FINANZIERUNG
LEASING
FACTORING
FLF

4

JULI 2023 · 70. JAHRGANG

Foto: Adobe Stock / wladimir1804

DIGITALER SONDERDRUCK

LEASING

Neue Wege der System-Architektur

Monolithischer versus modularer Aufbau

Martin Feith, Mitglied des Executive Board FSI, und
Martin Fröstl, Technical Head of Financial Services,
NAVAX Unternehmensgruppe

Neue Wege der System-Architektur

Monolithischer versus modularer Aufbau

Seit Jahrzehnten gibt es in der Softwarearchitektur von Leasing- und Factoring-Programmen die Diskussion, ob ein monolithischer oder ein modularer Aufbau die ideale Lösung für die Finanzdienstleistungsbranche darstellen. Neue technologische Entwicklungen bringen andere Aspekte in einen spannenden technologischen Diskurs. Die Autoren betrachten den Stand der Dinge und erläutern die Vor- und Nachteile der jeweiligen Softwarearchitektur. Außerdem machen sie darauf aufmerksam, dass diese vor dem Einsatz neuer Finanzanwendungen genau betrachtet werden müssen. (Red.)

Die erste Annäherung an das Thema rückt zunächst das Umfeld in den Vordergrund. Handelt es sich um ein Konzernunternehmen, das bereits relativ viele Softwaremodule aus dem Gruppenumfeld vorgegeben hat? Handelt es sich um eine eigentümergeführte Gesellschaft, die eine smarte All-in-one-Lösung mit modernen Anbindungsmöglichkeiten an Drittanbieter benötigt oder hat das Unternehmen einen stark intern getriebenen Prozess mit den Kunden? Die Antworten auf diese Fragen sind essenziell, um die für die jeweiligen Gegebenheiten optimale Lösung zu finden.

Die letzteren zwei Varianten sprechen prima vista eher für eine monolithische Gesamtlösung. Die erste Variante des großen Konzernunternehmens mit vielen vorgegebenen Systemen (inklusive Verwaltung und Finanzbuchhaltung) eher für eine modular aufgebaute Lösung als Ergänzung (zum Beispiel in diesem Fall mit Front und Middle Office).

Monolith einst und heute

Die monolithischen Systeme der 1990er und 2000er Jahre waren sehr starr im Aufbau, mit wenig Veränderungsmöglichkeiten, außer mittels aufwendiger Programmierungen. Die Anbindung an externe Schnittstellen und Portale war komplex und teuer. Ein moderner Monolith ist heute eine Standardsoftware mit umfangreichen Konfigurationsmöglichkeiten und einfacher Anbindung über Programmschnittstellen (Application Programming Interfaces, APIs) und sonstiger moderner Schnittstellentechnologien an diverse Spezialisten-Systeme von Bonitätsanbietern, Wertverlaufskurvenprovidern et cetera, auch für Themen wie Geldwäsche, PEPs, unterschiedlichste individuelle Portale und ähnliches.

Eine monolithische Softwarearchitektur bringt einerseits Vorteile, weil sämtliche Informationen quasi ohne Schnittstellen in einem einzigen aber sehr oft durchaus komplexen System zugänglich sind. Damit verbundene Nachteile sind gegebenenfalls aufwändige Updates, wenn der gesamte Monolith ausgetauscht werden muss. In einem Monolith sind sämtliche Thematiken installiert, auch wenn diese der Kunde gar nicht benötigt. Es werden Tabellenstrukturen mitgeschleift die gar nicht verwendet werden.

Vorteil sind die kurzen Wege und der meist sehr schnelle Zugriff zu den unterschiedlichen Daten, weil diese oftmals in einer einzigen Datenbank liegen und damit einen gemeinsam integrierten Datenbestand darstellen. Gerade bei komplexen kaufmännischen Buchungsprozessen und komplexen Datenstrukturen ist die volle Zusammengehörigkeit ein weiterer Vorteil.

Modularer Aufbau

Die grundlegende Idee dabei ist es, möglichst gut abgrenzbare Funktionalitäten in einem Modul zusammenzufassen und damit die Entscheidung zu ermöglichen, ob und welche Module zum Einsatz kommen sollen.

Ziel soll es sein, die Gesamtanwendung möglichst auf das Erforderliche reduzieren zu können. Wenn nicht notwendige Module weggelassen werden, vereinfachen diese die Anwendung und die

MARTIN FEITH

ist Mitglied des Executive Board FSI der NAVAX Unternehmensgruppe, Wien.

E-Mail:
m.feith@navax.com

MARTIN FRÖSTL

ist Technical Head of Financial Services der NAVAX Unternehmensgruppe, Wien.

E-Mail:
m.froestl@navax.com

Komplexität. Auch die Performance einer Anwendung kann dadurch in vielen Fällen positiv beeinflusst werden, weil die Lösung einfach schlanker ist.

Module können unabhängig voneinander sein oder aufeinander aufbauen und ein anderes Modul voraussetzen. Dadurch können unterschiedliche Abhängigkeiten zwischen Modulen entstehen, die natürlich zu berücksichtigen sind. In den meisten Fällen verwenden unterschiedliche Module einer Anwendung die gleiche Technologie und die gleiche Datenbank.

Eine extreme Form stellt die Idee der sogenannten Microservices dar. Die Microservice-Architektur ist ein Modell, welches darauf aufbaut, eine Anwendung aus vielen kleinen, lose miteinander gekoppelten Services zusammenzusetzen. Die einzelnen Services sind möglichst klein gehalten und werden unabhängig voneinander bereitgestellt. Jeder Service verwaltet seine eigenen Daten, hat sein eigenes Datenmodell und setzt auf seiner eigenen Technologie auf. Ein Microservice sollte eine möglichst klar strukturierte und technologische „State of the Art"-Kommunikationsschnittstelle zur Verfügung stellen. Dafür ist der jeweilige Hersteller eines Microservices verantwortlich.

Ein Vorteil dieser Architektur ist, dass ein einzelner Service unabhängig erweitert, aktualisiert und gegebenenfalls auch durch ein gleichwertiges anderes ersetzt werden kann. Voraussetzung ist, dass Inhalt und Technik der Kommunikation mit anderen Services kompatibel bleiben.

Wenn aber kaufmännische Funktionalitäten stark interagieren müssen, stößt diese Architektur schnell an ihre Grenzen. Wenn sich der erforderliche Kommunikationsaufwand zwischen den einzelnen Services aufgrund erforderlicher Abhängigkeiten höher und aufwändiger gestaltet, können rasch Performance-Engpässe und übermäßige Komplexitäten im Austausch zwischen den Microservices entstehen, was die Idee dieser Architektur schnell ad absurdum führen kann.

Sehr ähnliche Ziele verfolgt die serviceorientierte Architektur, auch SOA genannt, die bereits in den späten 1990er Jahren entstand. SOA ist etwas weiter gefasst und verfolgt ein unternehmensweites Konzept und eine Integrationsarchitektur, in der Anwendungen mit spezifischen Geschäftsfunktionen über lose gekoppelte offengelegte Schnittstellen interagieren und wiederverwendet werden können. Bei Microservices liegt der Schwerpunkt eher darauf, eine bestimme Anwendung in möglichst kleine unabhängige Teile zu zerlegen.

Einen etwas anderen Ansatz verfolgt Microsoft in Dynamics 365 Business Central mit der Extensions Technologie basierend auf der Entwicklungssprache AL.

Das grundlegende Konzept der Extensions basiert unter anderem auf folgenden Elementen:

- Zusammengehörige Funktionalitäten können in einer Extension gekapselt und damit von anderen Extensions getrennt werden.
- Eine Extension baut auf einer vorhandenen Basisfunktionalität auf und erweitert diese.
- Eine Extension verändert die darunterliegende Basisfunktionalität optimalerweise nicht, damit die Releasefähigkeit gewährleistet bleibt.
- Für technisch Versierte: Die Erweiterungen erfolgen auf Basis von Events.
- Eine Extension wird als eigene App ausgeliefert, um Installation und Deinstallation möglichst unabhängig vom restlichen System vornehmen zu können.
- Es steht damit zur Auswahl, ob eine Extension installiert wird oder nicht. Somit wird die zum Einsatz kommende Funktionalität so schlank wie möglich gehalten.

In einer Extension stehen sämtliche Möglichkeiten für Erweiterungen offen, welche die Basistechnologie von Dynamics 365 Business Central anbietet. Damit sind sehr moderne und umfangreiche Funktionalitäten genauso umsetzbar wie auch sehr einfache und klein gehaltene Erweiterungen.

Vergleichende Betrachtung

Rein monolithische Systeme können den Anforderungen von modernen IT-Systemen und IT-Anwendungsanforderungen nur mehr bedingt gerecht werden. Zu viele Integrationsanforderungen, Schnittstellenanforderungen zu anderen Systemen, Individualisierungsbedarf und die Releasefähigkeit in einer sich schnell drehenden Technologielandschaft prallen hier aufeinander.

Eine rein auf Microservices abzielende Anwendungsarchitektur kann das Zusammenspiel und den dafür erforderlichen Datenaustausch sehr umfangreich und komplex gestalten, womit auch die Performance-Aspekte und die damit verbundene Komplexität dieses Architekturkonzeptes für kaufmännisch integrierte Finanzanwendungen schnell vor schwer überwindbaren Hürden stehen können.

Aktuelle Systemarchitekturen bringen die Vorteile aus mehreren Architekturansätzen zusammen und ermöglichen das Zusammenfließen aus monolithischem und modularem Ansatz. Dabei wird eine zentrale Kernfunktionalität auf einer gemeinsamen Datenbasis mit der Flexibilität und Austauschbarkeit von Apps kombiniert. Zum Beispiel liefert hier Microsoft mit der Extensions-Technologie in Dynamics 365 Business Central einen solchen Kombinationsansatz, der speziell auf die Umsetzung kaufmännisch integrierter ERP-Anwendungen ausgerichtet ist und eine gute Basis für Finanzanwendungen ermöglichen kann.

Schlussendlich ist es wichtig, dass die oben angeführten Aspekte vor dem Einsatz neuer Finanzanwendungen genau beleuchtet werden, die Vorteile und Nachteile der jeweiligen Architektur für die geplante IT-Strategie im Unternehmen genau abgewogen werden und auf Verfügbarkeit und Releasefähigkeit geachtet wird.