8

Leistungsbewertung und Dienstqualität von Echtzeitsystemen

8.1 Leistung von Echtzeitsystemen

Immer öfter werden wir mit der Notwendigkeit konfrontiert, Entscheidungen hinsichtlich der Vor- und Nachteile beim Einsatz von Datenverarbeitungssystemen zu treffen. Für diese Aufgabe muss man sich über geeignete Bewertungskriterien im Klaren sein, die für jede Anwendung unterschiedlich sind. Während früher immer die Rechengeschwindigkeit als wichtigstes Kriterium galt, legen die Nutzer schon seit den 1980er Jahren immer mehr Wert auf andere Kriterien, wie z.B.:

- Benutzerfreundlichkeit (Erlernbarkeit, Ergonomie, Hilfefunktion u.ä.)
- Wartbarkeit (klar strukturierter Aufbau, gut definierte Schnittstellen u.ä.)
- Flexibilität (Programm- und Parameteränderungen, Erweiterbarkeit u.ä.)
- Dienstleistungsangebot (verfügbare Dienste, Anschlussmöglichkeiten u.ä.)
- Sicherheit (Robustheit, Korrektheit u.ä.)
- Datenschutz (Autorisierungs- und Verschlüsselungsmechanismen, Datenschutzmechanismen gegen fremde Zugriffe)
- Leistungsfähigkeit (Geschwindigkeit, Antwortzeiten, Echtzeitverhalten, Speicherbedarf u.ä.)

Einige dieser Kriterien können mit Hilfe von Bewertungs- oder Monitorprogrammen überprüft werden. Es ist jedoch empfehlenswert, für jede Anwendung ein gesondertes Anforderungsprofil zu entwickeln und eine spezifische Bewertung vorzunehmen. Die genannten Bewertungskriterien können dabei als Orientierungshilfe dienen. Da man gewöhnlich mit mehreren alternativen Lösungen konfrontiert wird, möchte man auf der Basis der „gemessenen“ Bewertungskriterien Entscheidungen treffen und verschiedene Lösungen miteinander vergleichen. Basierend auf den Resultaten der Evaluierung möchte man die Systemlösungen, die anwendungsrelevante Kriterien erfüllen, mit einem Zertifikat auszeichnen.

Welche Bedeutung sollte der Leistungsfähigkeit von Echtzeitsystemen zugewiesen werden? Nach [143] spielt dabei nicht nur die Schnelligkeit von Be-

rechnungen eine Rolle, sondern auch höhere Ein-/Ausgabebandbreiten und schnellere Reaktionszeiten, als man sie bei Systemen für allgemeine Nutzung vorfindet (siehe Abb. 8.1). Sind aber CPU-Taktfrequenz oder E/A-Übertragungsraten als Kriterien wirklich relevant und/oder ausreichend? Würde mehr Speicherkapazität die Datenverarbeitungsleistung steigern? Wie wir schon vorher angedeutet haben, ist es nicht leicht, generell über Systemleistung zu sprechen – ganz zu schweigen davon, dass man bei Echtzeitsystemen die Zeitpunkte von Task-Aktivierungen und -Ausführungen als entscheidend betrachten sollte.

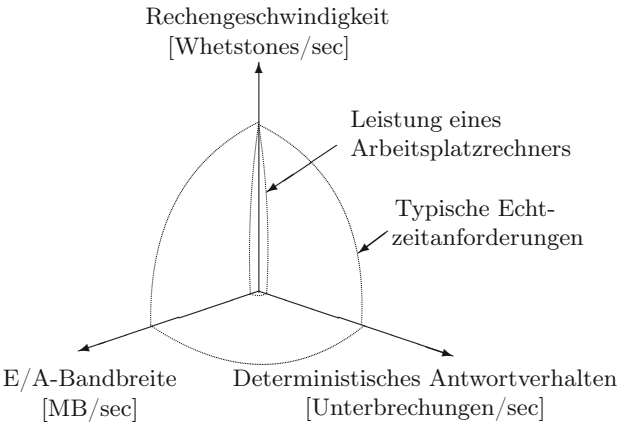

Abb. 8.1. Anforderungen an Echtzeitanwendungen

Die Definition des Echtzeitbetriebs impliziert die Notwendigkeit, für Echtzeitsysteme eigene Leistungskriterien zu formulieren [136]: „Teilnehmerbetriebssysteme oder nur einfach schnelle Systeme sind keine Echtzeitsysteme“. Rechtzeitiger Task-Ablauf ist viel wichtiger als (durchschnittliche) Geschwindigkeit. Stattdessen treten Aspekte wie schlimmstmögliche Szenarien, maximale Ausführungszeiten und maximale Verzögerungen in den Vordergrund. Verklemmungen müssen grundsätzlich verhindert und Fristen auf jeden Fall eingehalten werden. Bei der Realisierung sicherer Echtzeitsysteme mit vorhersehbarem Ausführungsverhalten ist die Berücksichtigung physischer Beschränkungen essentiell.

Die Anwendungen stellen viele und diverse Anforderungen an die verschiedenen Komponenten eines Echtzeitsystems. Basisanforderung ist die Fähigkeit, nach kurzen, deterministischen Fristen auf externe Ereignisse zu reagieren. Daher müssen Echtbetriebssysteme unter anderen die folgenden Merkmale aufweisen: jederzeit unterbrechbarer Kern, zeitgerechte Zuteilung und allmähliche Leistungsabsenkung als Reaktion auf Fehlfunktionen. Außerdem sollten wegen ihrer Bedeutung für die Synchronisation von Betriebsmittelzugriffen die Antwortzeiten, oder obere Schranken davon, der Systemaufrufe im voraus bekannt sein.

Besondere Anforderungen werden auch an Echtzeitprogrammiersprachen gestellt. Unter anderem müssen sie Mechanismen zur Kommunikation mit der Umgebung und zwischen Tasks untereinander (auch in verteilten Umgebungen) besitzen. Sie müssen Zeitverwaltung sowie zeitbedingte Task-Ausführung ermöglichen. Um die Zuverlässigkeit zu erhöhen, müssen sie Redundanzmaßnahmen unterstützen. Die Übersetzer von Echtzeitprogrammiersprachen müssen ausführbare Programme mit zeitlich vorhersehbarem und deterministischem Ausführungsverhalten erzeugen. Zu diesem Zweck müssen Sprachkonstrukte so eingeschränkt werden, dass Nichtdeterminismus verhindert wird. Um realistische Laufzeitabschätzungen zu erhalten, empfiehlt es sich, die Analysatoren in die Übersetzer zu integrieren, weil dort die internen Programmstrukturen bekannt sind. Programme für Echtzeitanwendungen sollten auch die Behandlung von Überlastungs- und Fehlersituationen, z.B. durch allmähliche Leistungsabsenkung, ermöglichen.

Nach der Erläuterung der Problematik stellen wir nun zuerst einige Methoden vor, die zur Leistungsbewertung von Echtzeitsystemen entwickelt wurden. Dann tragen wir die relevanten Leistungskriterien zusammen, die wir wegen ihrer qualitativen Natur unter dem Begriff „Dienstqualität“ subsummieren. Es wird auf kritische Punkte hingewiesen, wo diese Kriterien bei der Entwicklung eines Systems erfüllt werden müssen. Außerdem werden die Standards erwähnt, die Dienstqualität unterstützen und Zertifizierung von Echtzeitsystemen auf Dienstgüte hin ermöglichen sollen.

8.2 Leistungsbewertung von Echtzeitsystemen

Das Deutsche Institut für Normung hat in der Norm DIN 19242 [37] messbare Leistungsparameter für Prozessrechner zusammengetragen. Dafür geeignete Messmethoden stellte Uhle in [149, 10] detailliert dar und definierte 14 Programmbeispiele in C, FORTRAN, Pascal und PEARL zur Messung der Parameter. Die Tests umfassen Programmsegmente für Multiplikation, bedingte Verzweigungen, Sinus-Berechnung, Matrizeninvertierung, Bitmustererkennung, Unterbrechungsbehandlung mit Programmstart, E/A-Operationen mit externen Speichern, digitale E/A, Bediendialog, nachrichtenorientierte Synchronisation zwischen zwei Tasks, Lesen/Schreiben von Datensätzen aus/in Dateien mit direktem Zugriff, Datenübertragung über eine fehlerfreie Strecke und Erzeugung ausführbaren Codes mit anschließendem Programmstart. In der Norm DIN 19243 [38] wurden schließlich Methoden der Mess-, Steuerungs- und Regelungstechnik, geeignete analoge und binäre Größen sowie Datenerfassung, -verarbeitung und -ausgabe definiert.

Die Laufzeitmessmethode (siehe Abb. 8.2) nach DIN 19242 basiert auf folgenden Annahmen:

- der untersuchte Prozessrechner besitzt ein Mehrprozessbetriebssystem;

- zwei Programme konkurrieren um den Rechner, und zwar ein Prozess niedriger Priorität (NP), der eine rechenintensive Aufgabe bearbeitet und außer dem Prozessor keiner weiteren Betriebsmittel bedarf, und ein Prozess hoher Priorität (HP), der die Ausführung des niedrig priorisierten Prozesses blockiert;
- die Laufzeit des niederprioren Prozesses (t_{LP}) wird genau um die Ablaufzeit des hochprioren Prozesses (t_{HP}) verlängert.

Die Messmethode erfasst zuerst die Ablaufzeit des niederprioren Prozesses. Dann wird diejenige beider Prozesse (T) gemessen. Die Differenz der gemessenen Ablaufzeiten (T_{diff}) stellt die Laufzeit des hochprioren Prozesses dar und umfasst zugleich die notwendigen Kontextwechsel.

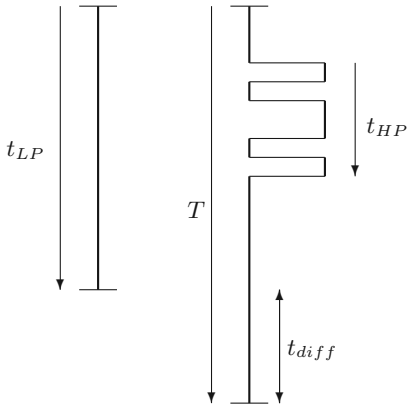

Abb. 8.2. Laufzeitmessung nach DIN 19242

Einen weiteren Anwendungsbereich deckt DIN 66273 [43] ab. Diese Norm definiert Datenverarbeitungsaufgaben, Messmethoden und Bewertung von Rechensystemen nach ihrer Datenverarbeitungsleistung mit besonderem Augenmerk auf erfass- und messbare Kenngrößen wie z.B. Antwortzeiten.

Zur Bewertung von Prozessrechensystemen werden also zum einen erfass- und messbare Leistungskenngrößen und zum anderen Messmethoden benötigt, die zusammen mit Testprogrammen auch „Benchmarks“ genannt werden. Benutzt man ein Anwendungsprogramm selbst als Testprogramm und trennt die Messdatenerfassung davon ab, so werden dieses Verfahren „Monitoring“ und die dazu benötigte Hard- und Software *Monitor* genannt.

8.2.1 Beispiele für Benchmark-Programme

Der **Rhealstone**-Benchmark [88, 89, 90] besteht aus sechs C-Programmen, mit denen folgende Charakteristika von Echtzeitbetriebsystemen gemessen werden können:

mittlere Umschaltzeit (t_1) zum Kontextwechsel zwischen zwei unabhängigen aktiven Prozesse gleicher Priorität;

mittlere Verdrängungszeit (t_2) die ein hochpriorer Prozess benötigt, um von einem Prozess niedrigerer Priorität die Kontrolle über das System zu übernehmen;

mittlere Unterbrechungslatenzzeit (t_3) vom Auftreten eines Unterbrechungssignals bis hin zur Ausführung der ersten Instruktion der zugeordneten Unterbrechungsbedienroutine;

mittlere Wartzezeit bei Synchronisationen (t_4) zwischen der Anforderung eines von einem anderen Prozess belegten Semaphors und der Zuweisung des zugeordneten Betriebsmittels;

mittlere Auflösungszeit von Verklemmungen (t_5) wenn ein Prozess niedriger Priorität, der ein Betriebsmittel belegt hat, durch einen dasselbe Betriebsmittel anfordernden Prozess höherer Priorität verdrängt wird;

mittlere Nachrichtenlatenzzeit (t_6) bei der Übermittlung von Nachrichten zwischen zwei Prozessen.

Die gemessenen Zeiten werden zu einem einzigen Wert $t' = \frac{1}{6}(t_1 + t_2 + t_3 + t_4 + t_5 + t_6)$ ausgedrückt in Sekunden zusammengefasst. Der sogenannte Rhealstone-Wert ist als dessen Kehrwert $\frac{1}{t'}$ definiert. Die Leistung eines Echtzeitsystems gilt als umso höher, je größer dieser Wert ist. Die Hauptschwächen dieser Bewertungsmethode sind, dass sie durchschnittliche Zeiten anstatt der wichtigeren maximal möglichen misst und dass diese Zeiten mit gleichen Gewichten in einem Wert zusammengefasst werden, ohne die Auswahl der Gewichte zu bedenken.

MiBench [55] ist eine in sechs Gruppen eingeteilte Sammlung aus 35 Anwendungsprogrammen, von denen jede einen spezifischen Einsatzbereich eingebetteter Systeme abdeckt: (1) Steuerung und Regelung in Automobilen und Industrie, (2) Endverbrauchergeräte, (3) Büroautomatisierung, (4) Netze, (5) Sicherheit und (6) Telekommunikation. Für harte Echtzeitsysteme ist nur die erste Gruppe von Nutzen. Alle Programme sind in Standard-C-Code verfügbar. Für gewisse Fälle bietet MiBench kleine und große Datenbestände an, denen eher Testanwendungen bzw. ganz realer Einsatz entsprechen. Trotz vieler Ähnlichkeiten handelt es sich bei MiBench und EEMBC [44] um unterschiedliche Benchmarks.

8.2.2 Laufzeitanalysatoren

Die Ablaufzeit von Programmen wird sowohl statisch als auch dynamisch analysiert. Mit ersterem Verfahren sollen im übersetzten Code Engpässe gefunden und a priori zeitgerechte Ausführbarkeit geprüft werden – die Methode hat Beweischarakter. Im Rahmen dynamischer Analysen werden dagegen Ablaufzeiten von Anwendungs- oder Benchmark-Programmen mit bekannten Eigenschaften gemessen – eine Methode mit der Qualität von Tests.

Wegen der Komplexität von Echtzeitanwendungen und der eingesetzten Prozessoren ist es äußerst schwierig, exakte Programmlaufzeiten statisch zu

bestimmen – auch auf Maschinensprachniveau. Um nicht zu pessimistische oder nur für Spezialfälle oder kleine Teilsysteme zutreffende Schätzungen zu erhalten, ist es deshalb unvermeidlich, die Nutzung bestimmter hochsprachlicher Konstrukte einzuschränken und möglichst einfache Prozessoren ohne Gleitkommaarithmetik, Caches oder Pipelining zu benutzen. Aus diesen Gründen werden Echtzeitanwendungen in der derzeitigen Praxis nur sehr selten statisch analysiert. Statt dessen beschränkt man sich trotz höheren Sicherheitsrisikos und geringerer Genauigkeit auf Messungen.

Als ein Beispiel für Laufzeitanalysatoren sei hier zunächst die **Papa-Bench**-Suite [111] genannt, die unter der GNU-Lizenz verbreitet wird. Sie basiert auf den im Rahmen des *Paparazzi*-Projekts für verschiedene unbemannte Flugkörper entwickelten Echtzeitanwendungen. Der Analysator dient zur Bestimmung längstmöglicher Ausführungszeiten (Worst-Case Execution Time, WCET) und zur Zuteilbarkeitsanalyse. Seine Programme sind von einfacher Struktur: mit Ausnahme zweier while-Schleifen bestehen sie nur aus nicht geschachtelten Zählschleifen mit konstanten oberen Grenzen. Das vereinfacht die Laufzeitanalyse deutlich und lässt abgeschätzte WCETs realen sehr nahe kommen.

Der **Hartstone**-Analysator [135] besteht aus einer Reihe von Anforderungen an ein synthetisches Anwendungsprogramm zum Testen harter Echtzeitsysteme. Er wurde an der Carnegie Mellon-Universität in Ada geschrieben. Diese Anforderungen definieren mehrere Testserien für verschiedene Typen von Prozessen, und zwar periodische mit harten Fristen, aperiodische mit weichen oder harten Fristen und Synchronisationsprozesse ohne Fristen. Kritisch dabei sind die Laufzeiten von Synchronisationsprozessen, die periodische und aperiodische Klientenprozesse bedienen. Ein Test gilt als bestanden, wenn die ihm gesetzte Zeitschranke eingehalten wird. Für periodische Prozesse sieht der Hartstone-Analysator folgende Testserien vor:

1. periodische Prozesse, harmonische Frequenzen;
2. periodische Prozesse, nicht-harmonische Frequenzen;
3. periodische Prozesse, harmonische Frequenzen mit aperiodischer Ausführung;
4. periodische Prozesse, harmonische Frequenzen mit Synchronisation;
5. periodische Prozesse, harmonische Frequenzen mit aperiodischer Ausführung und Synchronisation.

Für jede Testserie gibt es eine Grundprozessmenge mit exakt definierten Parametern Rechenlast, Zeitschranken und Periodenlängen. Ein Test fängt mit dieser Menge an und stellt fest, ob die Zeitschranken eingehalten werden. In jedem nachfolgenden Test wird je ein Parameter der Prozessmenge geändert. Ein Test wird solange wiederholt, bis eine Fristverletzung festgestellt wird. Das Ergebnis einer Testserie ist der Zustand der Parameter der Prozessmenge bei Eintritt von Fristverletzungen unter größter Rechenlast. Anstatt eines Anwendungsprogramms kann als synthetische Rechenlast der Whetstone-Benchmark [29] benutzt werden.

8.2.3 Leistungsmonitore

Es gibt mehrere Gründe, die Leistung von Echtzeitsystemen fortlaufend zu erfassen, z.B. um Lastcharakteristika zu bestimmen, Korrektheit zu verifizieren oder um Zuverlässigkeit festzustellen. Leistungsmonitore erfassen die Betriebsdaten eines Systems und bereiten sie in einer zur Interpretation geeigneten Form auf. Man unterscheidet Monitore für Hard-, Soft- und Firmware sowie hybride Monitore [101]. Benchmarks könnte man als Software-Monitore betrachten. Um genauere Resultate zu erzielen, werden oft externe Hardware-Monitore eingesetzt oder Zeiten auf niedrigem Niveau beobachtet (Firmware-Monitore). Wir sprechen von hybriden Monitoren, wenn Hardware-Monitore über interne Logik zur Datenaufbereitung und -interpretation verfügen.

Der Monitor **Real-Time Space-Time Adaptive Processing (RT_STAP)** [27] wird zur Bewertung skalierbarer Hochleistungsrechner eingesetzt, die in flugzeuggestützten Radar-Geräten Störsignale und Interferenzen entfernen sollen. Aus Gründen der Skalierbarkeit verfügt RT_STAP über adaptive Algorithmen variabler Komplexität und unterschiedlichen Verfeinerungsgrades. Der Monitor bietet harte, mittelschwere und leichte Anwendungsfälle basierend auf drei adaptiven Algorithmen zur Verarbeitung dem Doppler-Effekt unterliegender Signale an.

8.3 Kriterien der Dienstqualität von Echtzeitsystemen

Die Qualität eines Echtzeitsystems muss ganzheitlich bewertet werden – als Gesamtheit seiner Eigenschaften im Vergleich mit den Nutzererwartungen. Dabei werden Entwicklungsmethodik, Architektur, Betriebssystem, Zuteilungsstrategie und Programmiersprache mit in Betracht gezogen.

8.3.1 Vorhersehbarkeit und Verlässlichkeit

Die Hauptqualitätsmerkmale von Echtzeitsystemen sind die Vorhersehbarkeit ihres Ausführungsverhaltens und ihre Verlässlichkeit. Eine vereinfachte Definition bezeichnet Verlässlichkeit als funktional-korrekte und Vorhersehbarkeit als zeitgerechte Arbeitsweise.

Vorhersehbarkeit ist der Grad des Vertrauens darin, dass korrekte, qualitative oder quantitative Prognosen über den Zustand eines Systems gemacht werden können. Sie stellt, sogar in Fehlersituationen, den Grad des Verhaltensdeterminismus und der Zeitgerechtheit des Systems dar. Dabei werden funktionale und zeitliche Vorhersehbarkeit als eine integrale Eigenschaft betrachtet.

Aussagen über die Vorhersehbarkeit eines Systems werden auf Grund des Anteils der Fälle gemacht, in denen es sich wie vorhergesagt verhält; so ist es auch möglich, zwei Systeme hinsichtlich ihrer Vorhersehbarkeit zu vergleichen. Als

Beispiel für „Messung“ und Vergleich von Vorhersehbarkeit sei ein Monitor genannt, der die Raten von Ausfällen und Fristverletzungen in verschiedenen Lastsituationen und Szenarien misst. Die gewonnenen Ergebnisse lassen sich dann mit den Spezifikationen vergleichen.

Verlässlichkeit ist der Grad der Vertrauenswürdigkeit eines Systems, die es erlaubt, sich auf dessen Dienste/Funktion begründet zu verlassen. Verlässlichkeit wird nach [12] als kumulative Eigenschaft aus Verfügbarkeit, Zuverlässigkeit, funktionaler Sicherheit, Betriebsschutz, Integrität und Wartbarkeit betrachtet. Je nach Anwendungsfall verschiebt sich die relative Bedeutung dieser Komponenten.

Im Folgenden werden wir im Detail die erwähnten Eigenschaften behandeln, die zusammengenommen Vorhersehbarkeit und Verlässlichkeit definieren.

8.3.2 Qualitativ-exklusive Kriterien

Funktionale Korrektheit (X1)

Funktionale Korrektheit ist für Echtzeitanwendungen genau so wichtig wie für jede andere Anwendung. Es ist schwer, sie formal nachzuweisen. Oft werden Tests aller Art durchgeführt, wodurch die meisten funktionalen Fehler entdeckt werden können, vorausgesetzt, die Spezifikation ist präzise genug. Solche Tests können funktionale Korrektheit zwar nicht beweisen, sind aber meist ausreichend, um ein System „korrekt in Bezug auf die Spezifikation“ nennen zu können. Danach sollten keine Änderungen mehr gemacht werden, da sie neue Fehlerquellen darstellen. Die Korrektheit hängt aber bei Echtzeitanwendungen auch von zeitgerechter Verarbeiung ab.

Zeitgerechtheit (X2)

Zeitgerechtheit ist die Fähigkeit, alle Fristen einzuhalten. In harten Echtzeitumgebungen ist dies ein exklusives Kriterium, d.h. die Fristen werden entweder überschritten oder nie verletzt. Fristen nicht einhaltende Systeme sind für harte Echtzeitanwendungen nicht einsetzbar.

Wie funktionale Korrektheit ist Zeitgerechtheit als solche schwer nachzuweisen. Zu diesem Zweck wurden diverse Monitore, Benchmarks sowie Laufzeitanalysatoren entworfen. Diese agieren auf verschiedenen Niveaus, von Hochprogrammsprachen- über das Maschinensprachen- bis zum Hardware-Niveau, um Zeitanalysen zu ermöglichen. Wegen der Komplexität höherer Programmiersprachen und der hoch parallelen Architektur moderner Prozessoren, für die sogar die Ausführungszeiten einzelner Instruktionen nicht mehr genau angegeben werden können, liefern die genannten Werkzeuge allerdings keine genauen Resultate. Im Allgemeinen erhält man nur obere Zeitschranken, die oft zu pessimistisch sind. Deswegen gibt man sich in der Praxis meist

mit *zeitlich deterministischem Verhalten* zufrieden. Um es festzustellen, werden mit Monitoren alle denkbaren und spezifizierten Ablaufszenarien auf ihr Antwortzeitverhalten hin überprüft.

Permanente Bereitschaft (X3)

Permanente Bereitschaft bedeutet ununterbrochene Verfügbarkeit zeitgerechter und korrekter Dienste. Sie beinhaltet deterministisches Verhalten, d.h. Abwesenheit unzulässiger Zustandswechsel, unbeschränkter Verzögerungen oder beliebig langer Ablaufzeiten.

Permanente Bereitschaft wird durch Verklemmungsfreiheit und Fehlertoleranz unterstützt. Außer hoher Verfügbarkeit bedeutet permanente Bereitschaft auch, dass sich die Reaktionszeiten eines Systems während seiner gesamten Betriebszeit innerhalb vorgegebener Schranken bewegen. Zur Gewährleistung permanenter Bereitschaft ist mit den Vorgängen in der Umgebung schritthaltender Systembetrieb erforderlich. Um dies zu erleichtern, bedarf es Mechanismen zur dynamischen Rekonfiguration, die deterministisches Entfernen und Inbetriebnehmen von Systemkomponenten nach wohldefinierten Bedingungen mit minimalen und vorhersehbaren Auswirkungen auf die Laufzeit zulassen.

Einhaltung physikalischer Einschränkungen (X4)

Physikalische Einschränkungen sind durch die eingesetzten Hardware-Plattformen, E/A-Schnittstellen sowie die Strukturen der Anwendungsprogramme gegeben. Um zu gewährleisten, dass diesen Einschränkungen Rechnung getragen wird, sollten die System- und Anwendungskomponenten perfekt aufeinander abgestimmt werden. Gegebenenfalls sind nur statische Datenstrukturen und reale Werte zu benutzen. Nichtberücksichtigung dieser Richtlinien kann zu Sicherheitsrisiken führen, die dann adäquat behandelt werden müssen, um deterministischen und sicheren Betrieb aufrecht zu erhalten.

Zertifizierbarkeit (X5)

Um die Qualität eines Systems zertifizieren zu können, muss es gemäß eines genormten Schemas entwickelt worden sein, z.B. nach dem Capability Maturity Model (CMM) für Software [117], das die Voraussetzungen zur Zertifizierung eines Software-Projekts nach ISO 9001 darstellt. Wird eine Zertifizierung auf Systemebene angestrebt, so sollte man Komponenten von nach ISO 9001 zertifizierten Herstellern wählen, die sich dann einfach in das eigene Qualitätssystem integrieren lassen. Zertifikate nach den Normen der Familie ISO 9000 bescheinigen nicht die Qualität einzelner Produkte, sondern nur deren qualitätsbewusste Entwicklung. Bei der Zertifizierung von Sicherheit müssen die relevanten Normen, z.B. IEC 61508, berücksichtigt und die der jeweiligen Anwendung angemessene Sicherheitsanforderungsklasse bestimmt werden. Ähnlich verfährt man bei der Zertifierung von Daten- und Zugriffsschutz, wofür

es analog Vertraulichkeitsklassen, z.B. laut [113], gibt. Zertifikate werden immer für einen Bereich oder eine Anwendung auf der Grundlage von Prüfungen gegen geltende Normen erteilt.

8.3.3 Qualitativ-graduelle Kriterien

Obwohl wir einige graduelle Kriterien auch analytisch darstellen und statistisch verarbeiten könnten, lassen sie sich generell nicht quantifizieren – meistens nehmen sie Werte wie *hoch, mittel, niedrig* an oder man sagt, dass ein System ein Kriterium zu einem höherem Grade als ein anderes erfülle.

Zeitgerechtheit (G0)

Obwohl Zeitgerechtheit für harte Echtzeitumgebungen ein exklusives Kriterium ist, ist es für die meisten Echtzeitsysteme eher ein graduelles und wird dementsprechend unterschiedlich kritisch berücksichtigt. Auf Basis des Kriteriums X2 wird ein System nicht zugelassen, falls es Fristen überschreitet. Auf Basis des Kriteriums G0 wird es zugelassen, sofern ein bestimmter Anteil von Fristverletzungen oder ein bestimmter Spielraum dabei nicht überschritten werden. Nach Anzahlen von Fristüberschreitungen oder nach zugelassenen Spielräumen lassen sich dann Systeme miteinander vergleichen und als „mehr oder weniger zeitgerecht“ einstufen.

Verfügbarkeit (G1)

Verfügbarkeit ist der Anteil der Zeit, in dem sich ein System in einem funktionierendem Zustand befindet. Sie repräsentiert auch die Bereitschaft zur korrekten Diensterbringung oder den Grad, zu dem es durch aufgetretene Fehler daran gehindert wird. Ihr exklusives Äquivalent ist permanente Bereitschaft (X3).

Systemverfügbarkeit kann durch allmähliche Leistungsabsenkung ergänzt werden. *Allmähliche Leistungsabsenkung nach Ausfällen stellt das Maß an Funktionalität dar, das ein System in Fehlersituationen noch erbringt.* Wenn die Dienstqualität des Systems überhaupt nachlässt, so soll dies proportional zur Schwere der Fehler erfolgen. Mechanismen aus dem Bereich der Fehlertoleranz werden eingesetzt, um dieses Ziel zu erreichen. *Fehlertoleranz stellt den Grad der Befähigung eines Systems dar, seinen Betrieb im Falle von Komponentenausfällen ordnungsgemäß fortzusetzen.*

Zuverlässigkeit (G2)

Zuverlässigkeit ist der Grad der Fähigkeit eines Systems, seine geforderte Funktionalität unter gegebenen Umständen während einer spezifizierten

Zeit bereitzustellen. Zuverlässigkeit bedeutet Kontinuität korrekter (funktional und zeitgerecht) Diensterbringung und ist dabei auch von anderen Verlässlichkeitskriterien abhängig.

Zuverlässigkeit wird während des Systementwurf am besten durch Fehlervermeidung unterstützt und während des Betriebs durch Fehlertoleranz- und Sicherheitsmechanismen gewährleistet.

Funktionale Sicherheit (G3)

Funktionale Sicherheit stellt den Grad des Schutzes gegen durch Fehler verursachte Risiken dar. Sie hängt eng mit Zuverlässigkeit und Robustheit zusammen, die beide die Sicherheit eines Systems fördern.

Eine Anwendung wird als sicherheitskritisch bezeichnet, wenn Fehlfunktionen Leben oder Gesundheit von Menschen oder die Umwelt gefährden oder aber materielle Schäden am System selbst oder in seiner Umgebung hervorrufen können. Um die Risiken von Fehlern zu mindern, sieht sicherheitsgerichteter Entwurf eine Reihe von Strategien und Mechanismen vor. Abhängig von einer entsprechenden Bewertung können Aussagen über den Grad der Sicherheit eines Systems (auch formal in Form von Sicherheitsanforderungsklassen [75]) in Bezug auf die Anforderungen aus der Systemspezifikation gemacht werden.

Betriebsschutz (G4)

Betriebsschutz stellt den Grad des Schutzes gegen Konsequenzen von Fehlern dar, die von der Systemumgebung hervorgerufen werden. Betriebsschutz wird nach [12] als kumulative Eigenschaft aus *Verfügbarkeit*, *Vertraulichkeit* und *Integrität* im Sinne der Abwesenheit nicht autorisierter Zustandsänderungen betrachtet, wobei Vertraulichkeit den Grad des Schutzes vor nicht autorisierter Offenlegung von Informationen bezeichnet.

Wie bei der funktionalen Sicherheit hängt der Grad erreichten Betriebsschutzes von den angewendeten Strategien und Methoden ab. Die Fragen, die sich bei der Bewertung vom Betriebsschutz stellen, beziehen sich auf diese Strategien und Mechanismen. Aussagen über Betriebsschutz werden hinsichtlich der Erfüllung gegebener Richtlinien, wie z.B. [113, 112], gemacht.

Integrität (G5)

Integrität stellt die Abwesenheit unzulässiger Systemzustandsänderungen dar. [12]

Sie ist Voraussetzung für Sicherheit, Verfügbarkeit und Zuverlässigkeit und wird selten als eigenständiges Attribut betrachtet. Ein gutes Beispiel einer Umgebung zur Gewährleistung von Integrität sind *Sicherheitsschalen* nach

[95]. Den darin enthaltenen Monitoren obliegt es, die Integrität der Anwendungen sicherzustellen. Wann immer ein Systemparameter droht, einen unsicheren oder unzulässigen Wert anzunehmen, wird eine Korrektur initiiert, um ihn wieder in den sicheren Bereich zurückzuführen. Integrität wird mittels Fehlertoleranzmaßnahmen erreichter Robustheit und durch Einfachheit des Systementwurfs unterstützt. Mit hoher Wahrscheinlichkeit stellt ein komplizierter Entwurf ein hohes Risiko für die Integrität eines Systems dar.

Robustheit (G5.1)

Robustheit ist der Grad der Widerstandsfähigkeit eines Systems, wenn es mit hohen Belastungen, ungültigen Eingaben oder Veränderungen in seiner internen Struktur oder in seiner Umgebung konfrontiert wird.

Robustheit unterstützt Integrität und ist eng mit Zuverlässigkeit verbunden. Im Gegensatz zu letzterer bezieht sich Robustheit mehr auf die Anzahl von Veränderungen, Fehlern oder Fristüberschreitungen, die ein System intakt „überlebt".

Komplexität (G5.2)

Die zeitliche Komplexität eines Problems ist durch die Anzahl der zur Lösung eines seiner Instanzen mit Hilfe des effizientesten Algorithmus erforderlichen Schritte als Funktion des Eingabeumfangs definiert. Damit lassen sich Probleme in Komplexitätsklassen einteilen. Analog wird auch die räumliche Komplexität, d.h. der Speicherbedarf, von Algorithmen betrachtet.

Mittel zum Ausdrücken von Komplexität sind z.B. die Anzahl logischer Pfade durch ein Anwendungsprogramm oder die Anzahl der Zustände in dessen äquivalenter Implementierung als endlicher Automat. Man könnte Komplexität auch als Anzahl der zur Hardware-Implementierung (z.B. in einem FPGA) des Programms benötigten Gatterfunktionen sehen. Generell gilt, dass (unnötige) Komplexität die Integrität mindert, Einfachheit letztere dagegen (indirekt) unterstützt.

Wartbarkeit (G6)

Wartbarkeit stellt den Grad der Eignung eines Systems für Reparaturen und Modifikationen dar.

Wartbarkeit ist eine gesamtheitliche Eigenschaft, die durch Portabilität und Flexibilität unterstützt wird.

Portabilität (G6.1)

Portabilität stellt den Grad des nötigen Aufwandes zur Übertragung eines (Software-) Systems auf eine andere Plattform oder Umgebung dar.

Portabilität wird von der Komplexität des (Software-) Systems und der Anzahl/Komplexität der zu ändernden bzw. auszutauschenden Komponenten beeinflusst. Sie wird auch durch die Anzahl der Plattformen bestimmt, auf denen die Anwendung laufen soll. Dieses Kriterium kann auf Anpassungsfähigkeit im Sinne der Anzahl der Änderungen hin übertragen werden, die zur Ermöglichung von Portierungen erforderlich sind.

Flexibilität (G6.2)

Flexibilität stellt den Grad der Anpassungsfähigkeit eines Systems an eine neue Umgebung dar. Sie kann auch als dessen Widerstandsfähigkeit gegen Beeinflussungen von der oder Störungen durch die Umgebung gesehen werden.

Ähnliches wie für Portabilität gilt auch für Flexibilität, die ein Maß an Aufwand zur Erzielung von Anpassungsfähikeit im Sinne von Portabilität und späterer Aktualisierbarkeit darstellt. Der Grad an Flexibilität ist für solche Anwendungen am höchsten, bei deren Entwurf Portabilität und Anpassungsfähigkeit von Anfang an berücksichtigt wurden.

8.3.4 Quantitative Kriterien

In diesem Abschnitt werden quantifizierbare Kriterien vorgestellt. Dabei werden auch die quantitativen Aspekte schon erwähnter exklusiver und gradueller Kriterien betrachtet.

Zeitgerechtheit (Q0)

Die schon als qualitativ exklusives (X2) und graduelles (G0) Kriterium erwähnte Zeitgerechtheit besitzt auch quantitative Aspekte. Generell könnte man Zeitgerechtheit T in Abhängigkeit von Leistung E und Last B in der Form $T = \frac{E}{B}$ darstellen. Mit Leistung ist hier funktional-korrekte rechtzeitige Ausführung gemeint. Umfang oder Frequenz der Eingangsdaten wird als Last bezeichnet. Sobald der Quotient unter 1 fällt, werden Fristen überschritten. Es ist Sache der Anwendungsspezifikation, eine untere Schranke -– erforderlicher „Grad an Zeitgerechtheit“ – oder eine obere für „Höchstbelastung“ zu definieren. Wenn für ein System gemäß seiner Betriebsumgebung eine maximale Anzahl von Fristenüberschreitungen oder ein maximaler Spielraum dafür angegeben werden, so können diese quantitativen Daten dazu benutzt werden, auf die Eignung des Systems für eine Anwendung zu schließen. Nach DIN 66273

wird Zeitgerechtheit in ähnlicher Weise bewertet und als quantivatives Kriterium ausgedrückt. Auf der Grundlage messbarer Leistungsparameter lassen sich die drei folgenden Benchmark-Kriterien definieren:

Durchsatz $L_1 = \frac{B}{B_{ref}}$,
Verarbeitungszeit $L_2 = \frac{t}{t_{ref}}$, und
Termintreue $L_3 = \frac{E}{B}$ $(L_3 \leq 1)$.

wobei B die aktuelle Last, E die erbrachte Leistung sowie B_{ref} und t_{ref} aus anderen Implementierungen, Simulationen oder durch Expertenschätzungen gewonnene Referenzwerte seien.

Die nachfolgenden Kriterien stellen verschiedene Aspekte der Zeitgerechtheit als (Benchmark-) Kriterien dar.

Antwortzeiten auf Ereignisse (Q1)

Antwortzeiten beziehen sich auf Ereignisse, denen bei korrekter Funktionsweise einzuhaltende Zeitschranken zugeordnet sind. Werden die entsprechenden Reaktionen rechtzeitig beendet, so spricht man von Echtzeitverhalten. Im Rahmen der Betrachtung von Antwortzeiten spielen die folgenden Kenngrößen von Echtzeitbetriebssystemen eine große Rolle:

Unterbrechungslatenzzeit ist der Zeitraum vom Auftreten eines Unterbrechungssignals bis hin zum Beginn der Ausführung eines anwendungsspezifischen Bedienprozesses,

Bedienzeiten von Systemaufrufen müssen nach oben hin beschränkt und unabhängig von der Anzahl im System (aktiver) Objekte sein,

Maskierungszeit, d.h. die maximale Dauer der Maskierung und mithin der Nichtbeachtung ankommender Unterbrechungssignale durch das Betriebssystem und seine Gerätetreiber, muss beschränkt und so kurz wie möglich sein.

Dauer bis zur Fehlererkennung und -behebung (Q2)

Die Zeiten zur Erkennung und Behebung von Fehlern beziehen sich auf die Aktivitäten, die z.B. zur Datenübertragung oder Signalfilterung verantwortlich sind. Diese Zeiten stellen die Antwortzeiten zugehöriger Aktivitäten dar, die beschränkt sein müssen, da ihre Überschreitung ebenfalls zum Fristverletzungen oder Systemfehlfunktionen führen kann.

Rauschunterdrückung (Q3)

Bei Signalfilterung und -erkennung muss der Rauschpegel innerhalb bestimmter Schranken gehalten werden. Deshalb muss auch die Zeit zur Anpassung an Interferenzen, die extreme Signal-/Rauschverhältnisse darstellen, begrenzt sein. Die entsprechenden Schranken stellen Grenzen der funktionalen (X1) sowie zeitlichen Korrektheit (Q1) dar.

MTBF, MTTF und MTTR (Q4)

Die mittleren Zeiten zwischen Ausfällen (MTBF), bis zum Auftreten von Ausfällen (MTTF) und bis zur Behebung von Ausfällen (MTTR) sind Kenngrößen, die während des Systembetriebs gewonnen und dann statistisch ausgewertet werden können. Da es sich um Mittelwerte handelt, sind sie nur in weichen Echtzeitumgebungen anwendbar. Für harte Echtzeitanwendungen haben diese Größen nur sehr begrenzten Wert, weil die von ihnen ausgedrückten Zeitdauern nicht garantierbar sind.

Kapazitätsreserven (Q5)

Benchmarks wie Rhealstone und insbesondere anwendungsspezifische wie Hartstone sehen Leistungsindikatoren vor, aus denen man auf zeitgerechtes Verhalten schließen kann. Die entsprechenden Messwerte sind auch hilfreich bei der Bestimmung von Kapazitätsreserven für die Antwortzeiten einzelner Aktivitäten. Bei Betriebsmitteln sind Kapazitätsreserven in Form freier Einsteckplätze für Platinen, Unterschieden zwischen benötigter und verfügbarer Speicherkapazität, Rechengeschwindigkeit, maximaler Anzahl aktiver Prozesse usw. zu finden.

Gesamtkosten (Q6)

In vielen Projekten ist der Preis die alles überwiegende Bewertungsgröße, weshalb Gesamtkostenminimierung ein allgemeines Optimierungsziel ist. Da sich Kosten additiv zusammensetzen, bestehen viele Möglichkeiten gegenseitigen Austauschs und für Kompromisse bei der Komponentenauswahl unter Beachtung der spezifizierten Randbedingungen.

8.4 Schlussbemerkung

Leistungsbewertung sowie die wichtigsten Dienstgütekriterien eingebetteter Echtzeitsysteme wurden detailliert erläutert. Nutzer erwarten von einem Echtzeitsystem vorhersehbares und verlässliches Verhalten. Die Kosten eines solchen Systems sind im Kontext des zu automatisierenden Prozesses zu sehen. Wenn für ein eingebettetes System keine Garantien für die Einhaltung spezifizierter Reaktionszeiten gegeben werden können, ist es möglich, dass gewisse und insbesondere gefährliche Situationen nicht zeitgerecht behandelt werden, was den Sicherheitsanforderungen des zu automatisierenden Prozesses zuwider läuft. Normalerweise ist der Preis des eingebetteten Systems im Vergleich zu dem des umgebenden technischen Prozesses sehr gering – insbesondere, wenn man die Folgekosten einer Fehlfunktion mit einbezieht. Daher sind die Kosten eines solchen Rechensystems vor allem in Anbetracht der noch immer weiter fallenden Hardware-Preise eher zu vernachlässigen.

http://www.springer.com/978-3-642-01595-3

Software-Entwicklung für Echtzeitsysteme
Benra, J.T. (Hrsg.)
2009, XVIII, 262 S., Softcover
ISBN: 978-3-642-01595-3