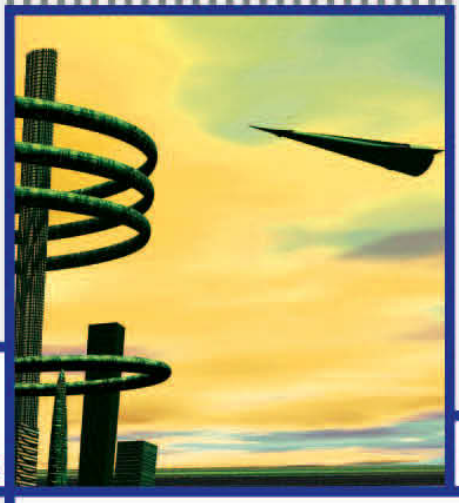
WWW.PEARSON-STUDIUM.DE
CWS
COMPANION WEBSITE

it informatik



Ian Sommerville

Software Engineering

8., aktualisierte Auflage

ADDISON-WESLEY

PEARSON Studium

Kritische Systeme

3

ÜBERBLICK

Lernziele Das Ziel dieses Kapitels ist es, Sie mit dem Konzept des kritischen Systems bekannt zu machen. Es handelt sich dabei um ein System, bei dem die Verlässlichkeit die wichtigste Eigenschaft darstellt. Wenn Sie dieses Kapitel gelesen haben, werden Sie

- verstehen, dass ein Systemausfall in einem kritischen System schwerwiegende menschliche und wirtschaftliche Konsequenzen haben kann
- vier Aspekte der Verlässlichkeit kennen: Verfügbarkeit, Zuverlässigkeit, Betriebssicherheit und Systemsicherheit
- verstehen, dass Sie während der Systementwicklung Fehler vermeiden, während der Systembenutzung Fehler entdecken und beseitigen sowie den von Betriebsausfällen verursachten Schaden begrenzen müssen, um Verlässlichkeit zu erreichen

Softwareausfälle treten relativ häufig auf. In den meisten Fällen verursachen diese Ausfälle Unannehmlichkeiten, aber keine ersten, langfristigen Schäden. Es gibt jedoch einige Systeme, bei denen der Ausfall bedeutende wirtschaftliche Verluste, physischen Schaden oder Gefahren für Gesundheit und Leben von Menschen zur Folge haben kann. Diese Systeme werden *kritische Systeme* genannt. Kritische Systeme sind technische oder soziotechnische Systeme, von denen Menschen bzw. Unternehmen abhängen. Wenn diese Systeme ihren erwarteten Dienst versagen, kann es zu schwerwiegenden Problemen und bedeutenden Verlusten kommen.

Es gibt drei Hauptarten von kritischen Systemen:

1. **Sicherheitskritische Systeme**: Ein System, dessen Ausfall zu Verletzungen, Tod oder schwerwiegenden Umweltschäden führen kann. Ein Beispiel für ein sicherheitskritisches System ist ein Steuerungssystem für eine Chemiefabrik.

2. **Aufgabenkritische Systeme**: Ein System, dessen Ausfall das Scheitern einer zielgerichteten Aktivität zur Folge haben kann. Ein Beispiel für ein aufgabenkritisches System ist ein Navigationssystem für ein Raumfahrzeug.

3. **Geschäftskritische Systeme**: Ein System, dessen Ausfall sehr hohe Kosten für das Unternehmen, das dieses System verwendet, zur Folge haben kann. Ein Beispiel für ein geschäftskritisches System ist das Kundenkontosystem in einer Bank.

Die wichtigste Eigenschaft eines kritischen Systems ist seine Verlässlichkeit. Der Begriff *Verlässlichkeit* wurde von (Laprie 1995) vorgeschlagen, um die damit zusammenhängenden Systemattribute Verfügbarkeit, Zuverlässigkeit, Betriebssicherheit und Systemsicherheit zu behandeln. Wie ich in Abschnitt 3.2 zeigen werde, sind diese Eigenschaften untrennbar miteinander verbunden. Es ist daher sinnvoll, einen zusammenfassenden Begriff für all diese Attribute zu verwenden.

Es gibt mehrere Gründe dafür, dass die Verlässlichkeit die wichtigste Eigenschaft von kritischen Systemen ist:

- *Unzuverlässige oder unsichere Systeme werden oft von den Benutzern abgelehnt.* Wenn Benutzer einem System nicht vertrauen, werden sie sich auch weigern, es zu verwenden. Außerdem können sie sich ebenfalls weigern, Produkte desselben Unternehmens zu erwerben oder zu verwenden, da sie glauben, dass diese ebenso wenig vertrauenswürdig sind wie das vorliegende System.

- *Die Kosten eines Systemausfalls können enorm sein.* Bei einigen Anwendungen, wie zum Beispiel Regelungssysteme für Reaktoren oder Flugleitsysteme, können die Kosten eines Systemausfalls weit höher sein als die Kosten für das Regelungssystem selbst.
- *Unzuverlässige Systeme können Informationsverluste verursachen.* Die Erfassung und Pflege von Daten ist sehr teuer. Diese sind manchmal sogar wertvoller als das Computersystem, auf dem sie verarbeitet werden. Ein großer Anteil an Aufwand und Geld muss zur Sicherung von Daten aufgewendet werden, um wertvolle Daten vor Verfälschung zu schützen.

Die hohen Kosten bei Ausfällen von kritischen Systemen machen es erforderlich, verlässliche Methoden und Techniken für die Entwicklung zu verwenden. Folglich werden kritische Systeme üblicherweise mit Hilfe bewährter Techniken entwickelt anstatt mit neuen Techniken, für die noch keine umfassenden praktischen Erfahrungen vorliegen. Anstatt neuen Techniken und Methoden aufgeschlossen gegenüber zu stehen, sind Entwickler von kritischen Systemen daher von Natur aus eher konservativ. Sie bevorzugen ältere Techniken, deren Stärken und Schwächen bekannt sind, anstatt neue Techniken anzuwenden, die offensichtlich besser sind, doch bei denen die langfristigen Probleme nicht bekannt sind.

Teure Software Engineering-Techniken, die sich für unkritische Systeme nicht rechnen, werden manchmal bei der Entwicklung kritischer Systeme jedoch angewandt. Formale mathematische Methoden der Softwareentwicklung (wie in Kapitel 10 erläutert) wurden z. B. erfolgreich für betriebs- und systemsicherheitskritische Systeme eingesetzt (Hall, 1996; Hall and Chapman, 2002). Ein Grund für den Einsatz dieser formalen Methoden besteht darin, dass der erforderliche Testumfang reduziert wird. Bei kritischen Systeme sind die Kosten für Verifikation und Validierung in der Regel sehr hoch. Sie betragen meist mehr als 50 % der Gesamtentwicklungskosten für das System.

Obwohl einige der Steuerungssysteme komplett automatisch betrieben werden, handelt es sich bei den meisten kritischen Systemen um soziotechnische Systeme, bei denen Menschen den Betrieb der computerbasierten Systeme überwachen und steuern. Die beim Ausfall eines kritischen Systems anfallenden Kosten sind in der Regel so hoch, dass wir Personal brauchen, das mit ungewöhnlichen Situationen umgehen und häufig die Schwierigkeiten beheben kann, wenn etwas nicht richtig funktioniert.

Doch genauso wie Systembediener bei der Beseitigung von Problemen helfen können, können sie genauso Probleme verursachen, wenn sie selbst Fehler machen. Ausfälle des kritischen Systems können in drei „Systemkomponenten“ auftreten:

1. Systemhardware kann auf Grund von Entwurfsfehlern, wegen des Ausfalls von Komponenten durch Herstellungsfehler oder auf Grund von Komponenten, die sich dem Ende ihrer Lebensdauer nähern, ausfallen.

2. Systemsoftware kann auf Grund von Spezifikations-, Entwurfs- oder Implementierungsfehlern ausfallen.

3. Menschliche Bediener des Systems, die bei der Handhabung des Systems Fehler machen. Da Hard- und Software immer verlässlicher werden, liegt die vorrangige Ursache für Systemausfälle wahrscheinlich heutzutage bei Bedienungsfehlern.

Diese Fehler bzw. Ausfälle können sich gegenseitig bedingen. Eine ausgefallene Hardwarekomponente kann bedeuten, dass Systembediener mit einer unerwarteten Situation und Mehrarbeit konfrontiert werden. Sie fühlen sich gestresst – und unter Stress werden häufig Fehler gemacht. Dies kann dazu führen, dass die Software ausfällt, was noch mehr Arbeit für die Bediener bedeutet und noch mehr Stress usw.

Demzufolge ist es besonders wichtig, dass die Entwickler eines kritischen Systems von einer ganzheitlichen Sichtweise des Systems ausgehen, anstatt sich auf einen einzelnen Aspekt des Systems zu konzentrieren. Wenn die Hardware-, Software- und Arbeitsprozesse getrennt entworfen werden, ohne die möglichen Schwächen der anderen Teile des Systems zu berücksichtigen, treten an den Schnittstellen zwischen den verschiedenen Teilen des Systems wahrscheinlich eher Fehler auf.

3.1 Ein einfaches sicherheitskritisches System

Es gibt viele Arten von kritischen computerbasierten Systemen, angefangen bei Steuerungssystemen für Geräte und Maschinen bis hin zu Informations- and E-Commerce-Systemen. Sie könnten ausgezeichnete Fallstudien für ein Buch über Software Engineering darstellen, denn bei der Entwicklung dieser Systeme werden häufig fortschrittliche Software Engineering-Techniken verwendet. Diese Systeme zu verstehen kann jedoch sehr schwierig sein, da Sie die Funktionen und ihre Geltungsbedingungen im Anwendungsbereich, in dem sie arbeiten, kennen müssen.

Folglich handelt es sich bei der Fallstudie eines kritischen Systems, die ich in mehreren Kapiteln dieses Buches verwende, um ein medizinisches System, das die Funktion der Bauchspeicheldrüse (ein inneres Organ) stimuliert. Ich habe dieses Beispiel gewählt, weil wir alle schon einmal mit medizinischen Problemen konfrontiert wurden, und es daher klar ist, warum Sicherheit und Zuverlässigkeit für diese Systemart so wichtig sind. Das gewählte System soll Patienten helfen, die an Diabetes leiden.

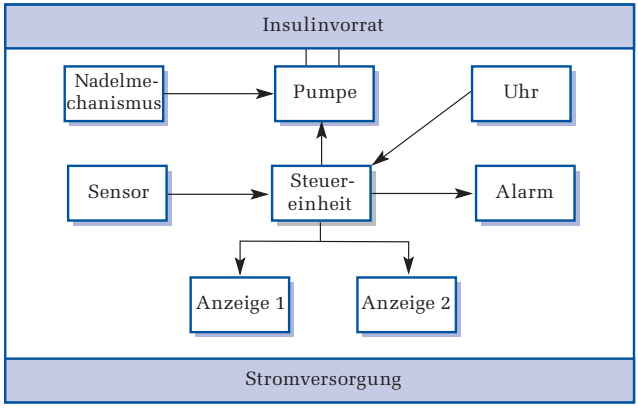


Abbildung 3.1: Aufbau einer Insulinpumpe

Diabetes ist eine relativ verbreitete Krankheit, bei der die menschliche Bauchspeicheldrüse nicht in der Lage ist, ausreichende Mengen eines Hormons namens *Insulin* zu produzieren. Insulin wird im Blut zu Glukose verarbeitet. Die herkömmliche Behandlung von Diabetes beinhaltet regelmäßige Injektionen von genetisch hergestelltem Insulin. Diabetiker messen ihren Blutzuckerspiegel mit einem Messgerät und berechnen dann die Dosis an Insulin, die injiziert werden sollte.

Das Problem der Behandlung besteht darin, dass die Höhe des im Blut enthaltenen Insulins nicht vom Glukosegehalt im Blut abhängt, sondern eine Funktion der Zeit ist, zu der die Insulininjektion vorgenommen wurde. Dies kann zu einem sehr niedrigen Glukosegehalt im Blut (falls zuviel Insulin vorhanden ist) oder zu einem sehr hohen Blutzuckergehalt führen (falls zu wenig Insulin vorhanden ist). Ein niedriger Blutzuckergehalt stellt kurzfristig eine ernst zu nehmende Gefährdung dar, da er zeitweilige Fehlfunktionen des Gehirns und letztlich Bewusstlosigkeit und Tod zur Folge haben kann. Langfristig führt ein andauernd hoher Blutzuckergehalt zu Augenschäden, zur Schädigung der Nieren und zu Herzproblemen.

Die gegenwärtigen Fortschritte bei der Entwicklung von Miniatursensoren bedeuten, dass es jetzt möglich ist, automatisierte Dosiersysteme für Insulin zu entwickeln. Diese überwachen den Blutzuckergehalt und geben, falls erforderlich, eine angemessene Insulindosis aus. Dosiersysteme für Insulin wie dieses existieren bereits für die Behandlung von Krankenhauspatienten. Zukünftig könnte es für viele Diabetiker möglich sein, diese Systeme dauerhaft an ihrem Körper anzubringen.

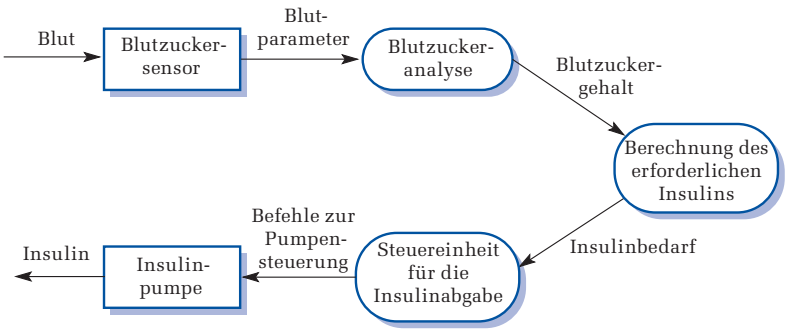


Abbildung 3.2: Datenflussmodell einer Insulinpumpe

Ein softwaregesteuertes Dosiersystem für Insulin könnte mit Hilfe eines Mikrosensors arbeiten, der in den Patienten eingebettet ist, um einige Blutparameter zu messen, die sich proportional zum Zuckergehalt verhalten. Diese werden dann an die Pumpensteuerung gesendet. Diese Steuereinheit berechnet den Zuckergehalt und die benötigte Insulinmenge. Sie sendet dann Signale an eine Miniaturpumpe, um das Insulin mit Hilfe einer dauerhaft befestigten Nadel abzugeben.

In ▶ Abbildung 3.1 sehen Sie die Komponenten und den Aufbau der Insulinpumpe. ▶ Abbildung 3.2 zeigt ein Datenflussmodell, das verdeutlicht, wie ein gemessener Blutzuckergehalt in eine Folge von Pumpensteuerbefehlen umgewandelt wird.

E gibt zwei allgemeine Verlässlichkeitsanforderungen für dieses Insulindosiersystem:

1 Das System soll in der Lage sein, Insulin abzugeben, wenn dies erforderlich ist.

2 Das System soll zuverlässig sein und die genaue Insulinmenge abgeben, um dem aktuellen Blutzuckergehalt entgegenzuwirken.

Ein Ausfall des Systems könnte im Prinzip dazu führen, dass übermäßige Dosen an Insulin abgegeben werden, was das Leben des Patienten gefährden könnte. Es ist also ausgesprochen wichtig, dass keine Überdosis an Insulin abgegeben wird.

3.2 Systemverlässlichkeit

Jedem von uns sind Probleme bei Systemausfällen des Rechners bekannt. Computersysteme stürzen manchmal aus unerfindlichen Gründen ab und liefern nicht die angeforderten Dienste. Programme, die auf diesen Computern laufen, können nicht wie erwartet ausgeführt werden und verfälschen gelegentlich die vom System verwalteten Daten. Wir haben gelernt, mit diesen Ausfällen zu leben, und nur wenige von uns vertrauen den PCs, die wir üblicherweise benutzen, uneingeschränkt.

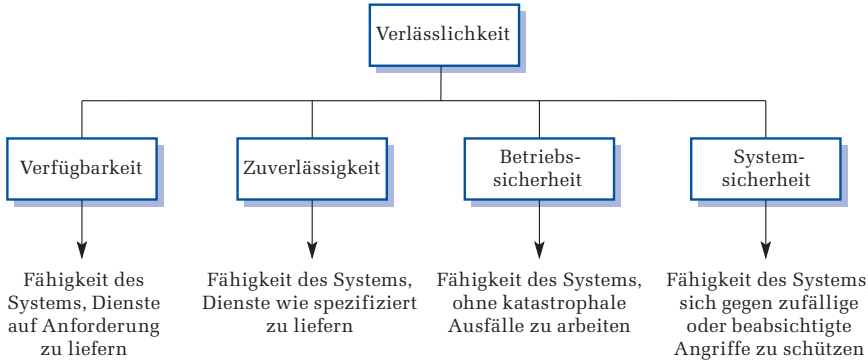


Abbildung 3.3: Aspekte der Verlässlichkeit

Die Verlässlichkeit eines Computersystems ist eine Systemeigenschaft, die mit seiner Vertrauenswürdigkeit gleichgesetzt werden kann. Vertrauenswürdigkeit bezieht sich im Wesentlichen auf den Grad des Benutzervertrauens, d. h. dass das System wie erwartet läuft und das System beim Normalgebrauch nicht abstürzt. Diese Eigenschaft kann nicht in Zahlen ausgedrückt werden, aber wir verwenden relative Begriffe, wie zum Beispiel „nicht verlässlich“, „sehr verlässlich“ und „außerordentlich verlässlich“, um die einem System entgegengebrachten Vertrauensgrade widerzuspiegeln.

Vertrauenswürdigkeit und Nützlichkeit sind selbstverständlich nicht dieselben Dinge. Ich glaube nicht, dass das Textverarbeitungsprogramm, das ich zum Schreiben dieses Buchs verwendet habe, ein sehr verlässliches System ist, aber es ist äußerst nützlich. Mein mangelndes Vertrauen in das System spiegelt sich darin wider, dass ich häufig meine Arbeit speichere und mehrere Sicherungskopien davon aufbewahre. Ich kompensiere die mangelnde Verlässlichkeit des Systems durch Handlungen, die den Schaden begrenzen, der sich aus einem Systemausfall ergibt.

Wie in ▶ Abbildung 3.3 zu sehen, gibt es vier wesentliche Aspekte der Verlässlichkeit:

1. *Verfügbarkeit*: Unter der Verfügbarkeit eines Systems versteht man die Wahrscheinlichkeit, dass das System jederzeit läuft und nützliche Dienste liefern kann.

2. *Zuverlässigkeit*: Unter der Zuverlässigkeit eines Systems versteht man die Wahrscheinlichkeit, dass das System während einer festgelegten Zeitspanne Dienste korrekt und wie vom Benutzer erwartet liefert.

3 *Betriebssicherheit*: Die Betriebssicherheit eines Systems beurteilt die Wahrscheinlichkeit, mit der das System bei Menschen oder seiner Umwelt Schaden anrichtet.

4 *Systemsicherheit*: Die Systemsicherheit eines Systems beurteilt, wie wahrscheinlich es ist, dass das System zufälligen oder beabsichtigten Angriffen widerstehen kann.

Diese komplexen Eigenschaften können in weitere, einfachere Eigenschaften untergliedert werden. Die *Systemsicherheit* umfasst z.B. *Integrität* (stellt sicher, dass die Programme und Daten des Systems nicht beschädigt werden) und *Vertraulichkeit* (stellt sicher, dass Informationen nur von autorisierten Personen abgerufen werden können). *Die Zuverlässigkeit* umfasst *Korrektheit* (stellt sicher, dass die Systemdienste den Spezifikationen entsprechen), *Genauigkeit* (stellt sicher, dass Informationen mit der erforderlichen Genauigkeit geliefert werden) und *Pünktlichkeit* (stellt sicher, dass die Informationen zur angeforderten Zeit geliefert werden).

Die Verlässlichkeitseigenschafen Verfügbarkeit, Systemsicherheit, Zuverlässigkeit und Betriebssicherheit stehen alle in Beziehung zueinander. Ein sicherer Systembetrieb hängt in der Regel davon ab, dass das System verfügbar ist und die Funktionen zuverlässig arbeiten. Ein System kann unzuverlässig werden, weil dessen Daten durch einen Eindringling beschädigt wurden. Denial-of-Service-Angriffe auf ein System zielen darauf ab, die Verfügbarkeit des Systems zu gefährden. Wenn ein System, das als sicher eingestuft wurde, mit einem Virus infiziert wird, kann nicht länger von einem sicheren Betrieb ausgegangen werden. Aufgrund dieser engen Beziehungen wurde der Begriff der Systemverlässlichkeit als eine umfassende Eigenschaft eingeführt.

Wie diese vier Hauptbereiche können auch andere Systemeigenschaften unter dem Aspekt der Verlässlichkeit betrachtet werden:

1 *Reparierfähigkeit:* Systemausfälle sind unvermeidlich, doch die dadurch verursachten Unterbrechungen können minimiert werden, wenn das System rasch repariert werden kann. Damit dies auch gelingt, muss es möglich sein, das Problem zu diagnostizieren, auf die ausgefallenen Komponenten zuzugreifen und Änderungen vorzunehmen, um die Komponente zu reparieren. Die Reparier- und Korrigierfähigkeit von Software wird verbessert, wenn die das System verwendende Organisation auf den Quellcode zugreifen kann und über Kenntnisse verfügt, diesen zu verändern. Leider wird dies immer seltener möglich sein, da wir bei der Systementwicklung zunehmend auf Black Box-Komponenten von Drittanbietern zurückgreifen (siehe Kapitel 19).

2 *Wartbarkeit:* Wenn Systeme verwendet werden, entstehen neue Anforderungen. Es ist wichtig, den Nutzen eines Systems durch Veränderungen aufrechtzuerhalten, damit es diesen neuen Anforderungen gerecht wird. Wartbare Software ist Software, die wirtschaftlich so angepasst werden kann, dass sie neuen Anforderungen gerecht wird und bei der eine nur geringe Wahrscheinlichkeit besteht, dass die vorgenommenen Änderungen zu neuen Fehlern im System führen.

3 *Überlebensfähigkeit:* Ein sehr wichtiges Merkmal von Systemen, die auf dem Internet basieren, ist die Überlebensfähigkeit, die eng mit der Systemsicherheit und der Verfügbarkeit verbunden ist (Ellison et al., 1999). Unter der Überlebensfähigkeit versteht man die Fähigkeit eines Systems, die Lieferung von Diensten fortzusetzen, während es angegriffen wird und (möglicherweise auch) wenn ein

Teil des Systems nicht funktionsfähig ist. Die Arbeit an der Überlebensfähigkeit konzentriert sich darauf, Schlüsselkomponenten des Systems zu bestimmen und abzusichern, dass diese immer noch einen minimalen Dienst liefern können. Drei Strategien kommen zum Einsatz, um die Überlebensfähigkeit zu steigern: zum einen der Widerstand gegen Angriffe, dann die Erkennung von Angriffen sowie schließlich die Wiederherstellung nach einem Schaden, der durch einen Angriff verursacht wurde (Ellison et al., 1999; Ellison et al., 2002).

4 *Fehlertoleranz:* Diese Eigenschaft kann als Teil der Benutzerfreundlichkeit (wird in Kapitel 16 erläutert) angesehen werden und spiegelt wider, inwieweit das System Eingabefehler des Benutzers verhindert bzw. toleriert. Wenn es zu Benutzerfehlern kommt, sollte das System diese Fehler so weit wie möglich erkennen und diese entweder automatisch korrigieren oder den Benutzer auffordern, die Daten erneut einzugeben.

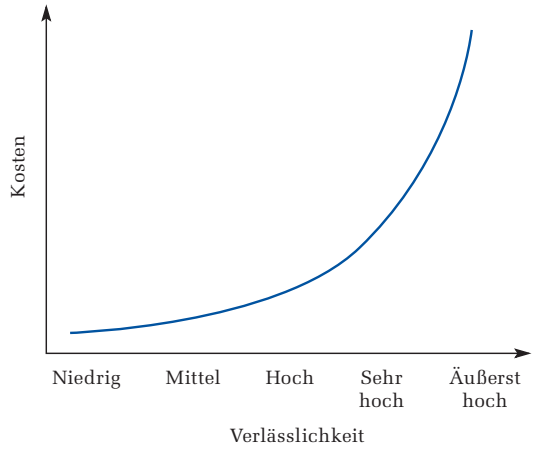


Abbildung 3.4: Kosten-/Verlässlichkeitskurve

Da Verfügbarkeit, Zuverlässigkeit, Betriebs- und Systemsicherheit die grundlegenden Verlässlichkeitseigenschafen darstellen, werde ich in diesen und in den weiteren Kapiteln, die die Spezifikation kritischer Systeme (Kapitel 9), Entwicklung kritischer Systeme (Kapitel 20) und die Validierung kritischer Systeme (Kapitel 24) beinhalten, näher darauf eingehen.

Natürlich können diese Verlässlichkeitseigenschaften nicht auf alle Systeme angewendet werden. Bei dem in Abschnitt 3.1 vorgestellten Insulindosiersystem sind die wichtigsten Eigenschaften die Verfügbarkeit (es muss bei Bedarf funktionieren), Zuverlässigkeit (es muss eine genaue Dosis Insulin abgeben) und Betriebssicherheit (es darf niemals eine die Gesundheit gefährdende Dosis an Insulin abgeben). Die Systemsicherheit spielt in diesem Beispiel kaum eine Rolle, da das Dosiersystem keine vertraulichen Informationen enthält und nicht vernetzt ist, so dass es zu keinem bösartigen Angriff kommen kann.

Entwickler müssen in der Regel zwischen der Systemleistung und der Systemverlässlichkeit abwägen. Im Allgemeinen kann ein hoher Grad an Verlässlichkeit nur auf Kosten der Systemleistung erreicht werden. Verlässliche Software beinhaltet zusätzlichen, oftmals redundanten Code, um die notwendige Überprüfung auf außergewöhnli-

che Systemzustände und die Wiederherstellung nach Systemausfällen durchführen zu können. Dadurch sinkt die Verarbeitungsgeschwindigkeit und der von der Software benötigte Speicher steigt. Außerdem steigen die Kosten der Systementwicklung beträchtlich.

Wegen zusätzlicher Entwurfs-, Implementierungs- und Validierungskosten kann die Verlässlichkeit eines Systems die Entwicklungskosten erheblich steigern. Insbesondere sind die Validierungskosten bei kritischen Systemen hoch. Neben der Überprüfung, ob das System die Anforderungen erfüllt, muss anhand des Validierungsprozesses einer externen Behörde, z. B. der Luftfahrtbehörde, möglicherweise bewiesen werden, dass das System verlässlich ist.

▶ Abbildung 3.4 zeigt die Beziehungen zwischen Kosten und stufenweiser Verbesserung der Verlässlichkeit. Je höher der von Ihnen geforderte Grad der Verlässlichkeit ist, desto mehr müssen Sie für Tests ausgeben, um zu überprüfen, ob Sie diesen Grad erreicht haben. Wegen des exponentiellen Verlaufs der Kosten-/Verlässlichkeitskurve lässt sich nicht beweisen, dass ein System hundertprozentig verlässlich ist, da die Kosten dafür ins Unendliche gehen würden.

3.3 Verfügbarkeit und Zuverlässigkeit

Systemverfügbarkeit und Systemzuverlässigkeit sind eng miteinander verbundene Eigenschaften, die beide als zahlenmäßige Wahrscheinlichkeiten dargestellt werden können. Unter der Zuverlässigkeit eines Systems versteht man die Wahrscheinlichkeit, dass das System Dienste entsprechend den Spezifikationen bereitstellt. Unter der Verfügbarkeit eines Systems versteht man die Wahrscheinlichkeit, dass das System bereit ist, die Dienste für Benutzer bereitzustellen, wenn diese sie anfordern.

Obwohl die beiden Eigenschaften eng miteinander verbunden sind, können Sie nicht davon ausgehen, dass zuverlässige Systeme stets zur Verfügung stehen und dass verfügbare Systeme stets zuverlässig sind. Einige Systeme können eine hohe Anforderung an die Verfügbarkeit und eine wesentlich geringere Anforderung an die Zuverlässigkeit stellen. Wenn Benutzer einen ununterbrochenen Dienst erwarten, dann sind die Anforderungen an die Verfügbarkeit hoch. Wenn die Folgen eines Ausfalls minimal sind und sich das System von diesen Ausfällen schnell erholen kann, kann das System geringe Anforderungen an die Zuverlässigkeit stellen.

Ein Beispiel für ein System, bei dem die Verfügbarkeit wichtiger als die Zuverlässigkeit ist, ist eine Telefonzentrale. Benutzer erwarten ein Freizeichen, wenn sie einen Telefonhörer abnehmen, und damit werden hohe Anforderungen an die Verfügbarkeit des Systems gestellt. Falls ein Systemfehler den Ausfall einer Verbindung zur Folge hat, ist dies oft korrigierbar. Vermittlungsschalter beinhalten für gewöhnlich technische Reparaturmöglichkeiten, die das System zurücksetzen und den Versuch wiederholen können, eine Verbindung aufzubauen. Dies kann sehr schnell erfolgen, so dass der Telefonbenutzer nicht einmal unbedingt bemerkt, dass ein Fehler aufgetreten ist. In solch einer Situation ist die Verfügbarkeit und nicht die Zuverlässigkeit die Schlüsselanforderung an diese Systeme.

Eine weitere Unterscheidung zwischen diesen Eigenschaften wird dadurch getroffen, dass die Verfügbarkeit nicht einfach vom System selbst abhängt, sondern auch von der Zeit, die benötigt wird, um die Fehler zu beseitigen, die die Verfügbarkeit des Systems behindern. Falls System A einmal im Jahr ausfällt, System B dagegen einmal im Monat, ist A zuverlässiger als B. Nehmen wir einmal an, dass System A drei Tage

für einen Neustart benötigt, wohingegen System B dafür zehn Minuten braucht. Über das Jahr gesehen ist die Verfügbarkeit von System B (120 Minuten Ausfallzeit) viel besser als die von System A (4.320 Minuten Ausfallzeit).

Zuverlässigkeit und Verfügbarkeit eines Systems können wie folgt präziser definiert werden:

- *Zuverlässigkeit:* Ist die Wahrscheinlichkeit eines fehlerfreien Betriebs zu einem bestimmten Zweck während einer bestimmten Zeitspanne in einer gegebenen Umgebung
- *Verfügbarkeit:* Die Wahrscheinlichkeit, dass ein System zu einem bestimmten Zeitpunkt in Betrieb und in der Lage ist, die angeforderten Dienste zu liefern.

Eines der praktischen Probleme bei der Entwicklung zuverlässiger Systeme besteht darin, dass unsere intuitiven Annahmen bezüglich der Zuverlässigkeit und Verfügbarkeit oftmals weiter gefasst sind als diese eingeschränkten Definitionen. Die Definition von *Zuverlässigkeit* legt fest, dass die Umgebung, in der das System verwendet wird, und der Zweck, für das es verwendet werden soll, berücksichtigt werden müssen. Wenn Sie die Zuverlässigkeit eines Systems in einer Umgebung messen, können Sie nicht davon ausgehen, dass die Zuverlässigkeit in einer anderen Umgebung gleich groß ist, wenn das System dort anders genutzt wird.

Messen Sie z. B. die Zuverlässigkeit eines Textverarbeitungsprogramms in einer Büroumgebung, in der sich die meisten Benutzer nicht für die Arbeitsweise der Software interessieren. Sie folgen den Bedienungsanweisungen und versuchen nicht, mit diesem System zu experimentieren. Wenn Sie die Zuverlässigkeit desselben Systems in einer Universitätsumgebung messen, dann kann diese ganz anders ausfallen. Dort testen die Studenten möglicherweise die Grenzen des Systems aus und verwenden es auf ungewöhnliche Weise. Dies kann zu Systemausfällen führen, die in einer stärker eingeschränkten Büroumgebung nicht auftreten würden.

Menschliche Wahrnehmungen und Anwendungsmuster spielen ebenfalls eine wichtige Rolle. Stellen Sie sich beispielsweise eine Situation vor, bei der im Scheibenwischersystem eines Autos ein Fehler auftritt, der zur Folge hat, dass bei starkem Regen die Scheibenwischer gelegentlich nicht mehr korrekt arbeiten. Die vom Fahrer wahrgenommene Zuverlässigkeit des Systems hängt davon ab, wo der Fahrer lebt und sein Auto benutzt. Ein Fahrer in Hamburg (feuchtes Klima) wird wahrscheinlich mehr von diesem Ausfall betroffen sein als ein Fahrer in Mallorca (trockenes Klima). Der Fahrer in Hamburg fasst das System als unzuverlässig auf, während der Fahrer in Mallorca dieses Problem möglicherweise nie bemerken wird.

Eine weitere Schwierigkeit mit diesen Definitionen besteht darin, dass sie die Schwere des Fehlers oder die Konsequenzen der Nichtverfügbarkeit außer Acht lassen. Menschen machen sich von Natur aus eher über Systemausfälle Gedanken, die ernsthafte Konsequenzen haben, und ihre Wahrnehmung der Zuverlässigkeit des Systems wird von diesen Konsequenzen beeinflusst. Nehmen wir einmal an, dass ein Fehler im Zündungsmechanismus der Motorensoftware bewirkt, dass das Auto unmittelbar nach dem Start ausgeht, aber nach einem Neustart, der das Zündungsproblem behebt, korrekt läuft. Der normale Betrieb des Autos wird dadurch jedoch nicht beeinträchtigt, und viele Fahrer würden sich keine Gedanken darüber machen, ob eine Reparatur erforderlich ist. Im Gegensatz dazu sind die meisten Fahrer jedoch der Meinung, dass ein Motor, der z. B. einmal im Monat bei hoher Geschwindigkeit ausgeht, sowohl unzuverlässig als auch unsicher ist und repariert werden muss.

Eine strenge Definition von Zuverlässigkeit setzt die Systemimplementierung mit ihrer Spezifikation in Beziehung. Das bedeutet, dass sich das System zuverlässig verhält, falls sein Verhalten mit dem in der Spezifikation definierten übereinstimmt. Ein allgemeiner Grund für wahrgenommene Unzuverlässigkeit ist jedoch, dass die Systemspezifikation nicht den Erwartungen der Systembenutzer entspricht. Leider sind viele Spezifikationen unvollständig oder inkorrekt, und es wird den Softwareentwicklern überlassen, zu interpretieren, wie sich das System verhalten sollte. Da sie keine Experten des Anwendungsgebiets sind, kann es passieren, dass sie nicht das von den Benutzern erwartete Verhalten implementieren.

Zuverlässigkeit und Verfügbarkeit werden durch Systemausfälle gefährdet. Dieser Ausfall kann entstehen, weil ein Dienst nicht angeboten wird, ein Dienst nicht so wie spezifiziert geliefert wird oder die Lieferung eines Dienstes auf unsichere Weise erfolgt. Einige dieser Ausfälle resultieren aus Spezifikationsfehlern oder Ausfällen miteinander verbundener Systeme, wie zum Beispiel von Telekommunikationssystemen. Viele Ausfälle sind jedoch auf ein fehlerhaftes Systemverhalten zurückzuführen, für das es im System bestimmte Ursachen gibt. Bei Ausführungen zum Thema Zuverlässigkeit ist es hilfreich, zwischen den Begriffen *Fehlerursache* (fault), *Fehlerzustand* (error) und *Ausfall* (failure) zu unterscheiden. Ich habe diese Begriffe in ▶ Abbildung 3.5 definiert.

Menschliche Fehler müssen nicht unweigerlich zu Systemausfällen führen. Die Fehlerursache kann in Teilen des Systems auftreten, die niemals verwendet werden. Fehlerursachen haben nicht unbedingt Systemfehlerzustände zur Folge, da der ursächliche Zustand temporär sein und korrigiert werden kann, ehe ein fehlerhaftes Verhalten auftritt. Systemfehlerzustände führen nicht unbedingt zu Systemausfällen, da das Verhalten ebenfalls temporär sein kann und keine beobachtbare Auswirkung verursachen muss oder das System einen Schutz enthält, der garantiert, dass das fehlerhafte Verhalten erkannt und korrigiert wird, ehe die Systemdienste betroffen sind.

Begriff	Beschreibung
Fehlfunktion oder Systemausfall (failure)	Ein Ereignis, das zu einem Zeitpunkt auftritt, wenn das System einen Dienst nicht so liefert, wie vom Benutzer erwartet
Systemfehlerzustand (error)	Fehlerhaftes Systemverhalten, bei dem das Verhalten des Systems nicht mit den Erwartungen der Systembenutzer übereinstimmt
Systemfehlerursache (fault)	Merkmal eines Softwaresystems, das zu einem Systemfehlerzustand führen kann. Wenn z. B. eine Variable nicht initialisiert wird, kann dies dazu führen, dass die Variable einen falschen Wert besitzt, wenn sie verwendet wird.
Menschliches Versagen	Menschliches Verhalten, das zu Fehlern im System führt

Abbildung 3.5: Terminologie zur Zuverlässigkeit

Diese in ▶ Abbildung 3.5 dargestellte Unterscheidung zwischen den Begriffen hilft uns bei der Bestimmung von drei einander ergänzenden Ansätzen, die verwendet werden, um die Zuverlässigkeit eines Systems zu erhöhen:

1. **Fehlervermeidung**: Es werden Entwicklungstechniken verwendet, die entweder die Möglichkeit von Fehlern minimieren und/oder Fehler finden, ehe sie Systemfehlerursachen zur Folge haben. Beispiele für solche Techniken sind die Vermeidung fehleranfälliger Programmiersprachenkonstrukte, wie zum Beispiel von Zeigern, und die Verwendung der statischen Analyse, um Programmanomalien zu entdecken.

2 **Fehlerentdeckung und -entfernung**: Hier werden Verifikations- und Validierungstechniken eingesetzt, die die Chance erhöhen, dass Fehlerursachen entdeckt und entfernt werden, ehe das System benutzt wird. Ein Beispiel für eine Technik zum Erkennen von Fehlerursachen ist das systematische Testen des Systems sowie die Fehlerbeseitigung.

3 **Fehlertoleranz**: Hierbei handelt es sich um Techniken, die garantieren, dass Systemfehlerursachen keine Systemfehlerzustände zur Folge haben, oder die absichern, dass Systemfehlerzustände nicht zu Systemausfällen führen. Beispiele für Fehlertoleranztechniken sind die Einbeziehung selbstprüfender Mechanismen in einem System und der Gebrauch redundanter Systemmodule.

Die Entwicklung fehlertoleranter Systeme wird in Kapitel 20 behandelt. Dort erläutere ich auch einige Techniken zur Fehlervermeidung. Ich erläutere vorgehensbasierte Verfahren für die Fehlervermeidung in Kapitel 27 und die Fehlerentdeckung in den Kapiteln 22 und 23.

Softwarefehlerursachen resultieren in Softwareausfällen, wenn der fehlerhafte Code mit einer Eingabemenge ausgeführt wird, welcher die Fehlerursache zum Tragen bringt. Bei den meisten Eingaben funktioniert der Code richtig. In ▶ Abbildung 3.6, die von Littlewood (Littlewood, 1990) abgeleitet ist, sehen Sie ein Softwaresystem als Abbildung einer Eingabe- in eine Ausgabemenge. Bei einer gegebenen Eingabe oder Eingabesequenz antwortet das Programm mit einer entsprechenden Ausgabe. Bei einer gegebenen Eingabe einer URL erzeugt ein Web-Browser eine Ausgabe in Form einer Anzeige der angeforderten Webseite.

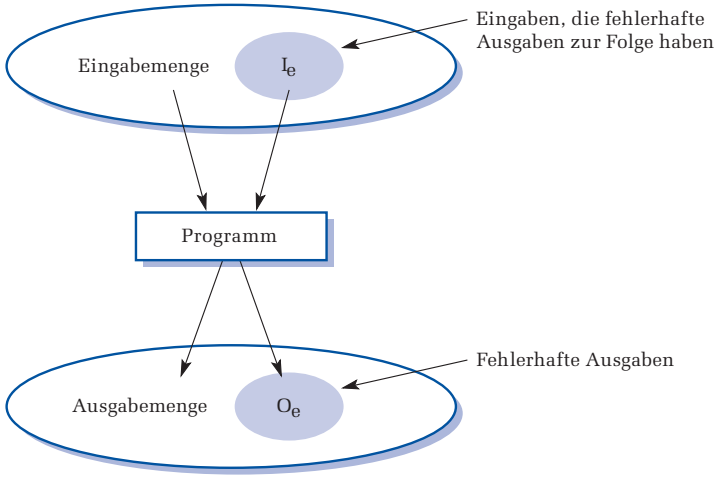


Abbildung 3.6: Ein System als Abbildung von einer Eingabe- in eine Ausgabemenge

Einige dieser Eingaben (in ▶ Abbildung 3.6 als schattierte Ellipse gezeigt) verursachen fehlerhafte Ausgaben. Die Zuverlässigkeit der Software bezieht sich auf die Wahrscheinlichkeit, dass bei einer bestimmten Ausführung des Programms die Systemeingabe Teil einer Eingabemenge ist, die eine fehlerhafte Ausgabe erzeugt. Wenn eine Eingabe, die eine fehlerhafte Ausgabe verursacht, mit einem häufig verwendeten Programmteil verbunden ist, wird es häufig zu Ausfällen kommen. Wenn die Eingabe jedoch mit einem nur selten verwendeten Code verbunden ist, werden die Benutzer die Ausfälle kaum bemerken.

Jeder Benutzer eines Systems verwendet die Eingabe auf andere Weise. Fehlerursachen, die die Zuverlässigkeit des Systems für einen Benutzer beeinflussen, werden bei einer anderen Anwendungsart eines anderen Benutzers möglicherweise nie aufgedeckt (▶ Abbildung 3.7). In ▶ Abbildung 3.7 entspricht die Menge fehlerhafter Eingaben der schattierten Ellipse in ▶ Abbildung 3.6. Die von Benutzer 2 erzeugte Eingabemenge schneidet sich mit der Fehler verursachenden Eingabemenge. Benutzer 2 wird daher einige Systemausfälle hinnehmen müssen. Benutzer 1 und Benutzer 3 verwenden jedoch nie Eingaben der Fehlerursachenmenge. Für sie wird die Software also immer zuverlässig sein.

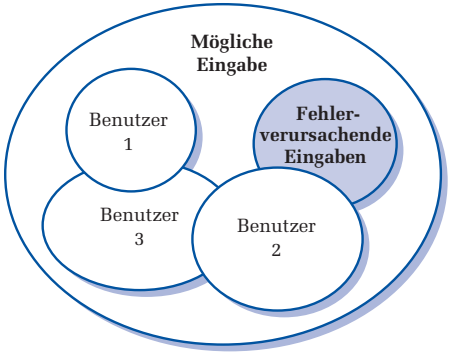


Abbildung 3.7: Arten der Softwareverwendung

Die Gesamtzuverlässigkeit eines Programms hängt daher größtenteils von der Anzahl der Eingaben ab, die bei den meisten Benutzern während des normalen Systemgebrauchs eine fehlerhafte Ausgabe zur Folge haben. Softwarefehler, die nur in Ausnahmesituationen auftreten, haben auf die Zuverlässigkeit des Systems lediglich einen geringen Einfluss. Die Entfernung von Softwarefehlern aus Teilen des Systems, die selten verwendet werden, beeinflusst die von Systembenutzern wahrgenommene Zuverlässigkeit des Systems kaum. Mills et al. (Mills et al., 1987) fanden heraus, dass es nur zu einer Zunahme der Zuverlässigkeit um 3 % kam, nachdem 60 % der bekannten Fehler in ihrer Software entfernt worden waren. In einer Studie von IBM-Softwareprodukten hat Adams (Adams, 1984) festgestellt, dass viele Fehlerursachen in den Produkten sehr wahrscheinlich erst nach Hunderten oder Tausenden von Monaten des Produktgebrauchs zu Ausfällen führen würden.

Benutzer eines soziotechnischen Systems passen sich einer als fehlerhaft bekannten Software an, indem sie sich gegenseitig darüber informieren, wie diese Fehler umgangen werden können. Sie umgehen möglichst Eingaben, von denen sie wissen, dass diese Probleme verursachen, so dass es niemals zu Programmfehlern kommt. Weiterhin umgehen erfahrene Benutzer oftmals Schwachstellen, die bekannt dafür sind, Softwareausfälle zu verursachen. Sie vermeiden bewusst die Verwendung von Systemfunktionen, bei denen bekannt ist, dass sie Probleme verursachen. Ich vermeide bestimmte Funktionen, wie z. B. die automatische Nummerierung im Textverarbeitungssystem, mit dessen Hilfe ich dieses Buch geschrieben habe. Die Beseitigung von Fehlerursachen in diesen Funktionen macht für die Burteilung der Zuverlässigkeit durch die Benutzer möglicherweise keinen praktischen Unterschied.

3.4 Betriebssicherheit

Unter sicherheitskritischen Systemen versteht man Systeme, bei denen ein kontinuierlicher sicherer Systembetrieb von entscheidender Bedeutung ist. Das heißt, das System darf *niemals* bei Menschen oder der Systemumgebung Schaden anrichten. Beispiele für sicherheitskritische Systeme sind Steuer- und Überwachungssysteme in Flugzeugen, Prozessleitsysteme in Chemie- oder Pharmaziefabriken sowie Steuersysteme für Automobile.

Die Hardwaresteuerung von sicherheitskritischen Systemen ist einfacher zu implementieren und zu analysieren als die Softwaresteuerung. Heutzutage bauen wir allerdings Systeme von solch einer Komplexität, dass diese nicht allein von der Hardware gesteuert werden können. Einiges an Softwaresteuerung ist unentbehrlich wegen der Notwendigkeit, eine große Anzahl von Sensoren und Aktoren mit komplexen Regelungsalgorithmen zu verwalten. Ein Beispiel für diese Komplexität findet sich bei modernen Kampfflugzeugen, die aerodynamisch instabil sind. Sie benötigen eine ständige softwaregesteuerte Einstellung ihrer Tragflächen, um nicht abzustürzen.

Sicherheitskritische Software wird in zwei Klassen eingeteilt:

1. *Primäre sicherheitskritische Software*: Dies ist Software, die als Steuereinheit in ein System eingebettet wird. Die Funktionsstörung einer solchen Software kann eine Funktionsstörung der Hardware verursachen, die Verletzungen bei Menschen oder Umweltschäden zur Folge hat. Ich konzentriere mich auf diese Softwareart.

2. *Sekundäre sicherheitskritische Software*: Dies ist Software, die indirekt Verletzungen zur Folge haben kann. Ein Beispiel für ein solches System sind computergestützte Entwicklungs- und Entwurfssysteme, deren Funktionsstörung zu einem Entwurfsfehler bei dem zu entwerfenden Objekt führt. Wenn das entworfene System nicht ordnungsgemäß funktioniert, kann dieser Fehler zu Verletzungen bei Personen führen. Ein anderes Beispiel eines sekundären sicherheitskritischen Systems ist eine medizinische Datenbank, die Details über Medikamente enthält, welche an Patienten verabreicht werden. Fehler in diesem System können zu inkorrekten Dosierungen von verabreichten Medikamenten führen.

Die Zuverlässigkeit und Betriebssicherheit von Systemen sind verwandte, aber verschiedene Merkmale der Verlässlichkeit. Natürlich sollte ein sicherheitskritisches System dahingehend zuverlässig sein, dass es mit seiner Spezifikation übereinstimmt und ohne Ausfälle läuft. Es kann fehlertolerante Funktionen beinhalten, um sogar bei auftretenden Fehlern ununterbrochene Dienste zur Verfügung zu stellen. Fehlertolerante Systeme sind jedoch nicht unbedingt sicher. Die Software kann versagen und ein Systemverhalten hervorrufen, das einen Unfall zur Folge hat.

Außer der Tatsache, dass wir nie 100-prozentig sicher sein können, dass ein Softwaresystem fehlerfrei und fehlertolerant ist, gibt es eine Reihe weiterer Ursachen, warum zuverlässige Softwaresysteme nicht unbedingt sicher sind:

- Die Spezifikation kann unvollständig sein, d. h., sie beschreibt nicht das erforderliche Systemverhalten in einigen kritischen Situationen. Ein großer Anteil an Funktionsstörungen eines Systems (Nakajo und Kume, 1991; Lutz, 1993) resultiert aus Spezifikations- und nicht aus Entwurfsfehlern. In einer Studie über Fehler in eingebetteten Systemen folgert Lutz:

  ... Schwierigkeiten mit Anforderungen sind die Hauptursache der sicherheitsrelevanten Softwarefehlerzustände, die bis zur Integration und zum Systemtest fortbestanden.“

- Funktionsstörungen der Hardware bewirken, dass sich das System unvorhersehbar verhält und die Software mit einer unerwarteten Umgebung konfrontiert wird. Kurz vor dem Ausfall verhalten sich Komponenten unberechenbar und erzeugen Signale, die sich außerhalb der Bereiche befinden, mit denen die Software korrekt umgehen kann.
- Der Bediener eines Systems können Eingaben machen, die alleine nicht inkorrekt sind, die aber in bestimmten Situationen zu einer Fehlfunktion des Systems führen können. Ein anekdotenhaftes Beispiel dazu ist, dass ein Mechaniker die Geräteverwaltungssoftware eines Flugzeugs anwies, das Fahrwerk einzufahren. Die Software führte die Anweisungen des Mechanikers umgehend aus. Leider befand sich das Flugzeug zu dieser Zeit am Boden – das System hätte den Befehl nur dann zulassen dürfen, wenn sich das Flugzeug in der Luft befunden hätte.

Es hat sich mittlerweile ein ganz spezielles Vokabular herausgebildet, um sicherheitskritische Systeme zu beschreiben, und es ist wichtig, dass man die speziellen Begriffe versteht. In ▶ Abbildung 3.8 finden Sie einige Definitionen, die ich aus Begriffen abgeleitet habe, die ursprünglich von Leveson (1985) definiert wurden.

Der Schlüssel zum Erreichen von Betriebssicherheit liegt darin, entweder sicherzustellen, dass keine Störfälle auftreten oder dass die Konsequenzen eines Störfalls minimal sind. Dies kann auf drei sich ergänzende Arten erreicht werden:

1. *Gefahrenvermeidung:* Das System wird so entworfen, dass Gefahren vermieden werden. Ein Schneidesystem, das vom Bediener verlangt, gleichzeitig zwei Knöpfe zu drücken, um die Maschine zu bedienen, vermeidet die Gefahr, dass sich die Hände des Bedieners in der Nähe des Schneideblatts befinden.

2. *Gefahrenerkennung* und -beseitigung: Das System wird so entworfen, dass Gefahren erkannt und beseitigt werden können, ehe es zu einem Störfall kommt. Eine Chemiefabrik kann zum Beispiel Überdruck feststellen und ein Überdruckventil öffnen, um den Druck zu reduzieren, ehe eine Explosion auftritt.

3. *Schadensbegrenzung:* Das System kann Schutzfunktionen beinhalten, die den aus einem Störfall resultierenden Schaden minimieren. Ein Flugzeugmotor enthält zum Beispiel normalerweise einen automatischen Feuerlöscher. Wenn ein Feuer auftritt, kann es oft unter Kontrolle gebracht werden, ehe es für die Passagiere oder die Crew eine Gefahr darstellt.

Begriff	Beschreibung
Unfall (Accident, Mishap)	Ein nicht geplantes Ereignis oder eine Folge von Ereignissen, die den Tod oder die Verletzung von Menschen, Sach- oder Umweltschäden zur Folge haben. Eine computergesteuerte Maschine, die ihren Bediener verletzt, ist ein Beispiel für einen Unfall.
Gefahr (Hazard)	Ein Umstand, der das Potenzial besitzt, einen Unfall zu verursachen oder dazu beizutragen. Der Ausfall eines Sensors, der ein Hindernis vor einer Maschine erkennt, ist ein Beispiel für eine Gefahr.

Abbildung 3.8: Terminologie zur Betriebssicherheit

Begriff	Beschreibung
Schaden (Damage)	Ausmaß des Verlustes, der aus einem Zwischenfall resultiert. Ein Schaden kann von vielen Menschen, die durch einen Unfall getötet werden, bis zu einer unbedeutenden Verletzung oder Sachschäden reichen.
Gefahrenausmaß (Hazard Severity)	Bewertung des schlimmsten Schadens, der sich aus einer bestimmten Gefahr ergeben könnte. Das Gefahrenausmaß kann von katastrophal (Tod vieler Menschen) bis hin zu unbedeutend (kleinere Schäden) reichen.
Gefahrenwahrscheinlichkeit (Hazard Probability)	Wahrscheinlichkeit für das Auftreten von Ereignissen, die eine Gefahr implizieren. Wahrscheinlichkeitswerte sind scheinbar willkürlich, reichen aber von *wahrscheinlich* (eine Chance von 1:100, dass ein Ereignis auftritt) bis zu unplausibel (keine vorstellbare Situation, bei der die Gefahr auftreten könnte).
Risiko (Risk)	Dies ist ein Maß der Wahrscheinlichkeit, dass das System einen Unfall verursachen wird. Das Risiko wird bewertet, indem die Gefahrenwahrscheinlichkeit, das Gefahrenausmaß und die Wahrscheinlichkeit, dass eine Gefahr zu einem Unfall führt, in Betracht gezogen werden.

Abbildung 3.8: Terminologie zur Betriebssicherheit (Forts.)

Im Allgemeinen treten Unfälle auf, wenn mehrere Dinge zum selben Zeitpunkt schief gehen. Eine Analyse schwer wiegender Unfälle (Perrow, 1984) hat ergeben, dass nahezu alle Unfälle auf eine Kombination von Funktionsstörungen und nicht auf einzelne Ausfälle zurückzuführen waren. Die unerwartete Kombination führte zu Interaktionen, die einen Systemausfall zur Folge hatten. Perrow hat außerdem bemerkt, dass es unmöglich ist, alle möglichen Kombinationen von Funktionsstörungen eines Systems vorherzusagen und dass Unfälle ein unumgänglicher Bestandteil bei der Verwendung komplexer Systeme sind. Software neigt dazu, die Systemkomplexität zu erhöhen, so dass die Verwendung einer Softwaresteuerung die Wahrscheinlichkeit eines Systemunfalls vergrößern *kann*.

Softwaresteuerung und -überwachung können die Betriebssicherheit von Systemen jedoch auch verbessern. Softwaregesteuerte Systeme können einen weiteren Bereich von Zuständen überwachen als elektromechanische Systeme. Außerdem lassen sie sich relativ einfach anpassen. Sie beinhalten die Verwendung von Computerhardware, die selbst eine relativ hohe Zuverlässigkeit besitzt und klein an Umfang und leichtgewichtig ist. Softwaregesteuerte Systeme können hoch entwickelte Sicherheitssperren zur Verfügung stellen. Sie können Steuerungsstrategien unterstützen, welche die erforderliche Zeit verringern, die Menschen in gefahrenreichen Umgebungen verbringen müssen. Obwohl die Softwaresteuerung mehr Anlässe vorsehen kann, bei denen ein System nicht richtig funktioniert, erlaubt sie dennoch eine bessere Überwachung und einen besseren Schutz und kann daher die Betriebssicherheit des Systems verbessern.

In allen Fällen ist es jedoch wichtig, bei der Systemsicherheit eine gewisse Verhältnismäßigkeit zu wahren. Es ist unmöglich, ein System zu 100 % sicher zu machen, und die Gesellschaft muss entscheiden, ob die Vorteile, die sich aus der Verwendung moderner Technologien ergeben, die Folgen eines gelegentlichen Unfalls aufwiegen. Es ist ebenfalls eine gesellschaftliche und politische Entscheidung, wie begrenzte nationale Ressourcen verfügbar gemacht werden sollen, um die Gefahren für die Bevölkerung als Ganzes zu verringern.

3.5 Systemsicherheit

Systemsicherheit ist ein Systemattribut, das die Fähigkeit des Systems widerspiegelt, sich selbst gegen externe Angriffe zu schützen, die auf Zufall oder Absicht zurückzuführen sind. Die Systemsicherheit wird zunehmend wichtiger, da immer mehr Systeme mit dem Internet verbunden sind. Internet-Verbindungen bieten zusätzliche Systemfunktionalität (Kunden sind zum Beispiel in der Lage, direkt auf ihre Bankkonten zuzugreifen), bedeuten aber auch, dass das System von Menschen mit feindlichen Absichten angegriffen werden kann. Außerdem bringt es die Internet-Verbindung mit sich, dass Details über spezifische Anfälligkeiten von Systemen einfach verbreitet werden können, so dass mehr Menschen in der Lage sind, das System anzugreifen. Genauso kann die Verbindung jedoch die Verteilung von Systemkorrekturen beschleunigen, um diese Anfälligkeiten zu beheben.

Beispiele für solche Angriffe sind Computerviren, die unberechtigte Verwendung von Systemdiensten, die unberechtigte Modifikation des Systems oder seiner Daten. Die Systemsicherheit ist für alle kritischen Systeme von immenser Bedeutung. Ohne ein vernünftiges Maß an Systemsicherheit sind die Verfügbarkeit, Zuverlässigkeit und Betriebssicherheit des Systems gefährdet, wenn externe Angriffe Schäden am System verursachen.

Der Grund für diesen Zusammenhang besteht darin, dass alle Methoden zur Sicherstellung von Verfügbarkeit, Zuverlässigkeit und Betriebssicherheit auf der Tatsache beruhen, dass das System dasselbe ist wie das ursprünglich installierte. Wenn dieses System auf irgendeine Weise in Mitleidenschaft gezogen wurde (zum Beispiel, wenn die Software durch einen Virus verändert wurde), können die ursprünglich vorgebrachten Argumente bezüglich der Zuverlässigkeit und Betriebssicherheit nicht länger aufrechterhalten werden. Die Systemsoftware kann verändert worden sein und sich unberechenbar verhalten.

Begriff	Beschreibung
Wirkung (Exposure)	Möglicher Verlust oder Schaden in einem Rechnersystem. Dabei kann es sich um den Verlust oder die Beschädigung von Daten oder den Verlust von Zeit und Aufwand handeln, wenn nach einer Sicherheitsverletzung eine Wiederherstellung erforderlich ist.
Anfälligkeit (Vulnerability)	Schwäche eines computerbasierten Systems, die ausgenutzt werden kann, um Verluste oder Schäden zu verursachen
Angriff (Attack)	Ausnutzen der Anfälligkeit eines Systems Im Allgemeinen geschieht dies von außerhalb des Systems und ist ein beabsichtigter Versuch, Schäden zu verursachen.
Bedrohungen (Threats)	Umstände mit der Möglichkeit, Verluste oder Schäden zu verursachen. Sie können sich diese als eine Anfälligkeit des Systems vorstellen, das einem Angriff unterworfen ist.
Kontrolle (Control)	Eine Schutzmaßnahme, die die Anfälligkeit eines Systems reduziert. Eine Verschlüsselung wäre ein Beispiel für eine Kontrolle, die die Anfälligkeit eines schwachen Zugriffskontrollsystems mindern würde.

Abbildung 3.9: Terminologie zur Systemsicherheit

Umgekehrt können Fehler bei der Systementwicklung zu Schwachstellen bei der Systemsicherheit führen. Wenn ein System auf unerwartete Eingaben nicht reagiert oder Indexgrenzen nicht überprüft werden, dann können Angreifer diese Schwächen ausnutzen, um

Zugriff auf das System zu erhalten. Bedeutende Zwischenfälle bezüglich der Systemsicherheit, wie der erste Internet-Wurm (Spafford, 1989) und der Code Red-Wurm mehr als 10 Jahre später (Berghel, 2001) haben die Tatsache ausgenutzt, dass Programme in C keine Überprüfung der Indexgrenzen vornehmen. Sie überschrieben einen Teil des Speichers mit Code, der unberechtigten Zugriff auf das System ermöglichte.

Bei einigen kritischen Systemen bildet die Systemsicherheit den wichtigsten Aspekt der Verlässlichkeit. Militärische Systeme, Systeme für E-Commerce sowie Systeme, die die Verarbeitung und den Austausch vertraulicher Informationen beinhalten, müssen so entworfen werden, dass sie einen hohen Sicherheitsgrad erreichen. Wenn zum Beispiel ein Reservierungssystem einer Fluggesellschaft ausfällt, führt dies zu Unannehmlichkeiten und einigen Verzögerungen bei der Ticketausgabe. Wenn das System jedoch nicht (zugriffs-)sicher ist und gefälschte Buchungen akzeptiert, kann die Fluggesellschaft, die das System besitzt, große Verluste erleiden.

Es gibt drei Schadensarten, die durch externe Angriffe verursacht werden können:

1. *Verweigerung von Diensten (Denial of Service, DOS):* Das System kann in einen Zustand gezwungen werden, in dem seine normalen Dienste nicht verfügbar sind. Dies beeinträchtigt dann natürlich die Verfügbarkeit des Systems.

2. *Beschädigung von Programmen oder Daten:* Die Softwarekomponenten des Systems werden möglicherweise unrechtmäßig verändert. Dies kann das Verhalten des Systems und somit dessen Zuverlässigkeit und Sicherheit beeinträchtigen. Wenn der Schaden beträchtlich ist, kann dies Auswirkungen auf die Verfügbarkeit des Systems haben.

3. *Offenlegung vertraulicher Informationen:* Die vom System verwalteten Informationen sind möglicherweise vertraulich und durch externe Angriffe besteht die Gefahr, dass diese unter Umständen an unbefugte Personen weitergegeben werden. Je nach Datentyp kann dies die Sicherheit des Systems gefährden und späteren Angriffen Vorschub leisten, die die Verfügbarkeit bzw. Zuverlässigkeit des Systems beeinträchtigen.

Wie bei anderen Aspekten der Verlässlichkeit gibt es auch für die Systemsicherheit eine eigene Terminologie. Einige wichtige Begriffe, die von Pfleeger (1997) eingeführt wurden, werden in ▶ Abbildung 3.9 definiert.

Es gibt eine deutliche Übereinstimmung mit einem Teil der Terminologie bezüglich der Betriebssicherheit, so dass die Wirkung mit einem Unfall und die Anfälligkeit mit einer Gefahr gleichgesetzt werden kann. Es gibt daher vergleichbare Verfahren, die verwendet werden können, um die Sicherheit eines Systems zu gewährleisten:

- *Vermeidung der Anfälligkeit:* Das System wird so entworfen, dass es nicht anfällig ist. Falls ein System beispielsweise nicht mit einem externen öffentlichen Netz verbunden ist, besteht keine Möglichkeit, dass ein Angriff von Außenstehenden erfolgen kann.
- *Erkennung und Neutralisierung von Angriffen:* Das System wird so entworfen, dass es Anfälligkeiten erkennen und entfernen kann, ehe sie zum Tragen kommen. Ein Beispiel für die Erkennung und Entfernung von Anfälligkeiten ist die Verwendung eines Virenscanners, der eingehende Dateien auf Viren untersucht und diese Dateien verändert, um den Virus zu entfernen.

- *Beschränkung der Wirkung:* Die Folgen eines erfolgreichen Angriffs werden minimiert. Beispiele für die Beschränkung der Wirkung sind regelmäßige Sicherungskopien von Systemen und ein Konfigurationsmanagement, das es gestattet, beschädigte Software wiederherzustellen.

Ein Großteil der Anfälligkeiten in computerbasierten Systemen beruht eher auf menschlichem Fehlverhalten als auf technischen Problemen. Die Systembenutzer verwenden einfach zu erratende Passwörter oder notieren ihre Passwörter an leicht zugänglichen Orten. Systemadministratoren machen Fehler beim Einrichten der Zugriffskontrolle oder der Konfigurationsdateien und Benutzer vergessen, Schutzsoftware zu installieren bzw. zu verwenden. Um die Systemsicherheit zu verbessern, müssen wir daher einen soziotechnischen Standpunkt einnehmen und uns Gedanken darüber machen, wie Systeme tatsächlich verwendet werden und nicht nur deren technische Merkmale berücksichtigen.

Zusammenfassung

- In einem kritischen System kann ein Ausfall bedeutende wirtschaftliche Verluste, physischen Schaden oder Gefahren für Gesundheit und Leben von Menschen zur Folge haben. Drei wichtige Klassen kritischer Systeme sind sicherheitskritische Systeme, aufgabenkritische Systeme und geschäftskritische Systeme.
- Die Verlässlichkeit eines Computersystems ist eine Systemeigenschaft, die den Vertrauensgrad des Benutzers in das System widerspiegelt. Die wichtigsten Aspekte der Verlässlichkeit sind Verfügbarkeit, Zuverlässigkeit, Betriebs- und Systemsicherheit.
- Unter der Verfügbarkeit eines Systems versteht man die Wahrscheinlichkeit, dass das System in der Lage ist, Dienste an seine Benutzer zu liefern, wenn sie gebraucht werden. Unter der Zuverlässigkeit versteht man die Wahrscheinlichkeit, dass die Systemdienste gemäß ihrer Spezifikation ausgeführt werden.
- Zuverlässigkeit und Verfügbarkeit werden für gewöhnlich als die wichtigsten Aspekte der Verlässlichkeit angesehen. Wenn ein System unzuverlässig ist, ist es schwierig, für Betriebs- oder Systemsicherheit zu garantieren, da diese bei Systemausfällen gefährdet sein können.
- Zuverlässigkeit steht im Zusammenhang mit der Wahrscheinlichkeit, dass während des Betriebs ein Fehler auftritt. Ein Programm kann daher bekannte Fehlerursachen enthalten, von seinen Benutzern aber noch als zuverlässig angesehen werden. Sie verwenden vielleicht niemals Systemfunktionen, die von diesen Fehlern betroffen sein könnten.
- Die Betriebssicherheit ist ein Systemmerkmal, das die Fähigkeit des Systems widerspiegelt, unter normalen wie nicht normalen Umständen zu funktionieren, ohne Menschen oder die Umgebung zu gefährden.
- Die Systemsicherheit ist für alle kritischen Systeme von immenser Bedeutung. Ohne ein vernünftiges Maß an Systemsicherheit sind die Verfügbarkeit, Zuverlässigkeit und Betriebssicherheit des Systems gefährdet, wenn externe Angriffe Schäden am System verursachen.
- Um die Verlässlichkeit zu verbessern, müssen Sie beim Systementwurf einen soziotechnischen Ansatz verfolgen und sowohl den menschlichen Faktor als auch die Hard- und Software im System berücksichtigen.

Ergänzende Literatur

„The evolution of information assurance“. Ein hervorragender Artikel über die Notwendigkeit, wichtige Informationen einer Organisation vor Störfällen und Angriffen zu schützen. (R. Cummings, *IEEE Computer*, 35 (12), Dezember 2002.)

Practical Design of Safety-critical Computer Systems. Ein allgemeiner Überblick über sicherheitskritische Systementwürfe, in dem Probleme der Betriebssicherheit erläutert werden, wobei nicht nur die Software, sondern das System als Ganzes beleuchtet wird. (W. R. Dunn, Reliability Press, 2002.)

Secrets and Lies: Digital Security in a Networked World. Ein exzellentes, sehr gut zu lesendes Buch über die Systemsicherheit von Computern unter soziotechnischem Aspekt. (B. Schneier, 2000, John Wiley & Sons.)

„Survivability: Protecting your critical systems“. Eine leicht zugängliche Einführung zum Thema Überlebensfähigkeit und deren Bedeutung. (R. Ellison et al., *IEEE Internet Computing,* Nov./Dec. 1999

Computer-related Risks. Sammlung, die dem Internet Risks-Forum zu Vorfällen, die in automatisierten Systemen aufgetreten sind, entnommen ist. Sie verdeutlicht, was tatsächlich alles in sicherheitsorientierten Systemen schief gehen kann. (P. G. Neumann, 1995, Addison-Wesley.)

Übungen Kapitel 3

3.1 Welches sind die drei Hauptarten von kritischen Systemen? Erläutern Sie die jeweiligen Unterschiede.

3.2 Geben Sie sechs Gründe an, weswegen die Verlässlichkeit in kritischen Systemen wichtig ist.

3.3 Welches sind die wichtigsten Aspekte der Verlässlichkeit eines Systems?

3.4 Warum steigen die Kosten zur Sicherung der Verlässlichkeit exponentiell?

3.5 Schlagen Sie unter Angabe einer Begründung für Ihre Antwort vor, welche Merkmale der Verlässlichkeit wahrscheinlich am wichtigsten für die folgenden System sind:

a. Ein von einem Internetdienstanbieter bereitgestellter Internetserver mit Tausenden von Kunden

b. Ein computergesteuertes Skalpell in der Schlüssellochchirurgie

c. Ein gerichtetes Steuerungssystem in einer Trägerrakete

d. Ein Internet-basiertes persönliches Finanzverwaltungssystem.

3.6 Bestimmen Sie sechs Produkte für den Endverbraucher, die heute oder in Zukunft sicherheitskritische Softwaresysteme enthalten könnten.

3.7 Die Zuverlässigkeit und die Betriebssicherheit sind verwandte, aber verschiedene Merkmale der Verlässlichkeit. Beschreiben Sie den wichtigsten Unterschied zwischen diesen Merkmalen und erläutern Sie, warum ein zuverlässiges System unsicher und ein unsicheres System zuverlässig sein kann.

3.8 Denken Sie sich für ein medizinisches System zur Behandlung von Tumoren mit radioaktiver Strahlung eine mögliche Gefahr aus und schlagen Sie eine Softwarefunktion vor, die verwendet werden kann, um zu garantieren, dass die erkannte Gefahr nicht zu einem Unfall führt.

3.9 Erklären Sie, warum es eine enge Beziehung zwischen der Systemverfügbarkeit und der Systemsicherheit gibt.

3.10 Erläutern Sie mit Hilfe von Begriffen der Computersicherheit den Unterschied zwischen einem Angriff und einer Bedrohung.

3.11 Ist es für einen Entwickler moralisch vertretbar, ein Softwaresystem mit bekannten Fehlern an einen Kunden auszuliefern? Macht es einen Unterschied, wenn dem Kunden die Existenz dieser Fehler vorher mitgeteilt wird? Wäre es unter diesen Umständen sinnvoll, Angaben zur Zuverlässigkeit der Software zu machen?

3.12 In Ihrer Eigenschaft als Experte für Computersicherheit ist eine Organisation an Sie herangetreten, die für die Rechte von Folteropfern eintritt. Sie werden gebeten, die Organisation bei ihren Bestrebungen zu unterstützen, unberechtigte Zugriffe auf die Computersysteme eines amerikanischen Unternehmens zu erlangen. Damit möchte die Organisation beweisen bzw. ausschließen, dass dieses Unternehmen Material verkauft, das direkt zur Folterung politischer Häftlinge eingesetzt wird. Erläutern Sie die ethischen Dilemmas, die diese Bitte heraufbeschwört, und wie Sie darauf reagieren würden.

Lösungen für ausgewählte Übungen dieses Kapitels finden Sie unter *www.pearson-studium.de* auf der Companion Website zum Buch.