Integrative
Parameter- und Regelsysteme

Bestandteil einer jeden modernen Anwendungsarchitektur

Autor: Andreas Rajewski
BOI Software GmbH

Dezember 2005

BOI Software GmbH
Spazgasse 4 -- 4040 Linz (Austria)
+43 732 7364230 – boi@boi.at

Inhaltsverzeichnis

Vorwort

Im Rahmen unserer Projekte und Erfahrungen am Markt stellen wir zunehmend fest, dass Mitarbeiter von IT Abteilungen ein Problem damit haben, den Begriff Parameter- und Regelsystem richtig zu positionieren.

Dieses Dokument soll aufzeigen, wie zentral und sensibel ein Parameter- und Regelsystem in eine moderne Anwendungsarchitektur integriert ist.

Weiterhin ist es das Ziel dieses Dokuments, mehr Transparenz und Sensibilisierung zum Thema Parameter- und Regelsystem zu schaffen.

Neben der grundsätzlichen Definition und Positionierung dieser Thematik werden auch betriebswirtschaftliche Konsequenzen diverser technologischer Teilentscheidungen innerhalb eines Parameter- und Regelsystems aufgezeigt.

Parameter- und Regelsysteme finden sich in jeder Anwendung wieder. Dies resultiert aus dem sinnvollen Gedanken, Anwendungen zu parametrisieren, um in der Folgezeit schneller und effizienter Programmwartung und Weiterentwicklung betreiben zu können.

Ein Parameter- und Regelsystem ist innerhalb einer jeden IT-Lösung ein zentraler Bestandteil. Aus diesem Grunde ist es wichtig, Details, Vorgehensweisen, Erfahrungen und Entscheidungskonsequenzen darzustellen.

Allgemein

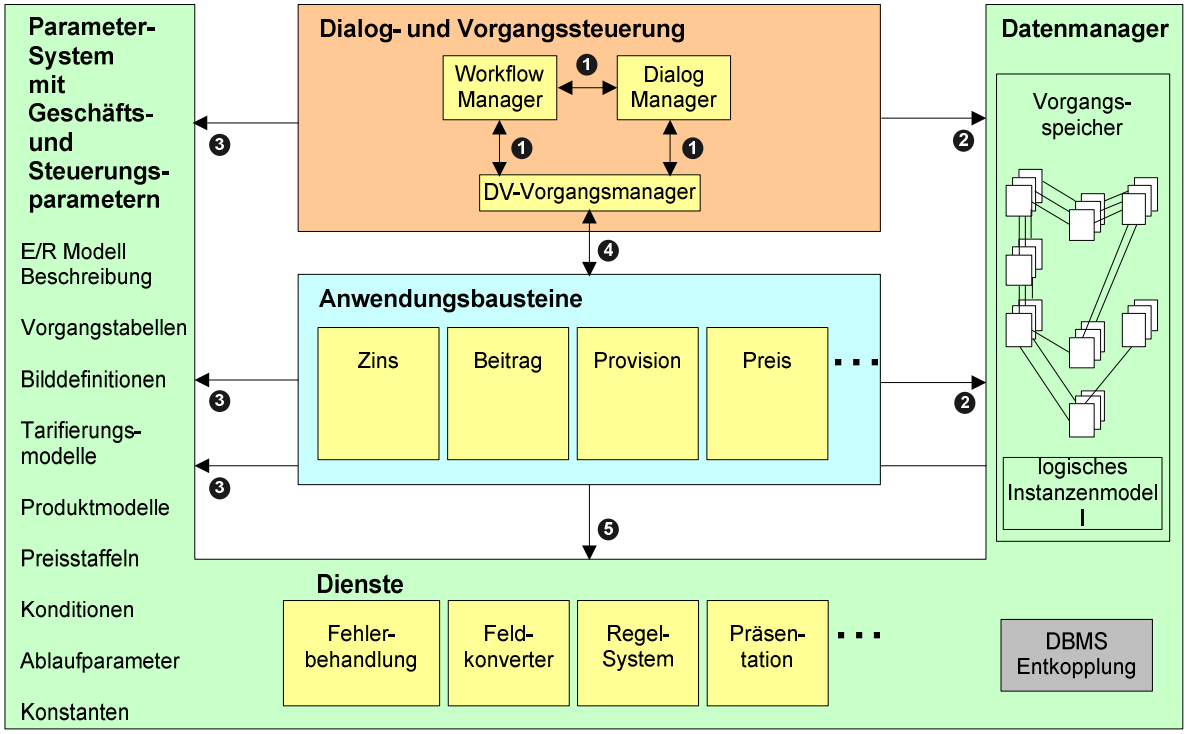
Das Parameter-Regelsystem ist eine der konstituierenden Software-Komponenten einer modernen Anwendungsarchitektur

Das Parametersystem stellt Funktionen zur Verwendung, Pflege und Verwaltung von Parametern und Regeldaten zur Verfügung, mit denen die Steuerungs- und Anwendungsbausteine konfiguriert werden.

In modernen Anwendungsarchitekturen werden die fachlichen Geschäftsprozesse eines Unternehmens mit Hilfe von Parametern direkt in der Software abgebildet, welche diese Geschäftsprozesse auf einer DV-Infrastruktur unterstützen soll.

Der Name "Parametersystem" verweist dabei explizit auf eine für die Anwendungsarchitektur fundamentale Kategorie von Daten, die für die Beschreibung der fachlichen Geschäftsprozesse sowie für die Definition von technischen Systemzusammenhängen bestimmt sind und für deren konkrete Ausprägungen sich im allgemeinen Sprachgebrauch die Begriffe "Parameter", "Parameterdaten" oder auch "Parameterobjekte" eingebürgert haben.

Grafik: Integration eines Parametersystems in eine moderne Anwendungsarchitektur



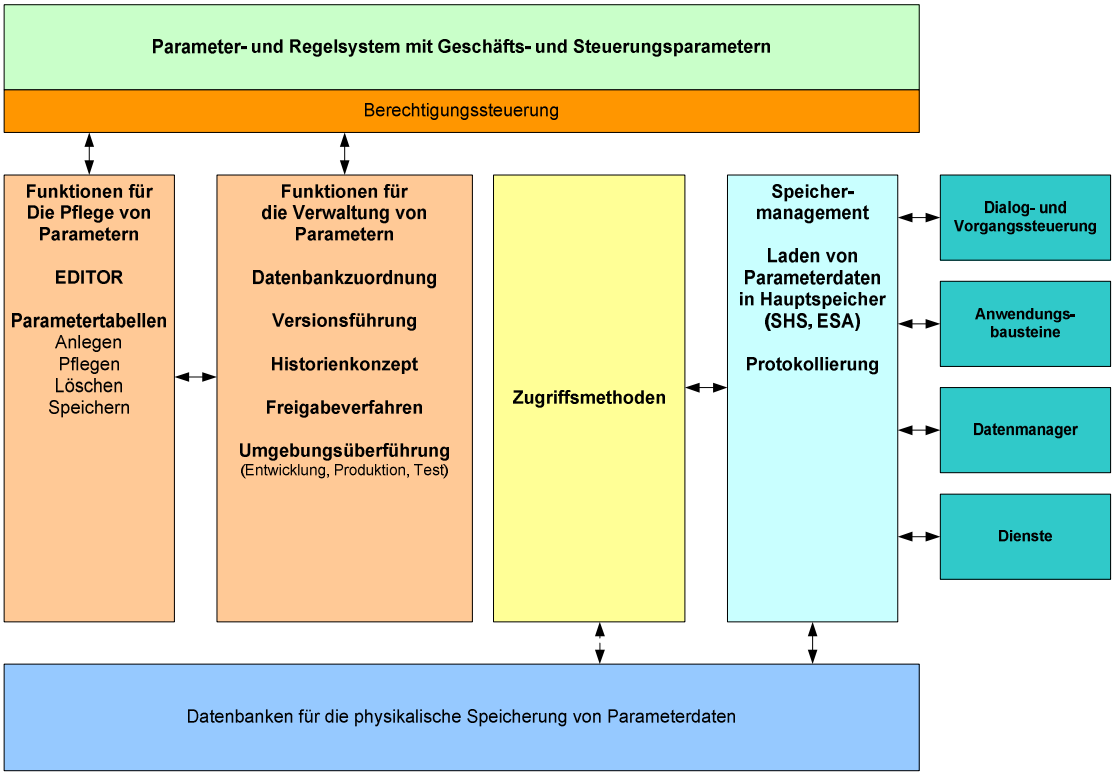
Ziele

Die vom Parametersystem bereitgestellten Dienste garantieren, dass die Anwendungsarchitektur in Bezug auf das in ihr mit Hilfe von Parametern abgebildete fachliche und technische Wissen konfiguriert werden kann (Flexibilität!)

und

dieses fachliche und technische Wissen mit durchgängigen Verfahren und Werkzeugen einheitlich behandelt werden kann (Zuverlässigkeit und Stabilität!)

Parameter- und Regelsystem im Überblick



Was möchte man mit dem Einsatz eines Parametersystems erreichen?

- ❑ Die benutzergerechte Erstellung neuen Wissens
- ❑ Eine einheitliche Verwaltung des Wissens
- ❑ Die Verfügbarkeit des Wissens in allen Nutzungsumgebungen
- ❑ Eine zeitgerechte Bereitstellung neuen Wissens
- ❑ Den hoch-performanten Zugriff auf das verfügbare Wissen
- ❑ Einsatz von relationalen Tabellenstrukturen auch für Daten der Parametrisierung und Konfiguration
- ❑ Spezialisierung auf das effiziente Handling von eher statischen Daten, d.h. insbesondere Änderungen ohne Systemunterbrechung zulassen
- ❑ Versionierung ermöglichen
- ❑ Handling verbessern
- ❑ Heterogene Techniken harmonisieren und erneuern
- ❑ Redundanzen beseitigen
- ❑ Mandantenfähigkeit herstellen
- ❑ Mehrsprachenfähigkeit herstellen
- ❑ Export/Import mit Batch-Prozessen ermöglichen

Elementare Funktionen zum Ausführungszeitpunkt (Run Time)

Bestimmte Zugriffsfunktionen des Parametersystems werden zur **Ausführungszeit (Run Time)** im Rahmen der dv-technischen Abwicklung fachlicher Geschäftsprozesse von allen anderen Softwarebausteinen der Anwendungsarchitektur mit einer **extrem hohen Frequenz** genutzt.

→ Parameterzugriff

Elementare Funktionen zur Entwicklungszeit (Build Time)

Zur Entwicklungszeit (Build Time) nutzen alle Personen, die mit der Konfiguration dieser Geschäftsprozesse betraut sind, die vom Parametersystem angebotenen Services. In der Regel sind dies Mitarbeiter aus DV- und Fachabteilungen.

→ Parameterpflege (Tabellenverwaltungssystem)
→ Versionsführung (TVS)
→ Überführungsverfahren (Entwicklung, Test, Vorproduktion, Produktion)
→ Datenverteilung über heterogene Plattformen hinweg

Parameter sind Konfigurationsdaten

Parameterobjekte sind Daten, die ausschließlich zur **Entwicklungszeit** entstehen und zur **Ausführungszeit** von den nutzenden Softwarebausteinen **nur gelesen** werden.

Auf diese greifen die Softwarebausteine der Anwendungsarchitektur im Rahmen einer konkreten Geschäftsprozessverarbeitung **signifikant häufiger zu** als auf die so genannten "operativen Datenbestände", welche in modernen Anwendungsarchitekturen ausschließlich über so genannte Datenmanager verwaltet werden.

Parameter sind charakteristische Konstanten

Parameterobjekte sind charakteristisch für die Softwarebausteine, von denen sie als Konstante verwendet werden, und können nach "ihren" Softwarebausteinen klassifiziert werden.

Das logische Datenmodell beispielsweise wird zur Run Time in der Anwendungsarchitektur exklusiv vom Datenmanager benutzt.

Parameter sind Typdaten für Geschäftsprozesse

Parameterobjekte legen den Typ des **konkreten Geschäftsprozesses** fest.

Damit werden die generischen Grundalgorithmen der Komponenten der Anwendungsarchitektur auf ihre für die jeweilige Geschäftsprozess- bzw. DV-Vorgangs-Instanz-spezifische Verarbeitung eingestellt.

Was für Komponenten müssen im Rahmen eines Parametersystems zur Verfügung gestellt werden

Im Rahmen der Architektur- und Projektplanung muss darüber nachgedacht werden, welche Teilkomponenten für ein Parametersystem zur Verfügung gestellt werden müssen. Nachfolgend eine Übersicht der wesentlichen Komponenten und Funktionen.

→ Parameter/Tabellen-Editor

→ Programme für die Verwaltung- und Pflege von Parameterdaten
- → Organisationsfunktionen für die Definition und Umsetzung eines Versionskonzepts
- → Organisatorische Funktionen für die Speicherung von Parametern auf unterschiedlichen Datenbanken in unter Umständen unterschiedlichen IT-Landschaften
- →Berechtigungskonzepte die steuern, wer darf was, wann und wo
- → Organisatorische Lösungen für die Freigabe von Parameter-Korrekturen (Freigabekonzept)
- → Organisatorische Lösungen für die Überführung von Parameterdaten (Entwicklung, Test, Vorproduktion, Produktion)

→ Synchronisationsverfahren für die revisionssichere Verteilung von Parameterdaten

→ Hoch-performanter Parameter-/Tabellenzugriff
- → Zugriffsmethoden müssen in hoher Entwicklungsqualität zur Verfügung gestellt werden, damit die notwendige hohe Zugriffsfrequenz auf Parameterdaten die Performance der Gesamtanwendung nicht negativ beeinflusst.

Welche Konsequenzen ergeben sich bei der Entwicklung von eigenen Parametersystem-Lösungen

Entscheidet man sich dafür, ein Parametersystem eigenständig umzusetzen, müssen unter anderem die folgenden Konsequenzen mit in Betracht gezogen werden.

Ein selbst-entwickeltes Parameter- und Regelsystem ist im Normalfall auf spezifische Anwendungsanforderungen fixiert. Ein globaler Entwicklungsansatz scheitert in den meisten Fällen an fehlenden finanziellen, zeitlichen oder personellen Ressourcen.

Im Normalfall verbauen sich viele Unternehmen die Strategie eines unternehmensweit einheitlichen Parametersystems aufgrund der oben angeführten Problematik.

Die Argumentation, dass solche zentralen Kernanwendungen immer selbst entwickelt werden sollten(müssen), um Know how im Hause zu schaffen (in den eigenen Reihen vorzubehalten), ist als kritisch zu betrachten, da nach produktiver Einführung der Anwendungen die Thematik in das operative Geschäft übergeht und Entwickler in andere Projekte abwandern, durch Fusionen (optimiert werden) oder aus anderen Gründen nicht mehr verfügbar sind.

Als Konsequenz folgt die Sicherheitsstrategie „never change a running system“. Dies wird insbesondere im Zusammenhang mit produktiven host-basierten Anwendungen deutlich.

Heute „verdienen“ Großkonzerne ihr Geld in der Regel mit IT-Anwendungen, die auf Großrechnern (IBM z/OS, FSC BS2000) laufen. Diese Anwendungen basieren auf Cobol, PL/1 oder Assembler. In Managementkreisen ist bekannt, dass viele dieser Programme aufgrund von Performance-Problemen zu hohe laufende Kosten produzieren, bzw. in periodischen Zyklen zu operativen Engpässen führen.

Im Rahmen von Datenbank-Tuning-Massnahmen mit und ohne ergänzenden SQL Optimierungen, versucht man mühsam und mit hohen Investitionen die Performance zu steigern.

Aufgrund der bestehenden Unsicherheit, hervorgerufen durch Wissenslücken und/oder Druck vom TOP Management, scheuen sich IT Manager, ihre bestehenden Parameter- und Regelsysteme zu analysieren. Diese Art von Projekten ist nicht Karriere-förderlich, da die Zukunft Standardsoftware auf UNIX oder INTEL basierten Plattformen heißt.

Unternehmen, die dennoch den Schritt wagen und ihre Parameter- und Regelsysteme optimieren, erzielen immer wieder hervorragende und vor allem auch langfristige Ergebnisse.

Ebenfalls fällt es den Unternehmen schwer, plattformübergreifende und datenbankneutrale Parametersysteme zu entwickeln, da meist nicht das notwendige Gesamt-Know How vorhanden ist. Somit gibt man sich in der Regel mit isolierten und intransparenten Parametersystem zufrieden.

SQL als Parameterzugriffssprache

An dieser Stelle beschäftigen wir uns intensiver mit der Thematik SQL als Parameterzugriffssprache. Um ein notwendiges Grundverständnis für SQL zu erlangen, haben wir nachfolgend nochmals die Definition und Kurzhistorie dieser Abfragesprache angeführt.

SQL ist eine deklarative Abfragesprache für Relationale Datenbanken. SQL hat eine relativ einfache Syntax, die an die englische Umgangssprache angelehnt ist, und stellt eine Reihe von Befehlen zur Definition von Datenstrukturen nach der Relationalen Algebra, zur Manipulation von Datenbeständen (Anfügen, Bearbeiten und Löschen von Datensätzen) und zur Abfrage von Daten zur Verfügung. Durch ihre Rolle als Quasi-Standard ist SQL von großer Bedeutung, da eine weitgehende Unabhängigkeit von der benutzten Software erzielt werden kann.

Die meisten SQL-Implementierungen bieten darüber hinaus allerdings noch Hersteller-spezifische Erweiterungen, die nicht dem Standard-Sprachumfang entsprechen. Dies hat zur Folge, dass von den Herstellern parallel entwickelte gleiche Funktionen unterschiedliche Sprachelemente benutzen.

Viele bekannte Datenbanksysteme wie DB2, Informix, Microsoft SQL Server, Pervasive P.SQL, MySQL, Oracle, PostgreSQL, Borland Interbase, Firebird, SQLite und die neueren Versionen von Access implementieren Teile des SQL Sprachstandards.

SQL hat eine langjährige Historie und hat sich zur führenden 4GL Sprache auf der Welt entwickelt.

Warum überhaupt darüber nachdenken, ob SQL die richtige Lösung für den hoch-frequenten Zugriff auf Parameterdaten ist?

Technisch betrachtet bietet SQL alle erdenklichen Möglichkeiten, sinnvoll und korrekt auf Parameterdaten zuzugreifen. **Jedoch nimmt sich SQL und die dazu gehörende Datenbankumgebung die entsprechende Zeit, um das gewünschte Ergebnis zu liefern.**

Im Rahmen eines direkten Vergleichs zwischen SQL (IBM-DB2) und einem am Markt etablierten Zugriffsystem wurden im Mai 2005 im Rahmen einer offiziellen GSE DB2 User Tagung in Bad Honnef folgende Vergleichsergebnisse vorgestellt:

Testszenarien:

- Zufälliges Lesen von 2.000.000 Datensätzen einer Tabelle mit 65.000 Einträgen
- Umstellung von zwei Unterprogrammen eines kritischen Batch-Prozesses

Testergebnisse I:

2.000.000 Zufalls-Zugriffe auf 65.000 Sätze

Messung 1: Single DB2 Selects aus dem Programm

Messung 2: TABEX/3 Zugriffe mit Schlüssel

Messung 3: Vorladen der Tabelle im Programm,
Zugriff mit SEARCH ALL

Scenario	CPU (s)	SRB	Elapsed Time (s)
DB2	184	1.486.799	224
Tabex/3	12	97.480	17
SEARCH ALL	17	130.391	17

Einsparung:
Tabex vs. DB2: **> 90% CPU, > 90% Elapsed**
Tabex vs. SEARCH ALL: **30% CPU**

Testergebnisse II

In einem performance-kritischen Batch-Ablauf werden zwei Unterprogramme geändert:

UP1: 550.000 Select-Zugriffe auf eine Parameter-Tabelle werden durch TABEX/3 ersetzt (Anteil: 13,75% der Job CPU)

UP2: Ein Join zwischen einer Parameter-Tabelle und einer anderen Tabelle wird abgeändert, sodass der Cursor nur noch die andere Tabelle verarbeitet und die Daten aus der Parameter-Tabelle über TA-BEX/3 Zugriffe zugesteuert werden (Anteil: 9,98% der Job CPU)

Ergebnis: Der CPU-Verbrauch des Jobs wurde um 20% gesenkt

Neutrale Testergebnisse wie diese sollten dazu ermuntern, auch nach sinnvollen Alternativen gegenüber SQL zu suchen. Im Kapitel ‚Betriebswirtschaftliche Betrachtungen’ werden wir nochmals dieses Thema reflektieren.

Hochperformanter Tabellenzugriff

Definition:
„ Im Rahmen eines Parameter- und Regelwerks müssen Parameterzugriffsmethoden zur Verfügung gestellt werden, die in Abhängigkeit zum gewünschten Lieferergebnis mit höchstmöglicher Performance auf Parameter zugreifen. Parameterzugriffe charakterisieren sich dadurch, dass sie mit höchstmöglicher Frequenz lesend auf Parameter zugreifen und diese in der Folgezeit der jeweiligen Anwendung schnellstmöglich zur Folgeverarbeitung zur Verfügung stellen“.

Warum ist das heute überhaupt noch ein Thema, was an dieser Stelle behandelt wird? Aufgrund der zur Verfügung stehenden Hardware müsste dieser Punkt doch vernachlässigbar sein.

Leider hat sich in vielen Managementkreisen der Trugschluss gefestigt, dass „billige“ Hardware die offensichtlichen Schwächen „moderner“ Zugriffmethoden kompensieren können.

Projekte aus dem Alltag sprechen hingegen eine andere Sprache:
Performance-Probleme, wachsende Serverfarmen, missglückte Entwicklungsprojekte, insbesondere dann, wenn es sich um komplexe Anwendungen mit großen und umfangreichen Datenvolumen handelt.

Aber wieso kommt es immer wieder zu diesem Trugschluss?

Das Problem liegt in vielen Fällen darin begründet, dass Manager, Entwickler und Architekten die Abhängigkeiten von einzelnen Anwendungskomponenten gar nicht mehr kennen und somit Konsequenzen ihrer Entscheidungen und Entwicklungen gar nicht mehr realistisch abschätzen können.

Moderne Entwicklungsplattformen und ergänzende Tuning-Werkzeuge kompensieren in vielen Fällen die vermeintlichen Entwicklungsschwächen, die z.B. zu Performance-Problemen führen. Jedoch muss an dieser Stelle angeführt werden, dass grundsätzliche architektonische Fehler nicht auf diese Weise gelöst werden können.

Insbesondere der Zugriff auf Parameter wird im Rahmen von Architekturplanungen immer wieder unterschätzt. Was sind die Gründe dafür?

- → Die einzelne Parameterinformation ist nicht besonders anspruchsvoll
- → Das Gesamtgerüst von notwendigen Parametern ist nicht transparent
- → Die Aufrufhäufigkeit dieser Parameter wird zu wenig in Relation eines wachsenden Verarbeitungsvolumens gesetzt.
- → Entwickler haben keine Vorstellung bezüglich der möglichen Konsequenzen eines schlecht definierten und realisierten Parameter- und Regelsystems
- → Der Aufruf von Parametern steuert unter anderem die Abhängigkeit von Programmen und damit verbunden Folgeverarbeitungen -- diese Tatsache wird zu wenig berücksichtigt und kann aber in einer gewissen Entwicklungsphase nur noch mit großem Aufwand optimiert werden.

Funktioniert ein Parameter- und Regelsystem nicht optimal, hat in der Regel die gesamte Anwendung ein elementares Problem. Performance-Probleme, organisatorische Lücken und Mehraufwendungen sind die Folge.

Entscheidungsträger berufen sich immer auf die Tatsache, dass sie alles Mögliche getan haben. Tuning-Maßnahmen, auf Standards gesetzt etc..... aber ist das wirklich ausreichend?

Die damit verbundenen Mehrkosten werden stillschweigend hingenommen, da man sich dazu verleiten lässt, gemäß dem Motto ‚Was 1000 andere entscheiden kann für mich auch nur gut sein’ zu handeln.

Hier erliegen die Manager jedoch einem großen Irrtum, da die daraus resultierenden (schleichenden) Mehrkosten ganz erheblich die eigenen IT-Budgets strapazieren. Im Rahmen des Rechenschaftsberichts gegenüber der Geschäftsführung oder dem Vorstand kommt es meist zum bösen Erwachen. Fehlende Millionen können nicht ausreichend argumentiert werden.

Als Ergebnis folgen die bekannten Kostenoptimierungsmaßnahmen. Gekürzte Budgets, Personalentlassungen und in vielen Fällen der Verlust des eigen Arbeitsplatzes sind die Folge dieser „**bedingungslosen** Standardisierungsstrategie“.

Mit einer „**sinnvollen** Standardisierungsstrategie“ wären viele dieser Maßnahmen zu vermeiden.

Betriebswirtschaftliche Konsequenzen

Im Gegensatz zu vielen wissenschaftlichen oder rein IT-technisch fokussierten Abhandlungen widmen wir diesem Teilbereich ein besonderes Augenmerk.

„Moderne“ Managemententscheidungen werden neben technologischen Gesichtspunkten immer stärker aufgrund von betriebswirtschaftlichen Fakten entschieden.

Hierbei konzentrieren wir uns auf die kurz- und langfristigen Investitionen und Kosten.

Kurzfristige Investitionen und Kosten bei Eigenentwicklungen

→ Schaffen von Projektrahmenbedingungen
→ Definitionsphase und Erstellung Teilprojektplan
→ Erstellung Pflichtenheft
→ Erstellung Entwicklungsplan
→ Analyse für die Integration in die Unternehmens-Anwendungsarchitektur
→ Sammlung von notwendigen Werkzeugen (Tools) für die Anlage, Pflege und Verwaltung von Parametern
→ Entwicklung von notwendigen Programmteilkomponenten für die Verwaltung und Organisation von Tabellen
→ Entwicklung von notwendigen Programmschnittstellen
→ Entwicklung von einem oder mehreren Parameterzugriffsmodulen
→ Programmtests (Funktions-, Logik- und Lasttests, etc.)
→ Realisierung Prototyp
→ Ergebnisanalyse
→ Dokumentationserstellung
→ Anwendertraining
→ Produktive Überführung

Erfahrungen haben ergeben, dass die Entwicklung und Integration eines solchen eigen-entwickelten Parameter- und Regelsystems mehrere Mannjahre in Anspruch nehmen kann.

Am konkreten Fall der KORDOBA GmbH & Co. KG ([*1)] Profil), hätten nach Schätzungen des Entwicklungsteams der Classics Line mehr als 15 Mannjahre für ein Classic Line-spezifisches Parameter- und Regelsystem investiert werden müssen.

Im Rahmen der Analyse wurde festgestellt, dass man für die Realisierung eines eigenständigen Parameter- und Regelsystems einen hochqualifizierten Stamm von Fachpersonal benötigt.

Im Falle der Classics Line-Entwicklung war es das Ziel, verschiedene Plattformen (IBM z/OS, SUN SOLARIS, FSC BS2000, WINDOWS und LINUX) zu unterstützen.

Schnell wurde deutlich, dass der notwendige Personenkreis aufgrund der Spezifikationen der einzelnen Plattformen hätte erweitert werden müssen.

15 Mannjahre müssen heute mit einem durchschnittlichen Jahressatz (Senior-Consultants) von Euro 100.000,00 angesetzt werden. Daraus resultierend hätte sich nur die Entwicklung eines eigenen Parameter- und Regelsystems auf ca. **1.500.000,00** Euro belaufen. Das notwendige Zeitfenster hätte das Gesamt-Redesign-Projekt der Classic Line gefährdet.

Aus diesem Grund hat KORDOBA sich für den Einsatz eines am Markt etablierten Parameter- und Regelsystems entschieden.

Das Beispiel der KORDOBA verdeutlicht, dass es durchaus sinnvoll ist, sich intensiv mit einem solchen sensiblen Thema zu beschäftigen.

Die Kosten für die Einführung der Standardlösung beliefen sich auf einen Bruchteil der oben angeführten Investition.

Betriebswirtschaftliche Betrachtung hinsichtlich der Entscheidung, SQL als Parameterzugriffsprache zu nutzen

Was spricht für einen Einsatz von SQL:

→ SQL ist weit verbreitet und entsprechendes Know How vorhanden
→ SQL wird als Standard am Markt akzeptiert (ISO)
→ Breites Einsatzspektrum

Was spricht gegen den Einsatz von SQL:

→ Performance-Grenzen auch nach entsprechender tool-basierter und manueller SQL Optimierung
→ Notwendiges Know How für die optimale Definition und Umsetzung der Anforderungen

Greifen wir nochmals das Kapitel „SQL als Parameterzugriffsprache“ auf und versuchen wir, die gewonnen Erkenntnisse betriebswirtschaftlich dazustellen.

Hierzu treffen wir für die Konkretisierung verschiedene Annahmen, die sich selbstverständlich von Anwendung zu Anwendung unterscheiden. Dennoch basieren diese Annahmen auf realen Informationen von Unternehmen und Veröffentlichungen.

Berechnungsbeispiele

In diesem Fall wurde die Annahme getroffen, der Kostenindex für eine CPU Sekunde beträgt 21 Cent. 20 % der CPU Auslastung erfolgt aufgrund des Parameterzugriffs mittels SQL.

Als Ergebnis werden die anfallenden SQL-Kosten ermittelt. Ebenfalls hat man in die Berechnung den alternativen Einsatz eines am Markt verfügbaren Standardparameter- und Regelsystem einfließen lassen.

Die Ergebnisse sind aus betriebswirtschaftlicher Betrachtung interessant.

Minuten/Tag	Sekunden/Tag	Kostenindex für eine CPU Preis Sekunde (*1)	Kosten/Tag
1440	86400	0,21 €	18.144,00 €

Anteil SQL-Zugriff (lesend) Parameter relevant (*2)	Anteilige Kosten mit SQL	Anteilige Kosten mit TABEX/3
20%	3.628,80 €	403,20 €

Optimierungsfaktor TABEX/3 (*3)	Tägliche Ersparnis	Monatliche Ersparnis
9	3.225,60 €	96.768,00 €

Jährliche Ersparnis pro CPU
1.161.216,00 €

(*1) Laut Compass Studien;
Stand: Sommer 2003
Preisindex der alle wesentlichen Kosten inkludiert

(*2) Durchschnittlicher SQL Lastanteil pro CPU (lesender Zugriff)

(*3) Optimierungsfaktor von TABEX/3 9=Dieser Faktor ist der bisher schlechteste Wert, den TABEX/3 im Gegensatz zu SQL erzielen konnte. Regelfall ist ein Faktor zwischen 15 und 30

Berechnungsbeispiel II

In diesem Fall wurde die Annahme getroffen, der Kostenindex für eine CPU Sekunde beträgt nur 16 Cent. 15 % der CPU Auslastung erfolgt aufgrund des Parameterzugriffs mittels SQL.

Als Ergebnis werden die anfallenden SQL-Kosten ermittelt. Ebenfalls hat man in die Berechnung den alternativen Einsatz eines am Markt verfügbaren Standardparameter- und Regelsystem einfließen lassen.

Die Ergebnisse sind aus betriebswirtschaftlicher Betrachtung interessant.

Minuten/Tag	Sekunden/Tag	Kostenindex für eine CPU Preis Sekunde (*1)	Kosten/Tag
1440	86400	0,16 €	13.824,00 €

Anteil SQL-Zugriff (lesend) Parameter relevant (*2)	Anteilige Kosten mit SQL	Anteilige Kosten mit TABEX/3
15%	2.073,60 €	414,72 €

Optimierungsfaktor TABEX/3 (*3)	Tägliche Ersparnis	Monatliche Ersparnis
5	1.658,88 €	49.766,40 €

Jährliche Ersparnis pro CPU
597.196,80 €

(*1) Laut Compass Studien; Stand: Sommer 2003 Preisindex der alle wesentlichen Kosten inkludiert

(*2) Durchschnittlicher SQL Lastanteil pro CPU (lesender Zugriff)

(*3) Optimierungsfaktor von TABEX/3 9=Dieser Faktor ist der bisher schlechteste Wert, den TABEX/3 im Gegensatz zu SQL erzielen konnte. Regelfall ist ein Faktor zwischen 15 und 30

Fazit ist, dass es durchaus interessante am Markt erhältliche Alternativen gibt, die in jedem Fall in den Entscheidungsprozess, welche Zugriffsstrategie man wählen möchte, einbezogen werden sollten.

Betrachten wir die beiden Ergebnisse einmal auf 10 Jahre:

Berechnungsbeispiel I: 11.612.160,00 Euro Mehrkosten durch SQL

Berechnungsbeispiel II: 5.971.960,00 Euro Mehrkosten durch SQL

Alternative Zugriffsmethoden

In den 80iger und 90iger Jahren wurden am Markt von unterschiedlichen Nischenspezialisten Parameterzugriffsmethoden angeboten, die einen hoch-performanten Parameterzugriff sicherstellten.

Diese Produkte hießen VTAS (Firma: Insoft, Düsseldorf), SPITAB (Firma: EDS jetzt Technologies, Paris) und TABEX (Firma: BOI, Linz)

Die ersten beiden Anbieter haben ihre Unternehmensstrategien geändert und sich auf andere Themenbereiche konzentriert.

Aktuelle Standardlösungen am Markt

BOI Software GmbH, Linz

Als Marktführer hat sich inzwischen das Produkt TABEX aus dem Hause BOI in der vierten Generation etabliert.

Über 60 Unternehmen aus der Region DACH haben sich bisher für den erfolgskritischen Einsatz der Produktfamilie TABEX entschieden.

Bisher wurde im Rahmen aller Vergleichstests zwischen SQL- und TABEX-Zugriffsmethoden ein Verhältnis von 9:1 (für SQL das beste Ergebnis) erreicht.

Details: www.boi.at

Orchestra Networks, Frankreich

Die EBX.Platform 3 ist ein XML basiertes Parametersystem, das sich auf den JAVA-spezifischen Einsatz konzentriert.

Performance-Vergleiche zu SQL sind derzeit nicht bekannt.

Details: www.orchestranetworks.com

Resümee

Integrative Parameter- und Regelsysteme sind auch aus zukünftigen Anwendungsentwicklungen nicht wegzudenken.

Die in diesem Dokument aufgezeigten positiven und auch negativen Konsequenzen, die Parameter- und Regelsysteme mit sich bringen können, sollen dazu beitragen, wieder mehr Bewusstsein für diesen Teil einer jeden Gesamtanwendungsarchitektur zu schaffen.

In jedem Fall nutzen die meisten Unternehmen heute bewusst oder unbewusst nicht alle sinnvollen am Markt zur Verfügung stehenden Möglichkeiten zur Schaffung von Zufriedenheit beim Anwender und zur Optimierung der Kosten.

Parameterbeispiele

Attribute

- Allgemeine Versicherungsbedingungen für die einzelne Sparten z.B. AHB, AKB, AFB, VHB
- Schlüssel Vertragsbezeichnung
- Versicherungssumme(n), Deckungssumme(n), Haftungssumme(n)
- Dynamikregelung

Beispiel

- K-Haftpflicht
- K-Vollkasko
- K-Teilkasko
- K-Unfall
- Verkehrs-RS
- Familien-RS
- Haftpflicht
- Hausrat

Attribute

- Klasse: Haupt- oder Nebengefahr
- Art: Risiko-, Wagnis-, Tarifart-Kennziffer
- Abweichungen vom Standard (Erweiterungen, Ein-/Ausschlüsse)

Beispiel

- bei Personen: Unfall, Tod, Krankheit, Berufsunfähigkeit
- bei Objekten: Feuer, Diebstahl, Raub, Wassereinbruch

Attribute

- Leistungs-Kennzeichen
- Schadensdaten
- Schaden-Registrierjahr
- Schadensursache, Todesursache
- Abwicklung
- Schaden-Reserve
- Einleitung Ermittlungsverfahren
- Leistungsaufwand

Beispiel

- Unfall, Feuer
- Tod der versicherten Person
- Rückzahlung von Fondsanteilen

Attribute

- Objektart
- Identifikationsbezeichnung
- Anschrift bzw. Standort

Beispiel

- Versicherte Person
- Immobilie
- Wohnungs-/Betriebseinrichtung
- Fahrzeug, Maschine
- Objekt auf Reisewegen, z.B. auf Schiffen in der Transportversicherung

Attribute

- Stammdaten
  - Kontonummer
  - Art des Kontos, z.B. Beitragskonto, Provisionskonto
  - Saldo
- Buchungen
  - Soll- und Haben-Betrag
  - Buchungsdatum
  - Buchungsschlüssel

Beispiel

- Beitragskonto
- Provisionskonto
- Leistungskonto
- Gehaltskonto

Attribute

- Regelwerk-Kennzeichen
- Prämienberechnungsgrundlage
- Gewinnbeteiligung
- Riskiertes Kapital
- Gebühren (-Regelung)
- Sterbetafel
- Risikobeitrag
- Sparbeitrag
- Einmalbeitrag
- Rabatt- / Zuschlagsregelung
- Tarifgruppe
- Überstundendivisor

Beispiel

- Versicherungsbedingungen
- Klauseln
- Tarifwerk
- Tarifvertrag
- Provisionsregelung
- Organisatorische Regelungen

Attribute

- Dokument-Kennzeichen
- Rechtskräftigkeit
- Datum

Beispiel

- Police
- Nachtrag
- Antrag
- Angebot
- Anmeldung
- Prämienrechnung
- Zahlungsträger
- Schriftwechsel
- Urkunde (z.B. Bürgschaft)
- Versicherungsübersicht
- Risikoanalyse-Ergebnis
- Rundschreiben

Attribute

- Versicherungsscheinnummer, Schadensnummer, Sachbearbeiternummer
- Datum (Termindatum)
- Art der Referenz

Beispiel

- Geschäftsprozessnummer
- Vorgangsart
- Verfügender Sachbearbeiter
- Durchlaufene Stellen
- Bearbeitungstermine
- Korrespondenz zum Vorgang

Parameterzugriffsfunktionen

Nachfolgend die wesentlichen bekannten Zugriffsfunktionen auf Parameterdaten.

- Zugriff über Primärschlüssel
- Zugriff über Zeilennummer
- Sequentielle Suche
- Teilsuche
- Generic-Suche und Generic-Teilsuche
- Zugriff über Matchcode
- Zugriff über Ergebnisindex
- Zugriff auf View-Tabellen

Begriffe

Adapter
Entkoppelungsschicht zwischen Basissystem des Parametersystems und Anwendungskomponenten zur RUNTIME.

Aktivierung
Zeitpunktgerechte Übergabe der Parameterobjekte in die Produktionsumgebung.

Data Dictionary
Das Data Dictionary enthält alle Informationen über die Objekte in der Datenverarbeitung je Unternehmen, sogenannte Metadaten.

DV-Vorgang
Ein DV-Vorgang bildet einen Teilprozeß aus dem Prozeßmodell in eine konkrete DV-Umgebung ab. Er kann aus mehreren Dialogschritten (Transaktionen?) und fachlichen Funktionen (Transaktionen?) bestehen.
Ein zeitlicher Prozeß, für den Parameter und Software konstant sind.

FSE
Full Screen Editor, dient der Anzeige und Änderungen von Tabellen und Parametern.

Generierungsprozeß
Der Prozeß für die Erzeugung der Parameter aus Source-Informationen,
Bsp.: Übersetzung eines Programms, Abspeicherung der Parameterdaten im internen Format mit Zugriffsmethode.

Geschäftsparameter
Alle für die Abwicklung der Geschäftsprozesse im Versicherungsunternehmen notwendigen Parameter.
Bsp.: Tarifschlüssel, Beitragsfaktoren.

Globalattribute
Diese Attribute werden in mehreren Ebenen pro Anwendungssystem festgelegt (z.B. pro Adressraum, Prozess) und gelten automatisch für alle darunter liegenden Aufrufe an das Parametersystem. Diese zusätzlichen Attribute kennzeichnen den Parameter als eindeutig.
Beispiele: Sprachmerkmal, Landeskennzeichen, Firma, Test, Produktion, Server, ESA.
Die Globalattribute werden sowohl zur RUNTIME als auch zur BUILD-TIME benötigt.

Globale Gültigkeit
Status des Parameters, mögliche Ausprägungen : Test, Systemtest, Produktion.

Installationsdaten

Alle für die Installation der Bausteine notwendigen Daten. Sie sind Teil des Parametersystems.

Konsistenz/Abhängigkeiten
Die Definition der gültigen Dateninhalte erfolgt auf formaler Ebene, durch andere Bausteine oder durch RI zu anderen Parametern oder Entitäten. Die Prüfung auf fachliche Integrität erfolgt durch die speziellen Parmatereditoren.

Konstante
Ein Wert, der für einen bestimmten Zeitraum fix ist. Der Zeitraum wird durch den Prozess definiert, in dem der Wert benutzt wird.

LOCATE-Modus
Übergabe der Parameteradressen an Softwarebausteine.

MOVE-Modus
Übergabe der Parameterdaten an Softwarebausteine.

Parameter
Charakteristische Konstante; sie legt das Wesen und die Eigenarten des betrachteten Problems fest.

Parametereditoren
Der Tabelleneditor soll Standard sein. Außerdem gibt es spezielle Editoren für die Parameter von Bausteinen.

Parameterobjekte
Zusammenfassung der zu den Parametern gehörenden Daten, Beschreibungen, Strukturen und Zugriffsregeln.

Parametersystem
Das Parametersystem stellt Funktionen zur Verwendung, Pflege und Verwaltung von Parametern und Regeldaten zur Verfügung, mit denen die Steuerungs- und Anwendungsbausteine konfiguriert werden.

Parametertypen
Die verschiedenen Parametertypen unterscheiden sich in der Zugriffsmethode und dem Generierungsprozeß.
Bsp.: Programme, Tabellen, unstrukturierte Parameter.
Periodische/aperiodische Änderung
Festlegung des Rhythmus der Änderungen für den Parameter.
Bsp.: periodisch --> einmal je Monat, aperiodisch --> nach Anforderung.

Persistenz
Dauerhafte Speicherung der Daten.

Realtime
Merkmal für die Wirksamkeit einer Aktivierung des Parameters in dem RUNTIME-System.
Mögliche Abstufung:
sofort in allen Systemen, auch in laufenden Prozessen,
sofort in allen Systemen, nach Beenden der aktiven Prozesse,
festen Terminen (z.B. Beginn BATCH-Produktion).

Regeln / Code
Regeln und Code sind logische Operationen oder Programm Statements, sie werden als Dateninhalte der Parameter betrachtet und nicht vom Parametersystem interpretiert.

Revisionsfähigkeit
Die Änderung und Speicherung der Daten muss den Anforderungen der Revisionsfähigkeit genügen, dies sind insbesondere folgende Punkte:
lückenlose Protokollierung der Änderungen
Versionsführung (kein UPDATE ?)
Datensicherung (Recovery-Verfahren)
Berechtigungssystem (Schnittstelle)

Schlüssel
Primärer Suchbegriff und Ordnungsbegriff einer Parametermenge.

Sensible Parameter
Maßstab für die Auswirkung der Änderung dieses Parameters für das VU,
0=minimale bis 4=maximale Auswirkung.
Bsp.: Beitragsfaktoren --> 4,IPL-Parameter --> 4, Fehlertexte --> 1

Standard Parameter-Editoren
Ein Tabelleneditor soll Standard sein. Außerdem gibt es spezielle Editoren für die Parameter von Bausteinen.

Sub-Parameter
Eine Ausprägung oder Exemplar des Parameters, z.B. : eine Zeile aus einer Parametertabelle, ein Wort aus einem Parameterstring.

Technische Parameter
Alle für die Abwicklung der DV-Vorgänge im VU notwendigen Parameter.
Bsp.: IPL-Parameter, Parameter für den Datenmanager

Zeitliche Gültigkeit
Festlegung der datumsabhängigen Gültigkeit des Parameters. Zwei Zeitachsen sind vorzusehen:
Dateistand, Termin der Änderung in der Datei
Datengültigkeit, Termin der Gültigkeit für den Inhalt des neuen Datensatzes.

Zugriffshäufigkeit

Anzahl der Zugriffe des RUNTIME-Systems auf den Parameter je Zeiteinheit. Dies ist ein Maßstab für die notwendige Performance.

Bsp.: IPL-Parameter --> geringe Anzahl, Parameter für den Datenmanager --> hohe Anzahl.

Zugriffssystem

Zusammenfassung der Funktionen der RUNTIME-Schnittstelle des Parametersystems.

Zusammenhängende Sub-Parameter

Die Sub-Parameter des Parameters sind sowohl für Prüfungen bei der Änderung, als auch aus Performance-Gründen für das RUNTIME-System zusammenhängend.

*1)Profil

Mit 25 Jahren Erfahrung in der Entwicklung von hochwertigen Standard-Softwareprodukten ist die **KORDOBA GmbH & Co. KG** eines der führenden Softwarehäuser im deutschen Bankenmarkt.

Die ganzheitliche Geschäftsverantwortung des Unternehmens erstreckt sich vom Consulting über Development bis hin zum Outsourcing. KORDOBA bietet damit Produkte und Leistungen entlang der gesamten Wertschöpfungskette an und übernimmt projektspezifisch auch die Rolle des Generalunternehmers.

Das Produkt-Portfolio umfasst das **plattformunabhängige** Core-Banking-System KORDOBA® Classic sowie buchungskernneutrale Lösungen für die Bereiche Wertpapiergeschäft, Banksteuerung und Autorisierung / SB-Netzbetrieb.

Um die hohe Qualität der KORDOBA-Produkte zu sichern und ihre Effizienz im Einsatz zu verbessern, werden diese ständig weiterentwickelt und gleichzeitig umfangreichen Integrations- und Systemtests bis hin zur Pilotierung unterzogen. Ausgewählte moderne Methoden und Tools unterstützen die Entwicklung und Pflege der Applikationen.

KORDOBA - Ihre Vorteile

- schnell und flexibel auf veränderte Marktbedingungen reagieren
- Pflege der Kundenbeziehungen signifikant verbessern
- Kosten senken
- Wettbewerbsfähigkeit erhöhen

Zum Kundenkreis gehören Privat- und Regionalbanken sowie Universalbanken und Direktbanken.

Die Anforderungen an die Produkte und die Umsetzung gesetzlicher Forderungen werden im engen Zusammenspiel mit den Kunden und Partnerunternehmen diskutiert, geplant und realisiert.
Standardschulungen und Spezialschulungen zu den KORDOBA-Produkten werden vor Ort oder im Training Center in München angeboten.