Das OSF Distributed Computing Environment

Grundlagen und Anwendung

Springer
Berlin
Heidelberg
New York
Barcelona
Budapest
Hongkong
London
Mailand
Paris
Santa Clara
Singapur
Tokio

Alexander Schill

Das OSF Distributed Computing Environment

Grundlagen und Anwendung

2., überarbeitete
und erweiterte Auflage

Mit 128 Abbildungen

Springer

Prof. Dr. Alexander Schill
Technische Universität Dresden
Fakultät Informatik
D-01062 Dresden

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Schill, Alexander:
Das OSF-distributed-computing-Environment : Einführung und Grundlagen / A. Schill. - 2., überarb. und erw. Aufl. -
Berlin ; Heidelberg ; New York ; Barcelona ; Budapest ;
Hongkong ; London ; Mailand ; Paris ; Santa Clara ; Singapur ;
Tokio : Springer, 1997
1. Aufl. u.d.T.: Schill, Alexander: DCE - das OSF-distributed-computing-Environment
ISBN-13:978-3-642-64531-0 e-ISBN-13:978-3-642-60731-8
DOI: 10.1007/978-3-642-60731-8

ISBN-13:978-3-642-64531-0 Springer-Verlag Berlin Heidelberg New York

Satz: Reproduktionsfertige Vorlage vom Autor
Umschlaggestaltung: Künkel + Lopka, Heidelberg
SPIN: 10558615 33/3142-5 4 3 2 1 0 – Gedruckt auf säurefreiem Papier

Vorwort

Das *Distributed Computing Environment (DCE)* der *Open Software Foundation (OSF)* stellt eine Reihe von Softwarekomponenten bereit, um die Erstellung verteilter Anwendungsprogramme auf heterogenen Rechnernetzen zu erleichtern. Dieses Buch hat zum Ziel, dem Leser einen detaillierten, praxisnahen Überblick über das OSF DCE zu vermitteln und ihm dabei konkrete Hilfestellungen beim Einsatz dieser Systemumgebung zur Entwicklung verteilter Anwendungen zu geben. Das Buch wendet sich zum einen an Fachleute auf dem Gebiet der Informatik und Telematik, die sich besonders für den Entwurf und die Realisierung verteilter Anwendungen interessieren, also etwa an Projektleiter, Software-Ingenieure und spezialisierte Programmierer. Zum anderen richtet sich das Buch aber auch an das technische Management, das besonders an einer gut überschaubaren und anschaulich gestalteten Einführung interessiert ist, um in kurzer Zeit eine Bewertung komplexer Systemkonzepte im Hinblick auf die eigenen Anforderungen vornehmen zu können. Im Bereich der Lehre möchte das Buch Dozenten und Studenten der praktischen Informatik ansprechen, die sich mit verteilten Rechnersystemen befassen und das DCE als Beispiel für eine dedizierte Systemarchitektur näher betrachten wollen.

Das Buch gibt zunächst einen Überblick über Eigenschaften, Vorteile und Probleme verteilter Anwendungen. Daraus wird der Bedarf nach weitergehender Systemunterstützung bei der Anwendungsentwicklung abgeleitet, wie sie vom OSF DCE geboten wird. Anschließend wird die Gesamtarchitektur des DCE vorgestellt und aufgezeigt, wie das System für eine konkrete industrielle Anwendung eingesetzt werden kann.

Die nachfolgenden Schwerpunktkapitel befassen sich dann anhand dieser Anwendung mit den Teilkomponenten des DCE, ihrer Konzepte und Funktionalität, ihrer Programmierschnittstelle und ihrer internen Realisierung. Zunächst wird ausführlich die Entwicklung von Anwendungen mit dem Remote Procedure Call (RPC) des DCE, auch unter Verwendung nebenläufiger Prozesse und einer verteilten Namensverwaltung, beschrieben. Zum RPC werden auch die Ergebnisse erster Leistungsanalysen am realen System diskutiert. Anschließend werden die Detailaspekte der wichtigsten unterstützenden DCE-Komponenten (insbesondere des Directory Service, des Security Service und des Distributed Time Service) vorgestellt. Danach wird die verteilte Dateiverwaltung mit dem DCE Distributed File System beschrieben und aufgezeigt, wie auch plattenlose Workstations und PCs in eine DCE-Umgebung eingebettet werden können. Das Buch schließt mit einer Bewertung der DCE-Konzepte und zeigt aktuelle Weiterentwicklungsmöglichkeiten, z.B.

in Richtung der verteilten objektorientierten Programmierung, sowie Bezüge zu Standards auf, beisplielsweise ISO/OSI, Open Distributed Processing, der Object Management Group und Unix International.

Eine besondere praktische Bedeutung hat das DCE vor allem dadurch, daß es herstellerunabhängig einsetzbar ist und deshalb die Kommunikation und Datenverwaltung in heterogenen offenen Systemen ermöglicht. Außerdem bietet es sehr viel mehr als reine Kommunikationsmechanismen an; bereits oben wurden Namensverwaltung, Sicherheit und Dateiverwaltung genannt. Nicht zuletzt ist das DCE heute eines der wenigen verteilten Systeme mit dieser Funktionalität, die in Produktqualität verfügbar und damit auch kommerziell breit einsetzbar sind.

Bei der Darstellung wird großer Wert auf konzeptionelle Klarheit gelegt; aus diesem Grunde werden nicht nur die einzelnen Komponenten des OSF DCE detailliert vorgestellt, sondern es werden auch die zugrundeliegenden Konzepte beschrieben. Obwohl häufig Programmierbeispiele zur praktischen Illustration verwendet werden, kann das Buch aber nicht Programmierhandbücher zum OSF DCE vollständig ersetzen, sondern versteht sich als tutorielle Grundlage, um beim Leser die Basis für das entsprechende weiterführende Material zu schaffen. Allerdings werden dar die wichtigsten ausgewählten Bereiche der Anwendungsentwicklung mit dem DCE in ihrer Gesamtheit dargestellr, so daß Hintergrundliteratur nur für spezielle Aspekte und nicht für grundsätzliche Entwicklungsschritte herangezogen werden muß.

Bei der Beschreibung der Programmbeispiele wurde Wert auf eine verständliche, kompakte Darstellung gelegt. Dies bedeutet, daß die Kernfunktionalität jeweils vollständig vorgestellt wird, daß jedoch auf andere, in der Praxis wichtige, aber zum Grundverständnis unwesentliche Details wie etwa eine vollständige Speicherverwaltung oder Fehlerbehandlung verzichtet wird. Ebenso werden nicht alle DCE-Funktionen ausführlich dargestellt (dies würde ein Vielfaches des Buchumfangs erfordern), sondern es erfolgt eine pragmatische Fokussierung auf die aus Sicht des Autors wesentlichsten Funktionen für gängige verteilte Anwendungen. Die Programmbeispiele wurden alle in einer realen DCE-Umgebung getestet (z.T. nach Vervollständigung um allgemeine Variablen- und Typdefinitionen, die aus Platzgründen im Buch an einigen Stellen weggelassen werden müssen). Für die Programmkommentare wurde der einfachen Darstellung halber die in C++ übliche Notation ("//...") verwendet, die allerdings nicht von üblichen C-Compilern akzeptiert wird. Für praktische Arbeiten mit den Programmstücken sind die Kommentare also entsprechend anzupassen.

Grundsätzlich beruhen die qualitativen Aussagen auf Erfahrungen mit realen Anwendungsimplementierungen, die in der eigenen Arbeitsgruppe durchgeführt wurden. Bei der verwendeten Version des OSF DCE handelt es sich um die *Revision 1.0*, die uns in der vorliegenden Form etwa Mitte 1992 verfügbar wurde. Da sich Funktionsschnittstellen, Konstantendefinitionen, Include-Files usw. im Rahmen von Folgeversionen sicherlich leicht ändern werden, kann die Funktionstüchtigkeit der beschriebenen Programmbeispiele beim Einsatz anderer DCE-Versionen beeinträchtigt werden; hierauf sei an dieser Stelle ausdrücklich hingewiesen.

Die Programmbeispiele richten sich vorwiegend an den Anwendungsentwickler. Auf den Bereich der DCE-Systemverwaltung wird aus Platzgründen meist nur in

kompakter Form auf konzeptioneller Ebene eingegangen; wo es sinnvoll scheint, wird dies durch einige Beispiele zum Einsatz von DCE-Dienstprogrammen ergänzt. Zur Erlernung der DCE-Systemverwaltung im Detail ist in jedem Fall auf die sehr umfangreiche Dokumentation des DCE Administration Guide zurückzugreifen [OSF5].

Die Darstellung der technischen Eigenschaften und der Programmierschnittstellen des DCE basiert auf der umfangreichen, aktuellen DCE-Dokumentation [OSF1-OSF6] sowie weiterführenden Quellen [LOC94, PET95, ROT93, SHM94]. Teilweise wird auf Beispiele und Konzepte aus dem vom Autor mit veröffentlichten Lehrbuch [MÜS92] zurückgegriffen. Hinsichtlich der einzelnen DCE-Komponenten wird auch auf wissenschaftliche Veröffentlichungen zu den zugrundeliegenden Systemen Bezug genommen, um die Grundkonzepte und ihre Entwicklung möglichst umfassend zu erläutern.

Das Buch entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter und Hochschulassistent am Institut für Telematik der Universität Karlsruhe. An dieser Stelle möchte ich dem Institutsleiter, Herrn Prof. Krüger ganz herzlich dafür danken, daß er dieses Buchprojekt durch die Gewährung entsprechender Freiräume bei meiner wissenschaftlichen Tätigkeit ermöglicht hat. Dem Springer-Verlag, namentlich Herrn Rossbach, sei für die Unterstützung bei der Vorbereitung des Buches ausdrücklich gedankt. Auch meinen Kollegen, namentlich Herrn Merz und Herrn Zeidler, möchte ich für ihre Hilfe beim Einsatz des DCE und vor allem bei der Durchführung des DCE-Systemmanagements meinen Dank aussprechen. Herrn Nonnenmacher danke ich für seinen Einsatz beim Test aller beschriebenen Programmstücke und bei der Durchführung von Leistungsmessungen sowie auch für wichtige inhaltliche Anmerkungen ganz herzlich. Dem Campusnahen Forschungszentrum (CEC) der Firma DEC in Karlsruhe, speziell Herrn Dr. Heuser, danke ich ganz besonders für die frühe Verfügbarkeit der Prototyp-Version des DCE. Herrn Stransky von der iXOS Software GmbH gebührt mein umfassender Dank für die Bereitstellung aktueller Informationen zum DCE sowie vor allem für die fachkundige Korrektur des Manuskripts. Nicht zuletzt gilt mein Dank natürlich auch meiner Ehefrau Ursula, deren Unterstützung ich am meisten zu schätzen wußte.

Karlsruhe, im Februar 1993 Alexander Schill

Vorwort zur zweiten Auflage

Die erste Auflage dieses Buches stieß auf erfreuliches Interesse bei Anwendern und Forschungseinrichtungen, was nun eine Neuauflage erforderlich macht. Dies gibt auch die Gelegenheit, neue Erfahrungen aus eigenen Vorlesungen, Seminaren und Praktika zu DCE mit einfließen zu lassen und aktuelle Entwicklungen zu berücksichtigen.

Mittlerweile konnte sich DCE weiter zu einem wichtigen De-facto-Standard für die verteilte Verarbeitung etablieren und erfuhr auch einige wesentliche Funktionalitätserweiterungen, auf die genauer eingegangen wird. Beispiele sind die Integration asymmetrischer Kryptoverfahren in den DCE Security Service und die Bereitstellung objektorientierter Programmierschnittstellen für DCE auf C++-Basis. Auch einschlägige Produkte zur verteilten Transaktionsverarbeitung auf DCE-Basis wie etwa Encina werden aufgrund ihrer besonderen Bedeutung für kommerzielle DCE-Anwendungen nun ausführlicher behandelt.

Mit OMG CORBA (Common Object Request Broker Architecture) existiert ferner eine mit DCE vergleichbare Lösung, die in den letzten Jahren stark an Bedeutung gewonnen hat. Aus diesem Grunde wird CORBA nun ebenfalls mehr Platz eingeräumt, um die wichtigsten Eigenschaften genauer darzustellen und einen konkreten Vergleich mit DCE zu ziehen.

An dieser Stelle möchte ich die Gelegenheit nutzen, allen Kollegen recht herzlich zu danken, die mit zum Erscheinen dieser Neuauflage beitrugen. Zunächst gilt mein ganz besonderer Dank Herrn Thoralf Reichelt, der die Erstellung und Überprüfung von Beispielen, die Informationsrecherche und die Aufbereitung des Textes unermüdlich und mit ganz außergewöhnlichem Engagement unterstützte. Ferner möchte ich Herrn Sascha Kümmel ganz herzlich für seine umfangreichen Arbeiten zur Leistungsanalyse von DCE danken. Herr Dr. Christian Mittasch und Herr Wolfgang König befaßten sich intensiv mit der Entwicklung von DCE-Anwendungen und mit der Weiterentwicklung unserer objektorientierten DCE-Erweiterung DC++, wofür Ihnen besonderer Dank gebührt. Ebenfalls möchte ich allen Mitarbeitern meiner Dresdner Forschungsgruppe für ihr großes Engagement und ihre ausgesprochen kollegiale Arbeitsweise ausdrücklich danken. Nicht zuletzt gilt mein Dank natürlich auch meiner Ehefrau Ursula, die große Geduld mit meinen außerplanmäßigen Aktivitäten wie etwa dieser Neuauflage hatte.

Dresden, im März 1997 Alexander Schill

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung

1.1 Einleitung

Verteilte Rechneranwendungen gewinnen zunehmend an Bedeutung in den verschiedensten Bereichen. Sie lassen sich beschreiben als Programme aus verschiedenen Modulen, die auf unterschiedlichen Rechnern plaziert sind und während ihrer Verarbeitung über ein Kommunikationsnetz interagieren [MÜS92]. Die einzelnen Module arbeiten weitgehend autonom und unterliegen insbesondere keiner zentralen Kontrolle.

Konkrete Beispiele sind verteilte Anwendungen zur Automatisierung von Bürovorgängen, zur Steuerung von Fertigungsabläufen oder zur Verwaltung von Management-Information. Diese Bereiche sind gekennzeichnet durch eine inhärente physische Verteilung ihrer Datenhaltungs- und Verarbeitungskomponenten; z.B. sind Fertigungsvorgänge auf verschiedene Maschinen verteilt, die oft dezentral kontrolliert werden.

Die Kommunikationsmechanismen sowie eine Vielzahl weiterer Konzepte der verteilten Programmierung werden durch ein *verteiltes System* unterstützt [SLK87, COD93, MUL89]; dieses setzt sich aus der Menge der beteiligten Rechnerknoten, dem Kommunikationsnetz sowie der systemnahen Software zur Realisierung der verteilten Verarbeitung zusammen. Generell läßt sich sagen, daß die Entwicklung verteilter Anwendungen umso einfacher wird, je mächtiger die Mechanismen des verteilten Systems sind.

Das *Distributed Computing Environment (DCE)* der *Open Software Foundation (OSF)* stellt ein solches verteiltes System dar, das relativ umfangreiche unterstützende Softwaremechanismen bietet. Ziel dieses Buches ist es, diese Softwareumgebung detailliert vorzustellen und dem Leser die Bewertung und den praktischen Einsatz ihrer Komponenten zu ermöglichen. Angesprochen werden dabei insbesondere Anwendungsentwickler und Endbenutzer des OSF DCE, sowie technische Manager, die mit der DCE-Umgebung in Berührung kommen. Für alle im Buch beschriebenen Systembereiche und Komponenten des DCE wird auch eine allgemeine Übersicht und Einführung mit Darstellung der Grundkonzepte gegeben, was für das Verständnis ohne spezielle Hintergrundliteratur von Bedeutung ist.

Im Februar 1996 fusionierten die OSF und die X/OPEN Company Ltd. zur *Open Group*, um gemeinsam die Entwicklung von verteilten Systemen fortzuführen.

A. Schill, *Das OSF Distributed Computing Environment*

1.2 Beispiel einer verteilten Anwendung

Abb. 1-1 zeigt ein Beispiel einer verteilten Anwendung aus dem Gesamtbereich der Büro- und Fertigungsautomatisierung.

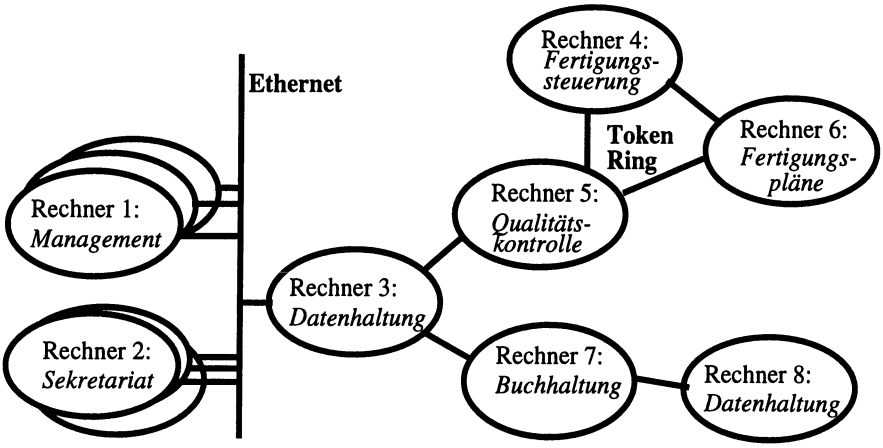

Abb. 1-1 Beispiel einer verteilten Anwendung

Struktur des verteilten Systems
Die Anwendung ist auf mehrere Rechner verteilt, die über ein Ethernet, einen Token Ring sowie über mehrere Punkt-zu-Punkt-Leitungen gekoppelt sind. Einige Rechner agieren dabei u.a. als Gateway, etwa Rechner 3, 5 und 7. Auf der Ebene des OSF DCE ist jedoch die Netzstruktur transparent; es wird grundsätzlich davon ausgegangen, daß die Rechner logisch voll vermascht sind, auch wenn das physisch nicht der Fall ist. Dadurch kann potentiell jeder Rechner Nachrichten an jeden anderen senden. Diese Funktionalität wird durch transportorientierte Protokolle wie z.B. TCP/IP erbracht, die vom DCE als existierend vorausgesetzt werden.

Abläufe innerhalb der Anwendung
Die einzelnen Komponenten der Anwendung sind nun - wie gezeigt - verteilt auf verschiedenen Rechnern plaziert. Typische verteilte Verarbeitungsabläufe haben etwa folgende Form:

Projektplanung (Abb. 1-2)
Ein Manager (auf Rechner 1) beschafft sich Daten zu früheren Produkt-Entwicklungsprojekten von der entfernten Komponente zur Datenhaltung (auf Rechner 3), etwa durch Aufruf einer entfernten Datenbankoperation. Danach werden lokale Planungsaufgaben unter Einsatz eines Spreadsheet-Programms durch den Manager durchgeführt. Gegebenenfalls sendet der Manager im Verlauf dessen Informations- oder Anfragenachrichten an andere Manager auf anderen Rechnern; auf diese Weise können z.B. Koordinationsfragen entfernt geklärt werden. Nach Abschluß einer

groben Produktplanungsphase sendet der Manager die resultierenden Daten zur weiteren Überarbeitung an seine Sekretärin (Rechner 2), indem er eine entfernte Speicherprozedur dort aufruft. Danach folgen zahlreiche Interaktionen mit anderen Abteilungen zur Feinplanung, z.B. mit der technischen Entwicklungsabteilung, die hier nicht gezeigt sind.

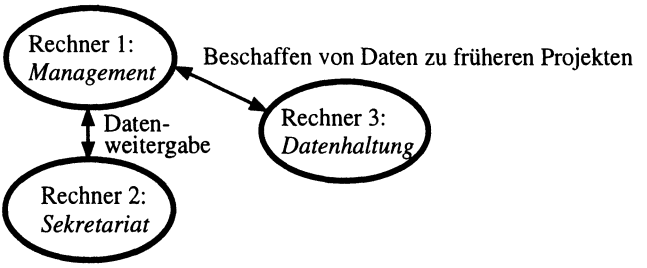

Abb. 1-2 Grobablauf bei der Projektplanung

Fertigung (Abb. 1-3)
Im Rahmen der Fertigung von Produkten wird die Fertigungssteuerung (Rechner 4) mit der Koordination beauftragt. Je nach Fertigungsauftrag beschafft sie sich einen zugehörigen Fertigungsplan von Rechner 6, der eine Datenbasis von Fertigungsplänen verwaltet. Anschließend werden ggf. mehrere Maschinensteuerungen mit den einzelnen Fertigungsschritten beauftragt, die wiederum auf verschiedene Rechner verteilt sein können (in der Abbildung nicht gezeigt). Schließlich nimmt die Fertigungssteuerung die Fertigmeldungen der Maschinen entgegen und beauftragt die Qualitätskontrolle (Rechner 5) mit der Abschlußprüfung des gefertigten Produkts.

Buchhaltung (Abb. 1-4)
Die Buchhaltung (Rechner 7) ist zunächst recht konventionell als eine zentrale Instanz realisiert. Die von ihr verwendeten Daten (Rechnungen, Lieferscheine etc.) werden aber z.B. wegen ihres großen Umfangs durch eine separate, entfernte Datenbank auf Rechner 8 verwaltet. Es ist daher erforderlich, dort entfernte Operationen zum Anfordern und Abspeichern von Daten aufzurufen. Außerdem greift die Buchhaltung teilweise auch auf die Dienste der Datenhaltung auf Rechner 3 zu, um z.B. bei Bedarf eine Zuordnung zwischen Rechnungstiteln und Produktbeschreibungen durchführen zu können.

Globale Interaktionen
Grundsätzlich sind neben diesen dedizierten Abläufen auch globale Interaktionen möglich, in die ggf. sehr viele Rechner involviert sind. Als Beispiel sei die Analyse eines Produkts durch einen Manager in bezug auf seine Ausschußrate bei der Fertgung und die erzielte Gewinnspanne genannt.

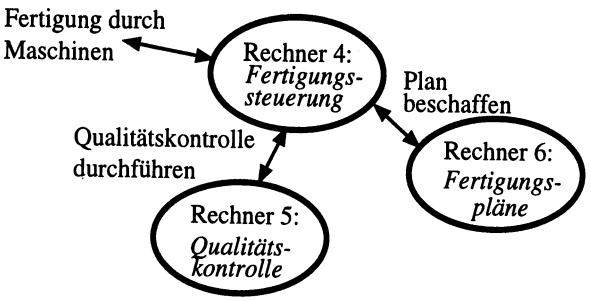

Abb. 1-3 Grobablauf bei der Fertigung

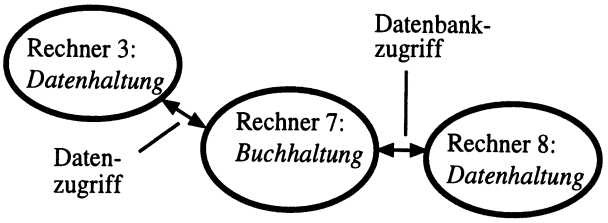

Abb. 1-4 Grobablauf bei der Buchhaltung

Wichtige Eigenschaften

Am Beispiel wurde aufgezeigt, daß eine bestimmte Plazierung von Anwendungskomponenten auf Rechnerknoten vorgenommen wird. Es ist auch offensichtlich, daß geeignete Kommunikationsmechanismen zwischen den Komponenten bereitstehen müssen.

Außerdem ist schon hier die weitergehende Entkopplung der verschiedenen Abläufe zu erkennen, die dadurch auch parallel ausgeführt werden können. Ebenfalls ist zu sehen. daß manche Dienste bzw. Betriebsmittel (z.B. die Projektdatenbank) von mehreren Komponenten verwendet werden und dadurch eine verbesserte Integration erzielt wird. Andererseits kann die Verwaltung komponentenspezifischer Daten (z.B. des Managements) stärker dezentralisiert und damit leichter durch die jeweilige Komponente kontrolliert werden. Diese und einige weitere Eigenschaften werden im folgenden vertieft.

1.3 Vorteile verteilter Anwendungen

Die Entwicklung verteilter Anwendungen ist - auch mit einer Systemumgebung wie dem OSF DCE - generell durch eine erhöhte Komplexität gekennzeichnet; daher stellt sich zunächst die Frage nach den Vorteilen solcher Ansätze, um den entsprechenden Entwicklungsaufwand technisch und wirtschaftlich zu rechtfertigen:

- *Dezentralisierung und Lokalität:* Durch die Verteilung der Module wird eine weitgehend dezentrale Verarbeitung möglich. Dadurch kann eine hohe Lokalität in bezug auf die Zuordnung von Daten und bearbeitenden Operationen erzielt werden. Dies wiederum verbessert die Laufzeiteigenschaften und erleichtert die organisatorische Verwaltung.
- *Parallele Verarbeitung:* Da jeder beteiligte Rechner eigene Betriebsmittel, insbesondere Prozessor und Speicher, bereitstellt, können die einzelnen Module stark parallel arbeiten; dadurch wird die Gesamtbearbeitungszeit eines Auftrags verkürzt.
- *Fehlertoleranz:* Die beteiligten Rechner sind weitgehend autonom; dies gilt auch für Systemausfälle; bei Ausfall eines Rechners bleibt der übrige Teil der Anwendung auf anderen Rechnern funktionsfähig. Explizite Fehlertoleranz kann zusätzlich durch Replikation von Daten und Operationen erzielt werden.
- *Gemeinsame Betriebsmittelnutzung:* Da verschiedene Rechner über ein Kommunikationsnetz gekoppelt sind, können sie auch gemeinsam auf die Ressourcen eines bestimmten Rechners zugreifen. Dadurch wird die gemeinsame Nutzung teurer Peripheriegeräte (z.B. Drucker, Plotter oder Speichermedien) sowie komplexer Softwarekomponenten (z.B. Datenbanken) ermöglicht.
- *Integration von Teilanwendungen:* Die Gesamtfunktionalität existierender Teilanwendungen kann oft durch ihre Integration zu einer verteilten Anwendung gesteigert werden; beispielsweise kann eine Management-Datenbank mit einem Spreadsheet-System gekoppelt werden, um automatisierte Datenauswertungen durchzuführen.

Gerade die beiden letztgenannten Punkte stellen in sehr vielen Fällen die wichtigste Motivation für den Einsatz verteilter Systemtechnologie in der industriellen Praxis dar. Sie lassen sich vergleichsweise einfach erreichen und können sehr rasch zu einer realen Kostenreduzierung führen, die meist auch relativ genau gegenüber dem zu erwartenden Entwicklungsaufwand abgeschätzt werden kann. Die drei anderen Punkte erfordern dagegen meist eine umfangreichere Planung bzw. Umstrukturierung von Anwendungen. Beispielsweise müssen unabhängige Verarbeitungseinheiten identifiziert oder konstruiert werden, um Parallelität echt auszunutzen. Ebenso ist der Einsatz und gegebenenfalls sogar die Entwicklung dedizierter Replikations- und Konsistenzerhaltungstechniken erforderlich, um einen hohen Grad der Fehlertoleranz zu erzielen. Die Ausnutzung von Lokalitätseigenschaften erfordert ebenso grundsätzliche Entwurfsentscheidungen im Rahmen der Anwendungsentwicklung oder -neustrukturierung. Dennoch sind diese Bereiche gerade für große Rechneranwendungen als Mittel zur Steigerung der Verarbeitungskapazität und -qualität von zentraler Bedeutung. Sie weisen meist ein sehr