Java-Plattform als Datenrüssel

Salesforce Integration

Stefan Lipowsky, Detlev Eberhardt, Gunter May

Wer daten- oder rechenintensive Prozesse mit Salesforce.com bearbeitet, stößt bei der Cloud-basierten CRM-Lösung schnell auf Probleme. Eine Lösung liegt darin, komplexe Teilprozesse außerhalb von Salesforce zu implementieren. Bei dem hier vorgestellten Ansatz werden Datenstrukturen und Daten mit einer Java-Schnittstelle in eine lokale Datenbank übertragen. Dort werden sie verarbeitet, nur das Ergebnis wird anschließend ins Salesforce-System zurückgespielt. Mit diesem Vorgehen lassen sich auch Systemintegration oder Datenmigration leicht implementieren.

Salesforce.com ist mittlerweile Weltmarktführer bei Lösungen fürs Kundenbeziehungs-Management (Customer Relationship Management, CRM). Ohne Hard- und Softwareinstallationen erlaubt die Cloud-basierte Plattform gerade kleinen und mittelgroßen Unternehmen, ihren Vertrieb zu steuern und Kunden zu betreuen.

Abgesehen von kleinen Standalone-Szenarien bringt der Einsatz von Salesforce allerdings Herausforderungen mit sich, wenn die Lösung in die Systemlandschaft integriert oder mithilfe von Programmierung erweitert werden soll, wenn große Mengen von Dokumenten bei den dort gespeicherten Kundeninformationen hinterlegt werden oder wenn bestehende CRM-Informationen nach Salesforce migriert werden sollen.

Governor Limits und andere Einschränkungen

Zum einen schränkt die von Salesforce.com verwendete Programmiersprache Apex den Anwender durch sogenannte Governor Limits ein. Die Salesforce-Cloud ist ein Mandantensystem, auf dem parallel Nutzer mehrerer Kunden arbeiten (Multi-Tenancy). Die Governor Limits haben die Funktion, die Stabilität der Plattform zu gewährleisten, sodass jeder Kunde auf bestmögliche Performance vertrauen kann. Der Anwender nimmt Governor Limits allerdings auch als Einschränkung wahr: Sie begrenzen beispielsweise die Zahl der API-Calls in einem bestimmten Zeitraum, die zur Verfügung stehenden Ressourcen für Datenbankaufrufe und die Zahl der maximal zu verarbeitenden Einträge bei einer Anfrage an das CRM-System. Außerdem kann der Anwender nicht planen, wann Salesforce Batch-Jobs abarbeitet, da der genaue Ausführungszeitpunkt sich nicht beeinflussen lässt.

Weitere Herausforderungen sind: Salesforce erlaubt kein vollautomatisches Deployment. Das verursacht für die Entwickler hohen Aufwand. Zudem erhöht teilmanuelles Deployment die Fehleranfälligkeit. Die Pflege von Testsystemen in Sandboxes ist außerdem nicht sehr nutzerfreundlich: Beispielsweise erlaubt Salesforce einen Refresh der Full Copy Sandbox, also des Testsystems mit allen Daten, nur einmal im Monat. Darüber hinaus ist Speicher in der Salesforce-Cloud gegenüber lokalem Speicher sehr teuer – um den Faktor 100 oder mehr.

Um Herausforderungen dieser Art zu umschiffen und die Salesforce-Cloud an die bestehende Systemlandschaft anzubinden, stehen Unternehmen Integrationsplattformen mit hohem Funktionsumfang zur Verfügung – darunter Informatica Cloud for Salesforce Integration und Magic xpi. Beim Einsatz beider beispielhaft genannten Lösungen fallen allerdings Lizenzkosten an. Außerdem muss das Anwender-Unternehmen Know-how für die Plattform-Konfiguration aufbauen.

Der hier beschriebene Ansatz stellt eine leistungsfähige und vergleichsweise kostengünstige Alternative auf Basis von Java vor, die zudem problemlos zu integrieren ist. Wir gebrauchen dafür den Begriff *Datenrüssel*. Dieses Bild veranschaulicht die Funktionsweise des Lösungsansatzes: Vereinfacht gesprochen werden Daten aus der Salesforce-Plattform in der Cloud mittels Java wie über einen Rüssel abgesaugt und in eine lokale Datenbank übertragen. Dort werden die replizierten Daten wie gewohnt verarbeitet, gegebenenfalls unter Einbindung weiterer Informationen aus anderen Systemen. Nur das Ergebnis wird anschließend wieder an Salesforce.com übertragen. Mit der Java-basierten Verarbeitung der Daten im eigenen Unternehmen kann der Anwender die Begrenzungen der Salesforce-Plattform überwinden.

In folgenden Fällen ist dieses Integrations-Konzept der richtige Ansatz:

1. Erstellen komplexer Geschäftslogik zum Verarbeiten von Daten aus Salesforce, gegebenenfalls mit weiteren Daten aus operativen Systemen
2. Ablage vieler und großer Dokumente
3. Integration von Salesforce in die Systemlandschaft und Daten- oder Nachrichtenaustausch zwischen operativen Systemen und Salesforce
4. Laden von Kundendaten in Salesforce nach einer komplexen Transformation oder Migration

Anwendungsbeispiele aus unserer Projekterfahrung bei it-economics belegen die Vorteile unseres Ansatzes auf diesen vier Themenfeldern.

1. Abbildung komplexer Geschäftslogik

In einem unserer Projekte sollten Abrechnungen erzeugt werden. Dazu waren Kunden- und Vertragsdaten (gespeichert im Salesforce-CRM), Verbrauchsdaten und eine Abrechnungslogik notwendig, die die verschiedenen Produkte, ihre Abrechnungsweise und kumulierende Rabatte verarbeiten konnten.

Die komplexe Abrechnungslogik zu implementieren, war mit Salesforce-Bordmitteln eine Herausforderung. Apex ist zwar an Java angelehnt, besitzt aber starke Einschränkungen. Begrenzt ist etwa die Anzahl der Zeilen, die in einer Liste zurückgegeben werden können, ebenso die Zahl der Objekte (Klassen) und der Abfragen pro Aufruf. Die Entwicklungswerkzeuge bieten zum Beispiel keine Autovervollständigung und kein Debugging. Das Deployment ist nicht vollständig automatisierbar – etwaige „Custom Settings“ müssen vor dem Deployment manuell angelegt werden, sofern sie von Testern

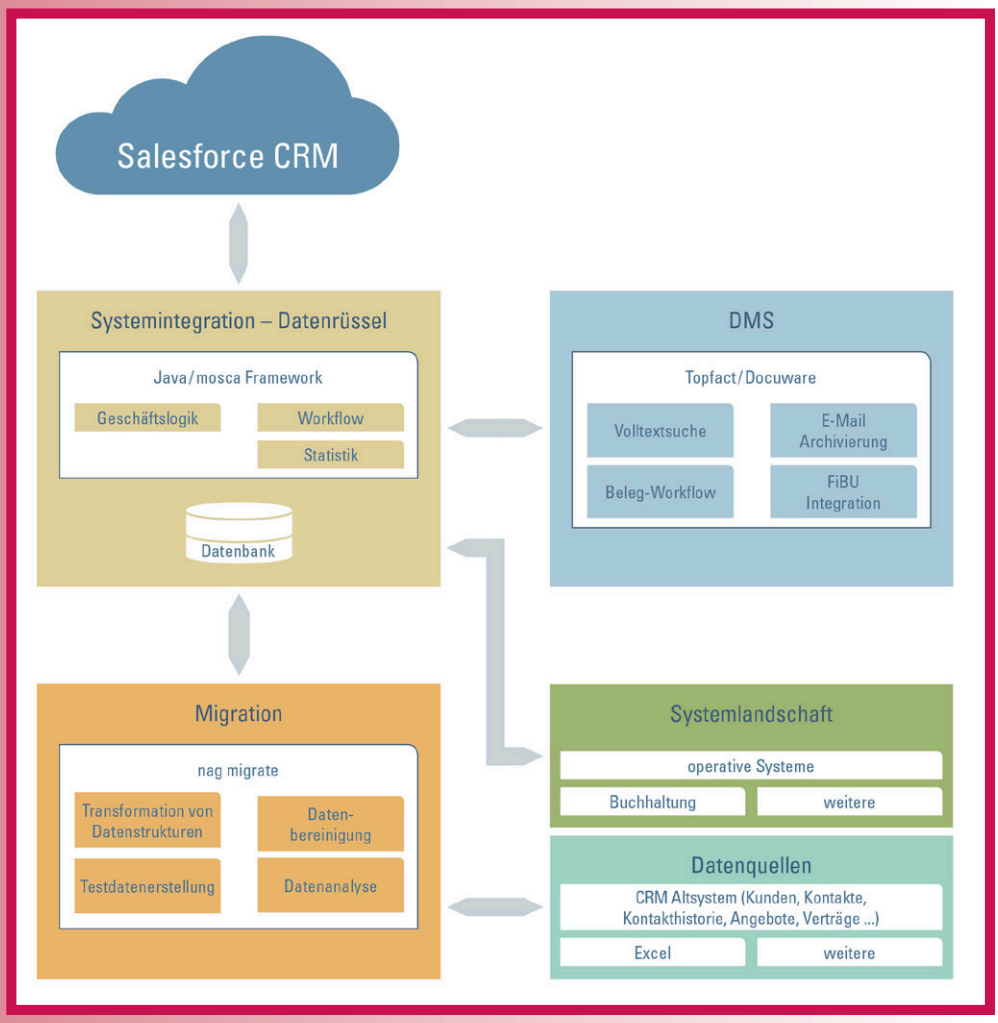


Abb. 1: Integrationsprozess

verwendet werden. Die Zusammenführung von Entwicklungssträngen zu einem deploybaren Artefakt erfordert manuelles Eingreifen, somit ist Continuous Integration nur unter großen Schwierigkeiten realisierbar.

Daher entschieden wir uns, die Kunden- und Vertragsdaten aus Salesforce über den Datenrüssel in eine lokale Zwischen-Datenbank zu laden. Diese Daten ändern sich vergleichsweise selten, es fällt folglich wenig Datenvolumen zum Transfer an. In dieselbe Datenbank luden wir die Verbrauchsdaten. Die Datenbank kann auf einem Server im Firmennetzwerk zu günstigeren Preisen skalieren – und oft bietet die bestehende Infrastruktur weitere Vorteile: Die Systeme sind beispielsweise von der internen Revision bereits abgenommen und in die firmenweite Backup-Strategie eingebunden. Direktzugriff der Administratoren bietet zudem nahezu unbeschränkte Möglichkeiten zum Performance-Tuning. Die Abrechnungslogik bauten wir in einem Modul innerhalb der Integrationsplattform als Java-EE-Applikation.

Hier konnten wir mit modernen Tools und Methoden wie Test-Driven Development (TDD) und Continuous Integration unsere Applikation funktionierend, performant und wartbar entwickeln. Die erzeugten Rechnungsdaten übergaben wir per Datei- und Webservice-Schnittstelle an weiterführende Systeme (PDF-Renderer, Rechnungsversand und Buchungssystem). Die Rechnungs-PDFs wurden in einem Dokumentenmanagement-System (DMS) archiviert.

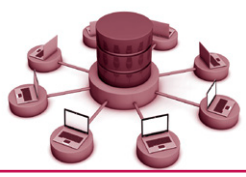
Nach Salesforce luden wir nur noch wichtige Daten (z. B. Rechnungssumme für den Monat) hoch und verlinkten das im DMS abgelegte Rechnungs-PDF. Über einen PDF-Betrachter kann der Benutzer sich das Dokument anzeigen lassen, ohne Salesforce zu verlassen. Den PDF-Betrachter entwickelten wir mithilfe von Force.com Canvas, einem Set von Werkzeugen und JavaScript-APIs. Sie erlauben es, Anwendungen in Salesforce.com zu integrieren. Auf DMS-Seite hat ein Wirtschaftsprüfer vollen Zugriff auf verschlagwortete Dokumente. Damit konnten wir ohne Einschränkungen beim Benutzer-Komfort in Salesforce Speicherplatz sparen und gleichzeitig die Anforderungen der internen IT nach Datensicherheit und Stabilität sowie die der internen Revision nach Revisionssicherheit erfüllen und die Grundsätze ordnungsmäßiger Buchführung (GoB) einhalten.

Ein weiterer Vorteil der lokalen Datenverarbeitung liegt im komfortableren Debugging. Debugging läuft bei Salesforce.com online in der Developer Console und mithilfe von Debug-Logs. Der erstellte Code lässt sich ausführen, ein Blick in die Console oder das Debug-Log zeigt mögliche Fehler. Allerdings lassen sich beim Debugging auf Salesforce.com keine Breakpoints setzen. Unter Java ist das dagegen möglich. Die Breakpoints erlauben es, schrittweise vorzugehen und jeden Zustand zu inspizieren.

2. Dokumentenablage

In unserem Projekt wurden E-Mails von Kunden zu Calls in Salesforce konvertiert. Viele Mails enthielten Attachments, etwa eingescannte Dokumente, die oft 1 MB groß oder größer waren. Unser Kunde nutzte die Salesforce Enterprise Edition, bei der jeder Benutzer, in diesem Fall Service-Team-Mitarbeiter, 20 MB Speicher hat. Ein Mitarbeiter bearbeitet eine Mail in durchschnittlich 2,5 Minuten und kommt damit auf knapp 200 pro Tag. Schon wenn nur an jeder zwanzigsten Mail ein Attachment hängt, ist sein Speicher auf Salesforce.com in zwei Tagen verbraucht.

Auch bei den Verbrauchsdaten aus operativen Systemen wird die Datenmenge schnell zum Problem: Für jeden der 50.000 in Salesforce gespeicherten Kunden liefern die operativen Systeme pro Tag einen Datensatz mit Verbrauchswerten. In Salesforce benötigt ein Datensatz 10 KB, somit sind 500 MB Speicherplatz täglich nötig (> 150 GB jährlich). In Salesforce sind in jedem Benutzeraccount 612 MB Speicherplatz in der Grundgebühr enthalten – hat ein Unternehmen durch 100 Salesforce-Accounts 61 GB Speicherplatz, wäre dieser in weniger als 100 Tagen belegt. Daher gilt es, die Daten außerhalb von



Salesforce auf die für die Kundenbetreuung notwendigen Werte zu reduzieren, etwa Durchschnitts- oder Spitzenwerte, Rechnungssummen oder offene Zahlungen. Diese Daten lassen sich bei unserem Ansatz innerhalb der Datenbank außerhalb von Salesforce errechnen und nach Salesforce hochladen.

Die lokale Ablage im DMS – in unserem konkreten Projekt setzten wir eine Lösung von Topfact ein – erlaubt dem Kunden zudem eine Verschlagwortung etwa nach Datum, Name, Dokumententyp. Per Texterkennung lassen sich eingescannte Dokumente zusätzlich automatisch indizieren, sodass auch die Volltextsuche möglich ist. Dies kann die Recherche, ob beispielsweise ein Kunde ein Kündigungsschreiben geschickt hat, erleichtern. Auf der Salesforce-Plattform wird nur ein Link zum Dokument hinterlegt.

Weiteres Argument für den Einsatz eines lokalen DMS: Dokumentenablage auf nicht revisionssicheren amerikanischen Servern ist nicht jeder internen Revision gut erklärbar. Manche Unternehmen wollen bestimmte Daten zudem aus grundsätzlichen Erwägungen nicht außerhalb der eigenen IT-Infrastruktur speichern oder zumindest nicht auf Cloud-Server außerhalb Deutschlands oder Europas übertragen. Salesforce.com stellt für Ende 2014 die Fertigstellung eines europäischen Rechenzentrums in Großbritannien in Aussicht. Den Bau eines Rechenzentrums in Deutschland hat das Unternehmen ebenfalls angekündigt. Ob damit – in einem Multitenant-System – die Anforderungen an Daten- und Revisionssicherheit erfüllt sind, ist allerdings im Einzelfall zu prüfen.

3. Integration von Salesforce in die Systemlandschaft

In unserem Projekt wurden Kunden-Vertragsdaten in Salesforce gespeichert. Dazu mussten auch Produkte, mögliche Produktkombinationen und Rabattierungen in Salesforce hinterlegt werden. Kunden konnten hier Kontingente zur Benutzung der operativen Internet-Systeme eines Online-Handelsportals erwerben. Um die Kunden auf die operativen Systeme zu berechtigen, musste Salesforce diese Systeme benachrichtigen und relevante Kunden- und Vertragsdaten – wie Namen, Login, gekauftes Produkt, Vertragslaufzeit – übertragen.

Salesforce-Bordmittel bieten hier Möglichkeiten, mit denen beispielsweise bei Änderung eines Vertrages der Datensatz per Webservice an das operative System übergeben werden kann. Große Unsicherheit herrschte allerdings bei möglichen Massenänderungen. Diese kommen vor, weil Kunden oft ihre Leistungen zum nächsten Monat neu beziehen, ändern oder kündigen. Daher werden derlei Vertragsänderungen in Salesforce gespeichert und am Monatswechsel zum operativen System übertragen. Mit Salesforce-Mitteln wären wir auf Salesforce-Batches angewiesen, die nicht kontrollierbar laufen, und wir wären bei komplexeren Datensammlungen auf Salesforce-Programmierung angewiesen, die schwer zu beherrschen ist.

Um dies zu umgehen, ließen wir Salesforce bei Vertragsänderung nur eine Zeile in eine Steuertabelle schreiben: Welcher Vertrag soll zu welchem Zeitpunkt übertragen werden? Da dies jeweils bei Vertragsänderung passiert – und sich diese Vertragsänderungen über den Monat hinweg gleichmäßig verteilen –, ist das Schreiben dieser Information kein Problem.

In unserem Projekt wurden diese Informationen gesammelt und zum Monatswechsel an die operativen Systeme übertragen. Da die relevanten Kunden- und Vertragsdaten ohnehin in unserer Zwischendatenbank gespiegelt vorlagen, konnten wir die Daten-Sammel-Logik und die Erzeugung der notwendigen XML-Nachrichten-Dateien in Java implementieren. Die lokalen Server ermöglichten uns volle Kontrolle und nahezu unbegrenzte Skalierbarkeit.

4. Komplexe Transformationen/Migration

In unserem Projekt wollte ein Unternehmen sein selbstgebautes CRM-System durch Salesforce ablösen. Die über Jahre im eigenen Haus entwickelte und gepflegte Lösung enthält neben dem eigentlichen CRM-System auch Komponenten zu Vertragshaltung, Abrechnung, Statistik und Integration. Solche gewachsenen, komplexen Strukturen sollten vor der Migration der Daten nach Salesforce an dessen Datenstrukturen angepasst werden. Datenbestände sind zu bereinigen und zu validieren – manche Unternehmen haben sogar mehrere CRM-Systeme gepflegt und müssen Datenbestände vor der Migration von Dubletten säubern und Daten aus verschiedenen Quellen konsolidieren. Im nächsten Schritt müssen auf Grundlage der Analyse der Daten die verwendeten Produktkombinationen, Abrechnungsvarianten und unter Umständen Rabattkombinationen im Salesforce.com-System und der Abrechnungsmaschine (s. o.) implementiert werden.

Um Migration und Funktionalität testen zu können, ist es notwendig, relevante Testdatenbestände aus den Daten zu ermitteln und die verschiedenen Systeme immer wieder auf einen konsistenten Anfangsstand zurücksetzen zu können.

Um die Datenbestände zu analysieren, zu bereinigen und zu transformieren, bietet sich ein Werkzeug an, das sich an Geschäftsobjekten, beispielsweise „Kunde mit seinen Verträgen“, orientiert. ETL-Tools gehen dagegen tabellenweise vor: Zuerst werden die Kunden migriert, dann deren Verträge. Geht im zweiten Schritt etwas schief, ist der erste Schritt bereits abgeschlossen.

Anders verhält es sich, wenn der komplette logische Baum eines Geschäftsobjektes migriert wird. Bei dieser Vorgehensweise kann direkt nach dem Transformationsschritt eines Geschäftsobjektes (z. B. Kunde) durch Konsistenzprüfungen die Validität ermittelt werden. Passen etwa die Rechnungsdaten nicht zum Vertragssaldo, kann genau dieser Datensatz und genau dieser Kunde ausgesteuert werden. So können geprüfte und richtige Daten ins neue System übernommen werden, die Fehlerfälle weiter analysiert und die Migrationsregeln für diese Fälle angepasst werden.

Das in unserem Projekt eingesetzte Migrationstool nag migrate erlaubt neben grafischer Erfassung einfacher und mittlerer Migrationsregeln die Programmierung von komplexen Regeln. Es bietet Unterstützung zur Datenanalyse und damit zum Auffinden von Sonder- und Spezialfällen. Damit können leicht repräsentative Testdatenbestände definiert werden.

Da bei unserem Integrationsansatz Datenstrukturen und Daten aus Salesforce in die lokale Datenbank kopiert und durch einen Automatismus aktuell gehalten werden, lässt sich die Migration immer gegen den aktuellen Datenstruktur- und Datenstand entwickeln und testen: Sie ist nicht mehr beschränkt dadurch, dass sie eine Full Copy Sandbox nur einmal im Monat abziehen kann.

Vorteile des Datenrüssel-Ansatzes

Die Beispiele machen die Vorteile des beschriebenen Ansatzes deutlich. Je spezieller die Anforderungen eines Unternehmens sind, desto schneller stoßen die Anwender an die Beschränkungen der Salesforce-Plattform. Mit dem Ansatz, komplexe Prozesse außerhalb der Cloud lokal zu implementieren, lässt sich das Salesforce-CRM an individuelle Anforderungen anpassen – und zwar innerhalb der eigenen Systemlandschaft.

Die Java Enterprise Edition (JEE) ist für solche komplexen Entwicklungs- und Wartungsaufgaben sehr gut geeignet, da sehr vielfältige und mächtige Werkzeuge und Bibliotheken zur

Verfügung stehen. Weltweit organisierte Benutzergruppen haben für nahezu jede Problemstellung Lösungs-Skizzen veröffentlicht. Die benötigten Java-Bibliotheken sind zudem in der Regel ohne Lizenzkosten verfügbar. Hinzu kommt: Mit den Integrationsplattformen Magic xpi und Informatica Cloud for Salesforce Integration vertraute Fachleute sind schwer zu finden. Auch auf Salesforce.com spezialisierte Entwickler sind auf dem Markt rar und verlangen deutlich höhere Stundensätze als die in hoher Zahl verfügbaren Java-Experten.

Für die Umsetzung des dargestellten Ansatzes ist es allerdings notwendig, auf Java-Experten mit Java-EE-Know-how und Integrationserfahrung zurückzugreifen. Die komplexe Kopplung von Systemen, das disziplinierte Entwickeln etwa nach TDD- und Clean-Code-Ansatz, die Erfahrung mit verschiedensten Schnittstellen und Technologien ist Voraussetzung. Systemübergreifende Tests erfordern neben den Testsystemen auch Erfahrung in Testmanagement und Testfallerstellung sowie Organisation von komplexen Projekten.

Links

[CRMMarkt] Marktdaten für Customer-Relationship-Management-Software, `https://www.gartner.com/doc/2711518` und `http://www.forbes.com/sites/louiscolumbus/2014/05/06/gartners-crm-market-share-update-shows-41-of-crm-systems-are-saas-based-with-salesforce-dominating-market-growth/`

[GoBS] Grundsätze ordnungsmäßiger DV-gestützter Buchführungssysteme, `http://www.bundesfinanzministerium.de/Content/DE/Downloads/BMF_Schreiben/Weitere_Steuerthemen/Betriebspruefung/015.pdf?__blob=publicationFile&v=3`

[SalesfoceAnn] Ankündigung von Salesforce.com zur Errichtung eines Rechenzentrums in Europa, `http://www.salesforce.com/de/company/news-press/press-releases/2013/05/130506-2.jsp`

[SalesfoceAPILimit] API-Verwendungsgrenzen, `http://help.salesforce.com/apex/HTViewHelpDoc?id=integrate_api_rate_limiting.htm&language=de`

[SalesfoceForcecom] Das Canvas-Framework von Salesforce.com, `http://www.salesforce.com/us/developer/docs/platform_connectpre/canvas_framework.pdf`

[SalesfoceJavaApex] Unterschiede zwischen Java und Apex, `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_classes_java_diffs.htm`

Stefan Lipowsky ist seit 1999 in der IT-Branche als Entwickler, technischer Architekt, und Projektleiter auf den Gebieten Versicherung und Finanzdienstleister, Automobil, Internet, Telekommunikation und Medien tätig. Seit 2009 arbeitet er als Senior Manager gemeinsam mit Kollegen der it-economics GmbH im Projektmanagement. Spezialisiert auf klassische und agile Projektmanagementmethoden betreut er traditionell geführte sowie komplexe agile Projekte mit bis zu 250 Mitarbeitern und leitet Softwareentwicklungs- und Integrationsprojekte, u. a. im Salesforce-Umfeld. Stefan Lipowsky ist erster Ansprechpartner einiger Key Accounts des Unternehmens und Lehrbeauftragter an der Hochschule für angewandtes Management in Erding.
E-Mail: stefan.lipowsky@it-economics.de

Detlev Eberhardt ist geschäftsführender Gesellschafter der pro effectus GmbH. Er begann seine Karriere mit spezifischen Konzernlösungen, wie Logistiklösungen für Hewlett Packard, und arbeitete im Contract-Management für die Württembergische Gebäudeversicherung. Ab 2001 erfolgte die Ausrichtung auf strategische Anbindungen von Dokumentenmanagement-Systemen (DMS) zu datenführenden Systemen von Kundenprozessen. 2005 wurde die pro effectus GmbH zur Entwicklung und Markteinführung von topfact Dokumentenmanagement als strategische Anwendung für Transparenz und Nachhaltigkeit im Geschäftsprozess gegründet.
E-Mail: vertrieb@topfact.de

Gunter May ist Mitglied der Geschäftsleitung der nag. Dort ist er für das Geschäftsfeld Datenmanagement sowie für den Vertrieb zuständig. Zuvor hat er bereits langjährige Erfahrung in mehreren Führungspositionen in der Versicherungsbranche und bei IT-Dienstleistern gesammelt. In den letzten Jahren hat er die nag-Lösungen im Umfeld Daten-Migration/Analyse/Management in Verbindung mit dem Werkzeug nag migrate am Markt in vielen Projekten erfolgreich eingesetzt.
E-Mail: gunter.may@nag.ch