3 Beispielszenarien und Voraussetzungen

Die Situation, die man in vielen IT-Abteilungen heute vorfindet, lässt auf den ersten Blick kaum hoffen, dass eine Vereinfachung oder gar Vereinheitlichung der Systemlandschaften möglich ist. Analysiert man jedoch das Vorgefundene genau und konzentriert sich bei der Neustrukturierung auf einige wenige zukunftsfähige Standards, beginnt sich der Nebel rasch zu lichten.

3.1 Die nahe Zukunft

Nachdem die letzten Jahre zwar durch das Internet geprägt waren, aber kaum eine klare Linie erkennbar wurde, auf welche Technologien sich die IT-Welt letztlich einigen würde, kristallisieren sich mittlerweile ein paar Standards heraus, die in naher Zukunft das Geschehen im Enterprise-Computing dominieren werden.

3.1.1 XML-Datenbanken

XML dient der Datenstrukturierung

XML (*Extensible Markup Language*) ist eine Sprache zur strukturierten Beschreibung von Daten. Da die meisten Applikationen mittlerweile die Daten von einem externen Partner in XML-Form verlangen, liegt es nahe, dass die Datenbank Anfragen und Ergebnisse direkt in XML mit dem Client austauscht.

XML ist jedoch kein alleinstehender Standard; in seinem Umfeld finden sich vielmehr verschiedenste Spezifikationen, die alle auf den Grundprinzipien von XML aufbauen. Ein wichtiger Standard innerhalb dieser »XML-Familie« zum Austausch von Dokumenten ist das *Simple Object Access Protocol* (SOAP).

An die Stelle von Open SQL, ODBC oder JDBC tritt somit also XML. Für den Anwendungsentwickler verhält sich die Datenbank so, als seien die Daten als Dokumente im hierarchischen XML-Format abgespeichert.

3.1.2 Entwicklung in mehreren Programmiersprachen

Microsoft setzt Standards

Bei der Öffnung der eigenen Plattform für verschiedene fremde Programmiersprachen hat Microsoft ganz deutlich den Trend gesetzt. Jede wie auch immer geartete Programmiersprache wird ihre Freunde und ihre

Gegner haben. Begeisterung und Ablehnung sind manchmal in nachvollziehbaren Argumenten begründet, meistens jedoch basieren sie auf eher emotionalen Beurteilungen. Mit Microsoft.NET hat Microsoft unmissverständlich klar gemacht, dass es keine Programmiersprache in irgendeiner Weise bevorzugen oder benachteiligen will. Ab sofort kann man auf Microsoft.NET mit allen unterstützten Sprachen gleichzeitig entwickeln. Dies war freilich auch schon Ende der 80er Jahre mit Borlands Turbo Pascal 3 möglich, nur blieb dies eine Spezialität, die ihrer Zeit um zehn Jahre voraus war.

Derzeit existiert Microsoft.NET noch ausschließlich auf Windows-Rechnern, obwohl das Framework ähnlich wie Java so entwickelt wurde, dass es selbst als vollständiges Betriebssystem agieren kann. Somit ist auch zu erwarten, dass eine Portierung auf andere Plattformen rasch erfolgen wird.

Bis Ende 2003 kann man mit Microsoft.NET auf Mac OS rechnen, und selbst ein Zuschnitt auf Linux wird nur eine Frage der Zeit sein, auch wenn immer wieder unterstellt wird, Microsoft.NET sei vor allem ein Marketing-Vehikel für Windows. Setzt man jedoch die enorme Leistungsfähigkeit des Frameworks dagegen, kann man durchaus erwarten, dass es auf lange Sicht ein Eigenleben entwickeln wird.

SAP NetWeaver

Auch SAP öffnet sich mit dem SAP NetWeaver für die Entwicklung mittels verschiedener Programmiersprachen. Heute wird auf dem Web AS bereits die Applikationsentwicklung mit ABAP und Java unterstützt und es existiert auch bereits ein Konnektor für die .NET-Plattform.

3.1.3 Grid-Computing

Bisher gingen wir immer von der Vorstellung aus, dass in einem Netzwerk eine ganze Reihe von spezialisierten Computern stehen, die jeweils eine oder ein paar wenige Programme ausführen. Dabei wird zum Beispiel SAP R/3 auf Box SAP1, SAP2 und SAP3 ausgeführt, die hauseigene, selbst geschriebene Produktionsplanung auf Box PP01 und gedruckt wird auf einer Windows-Box NT501. Nun kann es in der Praxis schon einmal vorkommen, dass sich SAP1 langweilt, während NT501 beim Formatieren der Monatsrechnungen heftig ins Glühen kommt.

Lastverteilung

An diesem Punkt setzt nun das Konzept des Grid-Computing an. Anstatt eine Software genau einer Applikations-Box zuzuweisen, werden die Hardware-Boxen nach ihren Möglichkeiten klassifiziert. Wenn zum Beispiel der Printer-Server feststellt, dass seine Box überlastet ist, sendet er eine Anfrage nach mehr Rechnerpower an einen Message-Server. Der

Message-Server sucht nun in der Liste der verfügbaren Boxen eine aus, die momentan wenig ausgelastet ist, lädt dort die notwendige Software und verteilt die Last teilweise auf diese Box um.

3.2 Komponenten und Protokolle

Zu den Kernkompetenzen in einem Integrationsteam gehört das Wissen um Protokolle und das Denken und Entwickeln in Komponenten. Die verschiedenen Frameworks bieten jeweils eigene Komponentenmodelle an. Dabei ist unter »Komponenten« eine abgeschlossene modulare, transaktionale Einheit zu verstehen. Solche Komponenten heißen in Java *Beans*, in SAP je nach Betrachtungsweise *BAPI* oder *Logical Unit of Work* (LUW) und in Microsoft-Umgebungen *COM-Objekte*.

Die wichtigsten Protokolle, die zwischen den Komponenten den Datenaustausch kontrollieren, sind COM+ bei Microsoft, CORBA und RMI bei Java und RFC bei SAP. Diese Protokolle dienen dazu, die von der jeweiligen Umgebung bereitgestellten APIs der Komponenten sauber aufzurufen. Abbildung 3.1 und Tabelle 3.1 vermitteln aber ein Bild davon, welche Fähigkeiten für einen erfolgreichen Technikberater unverzichtbar sind.

	Micro-soft.NET	**Visual Basic (Microsoft Office)**	**J2EE**	**R/3, SAP Web AS**
Microsoft.NET	.NET	COM+	RMI/ CORBA	1. RFC Controls 2. .NET Connector
Visual Basic (Microsoft Office)	COM+	COM+	COM+	RFC Controls
J2EE	JNI	JNI	RMI/ CORBA	Java Connector (JNI)
SAP R/3	RFC via OLE2	RFC via OLE2	RFC	RFC
SAP Web AS	RFC via OLE2	RFC via OLE2	RMI	RFC

Tabelle 3.1 Zugriffstechniken zwischen den Frameworks

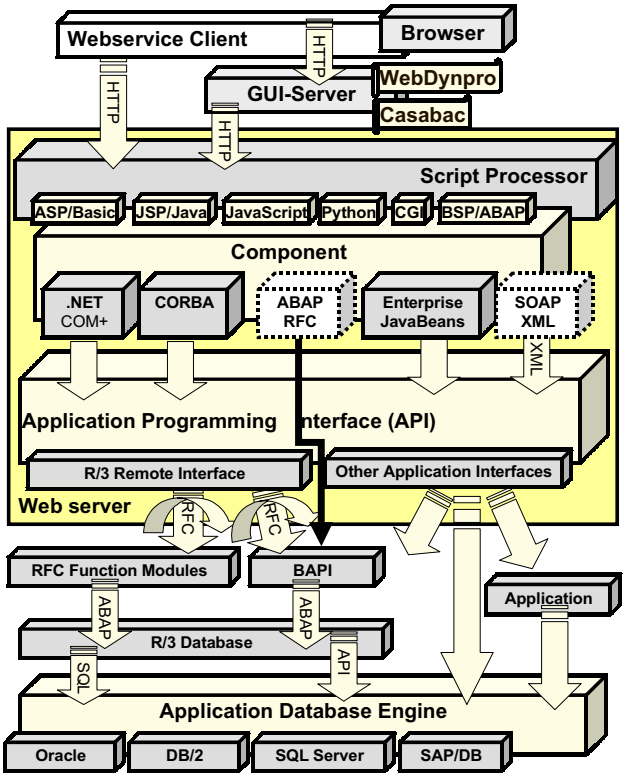

Abbildung 3.1 Komponenten, API und RPC-Protokolle

3.3 Ein Beispiel einer heutigen Unternehmens-IT

So wie früher der Großvater vom Krieg und so mancher gestandene Professor von seiner Studienzeit erzählt hat, erzählen heute die COBOL- und FORTRAN-Programmierer von Lochkartenstapeln und Magnetbändern, Assembler-Programmen mit so mystisch klingenden Befehlen wie `Branch-and-Link`, nächtelangen Jobs zum Erstellen von Fakturen und Monatsabschlüssen, geheimnisvoll im Halbdunkel aufgestellten Monitoren mit leuchtend grüner Schrift und wie man die Siebzigjährigen aus Mallorca hatte einfliegen lassen, als beim Upgrade der Zentraleinheit der Rechner partout nicht mehr den IPL machen wollte.

Austausch hoch spezialisierter Software ist riskant

Ganz so lange ist das wahrscheinlich doch noch nicht her, und in so manchem Betrieb gibt es zwar vermutlich keine Lochkarten mehr, aber schwarz-grüne Monitore gibt es immer noch zuhauf. Denn es ist eine Tat-

sache: Man trennt sich ungern von den alten, zuverlässigen Maschinen, die direkt die Produktion kontrollieren und in denen all das Wissen steckt, das so manch ein Programmierer mit ins Grab genommen hat.

Typische Computer-Infrastruktur heute

Schauen wir uns einmal eine typische Computer-Infrastruktur eines produzierenden Betriebs an. Wohlgemerkt: Die Rede ist von produzierenden Betrieben, also solchen Firmen, die gewöhnlich einen individuell entworfenen, hochkomplexen Produktionsprozess steuern. Dies ist anders als etwa beim Handel, wo die Prozesse überall dem gleichen Schema folgen und deshalb eine einmal geschriebene Software für eine ganze Branche taugt. In einem solchen produzierenden Betrieb findet man also z. B. die folgenden Systeme (kein Idealzustand, sondern ein realistisches Szenario):

- zentrales SAP R/3-System (neun Applikationsserver, ein Datenbankserver) für
  - FI/CO: Buchhaltung und Controlling
  - MM: Materialwirtschaft
  - SD: Verkaufsabwicklung
  - PP: Produktionsgrobplanung
- dezentrales SAP R/3-System (zwei Applikationsserver, ein Datenbankserver) für
  - die Verkaufsbüros in Spanien und in Singapur
  - teilweise Replizierung der Daten über ALE
- Transportabwicklungssystem (ein PC) mit selbst geschriebener Software für
  - Lagerverwaltung
  - Etikettendruck
  - Gefahrgutdokumentendruck
  - Warenausgangserfassung
- Produktionsrechner (drei AS/400)
  - Rezepturverwaltung
  - Chargenverwaltung
  - Produktionsfeinplanung
- Maschinenleitstand
  - Steuerung der Produktion
  - Chargenverwaltung (sic! siehe oben)

- PC mit hauseigener Software für
  - Beschaffung Produktion
  - Entsorgung Reste
- ein PC
  läuft Tag und Nacht und steuert den Datenverkehr mit Zulieferern, ist aber in keinem Plan verzeichnet
- eine AS/400
  darauf laufen alle Programme, die nicht offiziell abgenommen wurden
- acht zentrale NT-Server für das Office-Netzwerk
- drei Linux-Server
  von denen aber keiner genau weiß, was sie tun
- drei WWW-Server für das Internet
- ein WWW-Server für das Intranet
- ein WWW-Server
  für den Internetshop, in dem die Bestellungen erfasst werden, dann ordentlich ausgedruckt und von Hand im R/3-Zentralsystem eingegeben werden
- ein Firewall-Server
- ein NT-Proxy-Server
  über den allerdings gar kein Netzverkehr läuft; wahrscheinlich dient doch einer der Linux-Rechner als Proxy-Server für das WWW
- drei extern aufgestellte NT-Server zum Daten-Backup
- zwei EDI-Server (von denen sich einer nicht mehr booten lässt)
- zwei weitere WWW-Server
  die aber als normale Workstations geführt sind; schaltet man sie ab, bricht der Einkauf zusammen, obwohl der eigentlich über SAP R/3 abgewickelt wird
- ...

Programme sind das Know-how einer Generation

Es gibt viele Gründe, warum eine zentral organisierte, redundanzfreie IT-Landschaft nicht realistisch ist. Besonders schwer wiegen psychologische Gründe wie Angst vor Neuem, mangelndes Vertrauen in die neue Software oder fehlende Ausbildung. Der wesentliche Grund ist aber: Computer speichern Wissen und die Programme modellieren mit ihren Algorithmen komplexe Arbeitsprozesse. Genauer gesagt: Die Programme sind die einzigen wirklich zuverlässigen Arbeitsanweisungen und Prozessbeschreibungen. Da aber Arbeitsprozesse komplex sind und Programmierspra-

chen von Natur aus kryptisch (das gilt auch und vor allem für RPG, C und Java), lassen sich die so dokumentierten Arbeitsanweisungen und Prozessbeschreibungen nur schwer lesen.

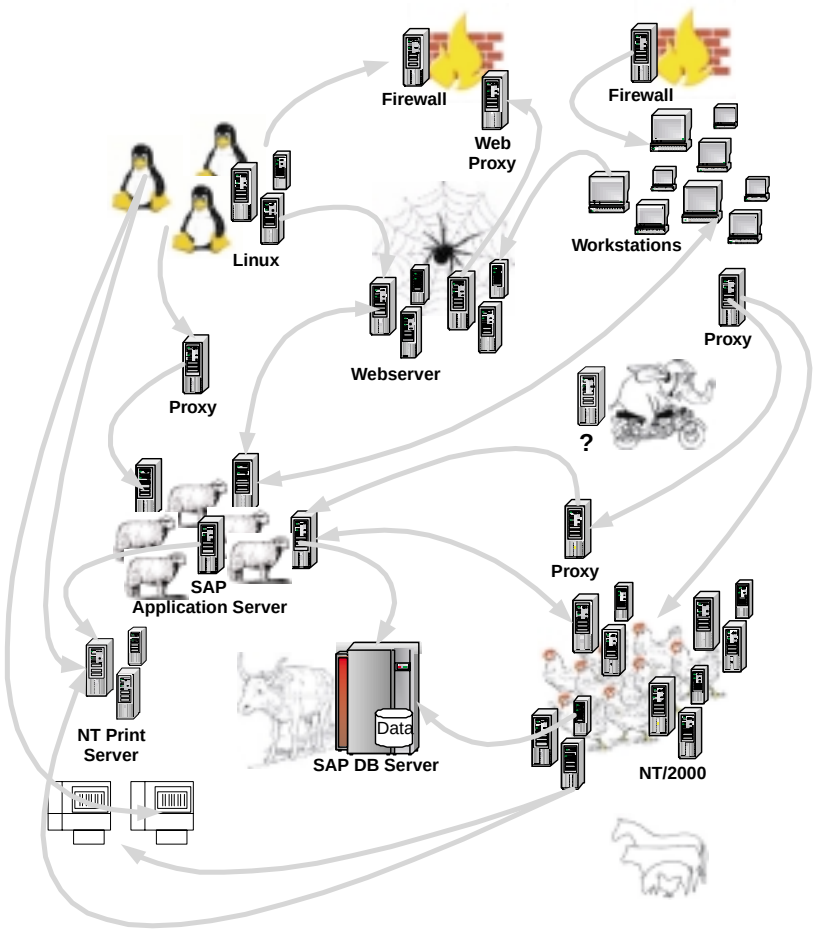

Abbildung 3.2 Vereinfachte Darstellung einer Serverfarm

3.3.1 Modellsichten und Abstraktion

Realistisch betrachtet, sind solche Serverfarmen keineswegs so chaotisch, wie sie auf den ersten Blick erscheinen mögen. Tatsächlich spiegeln sie nur die ebenso komplexe Realität des Tagesgeschäfts eines Unternehmens wider. Auch dieses ist selten gradlinig, einfach und frei von Redundanz. Um dies zu veranschaulichen, betrachten wir einmal das tägliche Geschäft eines Unternehmens.

Vertriebswege Die Produkte werden auf verschiedenen Wegen an den Endkunden geliefert und jeder Vertriebsweg und sogar jeder Kunde kann noch einmal ein eigenes Verfahren bei der Abwicklung nach sich ziehen. So verkauft ein Unternehmer die Produkte unter anderem:

- direkt
- über Großhändler
- indirekt als Streckengeschäft an Zwischenhändler, zum Beispiel ein Versandhaus. Hierbei lassen sich dann folgende Fälle unterscheiden:
  - Der Zwischenhändler erhält für jeden Verkauf eine Sofortprovision.
  - Der Zwischenhändler erhält für jede bezahlte Ware eine Endprovision.
  - In Einzelfällen fakturiert und liefert das Unternehmen im Namen des Versandhauses und macht auch für einzelne Händler das Inkasso beim Endkunden.
- über einen selbst betriebenen Webshop (mit dem Angebot von Sonderfertigungen)

Wer glaubt, man könne den Kunden seine eigenen standardisierten Strukturen verordnen, ist nicht nur kundenunfreundlich, sondern verlagert das Problem schlicht auf die Seite des Kunden, denn dieser hat es ja auch mit vielen Lieferanten zu tun, die sicher nicht alle die gleichen Vertriebsstrukturen haben. Es gibt nur ganz wenige Branchen, wie zum Beispiel den Buchhandel, die es bis heute geschafft haben, einheitliche Vertriebsstrukturen konsequent durchzusetzen.

Produktion In der Produktion existieren folgende Fertigungsvarianten:

- Serie auf Lager
- Serie auf Termin
- Make-to-order (Variantenkonfiguration)
- Make-to-order mit Überschuss aufs Lager

Versand Auch im Versand gibt es verschiedene und kundenindividuelle Varianten:

- direkte Express-Auslieferung
- direkte Auslieferung durch Sammel-Spedition (Ground-Collect)
- Lagerauffüllung nach Bedarf (KANBAN) und Termin (Lieferplan)
- Bereitstellung zur Abholung auf Termin

Technische Sicht

Die Sicht des Anwenders ist nicht die des Ingenieurs. Um die Szenarien auf einem Computer zu realisieren, brauchen wir eine Abstraktion der gezeigten Anwendungsfälle auf Software-Applikationen, was uns zu folgender minimaler Installation führt:

- ein WWW-Server und eine Webshop-Applikation
- eine Applikation zur Verwaltung der Kundenaufträge
- eine Applikation zur Bestellung der Rohwaren
- eine Applikation zur Verwaltung des Lagers
- eine Applikation zur Produktionsplanung
- eine Applikation zur Versandplanung und Versandsteuerung

Ablauf

Um das Ganze noch einmal aus einem anderen Blickwinkel zu betrachten, beleuchten wir im Folgenden die mögliche zeitliche Abfolge eines Verkaufs:

1. Ein Kunde ruft die Webseite auf und fragt ein Produkt an.
2. Der Webserver simuliert die Machbarkeit durch Simulation eines Kundenauftrags mit Variantenkonfiguration.
3. Der Kundenauftragsrechner lässt sich vom Produktionsrechner ein geschätztes Fertigstellungsdatum errechnen.

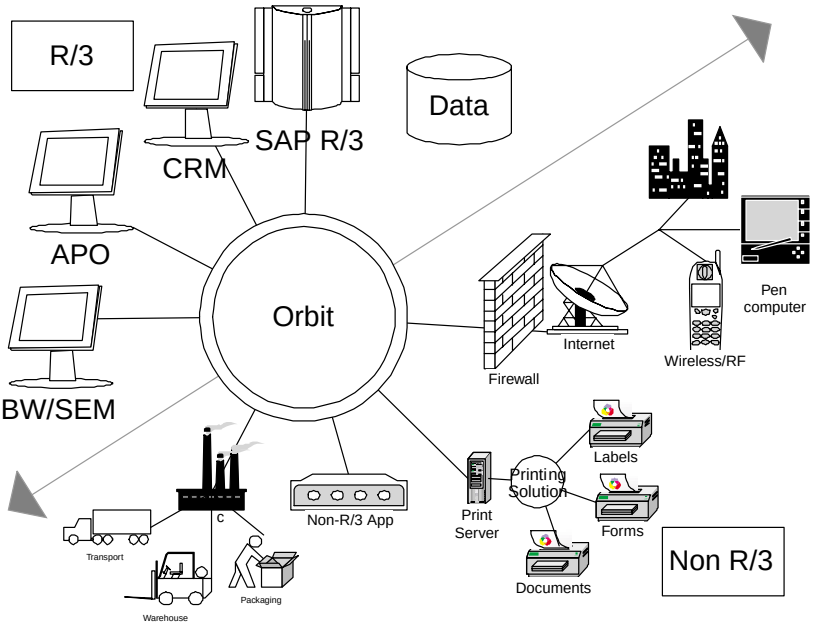

Abbildung 3.3 Kommunikationsorbit eines Unternehmens

4. Der Webserver lässt sich vom Kunden den Auftrag bestätigen und sammelt die Zahlungsinformation (Kreditkarte, Bankeinzug etc.) ein.
5. Der Kundenauftrag wird angelegt und bittet den Produktionsrechner, die Produktion fest einzuplanen.
6. Die Produktion teilt dem Versand die Verfügbarkeit mit.
7. Der Versand avisiert dem Kunden die Lieferung.
8. Der Versand ordert die Spedition entsprechend den Versandbedingungen.
9. Der Versand bestätigt den Versand.
10. Der SD-Rechner fakturiert die versandte Ware.

3.3.2 Ansatz für eine zeitgemäße Lösung: Integration mit HTTP und XML

Komplexe Systeme lassen sich nicht planen und modellieren

Es ist müßig, sich in Theorien zu ergehen, wie man eine solche Struktur durch eine saubere und gezielte Planung aufräumen kann. Solche Pläne funktionieren nur in der Theorie, denn sie sind fast niemals redundant und sturmsicher. Außerdem wird immer wieder übersehen, dass Pläne selten so umgesetzt werden, wie sie ursprünglich angesetzt wurden, stattdessen werden sie im Alltag rasch zerfleddert und selten mit Konsequenz zu Ende gebracht. Diese Erkenntnisse aus der Theorie der komplexen Systeme (auch *Chaostheorie* genannt) – und ein Produktionsbetrieb ist ein solches komplexes System – begründen aber nur in nachvollziehbarer Weise, was in der Praxis bereits eine Binsenweisheit ist.

Integration als einzig praktikable Lösung

Klinisch saubere Strukturen sind starre Strukturen. In einem lebendigen Betrieb, wie ihn eine Produktion darstellt, sind aber Flexibilität und Geschwindigkeit der Reaktion das A und O. Und deshalb werden vor allem die produktionsnahen Rechner und die Rechner aus der Lagerverwaltung so bleiben, wie sie sind, weil niemand die Verantwortung übernehmen wird, einen funktionierenden Rechner ohne Not durch einen anderen, nicht erprobten auszuwechseln. Als Lösung bietet sich also lediglich die Integration der spezialisierten Einheiten in das gesamte *Enterprise Network* an.

Um den Planungs- und Abstimmungsaufwand gering zu halten, funktioniert die Integration der Applikationen immer nach dem gleichen Schema. Dieses fußt auf HTTP und transportiert Daten zwischen den Systemen grundsätzlich in XML. Die Aufbereitung und Konvertierung von Daten erfolgt durch den konsequenten Einsatz von XSLT-Schablonen.

In Abbildung 3.2 ist der Browser repräsentativ für einen beliebigen Client (der Client kann aber auch ein anderes Programm sein). Somit ist der Ablauf wie folgt:

1. Ein Client fragt eine bestimmte Ressource an.
2. Ein HTTP-Server nimmt die Anfrage entgegen und führt eine Applikation aus.
3. Sobald ein Ergebnis vorliegt, wird dieses in XML aufbereitet.
4. Der Browser nimmt das XML entgegen und formt es mit Hilfe eines XSLT-Dokuments um.
5. Nun liegt das Ergebnis in der gewünschten Form vor und kann weiterverarbeitet werden.

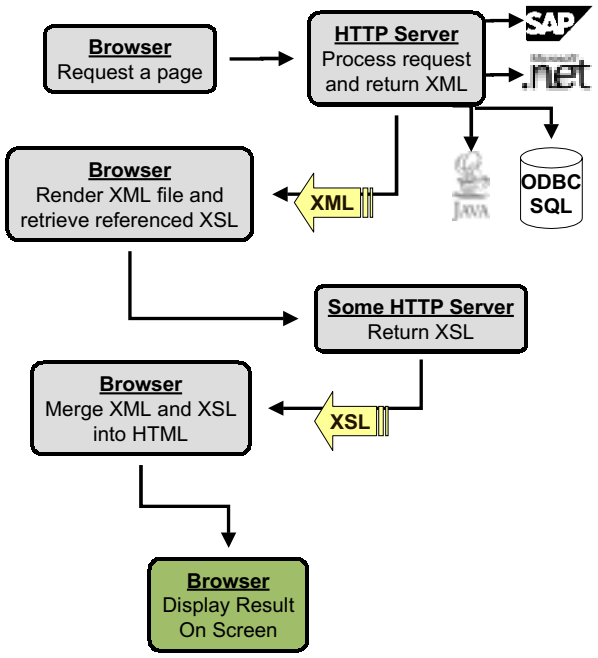

Abbildung 3.4 Typischer Programmfluss einer Client-Server-Applikation mit HTTP

Teil 2
Remote Procedure Calls und Business API

Remote Procedure Calls (RPC) sind das Herz jeder Client-Server-Architektur. Durch sie wird es möglich, Programme von einem Computer aus auf einem anderen Computer auszuführen. Durch dieses Konzept kann man ein Netzwerk von spezialisierten Einzelcomputern entwickeln, die reibungslos miteinander kommunizieren.