2 Middleware

Dieses Kapitel gibt einen Überblick über die verschiedenen Komponenten und Konzepte innerhalb einer Systemlandschaft. Im weiteren Verlauf dieses Buches werden wir immer wieder auf sie zurückkommen und Wege aufzeigen, mit welchen Mitteln diese Konzepte realisiert und die Komponenten in einen produktiven Austausch gebracht werden können.

Als *Middleware* bezeichnet man jede Art von Software, die eine Vermittlerrolle zwischen zwei oder mehreren über ein Netzwerk kommunizierenden Softwarekomponenten übernimmt. Eine Middleware arbeitet dabei immer nach dem Prinzip einer Postverteilstation, an der Nachrichten entgegengenommen und an einen Empfänger zugestellt werden. Neben der eigentlichen Nachrichtenverteilung übernimmt die Middleware aber auch jede Menge Routineaufgaben, wie die Konvertierung der Nachrichten zwischen den Formaten des Senders und des Empfängers – zum Beispiel von EDIFACT nach IDoc –, die zeitversetzte Auslieferung einer Nachricht, wenn Sender und Empfänger verschiedene Arbeitsgeschwindigkeiten haben, oder die Ansteuerung und Überwachung von Automatisierungs- und Workflow-Diensten.

2.0.1 Proxy-Dienste

Ein *Proxy* ist eine Server-Software, die die Kommunikation zwischen zwei entfernten Applikationen herstellt. Ein Proxy kann dabei die Anfragen filtern, bei Bedarf auch formale Transformationen vornehmen und gegebenenfalls Workflows anstoßen.

Firewall als Sonderform des Messaging-Dienstes

Die Grenzen zwischen Middleware und Proxy-Servern für Sonderdienste sind fließend, und häufig ist jeder Versuch einer terminologischen Abgrenzung Haarspalterei. In der Praxis werden alle Zugriffe von Internetnutzern auf einen Unternehmenscomputer durch eine Firewall auf der Seite des Anbieters geleitet. Eine Firewall ist ein Proxy-Server mit einer Sonderaufgabe, der alle Nachrichten entgegennimmt, die zwischen zwei Computern ausgetauscht werden, diese analysiert und dann an den eigentlichen Ziel-Computer weiterleitet. Gegebenenfalls weist der Proxy die Anfrage zurück, wenn sie verdächtig oder nicht identifizierbar ist.

Durch den Einsatz eines Proxys erscheint somit für den Client das Unternehmensnetzwerk wie ein einzelner Computer, denn der Anfragende sieht lediglich den Firewall-Computer, von dem er auch alle Antworten erhält. Hinter dieser Firewall können sich eine Vielzahl an individuellen Computern und spezialisierter Software befinden.

Firewall kann ein Message-Server sein

Es liegt nahe, dass der Firewall-Computer die Anfragen vorsortiert und sie entsprechend dem Inhalt an einen geeigneten Computer weiterleitet. In diesem Fall mutiert die Firewall zu einem ausgewachsenen Message-Server. Ein solcher Message-Server stellt dann eine Reihe von Funktionalitäten zur Verfügung, die von sehr vielen Applikationen gewünscht werden.

2.0.2 Proxy-Dienste im Sicherheitsbereich

Proxy-Dienste sind das Handwerkszeug jeder Sicherheitsadministration. Ein Proxy dient nicht nur als Firewall oder Vermittler von Diensten, sondern auch als elektronischer Gendarm und Geheimbote, der die Daten zwischen den Partnern verschlüsselt überträgt und die Identität der Partner überprüft.

Secure Remote Password und Single Sign On

Eine solch besondere Form eines Proxys stellt auch *Secure Remote Password* (SCR) dar beziehungsweise als Variante davon *Single Sign On* (SSO). Dabei handelt es sich um einen Server, der die Zugangsdaten (englisch: *credentials*) für den Zugriff auf eine Ressource an einer geheim gehaltenen Stelle und nach einer ebenfalls geheimen Strategie ablegt. Sobald ein Client den Zugriff auf die Ressource verlangt, entscheidet der SCR-Server, auf Basis welcher Informationen der Zugriff gestattet werden kann. Ein Beispiel für kommerzielle SSO-Dienste ist Microsoft Passport; ein einfacher SSO-Mechanismus für interne Applikationen zum Zugriff auf SAP ist im Anhang in der Visual-Basic-Klasse `Credentials` dargestellt.

Dabei ist es weniger wichtig, die Berechtigung eines erkannten Benutzers zuverlässig zu überprüfen. Vielmehr ist das entscheidende Problem, die wahre Identität eines Anfragenden zu ermitteln. Kombinationen von Usernamen und Passwort lassen sich allzu leicht stehlen, als dass dies ein zuverlässiger Mechanismus zur Bestätigung der Identität des Anfragenden sein kann.

IP-Netzwerke lassen sich leicht abhören

Das Stehlen solcher Zugangsdaten ist besonders einfach, wenn solche Daten ohne nennenswerten Schutz direkt über das Internet geschickt werden. Ein solches Vorgehen ist genauso sicher, wie die Angaben als Kleinanzeige in einer Zeitung zu veröffentlichen, da die Internetstrecken öffentlich sind und – zumindest rein theoretisch – über jeden beliebigen Rechner geleitet werden können, der gerade online ist.

Auch die überall gerne verwendeten SSL-Verbindungen, die an dem URL-Präfix *https://* zu erkennen sind, sind keineswegs wirklich sicher, auch wenn dort das Abhören der Leitung kaum zum Erfolg führt. Hier führen andere Mechanismen zum Ziel, zum Beispiel ein »Potemkinsches Dorf«, das eine Abwandlung der bekannten Trojaner-Viren ist. Dazu bietet jemand einen Dienst an, für den er eine Registrierung in Form eines frei wählbaren Usernamens und eines Passworts verlangt. Bei der großen Zahl an Registrierungen, die ein durchschnittlicher Internetuser pflegt, ist es jedoch kaum realistisch, für jeden Dienst einen anderen Usernamen und ein neues Passwort zu vergeben. Damit kann der Anbieter eines solchen Dienstes in der Regel zu Recht annehmen, dass der neue Client die gleichen Anmeldedaten auch für andere Dienste nutzt, zum Beispiel, um sich bei Yahoo!-Shopping anzumelden. Es bedarf wenig krimineller Phantasie, um sich die Missbrauchsmöglichkeiten vorzustellen.

Challenge Authenticate

Die Beispiele machen deutlich, wie wichtig es ist, die wahre Identität des Anmeldenden zu authentifizieren. Um dies zu erreichen, fordert der Server vom Client glaubwürdige Beweise für die behauptete Identität. Diesen Vorgang nennt man im englischen Fachjargon einen *Challenge*.

Cerberus

Als Beispiel soll hier eine Strategie zur Sprache kommen, die bereits in dem legendären Computerspiel »Leisure Suit Larry I« zur Verifikation des behaupteten Alters des Spielers verwendet wurde. Dieses Verfahren ist nach dem griechischen Wächter der Unterwelt, dem Höllenhund Cerberus, benannt. Hierzu muss dem Computer eine Liste von Fragen und Antworten bekannt sein, die unter normalen Umständen nur die berechtigte Person selbst beantworten kann. In einem einfachen Fall wird hierzu die Biographie des Benutzers abgespeichert und der Cerberus stellt nun eine oder mehrere Fragen aus dem Repertoire, etwa nach dem Geburtstag der Mutter, dem Beruf des Großvaters oder dem Namen des Arbeitgebers vor exakt zehn Jahren.

2.0.3 Message-Queues

In den Zeiten der alten Mainframes, als Programme vor allem im Batch-Betrieb ausgeführt wurden, waren Message-Queues etwas ganz Selbstverständliches. Jeder Job musste sich in eine Queue einreihen und auf seine Abarbeitung warten, bis er an der Reihe war. Die Eingabe- und Ausgabeströme wurden in speziellen Input- und Output-Queues zwischengespeichert.

MQs haben eigene Intelligenz

Ausgehend von diesen einfachen Verwaltungsqueues stellt man heute allen Applikationen zentrale Message-Queues zur Verfügung, die nicht nur Nachrichten speichern und verwalten, sondern selbst eigene Intelligenz besitzen, um Verarbeitungen anzustoßen, auf Fehler zu reagieren oder ganze Flüsse von Verarbeitungen (Workflows) zu kontrollieren.

Beispiel

Das sieht zum Beispiel wie folgt aus: Falls eine Anfrage nicht sofort zustellbar ist, weil ein geeigneter Computer zeitweise nicht zur Verfügung steht oder überlastet ist, kann die Anfrage vorübergehend zwischengespeichert und zu gegebener Zeit an den Empfänger zugestellt werden. Falls der anfragende Client in der Anfrage zu verstehen gibt, dass er nur eine komplette Antwort wünscht, kann der Message-Server für die Anfrage ein Postfach einrichten, dort die Ergebnisdaten sammeln und, wenn das Ergebnis komplett ist, den Client darüber benachrichtigen.

2.0.4 Workflow

Aufbauend auf einer funktionierenden Message-Queue entwickeln sich Workflows. Darunter versteht man ganz allgemein die automatische Abfolge von regelmäßig wiederkehrenden Prozessen. Im Normalfall handelt es sich dabei um ein einfaches Verketten von Programmaufrufen, wie es bei klassischen Jobs auch der Fall ist.

So meldet z. B. Programm A, dass ein bestimmtes Ereignis stattgefunden hat, zum Beispiel, dass es schlicht zu Ende ist. Der Workflow-Manager fängt dieses Ereignis ab und bestimmt anhand weiterer Daten, die Programm A hinterlassen hat, welches Programm als Nächstes auszuführen ist.

Intelligente Verknüpfungen

Damit sind nicht nur stupide Programmverkettungen gemeint, sondern auch intelligente Verknüpfungen, das soll heißen, dass der Workflow-Manager die Daten des ersten Programms konvertiert, filtert und gegebenenfalls nach bestimmten Kriterien eine Entscheidung trifft, welches Programm als Nächstes auszuführen ist.

2.0.5 GUI-Server

GUI-Server optimieren den Datenfluss zum Browser

Mit der zunehmenden Nachfrage nach browserorientierten Applikationen öffnet sich auch ein neuer Bedarf an Software, die Inhalte in einer Weise aufbereitet und optimiert, dass dem Nutzer eine einheitliche Präsentation gewährleistet wird. Gleichzeitig erfordert die weite Verzweigung von Internetanwendungen immer die Optimierung des Datenverkehrs zwischen Browser und Server.

Hierfür bieten sich GUI-Server als eine neue Form der Middleware an. Dabei handelt es sich um Proxy-Server, die HTML-Seiten auf der Basis von XML-Dokumenten und abstrakten Modellbeschreibungen aufbauen. Dabei geht man davon aus, dass auf der Ebene der Datenübertragung und Transaktionsverwaltung die abstrakte Sicht auf den Datenfluss bei sehr vielen Anwendungen identisch ist.

Beispiel

Nehmen wir zum Beispiel einen Webshop, die »Hello-World!«-Applikation für Webentwickler. Die Seite eines Webshops besteht meistens aus einer Navigationsleiste, einem Block mit Detaildaten und einer Liste. Detaildaten und Listen bestehen selbst aus Feldern. In Teilen der Felder soll es dann möglich sein, Eingaben zu machen. Vergleichen wir damit eine Lagerverwaltung im Materialwesen, so befinden wir, dass auch hier die Struktur identisch ist. Statt der gekauften Artikel sehen wir die Bestände eines Lagerplatzes, statt der Kundendetaildaten sehen wir die Information zum Lagerplatz.

2.0.6 3-Tier-Lösungen

Der Einsatz einer Message-Queue bedeutet in der Praxis zusätzliche Software und somit zusätzliche Kosten für Anschaffung und Wartung. Es stellt sich also die Frage, warum dennoch eine 3-Tier-Lösung attraktiv sein sollte.

Eine 3-Tier-Lösung, bei der SAP R/3 über einen Middleware-Broker mit einer externen Lösung kommuniziert, erweist sich in der Praxis als deutlich wartungs- und entwicklungsfreundlicher als eine 2-Tier-Lösung. Erwarten sollte man eigentlich das Gegenteil, denn immerhin muss ein zusätzliches Produkt mit gewartet und programmiert werden. Ein Teil der Lösung liegt im »Gewicht« der Verbindung. Darunter versteht man den relativen Aufwand zum Aufbau einer Verbindung und zum Austausch von Daten zwischen einem Client und einem Server.

Heavy und Thin Clients

Dabei spricht man von einem *Heavy Client*, wenn der Aufwand für das Protokoll zum Datenaustausch sehr hoch ist, während ein unmittelbarer Datenaustausch als *Thin Client* (manchmal auch als *Slim Client*) bezeichnet wird. Für die Protokolle sind die Begriffe *Heavy weighted protocol* und *Light weighted protocol* gängige Termini.

Payload und Envelope

Als Anmerkung seien noch die beiden Begriffe *Envelope* und *Payload* angeführt, die bei protokollbasierten Kommunikationen gerne verwendet werden. Unter *Payload* versteht man die durch das Protokoll tatsächlich übertragene Information, also die reine Netto-Information ohne die Ver-

waltungsdaten, die zur sicheren Übertragung der Information mitgesandt werden. Der *Envelope*, auf Deutsch Umschlag, ist die Information, die um eine Nachricht herumgepackt wird, um bei der Übertragung den Paketen eine einheitliche Form zu geben.

Zwischen SAP R/3 und DCOM besteht eine schwere Verbindung, weil RFC und OLE/DCOM zwei völlig verschiedene Protokollarten sind. Hinzu kommt, dass SAP R/3 in aller Regel auf einer Nicht-Windows-Plattform läuft. Aber auch eine Windows NT/2000-Version von SAP R/3 wird gewöhnlich betriebssystemneutral betrachtet, das heißt, dass alle Zugriffe von SAP R/3 nach draußen über die R/3-API-Schnittstellen erfolgen.

Ein HTTP-Server auf einer Windows-Plattform basiert selbst auf dem COM-Protokoll. Damit laufen alle Applikationen, die vom HTTP-Server in seinem eigenen Kontext gestartet werden, als COM+- beziehungsweise .NET-Instanzen, womit der Protokoll-Overhead zwischen Client und Server wegfällt beziehungsweise auf dem Niveau von internen Aufrufen bleibt.

Die 3-Tier-Architektur ermöglicht es nun, die Heavy-Client-Verbindung von und nach SAP R/3 zu eliminieren, indem zur Kommunikation mit SAP R/3 ein HTTP-Server zwischengeschaltet wird. Die Kommunikation zwischen HTTP und SAP R/3 ist eine Light-Weight-Verbindung, denn HTTP ist ein ganz einfaches textbasiertes Protokoll, das über TCP/IP ausgetauscht wird, dem Netzwerk-Transport-Protokoll, das auch SAP R/3 für die eigene Kommunikation verwendet. Durch TCP/IP entsteht demnach kein zusätzlicher Overhead für die Kommunikation, während die HTTP-Textnachrichten sehr leicht in SAP R/3 aufgebaut werden können, ohne dass der Entwickler auf R/3-Seite besondere Kenntnisse haben muss.

Message-Server wirkt als Katalysator

Der zwischengeschaltete Message-Server hat somit eine katalytische Wirkung, indem er den Aufwand der Kommunikation zwischen den Partnern im Vergleich zur Direktverbindung deutlich reduziert.

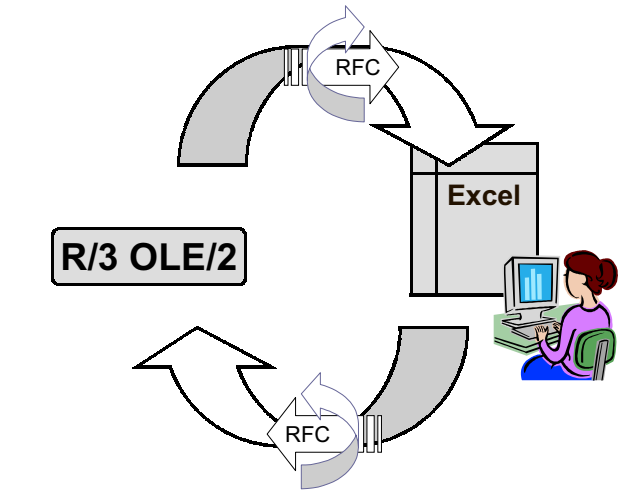

Abbildung 2.1 2-Tier-Kommunikation zwischen Excel und SAP R/3

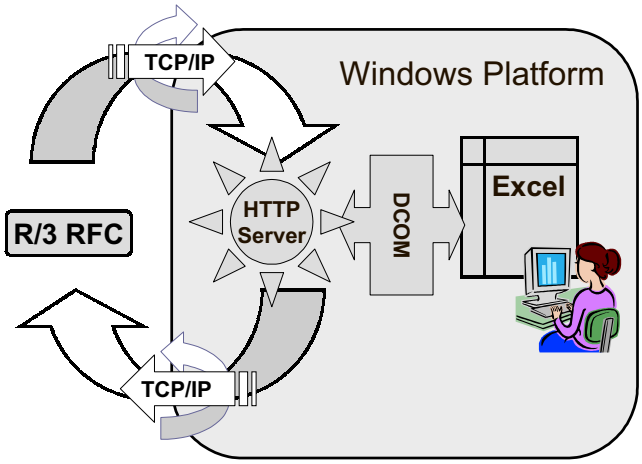

Abbildung 2.2 3-Tier-Kommunikation zwischen Excel und SAP R/3