Next Generation SAP

SAP befindet sich im Umbruch, seit einigen Jahren werden in Walldorf grundlegende Weichenstellungen umgesetzt, die den Abschied von grundlegenden Paradigmen bedeuten. Weg von proprietären Lösungen, hin zu offenen Standards – Was dies bedeutet, erschließt sich erst, wenn man diese grundlegende Umwälzung betrachtet.

Der Abschied von der SAP GUI

Mit Business Server Pages und WebDynpro ABAP wurden die ersten Schritte getan, die vom SAP GUI wegführten. Ehemals als besonders fortschrittliche Technik gelobt, steht das Konzept SAP GUI heute als Relikt einer früheren Softwaregeneration, denn Scharen von Anwendern greifen heute über neuere Technologien auf Daten zu.

Zunächst per Business Server Pages, in der Folge dann auch als WebDypro ABAP, mittlerweile in Form von großen, systemübergreifenden Portalen wird die Loslösung von der SAP GUI schon länger praktiziert, wobei keine der genannten Technologien eine Ablösung der anderen darstellt, vielmehr hat jede dieser Technologien ihren Platz im Spektrum möglicher Anwendungen.

Dies stellt den ABAP-Entwickler vor neue Herausforderungen. Auf der einen Seite muss er sich weniger Gedanken über die Ausgabedetails machen, weil die Ausgabe auf der Ebene des Portals erzeugt wird, auf der anderen Seite sind Programmlaufzeiten von mehreren Minuten (wie sie bei aufwendigen Reports durchaus anfallen können) für Portale inakzeptabel, da diese nach nur wenigen Sekunden einen Timeout auslösen. Performanceoptimierung hat nun eine sehr hohe Priorität, wo man früher noch ein Auge zugedrückt hat.

Eine weitere Baustelle sind die neuen Clients: Statt am Windows-PC sitzt eine Vielzahl von Anwendern an Clients, die am Markt völlig neu sind, Aufträge werden z. B. vom Vertrieb auf dem iPad erfasst, die Basis erfährt am SmartPhone von Störungen, Fehlern und Engpässen, etc. Der Markt der Post-PC-Devices wächst nicht, er explodiert geradezu, da sie klein, handlich und (im Vergleich zu klassischen Windows-PCs) praktisch wartungsfrei sind.

Das Ende von SAPscript und SmartForms

Ebenso proprietär wie die SAP GUI ist die Formularentwicklung im SAP bisher gewesen. SAPscript war aus Entwicklersicht eine ergonomische Katastrophe und Formularänderungen erzeugten regelmäßig hohe Aufwände, wofür es an Verständnis beim Anwender mangelte, der sich unter Formularentwicklung etwas Ähnliches vorstellte wie die Serienbrieferstellung in einem Textverarbeitungsprogramm. Gleichwohl war die Performance von SAPscript mit der eines Textverarbeitungsprogramms nicht vergleichbar. Das Beispiel „Drucken Sie mal bitte ein paar tausend Rechnungen mit Word, dann kennen Sie den besonderen Vorteil von SAPScript“ wird immer wieder angeführt, und das ist auch durchaus richtig, denn dies ist ein typischer Anwendungsfall von SAPscript: Nicht das Drucken eines Dokumentes auf einem lokalen Drucker, sondern das Bereitstellen von einer Vielzahl von Rechnungen, Mahnungen und ähnlicher Dokumente, an deren Ende eine Druckstraße steht, die fertig kuvertierte, frankierte und vorsortierte Post zum Versand bereitstellt.

SmartForms ist eine erhebliche Erleichterung in Entwicklung und Wartung von Formularen, wenngleich „unter der Haube“ natürlich immer noch SAPscript steckte. Die Entwicklungsumgebung ist jedoch ungleich moderner und nimmt dem Entwickler die explizite Programmierung des Formularaufbaus ab. „OPEN_FORM“, „WRITE_FORM“ etc. waren nun verborgen in einem Funktionsbaustein, den das System bei Aktivierung des Formulares automatisch bereitstellt. Der Anwender muss „nur“ noch das Datenbeschaffungsprogramm schreiben, die Daten für das Formular an den generierten Funktionsbaustein übergeben und das Formular selbst erstellen. Spezielles Coding im Formular selbst fließt automatisch in den generierten Funktionsbaustein ein.

Dennoch: Die Erwartungshaltung des Anwenders entwickelt sich stets weiter. An zwei Problemen kam man auch

bei SmartForms nicht vorbei: Interaktive Formulare und systemübergreifende Formular-Workflows ließen sich auch mit SmartForms nicht realisieren, da ein SmartForm außerhalb des SAP-Systems nicht mehr ist als „irgendeine proprietäre Formularlösung“. So entschied man sich zu einer Kooperation mit Adobe und unterstützt nun standardmäßig Adobe PDF in Form von *Adobe PDF Interactive Forms for SAP*, ohne jedoch die Tür für andere Formularlösungen zuzuschlagen. So kann die SAP oder ein Drittanbieter jederzeit auch eine andere Formularlösung verwenden, aber auf absehbare Zeit wird das Werkzeug des Formularentwicklers der Adobe Livecycle Designer sein.

Interaktive PDF-Formulare sind Formulare, die auf dem Stand der Technik sind (und eine ganze Weile bleiben werden), Interaktivität ebenso erlauben wie die Verwendung des Formulars über Plattformgrenzen hinweg, so dass ein interaktives Formular per WebDynpro oder z. B. auf einem Tablet mit Daten befüllt werden kann und die angereicherten Daten nach Zurücksendung im SAP-System ausgewertet werden können.

ABAP reloaded – Es geht dem Entwickler ans Werkzeug

Bisher war die Applikationsentwicklung in ABAP an die im SAP bereitgestellte Entwicklungsumgebung gebunden: Der Transaktion SE80 und ihrer „Untertransaktionen“. Um diesem ein Ende zu bereiten, haben sich zwei Entwickler bei der SAP vor einigen Jahren an Wochenenden zusammengesetzt mit dem ehrgeizigen Ziel, die ABAP-Entwicklung in Eclipse zu ermöglichen. Aus der Idee entstand ein Prototyp, aus dem Prototyp ein Projekt und inzwischen sitzt eine deutlich zweistellige Zahl an Entwicklern daran, das neue Eclipse-basierte Framework voranzutreiben. Und ja, es ist inzwischen sehr gut benutzbar, wenngleich die Entwicklung natürlich nicht abgeschlossen ist – eine Software ist nie „fertig“ im finalen Sinne und wenn man die Entwicklung an einem Programm abgeschlossen hat, bedeutet dies, dass sie das End-of-Life erreicht hat. So geschehen mit der Transaktion SE80, die in der Tat nicht mehr weiterentwickelt wird – wer neue Funktionalitäten, zum Beispiel im HANA-Kontext nutzen will, muss zu Eclipse greifen.

Eclipse ist gerade (aber nicht nur) bei Java-Entwicklern bekannt, die oft Hand in Hand mit den ABAP-Entwicklern zusammenarbeiten (wenn sie nicht gar beides in Personalunion sind) und für die sowohl die Navigation im System als auch in der Entwicklungsumgebung nicht selten eine Benutzungsschwelle darstellen. Diese wird deutlich überwindbarer gemacht, wenn ein Nicht-SAP-Entwickler die Entwicklungsumgebung und ihre Eigenarten schon kennt. Ganz besonders vorteilhaft wirkt sich jedoch aus, dass Kooperationsentwicklungen (also systemübergreifende Entwicklungen, die teilweise in Java und teilweise in ABAP durchgeführt werden), zusammen in einer Umgebung durchgeführt und gepflegt werden können.

Zwar hat die ABAP-Entwicklungsumgebung im Eclipse noch nicht den vollen Funktionsumfang wie eine SE80 der SAP, diesen „Makel“ hat man aber seitens SAP sehr klug gelöst: Für alles, was in Eclipse noch nicht geht, wird die SAP GUI nahtlos ins Eclipse integriert. Diese Einbindung ist sehr transparent gelöst, so dass sie auch zurück funktioniert: Ein Doppelklick auf den Aufruf einer DDIC-Suchhilfe führt in die Transaktion SE11 der in das Eclipse-Editorfenster eingeblendeten SAP GUI. Ein dortiger Aufruf eines Suchhilfeexits (der aus Coding besteht) wieder zurück in den Eclipse-Codingeditor.

Insbesondere bietet Eclipse den Vorteil einer offenen Lösung: Es gibt Heerscharen von Entwicklern, die nicht nur mit Eclipse arbeiten, sondern auch Plug-Ins dafür entwickeln und zur Verfügung stellen. Ein ganzes Eclipse-Ökosystem ist auf diese Weise entstanden. Die Vielzahl an Entwicklern, die damit arbeiten und qualifiziertes Feedback zu Programmfehlern geben, sorgen dafür, dass Eclipse sehr stabil arbeitet.

Noch ein kleines Detail: Jeder Entwickler kennt das Problem, dass schon ein sehr kurzzeitiger Abriss der Verbindung zwischen SAP GUI und Backend (wie er zum Beispiel beim Entnehmen des Laptops aus der Docking Station erfolgt, weil die Netzverbindung von LAN auf WLAN wechselt) zur Folge hatte, dass man sich komplett neu anmelden muss. Hier sieht man den ersten Vorteil der Eclipse-Lösung, die das Weiterarbeiten erlaubt und im Hintergrund eine neue Verbindungsmöglichkeit sucht und, soweit gefunden, nutzt. Der Entwickler kann nahtlos weiterarbeiten.

Von SAP R/2 zu SAP R/3, von SAP R/3 zu SAP HANA

Abseits von Marketingschlagworten wie „Big Data“ ist die Performance vieler Kundensysteme Anlass für häufig geäußerte Kritik, nicht zuletzt wird das Kürzel SAP oftmals übersetzt mit „Sanduhr-Anzeige-Programm“. Die Ursachen können hausgemacht oder systembedingt sein. Wenn wir mal von Kardinalfehlern wie zu klein dimensionierte Server oder Netze bzw. unperformanter ABAP-Entwicklung absehen, sind die häufigsten Quellen zu geringer Performance die folgenden:

- Dem Datenmodell widersprechende Anforderungen

  Nicht selten erleben Entwickler, dass fachliche Anforderungen formuliert werden, die das dem SAP zugrundeliegende Datenmodell vollumfänglich ignorieren und daher Selektionsroutinen notwendig machen, die von Haus aus gar nicht performant durchgeführt werden können. Gerade hier zeigt sich, dass fachliches Wissen, Beraterwissen und Entwickler-Know-How Hand in Hand gefragt sind, damit es gar nicht erst zu solchen abwegigen Anforderungen kommt.

- Der Flaschenhals Datenbankanbindung

  Die Entwicklung im SAP beruht auf dem Prinzip, dass ständig Zwischenergebnisse zwischen Applikationsserver und Datenbankserver ausgetauscht werden, weil die Daten in dem einen System gespeichert, aber im anderen System verarbeitet werden. So leidet die Gesamtperformance der Serverkette an ihrem schwächsten Glied: Dem Flaschenhals zwischen beiden Systemen in Form ihrer Verbindung.

- Die Performance der Datenbank selbst

  Man kann sich praktisch beliebig viele Applikationsserver „in den Keller stellen“, aber die Datenbank läuft prinzipbedingt nur auf einem, auf dem alle Anfragen abgearbeitet werden müssen. Man kann zwar als ABAP-Entwickler versuchen, die Zahl der Datenbankzugriffe zu verringern, um den Server so wenig wie möglich zu belasten, am Grundproblem ändert dies jedoch nichts. Also muss man die Zugriffsgeschwindigkeit im SB-System selbst verringern.

SAP HANA schlägt all diese (und noch mehr) Fliegen mit einer Klappe: Zum Einen werden erhebliche Teile der programmierten Logik mittels SQLscript auf den DB-Server selbst verlagert und so erreicht, dass nur die finalen Ergebnisse von Programmläufen durch den Flaschenhals müssen. Zum Zweiten wird die Datenbank von der Festplatte in den Arbeitsspeicher verlagert, was die Zugriffszeiten massiv beschleunigt. Kombiniert mit einer nicht mehr rein zeilenorientierten Sicht auf Datenbanktabellen, wie sie für transaktionsbasierte Systeme wie SAP wie geschaffen war, werden Laufzeiten selbst bei hohen Datenmengen so drastisch reduziert, wie man es kaum für möglich gehalten hätte. Die Datenmengen selbst werden durch spaltenorientierte Sichtweisen (column views) auf die Daten deutlich verringert, da sich die Daten in einer Spalte oft wiederholen und sich daher zum Komprimieren hervorragend eignen.

Nicht zuletzt ist es Hasso Plattner, der SAP HANA bezeichnet als „SAP, wie man es heute entwickeln würde“, der Wechsel von R/3 zu HANA ist in etwa vergleichbar wie der von R/2 zu R/3.

Gerüstet für das Next Generation Computing

In der Zusammenfassung kann man feststellen, dass derzeit ein deutlicher Wandel der SAP-Plattform stattfindet. Theatralischer formuliert: Wir beobachten gerade eine aus SAP-Sicht historische Weiterentwicklung der Plattform, die unser Arbeitsleben dominiert – ob als Anwender, Berater oder Entwickler. Das bisher streng hierarchische Client-Server-Konzept, das den SAP-Server mit der Datenbank im Hintergrund in den Mittelpunkt der IT stellte, weicht einem moderneren, weniger proprietären und erheblich selbstbewusster wirkenden Ansatz, bei dem SAP als Datendienstleister für eine Vielzahl unterschiedlichster Frontends auftritt.

Für Anwender, Berater und Entwickler stellen sich damit wieder einmal neue Herausforderungen, . Der informationstechnische Hintergrund insbesondere der Entwickler muss deutlich tiefer sein als in früheren Zeiten; bis hin zum bisher nicht verwendeten Paradigma der Funktionalen Programmierung. Gerade hier zeigt sich: Wer in der IT still steht, entwickelt sich effektiv zurück.