0. Vorwort

Sowohl in der Kerninformatik als auch in verschiedenen Bereichen der Angewandten Informatik beschäftigt man sich mit einer Klasse von Systemen, die sich (auf einem der verschiedenen möglichen Betrachtungsniveaus) anschaulich wie folgt beschreiben lassen:

interessierende Systemklasse

Das System besteht aus einer Menge von Betriebsmitteln (auch: Ressourcen, Funktionseinheiten, Stationen), die in beliebiger (oder eingeschränkter) Reihenfolge benutzt (auch: beauftragt, belastet) werden können. Die Benutzung der einzelnen Betriebsmittel geschieht im Rahmen der Benutzung (Beauftragung) des Gesamtsystems: Aufträge an das Gesamtsystem sind in mehr oder weniger festgelegter Weise so geartet, daß sie die Benutzung bestimmter Betriebsmittel in bestimmter Reihenfolge erfordern. Bei gleichzeitiger "Anwesenheit" (Bearbeitung) mehrerer Aufträge kann es dabei offensichtlich zu wechselseitigen Behinderungen an solchen Betriebsmitteln kommen, deren Arbeitsfähigkeit nicht beliebig viele gleichzeitig bearbeitete Teilaufträge zuläßt. Es resultieren Wartesituationen vor Betriebsmitteln, insgesamt Verzögerungen im Ablauf der Bearbeitung von Aufträgen.

Beispiele

Leicht lassen sich viele Beispiele solcher Systeme identifizieren: Rechen- und Kommunikationssysteme (Betriebsmittel: Prozessoren, Kanäle, Leitungen, ...; Aufträge: Bearbeitungs- und Kommunikations-"tasks", ...), Fertigungssysteme (Betriebsmittel: Bearbeitungsmaschinen, Transportelemente, ...; Aufträge: Fertigungsaufträge für Produkte), aber auch Bürosysteme, Supermärkte, Reparaturbetriebe, Verwaltungssysteme und viele mehr.

interessierende Eigenschaften

Für solche Systeme besteht ein natürliches Interesse an ihren "quantitativen" Eigenschaften:

- Wie lange dauern Abwicklungen von Aufträgen ?
- Wieviele Aufträge pro Zeiteinheit lassen sich abwickeln?
- Wie ausgelastet sind die einzelnen Betriebsmittel?

und ähnliche mehr. Diese quantitativen Eigenschaften werden in verschiedenen Wissensdisziplinen unter unterschiedlichen Bezeichnungen studiert und analysiert: "Leistungsanalyse" in der Informatik, "Verkehrstheorie" in der Nachrichtentechnik, "flow-shops" (bzw. job-shops) in Operations Research und Maschinenbau, ... Konkret entstehen selbstverständlich Unterschiede zwischen den Disziplinen aufgrund spezifischer Eigenschaften von Betriebsmitteln und Aufträgen im Anwendungsbereich, abstrakt gesehen sind die Analyse-Probleme und Analyse-Techniken aber sehr ähnlich und werden unter Überschriften wie Wartesysteme, Warteschlangennetze, Verkehrssysteme und -Netze u. ä. behandelt.

Wir werden in dieser Vorlesung, wo erforderlich, den für uns nächstliegenden Anwendungsbereich "Rechen- und Kommunikationssysteme" berücksichtigen und uns damit im Gebiet der "Leistungsbewertung von Rechen- und Kommunikations-Systemen" bewegen; die prinzipielle Übertragbarkeit der vorzustellenden Analysemethoden in andere Anwendungsbereiche ist aber gegeben und wird offensichtlich sein.

LEERSEITE

1. Leistungsbewertung von Rechensystemen: Eine Einführung

informationsverarbeitende Systeme:

Die Wissensdisziplin Informatik beschäftigt sich generell mit informationsverarbeitenden Systemen, d.h. mit Systemen zur automatischen Transformation, Speicherung und Übertragung von Daten (bzw. von Information, wenn die Interpretation von Daten zu betonen ist). Bei der "Beschäftigung" mit Rechensystemen lassen sich zwei unterschiedliche Blickrichtungen unterscheiden:

Erstellung

- Das Interesse kann, einerseits, auf die Erstellung eines derartigen Systems gerichtet sein, also auf die Realisierung eines Rechensystems mit Fähigkeiten der oben genannten Art, kurz: mit der Fähigkeit, gewisse (geforderte!) Dienste zu erbringen.

Verwendung

- Das Interesse kann, andererseits, auf der Verwendung eines derartigen Systems liegen, auf der Festlegung der Art und Weise also, wie gewisse (vorhandene!) Dienste eines Rechensystems genutzt werden.

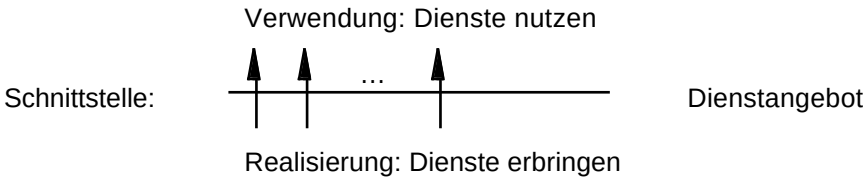

Abbildung 1.1: Aspekte der Verwendung bzw. Realisierung von Rechensystemen mit Dienst-Schnittstelle

Schnittstelle: Dienste

Die Unterscheidung dieser beiden Blickrichtungen bezieht sich offenbar auf eine ganz bestimmte Schnittstelle in Form einer bestimmten Menge zu nutzender/zu erbringender Dienste. Wir sind uns als Informatiker dessen bewußt, daß unterschiedliche derartige Schnittstellen definierbar sind (denken Sie z.B. an das unterschiedliche "Niveau" der Dienste Gleitpunktmultiplikation, Dateitransfer, Flugbuchung) und daß daher Differenzierungen dieser einfachen Vorstellung nötig werden (diese führen zwanglos zu der bekannten "Hierarchie virtueller Maschinen", die wir in diesem Einführungskapitel durchaus noch berücksichtigen werden). Bleiben wir aber zunächst bei der einen Schnittstelle und legen diese dahin, wo "der Benutzer" einen Dienst "beim Rechensystem" anfordert, indem er (wie früher üblich:) einen Lochkartenstapel an einem Schalter abgibt bzw. (heutzutage:) per "Knopfdruck" (Tastatur o.ä.) eine Nachricht abschickt. Machen wir uns auch von dem angeforderten Dienst ein konkreteres Bild, indem wir uns vorstellen, es handle sich bei diesem um eine (in sich abgeschlossene) Aktivität, welche ein Rechensystem zur Lösung eines bestimmten Problems beiträgt (z.B. zur statischen Auslegung eines Gebäudes, zur Jahresbilanz einer Firma, zum Stundenplan einer Schule). Dem einzelnen Benutzer eines Rechensystems bietet sich damit ungefähr der in Abb. 1.2 skizzierte Ablauf zur Problemlösung dar. Dabei ist die Übertragung des "Problems" in Daten (bzw. als Daten bereitgestellte Programme) sowie der als Daten ermittelten Resultate in die "Problemlösung" Aufgabe des menschlichen Benutzers, die Transformation von (Eingabe-)Daten in (Ausgabe-)Daten der Beitrag des Rechensystems zur Gesamtaufgabe.

Prinzip Problemlösung

Anforderungen:

Die Erwartungen/Anforderungen eines Benutzers hinsichtlich eines Rechensystems sind zumindest, daß

Dienste existent

- ihm geeignete Dienste de fakto zugänglich sind, in unserer Beispiels-Vorstellung also etwa Dienste der Art
  EXECUTE <Programm a>
  mit <Daten b>
  und <Ausgabe c>
  einer (Betriebssystem-)Kommando-Ebene;

und korrekt

- er bei korrekter Formulierung der Eingabedaten auf die Korrektheit der Resultate schließen darf.

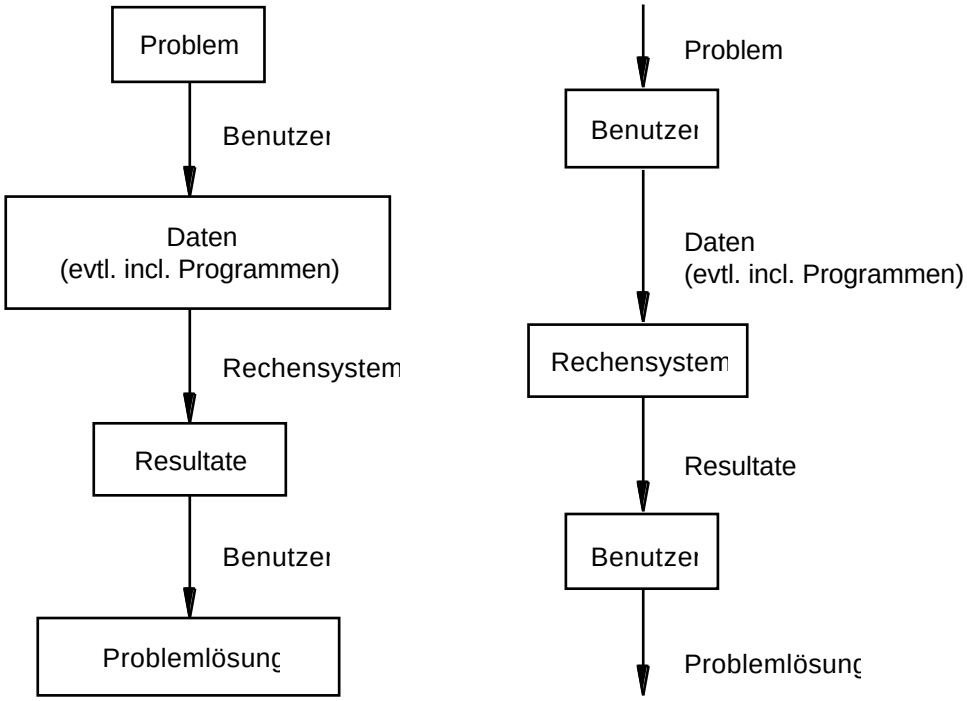

Abbildung 1.2: Vorgang der rechnergestützten Problemlösung; zwei duale Skizzen des Ablaufs

Etwas konkreter als zuvor sind wir damit wieder bei Anforderungen an die Realisierung eines Rechensystems angelangt: eine gewisse geforderte Funktionalität aufzuweisen (d.h. bestimmte Dienste bereitzustellen) und die Gewähr für eine korrekte Ausführung dieser Dienste zu bieten. Funktionalität und Korrektheit sind primäre Anforderungen an die Realisierung von Rechensystemen - der weit überwiegende Teil der Kerninformatik beschäftigt sich mit genau diesen Anforderungen.

Testfrage 1.3: Welche "Dienste" identifizieren Sie bei einer "anderen" Wahl der Schnittstelle, z.B. an der Hardware/Software-Schnittstelle?

weitere Anforderungen

Es gibt aber durchaus weitere legitime Anforderungen an ein Rechensystem (RS). Stellen wir uns vor, wir hätten zum Zwecke der Datentransformation im Rahmen einer bestimmten Problembearbeitung mehrere verschiedene Rechensysteme RSi (i=1,2,…,n) zur Verfügung. Diese mögen (für unsere Zwecke) von gleicher Funktionalität sein, so daß wir sie mit der gleichen Datentransformation

Rechensysteme gleicher Funktionalität:

beauftragen können, wie in Abb. 1.4 skizziert. Sehen wir im Moment davon ab, daß die Aufgabenformulierung (die Eingabedaten) bzgl. der verschiedenen Rechensysteme in der Praxis nur "sinngemäß" gleich sein werden (unterschiedliche Eingabesyntax auch bei gleicher Funktionalität), dann werden bei gleicher Aufgabe und korrekter Realisierung aller RSi letztlich sinngemäß (wieder: abgesehen von Syntaxfragen) von allen die gleichen Resultate geliefert.

gleiche Resultate,

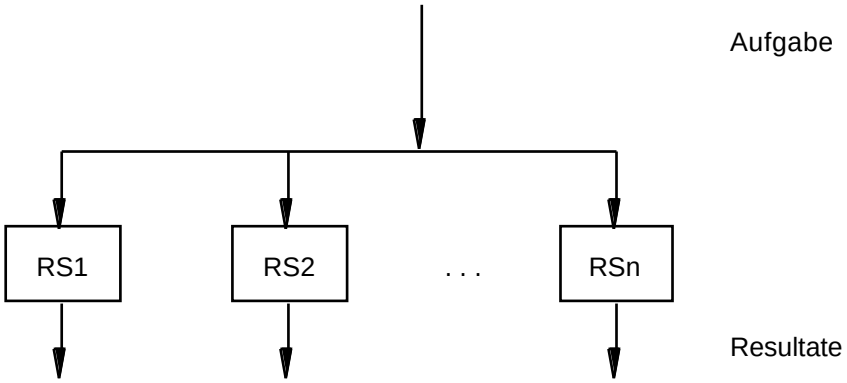

Abbildung 1.4: Gleiche Beauftragung von n Rechensystemen gleicher Funktionalität

Vom Standpunkt der tatsächlichen und korrekten Bearbeitung aus bieten sich dem Benutzer also keine Unterschiede zwischen den RSi dar. Unterschiede anderer Art sind aber sicher vorhanden: So benötigen die RS jeweils spezifische, untereinander i.allg. verschiedene Zeiten für die Bewältigung der (gleichen) Aufgabe. Im Sinne einer Bewertung der verschiedenen RS wird der Benutzer ein "schnelleres" höher einstufen als ein "langsameres"; und zusätzlich zu seinen Anforderungen hinsichtlich Funktionalität und Korrektheit eines RS wird er Anforderungen bezüglich der (maximal tolerierbaren) Bearbeitungszeit für angeforderte Dienste (formulierte Aufgaben) haben.

aber Unterschiede der Abarbeitung

Halten wir fest: Neben Anforderungen hinsichtlich Funktionalität und Korrektheit eines Rechensystems haben wir Anforderungen, die eine quantitative Bewertung der ("Güte" der) Bearbeitung angeforderter Dienste betreffen; im obigen Beispiel war dies die Zeit, welche für die Bearbeitung benötigt wurde. Schematisch gesehen, resultiert (Abb. 1.5) aus einer Bearbeitung neben dem "eigentlichen" Resultat auch die Beobachtung/Bewertung der Bearbeitungsgüte.

Bearbeitungsgüte als Resultat

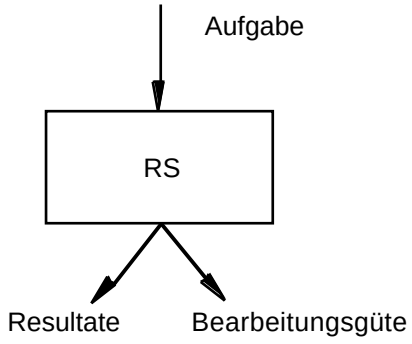

Abbildung 1.5:

Bearbeitungsgüte als zusätzliches Ergebnis einer Bearbeitung

Für diese Vorlesung wird die Bewertung der Bearbeitungsgüte eines RS das zentrale Thema sein: Wir setzen voraus, daß alle Anforderungen an Funktionalität und Korrektheit erfüllt sind, daß also wie oben skizziert die Resultate (Ausgabedaten) jeweils zu betrachtender RS sinngemäß gleich sind und zu einer Unterscheidung nichts beitragen, so daß wir diese Resultate schlichtwegs nicht mehr zu berücksichtigen brauchen, was sich in folgender Skizze niederschlägt:

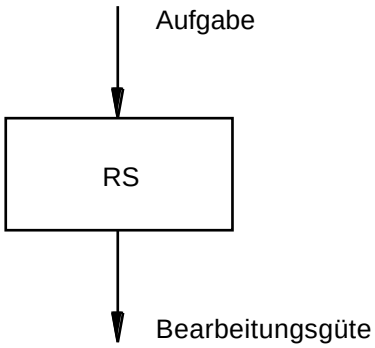

Abbildung 1.6: Bearbeitungsgüte als Bearbeitungsergebnis (Ignorieren der Ausgabedaten)

Leistung: Bearbeitungszeit

Derlei Bewertungen der Bearbeitungsgüte eines RS belegen wir mit der Bezeichnung **Leistung** des RS. In der Beispielsbetrachtung handelte es sich konkret um die Bearbeitungszeit für eine bestimmte Aufgabe. Dies ist allerdings nur einer von vielen möglichen Aspekten der Leistung eines RS im Sinne der quantitativen Bewertung seiner Bearbeitungsgüte: So werden z.B. auf ein Mehrbenutzersystem Aufgaben verschiedener Benutzer zukommen. Bestand oben die "Belastung" des RS aus genau einer Aufgabe, so besteht sie jetzt aus mehreren Aufgaben, deren Bearbeitung zu bestimmten Zeitpunkten gefordert wird:

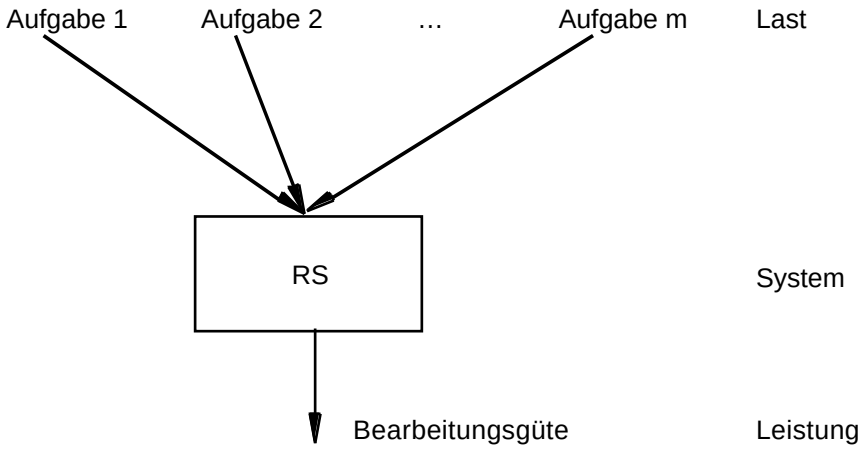

Abbildung 1.7: Belastung eines Mehrbenutzersystems

Last

Für die Belastung eines RS, d.h. die in Auftrag gegebenen Aufgaben (wo erforderlich, inklusive Beauftragungszeitpunkten), führen wir die Bezeichnung **Last**

ein. Bei der Last der Abb. 1.7 drückt sich Bearbeitungsgüte (also: Leistung) für den einzelnen Benutzer nach wie vor durch die Zeit der Bearbeitung seiner Aufgabe aus.

Für den Betreiber des RS (das "Rechenzentrum") tritt (zumindest) ein weiterer Leistungsaspekt hinzu: Er wird ein RS, das mehr Aufgaben pro Zeiteinheit (ZE) bewältigt, höher ("leistungsstärker") einstufen als eines, das weniger Aufgaben pro ZE bearbeitet. Technischer gesprochen: Neben dem Leistungskriterium/Leistungsmaß **Antwortzeit** (Bearbeitungszeit für Aufgabe) existiert zumindest ein weiteres legitimes Leistungskriterium/Leistungsmaß **Durchsatz** (Zahl bearbeiteter Aufgaben pro ZE); letzteres kommt übrigens der in der Physik eingesetzten Bedeutung des Begriffs "Leistung", nämlich "Arbeit/Zeit", recht nahe. Wir werden im Verlauf weitere Leistungskriterien und zugeordnete Leistungsmaße kennenlernen. Halten wir, etwas verallgemeinernd, zunächst fest:

Leistung: Antwortzeit, Durchsatz

Definition 1.8: Die Leistung eines Rechensystems kann anhand einer Reihe unterschiedlicher Kriterien beurteilt werden, die alle auf die Beurteilung der Art/Güte der Bearbeitung einer Last zielen. Die Leistung eines Rechensystems läßt sich durch eine Beobachtung der Arbeit dieses Systems feststellen; **Leistung** ist eine **Antwort** des **Systems** auf die überantwortete **Last**. Zur quantitativen Bewertung eines Leistungskriteriums bedarf es eines zugeordneten Leistungsmasses.

Definition Leistung

Wir sollten uns an dieser Stelle klarmachen, daß der Vorgang einer Bewertung, also die Festlegung eines Gütekriteriums samt Definition eines zugehörigen Gütemaßes, regelmäßig eine stark subjektive Komponente enthält, da ja explizit ausgedrückt wird, was für "wesentlich" gehalten wird (und implizit, was als "unwesentlich" gilt). Dennoch stellt sich in jedem Bereich von Bewertung über kurz oder lang ein gewisser Konsens über wichtige Gütekriterien ein. Im hier diskutierten Bereich der Bewertung der Leistung von Rechensystemen ist es üblich, Kriterien der beiden folgenden Arten einzusetzen:

Typen von Leistungs-Kriterien:

- Effektivitäts-Kriterien/Maße dienen der Feststellung der (extern beobachtbaren) "Wirksamkeit" eines Rechensystems. Zu ihnen zählen die Kriterien der Dauer von Dienstabwicklungen und der Rate von Dienstabwicklungen, also Maße wie Antwort/Reaktions-Zeiten und Durchsätze.

Effektivität

- Effizienz-Kriterien/Maße dienen der Feststellung der (nur intern beobachtbaren) "produktiven Ausnutzung" eines Rechensystems in Bezug auf dessen prinzipielle Fähigkeiten. Zu ihnen zählen die Kriterien der Tätigkeitszeit von Prozessoren und des Ausmaßes unproduktiver Verwaltungsarbeit, also Maße wie Auslastungen und overhead.

Effizienz

Wegen des Ortes der möglichen Beobachtung, wie oben schon genannt, bezeichnet man die Effektivitätsmaße auch als **externe Leistungsmaße**, die Effizienzmaße als **interne Leistungsmaße**. Das Interesse an einem der beiden Typen von Maßen entspricht auch der Konzentration auf eine der beiden eingangs diskutierten Blickrichtungen: So wird der "Verwender" eines RS sich bei einer quantitativen Überlegung auf Effektivitätsmaße abstützen, während der "Ersteller" eines RS bei einer quantitativen Überlegung (die bei ihm auf der Forderung nach einer gewissen Effektivität beruht) insbesondere Effizienzaspekte zu berücksichtigen hat (um nämlich mit "seinem" RS die Effektivitätsanforderungen zu erfüllen). Schließlich, um das Bild abzurunden, sind sowohl Effektivität als auch Effizienz zu diskutieren, wenn ein wesentlicher weiterer Aspekt bedacht wird, näm-

Kosten

lich die Kosten eines RS; es leuchtet ein, daß bei festen Kosten ein effizienteres RS i.allg. auch effektiver sein dürfte und daß bei konstanter Effektivität eine erhöhte Effizienz sich i.allg. in niedrigeren Kosten niederschlagen sollte.

Voraussetzung Funktionalität,

Ignorierung weiterer Kriterien,

Wir haben uns eine Vorstellung über jene Eigenschaften eines RS erarbeitet, die wir mit dem Begriff Leistung (englisch: performance) belegen wollen und insbesondere zwei Typen von Leistungskriterien identifiziert, die mit Effektivität (englisch: effectiveness) und Effizienz (englisch: efficiency) bezeichnet wurden. Wir haben dabei die sicherlich zentrale Eigenschaft der Korrektheit als gegeben vorausgesetzt und bewußt ignoriert. Sei hier zumindest daran erinnert, daß weitere potentiell interessante Eigenschaften von RS existieren, die wir ebenfalls ignoriert haben und auch weiterhin ignorieren werden: So deren Zuverlässigkeit, Akzeptanz, Ergonomie, ihren Stromverbrauch, ihren Platz- und Kühlungsbedarf, etc.. Sie alle können neben den Funktionalitäts- und Leistungs-Eigenschaften zu einer (allgemein verstandenen) "Bewertung" eines RS beitragen.

Konzentration auf Leistung

Notwendige Betrachtung von System und Last

Wir bleiben von nun an bei der alleinigen Berücksichtigung von Leistungseigenschaften, der Beurteilung also (Def. 1.8) der Güte der Bearbeitung einer Last durch ein Rechensystem im Sinne der inzwischen erfolgten Konkretisierungen hinsichtlich Effektivität und Effizienz. Obzwar nun schon verschiedentlich explizit so formuliert, lohnt es sich, deutlich darauf hinzuweisen, daß wir nicht nach der **Leistung eines RS per se** fragen (können!), sondern dabei **immer** eine **bearbeitete Last mit einbeziehen** (müssen!). Wir betrachten also regelmäßig nicht nur das "Dienste-anbietende" Rechensystem, sondern gleichzeitig seine "Dienstefordernde" Umgebung. Im Unterschied übrigens zur üblichen Betrachtungsweise bei isolierter Verfolgung des Funktionalitätsaspekts, wo gewisse Dienste und ihre korrekte Ausführung seitens eines RS zuzusichern sind, ohne auf eine konkrete Benutzung (d.h. Umgebung) Bezug zu nehmen, da ja in der Regel die korrekte Funktionalität (unausgesprochen) für "jede mögliche Benutzung/Umgebung" gefordert ist. Mag die für Leistungsbetrachtungen geforderte Berücksichtigung der Umgebung (Last) zunächst ungewohnt anmuten, so ist sie doch nach kurzem Nachdenken selbstverständlich: Wie sollte man auch z.B. die Antwortzeit für die Ausführung einer bestimmten Aufgabe ermitteln, die einem Mehrbenutzersystem zur Bearbeitung übergeben wird, ohne die "nebenher" abgewickelten anderen Aufgaben zu kennen?

Auftreten von Leistungsforderungen

Wann und wo treten denn nun konkrete Forderungen nach einer bestimmten RS-Leistung (von seiten einer "benutzenden Umgebung") und konkrete Bemühungen um die Sicherstellung geforderter RS-Leistungen (auf seiten gewisser "System-Realisierer") auf? Kurz gesagt: Immer und überall! Und etwas präziser ausgedrückt: In allen Phasen des Lebenszyklus' eines RS bzw. seiner Komponenten, während des Entwurfs, der Realisierung, des Betriebs bestehen (neben anderen Forderungen) auch Leistungsforderungen; in allen (aufeinander aufbauenden) Schichten eines RS bzw. seiner Komponenten, in der Hardware-Schicht, der Firmware-Schicht, der Schicht der System-Software, der Schicht der Anwendungs-Software sind (neben anderen Aspekten) auch Leistungsaspekte zu berücksichtigen. Die Prinzipskizze 1.9 möge dies verdeutllichen: In jedem "Kästchen" der Skizze treten spezifische Leistungsforderungen auf, sind spezifische Leistungsprobleme zu lösen.

Phasen / Schichten	Entwurf	Realisierung	Betrieb
Anwendungs-Software			
System-Software			
Firmware			
Hardware			

Abbildung 1.9: Phasen und Schichten des Auftretens von Leistungsforderungen/Leistungsproblemen

Betroffene Personengruppen

Unsere Erkenntnis (Def. 1.8) wieder aufgreifend, daß Leistung sowohl durch Eigenschaften eines RS als auch durch die Art der Belastung des RS beeinflußt wird, erkennen wir auch, daß viele Personen/Personengruppen am "Zustandekommen" von Leistung beteiligt sind, jeweils aber spezifische Einflußmöglichkeiten besitzen. Zur Verdeutlichung, als grobe Prinzipskizze, Abb. 1.10.

Personenkreis (z.B.)	Einwirkungsmöglichkeiten auf	
	System	Last
Hersteller	Komponenten: Hardware, Firmware, Systemsoftware Konfigurierung von Komponenten	keine (außer: Einsatz-/ Verwendungs-Empfehlung)
Rechenzentrum	Konfiguration Hardware/ Software Systemgenerierung, tuning	Betriebliche Regelungen, pricing
Benutzer	keine (außer: Äußern von Wünschen, Forderungen)	Algorithmen, Portionierung, Laufentscheidungen

Abbildung 1.10: Einwirkungsmöglichkeiten diverser Personenkreise auf System bzw. Last (und damit auf "Leistung")

Testfrage 1.11: Sind Ihnen alle verwendete Begriffe der Abb. 1.10 klar und ihre Einordnung erklärbar? Was ist "tuning", "pricing", "Portionierung"? Welche Rolle spielen "Systemgenerierung", "Betriebliche Regelungen", "Laufentscheidungen"?

Diskussion "Schnittstelle"

Wir hatten uns eine Vorstellung vom Begriff "Leistung" eines Rechensystems auf der Basis einer Schnittstelle (vgl. Abb. 1.1) verschafft, die durch eine Menge ausführbarer Dienste charakterisiert war; an dieser Schnittstelle traf eine (Dienste-anfordernde) Umgebung auf ein (Dienste-realisierendes) System und rief damit einen "Betrieb" hervor (nämlich den Vorgang der Abwicklung der Last durch

das System); zur Beurteilung dieses Betriebs legten wir uns Gütekriterien und zugeordnete Maße (der Typen Effektivität oder Effizienz) zurecht, welche die "Leistung" (dieses Systems unter dieser Last) ausdrückten. Wir hatten gleich zu Beginn festgestellt, daß wir bei der Wahl der Schnittstelle zwischen Umgebung und System gewisse Freiheiten haben, waren dieser Feststellung aber nicht weiter gefolgt. Nach unserem zu Illustrationszwecken unternommenen Ausflug zur Erörterung von Zeitpunkt und Ort des Auftretens leistungsorientierter Fragestellungen (vgl. Abb. 1.9) und zur Diskussion betroffener Personenkreise und ihrer Einflußmöglichkeiten (vgl. Abb. 1.10) bietet sich eine Wiederaufnahme der Schnittstellendiskussion auf der Basis der in Abb. 1.9 aufgeführten "Schichten" eines RS an. In Wiederholung des Ihnen sicherlich geläufigen Konstruktionsprinzips der "Hierarchie virtueller Maschinen" (gelegentlich auch als "Zwiebelschalenmodell" kreisförmig dargestellt) stellen wir fest: Programme der Anwendungssoftware (zur Verdeutlichung: in compilierter, "lauffähiger" Form) können über Kommandos einer Kommandosprache zumindest "aufgerufen", also gestartet werden, u.U. auch zusätzlich "beim Ablauf" beeinflußt werden. Während ihres Ablaufs stützen sich Anwendungsprogramme auf Funktionen der System-Software ab, rufen diese auf (Prozeduren und/oder Supervisor Calls des Betriebssystems, Funktionen der "sonstigen" System-Software wie Datenbanken, Laufzeitsysteme der diversen Programmiersprachen, etc.); sie stützen sich darüber hinaus natürlich auf den verfügbaren Instruktionssatz des Rechners ab, rufen dessen Instruktionen auf. Instruktionen werden selbstverständlich auch seitens der System-Software verwendet, aufgerufen. Ist eine explizite Firmware-Schicht vorhanden, dann stützt sich diese (in Ausführung der aufgerufenen Instruktionen) auf Operationen der Hardware ab, ruft diese auf. Insgesamt erkennen wir somit die bekannte Hierarchie von Interpreter-Schichten, deren jede eine gewisse Menge von Diensten "nach oben" anbietet und eine gewisse Menge von Diensten "weiter unten" liegender Schichten benutzt. "Zwischen" Schichten liegen Schnittstellen, die wir von jetzt ab **Ebenen** nennen wollen. Damit nicht genug: Die Programme der Anwendungs-Software, und jene der System-Software, sind (jedenfalls wenn sie entsprechend moderner Software-Engineering-Maximen konstruiert sind) ihrerseits hierarchisch aufgebaut: Das Bild der hierarchisch aufeinander aufbauenden Schichten/Ebenen wiederholt sich innerhalb der Software-Schichten (und, wenn man den Trend richtig einschätzt, zunehmend auch in den Firmware-/Hardware-Schichten). Um nicht unrealistisch zu erscheinen: Obige Prinzip-Skizze der Struktur von Rechensystemen impliziert keine strikte Hierarchie; sie fordert auch keine systemweiten Ebenen; auch wird sie nicht allen Phänomenen gerecht (so ist sicher der Aspekt der Rekursivität nicht direkt abgedeckt und muß "innerhalb" einer Schicht versteckt werden).

RS-Struktur: Hierarchie virtueller Maschinen

Schnittstellen/ Ebenen

Überprüfen wir unsere Vorstellung von der "Leistung" von RS und die darauf aufbauenden Definitionen an dieser hierarchischen Struktur, dann erkennen wir leicht, daß die erwähnte Freiheit bei der Wahl der Schnittstelle Umgebung/System durch die Bezugnahme auf eine bestimmte der verschiedenen Ebenen ausgefüllt wird. Wie in Abb. 1.12 verdeutlicht, befindet sich "oberhalb" der Ebene ein hierarchisches (Teil-) System (oberste Schicht: Verhalten der Benutzer bzw. einer technischen RS-Umgebung), das im Verlauf des Betriebs die an der Ebene angebotenen Dienste aufruft/nutzt; befindet sich "unterhalb" der Ebene ein hierarchisches (Teil-)System (unterste Schicht: festverdrahtete Hardware), das die im Verlauf des Betriebs an der Ebene angeforderten Dienste ausführt/realisiert.

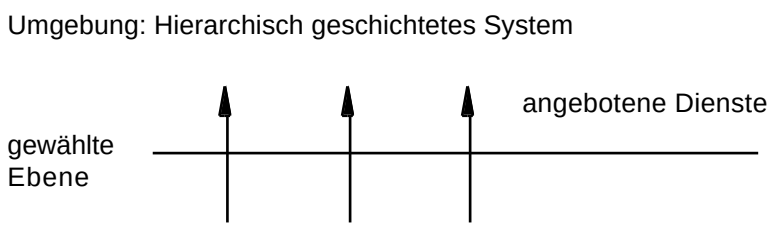

Abbildung 1.12: Beliebige Ebene als Schnittstelle
in einem hierarchisch geschichteten RS

Effektivität: ebenenspezifisch

Unsere Leistungskriterien vom Typ Effektivität (Dauer und Rate von Dienstabwicklungen) lassen sich zwanglos bezüglich jeder gewählten Ebene definieren: Eine Ebene ist durch eine spezifische Menge angebotener Dienste charakterisiert; für alle Dienste der Ebene, oder für eine Teilmenge dieser Dienste, oder für jeden einzelnen Dienst stellen

- die Ausführungsdauer (Ausführungszeit) der Dienste/des Dienstes (bei Schwankungen auch: die "mittlere" Ausführungszeit);

- die Anzahl ausgeführter Dienste pro Zeiteinheit (ZE), also der sog. Durchsatz (bei Schwankungen auch: der "mittlere" Durchsatz)

sinnvolle (Effektivitäts-)Leistungsmaße dar. Zur weiteren Verdeutlichung: Wir könnten uns z.B. interessieren für:

Ebene	Zeit/Durchsatz von:
Registertransfer-Ebene	Mikrooperationen
Instruktions-Ebene	Instruktionen
Betriebssystem-Ebene	Supervisor Calls (SVC's)
Gesamtsystem-Ebene	Jobs, Transaktionen

Beispiele

Wir erkennen direkt eine ganze Reihe von "üblichen" Leistungsmaßen, so z.B. solche vom Durchsatz-Typ:

- MIPS, MOPS, MFLOPS:
  Mega Instructions Per Second,
  Mega Operations Per Second,
  Millions of Floating Point Operations Per Second
  an der Instruktionsebene, gemeint im Sinne maximaler Leistung (also "Leistungsfähigkeit") bei Vollast, d.h. bzgl. einer permanent anfordernden Umgebung;

- E/A-Rate:
  Anzahl von Ein-/Ausgabe-Operationen pro Zeiteinheit
  an der Betriebssystemebene (bestimmte SVC's), gemeint im Sinne beobachteter Leistung unter einer bestimmten Last;

- Job-Durchsatz, Transaktions-Rate:
  Anzahl von Job-Ausführungen pro ZE,
  Anzahl Transaktionsabwicklungen pro ZE (TPS: Transactions Per Second)
  an der Gesamtsystem-Ebene, gemeint im Sinne beobachtbarer Leistung unter einer bestimmten Last.

Entsprechend tauchen die Begriffe Antwortzeit/Ausführungszeit/Reaktionszeit bzgl. jeder Ebene (d.h. bzgl. deren Diensten) auf.

Soweit passen unsere Vorstellungen von RS-Leistung und die Tatsache der hierarchischen Schichtung von Rechensystemen zumindest gut zusammen. Darüber hinaus eröffnet uns aber eine bewußte Berücksichtigung der hierarchischen Schichtung einen tiefen Einblick in die Zusammenhänge und Abhängigkeiten von Effektivitätsmaßen an verschiedenen Ebenen (s. Abb. 1.13):

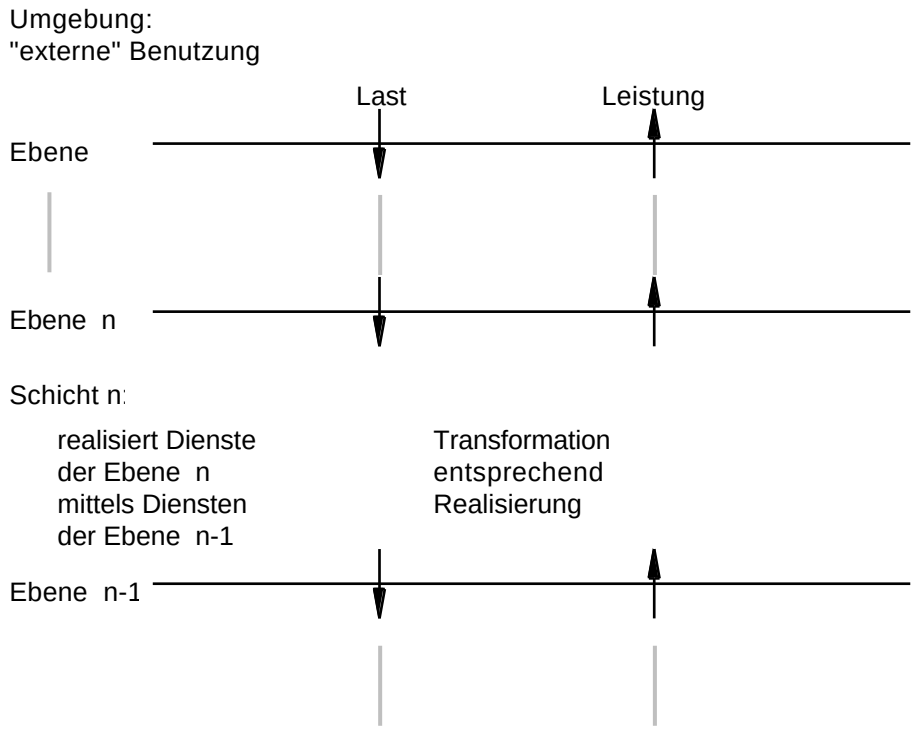

Abbildung 1.13: Prinzip der Last- und Leistungstransformation in hierarchischen Mehrebenen-/Mehrschichten-Strukturen

Schicht: An jeder Ebene werden gewisse Dienste (nach oben) zur Benutzung angeboten. Bei Aufruf/Benutzung eines dieser Dienste (von oben) setzt die darunter liegende Schicht den Aufruf in Dienstaufrufe an der nächsttieferen Ebene um, realisiert den Dienst mittels tieferer Dienste. Last wird gewissermaßen "von oben nach unten" transportiert. Umgekehrt begrenzt die maximale Bearbeitungsrate an der unteren Ebene (der maximale Dienstdurchsatz) die Bearbeitungsrate an der oberen Ebene - die Fertigstellung des höheren Dienstes ist ja an die Fertigstellung gewisser tieferer Dienste gebunden; entsprechend wirkt sich die Dauer der Abwicklung tieferer Dienste auf die Dauer der Abwicklung höherer Dienste aus.

transformiert Last,

transformiert Leistung Leistung (Durchsätze, Antwortzeiten) wird gewissermaßen "von unten nach oben" transportiert. Die "Transporte" (in beiden Richtungen) zwischen benach-

barten Ebenen beruhen auf der dazwischenliegenden Schicht, auf der Art und Weise, wie ein höherer Dienst durch tiefere realisiert wird, auf einem Algorithmus (einem Programm, im Falle einer Software-Schicht).

Nochmals: Leistung ist Systemantwort auf Last bzgl. Ebene

Wenn wir nun zur Abrundung unserer Betrachtungen die Struktur-Vorstellungen der Abb. 1.13/1.12 mit der verbalen Definition von Leistung im Sinne der Def. 1.8 konfrontieren, stellen wir fest: Leistung ist eine Antwort des Systems auf die überantwortete Last, bezogen auf eine festzulegende Ebene des (gesamten) hierarchisch geschichteten Systems; Last wird hervorgerufen durch das Verhalten des geschichteten Teilsystems oberhalb der festgelegten Ebene und manifestiert sich in Dienstaufrufen bezüglich dieser Ebene; Leistung ist eine Beurteilung des Verhaltens des hierarchischen Teilsystems unterhalb der festgelegten Ebene, bei Bearbeitung der vorliegenden Last.

Maschine

Es mag Ihnen bereits aufgefallen sein: Die dauernde Verwendung des überfrachteten Wortes "System" (was ist schon im betrachteten Kontext kein "System" ?) trägt nicht zur Klärung bei. Ändern wir ab sofort unsere Terminologie unter Einführung des Begriffs **Maschine** im folgenden Sinn: An einer Ebene treffen Anforderungen einer Last (das fordernde Teilsystem über der Ebene) auf eine Maschine (das realisierende Teilsystem unter der Ebene); die Maschine ist für unsere jetzigen Ausführungen hinreichend nach oben/außen charakterisiert durch ihre Schnittstelle, eine Ebene, die spezifische benannte Dienste zur Ausführung anbietet.

leistungsorientierte Problemstellungen

Auf der Basis unserer weiter gefestigten Vorstellungen über die Struktur von RS und den Begriff der Leistung jetzt wieder ein Ausflug in Richtung reale Welt: In welchen Formen treten denn leistungsorientierte Fragestellungen bei Entwurf, Realisierung und Betrieb von RS auf? Listen wir einige Problemformen:

- **Entwurfs-Probleme**

  Gegeben:
  - eine Last, d.h. eine Menge geforderter Dienste und die Art und Weise ihrer Inanspruchnahme;
  - Leistungsforderungen bezüglich Durchsatz und Bearbeitungszeit der geforderten Dienste.

  Gesucht:
  - eine Maschine, d.h. eine Realisierung der geforderten Dienste unter Einhaltung der Leistungsforderungen bei einem Betrieb gemäß gegebener Last.

  Entwurfsprobleme sind meist auf jeweils eine bestimmte Schicht konzentriert.

  Beispiele:
  - Instruktionsmix, MIPS-Zahl
    → CPU (u.U. bzgl. gegebener Register-Transfer-Ebene)
  - Funktionen Flugbuchungssystem, Transaktionsraten
    → Software (auf gegebenen Betriebssystem- und Instruktionsebenen)

- **Auswahl/Verbesserungs/Optimierungs-Probleme**

  Gegeben:
  - eine bestimmte Menge von Maschinen, d.h. alternative Maschinen gleicher Funktionalität;
  - eine Last, d.h. eine bekannte Umgebung,

welche eine Maschine dieser Funktionalität benutzen möchte;
- ein Leistungsmaß, d.h. eine Festlegung, welche quantitative Größe die maßgebende Rolle bei der Beurteilung der Maschinen übernehmen soll.

Gesucht:
- die im Sinne des festgelegten Leistungsmaßes optimale unter den Maschinen bei der gegebenen Last.

Optimierungsprobleme sind meist mit Nebenbedingungen bzgl. anderer Leistungsmaße (neben dem zentralen Leistungsmaß) und auch mit Nebenbedingungen bzgl. Kosten behaftet.

Beispiele:
- Konfigurationen einer Systemfamilie,
  installationsspezifische Anwendungslast,
  Antwortzeiten (bei zu überschreitender Durchsatzgrenze und zu unterschreitender Kostengrenze)
  → optimale Konfiguration.

- Einstellbarer Bereich Betriebssystem-Parameter,
  Anwendungslast,
  Antwortzeit trivialer Editier-Kommandos
  (bei zu überschreitendem job-Durchsatz)
  → optimaler Parameterwert.

In diesen Bereich fällt auch die duale Fragestellung nach der optimalen Last für eine feste Maschine bei gegebenem Leistungsmaß (z.B. bei der Arbeitsvorbereitung und bei der Planung von Tag-, Nacht-, Wochenend-Betrieb).

- **Was-Wenn-Probleme**

Gegeben:
- irgendeine Befürchtung, Idee, Ahnung

Gesucht:
- die zugehörige Auswirkung auf die RS-Leistung.

Dieser weitgehend unstrukturierte Problemkreis ist der realiter am häufigsten auftretende.

Beispiele:
- Was, wenn sich die Last ändert?
  (Auswirkung von Last-Evolution, von neuen Applikationen, ...)
- Was, wenn sich das System ändert?
  (Auswirkung von geplanten tuning-Ideen, von upgrading-Maßnahmen, von Re-Implementierungen wesentlicher System-Software-Moduln,...)

Lösungen? Wie lassen sich Lösungen für diese vielfältigen Problemkreise finden? Nun, eines haben uns unsere prinzipiellen, strukturellen Überlegungen bereits gebracht: Wir erkennen leicht, daß sich jede der obigen Fragestellungen auf eine konkrete Ebene als Schnittstelle zwischen Last und Maschine bezieht und daß in jeder der Fragestellungen bestimmte Variationen der Last, oder Variationen der Maschine, oder Variationen von Maschine und Last anvisiert sind, deren Auswirkungen auf festgelegte Leistungsmaße interessieren.

Lösungsansatz abstrakt gesehen Sehen wir uns diese Situation zunächst ganz abstrakt an: Bezüglich jeder denkbaren Ebene (also: Menge verfügbarer/benutzter Dienste) gibt es eine Menge von Lasten, die auf dieser Ebene aufsetzen können, darin eine (i.allg. sehr be-

grenzte) Teilmenge "momentan interessanter" Lasten; andererseits gibt es bezüglich dieser Ebene eine Menge von Maschinen, welche die Dienste dieser Ebene realisieren können, darin eine (i.allg. sehr begrenzte) Teilmenge "momentan interessanter" Maschinen; ferner gibt es bezüglich dieser Ebene eine Menge definierbarer Effektivitätsmaße, von denen einige "momentan interessant" sind. Jede der Maschinen aus der betrachteten Menge (nennen wir diese Menge "MASCHINEN") wird unter jeder Last aus der betrachteten Menge ("LASTEN") einen spezifischen Wert für jedes der betrachteten Leistungsmaße erzeugen; der Leistungs-Wert (bei mehreren gleichzeitig betrachteten Maßen ein vektorieller Wert) liegt dabei in einer Menge überhaupt möglicher Werte ("L-WERTE"). Abstrakt gesehen, beschäftigen wir uns demnach mit einer Abbildung von MASCHINEN x LASTEN in L-WERTE, einer **Leistungsfunktion:**

Menge "interessanter" Maschinen, Lasten

Leistungs-funktion

(1.14a) L: MASCHINEN x LASTEN —> L-WERTE

Die Skizze der Abb. 1.14b veranschaulicht diese Auffassung, in dem sie nochmals hervorhebt, daß zu einem bestimmten (Maschine,Last)-Paar ein bestimmter L-Wert "gehört".

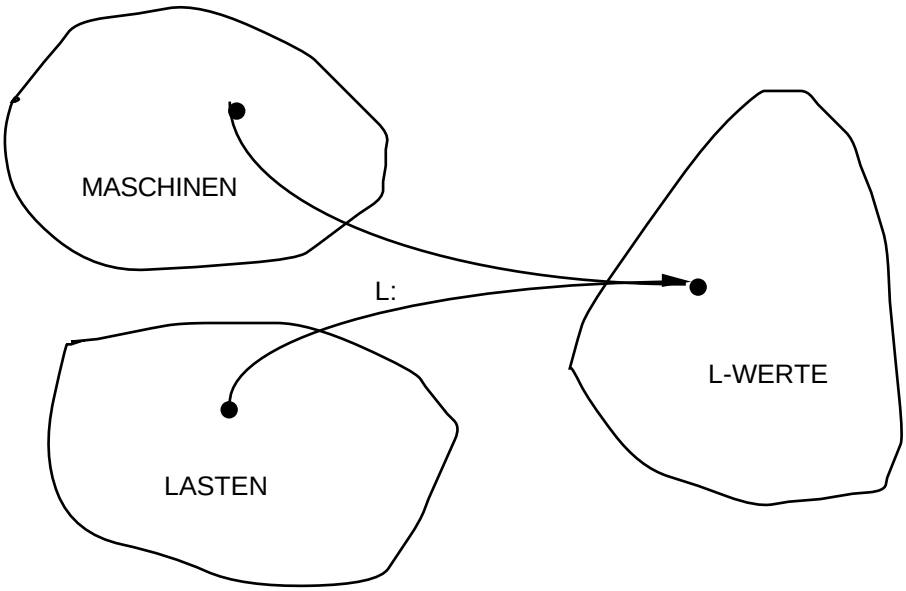

Abbildung 1.14b: Veranschaulichung der Leistungsfunktion

Auf der Suche nach Lösungsmöglichkeiten für die diversen leistungsorientierten Problemkreise wäre nun (idealerweise!) nach einer geschlossenen mathematischen Form für L zu suchen (diese Form wäre ein "Modell" im gleichen Sinne, wie die bekannte Formel $s=g \cdot t^2/2$ ein Modell des freien Falles ist); mit Hilfe dieser Formel ließen sich Lösungen für alle denkbaren einschlägigen Fragen rein auf formalem, mathematischem Wege bestimmen. Dies alles im Konjunktiv gesprochen, denn angesichts der potentiellen Vielfältigkeit der MASCHINEN- und LASTEN-Mengen und den daraus resultierenden Schwierigkeiten bei der Parametrisierung des Modells erscheint dieser Ansatz (zwar ideal, aber) doch reichlich idealistisch und höchstens für einfache Sonderfälle de facto gangbar.

Lösungs-möglichkeiten

ideales Modell

Was bleibt uns neben diesem idealen Weg noch an Möglichkeiten? Nun, eine Art "zweitbester" Weg eröffnete sich, wenn (zwar keine geschlossene Form für L vorläge, aber) L punktweise abtastbar wäre, d.h.

punktweise Abtastung

(1.15) L: (Maschine,Last) |—> L-Wert

für jedes einzelne, konkrete (Maschine,Last)-Paar ausgewertet werden könnte. Dies ist deshalb nur ein zweitbester Weg, weil von den aufgelisteten Problem-Formen nur die Was-Wenn-Probleme direkt faßbar werden, während Entwurfs- und Optimierungs-Probleme zwangsläufig auf einen iterativen Lösungsprozeß angewiesen sind, in dessen Verlauf wiederholt einzelne (M,L)-Paare bewertet und beurteilt werden, aus der Bewertung heraus neue, "hoffnungsvollere" (M,L)-Paare vorgeschlagen und bewertet werden, u.s.f.

Erfreulicherweise ist dieser Ansatz aber ein sehr realistischer. Er verweist uns einerseits auf den Weg der **Messung und Bewertung**, der uns sehr natürlich erscheinen sollte, haben wir doch den Begriff Leistung auf der Vorstellung eines tatsächlich "laufenden" (im Betrieb befindlichen), eine Last bearbeitenden RS aufgebaut, dessen Verhalten im Hinblick auf Effektivität/Effizienz zu beobachten war. Wir wollen diesen Weg, wegen der direkten Involvierung des zu beurteilenden Objekts Rechensystem, mit dem Term **Objekt-Experimente** bezeichnen.

Objekt-experimente

Bei Einsatz von Objekt-Experimenten ist das zu beurteilende Rechensystem (die Maschine) jeweils tatsächlich zu realisieren/zu implementieren/aufzubauen, mit einer lauffähigen Last zu beaufschlagen und während des Betriebs zu beobachten, d.h. geeignet zu messen. Wie schon gesagt, dieser Weg erscheint gangbar und natürlich, stößt allerdings bei Entwurfs-Problemen auf prinzipielle Schwierigkeiten, da ja während eines Entwurfsvorgangs per def. keine lauffähige Maschine vorhanden ist.

Die Idee des punktweisen Abtastens von L eröffnet uns andererseits (neben den Objekt-Experimenten) einen alternativen Weg: Wie wäre es, wenn wir nach einem Kalkül fahnden würden, der auf der Basis einer Beschreibung der zu beurteilenden Maschine und einer Beschreibung der zu verwendenden Last in der Lage wäre, den zugehörigen L-Wert zu errechnen/zu ermitteln? Den Kalkül sollten wir uns als Informatiker sicher von vornherein als für unseren "Rechenknecht" Computer aufbereitet vorstellen, also als Programm. Bildlich ausgedrückt, suchen wir damit eine Möglichkeit gemäß Abb. 1.16:

Abbildung 1.16: Prinzip der Modell-Experimente

Der "Kasten" Kalkül/Programm erlaubt erneut die Bezeichnung "Modell"; dies Modell ist aber von anderem (gelegentlich "numerisch" genanntem) Typ als die oben zunächst anvisierte "geschlossene Formel" - zumindest erlaubt er andere Typen; der Umgang mit dem Modell ist völlig konzentriert auf einen experimentellen Vorgang (im gleichen Sinne wie bei den Objekt-Experimenten): (M,L)-Paar nach (M,L)-Paar ist einer Beurteilung unterziehbar. Wir bezeichnen diesen Ansatz daher ab jetzt mit dem Term **Modell-Experimente**.

Modell-experimente

Vorteile

Haben wir mit der Idee der Modell-Experimente etwas gewonnen? Die Antwort

hängt sicher davon ab, welche Art von Kalkül/Programm wir durch den Modell-Kasten der Abb. 1.16 repräsentiert sehen. Zum einen ist offensichtlich jeder formelmäßige Zusammenhang potentiell subsumierbar. Darüber hinaus aber ist es denkbar, daß der Modell-Kasten den durch Maschine und Last beschriebenen, potentiell realen Betrieb eines RS per Programm "imitiert", so daß auch der breite Bereich der **Simulation** abgedeckt ist.

Ausblick

Wir werden uns im weiteren Verlauf der Vorlesung im wesentlichen um mathematische Leistungs-Modelle in der Idealform von (1.14) bemühen (d.h., um sog. "analytische" Modelle), werden bei der effektiven Berechnung von Leistungswerten aber vielfach auf das numerische Vorgehen im Sinne von (1.15) angewiesen sein.

Zusammenfassung

Fassen wir die Gedankengänge dieses Einführungs-Kapitels kurz zusammen:

- Der Begriff "Leistung von Rechensystemen" wurde motiviert und definiert als quantitative Bewertung der Güte des Betriebs eines Rechensystems unter einer Rechenlast; als Gütekriterien wurde insbesondere auf Effektivität und Effizienz verwiesen.

- Die Struktur von Rechensystemen wurde diskutiert, Leistungsprobleme wurden in einen hierarchischen Mehrebenen/Mehrschichten-Aufbau eingebettet; insbesondere wurde erkannt, daß Leistungsfragen jeweils bezüglich einer konkreten Ebene auftreten, und daß Leistungswerte bezüglich verschiedener Ebenen wechselseitig abhängig sind.

- Das Auftreten von Leistungsfragen im Verlauf des Lebenszyklus' von Rechensystemen, in verschiedenen Realisierungs-Schichten, für verschiedene Personenkreise, von verschiedenem Problemtyp wurde zur Unterstützung des Verständnisses erörtert.

- Möglichkeiten der Lösung von Leistungsproblemen wurden andiskutiert; darunter der vielversprechende Weg des Experimentierens, einerseits auf der Basis von Objekt-Experimenten, andererseits auf der von Modell-Experimenten.

LEERSEITE

Literatur **Leistungsbewertung von Rechensystemen,**
Lehrbücher in chronologischer Ordnung

Drum73 Drummond M.E.; Evaluation and measurement techniques for digital computer systems; Prentice-Hall 1973

HeCo75 Hellermann H./Conroy T.F.; Computer system performance; McGraw-Hill 1975

Stim76 Stimler S.; Leistungsbewertung, Leistungsmessung und Leistungsverbesserung von Daten verarbeitungssystemen; Oldenbourg 1976

ChYe78 Chandy K.M./Yeh R.T. (ed's); Current trends in programming methodology, vol.III: Software modelling; Prentice-Hall 1978

Ferr78 Ferrari D.; Computer systems performance evaluation; Prentice-Hall 1978

Koba78 Kobayashi H.; Modelling and analysis - An introduction to system performance evaluation methodology; Addison-Wesley 1978

Svob78 Svobodova L.; Computer performance measurement and evaluation methods: Analysis and applications; Elsevier 1978

BoNe79 Borovits I./Neumann S.; Computer systems performance evaluation; Lexington Books 1979

Berg80 Bergholz G.; Verhaltensmodelle von Prozessrechnern; Akademie-Verlag 1980

BrBa80 Bruell S.C./Balbo G.; Computational algorithms for closed queueing networks; Prentice-Hall 1980

GeMi80 Gelenbe E./Mitrani I.; Analysis and synthesis of computer systems; Academic Press 1980

SaCh81 Sauer C.H./Chandy K.M.; Computer systems performance modelling; Prentice-Hall 1981

BoAk82 Bolch G./Akyildiz I.F.; Analyse von Rechensystemen; Teubner 1982

FeSZ83 Ferrari D./Serazzi G./Zeigner A.; Measurement and tuning of computer systems; Prentice-Hall 1983

Lave83 Lavenberg S.; Computer performance modelling handbook; Academic Press 1983

LZGS84 Lazowska E.D./Zahorjan J./Graham G.S./Sevcik K.C.; Quantitative system performance - Computer system analysis using queueing network models; Prentice Hall 1984

StAr85 Stuck B.W./Arthurs E.; A computer and communications network performance analysis primer; Prentice Hall 1985

Pflu86 Pflug G.; Stochastische Modelle in der Informatik; Teubner 1986

LaCh87 Lam S.F. Chan K.H.; Computer capacity planning - Theory and Practice; Academic Press 1987

Mitr87 Mitrani I.; Modelling of computer and communication systems;
Cambridge University Press 1987

Leun88 Leung C.H.C.; Quantitative analysis of computer systems; Wiley 1988

MKer88 McKerrow P.; Performance measurement of computer systems; Addison-Wesley 1988

Gele89 Gelenbe E.; Multiprocessor performance; Wiley 1989

Berg89 Bergholz G.; Leistungsmodellierung von Rechnersystemen; Akademie-Verlag 1989

Bolc89 Bolch G.; Leistungsbewertung von Rechensystemen
mittels analytischer Warteschlangensysteme; Teubner 1989

Moll89 Molloy M.K.; Fundamentals of performance modelling; MacMillan 1989

CaHo90 Cady J./Howarth B.; Computer systems performance management
and capacity planning; Prentice-Hall 1990

King90 King P.J.B.; Computer and communications system performance modelling;
Prentice-Hall 1990

Robe90 Robertazzi T.; Computing networks and systems:
Queueing theory and performance evaluation; Springer 1990

Smit90 Smith C.; Performance engineering of software systems; Addison-Wesley 1990

Taka90 Takagi H.; Stochastic analysis of computer and communication systems;
North-Holland 1990

Jain91 Jain R.; The art of computer systems performance analysis; Wiley 1991

Lang91 Langendörfer H.; Leistungsanalyse von Rechensystemen; Hanser 1991

HaPa92 Harrison P.G./Patel N.M.; Performance modelling of communication networks and
computer architectures; Addison-Wesley 1992

Kant92 Kant K.; Introduction to computer system performance evaluation;
McGraw-Hill 1992

Cass93 Cassandras C.; Discrete event systems: Modeling and performance analysis;
Aksen Ass. 1993

BDMS94 Buchholz P./Dunkel J./Müller-Clostermann B./Sczittnick M./Zäske S.;
Quantitative Systemanalyse mit Markovschen Ketten; Teubner 1994

Dirl94 Dirlewanger W.; Messung und Bewertung der DV-Leistung; Hüthig 1994

HaZo95 Haas M. Zorn W.; Methodische Leistungsanalyse von Rechensystemen;
Oldenbourg 1995 (Handbuch der Informatik vol. 2.6)

Kris95 Krishna C.M.; Performance modelling for computer architects;
IEEE Computer Society Press 1995

Tran96 Tran-Gia P.; Analytische Leistungsbewertung verteilter Systeme; Springer 1996

Higg97 Higginbottom G.; Performance evaluation of communication networks; Artech House 1997

BGMT98 Bolch G. Greiner S. de Meer H. Trivedi K.S.; Queueing networks and Markov chains: Modeling and performance evaluation with computer science applications; Wiley 1998

GePu98 Gelenbe E. Pujolle G.; Introduction to queueing networks 2nd ed; Wiley 1998

Gunt98 Gunther N.; The practical performance analyst; McGraw-Hill 1998

Have98 Haverkort B.R.; Performance of Computer-Communication Systems; Wiley 1998

Lind98 Lindemann C.; Performance modelling with deterministic and stochastic Petri Nets; Wiley 1998

MeAl98 Menasce D.A. Almeida V.A.F.; Capacity planning for Web performance: Metrics, models, and methods; Prentice-Hall 1998

Germ00 German R.; Performance analysis of communication systems - Modeling with Non-Markovian stochastic Petri Nets; Wiley 2000

MeAl00 Menasce D.A. Almeida V.A.F.; Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning; Prentice-Hall 2000

LEERSEITE

Lösungshinweise zu den Testfragen

1.3:

An der Hardware-/Software-Schnittstelle befinden wir uns auf einem Niveau, auf der die Ausführbarkeit von "Instruktionen" aus einer festen Menge von Maschinen-Instruktionen vorausgesetzt wird, eben "auf Instruktions-Ebene". "Dienste" sind hier, entsprechend Angebot, Transferinstruktionen, Sprunginstruktionen, arithmetisch/logische Operationen,...

1.11:

- Tuning ist der Vorgang der "Feinabstimmung" eines RS auf die "normalerweise" auftretende Last. Tuning wird seitens eines Rechenzentrums vorgenommen, indem (veränderbare) Betriebssystem-Parameter geeignet gesetzt werden, Datei-Orte geeignet festgelegt werden, Puffergrößen aller Arten bestimmt werden, Freiheiten der E/A-Konfiguration geeignet genutzt werden, ...

- Pricing bezeichnet zunächst die Regeln, nach denen kostenpflichtige RS-Benutzer gemäß ihres Verbrauchs an Ressourcen zur Kasse gebeten werden. Der Begriff "pricing" (im Gegensatz zum normalen "accounting") trägt aber die bewußte Berücksichtigung der Tatsache in sich, daß das Benutzerverhalten sich über den Vorgang der Bezahlung verändern läßt. So lassen sich Benutzer i.allg. bewegen, "ungünstige" Fassungen ihrer Aufträge in "günstige" zu verändern: Last läßt sich beeinflussen. Als Beispiel: "Große Aufträge" in den Hauptbetriebszeiten (z.B. zwischen 9 und 16 Uhr) zu starten, kann preislich "bestraft" werden, so daß Benutzer nach Möglichkeit dazu tendieren werden, die Abwicklung dieser Aufträge in betriebsärmeren Zeiten zu verlangen. Vgl. auch nächsten Punkt.

- Portionierung ist eine der Abwehrmaßnahmen, die Benutzer bewußt oder unbewußt entwickeln, um den pricing-Mechanismen zu entgehen. Z.B.: Wenn ein großer Auftrag überproportional viel "kostet", dann wird die Tendenz bestehen, ihn in einige kleine Aufträge aufzuteilen (und umgekehrt!), also über die Gestaltung der Aufgaben-"Portionen" gut wegzukommen.

- Mit Systemgenerierung bezeichnet man den Vorgang der Festlegung von Optionen, die der Hersteller eines Betriebssystems der einzelnen RS-Installation beläßt (ggf. inclusive Installierung des Betriebssystems: "Kalt-Start"); hierbei können u.a. eine Reihe von Parametern gesetzt werden - vgl. Punkt "tuning".

- Betriebliche Regelungen können es verhindern, daß gewisse Aufträge zu "ungünstigen" Zeitpunkten gestartet werden (z.B. "tagsüber virtueller Speicherraum auf 3 MByte beschränkt") - sie sind gewissermaßen das "regulative" Analogon zu dem "psychologischen" pricing.

- Laufentscheidungen sind die Entscheidungen des Benutzers, wann er seine Aufträge bearbeitet sehen will. Maximen? Vgl. Punkt "Portionierung".

LEERSEITE