht-agilen Umgebungen können von User Stories profitieren. Üblich sind diese jedoch für Lasten- und Pflichtenhefte in klassischen Wasserfall-Projekten nicht. Ein Lastenheft, das die Projektleitung verfasst, umfasst in der Regel die folgenden Inhalte:

- **Skizze der IST-Situation:** Beschreibung des Status quo mithilfe von Story Mappings und einer Systemübersicht. Dokumentation gängiger Features auf Basis von Anwender-Interviews und -Workshops und der Dokumentation des Altsystems.
- **Beschreibung der SOLL-Situation:** Definieren der Anforderungen an die Software mithilfe gängiger Requirements-Engineering-Standards wie z. B. Unified Modeling Language (UML) oder der Business Process Model and Notation (BPMN)*.

Beispiel für ein BPMN: Produkt in den Warenkorb legen

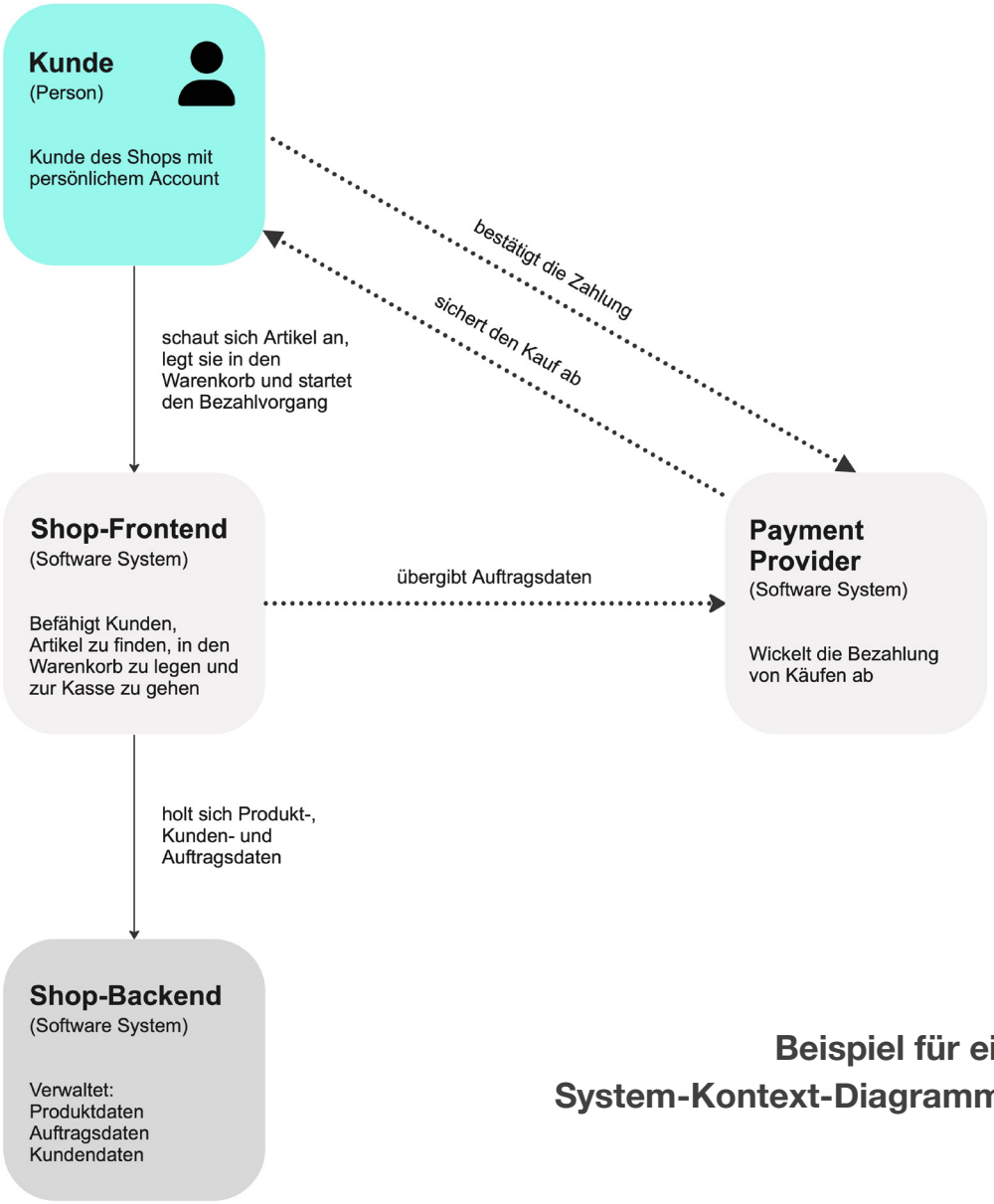


- **Dokumentation der funktionalen und Qualitätsanforderungen:** Beschreibung der Systemanforderungen nach Standards des International Requirements Engineering Board (IREB), die sich auch für die Beschreibung der Qualitätsanforderungen eignen.
- **Sonstiges:** Rechtliche Rahmenbedingungen, Servicelevel-Vereinbarungen, Regelungen zur Abnahme, Glossare etc.

Aus dem Lastenheft entwickelt das Team, das das Replatforming umsetzen wird, ein Pflichtenheft. Dieses hat den Anspruch auf Vollständigkeit und soll nach seiner Abnahme nicht mehr verändert werden, was Projekte in der Regel sehr unflexibel macht.

Änderungen in den Anforderungen an eine neue E-Commerce-Plattform, die während der Entwicklung vorgenommen werden, sind in der Regel mit Aufwand und Kosten verbunden, wenn Milestones im Rahmen des Changemanagements verändert oder sogar rückgängig gemacht werden müssen.

Deshalb soll der Übergang vom Lasten- zum Pflichtenheft Klarheit darüber schaffen, ob das Team die Anforderungen verstanden und technisch einwandfrei geplant hat. Normalerweise durchlaufen Lastenhefte mehrere Iterationen, was viel Zeit bis zum Projektstart kosten kann.



Beispiel für ein System-Kontext-Diagramm*

Praxistipp: Da sich auch bei minutiös geplanten sequentiellen Projekten im Laufe der Entwicklung oft Änderungen in der Software-Architektur ergeben, sollten diese per Use-Case-Diagrammen* oder System-Kontext-Diagrammen skizziert werden.

Kommunikation

Das Pflichtenheft sollte während des gesamten Projektes für alle Mitarbeitenden des Unternehmens öffentlich zugänglich sein, damit alle jederzeit die Möglichkeit haben, wertvolle Hinweise auf mögliche Planungsfehler oder fehlende Anforderungen beizusteuern.

6. Tools für die Organisation und Vorbereitung

Zur konkreten Umsetzung eines Projekts sollte man sich der im Unternehmen vorhandenen Tools bedienen. Zusätzlich kann es nötig sein, hier auch bei Bedarf nachzurüsten. Insbesondere aktuelle, cloudbasierte Tools und Services können da enorm nützlich sein. Die folgende Übersicht gibt hier eine Orientierung für die jeweils benötigten Bereiche.

Dokumentation	Microsoft Office365 Sharepoint Atlassian Confluence Google Docs
Planung	Microsoft Office365 Miro Atlassian Jira Google Docs Trello
Kommunikation	Microsoft Teams Zoom

03 Planung

In Abschnitt 2 wurde erläutert, wie Unternehmen je nach Arbeitsweise und Philosophie entweder ein klassisch-sequenzielles Projekt oder die agile Produktentwicklung organisatorisch managen. Dabei wurde zwischen agil und nicht-agil unterschieden.

In der Praxis lässt sich diese strenge Unterscheidung häufig nicht aufrechterhalten – etwa wenn beispielsweise in einem nicht-agilen Unternehmen die Entwicklung für den E-Commerce oder Teile davon von einem externen, agil-arbeitenden Team umgesetzt wird.

1. Agil und Wasserfall sinnvoll verbinden

Für die agile Umsetzung in einem nicht-agilen Umfeld müssen Wege gefunden werden, beide Ansätze sinnvoll zu kombinieren. Die Lösung liegt in einem hybriden Vorgehen, das im Idealfall bereits in der Vorbereitung des Vorhabens beginnt.

Wichtig: Der hier skizzierte hybride Ansatz ist kein Optimum, sondern ein gangbarer Kompromiss für den Fall, dass die rein agile Umsetzung eines Software-Projekts – aus welchen Gründen auch immer – nicht möglich ist.

Waterscrum*

Die Idee, agiles und nicht-agiles Vorgehen sinnvoll zu kombinieren, um das Beste aus beiden Welten für die Umsetzung komplexer Projekte nutzen zu können, ist nicht neu. Sie findet sich beispielsweise in der Waterscrum-Methode wieder.

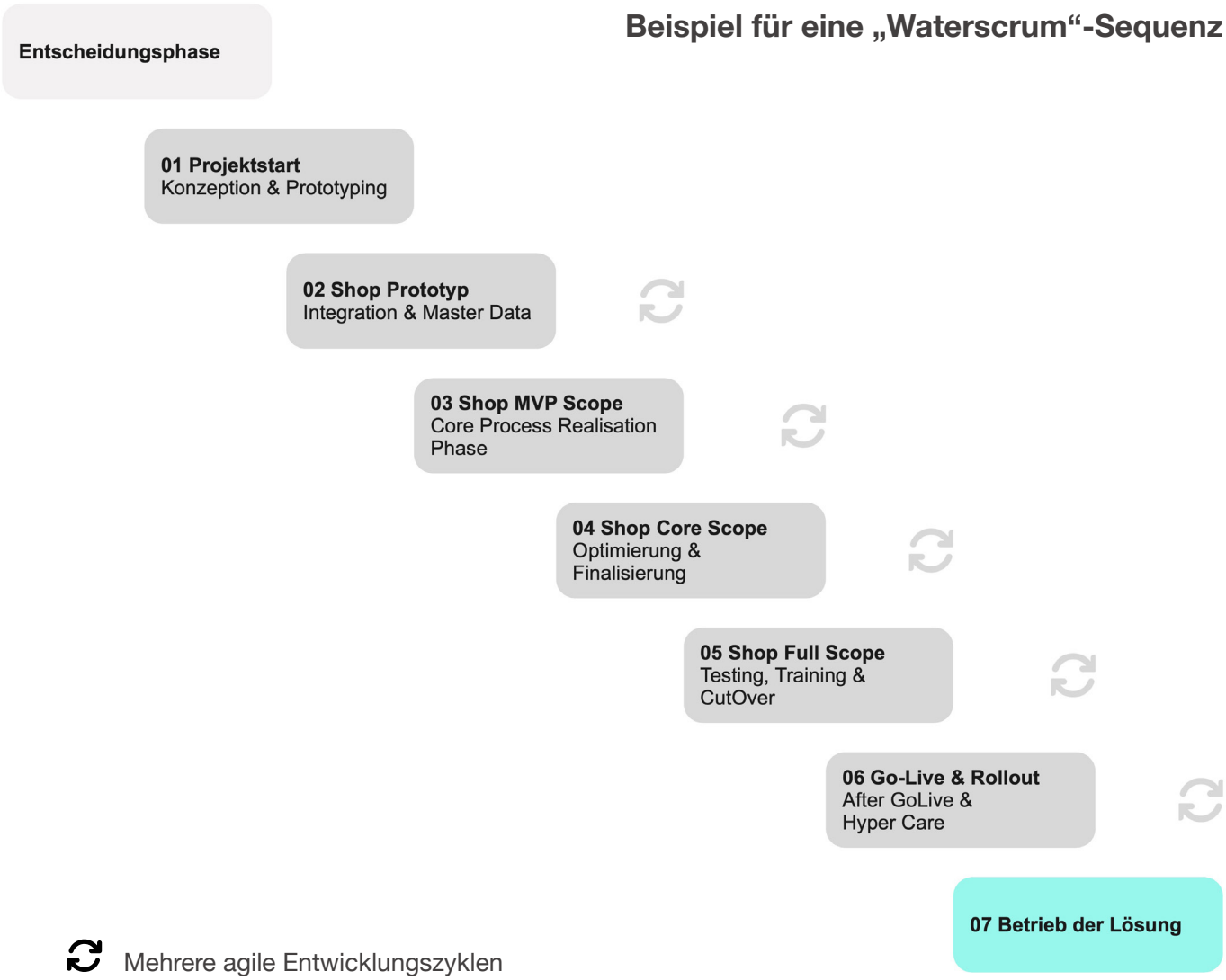
Diese sieht im Sinne des Wasserfall-Ansatzes aufeinanderfolgende Projektphasen vor. Die Inhalte der einzelnen Projektphasen werden vor dem Projektstart sinnvoll gewählt und zugeordnet, die Reihenfolge kann jedoch im Projektverlauf verändert werden.

Innerhalb der einzelnen Phasen erfolgt die Umsetzung nach agilen Prinzipien. Die zugeordneten Inhalte einer Phase werden in kurzen Sprints realisiert und stehen am Ende der jeweiligen Projektphase zum Testen zur Verfügung.

Im Gegensatz zum klassischen Wasserfall-Vorgehen ist die Waterscrum-Sequenz in User Stories aufgeteilt, die in einer definierten Abfolge umgesetzt werden. Damit können Projektleiter zwei zentralen Anforderungen bzw. Erwartungen gerecht werden:

1. Auftraggeber und Stakeholder definieren zu Beginn des Projektes ihre Anforderungen und Erwartungen und schaffen (im Idealfall) ein gemeinsames Verständnis vom Endprodukt.

Beispiel für eine „Waterscrum“-Sequenz



2. Projektleitung und Entwicklung erhalten die notwendige Flexibilität, die Projektstruktur – Reihenfolge und Umfang der Projektphasen – bei Bedarf im laufenden Prozess anzupassen.

Erfahrene Software-Dienstleister sind in der Lage, agiles und nicht-agiles Vorgehen sinnvoll zu kombinieren, um die optimale Vorgehensweise für individuelle Anforderungen wählen zu können. Die Methodik sollte stets den Anforderungen im Einzelfall folgen, nicht umgekehrt.

2. Business Owner* einbinden

Digitale Geschäftsprozesse sollten nicht auf der Basis von theoretischen Annahmen geplant werden. Neben den Stakeholdern und Sponsoren eines Replatforming-Projekts sollten deshalb auch Business-Owner aus den beteiligten Fachbereichen ihr Wissen beisteuern.

In E-Commerce-Projekten sind dies je nach Unternehmensgröße und -struktur Business-Owner aus den Bereichen Einkauf, Logistik, Category- und Datenmanagement, Auftragsqualifizierung und Kundenservice sowie IT, Marketing und Vertrieb. Im Idealfall gelingt es der Projektleitung für jeden Business-Prozess, der im neuen System abgebildet werden soll, eine Person aus dem entsprechenden Fachbereich als „Beirat" und Unterstützung für die fachliche Planung zu gewinnen.

Beispiel für Digitale Geschäftsprozesse mit fachlichem Beirat (blau)

Michael Müller	Sabine Schmidt	Robert Richter	Stefan Schneider	Lisa Lehmann	Katharina Krause
1	2	3	4	5	6
Einkauf	**Produkt-präsentation**	**Marketing & Vertrieb**	**Kaufab-wicklung**	**Logistik**	**Kunden-service**
	Produktdaten		Aufträge	Versand	
	User Interface		Payment	Retoure	

3. Anforderungen strukturieren

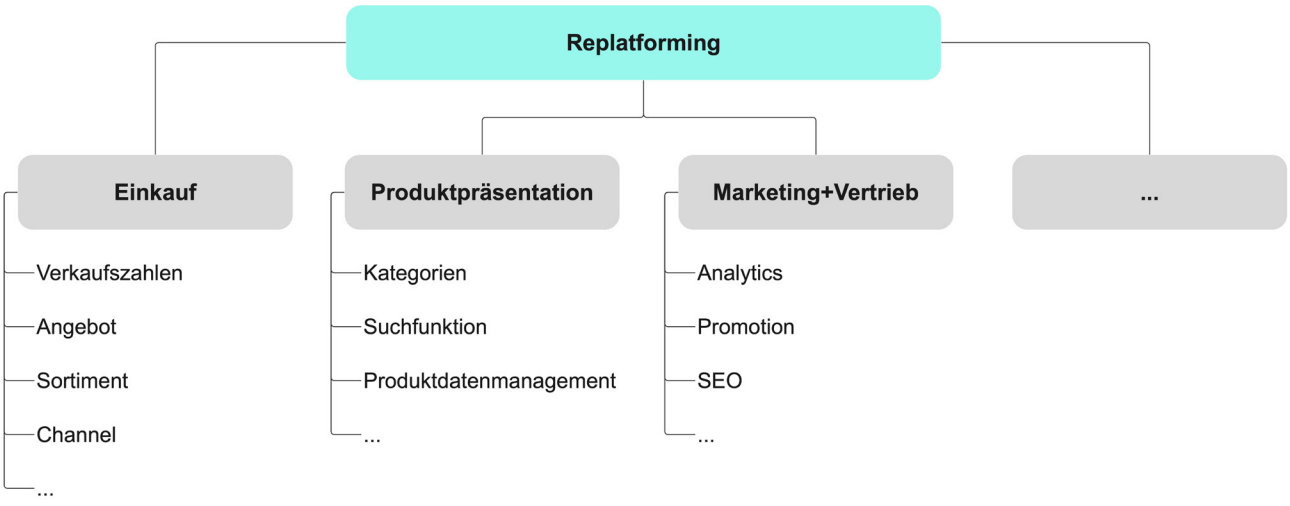
3.1 Epics* entwickeln

Epics bezeichnen eine übergeordnete Struktur, in der User Stories gebündelt werden. Sie dienen dazu, Anforderungen skalierbar und leichter organisierbar zu machen. Ein Epic „Suchfunktion“ umfasst beispielsweise alle User Stories im Kontext der Produktsuche.

In Interviews und Workshops mit Stakeholdern und Business-Ownern werden die „Themen” für die Softwareentwicklung gesammelt und in Epics gebündelt. Das Ziel ist es, eine möglichst vollständige Themensammlung mit groben Beschreibungen zu erstellen.

Das Ergebnis wird mithilfe einer groben Work Breakdown Structure (WBS) dokumentiert, die sich an den ermittelten Business-Prozessen orientiert. Jedem Business-Prozess werden die zu Epics gebündelten Themen zugeordnet.

Beispiel-WBS für ein Replatforming-Projekt



3.2 Ressourcenbedarf und Projektdauer ermitteln

Auf der abgestimmten Work Breakdown Structure werden im Entwicklungsteam grob die Entwicklungsressourcen für jedes Epic abgeschätzt – Arbeitswochen pro Person genügen. Es geht dabei vor allem darum, Epics in Größenordnungen untergliedern zu können.

Für die Schätzung der Aufwände bietet sich das sogenannte Planning Poker bzw. Scrum Poker an – eine agile, spielerische Methode, bei der Mitglieder des Entwicklungsteams gemeinsam die Komplexität der zu entwickelnden Aufgaben quantifizieren.

Summe der Arbeitswochen aller Epics / Teamstärke = Netto-Projektdauer

Auf Basis der Schätzungen für die benötigten Ressourcen bei einer gegebenen Größe des Teams lässt sich sehr einfach die Projektdauer abschätzen. Lassen sich damit gewünschte Fristen für die Fertigstellung nicht einhalten, muss das Team ggf. vergrößert werden.

3.3 Projektkosten

Anhand der kumulierten Entwicklungsdauer aller Epics lassen sich die Personalkosten eines Projekts grob ermitteln. Bei einem komplexen Replatforming-Projekt im E-Commerce machen diese in etwa 90 – 95 % der gesamten Projektkosten aus.

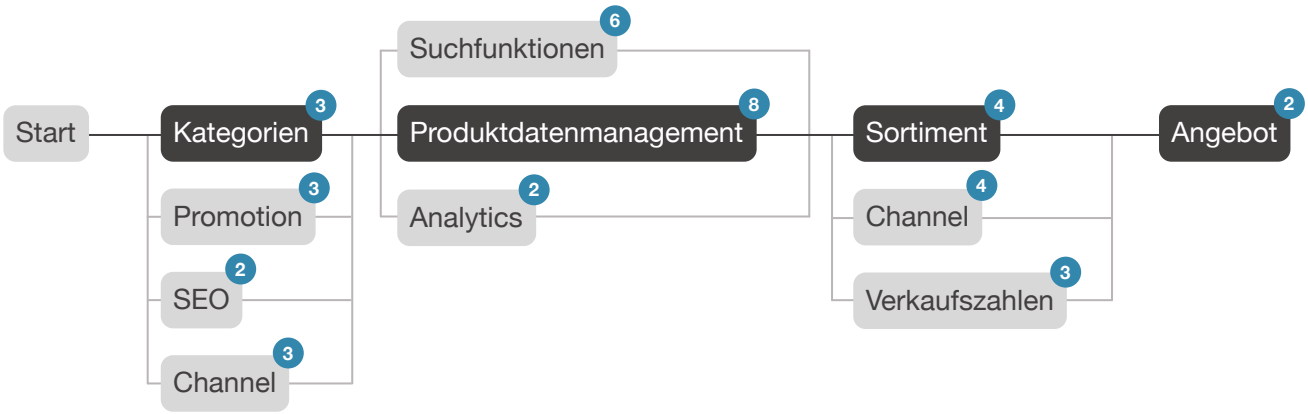
3.4 Epics planen und priorisieren

Wie die Projektdauer wird auch die Entwicklungsdauer für einzelne Epics geschätzt, um diese in eine sinnvolle Reihenfolge für die Entwicklung zu bringen. Manche Epics können oder müssen parallel bearbeitet werden, wenn sie gemeinsame User Stories beinhalten.

Aus der vorläufigen Reihenfolge der Epics ergibt sich ein erster Netzplan für die Umsetzung, der im weiteren Scrum Prozess vom Product Owner laufend angepasst wird. Der Netzplan bildet in hybriden Projekten die Schnittstelle zwischen agilem und nicht-agilem Vorgehen.

Der Netzplan umfasst die obligatorischen Schritte, um die Basis- und die kritischen Leistungsfaktoren eines Zielsystems zu implementieren und vereinfacht es, kritische Entwicklungspfade im Projekt frühzeitig zu erkennen.

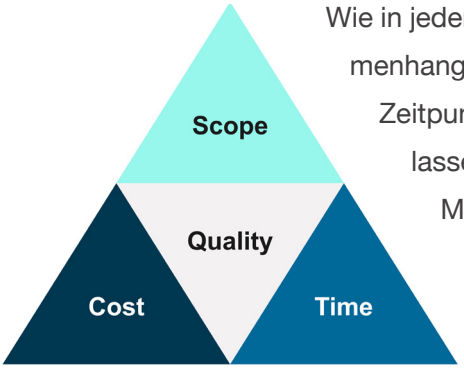
Beispiel für Netzplan mit kritischem Pfad (dukelgrau) und geschätzten Aufwänden (blau)



Wichtig: Anders als in Wasserfall-Projekten orientiert sich die Reihenfolge der Implementierung in agilen Projekten nicht an der Relevanz von Features, sondern am Geschäftswert von User Stories, z. B. Kosteneinsparung oder strategischer Nutzen.

Weitere Informationen im eCube-Magazin:
„Feature Priorisierung vs. Business Value“

3.5 Scope* anpassen



Wie in jedem Software-Projekt besteht auch im Replatforming ein direkter Zusammenhang zwischen Projektumfang (Scope), Zeit und Kosten. Häufig sind der Zeitpunkt der Fertigstellung und der Kostenrahmen für das Projekt fix oder lassen wenig Spielraum. In diesem Fall bleibt der Projektleitung nur die Möglichkeit, den Projektumfang auf das Wesentliche zu reduzieren. Um den kritischen Pfad zu verkleinern, müssen die Basis- und Leistungsfaktoren frühzeitig auf Ihre Wichtigkeit für den Geschäftserfolg geprüft werden.

Zusammenhang zwischen Zeit, Umfang und Kosten

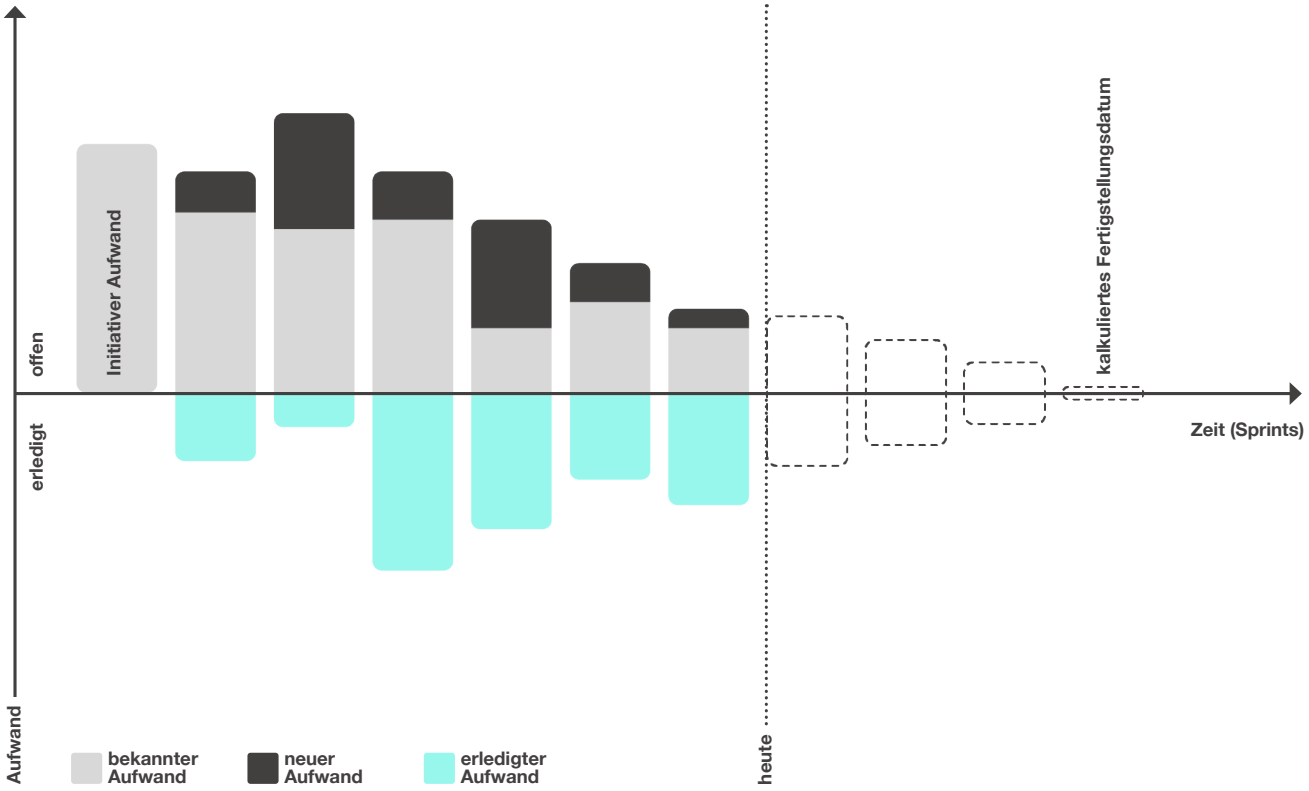
Müssen sowohl der Projektumfang als auch der Termin für die Fertigstellung gehalten werden, bleibt nur das Team zu vergrößern. Müssen dafür gegebenenfalls externe Entwickler ergänzt werden, führt dies logischerweise zu höheren Kosten.

3.6 Fertigstellungstermin ermitteln

Agile Software-Projekte starten mit einer groben Annahme für die Projektdauer, die dann mithilfe realer Projektdaten konkretisiert wird. In Abhängigkeit von der Arbeitsgeschwindigkeit des Teams lässt sich der Fertigstellungstermin prognostizieren.

Wird die Geschwindigkeit des gesamten Teams laufend auf Epic-Basis gemessen, ist die Projektleitung jederzeit in der Lage, Ressourcen in Richtung kritischer Themen verschieben. Ein Release- bzw. Epic-Burn-Down gibt wertvolle Hinweise, ob die Planung eingehalten wird.

Beispiel für eine Prognose des Fertigstellungstermins per Epic-Burn-Down-Chart

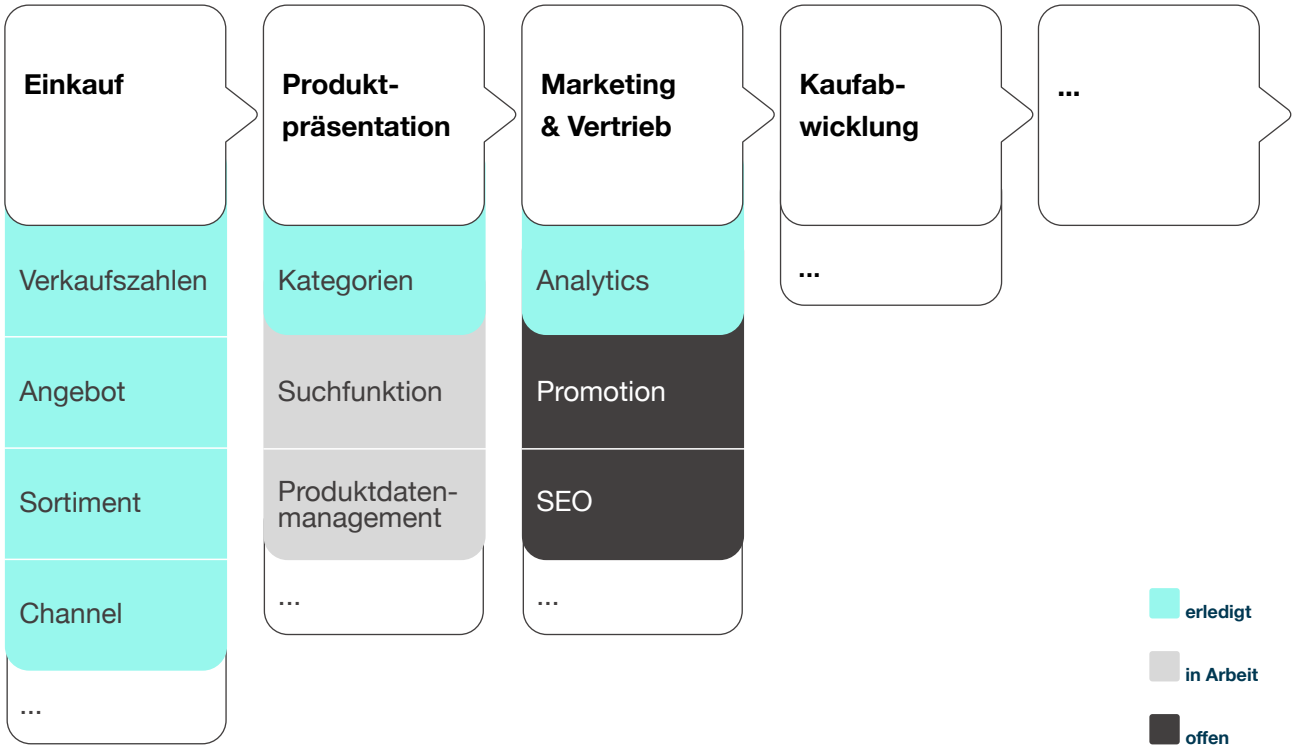


4. Kommunikation

Erfolgreiche Projektkommunikation besteht nicht nur darin, Informationen und Entscheidungen rechtzeitig auf die richtige Art und Weise zu kommunizieren. Es geht insbesondere auch darum, Erwartungen von Stakeholdern und Team richtig zu steuern.

- **Entwicklung visualisieren:** Da sich der Erwartungshorizont besonders in komplexen Software-Projekten laufend ändern kann, empfiehlt es sich, diese Dynamik mit entsprechend dynamischen Möglichkeiten der Visualisierung – Backlog in Verbindung mit GANTT-Diagramm – abzubilden.
- **Planung kommunizieren:** Neben dem Gesamtprojektplan, der sich besonders in agilen Projekten regelmäßig ändert, sollten alle Stakeholder vertikal und horizontal laufend per Backlog darüber informiert werden, woran das Team in den laufenden zwei bis vier Wochen arbeitet.
- **Ergebnisse gemeinsam verifizieren:** Zum Abschluss von Sprints finden regelmäßig sogenannte Sprint-Review statt, in denen die Stakeholder die bearbeiteten User Stories bzw. Features fachlich verifizieren und abnehmen. Dies wird für alle Beteiligten zugänglich dokumentiert.
- **Gesamtfortschritt visualisieren:** Mithilfe eines GANTT-Diagramms wird der Fortschritt des Projekts getrackt und visualisiert. Zusätzlich werden Prozess- und Architekturskizzen des E-Commerce-Systems mit Statusinformationen angereichert, um den aktuellen Projektstatus zu kommunizieren.

Beispiel für Projektstatus aus Prozesssicht

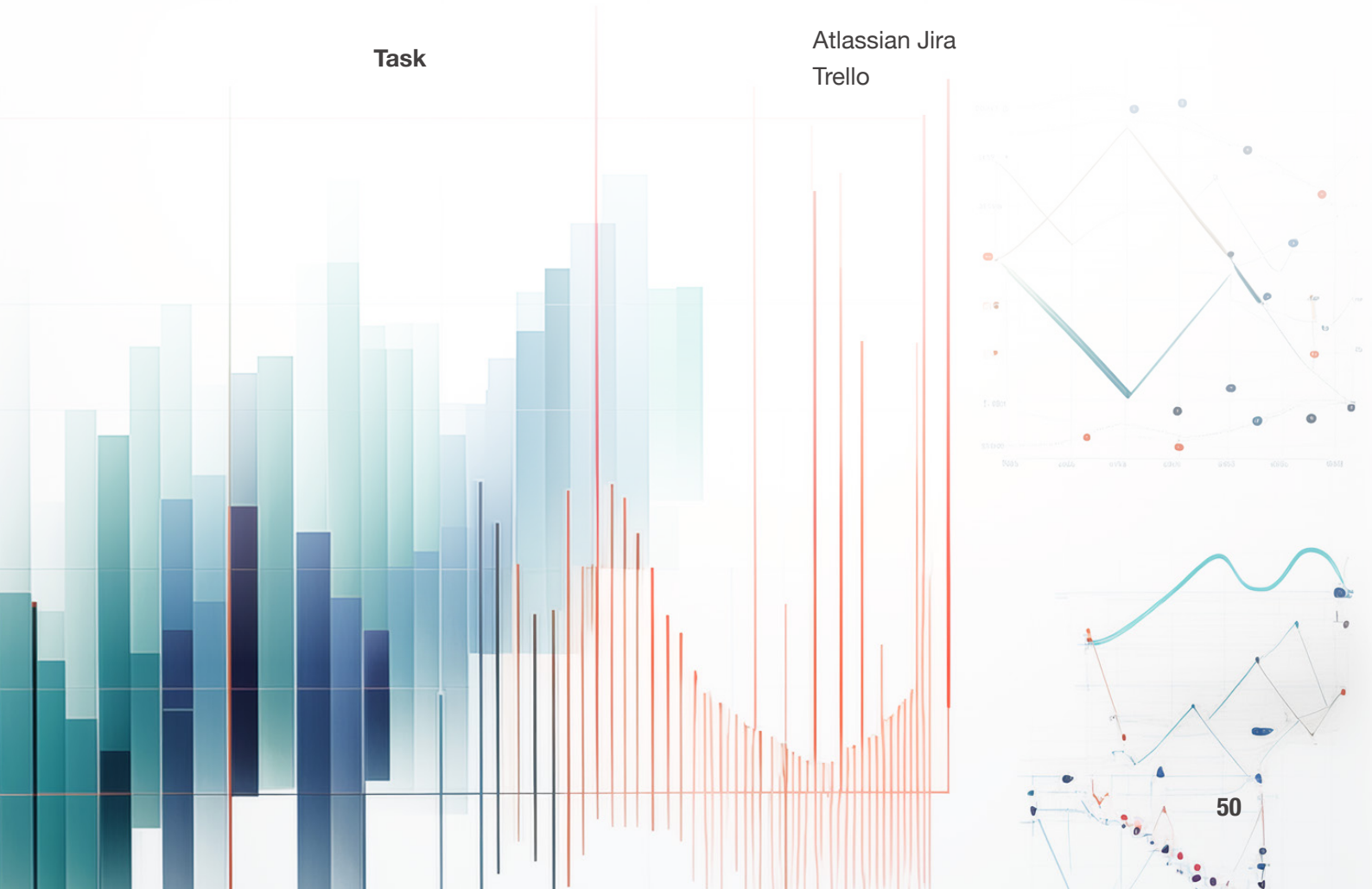


5. Tools für die Planung

Projektmanagement-Tools wie Microsoft Project vereinfachen die Erstellung und das Management von Netzplänen. Diese können bei Bedarf auf Knopfdruck in GANTT-Diagramme für sequenzielle Vorgehensweisen umgewandelt werden.

Für agile Projekte bietet sich der Einsatz von Anforderungsmanagement-Tools wie Asana oder JIRA an. In diesen Tools werden die Anforderungen – User Stories und Epics – agil in einem Backlog gesammelt und abgearbeitet.

Planung	Microsoft Office365 Miro Atlassian Jira Asana Google Docs Trello
Task	Atlassian Jira Trello



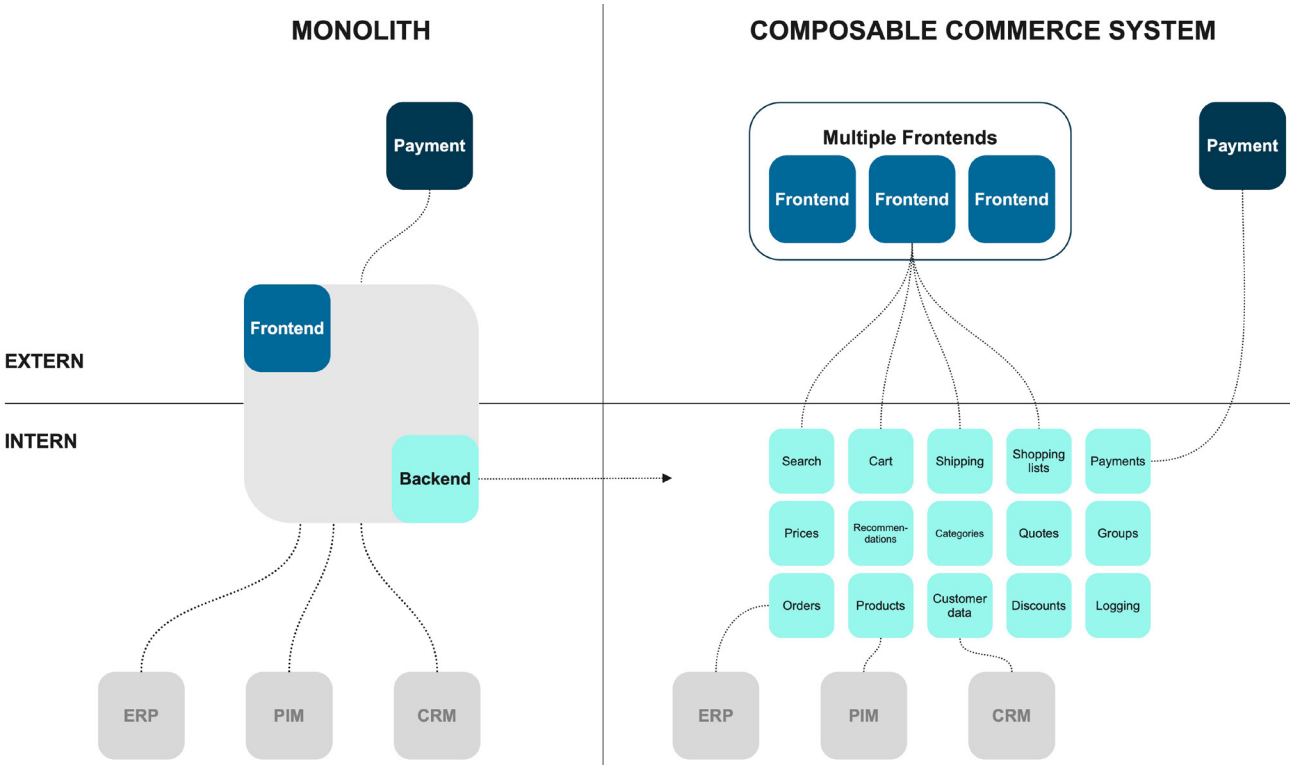
04 Umsetzung am Beispiel

1. Migration vom Monolithen zu Composable Commerce

Steht die Vision und sind die wesentlichen Rahmenbedingungen und die ersten Schritte für die Entwicklung geklärt, kann die Umsetzung beginnen. Im folgenden Beispiel wird ein monolithisches Altsystem durch ein flexibel orchestrierbares E-Commerce-System ersetzt.

Ein Altsystem, bei dem Daten, Funktionen und Prozesse in einer monolithischen Software vereint sind, ist im Laufe der Zeit zu starr und zu anfällig geworden, um mit den Anforderungen an modernen E-Commerce schritthalten zu können.

Ein modernes Composable-Commerce-System, bei dem alle Architekturkomponenten im Frontend und Backend nach Bedarf, flexibel orchestriert werden können, soll das alte System möglichst schnell in einem schrittweisen Prozess ablösen.



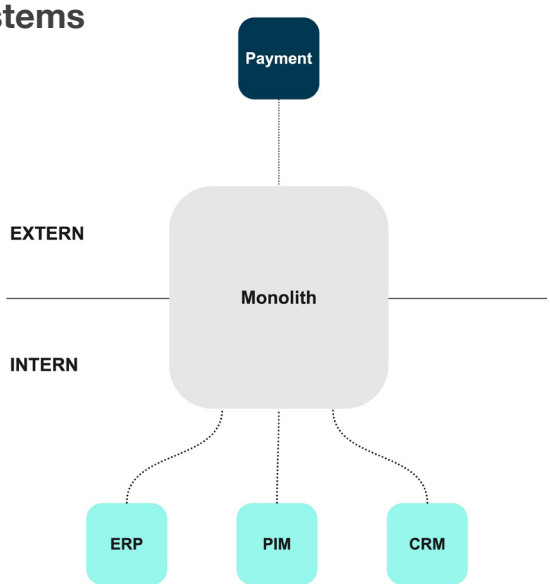
1.1 Motivation für das Replatforming

- Der Shop kann nicht mehr ohne hohen Aufwand erweitert werden.
- Neue Produkte stellen Anforderungen an den Monolithen, denen er nicht gerecht werden kann.
- Das System skaliert nicht ausreichend.
- Die Wartung des Monolithen ist mittlerweile zu kostenintensiv.
- Es sollen zusätzliche Endpunkte für den Verkauf angeschlossen werden, wie native Smartphone Apps oder IOT-Applikationen.

1.2 Annahmen zum Status quo

- Bestehendes System verfügt über eine API.
- Persistente Daten werden Drittsysteme verwaltet (ERP, PIM, CRM, PSP).
- Drittsysteme sollen im neuen System vorerst beibehalten werden.
- Shop soll ein neues, flexibles Frontend erhalten.

Architektur des Altsystems



2. Iterative Transformation

Viele Änderungsprozesse verlieren ihr Risiko, wenn sie Schritt für Schritt, statt mit einem großen Knall durchgeführt werden. Das gilt besonders für das Replatforming komplexer E-Commerce-Systeme, die ggf. über Jahre organisch gewachsen sind.

2.1 Transform Eliminate Pattern*

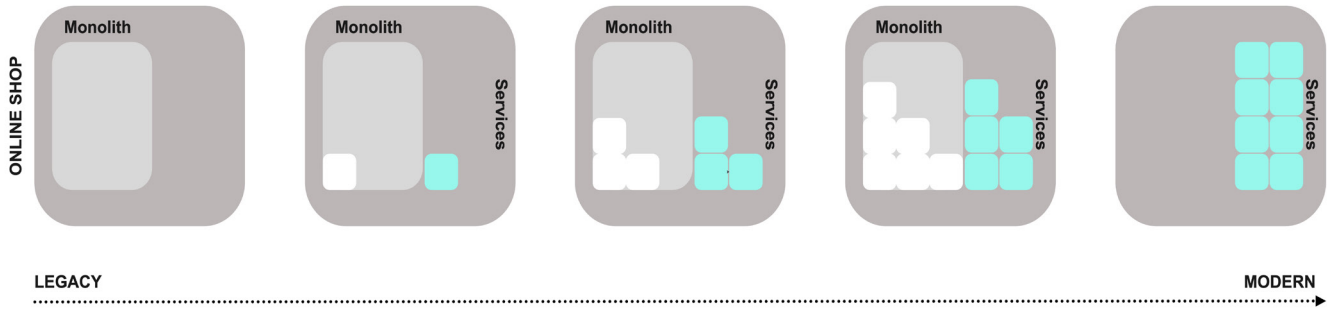
1. Transform: Einzelne Funktion des Monolithen extrahieren, in der neuen Architektur abbilden und direkt aktivieren.

2. Co-exist: Alte Funktion im Monolithen aktiv halten und im Wechsel mit der neuen Architektur parallel verwenden.

3. Eliminate: Alte Funktion aus dem Monolithen entfernen und die Last komplett auf die neue Architektur umschalten.

2.2 Strangler Fig Pattern*

Strangler Fig Pattern beschreibt die konkrete, schrittweise Umsetzung des Transform Eliminate Patterns: Die neuen Funktionen des neuen Systems wachsen quasi um den alten Monolithen wie ein Feigenbaum herum und schnüren ihn Stück für Stück ab.



3. Schritt für Schritt zum Composable Commerce

Moderne Anbieter bieten ihre Lösungen für Composable Commerce als Software as a Service (SaaS) an – im Idealfall „ready to use“ in einer Managed Cloud. Mit der Anmeldung im System werden Composable-Commerce-Komponenten automatisch bereitgestellt.

Sollen Komponenten von verschiedenen Anbietern nach dem Best-of-breed-Prinzip individuell zusammengestellt werden, empfiehlt es sich im Einzelfall, einen Accelerator* wie den **Precomposer*** von eCube zu nutzen, der ein fertiges Setup von Services beinhaltet.

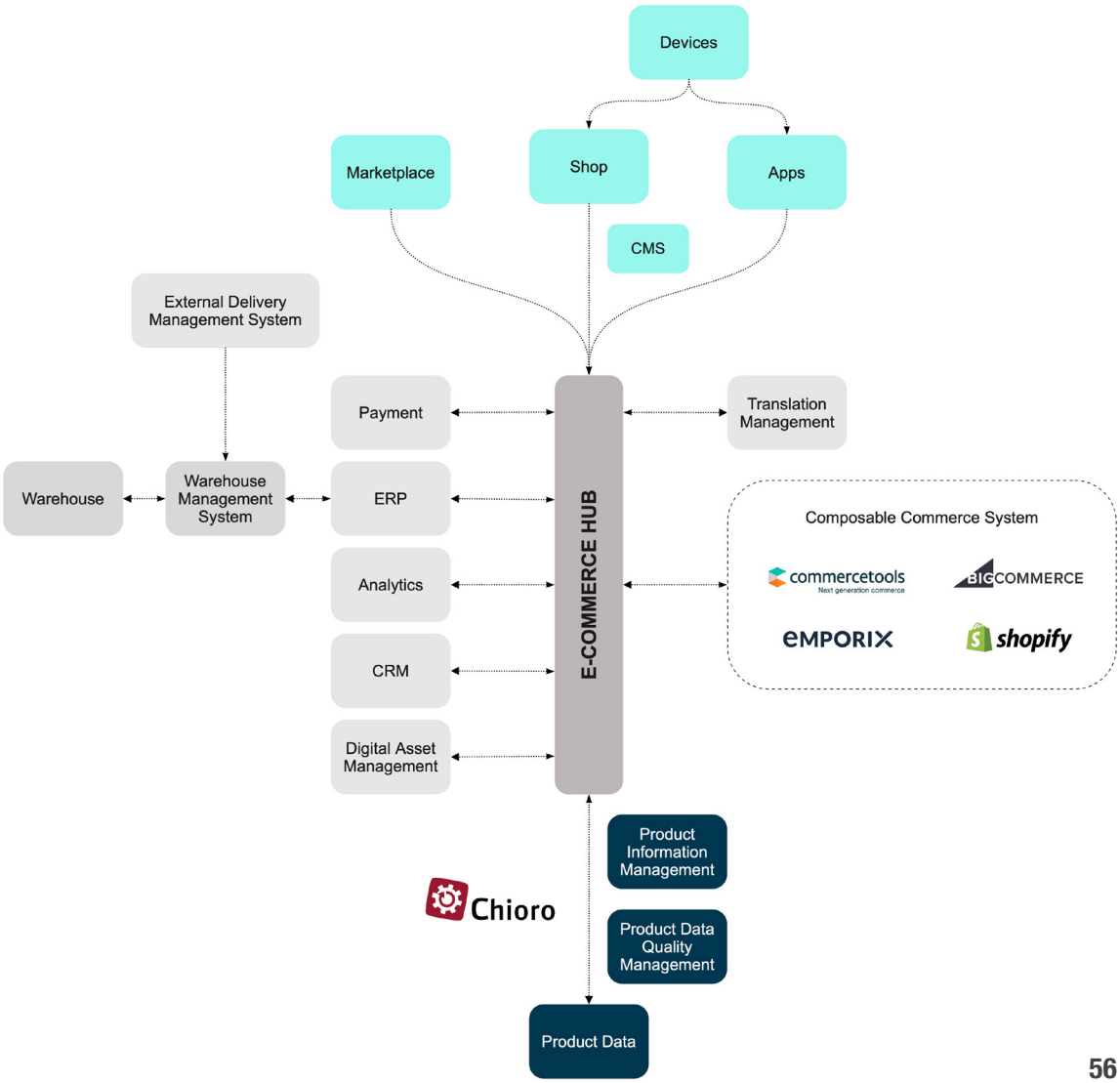
3.1 Composable-Commerce-System einrichten

- **Account anlegen:** Mit wenigen Klicks lässt sich in der Regel zunächst ein kostenloser Trial-Account anlegen, um ohne Risiko zu testen.
- **System konfigurieren:** Im Idealfall über eine grafische Oberfläche zum Konfigurieren der Services; alternativ per API-Call.
- **Typischer Funktionsumfang eines Composable-Commerce-Systems:**
  - Management von Kundendaten
  - Category Management
  - Produktdatenmanagement
  - Produkte empfehlen und bewerten
  - Produktsuche
  - Wunschlisten und Einkaufswagen
  - Ordermanagement und -historie
  - Management von Preisen und Rabatten sowie Steuern
  - Shipping
- **Services verwenden (GET/POST):** API-Standard-Anfragen an Services zum Abfragen oder Senden von Informationen erstellen.

3.2 Aggregator-Service* und Drittsysteme einrichten

- **Zentrale Kommunikation herstellen:** Ein selbst entwickelter Aggregator-Service zentralisiert Kommunikation zwischen Services und reduziert Komplexität.
- **ERP, PIM und CRM anbinden:** per Aggregator-Service oder API; bei älteren Systemen per SOAP- oder FTP-Schnittstelle.

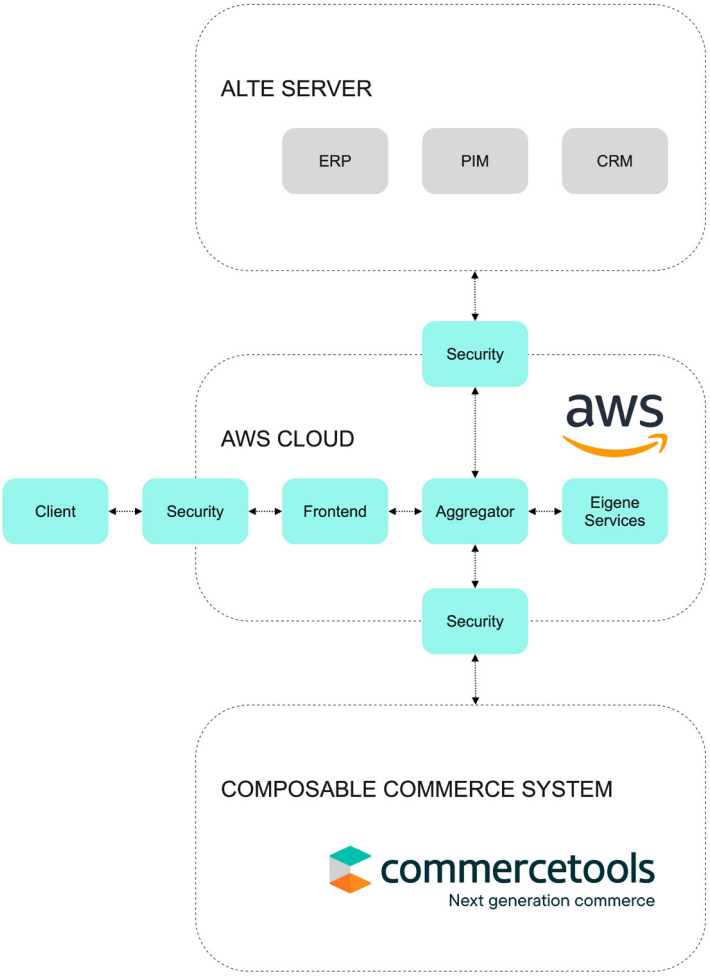
Beispiel für die Nutzung eines schlanken Aggregator-Services (E-Commerce Hub)



3.3 Eigene Services implementieren

- **Eigene Services entwickeln:** Funktionsumfang des Systems erweitern oder Alternativen für Standardkomponenten z. B. per SDK entwickeln.
- **API-Kommunikation herstellen:** gemäß Anforderungen des Composable-Commerce-Systems per HTTP-API, JSON, IDs etc.
- **Hosting:** neues System in der Managed Cloud, z. B. Amazon Web Services; Drittsysteme ggf. in bestehender Cloud-Umgebung belassen.

Beispiel für (vorläufiges) Hostingkonzept



3.4 Frontend einrichten

- **Frontend-Accelerator* aufsetzen:** gemäß Dokumentation des jeweiligen Anbieters, z. B. commercetools-Frontend / alternativ: Setup der Frontend-Basis mit quasi Webstandard-Technologien, wie z. B. Next.js
- **Templates anpassen:** UI-Komponenten aus dem System oder von Drittanbietern in die Frontend-HTML-Templates integrieren.
- **UI-Komponenten verwenden:** Vorgefertigte Komponenten als HTML-Knoten in den Templates nutzen.
- **(Headless) CMS integrieren:** Komponenten und deren Inhalt editierbar machen z. B. mit Storyblok
- **Composable-Commerce-System anbinden:** Frontend und Backend per API Clients verknüpfen und konfigurieren.
- **Frontend technisch optimieren:** insbesondere SEO-Optimierung nach dem Perfect-Storefront-Prinzip*.

Weitere Informationen im eCube-Magazin:
„Technische Frontendoptimierung ‚Perfect Storefront' (Teil 1)"
„Technische Frontendoptimierung ‚Perfect Storefront' (Teil 2)"

3.5 Migration

- **Prozesse entwerfen:** Bewährte Prozesse nachbilden und ggf. um neue Prozesse z. B. für Spezialfälle im Kaufprozess ergänzen.
- **Minimum Viable Product entwickeln:** Architektur und Prozesslandschaft per BPMN (Business Process Model and Notation) entwerfen.
- **Shopseiten bauen:** im Frontend des Composable-Commerce-Systems oder Storefront-Lösung eines Drittanbieters.
- **Kaufprozesse umsetzen:** Start mit Shop-Funktionen, die das geringste Risiko und die geringste Komplexität für die Entwicklung bedeuten.
- **Migration per „Strangler Fig Pattern“:** schrittweise Ablösung, beginnend im Frontend oder im Backend des vorhandenen Systems. Betrifft sowohl Daten (z. B. Kundendaten oder Alt-Bestellungen) als auch redaktionellen Content (Landingpages, Marketing-Texte, Digitale Assets usw.).

4. Tools für die Umsetzung

Dokumentation	Microsoft Office365 Sharepoint Atlassian Confluence Google Docs
Planung	Microsoft Office365 Miro Atlassian Jira Google Docs Trello
Kommunikation	Microsoft Teams Zoom
Tasks	Atlassian Jira Trello
Versionsverwaltung	github gitlab BitBucket
Design	Figma Storybook Adobe Photoshop

06 Onboarding

„A fool with a tool is still a fool“ – neue Software-Systeme und Tools alleine schaffen keinen Geschäftserfolg. Anwender müssen befähigt werden, diese zu verstehen und optimal zu nutzen. Deshalb ist ein umfassendes Onboarding entscheidend für den Erfolg von Software-Projekten.

Wichtig ist dabei, dass das Anwender-Onboarding nicht erst am Ende des Projekts startet, sondern bereits zu Beginn, u. a. durch die Einbindung der Stakeholder in die Planung und Umsetzung. Das stärkt die Akzeptanz der späteren Anwender und ermöglicht eine frühzeitige Gewöhnung an neue Tools.

Für ein wirksames Onboarding zum Ende des Projekts werden die Anwender und andere Stakeholder typischerweise per Schulung, Workshop, Benutzerdokumentationen und Testphasen mit den neuen Technologien vertraut gemacht.

Schulungen vermitteln nicht nur die Bedienung, sondern auch den Zweck und die Vorteile der Software. Workshops bieten darüber hinaus die Möglichkeit, Feedback zu sammeln und Lösungen zu erarbeiten. Benutzerdokumentationen sollten klar und auch für Nicht-Entwickler verständlich sein.

06

eCube als Migrations-partner

eCube unterstützt Unternehmen auf strategischer, fachlicher und technischer Ebene dabei, die richtigen Maßnahmen und Technologien für ein schnelles und nachhaltiges Unternehmenswachstum zu identifizieren, auszuwählen und umzusetzen.

Wir stellen die Erfahrung aus einer Vielzahl erfolgreicher Projekte, das Fachwissen und die Werkzeuge zur Verfügung, die notwendig sind, um digitale Commerce-Ökosysteme nach individuellen Anforderungen entwerfen und nachhaltig zu implementieren.

Das „Team Better Commerce“ von eCube implementiert nicht nur, sondern versetzt seine Kunden auch strategisch in die Lage zu versetzen, die volle Kontrolle über ihr digitales Geschäft zu übernehmen und es erfolgreich auf die nächste Stufe zu heben.

„Unser Ansatz bei der Beratung, Implementierung und dem Know-how-Transfer ist pragmatisch. Wir wollen unsere Kunden ins Handeln bringen. Wissen gehört geteilt und in die Praxis gebracht.“ Günter Heiß // Gründer von eCube

Beratung

Unser Ziel ist es, Sie für Ihren Markt mit den besten verfügbaren Lösungen, Vorgehensweisen und Know-how auszustatten. Als Partner bieten wir Ihnen dafür unser Expertenwissen aus unseren Projekten, den realistischen externen Blick, Insights in den Wettbewerb und ein Team, mit dem Sie zielorientiert in die Umsetzung kommen.

Software-Lösungen

Ob Front- oder Backend für Ihr Commerce-System – wir stehen Ihnen für die technische Realisierung zur Verfügung. Unsere Profis aus der Entwicklung sind versiert und können Sie auch bei komplexen Aufgaben jederzeit unterstützen. Als Realisierungspartner mit über 20 Jahren Erfahrung beherrschen wir die gesamte Bandbreite moderner Technologien und Tools.

Professional Services

Passend zu Ihren Anforderungen stellen wir für Sie dedizierte Expertenteams bereit. Sei es für Bereiche, die erst sukzessive aufgebaut werden, wenn ein schnelles Einsatzteam gebraucht wird, wenn es zu Anforderungsspitzen kommt oder wenn es kurz- bis mittelfristig Bedarf für ein ausgelagertes Team für Spezialaufgaben gibt.

Glossar

Accelerator: Funktionsfertiges Set-up-Paket für Software-Architekturen, das sich schnell installieren lässt und die wichtigsten Funktionen z. B. für den Einstieg in den Composable Commerce enthält.

Aggregator-Service: Ein Dienst, der Daten oder Services aus verschiedenen Quellen zusammenführt und eine einheitliche Schnittstelle für den Zugriff bereitstellt.

API-Plattform: Eine Plattform, die Werkzeuge und Services bereitstellt, um APIs (Application Programming Interfaces) zu erstellen, zu verwalten und zu überwachen.

Backlog: Eine Liste von Aufgaben, Anforderungen oder Fehlern, die in einem Projekt noch bearbeitet werden müssen, oft priorisiert und in Scrum- oder Agile-Methoden verwendet.

Best-fit: Eine Lösung oder Methode, die im individuellen Fall am besten zu den spezifischen Anforderungen und Bedingungen eines Projekts oder einer Aufgabe passt.

Best-of-breed: Die besten am Markt verfügbaren Produkte oder Technologien aus verschiedenen Bereichen, die zusammen eine optimale Lösung bieten.

Business Process Model and Notation (BPMN): Ein grafischer Standard zur Modellierung und Darstellung von Geschäftsprozessen.

Business Owner: Eine Person oder ein Team, das für den Geschäftserfolg eines Projekts oder Produkts verantwortlich ist und die Anforderungen aus geschäftlicher Sicht definiert.

Composable Commerce: Eine Technologie-Strategie im E-Commerce, die modulare Komponenten verwendet, um flexible und anpassbare Handelslösungen zu erstellen.

Cloud-native SaaS: Software-as-a-Service (SaaS), die speziell für den Einsatz in Cloud-Umgebungen entwickelt wurde und deren Vorteile wie Skalierbarkeit und Flexibilität nutzt.

Epic: Eine umfangreiche Benutzergeschichte (User Story) oder ein umfassendes Ziel in der Agile-Methodik, das in kleinere Aufgaben oder User Stories aufgeteilt wird.

Frontend-Accelerator: Werkzeuge und Frameworks, die die Entwicklung von Frontend-Anwendungen beschleunigen, indem sie passende Komponenten und optimierte Workflows bieten.

GANTT-Diagramm: Ein Balkendiagramm, das den zeitlichen Ablauf von Projektaktivitäten darstellt und zur Projektplanung und -verfolgung verwendet wird.

Headless Architektur: Ein Architekturansatz, bei dem das Frontend (die Benutzerschnittstelle) und das Backend (die Datenverarbeitung) getrennt sind und über APIs kommunizieren.

MACH Architektur: Ein Architekturprinzip, das auf Microservices, API-first, Cloud-native und Headless aufbaut, um flexible und skalierbare Softwarelösungen zu ermöglichen.

Microservices: Eine Architekturmethode, bei der eine Anwendung aus kleinen, unabhängigen Diensten besteht, die jeweils eine spezifische Funktion erfüllen und über APIs kommunizieren.

Migration: Der Prozess der Übertragung einer Softwareanwendung und/oder von Daten von einer Umgebung oder Plattform zu einer anderen.

Minimum Viable Product (MVP): Eine frühe Version eines Produkts mit den minimal notwendigen Funktionen, um Feedback von echten Nutzern zu erhalten und die weitere Entwicklungen zu steuern.

Monolithische Architektur: Eine Softwarearchitektur, bei der alle Komponenten einer Anwendung in einer einzigen Codebase oder einer einzigen Anwendung zusammengefasst sind.

Netzplan: Ein Diagramm, das die Abhängigkeiten und zeitlichen Abläufe von Projektaktivitäten darstellt und zur Projektsteuerung verwendet wird.

Perfect Storefront: Das Frontend eines Online-Shops oder einer E-Commerce-Anwendung, das bestmöglich auf die Anforderungen von Käufern und Suchmaschinen abgestimmt ist.

Precomposer: Accelerator für MACH-Architekturen auf Basis von commercetools und Storyblok, das sich schnell installieren lässt und die wichtigsten Funktionen für den Composable Commerce enthält.

Product Owner: Eine Rolle im Scrum-Team, die für die Maximierung des Produktwerts und das Management des Product Backlogs verantwortlich ist.

Qualitätsszenario: Beschreibt spezifische Bedingungen und Kriterien, unter denen die Qualität einer Softwarekomponente oder -lösung bewertet wird.

Replatforming: Der Prozess der Migration einer Anwendung oder Infrastruktur auf eine neue Plattform, oft zur Nutzung moderner Technologien und Architekturen.

Scope: Der gesamte Umfang und die Grenzen eines Projekts oder Produkts, einschließlich aller Funktionen und Anforderungen, die erfüllt werden müssen.

Scrum Master: Eine Rolle im Scrum-Team, die für die Förderung und Unterstützung der Scrum-Methodik sowie die Beseitigung von Hindernissen verantwortlich ist.

Software-Retrofit: Die Modernisierung einer Softwareanwendung durch Hinzufügen neuer Funktionen oder Verbesserung der bestehenden Funktionen, ohne das gesamte System neu zu erstellen.

System-Kontext-Diagramm: Ein Diagramm, das die Systemgrenzen und die Interaktionen des Systems mit externen Entitäten und Benutzern darstellt.

Strangler Fig Pattern: Software-Modernisierungsmuster für die inkrementelle Ablösung von Software-Monolithen nach dem Transform Eliminate Pattern.

Transform Eliminate Pattern: Ein Migrationsansatz, bei dem ein neues System schrittweise das alte System ersetzt, indem neue Funktionen hinzugefügt und alte nach und nach entfernt werden.

User Story: Eine kurze, einfache Beschreibung einer Anforderung aus der Perspektive eines Endnutzers, die oft im Rahmen agiler Entwicklungsprozesse verwendet wird.

Use-Case-Diagramm: Ein Diagramm, das die Interaktionen zwischen Benutzern (Akteuren) und einem System beschreibt, um die funktionalen Anforderungen zu erfassen.

Vendor Lock-in: Eine Situation, in der ein Unternehmen so stark von den Produkten eines Anbieters abhängig ist, dass ein Wechsel zu einem anderen Anbieter schwierig und kostspielig ist.

Waterscrum: Eine hybride Projektmanagementmethode, die Elemente aus dem traditionellen Wasserfallmodell und der agilen Scrum-Methodik kombiniert.

Work Breakdown Structure (WBS): Eine hierarchische Gliederung der Aufgaben eines Projekts, die zur Organisation und Verwaltung des Projektumfangs dient.

eCube GmbH
Agnesstr. 61 | 80797 München
E-Mail: kontakt@ecube.de
Telefon: +49 89 550 556-0
www.ecube.de