



Replatforming im Einzelhandel_

Mit modularen Architekturen zur
zukunftsfähigen Unternehmenssoftware

www.hyand.com

Das Wichtigste in Kürze_

- Gerade im Einzelhandel sind wichtige Systeme wie die Warenwirtschaft oft veraltet und bergen somit unternehmerische Risiken. In den letzten Jahren haben sich Technologien und Methoden etabliert, die neue Möglichkeiten für eine nachhaltige Modernisierung von Legacy-Systemen eröffnen.
- Die individuellen Anforderungen an die Unternehmens-IT geben vor, welcher Ansatz einer Modernisierung das beste Ergebnis liefert.
- Eine Softwarearchitektur mit Modulithen ist für Einzelhandelsunternehmen eine sinnvolle Alternative zu Microservices, um die Komplexität bei der IT-Modernisierung in den Griff zu bekommen.

IT-Systeme, insbesondere wenn es sich um Individualsoftware handelt, veralten mit der Zeit. Aufgrund sehr spezialisierter Prozesse, die nur unzureichend über Standards abgebildet werden können, ist gerade der Einzelhandel geprägt durch einen hohen Anteil an Individualentwicklung. Wichtige Systeme wie die Warenwirtschaft sind von dringendem Modernisierungsbedarf betroffen.

Neue Anforderungen, wie sie beispielsweise durch den Wettbewerb mit Online-Lieferdiensten entstehen, können die Systeme nicht oder nur mit enormem Aufwand stemmen. Gleichzeitig sind die betroffenen Systeme, z.B. eine Warenwirtschaft, unternehmenskritisch. Sollten sie ausfallen, steht für die Unternehmen viel auf dem Spiel. Änderungen am System sind demnach mit einem hohen Risiko verbunden.

Mit der passenden Herangehensweise können Unternehmen diese Herausforderung meistern. Denn in den vergangenen Jahren haben sich Technologien und Methoden etabliert, die neue Möglichkeiten für eine Modernisierung von Legacy-Systemen eröffnen. Mit der folgenden Strategie können Unternehmen ihre IT zukunftsfähig aufstellen.

Strategie für eine zukunftsfähige Unternehmenssoftware_

Unter dem Stichwort Replatforming beschreibt das MIT Center of Information Systems Research verschiedene Ansätze, Unternehmenssoftware zukunftsfähig zu machen. Je nach Situation werden unterschiedliche Vorgehensweisen bei der IT-Modernisierung empfohlen.

Altsysteme direkt und vollständig durch moderne Lösungen zu ersetzen, verspricht vielfach den größten Nutzen, ist aber meist mit hohen Risiken verbunden. Legacy-Systeme wurden oft über Jahrzehnte entwickelt und enthalten in großen Mengen wertvolle Daten, die es zu erhalten gilt. Die Entwicklung eines Neusystems ist zeitintensiv. Währenddessen muss das Unternehmen das bestehende System mit all seinen Problemen weiternutzen.



Unsere Erfahrung zeigt, dass ein kontrolliertes, schrittweises Vorgehen in Iterationen für Einzelhandelsunternehmen die bessere Strategie ist.



#1 Rediscover – Systeme verstehen, Schätze heben und Ballast abwerfen_

Eine der wichtigsten Voraussetzungen für die Modernisierung von Legacy-Systemen ist, sie zu verstehen. Aufgrund ihrer Komplexität gelingt dies nur in einem iterativ fortlaufenden Prozess. Dafür werden ein agiles Mindset und Erfahrung mit agilen Arbeitsweisen benötigt. Das Domain-Driven Design nutzt kollaborative Methoden, um komplexe Domänen systematisch zu erforschen. IT-Dienstleister erarbeiten gemeinsam mit den Fachexpertinnen und Experten des jeweiligen Kunden verständliche, oft visuelle Modelle für die Fachdomäne. Diese Modelle bilden eine für alle Beteiligten verständliche Basis für die Implementierung des neuen Systems. Ziel bei der Modellierung ist, das im Altsystem implementierte Fachwissen und sämtliche Unternehmensdaten zu erhalten und für die Zukunft nutzbar zu machen.

Weiterer Pluspunkt der Analyse: Unnötiger Ballast, wie lange nicht genutzte Services, wird identifiziert und entfernt, sowie Potential für Verbesserungen an bestehenden fachlichen Prozessen erkannt und umgesetzt.

#2 Restructure – Legacy-Systeme stabilisieren_

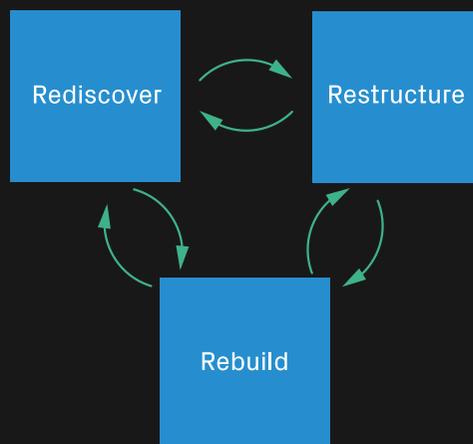
Bestandssysteme müssen nicht immer ersetzt werden. Oft reicht eine Stabilisierung aus, um das System wieder warten und erweitern zu können. Eine Restrukturierung oder ein Austausch eines veralteten Frameworks kann bestehende Systeme so weit verbessern, dass sie neuen Anforderungen wieder gewachsen sind.

Zum Beispiel haben unsere Expertinnen und Experten für einen Kunden die Benutzeroberfläche eines Legacy-Systems modernisiert, was den Anwenderinnen und Anwendern eine vollkommen neue, zeitgemäße User Experience ermöglichte. Oft ist dies der erste Schritt, die komplexe, monolithische Struktur aufzubrechen und zu modularisieren.

#3 Rebuild – Es müssen nicht immer Microservices sein_

Modularisierung spielt bei der Softwareentwicklung eine wichtige Rolle, um ungewollte technische Komplexität zu vermeiden und fachliche Komplexität handhabbar zu machen. Technische Komplexität entsteht durch Kopplung – ein System funktioniert nur aufgrund der Zusammenarbeit seiner Teile. Um unnötige Komplexität zu vermeiden, muss die Kopplung kontrolliert erfolgen. Dazu wird das System in Module strukturiert, die über Schnittstellen lediglich die Funktionalität nutzbar machen, die dafür explizit vorgesehen ist. Eine zufällige Verknüpfung, wie sie bei Systemen ohne harte Modulgrenzen immer wieder passiert, wird technisch unterbunden.

Eine besonders strikte – und bei Softwarearchitekten und -entwicklern beliebte – Form der Modularisierung sind Microservicearchitekturen. Module werden in Containern ausgeliefert, die als eigenständige Einheiten betrieben werden. Für die Nutzung ihrer Services bieten sie Schnittstellen an und kommunizieren über Nachrichten miteinander. Sender und Empfänger müssen sich dabei „nicht kennen“, Nachrichtempänger müssen zum Sendezeitpunkt nicht einmal verfügbar sein. Das erlaubt den Aufbau von hochelastischen und gut erweiterbaren Systemen. Solange ihre Schnittstellen stabil bleiben, können Microservices unabhängig voneinander gewartet und weiterentwickelt werden. So entstehen schnell große Systeme mit einer Vielzahl an Microservices, die als unabhängige Einheiten ausgeliefert und in Betrieb genommen



werden. Allerdings sind microservicebasierte Systeme verteilte Systeme, die über Netzwerke miteinander kommunizieren. Das verlagert im Vergleich zum klassischen Monolithen die Komplexität auf andere Ebenen. Microservices müssen zum Beispiel als Netzwerkknoten verwaltet werden. Der hohe Grad an Verteilung bringt Probleme mit sich, die bei einer einzigen Deployment-Einheit nicht in Erscheinung treten: Fehleranalyse und -behebung in verteilten Systemen, höhere Latenzzeiten durch Netzwerkkommunikation anstelle von prozesslokalen Methodenaufrufen, temporäre Netzwerkausfälle, Log-Aggregation, verteilte Transaktionen, Vermeidung von Redundanz und Inkonsistenz, Sicherheitsaspekte und vieles mehr.

Die technischen Rahmenbedingungen für Microservice-Architekturen stellen insgesamt oft eine derart hohe Hürde dar, dass dieser Ansatz von vielen internen IT-Abteilungen verworfen wird. Es wird nach technischen Konzepten verlangt, die die Vorteile der Modularisierung nutzen, ohne die Komplexität verteilter Systeme ausufern zu lassen.

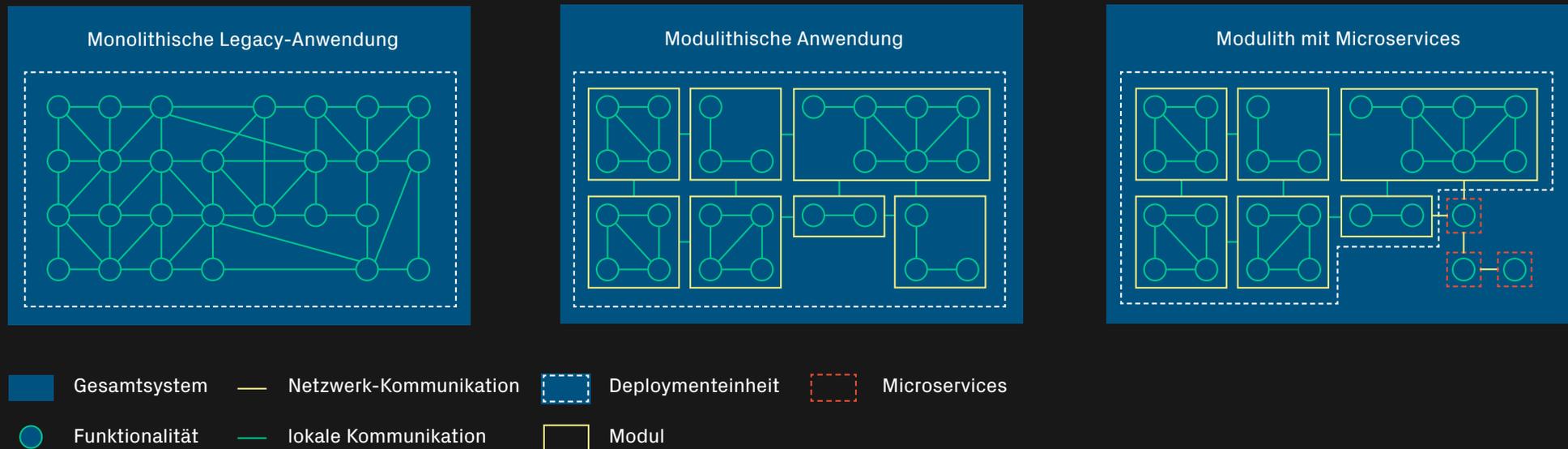


Daher wird heute oft ein Kompromiss aus monolithischen und microservicebasierten Architekturen angestrebt: Modulithen.

Modulare Architektur mit Modulithen und Microservices_

Modulithen setzen auf eine oder wenige Deployment-Einheiten. Viele der oben genannten Probleme werden dadurch vermieden. Ein Modulith ist aber kein fachlicher Monolith, sondern hat eine möglichst strikte, modulare und an fachlichen Kriterien ausgerichtete innere Struktur. Diese sorgt dafür, dass das Gesamtsystem aus kleinen, gut verständlichen, fachli-

chen Funktionalitäten besteht. Die Funktionen werden wie bei den Microservices unabhängig voneinander entwickelt und mit Nachbarsystemen über stabile Schnittstellen oder Nachrichten lose gekoppelt.



Wie entsteht eine modulare Softwarearchitektur?

Um eine modulare Softwarearchitektur aufzubauen, wird im einfachsten Fall durch statische Codeanalyse überprüft, ob Architekturstandards eingehalten werden. Arch-Unit ist ein zuletzt viel beachtetes Werkzeug, mit dem sich Regeln für die Systemarchitektur sehr flexibel formulieren lassen. Diese Regeln werden von Arch-Unit automatisiert geprüft und etwaige Verstöße erkannt. Die hohe Flexibilität des Frameworks hat allerdings einen Nachteil: Die Regeln müssen, wie das Softwaresystem selbst, korrekt implementiert und getestet werden.

Bei „echten“ Modulithen wird die fachliche Modulstruktur nicht nur durch Konventionen, sondern auch technisch wasserdicht abgesichert. Moderne Programmiersprachen bringen mittlerweile neue technische Features zur Sicherstellung der Modularität einer Codebasis mit. Im Fall von Java und dem JPMS kann bereits zur Compile-Zeit sichergestellt werden, dass sich keine unerwünschten Kopplungen und damit unnötige Komplexität in die Codebasis einschleichen.



Legacy-Software im Einzelhandel – Was ist das und muss das weg?

Legacy-Anwendungen sind in Unternehmen etablierte Softwaresysteme, die über viele Jahre kontinuierlich weiterentwickelt wurden. Ohne sie sind viele unternehmenskritische Geschäftsprozesse nicht denkbar.

Legacy-Systeme können sowohl Lizenzprodukte als auch Individualentwicklungen sein. Im Einzelhandel sind mit die größten Datenbanksysteme im Einsatz. Die Lizenzen verursachen jedes Jahr hohe Kosten. Daher verwundert es nicht, dass für die Branche wichtige Unternehmenssoftware datenbanknah, um nicht zu sagen „in der Datenbank“, entwickelt wurden. Die enge Verflechtung von Datenbank und Unternehmenssoftware stellt eine besonders große Herausforderung für die Modernisierung dar. Viele Probleme von Legacy-Systemen resultieren aus ihrer Architektur. Aufgrund fehlender technischer Möglichkeiten wurden sie in der Vergangenheit oft als monolithische Systeme entwickelt. Durch kontinuierliche Erweiterung ihrer Funktionalität entstand mit der Zeit eine Komplexität, die die Systeme an die Grenzen ihrer Wart- und

Erweiterbarkeit führten. Die daraus erwachsenden Herausforderungen sind offensichtlich und können unternehmenskritisch werden. Ein Beispiel sind die disruptiven Änderungen für den stationären Handel durch das Aufkommen von marktdominierenden Online-Händlern. Nicht selten waren ausbleibende oder zu späte Anpassungen der IT auch ein Grund für existenzbedrohende Krisen ehemals bedeutender Händler. Daher werden Alternativen zur monolithischen Softwarearchitektur immer wichtiger!

Weitere Gründe für eine IT-Modernisierung sind die genutzten Technologien und der bei veralteten Techniken besonders dramatische Fachkräftemangel: Für den Betrieb notwendige Hard- und Systemsoftware verschwinden vom Markt oder sind mittlerweile so alt, dass es für Unternehmen zunehmend schwerer wird, qualifiziertes Personal zu finden, das den Umgang mit den Technologien beherrscht. Herausforderungen, deren Lösung über die Zukunftsfähigkeit von Unternehmen entscheiden kann.

Fazit_

Je nach Situation gibt es verschiedene Ansätze, Legacy-Systeme zu modernisieren und zukunftsfähig zu machen. Unternehmen, die eine komplexe Infrastruktur durch Microservices vermeiden wollen, setzen auf Modulithen. Sie bringen Struktur in große Softwaresysteme und reduzieren deren Komplexität. Gerade im Einzelhandel mit seinen stark individualisierten Softwaresystemen lohnt sich der Einsatz von Modulithen, um die IT zukunftsfähig aufzustellen und unternehmerische Risiken zu reduzieren. Eine kontrollierte, schrittweise Ersetzung der Monolithen ermöglicht den sicheren Übergang zu einer modernen Softwarearchitektur im laufenden Betrieb.



Volker Koster
CTO

Telefon: +49 2102 30 961-125
Mobil: +49 173 5420310
Mail: volker.koster@hyand.com



Hyand Solutions GmbH
Balcke-Dürr-Allee 9
40882 Ratingen

Geschäftsführung
Siegfried Lassak
Jürgen Allmich

Amtsgericht Düsseldorf
HRB 99238
USt-IdNr.: DE169583853

Bildnachweise:
Hyand Solutions GmbH

Außer:
Titelbild: ©diloka107/freepik
Seite 3: ©DKosig/iStock
Seite 4: ©monsitj/iStock
Seite 8: ©metamorworks/iStock

