Langlebige Software mit hoher Wirkung

Durchdachte IT-Architektur bildet die Grundlage

IT-Architektur ist niemals Selbstzweck, sondern hat zum Ziel, Kundenanforderungen zu erkennen, deren Ursachen zu verstehen und eine Softwarestruktur zu finden, die alle Anforderungen abdeckt, so dass das geplante Budget eingehalten und eine langfristige Erweiter- und Wartbarkeit gewährleistet werden kann. Die Basis dafür bildet ein Zusammenspiel aus Technologie- und Methodenkompetenz, bewährten Entwurfsmustern und Erfahrung. Nur dann können die Anforderungen des Kunden erfolgreich umgesetzt werden.

Die Schnittstelle von Anwender zu IT-Anwendung ist die Benutzeroberfläche bzw. das Graphical User Interface (GUI). Obwohl diese Komponente nur einen Bruchteil der Gesamtanwendung ausmacht, ist sie oft für deren Schicksal entscheidend: Fachliche Anforderungen werden oftmals über das GUI abgestimmt und abgenommen. Datenqualität und korrekte Datenverarbeitung stehen in direktem Zusammenhang mit der Fähigkeit der Oberfläche, Daten zu erfassen, zu validieren und dem Nutzer entsprechendes Feedback zu geben. Und die Akzeptanz der Endnutzer steht und fällt mit der Bedienbarkeit.

Usability und Workflow sind daher wichtig für die Auswahl der Oberflächentechnologie. Ist die Anwendung zentraler Bestandteil der Arbeit, wie im Bestell- und Angebotsmanagement, werden andere Oberflächenmuster relevant als beispielsweise in bestimmten Bereichen der Logistik, bei denen Massendatenerfassung im Vordergrund steht. msg systems hat daher eine Vergleichsmatrix für die gängigen Oberflächentechnologien des .NET-Frameworks entwickelt, um für jede Anforderung das passende Werkzeug zu wählen.

Basis für gute Architektur

Die Auswahl und Strukturierung der Anwendungsdatenbasis ist von großer Bedeutung. Die Prozesse der Logistikbranche stellen spezifische Anforderungen, die die richtigen Werkzeuge benötigen. In diesem Umfeld ist der Microsoft SQL Server als natürliche Basis für eine .NET-Anwendung eine gute Wahl, daher setzen wir ihn in unseren Referenzarchitekturen häufig ein. Die gekoppelte Nutzung des MS SQL Servers in Verbindung mit der MS SQL Server Compact Edition auf mobilen Endgeräten bei Handscannern, Terminals oder PDAs schafft Mehrwert über die gesamte Prozesskette. Interagierende Systeme auf den gleichen DB-Management-Technologien erbringen bei der Anbindung und Nachnutzung bestehender IT-Systeme hohe Synergien bei der Entwicklung, Wartung und beim Betrieb. Von der intensiven Verzahnung der .NET-Technologien mit anderen Microsoft-Produkten, wie dem MS SQL Server, den MS SQL Server Reporting Services und den Internet Information Services, profitiert die Gesamtarchitektur.

Oberflächentechnologien im Vergleich

	Windows Forms	WPF	Silverlight	ASP.NET
Vorteile	Ausgereifte Oberflächentechnologie. Schnelle Entwicklung mit RAD-Ansätzen. Umfassende Komponenten für Datenerfassung, -präsentation und -auswertung. Schnelle / flüssige Nutzerinteraktion.	Moderne Oberflächentechnologie mit besonders reichhaltiger grafischer Darstellung und umfangreichen Datenanbindungsfeatures. Volle Unterstützung der Clienthardware inkl. Grafikkarte. Schnelle / flüssige Nutzerinteraktion.	Moderne Oberflächentechnologie für Browser und Desktop mit besonders reichhaltiger grafischer Darstellung und umfangreichen Datenanbindungsfeatures. Schnelle / flüssige Nutzerinteraktion.	Clientseitig plattformunabhängige Technologie. Integrierbar in andere Webanwendungen und Portale. Umfangreiche Features für häufigste Anforderungen einer webbasierten Lösung.
Nachteile	Aufwändige grafische Darstellung, begrenzte Anpassungsfähigkeit der User-Interface-Komponenten.	RAD-Ansätze noch in der Entwicklung. Komplexität und UI-Konzept führen zu höheren Einstiegshürden.	Einschränkungen aus Security-Gründen. Security ist komplex, erfordert frühzeitige Beachtung.	Inkompatibilitäten in unterschiedlichen Browsern. Geringere Performance. Usability erfordert höheren Aufwand (Einsatz von Ajax- oder Silverlight-Komponenten)
Erweiterbarkeit	Viele Komponenten externer Anbieter für nahezu jeden Anwendungsfall vorhanden. Tendenz stagnierend.	Anzahl der externen Komponenten wachsend. Am Markt bereits Anbieter vorhanden. Windows Forms Komponenten können in WPF gehostet werden.	Anzahl der externen Komponenten langsam wachsend. Kein Hosting von Legacy-Komponenten. Hosting von HMTL- und Reportingsystemen nur zum Teil möglich. Dem kann über JavaScript / ASP.NET-Anbindung entgegengewirkt werden.	Viele Komponenten externer Anbieter, für nahezu jeden Anwendungsfall vorhanden. Ergänzung um Silverlight- und AJAX-basierte Komponenten.
Marktverhalten (im .NET-Bereich)	Entwickler-Know-How auf dem Markt gegeben, langfristig sinkend.	Entwickler-Know-How auf dem Markt wachsend.	Entwickler-Know-How auf dem Markt langsam wachsend.	Entwickler-Know-How auf dem Markt gegeben.
Ausblick	Weiterentwicklung durch Microsoft unklar. Aktuell Teil des .NET-Frameworks, Support durch Microsoft gegeben.	Weiterentwicklung wird durch Microsoft zur Zeit forciert. Fehlender RAD-Ansatz mit jeder neuen Version verkleinert.	Weiterentwicklung wird durch Microsoft zur Zeit forciert. Fehlender RAD-Ansatz mit jeder neuen Version verkleinert. Funktionsumfang gleicht sich immer mehr WPF an. MS-Zieltechnologie für mobile Geräte.	Weiterentwicklung durch Microsoft aktuell gegeben. ASP:NET profitiert z.T. durch Entwicklung von Silverlight, da es die Silverlight-Komponenten hostet.
Architekturauswirkung	2-Tier, 3-Tier, N-Tier Tendenzen zur Vermischung von Präsentation und Logik werden durch Strukturierung der UI-Technologie verstärkt.	2-Tier, 3-Tier, N-Tier Klare Trennung von Präsentation und Logik werden durch UI-Technologie gefördert.	3-Tier, N-Tier Klare Trennung von Präsentation und Logik werden durch UI-Technologie gefördert. Datenzugriff immer über Silverlight enabled Webservices der Windows Communication Foundation. Webserver zwingend erforderlich.	3-Tier, N-Tier Webserver zwingend erforderlich. Benötigt als Webanwendung andere State-Management- und ggf. Persistierungsmechanismen.
Einsatzbereich	Ehemals Line-of-Business-Anwendungen. Aktuell für bestehende Anwendungen, budgetorientiert Datenerfassungs- und Industrieanwendungen auf mobilen Geräten (z.B. Handscanner).	Line-of-Business-/ Geschäftsanwendungen. Rich-Media-Anwendungen, Anwendungen mit hoher, visueller Nutzerinteraktion (z.B. touch-/gestenbasierte Systeme).	Webbasierte Line-of-Business-/ Geschäftsanwendungen und Anwendungen mit hohem Interaktionsgrad. Rich Internet Application / Rich-Media Application.	Plattformunabhängige, webbasierte Line-of-Business-/ Geschäftsanwendungen ohne hohen Interaktionsanteil. Webbasierte Berichtsanwendungen.
Fazit	Neuprojekte sollten nur auf Grund spezifischer Anforderungen Windows Forms einsetzen. Ansonsten sollte als Desktoptechnologie WPF / Silverlight verwendet werden. (Bsp. für Windows Forms: Nutzung von Embedded Geräten, Einbindung von Legacy-Komponenten, kleine, kurzlebige Anwendungen mit RAD-Ansatz)	Neuprojekte im Desktop-Bereich sollten auf Grund des Entwicklungsfokus von MS und der damit gegebenen Zukunftssicherheit, sowie auf Grund des größeren Funktionsumfangs auf WPF setzen. Insbesondere Interaktions- und Visualisierungsanforderungen sind ggf. einfacher umzusetzen.	Gute Alternative zu herkömmlichen Web-Technologien für höhere Usability und neue Funktionen. Vor einem Einsatz sollten zwingend die eigenen IT-Richtlinien geprüft werden. Nutzung sowohl als vollständige Anwendung als auch als Teil einer ASP.NET-Anwendung möglich.	Bewährte Technologie für die Umsetzung von Intranet- und webbasierten Projekten. Durch die Usability-Einschränkungen nicht als Ersatz für eine der anderen UI-Technologien gedacht. Silverlight kann in Form von zusätzlichen Komponenten eine wertvolle Ergänzung sein.

Datenzugriffstechnologien im Vergleich

	ADO.NET	ADO.NET Typisiertes DataSet	ADO.NET Entity Framework
Beschreibung	Grundlegende Datenzugriffsfunktionalität des MS .NET-Frameworks. Zugriff über DataReader / DataAdapter in eigenen Klassen oder allgemeine DataSets. Datenzugriff über SQL, Stored Procedures & Funktionen.	Etablierte Datenzugriffstechnologie, die den ADO.NET-Funktionsumfang um DataSets und Tabellen mit festen Feldern und einem generativen Ansatz erweitert. Datenzugriff über SQL und Stored Procedures.	Neue Datenzugriffstechnologie auf Basis von ADO.NET-Funktionalität. Abbildung als O/R-Mapper. Datenzugriff über automatisch generierte SQL-Statements. Anbindung von Stored Procedures möglich. Partielle Klasse ermöglicht Abbildung von Geschäftslogik auf den Entitäten.
Performance	Beste Performance-Eigenschaften bei vergleichbaren Abfragen. Abfragen können kontextspezifisch implementiert werden und sind sehr schlank, da sie nur die Daten lesen/schreiben, die für die jeweilige Operation nötig sind.	Mittel bis gering. Je höher die Spalten- und Verknüpfungszahl im geladenen DataSet, umso niedriger die Gesamtperformance. Dem kann zu Lasten von Komplexität und Implementierungsaufwand durch kontextspezifische DataSets mit einer Teilmenge an Spalten begegnet werden.	Mittel bis gering. Je größer die geladenen Objekte und deren Verknüpfungen, umso niedriger die Gesamtperformance. Performanceeinbußen besonders bei größeren Zahlen von geladenen Kind-Objekten zu erwarten. Dem kann durch Aufteilung und asynchrone Verarbeitung des Ladevorgangs begegnet werden.
Implementierungsaufwand	Sehr hoch. Ohne Abstraktion (auf Kosten der Performance) muss für jede Operation eine eigene Abfrage implementiert werden.	Mittel. Gute Designeranbindung im Visual Studio unterstützt die Generierung der Datasets und Tabellen. Datenzugriff muss implementiert werden	Gering. Gute Designeranbindung im Visual Studio unterstützt die Generierung der Datenzugriffs- und Datentransferobjekte. Datenzugriffscode eher kurz.
Technische Komplexität	Niedrig. Kein generierter Code. Logik ist direkt ersichtlich. Arbeit mit grundlegenden .NET-Standard-Funktionalitäten. Modelkomplexität ggf. hoch. Je aufwändiger die Entitäten miteinander verknüpft sind, umso komplexer die implementierte Logik.	Mittel bis hoch. Klassen werden zum Teil durch Visual Studio generiert. Je nach Verknüpfung der Entitäten kann die Verwaltung der Entitätszustände (hinzugefügt, geändert, gelöscht, Abgleich auf Parent-/Child-Ebene) in den DataSets komplex werden.	Hoch, durch hohes Abstraktionslevel und Mappingfeatures des Entity Frameworks. Implementierter Code vergleichsweise kurz, da der Datenbankzugriff über zur Laufzeit generierte SQL-Statements verarbeitet wird.
Änderungsresistenz	Gering. Änderungen in der Datenstruktur führen zu Anpassung/ Neuimplementierung der betroffenen Operationen.	Hoch. Viele Änderungen der Datenstruktur lassen sich durch das Neuerstellen der DataSets in Verbindung mit kleineren Anpassungen erledigen.	Sehr hoch. Sehr viele Änderungen der Datenstruktur lassen sich durch Neuerstellung der Entitäten in Verbindung mit kleineren Anpassungen erledigen. Durch die geringe Menge an eigens zu implementierendem Code ist der Anpassungsaufwand auf dieser Ebene auch gering.
Ausblick	Funktionsumfang gleichbleibend und stabil. Da diese Technologie Basis für die anderen Datenzugriffsmechanismen ist, bleibt sie durch diese aktuell.	Funktionsumfang gleichbleibend und stabil. Im Zuge der Weiterentwicklung des MS .NET-Frameworks zum Teil Erweiterungen (z.B. LINQ2DataSet-Anbindung).	Funktionsumfang wachsend. Microsoft betreibt aktiv Weiterentwicklung.
Architekturauswirkung	Keine. Kann in jedem Entwurfsmuster / jeder Architektur genutzt werden. Basisfunktionalitäten von ADO.NET können über bestehende Interfaces gekapselt werden. So ist ein gleichartiger Datenzugriff auf verschiedene DBMS möglich.	Spezifische Strukturierung der Datentransferobjekte als DataSet/Tabellen. Nutzung eines (Visual Studio) Code-Generators.	Stärkung der Anwendungsarchitektur durch Entkopplung des DataLayers und DataAccessLayers über einen O/R-Mapper. Mögliche Kopplung von Geschäftslogik an Entitäten.
Einsatzbereich	Anwendungen mit Spezialanforderungen an den Datenzugriff. (z.B.: hohe Performance, Strukturänderungen der Daten zur Laufzeit , Unterstützung unterschiedlichster DBMS ...).	Anwendungen, die wegen interner IT-Richtlinien nicht auf die aktuellste Version des .NET-Frameworks aufsetzen können. Anwendungen im mobilen Bereich auf Basis des MS .NET Compact Framework. Gelegentlich verbundene Anwendungen per ADO.NET Sync Framework. Verwendung in Web-Services, die tabellenstrukturierte Daten zur Verfügung stellen sollen.	Geeignet für alle Service-, Desktop- und Web-Anwendungen auf Basis des .NET-Frameworks. Hohe Synergien zu WPF und Silverlight durch DataBinding-Unterstützung. Im mobilen Umfeld nur bei Einsatz von Silverlight. Gute Eignung für objektbezogene Webservices.
Fazit	Einsatz wird nur bei speziellen Anforderungen empfohlen. Auf Grund des Implementierungsaufwands und der geringen Änderungsresistenz sollte ggf. über einen generativen Ansatz nachgedacht werden.	Einsatz wird nur bei speziellen Anforderungen oder umfangreichem Team-Know-How empfohlen. Sonst überwiegt der Implementierungsaufwand in Verbindung mit der vergleichbaren Komplexität und der mangelnden Mapping-Funktionaliät zu Gunsten des Entity Frameworks.	Wegen des hohen Funktionsumfangs, des generativen Ansatzes und geringen Implementierungsaufwands sollte das Entity Framework bevorzugt eingesetzt werden. Der höheren technischen Komplexität kann für Weiterentwicklung und Betrieb durch Schulung und Dokumentation begegnet werden.

So können Sicherheitskonzepte des IIS bei den Webservices und Reports eingesetzt werden oder die Serviceverwaltung des Windows 2008 R2 Servers über WCF-Services für eine Business-Tier verwendet werden.

Datentransfer und -anbindung

Datentransfer, -integration und -transformation sind bei jeder Aktion des angestrebten IT-Systems relevant. Daten aus Umsystemen müssen angebunden und ggf. umgewandelt werden, sie sind für das Berichtswesen zu verdichten und auszuwerten oder für einzelne Clients bereit zu stellen. Kaum eine andere Funktion in einer Geschäftsanwendung hat eine so direkte Auswirkung auf die Gesamtperformance, wie die ausgewählte Datentransfermethodik.

Neben der Frage, welche Daten für welche Aktion relevant sind, muss auch beantwortet werden, wie der eigentliche Datenaustausch stattfindet. Unsere Referenzarchitekturen unterscheiden im Bereich des Datentransfers zwischen Datenintegrations- und Datenzugriffsmechanismen. Auf beiden Ebenen müssen Anforderungsaspekte wie Sicherheitsprinzipien, Anpassungs- und Implementierungskomplexität, sowie Performance, Austauschbarkeit und Nachnutzbarkeit abgewogen werden.

Durch die MS SQL Server Integration Services lassen sich sowohl andere Datenbanken, als auch Unternehmenswebservices, externe Programmkomponenten oder Schnittstellendateien anbinden. Auf Seiten der Datenzugriffsmechanismen hat sich für Geschäftsanwendungen im Rahmen unserer Projekte das ADO.NET Entity Framework bewährt. Es bietet umfassende Zugriffsfunktionalität in Verbindung mit generierten Ansätzen und erlaubt dabei ein objektrelationales Mapping auf die internen Geschäftsobjekte bei überschaubarem Aufwand für die Datentransferlogik.

Das Ziel: langlebige Software

IT-Architektur und -Lösung sind dann erfolgreich, wenn sie ein bestehendes System ergänzen, sich nahtlos integrieren und zu einem systemweiten Mehrwert beitragen. Jede unserer .NET-Architekturen hat das Ziel, aktiv zum Mehrwert der IT-Systemlandschaft beizutragen und allen relevanten Anforderungen zu genügen.

Vorteile einer Architektur auf Microsoft-Basis

- Homogenisierung der Systemlandschaft mit hohen Synergieeffekten zwischen den eingesetzten Technologien
- Einfache Integration in bestehende IT-Infrastrukturen über Windows-Clients und MS Serverlösungen
- Nutzung von Architekturblaupausen und Best Practices von msg systems
- Sicherheit und Skalierbarkeit durch den MS SQL Server 2008 R2
- Langer Support- und Lebenszyklus der Microsoft-Technologien
- Integration oder Ablösung von MS Office-basierten Lösungen unter Beibehaltung des Look & Feel

Unsere Leistungen

- Konzeption, Architektur und Implementierung von mobilen und Unternehmensanwendungen auf Basis aktueller Microsoft-Technologien
- Bereitstellung von Best Practices, Referenzarchitekturen und Entwicklungsmustern für das .NET-Framework
- Integrations- / Migrationsplanung und Durchführung
- Architektur- und Anwendungs-Refactoring
- msg systems ist Microsoft Gold Certified Partner: Beratung in enger Abstimmung mit Microsoft bei der Technologie-Auswahl, auch über .NET hinaus.

msg systems ag
Robert-Bürkle-Straße 1 | 85737 Ismaning/München
Telefon: +49 89 96101-0 | Fax: +49 89 96101-1113
www.msg-systems.com | info@msg-systems.com

Microsoft Partner
Gold Server Platform
Gold Virtualization
Gold Business Intelligence
Silver Systems Management

L-030-1203-1