Reporting in Migrationsumgebungen ohne Umwege

Karsten Meyer-Kühl
IT-P GmbH
Hannover

Schlüsselworte
Reporting, Migrationen, Oracle Reports, Oracle Forms, APEX, ADF, JET, JasperReports, BIRT

Einleitung
In modernen Web-Anwendungen werden häufig Tools wie BIRT oder JasperReports zur Erstellung von Berichten oder Schreiben eingesetzt. Doch wie gestalte ich eine Systemarchitektur, die mir ermöglicht, flexibel zu bleiben und mich nicht langfristig auf die ausgewählten Tools festlegt?
Wie minimiere ich Abhängigkeiten von den eingesetzten Frameworks, die sich kontinuierlich entwickeln und von denen oft mehrere gleichzeitig genutzt werden sollen?
Wie kann ich während langer Migrationsphasen - beispielsweise ausgehend von Forms und Reports - dafür sorgen, dass ich nicht alles auf einmal umstellen muss, sondern erstellte Reports sowohl in der alten als auch in der neuen Umgebung nutzen kann?
Wie ermögliche ich dem Poweruser ohne tiefgehende Programmierkenntnisse selbst Reports zu verändern oder hinzuzufügen?
Und wie geht das alles wartungsfreundlich und bei überschaubaren Aufwänden und Kosten?
Anhand eines Migrationsprojekts von Forms in eine heterogene Umgebung mit ADF-, APEX- und JET-Komponenten werden Antworten auf all diese Fragen gegeben.

Ausgangsposition
Die thematische Ausgangsposition wird durch eine mittelgroße Anwendungslandschaft eines mittleren Unternehmens repräsentiert. Die Anwendungssuite wurde vor vielen Jahren mit Client-Server-Technologie aufgebaut und basiert auf Forms und Reports. Ein Beispiel, was wir bis heute häufig vorfinden.
Dabei handelt es sich meistens um Individualsoftware, die früher aus der Not nicht verfügbarer Standardsoftware heraus entwickelt wurde, heutzutage aber durch ständige Verfeinerung hin zu optimal angepassten Prozessen auch als Wettbewerbsvorteil gesehen wird.
Aus allseits bekannten Gründen, die in diesem Vortrag aber nicht vertieft werden, entstand – in den meisten Fällen schon vor einigen Jahren – der Wunsch, auf eine aktuelle Technologielandschaft zu migrieren. Dieser Wunsch wird nun immer dringlicher, so dass Migrationsprojekte in vielen Unternehmen anzutreffen sind oder zumindest geplant werden.
Als Zielplattform wird heute in der Regel eine Landschaft von Web-Anwendungen angestrebt, die sich nicht, wie in der Vergangenheit zu oft geschehen, als statisch erweisen darf. Stattdessen soll sie sich stets mit sich wandelnden Prozessen inhaltlich und auch mit aktuellen Trends technologisch und optisch weiterentwickeln und so immer modern erscheinen.
Man ist also gezwungen, auf Technologien und Architekturen zu setzen, die es ermöglichen, Komponenten zu aktualisieren oder auszutauschen, ohne das Gesamtsystem als solches verändern zu müssen.
Dass dabei der Einsatz von agilen Methoden in der Softwareentwicklung nicht nur sinnvoll, sondern schon fast zwangsläufig ist, sei an dieser Stelle nur eine Nebenbemerkung, deren Ausführung selbst einen eigenen Vortrag wert wäre.
Als aktuelle Technologien für Individualsoftware kommen in der Oracle-Welt zurzeit speziell ADF, APEX und JET in Frage. Selbstverständlich steht uns aber auch die ganze Java-Welt zur Verfügung, insbesondere im Reporting-Umfeld Frameworks wie JasperReports oder BIRT. Oracle BI-Technologien bleiben in diesen Kontexten häufig unberücksichtigt, da sie im Vergleich zu den

genannten kostenfreien Frameworks hohe Anforderungen an die Systemlandschaft und das Budget stellen.
Die bewährte Oracle Datenbank und oft auch zumindest in wesentlichen Teilen das Datenmodell sollen bestehen bleiben. Relevant ist zudem der Investitionsschutz, der nicht selten dazu führt, dass auch Unternehmenslogik wie z.B. Schnittstellen oder Rechenkerne, die in PL/SQL vorliegen, weiterverwendet werden sollen.
Da wir als Ausgangsposition von mittelgroßen Umgebungen ausgehen, müssen stets Kapazitäten und Budget beachtet werden. Daher können wir auch keine große SOA-Landschaft aufbauen und dabei z.B. die Oracle SOA-Suite zum Einsatz bringen, sondern müssen uns überschaubare Lösungen überlegen.
Gleichzeitig ist es meistens aus verschiedenen Gründen ebenfalls unmöglich, die komplette Landschaft auf einmal umzustellen, da in der Regel Analyse- und Testkapazitäten bei Anwendern und Unternehmens-IT nur begrenzt abrufbar sind.
In Bezug auf unser Thema kommt noch ein wichtiger Punkt hinzu, der ebenfalls sehr oft zu beobachten ist: Reports ändern sich häufiger als Forms, da sie als Schreiben und andere Dokumente verwendet werden und daher Außenwirkung, Rechts- oder Entscheidungsrelevanz haben. Lange Entwicklungszyklen sind nicht erlaubt, oft überholen Anforderungen die geplanten Releasezyklen. Daher müssen Reports nicht nur schnell angepasst werden, sondern auch schnell und ohne große Systemtests ausgetauscht oder ergänzt werden können. Oft will der interne oder externe Kunde dies auch selbst tun, um sich die maximale Flexibilität und Schnelligkeit zu bewahren. Dafür ist der Einsatz von agilen Methoden notwendig, aber noch nicht ausreichend, da man ja die aufrufenden Prozesse oft nicht losgelöst von anderen migrieren kann.
Ist die Migrationsentscheidung einmal getroffen, so möchte man möglichst wenig in die „alte Welt“ investieren. Dementsprechend brauchen wir eine Lösung, die es uns ermöglicht, Reports einzubinden, die unmittelbar funktionieren, gleichzeitig aber auch bei Migration der restlichen Umgebung unverändert bestehen bleiben können.

Konkrete Situation im Kundenumfeld des Beispiels
Im konkreten Beispiel geht es um eine integrierte Anwendungssuite für individuelle Geschäftsprozesse mit Oracle Forms und Oracle Reports, die bereits seit 1998 entwickelt wird und seitdem regelmäßig Redesigns und laufende Ergänzungen erfahren hat.
Dem System zugrunde liegt eine Oracle Datenbank, in der ein Unternehmensdatenmodell sowie Geschäfts- und Schnittstellenlogik angesiedelt sind. Der Zugriff darauf erfolgt sowohl über die Anwendungssuite selbst, als auch von externen Anwendungen – wie z.B. Portalen.
Über die Jahre gewachsen sind außerdem einige Funktionalitäten, die durch Konfigurationen erlauben, die Funktionsweise der Anwendungen zu steuern. Basis ist dabei eine Versionsverwaltung der Komponenten, um über ein Konfigurationsmanagement zu verfügen. Weiterhin sind ein Auswertungssystem, ein Regelsystem, etwa für tabellenübergreifende Integritätsbedingungen, sowie ein Job- und Funktionensystem zu nennen, mit dem im Zusammenspiel mit der Komponentenverwaltung Reports, Schnittstellen, Exporte, Mailings, Berechnungsläufe und einiges mehr einheitlich konfiguriert und gesteuert werden.
Gedanken über die Migration der Anwendungssuite machte man sich schon eine lange Zeit, jedoch hat sich die bestehende Landschaft als durchaus stabil und auch gut anpassbar herausgestellt. Dabei ergab sich aus finanziellen Erwägungen die Notwendigkeit, möglichst große Teile der Geschäftslogik weiterzuverwenden soweit es eine moderne Architektur zulässt. Insbesondere führten diese Überlegungen dazu, dass die Oracle-Datenbank nicht in Frage gestellt wurde und man nach Lösungen in der Oracle-Welt gesucht hat.
Im Laufe der Jahre ist der Migrationsdruck wie in vielen vergleichbaren Konstellationen stark angewachsen, so dass man sich eine Migration der gesamten Anwendungsumgebung innerhalb einer Projektlaufzeit von fünf Jahren vorgenommen hat. Als Haupttechnologie für die Abbildung der

Geschäftsprozesse wurde dabei ADF gewählt, für die Pflege der Konfiguration kommt APEX zum Einsatz.
Inzwischen sind bereits mehrere neue Module mit ADF ergänzt worden und die ersten Forms-Anwendungen werden dieses Jahr ersetzt.
Nach Start des Projektes vor ein paar Jahren verstärkte sich der Wunsch, Teile einiger Geschäftsprozesse auch von den Endkunden selbst durchführen bzw. beantragen zu lassen. Dafür stellte sich in der Analyse eine Architektur mit JET als optimal heraus, so dass man eine weitere Technologie im Portfolio hat.
Als strategisches Reportingtool wurde bereits vor der eigentlichen Migrationsentscheidung BIRT ausgewählt, insbesondere, weil dazu bereits Know How beim Kunden vorhanden war. BIRT-Reports müssen dabei sowohl von der alten Forms-Anwendung als auch von den neuen Technologien aus benutzbar sein.

Lösungsbedingungen
Als Bedingungen für unseren Lösungsansatz ergeben sich daraus einige Punkte:
Zum einen ist ein modularer Ansatz für die Reports unabdingbar. Vor allem muss die Reportingtechnologie unabhängig von der einzusetzen Technologie für die Dialoganwendungen sein. Wir benötigen also eine zentrale Stelle, um Reports anzusteuern. Als gemeinsame Basis finden wir unter diesen Bedingungen die Oracle Datenbank, die von allen Komponenten verwendet wird.

Lösungsansatz
Aufgrund dieser Bedingungen wurde ein Lösungsansatz entwickelt, der bereits vor Beginn der eigentlichen Migration der Anwendungen umgesetzt wurde.
In der Datenbank wurde ein Repository mitsamt zugehöriger API eingeführt, welches Aufträge für Reports entgegennehmen kann und diese Aufträge zur Ausführung bringen kann. D.h. eine beliebige Anwendungskomponente kann in der Datenbank einen Reporting-Auftrag ablegen und die Ausführung des Reports verlangen. Alternativ kann die Ausführung des Reports technisch gesehen auch von einer anderen Komponente angefordert werden. Damit können wir erzeugte Dokumente auch dann an der Benutzeroberfläche anzeigen, wenn diese z.B. während der Durchführung einer Datenbankfunktion erzeugt werden.
Zusammen mit dem in diesem Fall bereits vorhandenen Komponentenrepository sind wir nun schon in der Lage, die Reports mit den zugehörigen Parametern als Komponente zu konfigurieren. Dazu wird dann eine „Reporting-Funktion“ definiert, über deren Bezeichner der Report aufgerufen werden kann. Der Auftrag umfasst dementsprechend nicht nur die Reporting-Funktion an sich, sondern auch die Parameter des Reports sowie Umgebungsinformationen, z.B. welcher Anwender den Report angefordert hat.
Für die Ausführung der Reports darf es konsequenterweise nur eine zentrale Komponente geben, welche die Aufträge liest und von den Anwendungen der verschiedenen Technologien über https, also als Web-Service, beauftragt wird, das gewünschte Dokument bereitzustellen. Neben den Anwendungskomponenten der eigentlichen Applikationssuite können Reports auch direkt aus der Datenbank heraus oder aus Satellitenanwendungen aufgerufen werden, die möglicherweise in noch ganz anderen technologischen Umgebungen existieren.
Dieser Report-Server stellt nicht nur das Ergebnisdokument zur Verfügung, sondern kann es auch direkt in der Datenbank oder im Dateisystem ablegen und die Durchführung zum Auftrag als logging-Information speichern.
In der Regel wird man das erzeugte Dokument (zumeist pdf) nicht direkt vom Web-Service zurückgeben lassen, sondern einen Status oder einen Link auf das Dokument, so dass die aufrufende (Web-)Anwendung das Dokument ordnungsgemäß dem Anwender in der Web-Anwendung zum Download oder zur Anzeige anbieten kann.

Weitere Einsatzszenarien
Das vorgestellte Prinzip ist im eigentlichen Sinne technologieunabhängig, sowohl was Masken als auch was Reports angeht. Es funktioniert in der genannten Oracle-Umgebung optimal, eine Einschränkung darauf ist aber nicht vorhanden.
Ebenso sind wir nicht auf Migrationsszenarien beschränkt. Der Ansatz ist genauso sinnvoll für beliebige andere heterogene Umgebungen, die mehr als eine Technologie im Einsatz haben und Flexibilität benötigen. Derartige Projekte sind auch schon mehrfach durchgeführt worden. Insbesondere ist man so auch für zukünftige Migrationen gewappnet.

Weitere Vorteile der Lösung
Die Verallgemeinerung bzw. Zentralisierung der Reportingaufrufe ermöglichen die Nachvollziehbarkeit der Vorgänge durch das Logging der Aufträge und deren Durchführung. Zudem kann unmittelbar geklärt werden, welche Reports aktuell überhaupt noch eingesetzt werden – eine Frage, die sich nicht nur bei Migrationen häufig stellt.
Über die Verwaltung und Versionierung der Reports und Reporting-Funktionen im Repository (Configuration Management) können wir überschaubare und unabhängige DevOps-Prozesse für die Reports implementieren, die allen Qualitätskriterien genügen.

Vorstellung der konkreten Lösung
Wir schauen uns nun das Datenmodell für die Konfiguration der Reports an. Der zweite Teil des Datenmodells enthält die Aufträge und die Auftragsdurchführung, d.h. das Logging und die Dokumente.
In der Architektur des Systems haben wir als zentrale Programmkomponenten zum einen die API in PL/SQL-Packages, welche auch als Web-Services gekapselt werden können. Das ist für Forms- und ADF-Anwendungen nicht notwendig, ist aber bei JET-Anwendungen, die möglicherweise direkt bei Endkunden im Browser laufen, sinnvoll.
Außerdem haben wir den eigentlichen Report-Server, welcher die Report-Ausführung als Service bereitstellt. Dieser ist als Java-Anwendung implementiert und beherbergt sowohl eine Jasper- als auch eine BIRT-Serverkomponente. Weitere Reportingsysteme wären hier möglich.
Hier sei der Hinweis erlaubt, dass dies nur dann in einem Artefakt realisierbar ist, wenn man das „Glück“ hat, dass sich keine Probleme bei Versionen gemeinsam genutzter Bibliotheken ergeben. Letztlich reicht aber in den meisten Umgebungen auch die Implementierung einer Reporting-Technologie für die komplette Anwendungsumgebung aus. Alternativ kann man in einem zweistufigen Verfahren auch wieder über Web-Services die eigentlichen Report-Engines aufrufen. So wären auch Aufrufe von Oracle Reports, BI Publisher etc. möglich.
Es ist auch denkbar, die Architektur weiter zu verschlanken, indem der Report-Server nicht auf einem Applikationsserver angesiedelt wird, sondern als Java-Klasse direkt in der Datenbank läuft. Diese Idee entsteht vor allem in kleinen Umgebungen, die z.B. als APEX-Anwendung ohnehin ohne einen schwergewichtigen Applikationsserver auskommen. Hier ist aber die Versionsproblematik noch diffiziler, das beginnt bereits bei der in der Datenbankversion und der damit verbundenen Java-Version. Erfahrungsgemäß spannend sind in dieser Konstellation vor allem die Verwendung von Systemressourcen, insbesondere von speziellen Schriften in den Reports, so dass diese Architektur nicht unbedingt empfohlen werden kann.
Die Konfigurationsanwendung an sich ist eine APEX-Anwendung, hier kann man nicht nur die DevOps-Prozesse durchführen, sondern auch die Historie anschauen sowie über verallgemeinerte Parametermasken die Reports ausführen. Im Sinne unserer Architektur bedeutet das entsprechend, dass im Hintergrund ein entsprechender Auftrag angelegt und durchgeführt wird.
Zu diesen genannten Komponenten schauen wir uns Beispiele für Aufrufe und Oberflächen an.

Ablauf einer Report-Migration

Um Reports in der dargestellten Umgebung migrieren zu können, müssen zunächst die zentralen Komponenten bereitgestellt werden: Auftragsdatenbank, Komponentenrepository und Report-Server.

Im nächsten Schritt werden in jeder Technologiekomponente zentrale Funktionen geschaffen, die diese Komponenten bedienen. Befindet man sich am Anfang oder sogar wie in unserem Beispiel noch vor der eigentlichen Migrationsphase, so muss man in Forms eine pll bereitstellen, die für alle Forms-Module die Reports-Aufrufe kapselt. Zunächst einmal würde man hier nach wie vor die Oracle Reports aufrufen, sei es über den Report-Server oder nach wie vor mittels der vorhandenen Integration in Forms. Letzteres ist nur dann zu empfehlen, wenn keine Oracle Reports mehr von bereits migrierten Anwendungen verwendet werden sollen.

Die Migration eines einzelnen Reports ist dann denkbar einfach. Als erstes muss ein BIRT- oder JasperReport entwickelt werden, der die gewünschte bzw. die abzulösende Funktionalität abbildet. Dieser wird im Komponentenrepository registriert. Da der Reporting-Auftrag die Reporting-Funktion anfordert und nicht direkt den Report, reicht es aus, bei der Reporting-Funktion auf das neue Modul anstelle des Oracle Reports zu verweisen, um an allen Stellen der Anwendungssuite ab diesem Zeitpunkt den neuen Report zu verwenden. Wenn es sich dabei um eine 1:1-Migration handelt, wird der Anwender an dieser Stelle nicht einmal bemerken, dass sich etwas verändert hat.

Aspekte der Informationssicherheit

Wenn wir den Abruf eines Reports unabhängig von der Auftragserstellung als Web-Service zur Verfügung stellen wollen, müssen wir diesen auch schützen, um Daten nicht unbefugten Nutzern anzuzeigen. Es sind viele Varianten vorhanden, aus denen abhängig von den Anforderungen an die Anwendungen und die Architektur, der Art der zu verwaltenden Daten sowie den Sicherheitsvorgaben im Unternehmen die optimale Umsetzung ausgewählt werden muss.

Der einfachste Fall ist, dass alle Nutzer alle Daten sehen dürfen. Dann ist die Problematik lediglich, dass der eigentliche Auftraggeber schlimmstenfalls seinen Report nicht erhält. Weitere Vorkehrungen müssten nicht getroffen werden. Bekanntermaßen ist dies meistens nicht der Fall, insbesondere wenn die Anwendungen auch aus dem Internet heraus nutzbar sein sollen.

Basis jeglicher Security sollte zunächst sein, dass man auf Netzwerkebene den Report-Server abschottet. Wenn es die Umgebung zulässt, sind wesentliche Erfolge schon zu erzielen, wenn man Aufrufe nur aus der Datenbank oder nur von den Applikationsservern aus zulässt. Der erste Fall ist insgesamt der sicherste (eine gute Absicherung der DB setzen wir dabei voraus), da man in diesem Fall auch den Anwendungsbenutzer direkt kennt und dem Report-Server auf sicherem Wege mitteilen kann.

Die in der Datenbank abgelegten APIs schützt man grundlegend über die Mechanismen, die man auch für andere Datenbankobjekte wie Tabellen oder Packages nutzt.

Bei alten Forms-Anwendungen sind häufig Oracle-Datenbank-User im Einsatz. Will man diese, zumindest vorübergehend, weiterhin nutzen, wird man in seinen Web-Anwendungen wahrscheinlich Proxy User-Konzepte umsetzen. Diese sind dann auch im Report-Server anwendbar. Hier will man vom Anwender allerdings weder verlangen, sein Passwort für jeden Reportabruf einzugeben noch die direkte Übertragung seines Passwortes via https an den Report-Server abzubilden. Daher muss der Proxy User ohne Passwort funktionieren und stellt in diesem Moment ein Sicherheitsrisiko dar, welches auf anderem Wege abgefangen werden muss.

Hat man ein SSO-System im Einsatz, wird man den Report-Server dort auch einbinden und die Übergabe des Tokens oder einer Session-ID verlangen. Zusätzliche Sicherheit wird erreicht, wenn der Abruf eines Reports über eine sichere Auftrags-ID erfolgt, die sich nicht einfach aus einer Sequence bedient, sondern z.B. ähnliche Sicherheitsmerkmale wie eine Web-Session-ID aufweist.

Diese Methodik wird wichtiger, wenn das Umfeld größer wird, in dem der Report-Server aufrufbar ist. Wenn z.B. in einem Endkundenprozess ein Dokument zur Verfügung gestellt werden soll und die zugehörige Auftrags-ID per Mail versendet wird, muss hier besonders viel investiert werden. Zusätzlich muss man sich in einem solchen Verfahren Gedanken um die Sicherheit des Mailtransfers

machen. Bei einer derartigen asynchronen Bereitstellung von Reports ist ein Login an das System trotz aller anderer Maßnahmen zu empfehlen.
Diesen Sicherheitsthematiken kann man aus dem Weg gehen, indem man den Report-Server stets selbst nach neuen Aufträgen suchen lässt. Allerdings verschiebt man das Problem damit in vielen Fällen nur, da das erzeugte Dokument bzw. der Abruf desselben entsprechend geschützt werden muss.
Weiterhin muss man sich auch Gedanken über die Aufbewahrungszeit der in der Datenbank abgelegten Dokumente und Logging-Informationen machen. Bekanntlich kann es hier abhängig vom Geschäftsumfeld viele zu beachtende Reglementierungen geben.
Alles in allem hängt die Implementierung der Security wie so oft sehr stark vom konkreten Szenario ab.

Ausblick
Das dargestellte grundlegende Prinzip ist nicht nur offen hinsichtlich der Technologien der Anwendungs- und Reportentwicklung, sondern kann auch für andere Funktionalitäten von modernen Geschäftsanwendungen eingesetzt werden. Dafür kommen nahezu alle Funktionen in Frage, die in irgendeiner Form vom Backend durchgeführt werden können. Dies sind im einfachsten Fall Exporte, z.B. im csv oder XML-Format, oft aber auch die Erzeugung von Office- oder anderen Dokumenten. Automatisches Erstellen von (Massen-)eMails ist ebenso möglich wie die Bedienung von Schnittstellen. In Kombination kann ich über diese Funktionalitäten dann erzeugte Dokumente direkt verschicken.
Wenn das Verfahren zusätzlich an den Scheduler angeschlossen wird, kann das System zu einer überschaubaren Mini-SOA ausgebaut werden, mit der ein ebenso überschaubares Mini-BI-System angeboten werden kann. Überschaubar und Mini deutet an dieser Stelle darauf hin, dass dies auch für das einzusetzende Budget gilt – zumindest im Vergleich zu den klassischen größeren Ansätzen in diesen Disziplinen.
Weitere Ausbaumöglichkeiten liegen im DevOps-Bereich, z.B. indem durch das System unterstützt zunächst ein Testbetrieb einer Reports-Version für einen beschränkten Anwendungsfall vorgeschaltet wird, bevor man den Report tatsächlich produktiv, möglicherweise auch zeitgesteuert, austauscht.
Und da aktuell dieses Wort im Zusammenhang mit Oracle nicht fehlen darf: mit einer solchen Architektur ist man auch gut aufgestellt, um das Reporting in die Cloud zu verlagern.

Kontaktadresse:
Karsten Meyer-Kühl
IT-P GmbH
Seligmannallee 6
D-30173 Hannover

Telefon: +49 (0) 511-616804 31
Fax: +49 (0) 511-616804 17
E-Mail K.Meyer-Kühl@it-p.de
Internet: www.it-p.de