Modernisierung von Forms-Anwendungen ist bei vielen Forms-Kunden ein Damokles-Schwert, das respekteinflößend über ihnen schwebt. Oracle beteuert, Forms sei weiterhin voll unterstützt und es gäbe keine Pläne, es aus dem Support zu nehmen. Aber viele neue Funktionalitäten gab es im letzten Release auch nicht. Dies lässt den Schluss zu, dass es jetzt doch langsam ernst wird mit der Frage „Was tun mit meiner alten Forms-Applikation?" Aber wie vorgehen? Erst mal upgraden und dann nach Java, ADF oder Apex umstellen? Upgrade, Migration oder Konversion? Automatisch oder manuell? Selbst umstellen oder outsourcen? Wenn automatisiert wird, dann mit welchem Produkt?

Sinn und Unsinn der automatisierten Oracle-Forms-Modernisierung

Peter Sechser, Abocraft

Als der Autor 1991 bei Oracle anheuerte, war Forms mit der Version 2.3 auf dem Markt und die Version 3 stand kurz bevor. Die Erfassung und Bearbeitung von Problemfällen, die beim deutschen Oracle-Support, der guten alten Zentralen Kundenunterstützung (ZKU), gemeldet wurden, wurde gerade auf das Problem-Management-System (PMS) umgestellt. Ein System, das ganz fortschrittlich auf Forms 3 im klassischen Character-Mode entwickelt wurde und lange Zeit noch Verwendung fand. Alter Tobak? – weit gefehlt. Nach 16 Jahren bei Oracle, die er in unterschiedlichen Funktionen durchlaufen hat und viele Technologie-Generationen später sieht sich der Autor auch heute noch mit Anfragen von Oracle-Kunden konfrontiert, die Forms-Anwendungen im laufenden Betrieb nutzen, die zu einem ähnlichen Zeitpunkt wie das PMS entwickelt wurden und immer noch brav und zuverlässig ihren Dienst tun. Es trauen aber mittlerweile immer weniger dem Frieden und machen sich intensiv Gedanken, wie mit solchen Anwendungen umzugehen sei. Es besteht jedoch eine große Unsicherheit darüber, welche Alternative und welcher Weg dahin am besten seien.

Upgrade

Oracles Forms-Strategie von „Upgrade & Integrate" verhieß erst mal, ein Versions-Upgrade durchzuführen. Als Ziel wurde die letzte aktuelle Version genannt. Viele sind auch bereits durch die Mühen eines Upgrade-Projekts gegangen, haben sehr unterschiedliche Ergebnisse erzielt und mannigfaltige Erfahrungen gemacht. Die dabei entstandene Mühe konnte als Funktion mit mehreren Parametern empirisch bestimmt werden:

1. Normalitätsgrad der Funktionalitätsnutzung
2. Größe der Anwendung
3. Komplexität der Anwendung
4. Know-how-Level der Mitarbeiter
5. Dokumentation der Anwendung

Zunächst also ein Upgrade, also eine Versionsaktualisierung durchzuführen, hat sich allgemein als bester erster Schritt herumgesprochen. Aber besser manuell oder doch mit einem Werkzeug zur Automatisierung desselben Vorgangs? Dazu muss man sich die Parameter etwas genauer anschauen.

Upgrade – manuell oder automatisiert

Forms war lange Zeit ein unangefochtenes Entwicklungswerkzeug. Der Autor findet, dass es dies auch noch immer wäre, wenn es nicht nur 3-Tier sondern auch Client/Server im klassischen Sinne unterstützen würde. Der Grund liegt sowohl in der enormen Flexibilität, was den Funktionsumfang betrifft, als auch in der extremen Breite seines Anwendungsbereichs. Dass dies jedoch gleichzeitig zum Fluch werden könnte, dachte bei der Anwendungsentwicklung noch niemand.

Mit dem Normalitätsgrad der Funktionalitätsnutzung lässt sich der Umfang beschreiben, in dem außergewöhnliche Forms-Features verwendet wurden. Dazu könnte man beispielsweise die Nutzung von Oracle Graphics ebenso zählen wie die Desktop-Integration oder User Exits. Letzeres stellt dabei noch das eventuell kleinere Problem dar, da die Art und Weise, wie diese behandelt wurden, noch relativ stabil geblieben ist. Beschränkt sich der genutzte Funktionsumfang jedoch auf die nicht genannten Bereiche, liegt der Gedanke nahe, die Anwendung schnell mal manuell durchzumigrieren, also ein „Quick & Dirty"-Upgrade durchzuführen.

Hier kommt der zweite Parameter ins Spiel: die Größe der Anwendung. Eine Anwendung mit tausend Forms-Modulen und vielleicht nochmal so vielen Reports-Modulen wird nicht mal schnell aktualisiert. Und schon gar nicht, wenn die Anwendungen mit komplexem PL/SQL-Code bestückt sind, der wiederum intensiven Gebrauch von Built-ins macht. Wenn dann noch eine komplexe Abhängigkeit der Module untereinander besteht, ein Upgrade dann auch noch Auswirkungen nicht nur auf die Logik, sondern auch auf die Optik hat, wird die Anwendung als solche so komplex, dass sie manuell nur unter sehr hohem Aufwand zu aktualisieren ist.

Das bedeutet natürlich nicht, dass es unmöglich ist. Häufig besteht der Wunsch seitens der Entwickler, das eigene Team der Forms-Entwickler, soweit vorhanden und soweit es auch in Zukunft die Anwendung pflegen soll – in welcher Technologie auch immer –, für das Upgrade einzusetzen (siehe Parameter vier). Gegebenenfalls ist es auch sinnvoll, die noch fehlenden

Projekt-Ressourcen durch externe aufzustocken.

Ein teilweises oder eventuell sogar komplettes Outsourcing ergibt vor allem dann Sinn, wenn das Upgrade-Projekt für sich bereits einen hohen Teil an Neuprogrammierung notwendig macht. Das kann der Fall sein, wenn eben der Normalitätsgrad (siehe Parameter eins) sehr niedrig ist und erwähnte außergewöhnliche Funktionen in der Breite genutzt wurden, somit also der manuelle Anteil des Upgrades ohnehin sehr hoch ist.

Aber da kommt mit aller Vehemenz Parameter fünf: die gute Dokumentation. Welcher Entwickler erinnert sich nicht an die zehn Regeln, von denen eine aussagt, dass „es schon hart genug war, das Programm zu schreiben", sodass keine Zeit mehr bleibt, um im Rahmen von straffen Projektplänen auch noch gute und detaillierte Dokumentationen zu verfassen.

Wahr, aber eben auch fatal. Denn jetzt, wenn Forms-Entwickler Mangelware sind, Upgrade-Projekte extern vergeben werden und der Budgetdruck noch größer ist als zur Zeit der Initial-Entwicklung, ist die Dokumentation im wahrsten Sinne des Wortes Gold wert und würde helfen, Geld und vor allem Zeit zu sparen.

Bei manuellen Upgrade-Projekten wird meistens jede einzelne Form und jeder einzelne Report (neben der eventuell zusätzlich notwendig gewordenen Neuprogrammierung von Anwendungsbestandteilen) hochgezogen oder zumindest im White-Box-Verfahren auf seine weitere Funktionstüchtigkeit geprüft und optisch verifiziert. Ein Verfahren, bei dem kalkulatorisch gerade mal zwei bis drei Forms-Module pro Tag herauskommen, wenn nicht sogar weniger, und mal von dem Programmierungsanteil ganz zu schweigen. Übersetzt heißt das: Bei einer tausende Module umfassenden Forms-Anwendung kommen grob gerechnet zwischen 250 und 500 Manntagen nur für das Upgrade selbst zusammen. Was dies für die Frage bedeutet, welche Ressourcen im welchem Umfang, intern oder extern, unter Berücksichtigung des laufenden Geschäfts benötigt werden, kann jeder für sich selbst besser beantworten.

Bleibt die Frage des Budgets und auch der „elapsed time" – der Projektlaufzeit.

Was lässt sich so weit festhalten? Da wäre der Normalitätsgrad der Funktionalitätsnutzung. Er bestimmt letztlich, wie hoch der Anteil der Neuprogrammierung sein wird. Dieser Anteil, ob manuell oder bei automatisierter Vorgehensweise, lässt sich auf die gleiche Weise bestimmen, wie die Projektlaufzeit bei anderen Entwicklungsprojekten. Im direkten Vergleich zwischen „manuell" und „automatisch" spielt dieser Anteil wohl eine untergeordnete Rolle. In der Gesamtbetrachtung des Upgrade-Projekts nimmt er jedoch häufig den größten Teil ein.

Bleibt also im Vergleich derjenige Anteil in Betracht zu ziehen, der sich auf „Standard"-Forms-Funktionen bezieht. Natürlich ist für eine Zehn-Module-Anwendung die anteilige Rüstzeit für die Nutzung von Automatisierungstools unangebracht hoch, sodass man wohl für kleinere Anwendung im Generellen sagen kann, dass sich der manuelle Weg meistens als der schnellere entpuppt.

Jedoch bereits bei etwas größeren Anwendungen, also schon ab fünfzig Modulen, kann sich der Einsatz eines Tools erstens in Bezug auf die Projektlaufzeit und zweitens in Bezug auf das Budget lohnen. Vor allem dann, wenn die Anwendung zwar klein, aber extrem komplex ist oder wenn keiner da ist, der sich da noch auskennt, oder/und vor allem dann, wenn dann auch noch keinerlei Dokumentation vorhanden ist.

Selbst wenn der zeitliche Anteil der nach dem automatischen Durchlauf notwendigen Nachbearbeitungszeit relativ hoch angesetzt wird und in diesem Falle noch Lizenz- und möglichweise Wartungskosten für das Automatisierungstool anfallen, kann sich das Projekt im Vergleich zum rein manuellen Ansatz durchaus lohnen.

Natürlich hängt es vom genutzten Tool ab, wie effizient hier gearbeitet werden und wie schnell es erlernt werden kann. Welche dabei in Frage kommen, wird im Folgenden behandelt werden. Aber es sei bereits hier angemerkt, dass es solche Tools, limitiert auf die Anwendung als automatisiertes Upgrade-Tool, bereits im vier- bis fünfstelligen Euro-Bereich auf dem Markt gibt. Selbst 20.000 Euro sind mit manuellen Mitteln schnell ausgegeben, wenn man dabei nicht nur an die Entwicklung, sondern auch an das Projekt-Management und -Controlling sowie – wiederum nicht zu vergessen – an die Dokumentation denkt.

Upgrade automatisiert – aber womit?

Es spricht also einiges für ein Upgrade mithilfe eines Tools. Aber welches Produkt eignet sich? Der Markt ist in diesem Segment sehr übersichtlich. Die betrachteten Produkte sind in erster Linie auch diejenigen, die ohne große Programmier- oder Skripting-Kenntnisse genutzt werden können. Die hier aufgeführte Liste erhebt keineswegs Anspruch auf Vollständigkeit, sondern ist aus dem Erfahrungsbereich des Autors entstanden (in alphabetischer Reihenfolge):

- Jforms von Imining
- ORMIT von IMEX
- PITSS.CON von PITSS

Natürlich gibt es da noch weitere Tools wie den Forms API Master. Aber zum einen hat der Autor keinen Kontakt mit einem Kunden, der diesen einsetzte, noch könnte man dieses Tool als interaktiv bezeichnen. Skripting ist daher unerlässlich. Der Vorteil dieses Ansatzes besteht in der Flexibilität der Handhabung, da man sehr viel über das Skripting erreichen kann. Der Nachteil, dass man eben vieles selber machen muss. Daher sei dieses Tool hier nur kurz erwähnt, aber nicht vertieft.

CipherSoft/Unify nimmt für sich in Anspruch, ein weiterer Anbieter von Forms-Upgrade-Produkten zu sein. Das Tool, das man auf der Webseite findet, heißt Exodus und hilft bei der Konvertierung von Forms nach Java, nicht jedoch bei der Versions-Aktualisierung. CipherSoft/Unify beschränkt sich also auf den manuellen Upgrade-Ansatz, ein Produkt hierfür hat das Unternehmen nicht. Wir werden später nochmal darauf zu sprechen kommen, wenn wir

die „Forms nach Java"-Konvertierung betrachten. Welche Charakteristika sind wichtig bei der Betrachtung von Automatisierungswerkzeugen? Einfach ausgedrückt alles, was manuell lange dauert.

Die automatisierte Analyse

Die Analyse ist der Schlüssel zur erfolgreichen Umsetzung. Als Ergebnis der Analyse sollten Fakten und/oder Daten stehen, die eine akkurate Einschätzung der notwendigen Zeit direkt für das Upgrade erlauben. Alternativ dazu wird eine Liste aller Aufgaben und Aktionen ausgegeben, auf Basis derer die Zeiteinschätzung möglich ist. Das Optimum wäre natürlich beides zusammen, aber keines der erwähnten Migrationstools bietet diesen Komfort. Es gibt wiederum Tools, die sich nur auf das Reporting selbst beschränken, ohne den Upgrade-Prozess selbst automatisiert zu unterstützen. Die Ergebnisse lassen sich auch durchaus sehen, sind aber teuer erkauft, da solche Tools in der Regel sehr hohe Lizenzkosten verursachen und die Kosten schnell an das Budgetlimit treiben.

Von Bedeutung bei der Analyse ist die Frage, was man beim Upgrade in Betracht ziehen und wie man seine Anwendung anpassen muss, damit diese auf der neuen Version laufen kann. Die Frage selbst ist schnell gestellt, aber sehr schwer zu beantworten. Die Mannigfaltigkeit der Aspekte ist enorm, beinhaltet sie doch das gesamte Wissen über die Herausforderungen und die jeweiligen Lösungen für das Upgrade. Und zwar nicht nur von 9*i* auf 10*g*, sondern eben von einer möglichst frühen Version, zum Beispiel 2.3 oder 3, bis hin zur letzten Version von Forms. Optimal wäre es, wenn diese vorgegebenen Analyse-Möglichkeiten noch mit eigenen Anfragen kombiniert werden könnten.

Ein Tool, das ein Upgrade automatisieren soll, kann nur dann ernst genommen werden, wenn es möglichst viel Kenntnis über alle real notwendigen Änderungen besitzt und dann auch noch weiß, auf welche Weise diese umzusetzen sind. Also beispielsweise, wie ein Report aus Forms heraus aufgerufen wird und wie sich dieses Aufrufverhalten von einer Version zur anderen verändert hat. Oder wie sich die Parameterliste eines Built-ins in der Anzahl und Position der einzelnen Parameter verändert hat, an welcher Stelle ein bestimmter Parameter in der alten Version war und an welcher Stelle er sich in der neuen befindet.

Ein entscheidender Faktor besteht in der Nutzung der Analyse-Resultate. Bekommt man überhaupt ein Resultat angezeigt? Und falls ja, kann man damit eine Aktionsliste generieren und den Aufwand gesamtheitlich unter Einbeziehung einer manuellen Nacharbeit – ausgenommen der oben beschriebenen Neuprogrammierung – schätzen oder möglicherweise sogar gleich interaktiv mit den Einzelfakten weiterarbeiten?

Nach der Analyse-Phase kann beim automatisierten Ansatz je nach Qualität der Resultate relativ schnell in die Implementierungs-Phase übergegangen werden. Häufig überschneiden sich Analyse und Implementierung oder wiederholen sich in Subzyklen je nach Arbeitseinheit, die gerade bearbeitet wird, etwa auf Basis individueller Module oder logisch zu einem Prozess verbundener Modulverbände. Dieser Vorgang findet sich häufig bei Projektpiloten. Wünschenswert ist jedoch eine vollständige Analyse mit qualitativ hochwertigen Ergebnissen, die dann als Voraussetzung für die Durchführung eine korrekte Aufwandsschätzung ermöglichen.

Jforms von Imining

Jforms ist ein Werkzeug, das unverkennbar aus dem Projektgebrauch entstanden ist. Es ist – abgesehen von der Analyse und damit verknüpft den Upgrade-Anweisungen beziehungsweise den Regularien – ein Automat: vorne Applikation rein, hinten aktualisierte Anwendung raus. Die Analyse-Regeln adressieren nach Herstellerangaben circa 95 Prozent der gesamten Änderungen. Diese lassen sich auch anschauen und gegebenenfalls ändern oder erweitern. Sind die Regeln einmal festgelegt, kann man anhand der Analyse-Regeln die Anwendung durchsuchen oder den automatisierten Ablauf gleich anstoßen. Die eigentliche Durchführung des Upgrades ist dann ein vorgegebener Prozess und geschieht entweder im Kommandozeilen-Modus oder im Hintergrund als Batch-Operation. Eine Einwirkung auf individueller Ebene, beispielsweise das interaktive Editieren von Code-Snippets des veränderten oder des noch zu ändernden PL/SQL-Blocks, ist nicht möglich. Einen Vergleich in Vorher-Nachher-Manier kann man hingegen mit dem Original-Modul und dem aktualisierten Modul anstellen.

Bei Jforms sind maximal zwei Sprünge beim Versions-Upgrade nötig, von Version 3 oder 4.5 auf 6*i* und anschließend auf Version 10 oder 11. Für 6*i*-Anwendungen entfällt somit das Versions-Hopping komplett und die Recherche beziehungsweise Analyse ist mittels der voreingestellten Abfragen vollständig möglich. Allerdings ist das Upgrade auf Forms-Module, Menüs und PLLs limitiert. Das Versions-Hopping, das Upgrade von Version über Version, findet sich nicht nur bei Imining, sondern auch beim nächsten Kandidaten.

ORMIT von IMEX

Das Tool fällt gleich zurück, bieten doch die zwei anderen Produkte eine interaktive Analyse-Möglichkeit, die eine „Customization" durchaus erlaubt. IMEX hat nach eigenen Angaben Oracles Best Practices des Upgrades eingearbeitet. Einsehbar, überprüfbar oder gar veränderbar sind diese allerdings nicht. Somit muss man sich auf die Aussagen des Herstellers verlassen. Der „Friss oder Stirb"-Ansatz ist zwar einfacher, was die Bedienbarkeit betrifft, aber die Wahrscheinlichkeit einer umfangreicheren Nacharbeit oder Anpassung auf einer intermediären Ebene – beispielsweise bei 9*i* – ist groß.

Zudem muss man bei ORMIT mit einem ausgeprägten Versions-Hopping zurechtkommen. Wer noch eine Version-3-Anwendung hat, der muss zuerst einmal (wie auch immer) auf 6*i*, dann auf 9*i* und anschließend auf Version 10 oder 11. Aus einem Upgrade-Projekt werden dann plötzlich zwei

oder drei Unterprojekte, jedes für sich mit den entsprechenden Risiken und Herausforderungen. Eine vollständige Analyse des Aufwands ist da schwierig. Die Tendenz ist klar erkennbar: Zusätzlich zu den Lizenzkosten sollen noch Dienstleistungen abgegraben werden, die dann wiederum in Offshore-Manier nach Indien ausgelagert werden. Der geografische Hintergrund von IMEX wird hier am allerdeutlichsten.

Die Durchführung des Upgrades ist dann natürlich denkbar einfach. Klick, klick und durch. Eine Aussage über die Qualität des Upgrades ist im Allgemeinen nur schwer machbar.

PITSS.CON von PITSS

Dieses Tool hebt sich von der Grund-Architektur und Funktionsweise her vollständig von den anderen Produkten ab. Die zu analysierenden Forms sind in ein Repository zu laden. Dabei werden die Anwendungsbestandteile in ihre Einzelteile zerlegt und im Repository abgelegt. Die weitere Analyse und Recherche geschieht dann innerhalb des Repository. Daher spielt die Quellversion keine Rolle, denn die kommt nur während des Einlesevorgangs zum Tragen. Sobald dieser abgeschlossen ist, arbeitet man nur noch im PITSS.CON-eigenen Umfeld.

Die Analyse ist interaktiv oder im Batchmode durchführbar. Als Basis der Upgrade-Analyse stehen vorgefertigte Abfragen zur Verfügung. Laut Hersteller decken diese Abfragen alle potenziellen Änderungen von jeder Version zu jeder Version ab. Das Regelwerk ist einsehbar, modifizierbar und erweiterbar; die Erweiterungsmöglichkeiten sind unbegrenzt.

Die Qualität der Ergebnisse ist aufgrund ihres Detailgrads und ihrer Aussagekraft sehr hoch. So kann diese als Aktionsliste zur Aufwandsschätzung verwendet werden. Im weiteren Verlauf des Projekts kann dann auch gleich interaktiv mit der Aufgabenliste weitergearbeitet werden. Das heißt: Doppelklick auf das Ergebnis und direkt anschauen, anpassen oder verändern.

Auch die Modultypen sind nicht begrenzt. PITSS.CON behandelt Forms-Module wie Menüs und Reports auch als Bibliotheken. Auch was die Einflüsse auf die optischen, durch das Upgrade verursachten Veränderungen betrifft, lässt sich das Tool verwenden. Die potenzielle Nacharbeit kann dadurch noch um einiges vermindert werden. Sei es ein Zurücksetzen der Item-Hintergrundfarbe, die man in früheren Versionen nicht beachten musste, die aber in 10*g* sehr genau interpretiert oder anders dargestellt wird, oder die Identifikation von nunmehr sich überlappenden benachbarten Items.

Automatisierte Upgrades – Zusammenfassung

Alles in allem lohnt es sich sehr wohl, bei der Automatisierung von Versions-Upgrades mit einem Tool zu arbeiten. Aber aufgrund der unterschiedlichen Design-Ansätze kann eine Nacharbeit noch beträchtlichen Umfang einnehmen. Eine genauere Evaluierung je Einzelfall kann hingegen enorme Zeit- und vor allem Kosteneinsparungen bringen.

Integrate

Forms hat mit seinen letzten Releases einige Integrations-Features bekommen, die es einer Anwendung ermöglichen sollen, entweder von Forms heraus mit Java-Anwendungen zu interagieren oder umgekehrt auf von Java-Anwendungen getriggerte Ereignisse zu reagieren. Die Webservice-Integration ist da ein gutes Beispiel.

Dann jedoch von einer modernisierten Forms-Anwendung zu sprechen, ist wohl eher überzogen. Aber es reicht aus, um Zeit zu gewinnen. Mittelfristig muss man sich mit dem Gedanken anfreunden, dass Forms eben keine Java-Anwendung ist und auch keine werden wird. Auch keine werden wird? – Diese Aussage scheint dem Angebot des Marktes zu widersprechen. Es gibt durchaus Anbieter, die wiederum automatisiert eine Forms-Anwendung nach Java migrieren. Aber schon in dem Wort „migrieren“ liegt vielfache Interpretationsmöglichkeit. Bedeutet „migrieren“ nun eine Konvertierung oder neben der sprachlichen Übersetzung auch eine architektonische Umsetzung? Wieder lohnt sich ein etwas tiefergehender Blick auf die unterschiedlichen Produkte und Angebote.

Automatisiert von Forms zu Java

Von den vorher, für das Versions-Upgrade betrachteten Anbietern bleibt nicht viel übrig, aber es kommen dafür neue hinzu. Da das Unternehmen PITSS mit seinem Tool auch das Versions-Upgrade durchführt, sei es hier als Erstes genannt:

- PITSS.CON von PITSS
- Exodus von CipherSoft/Unify
- JFormsBuilder, JReportsBuilder von ASP
- JHeadstart von Oracle

Bei der Betrachtung muss man andere Charakteristika ansetzen als beim reinen Versions-Upgrade. Diese sind:

1. Architektur
2. Projektprozess-Schritte
3. Analysefähigkeiten
4. Anwendungsarchitektur
5. Entwickler-Skills

PITSS.CON von PITSS

Das Produkt arbeitet wie bereits erwähnt mit einem Repository, in das die Forms-Anwendung eingelesen und dann verarbeitet wird. Auch bei der Migration zu Java differenziert sich PITSS.CON von seinen Mitbewerbern. Das Tool konvertiert nicht einfach nur den Forms-Code zu Java. Vielmehr wählt PITSS einen modularen Ansatz, der eine unterschiedliche und sogar nahezu unabhängige Bearbeitung und Betrachtung der Businesslogik sowie der Darstellungsebene nach sich zieht. Er kommt da dem 3-Tier-Ansatz mit einer Anwendungslogik und einer Darstellungslogik gleich.

Durch diese architektonische Differenzierung ist auch eine andere Projekt-Vorgehensweise notwendig. Es ist nicht möglich, eine Forms-Anwendung einfach auf der einen Seite hereinzuschieben und auf der anderen

Seite eine Java-Anwendung herauszubekommen. Vielmehr ist ein Mehrphasen-Ansatz nötig. Dieser schreibt vor, sich zunächst der Businesslogik anzunehmen. Da diese in der Forms-Anwendung oder auch in der Datenbank als PL/SQL-Code vorliegt, wird sie nicht nach Java übersetzt. PITSS macht hier den Ansatz, den Code in der Sprache zu belassen, in der er erstellt und für die er optimiert wurde. Gleichzeitig muss der Forms-orientierte Code einem Service-orientierten Java-Umfeld angepasst werden. Da dieser Forms-Code eher Block-orientiert ist, während der Service-Gedanke vom Prozess beeinflusst wird, ist eine Konvertierung des Forms-Codes zu Prozessen und damit Diensten notwendig.

Die oben bereits erwähnten, starken Analyse-Möglichkeiten sind hier von enormer Wichtigkeit und helfen, diesen Prozess abzukürzen. Für den eigentlichen Vorgang hat PITSS einen SOA-Konverter entwickelt, mit dem eine Konsolidierung der Businesslogik aus den verschiedenen Blöcken in einen SOA-Prozess möglich ist. Dies ist aber definitiv nur ein semi-automatischer Vorgang, da er Entscheidungen während der Konsolidierung verlangt. Die Durchführung anhand der gefällten Entscheidungen ist dann wieder vollautomatisch möglich.

Wenn die Phase der Logik-Konvertierung abgeschlossen ist, folgt in der zweiten Phase die Umstellung der Darstellungsebene. Natürlich kann dies auch ein überlappender Prozess sein. Die Konvertierung einer Forms-Anwendung ist bei PITSS also genauer genommen eine Konvertierung der Darstellungsebene, während die Logik in PL/SQL bleibt.

Der ADF-Konverter von PITSS.CON setzt die visuellen Elemente nach ADF um. Das ist zum einen deutlich einfacher und zum anderen auch schneller. Zudem erübrigt sich die Frage, ob die Businesslogik der ADF-Architektur entspricht, wie sie von Oracle empfohlen wird, da sie sich dieser Betrachtung über das Verbleiben des PL/SQL-Codes in der Datenbank entzieht.

Verfechter von PL/SQL freuen sich darüber, hilft es doch dabei, die vorhandenen Entwickler-Ressourcen auch weiterhin einsetzen zu können, da die umgesetzte Anwendung eben nicht ausschließlich auf ADF aufbaut. Somit werden diejenigen, die die neue Anwendung zu pflegen haben, sowohl PL/SQL- als auch ADF-Skills aufweisen müssen. In der Regel sind diese nicht in Personalunion vorhanden. Vielmehr sind es meist sogar unterschiedliche Teams.

Exodus von CipherSoft/Unify

Exodus wurde seinerzeit von CipherSoft entwickelt und mit der Akquisition von Unify übernommen. Es war immer ausschließlich ein Werkzeug zur Konvertierung einer Forms-Anwendung nach Java. Allerdings geht CipherSoft/Unify einen anderen Weg als PITSS beziehungsweise Oracle. Exodus hat kein Repository. Es arbeitet direkt mit den Quellen, benötigt sie jedoch nur zum Lesen; sie werden also in dem Prozess der Konvertierung nicht verändert. Zudem hat CipherSoft die Forms Runtime in Java nachprogrammiert. Diese Bibliotheken werden dann während der Laufzeit auch auf dem Midtier Application Server benötigt. Die Runtime Library beinhaltet das gesamte Triggerhandling ebenso wie Built-ins – also alles, was Forms an Grundfunktionalität zu bieten hat.

Bevor Exodus jedoch überhaupt als solches zum Einsatz kommt, ist natürlich wieder eine Analyse von Bedeutung. Hierfür hat CipherSoft einen Analyzer entwickelt, der jedoch nicht als Produkt verfügbar ist, sondern den durchführenden Partnern als Projekt-Bestandteil bereitsteht. Dieser ist darauf ausgerichtet, Unverträglichkeiten zwischen den Technologiewelten zu identifizieren, zu quantifizieren und damit eine Projekt-Laufzeit- und -Kosten-Abschätzung zu ermöglichen. Unverträglichkeiten sind in diesem Falle beispielsweise GOTOs oder weitere Forms-Built-ins, die aufgrund ihrer Natur nicht direkt in die Java-Welt umzusetzen sind.

Ansonsten ist Exodus ein Tool, das „straight forward“ nach dem Prinzip arbeitet, vorne Forms-Anwendung rein, hinten Java-Anwendung raus. Dies kann die gesamte Anwendung einschließlich des PL/SQL-Codes sein, der bis dahin in der Datenbank als „stored procedures“ gespeichert war. In der Regel wird aber dieser Code auch weiterhin, ähnlich wie beim PITSS-Ansatz, in der Datenbank als PL/SQL-Code verbleiben. Aber im Gegensatz zu PITSS wird nichts konsolidiert und zu Prozessen zusammengefasst. Also wird der PL/SQL-Code entweder Zeile für Zeile nach Java konvertiert oder er verbleibt Zeile für Zeile als PL/SQL-Code in der Datenbank. Dem Autor sind keine Fälle bekannt, in denen der PL/SQL-Code der Datenbank nach Java konvertiert wurde.

Bleibt also der „Client“-Code von Forms einschließlich der Darstellung. Exodus unterscheidet zwar während der Konvertierung verschiedene Phasen wie Bibliotheken, Forms-Module etc. Diese Phasen beziehen sich jedoch ausschließlich auf die Vorgehensweise und haben keine Auswirkungen auf das Ergebnis. Schaut man sich den generierten Code an, so sieht man eine Forms-Anwendung im Java-Mantel. Die Nomenklatur orientiert sich vollständig nach Forms. Das hat Vorteile sowie Nachteile.

Besteht das Entwickler-Team, das die Anwendung nach der Konvertierung weiter pflegen soll, aus Forms-Entwicklern, werden diese sich leicht in der neuen Java-Anwendung wiederfinden. Triggernamen und Bezeichnungen sind genau so, wie man sie aus der Forms-Welt gut kennt. Besteht jedoch das Team für die neue Anwendung aus Java-Entwicklern, sieht man in den Augen nichts als große Fragezeichen. Zudem handelt es sich bei der generierten Java-Anwendung architektonisch immer noch um eine Forms-Anwendung. Eine Anwendung, die ein Java-Entwickler so wahrscheinlich nie geschrieben hätte.

Das große „Aber“ ist hier jedoch in der ursprünglichen Zielsetzung zu sehen. „Forms zu Java“-Konverter werden in Augenschein genommen, um die Forms-Anwendung nach Java zu übersetzen. Wenn es also Ziel ist, die Anwendung, so wie sie momentan in Forms läuft, auch unter Java bereitzustellen, ist die Arbeitsweise dieses Tools

Die Fachzeitschrift für Datenprofis!

Für Entwickler Administratoren Softwarearchitekten

Bestellen Sie Ihr kostenloses Kennenlern-Exemplar unter:

www.databasepro.de/probelesen

nicht zu beanstanden. „It works as designed“.

JFormsBuilder, JReportsBuilder von ASP

Ein neues Tool in diesem Segment ist von der südamerikanischen Firma ASP Solutions. De facto besteht das Tool aus zwei Produkten mit unterschiedlicher Zielsetzung: Forms und Reports. Auch ASP hat seine Laufzeitbibliothek für die Forms-Grundfunktionen in Java nachprogrammiert und vergleicht sich somit mit CipherSofts Exodus. Während CipherSoft jedoch sein Exodus sowohl als Standalone-Version als auch als JDeveloper-Plug-in anbietet, konzentriert sich ASP auf Eclipse als Entwicklungswerkzeug.

JFormsBuilder und JReportsBuilder haben eine Forms sehr ähnliche Entwicklungsoberfläche – auch oder gerade optisch. Eine Analyse ähnlich wie PITSS oder dem CipherSoft Analyzer bietet dieses Tool jedoch noch nicht, zeigt aber Fehler während der „Kompilierung“ auf, die man dann sukzessive abarbeiten kann. Im Gegensatz zu PITSS und CipherSoft ist jedoch ASP in der Lage, auch Reports umzusetzen. Eine Funktionalität, die ansonsten keiner der anderen Bewerber um diesen Markt herum bietet. Da ASP noch sehr neu im deutschen Markt ist, kann man auf Projekt-Umsetzungen gespannt sein, vor allem, was Reports betrifft.

JHeadstart von Oracle

Von Headstart hat man während der Oracle-Designer-Zeiten ja schon einiges gehört. Oracle Consulting in den Niederlanden hatte dieses Tool seinerzeit als Projekt entwickelt und der breiten Öffentlichkeit verfügbar gemacht. Nach Oracles Shift von Oracle Designer zu JDeveloper war es nur konsequent, dass Oracle Consulting in den Niederlanden auch mit JHeadstart eine Erweiterung entwickelte, die für die Konvertierung von Forms nach Java nützlich sein kann.

Die Zielsetzung von JHeadstart ist, die Oberfläche einer Forms-Anwendung nach Oracle ADF zu konvertieren. Auf dieser Ebene ist JHeadstart am ehesten mit PITSS.CON zu vergleichen, das ja mit dem ADF-Konverter die Darstellungsebene der Anwendung adressiert. Die Konvertierung auf dieser Ebene lässt auch keinerlei Wünsche offen. Jedoch gibt es zur Konvertierung der Businesslogik die Aussage, die – konform mit dem ADF-Produktmanagement – lautet, es wäre am besten, diese mit ADF nachzuprogrammieren. Das hat natürliche massive Auswirkungen auf die Projektlaufzeit und die damit verbundenen Kosten. Aber auf der anderen Seite ist die Frage berechtigt, welche Aussage man in diesem Kontext von Oracle erwartet hätte.

Konvertierung nach Apex

Apex hat in der Forms-Gemeinde eine große Anhängerschaft und bisweilen sogar vehemente Verfechter gefunden. Viele sehen in Apex das Forms der Neuzeit. Eine generelle Aussage in diese Richtung zu treffen ist – wie so oft in der IT – zu schwarz-weiß gedacht. Wie so oft liegt der Teufel im Detail, da für Anwendungen eines bestimmten Komplexitätsgrades Apex eine sehr gute Alternative sein kann, dagegen für andere überhaupt nicht zur Debatte steht. Die Frage, wann und unter welchen Voraussetzungen Apex eine gebotene Alternative für Forms-Anwendungen sein kann, entweder als Migrationsziel oder als Ersatz für Neuentwicklungen, sei hier nicht näher erörtert. Muss es auch nicht, denn hier steht die Frage im Vordergrund, inwieweit sich eine Forms-Migration mit automatischen Mitteln durchführen lässt.

Auch bei Forms-zu-Apex-Migrationen muss man wieder zwei Ebenen betrachten: Businesslogik und Darstellungsebene. Aus der Historie heraus lässt sich zunächst feststellen, dass Apex ja eigentlich nichts anderes war, als ein HTML-basiertes Interface zur Datenpflege – Forms in der simpelsten Form. Damals hieß Apex noch HTML-DB. PL/SQL war die Programmiersprache, Code-Blöcke blieben in der Datenbank als „stored procedures“.

Mittlerweile ist Apex erwachsen geworden und JavaScripts sind kein Mythos mehr. Aber wenn es darum geht, eine komplexe Forms-Anwendung nach Apex zu migrieren, nutzen solche Features relativ wenig, wenn man Automatismen nutzen und nicht in ähnliche Probleme laufen will wie bei der Konvertierung nach Java. Apex bietet zwar einen Forms-zu-Apex-Migrationspfad an, der jedoch eher einem Projektmanagement-Werkzeug ähnelt als einem automatischen Konvertierungstool. Apex kann die Metadaten von Forms-Modulen über den XML-Konverter einlesen. Auch für Reports gibt es diese Möglichkeit. Ebenso lassen sich Basisobjekte relativ einfach in eine Apex-Applikation einbringen.

Betrachtet man jedoch die Businesslogik, so steht man wieder vor der gleichen Herausforderung. Ein Extrakt aus der Apex-Dokumentation zeigt dies sehr deutlich auf: „In order to re-implement the business logic defined within Oracle Forms you must analyze the triggers, program units, and PL/SQL libraries and manually write corresponding logic within Oracle Application Express.“ (Quelle: Oracle Application Express Application Migration Guide). Somit ist eine „Forms zu Apex“-Konvertierung vergleichbar mit einer „Forms zu Java“-Konvertierung mit Oracle JHeadstart: Die Darstellungsebene ist relativ schnell adressiert, aber die komplexe Businesslogikebene wird ausgespart.

Ein kleiner Blick auf den Oracle-externen Markt zeigt, dass sich auch die von obiger Diskussion bereits bekannten Anbieter mit der Thematik befasst haben. CipherSoft/Unify hat sich da sehr schnell mit der Ankündigung auf den Markt begeben, eine automatische Migration von Forms zu Apex zu bieten. Aber ein genauerer Blick lässt das Produkt und die Substanz vermissen. CipherSoft/Unify bietet in diesem Segment, ähnlich wie beim Forms-Versions-Upgrade, Dienstleistungen an und reiht sich also auch hier in die lange Schlange der System-Integratoren ein.

PITSS hat vor kurzem ein Modul zur Erweiterung seines PITSS.CON auf den Markt gebracht, das die Apex-Konvertierung unterstützen soll. Der architektonische Ansatz ist wohl ähnlich dem der Java/ADF-Konvertierung: Businesslogik wird über den bereits oben beschriebenen Vorgang als Prozess kon-

solidiert, als PL/SQL-Stored-Procedures in der Datenbank belassen beziehungsweise mit dem Tool semiautomatisch übergeführt. Somit bleibt noch die Darstellungsebene, die der Apex-Konverter zu generieren verspricht. Da aber diese Produkterweiterung sehr neu ist, liegen dem Autor noch keinerlei praktische Erfahrungsberichte vor. Daher ist eine Einschätzung noch nicht möglich.

Also doch nach .Net?

Und da wäre noch Microsoft mit .Net. Neben der manuellen Migration nach .Net gibt es vereinzelte Produkte, die eine automatische Überführung versprechen. Der Autor ist als ehemaliger langjähriger Oracle-Mitarbeiter ein Verfechter der Oracle-Technologie, zudem er Stunde um Stunde und auch Abend für Abend an diversen Microsoft-Produkten trotz Fachkenntnissen verzweifelt ist. Diese Betrachtung sei daher anderen überlassen.

Alternative oder Zeit gewinnen: Paralleles Deployment

Für diejenigen, die sich zwischen den oben diskutierten Ansätzen nicht entscheiden wollen oder auch können, weil die Rahmen-Parameter es zu dem jetzigen Zeitpunkt nicht erlauben, verspricht nach der manuellen Modernisierung – ob neu schreiben oder manuell migrieren – dem Versions-Upgrade und der Konvertierung nach Java oder Apex ein vierter Ansatz zumindest etwas Zeit und kann damit Luft verschaffen: das parallele Deployment.

In England hat die Firma Quintessence Systems beschlossen, ihre Strategie zu ändern. Hatte sie bisher versucht, mit dem Produkt-/Dienstleistungsmix einer „Forms zu Java"-Migration mittels ihrer in2j-Technologie den Markt zu erobern, beschreitet sie nunmehr einen anderen Weg. Dieser verspricht, kurz- bis mittelfristig eine interessante Alternative zu den bisherigen Ansätzen zu werden. „Zu werden" deshalb, weil die Entwicklung noch nicht abgeschlossen ist.

Die Grundidee ist simpel: Man lasse seine Forms-Anwendung so weiterlaufen wie bisher und baue sich einen parallelen Einstiegsweg über J2EE auf. Dazu braucht es natürlich einen zusätzlichen J2EE-konformen Server, der wiederum die Forms-Runtime emuliert und die Schnittstelle zum Browser des Anwenders darstellt. Dies scheint der YoServer zu tun, wenn man den Aussagen auf der Webseite vertraut. Sogar der PL/SQL-Code verspricht, schneller durch YoServers Interpreter zu laufen. Somit ist eine schrittweise Migration von reinem Forms zu einer Java-Welt möglich. Nicht alles auf einmal, sondern Modul für Modul oder auch Prozess für Prozess. Soweit die Theorie. Der Praxistest muss hingegen noch bestanden werden.

Fazit

Für alle, die ihre Anwendung in eine SOA-Welt überführen wollen, ist der Weg schon nicht mehr so eindeutig. Ist beim Versions-Upgrade die Technik noch relativ überschaubar und ähnelt sich auf die eine oder andere Weise, gibt es bei der Java-Konvertierung große Unterschiede. Hier ist es also umso wichtiger, eine detaillierte Evaluierung durchzuführen. Für diejenigen, die einfach die Anwendung nach Java konvertieren wollen und in Java weiterlaufen lassen wollen, stehen Konverter sowohl für Forms als auch für Reports zur Verfügung.

Kontakt:
Peter Sechser
psechser@abocraft.com

Peter Sechser war seit 1991 bei Oracle und hat dort viele verschiedene Funktionen ausgeübt: beginnend bei der Zentralen Kundenunterstützung (ZKU) über die Leitung des Productline „Marketing" bis hin zum technischen Partner Account Management.