

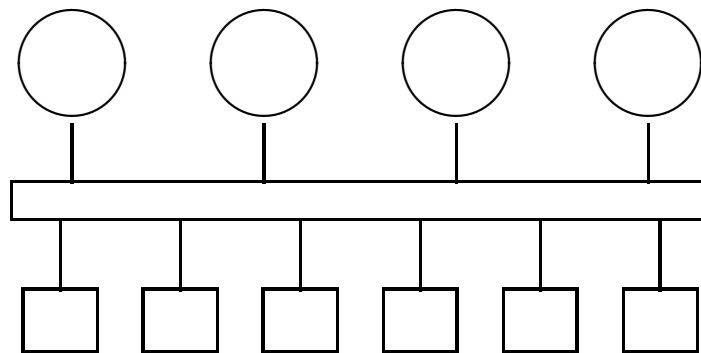
Verfügbarkeitsanalyse eines Mehrrechnersystems

Konzepte zur Spezifikation und Bestimmung traditioneller Verfügbarkeitsmaße

A. Warda, S. Zäske

{warda,zaeske}@ls4.informatik.uni-dortmund.de

Version 1.0 vom 1. Juni 1995



Universität Dortmund
LS Informatik IV
Prof. Dr.-Ing. H. Beilner
Projekt OSMOD 1994/'95
D-44221 Dortmund

Zusammenfassung

Es werden Konzepte zur analytischen und numerischen Analyse der Verfügbarkeit von (Rechen-) Systemen vorgestellt, die (u.a.) zur Validierung von Simulationsmodellen herangezogen werden, die mit dem in [Zä95] vorgestellten Tool erstellt worden sind.

Dieses Tool erlaubt die Ermittlung von Antwortzeitverteilungen, in die neben den Anforderungen der Last auch Teil- und Totalausfälle eingehen. Es berechnet somit ein kombiniertes Leistungs- und Verfügbarkeitsmaß. Zusätzlich kann es auch die klassischen Verfügbarkeitsmaße $MTTF$ (mittlere Zeit bis zu einem Ausfall), $MTTR$ (mittlere Reparaturzeit bei einem Ausfall) und *Ausfallwahrscheinlichkeiten* ermitteln.

Mit den hier vorgestellten hierarchischen Modellwelten lassen sich diese klassischen Verfügbarkeitsmaße analytisch bzw. numerisch berechnen oder zumindest schätzen, wenn die $MTTF$ und $MTTR$ der einzelnen System-Komponenten bekannt sind. Insofern können diese Modelle herangezogen werden, die (wesentlich komplexeren) Simulationsmodelle zu validieren.

Wahrscheinlichkeiten-Modelle erlauben die Berechnung der *Ausfallwahrscheinlichkeit* des gesamten Systems durch einfache Formeln. Die wichtigste Annahme in diesen Modellen ist *Unabhängigkeit* von Unter-Komponenten. Nimmt man weiter *Gedächtnislosigkeit* bei allen auftretenden Zeiten an, so lassen sich auch die $MTTF$ und $MTTR$ näherungsweise berechnen.

Mit einem *Markov-Modell* können *Abhängigkeiten* von Unter-Komponenten berücksichtigt und die $MTTF$ und $MTTR$ exakt berechnen werden. Die *Gedächtnislosigkeit* der Ausfall- und Reparaturzeiten muß allerdings auch hier gefordert werden.

Inhaltsverzeichnis

Zusammenfassung	3
1 Modellwelten	6
1.1 Begriffe	7
1.2 Modellierung der Verfügbarkeit	8
1.3 Bekannte Darstellungsformen für Verfügbarkeitsmodelle	10
1.3.1 Zuverlässigkeitsdiagramme	11
1.3.2 Fehlerbäume	11
1.3.3 Blockdiagramme	11
1.3.4 Beispiele und Vergleich der Darstellungsformen	12
1.4 Hierarchische Verfügbarkeitsmodelle	14
1.4.1 Grundbausteine	14
1.4.2 Erweiterungen	16
1.5 Eine Grammatik für hierarchische Verfügbarkeitsmodelle	18
1.5.1 Atomare Komponenten	19
1.5.2 Zusammengesetzte Komponenten	19
1.5.3 Geteilte Komponenten	20
1.5.4 Beispiel	21
2 Lösungskonzepte	24
2.1 Wahrscheinlichkeiten-Modell	26
2.1.1 Bezeichnungen	26
2.1.2 Und-Abhängigkeit	26
2.1.3 Oder-Abhängigkeit (Redundanz)	26
2.1.4 k-aus-n-Abhängigkeit (Teilredundanz)	27
2.1.5 Numerische Stabilität	29
2.2 MTTF/MTTR-Approximation	31
2.2.1 Beispiel: Und-Abhängigkeit von zwei atomaren Komponenten	32
2.2.2 Und-Abhängigkeit	37
2.2.3 Oder-Abhängigkeit (Redundanz)	38
2.2.4 k-aus-n-Abhängigkeit (Teilredundanz)	39

<i>INHALTSVERZEICHNIS</i>	5
2.3 Markov-Modell	42
2.3.1 Struktur und Spezifikation des Markov-Modells	42
2.3.2 Lösung	46
2.3.3 Tool-Unterstützung	46
2.4 Simulatives Modell	47
2.4.1 Struktur und Spezifikation des HIT-Modells	47
2.4.2 Lösung	51
2.4.3 Zusätzliche Erweiterungen der vorgestellten Modellwelt	52
3 Beispielanwendung	54
3.1 Das XCS als hierarchisches Verfügbarkeits-Modell	54
3.1.1 Die Komponenten des XCS-Modells	55
3.1.2 Ergebnisse	56
A Spezifikation mittels USENUM	58
A.1 Umsetzung von atomaren Komponenten	58
A.2 Umsetzung von Restriktionen	59
A.3 Umsetzung von Array-Komponenten	59
A.4 Umsetzung von Komponenten mit Unter-Komponenten	60
A.5 Weitere Spezifikationen	61
Literaturverzeichnis	63

Kapitel 1

Modellwelten

Es werden Systeme betrachtet, die *ausfallen* und *repariert werden* können. Ein System ist genau dann ausgefallen, wenn es aus Benutzersicht für die Bearbeitung von Aufgaben, für die das System entworfen worden ist, nicht *verfügbar* ist. (Anmerkung: hier wird eine vereinfachte Sichtweise dargestellt, für eine umfassende Einführung in das Thema und die Terminologie sei beispielsweise auf [Ech90] und [Lap85] verwiesen.)

Systeme lassen sich in Komponenten unterteilen, die wiederum selbst Systeme (Sub-Systeme) sein können, also weitere Komponenten enthalten können (z.B. KFZ = Karosserie + Antrieb + Fahrwerk, Antrieb = Motor + Getriebe, ...). Es ist nützlich, von dem Begriff der Komponente als *physikalisch* vorhandene und „greifbare“ Komponente zu abstrahieren und Komponenten als *funktional abgeschlossene Einheiten* eines Systems zu betrachten (beispielsweise kann man das Wissen des Fahrers über die Steuerung des KFZ auch als eine zum Funktionieren des Systems notwendige Komponente betrachten), d.h. eine Komponente läßt sich (zumindest hypothetisch) aus dem System herausnehmen und durch eine ähnliche Komponente ersetzen. Eine Unterteilung eines Systems in Komponenten ist nicht eindeutig, verschiedene Unterteilungen resultieren im allgemeinen aus *unterschiedlichen Sichtweisen* und *Abstraktionsniveaus*.

System-Ausfälle können verschiedene Ursachen haben. Diese Ursachen lassen sich eindeutig bestimmten Komponenten zuordnen, d.h. ein bestimmter System-Ausfall wird durch den Ausfall oder die Fehlfunktion (auch Fehlfunktionen lassen sich als Ausfall interpretieren) einer oder mehrerer System-Komponenten verursacht.

Andererseits muß nicht jeder Komponenten-Ausfall einen System-Ausfall zur Folge haben. Häufig werden Systeme so entworfen, daß sie ein gewisses Maß an *Fehlertoleranz* aufweisen, indem beispielsweise gewisse Komponenten oder ganze Sub-Systeme mehrfach vorhanden, also *redundant*, sind. Bei Komponenten-Ausfällen, die nicht zum *Total-Ausfall* des Systems führen, spricht man von *Teil-Ausfällen*. Das System ist dann zwar noch verfügbar, aber im allgemeinen wird es bei Teilausfällen nicht mehr im vollen Umfang verfügbar sein, also nicht seine volle Leistungsfähigkeit erreichen. (Z.B. Ausfall eines einzelnen Zylinders im Motor eines KFZ.) Der Einfluß von (Teil-)Ausfällen in einem Rechensystem auf das Leistungsvermögen wird in [Zä95] untersucht.

Ausgefallene Komponenten werden repariert oder ersetzt, was aber in gewissem Sinne einer Reparatur gleich kommt. Zur Vereinfachung ist es sinnvoll, die in der Realität durchaus vorhandene Zeitspanne vom Ausfall bis zum Beginn der Reparatur einer Komponente der Reparaturdauer zuzurechnen, d.h. die Reparaturdauer beginnt mit dem Ausfall der Komponente und endet genau dann, wenn die Komponente wieder verfügbar ist. In dieser Sichtweise wird mit der Reparatur also unmittelbar nach Ausfall einer Komponente begonnen. Deshalb kann der Zustand einer Komponente durch zwei verschiedene Werte, „ausgefallen“ (= wird repariert) und „verfügbar“ charakterisiert werden.

Jede Vereinfachung trägt die Gefahr der Ungenauigkeit oder sogar von Versagen in sich, falls man sich der vereinfachten Modellannahme nicht ständig bewußt ist. Hier ist die Gefahr darin gegeben, daß in der Realität bei Komponenten-Ausfällen, die einen Total-Ausfall zur Folge haben, Reparaturen meist unmittelbar veranlaßt werden, wohingegen die Beseitigung von Komponenten-Ausfällen, die das Leistungsvermögen des Systems nur gering oder gar nicht einschränken, nicht selten verdrängt wird (bis dann das System schließlich wegen weiterer Teil-Ausfälle unnötigerweise total ausfällt. . . hierfür sind wohl jedem hinreichend viele Beispiele bekannt).

In den folgenden Abschnitten werden traditionelle Verfügbarkeitsmaße (= quantitative Größen zur Beschreibung der Verfügbarkeit bzw. des Ausfall- und Reparaturverhaltens von Systemen) definiert und bewährte Modellwelten zur Modellierung von ausfallbehafteten Systemen vorgestellt. Das zweite Kapitel widmet sich der Lösung von Modellen, also der Berechnung bzw. Approximation der Verfügbarkeitsmaße, die nur durch weitere vereinfachende Annahmen möglich ist. Im dritten Kapitel werden die vorgestellten Techniken auf die Analyse eines Mehrrechen-systems angewandt.

1.1 Begriffe

In den technischen Datenblättern von elektronischen Geräten werden häufig hinter den Abkürzungen *MTTF* oder *MTBF* gewisse Zeitspannen genannt. Diesen Zahlen kann ein Benutzer entnehmen, mit welcher Lebenszeit er für das Gerät rechnen kann.

- Mit *MTTF* (mean time to failure) wird die mittlere Zeit eines Systems oder einer Systemkomponente in einem Verfügbarkeitszustand, also bis zum Eintreten eines Ausfalls bezeichnet.
- *MTTR* (mean time to repair) ist die mittlere Reparaturdauer, also die mittlere Zeit vom Eintreten eines Unverfügbarkeitszustands bis zum nächsten Verfügbarkeitszustand, insofern Reparaturen überhaupt vorgesehen sind.
- Die Bezeichnung *MTBF* (mean time between failure) ist sehr gebräuchlich und (wenn Reparaturen vorgesehen sind) durch $MTBF = MTTF + MTTR$ definiert.

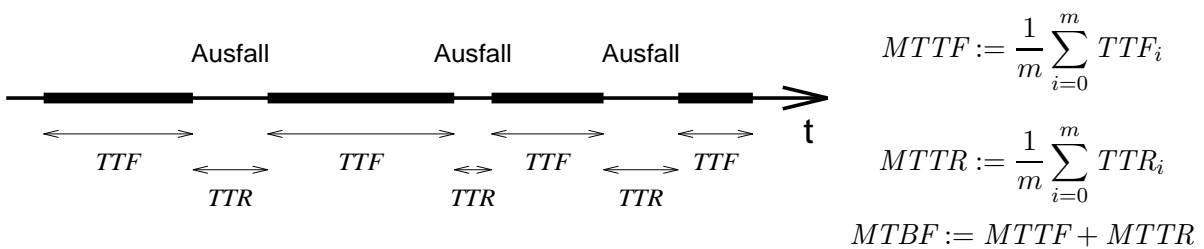


Abbildung 1.1: Die *MTTF* (*MTTR*) setzt sich zusammen aus dem Mittel der *m* beobachteten Verfügbarkeitsdauern (Reparaturdauern).

Manchmal wird mit *MTTF* auch die mittlere noch zu erwartende Lebensdauer bezeichnet, wenn die Beobachtung zu einem zufälligen Zeitpunkt innerhalb einer Verfügbarkeitsperiode beginnt. Unter dieser Interpretation ist beispielsweise bei gleichverteilten Verfügbarkeitsdauern von im Mittel zwei Stunden die *MTTF* gleich eine Stunde.

Da sich die Größenordnungen von *MTTF* und *MTTR* häufig um mehrere Zehnerpotenzen unterscheiden (z.B. 5 Jahre gegenüber 30 Sekunden), werden manchmal *MTBF* und *MTTF* als

Synonyme verwendet. Die Begriffe *MTBF* und *MTTF* werden allerdings in diesem Dokument stets unterschieden, *MTTF* und *MTTR* wie in Abb. 1.1 definiert und *MTBF* als $MTTF + MTTR$ definiert.

Als *Ausfallwahrscheinlichkeit* bezeichnet man den Quotienten

$$q = \frac{MTTR}{MTBF} \quad (1.1)$$

und komplementär dazu die *Verfügbarkeitswahrscheinlichkeit* (oder kurz: *Verfügbarkeit*, engl. *availability*)

$$p = \frac{MTTF}{MTBF} \quad (1.2)$$

Die Ausfallwahrscheinlichkeit entspricht dem zu erwartenden relativen Zeitanteil einer längeren Zeitdauer, in dem das System nicht verfügbar ist, also der Wahrscheinlichkeit, daß ein Beobachter zu einem zufällig gewähltem Zeitpunkt einem System-Ausfall sieht. Die Verfügbarkeit ist dagegen der zu erwartende relative Zeitanteil, in dem das System verfügbar ist, also die Wahrscheinlichkeit, daß das System an einem zufällig gewählten Zeitpunkt nicht ausgefallen ist.

Offensichtlich gelten:

$$p + q = 1 \quad (1.3)$$

$$\frac{MTTF}{p} = \frac{MTTR}{q} = MTBF \quad (1.4)$$

1.2 Modellierung der Verfügbarkeit

Ein *Modell* zur Verfügbarkeits-Analyse eines Systems besteht aus

1. der Spezifikation des Verfügbarkeits- und Ausfallverhaltens der System-Komponenten und
2. einer monotonen¹ bool'sche Funktion, die die Verfügbarkeit des Systems in Abhängigkeit von seinen Komponenten beschreibt (wahr = verfügbar = 1, falsch = ausgefallen = 0).

Die Spezifikation des Verfügbarkeits- und Ausfallverhaltens der Komponenten kann auf vielfältige Weise – im wesentlichen durch das gewählte Abstraktionsniveau der Modellierung bestimmt – geschehen, z.B. durch

- deterministische oder stochastische Prozesse,
- Verteilungen von Ausfall- und Reparaturzeiten,
- mittlere Ausfall- und Reparaturzeiten (*MTTF* und *MTTR*),
- Ausfallwahrscheinlichkeiten, ...

Die Beschreibung und Darstellung der bool'schen Funktion kann ebenfalls auf verschiedene — jedoch im wesentlichen äquivalente — Weisen geschehen. Die Wahl einer Darstellungsart wird vor allen Dingen durch die typischen Einsatz-Szenarien und durch die bevorzugte Sichtweise auf die zu modellierenden Systeme bestimmt. In den folgenden Abschnitten werden bekannte

¹eine bool'sche Funktion $f : \{0,1\}^n \rightarrow \{0,1\}$ heißt *monoton*, wenn für alle $(x_1, \dots, x_n) \in \{0,1\}^n$ und alle $i, 1 \leq i \leq n$, gilt: $f(x_1, \dots, x_n) = 1 \Rightarrow f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n) = 1$. Dies drückt aus, daß ein „vernünftiges“ System nicht durch den Ausfall einer Komponente repariert werden kann.

Darstellungsformen verglichen und eine dem Szenario *Verfügbarkeitsanalyse von Rechensystemen* angemessene Form ausgewählt.

Neben dem reinen „Beschreibungsaspekt“ sind bei dynamischen Modellen das Modellverhalten und mögliche Rückschlüsse auf das Verhalten des modellierten Systems von größerem Interesse. Um Aussagen über das Verfügbarkeits- und Ausfallverhalten des Systems bzw. dessen Modells erhalten, gibt es prinzipiell drei Vorgehensweisen:

Entwicklung eines mathematischen Modells und Lösen durch

1. analytische Methoden oder
2. numerische (markovsche) Methoden.

Erstellen eines Simulationsmodells und Durchführung von Experimenten durch

3. Simulation.

Selbstverständlich besteht ein enger, wechselseitiger Zusammenhang zwischen

- Art der Spezifikation des Verfügbarkeits- und Ausfallverhaltens der System-Komponenten,
- Lösungsmethode und
- ermittelbaren Verfügbarkeitsmaßen für das gesamte System.

Diese Arbeit konzentriert sich auf die Bestimmung der Verfügbarkeit, Ausfallwahrscheinlichkeit, *MTTF* und *MTTR* mit analytischen und numerischen Methoden, wohingegen [Zä95] simulative Methoden zur Bestimmung der Antwortzeitverteilung unter Berücksichtigung von (Teil-) Ausfällen behandelt. Die Spezifikation des Verfügbarkeits- und Ausfallverhaltens der System-Komponenten wird in dem hier dargestellten Konzept durch die Angabe von *MTTF* und *MTTR* vorgenommen.

1.3 Bekannte Darstellungsformen für Verfügbarkeitsmodelle

Die Spezifikation des Ausfall- und Verfügbarkeitsverhaltens der Komponenten eines Verfügbarkeitsmodells kann auf verschiedene Weisen geschehen. Die Wahl einer Spezifikationsform, z.B. Angabe von Ausfall- und Reparaturraten, ist i.a. dadurch stark eingeschränkt, daß nur bestimmte Kenntnisse über das Komponentenverhalten vorliegen bzw. nur gewisse Annahmen zu rechtfertigen sind. Der Umfang der Kenntnisse von den Komponenten bestimmt also in besonders starkem Maße die Art der Modellierung, Lösungsverfahren und somit natürlich auch, welche Verfügbarkeitsmaße für das System (Modell) überhaupt berechenbar sind.

Größere Freiheiten bestehen bei der Darstellung der bool'schen Funktion, die beschreibt, wie die Verfügbarkeit des Systems von seinen Komponenten abhängt.

Selbstverständlich könnte eine Modellierung der Verfügbarkeit eines Systems durch Angabe eines bool'schen Ausdrucks, wie etwa

$$\text{Verfügbarkeit}(A, B, C, \dots, H) = (A \wedge (B \vee C \wedge D) \wedge (B \vee E) \vee K) \vee (F \wedge (G \vee H))$$

und eine Auflistung von Komponenten-Eigenschaften bzw. einer algorithmischen Beschreibung des Komponenten-Ausfall- und -Reparatur-Verhaltens, wie beispielsweise hier dargestellt, erfolgen:

$\begin{aligned} \text{MTTF}(A) &= 3245h \\ \text{MTTF}(D) &= 22d \\ \text{MTTR}(E) &= 32s \\ \dots p(B) &= 1.45 \cdot 10^{-6} \\ q(B) &= 1 - p(B) \\ p(C) &= 0.99 \\ q(C) &= 0.01 \\ \dots \end{aligned}$	<pre> PROCESS life_of_K; BEGIN WHILE (true) DO K := 1; { available } HOLD (12000 + normal(1400, 130)); K := 0; { unavailable } HOLD (negexp(1/200)); END WHILE; END PROCESS life_of_K; </pre>
--	---

Jedoch scheinen Beschreibungen dieser Form nicht besonders gut dazu geeignet zu sein,

- Struktur und
- wesentliche Eigenschaften

des modellierten Systems auf einen Blick zu erfassen,

- Menschen das Verhalten des Systems zu beschreiben,
- das Modell bzw. seine Beschreibung leicht modifizieren zu können.

Aus der System-Modellierung sind eine Vielzahl von graphischen Darstellungsformen für Systeme bekannt; im weiteren Verlauf werden nun im Bereich der Fehlertoleranz und Leistungsbeurteilung häufig verwendete Darstellungsformen (für die bool'sche Funktion, die die Beziehung zwischen der Verfügbarkeit des Systems und den einzelnen Komponenten festlegt) vorgestellt und verglichen.

1.3.1 Zuverlässigkeitsdiagramme

Dies ist die bekannteste Darstellungsform für Modelle aus dem Bereich der Fehlertoleranz. Ein Zuverlässigkeitsdiagramm [Gö88] ist ein Baum (azyklischer, gerichteter Wurzelgraph), dessen innere Knoten logische Verknüpfungen (etwa „UND“, „ODER“, ...) darstellen. Die Blätter repräsentieren bool'sche Variablen (wahr = verfügbar = 1, falsch = ausgefallen = 0).

Die Semantik von Zuverlässigkeitsdiagrammen

- Ein Blatt ist „wahr“, wenn die repräsentierte Variable den Wert „wahr“ hat, ansonsten „falsch“.
- Ein innerer Knoten ist „wahr“, wenn die (in dem inneren Knoten dargestellte) logische Verknüpfung der Kindknoten den Wert „wahr“ hat, ansonsten „falsch“.
- Das System ist verfügbar, wenn die Wurzel „wahr“ ist, ansonsten ausgefallen.

1.3.2 Fehlerbäume

Fehlerbäume [Gö88] sind die (logischen) Komplemente von Zuverlässigkeitsdiagrammen. Ein Fehlerbaum ist also ein „Zuverlässigkeitsdiagramm“ für den negierten bool'schen Ausdruck der Verfügbarkeit eines Systems. Die Semantik von Fehlerbäumen ist komplementär zu der von Zuverlässigkeitsdiagrammen: das System ist ausgefallen, wenn die Wurzel „wahr“ ist.

1.3.3 Blockdiagramme

Blockdiagramme sind netzbasierte Darstellungsformen für Verfügbarkeitsmodelle. Ein Blockdiagramm [Gö88] ist ein ungerichteter Graph, dessen Knoten wieder bool'sche Variablen repräsentieren. Der Graph hat einen Eingang und einen Ausgang (deren Rollen aber vertauschbar sind).

Die Semantik von Blockdiagrammen

Das System ist verfügbar, wenn ein Weg zwischen den Ein-/Ausgängen existiert, der nur Knoten enthält, deren Variablen sämtlich „wahr“ (also verfügbar) sind.

1.3.4 Beispiele und Vergleich der Darstellungsformen

Im folgenden werden die drei Darstellungsformen anhand von Beispielen verglichen.

Beispiel 1: Fertigungsanlage

Eine Fertigungsanlage (Abb. 1.2) besteht aus zwei unabhängigen Förderbändern. An jedem der Förderbänder sind eine Maschine zur Vorbearbeitung (A, B) und eine Maschine zur Nachbearbeitung (C, D) der Produkte installiert.

Die Fertigungsanlage ist verfügbar, wenn (A und C) oder (B und D) funktionsfähig sind. Die Anlage ist ausgefallen, wenn A oder C ausgefallen sind und gleichzeitig B oder D ausgefallen sind.

Beispiel 2: Fertigungsanlage mit Brücke

Die Fertigungsanlage aus Beispiel 1 wird um eine Brücke E erweitert (Abb. 1.3). Nun können Produkte nach der Vorbearbeitung auf einem der Förderbänder zur Nachbearbeitung auf das andere Förderband umgeleitet werden.

Die Fertigungsanlage ist nun selbst dann noch verfügbar, wenn bspw. A und D ausgefallen sind, vorausgesetzt, die Brücke E ist verfügbar.

Das Blockdiagramm in Abb. 1.3 ist eine deutlich kompaktere und übersichtlichere Darstellung für dieses Beispiel als das dargestellte Zuverlässigkeitsdiagramm (welches im wesentlichen die möglichen Pfade des Blockdiagramms aufzählt).

Beispiel 3: Dreistrahliges Düsenflugzeug

Ein dreistrahliges Düsenflugzeug kann seinen Flug auch dann noch sicher fortsetzen, wenn eines seiner Triebwerke A, B, C ausgefallen ist (Abb. 1.4).

Das Zuverlässigkeitsdiagramm erlaubt in diesem Beispiel (unter Verwendung der abkürzenden Schreibweise $A+B+C \geq 2$ für $(A \wedge B) \vee (A \wedge C) \vee (B \wedge C)$) eine kompaktere und übersichtlichere Darstellung als die Blockdiagramme.

Vergleich der Darstellungsformen

Schon diese drei Beispiele machen deutlich, daß keine der Darstellungsformen für alle möglichen Systeme optimal bzgl. Kompaktheit und Übersichtlichkeit ist. Einerseits scheinen Zuverlässigkeitsdiagramme und Fehlerbäume i.a. für hierarchische Systemsichtweisen besser geeignet zu sein als Blockdiagramme, welche andererseits Materialfluß-orientierte Sichtweisen besser reflektieren.

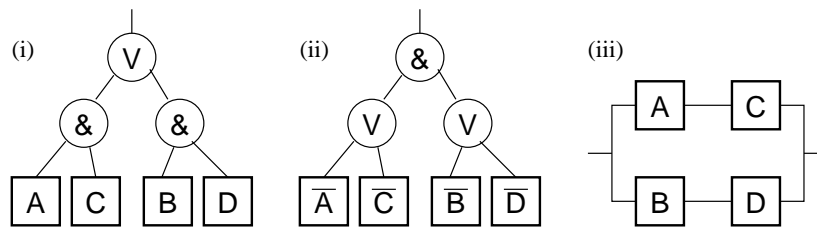


Abbildung 1.2: Die Fertigungsanlage aus Beispiel 1 ist ein Parallel-Serien-System: (i) ein Zuverlässigkeitsdiagramm, (ii) ein Fehlerbaum, (iii) ein Blockdiagramm

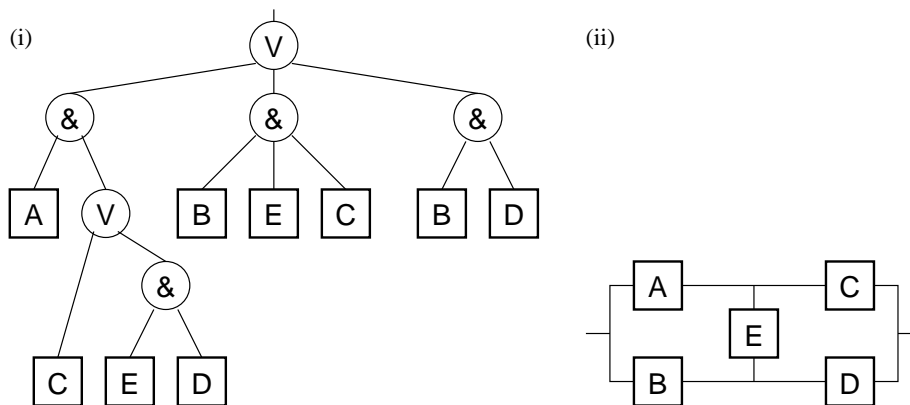


Abbildung 1.3: Die Fertigungsanlage mit Brücke aus Beispiel 2: (i) ein Zuverlässigkeitsdiagramm, (ii) ein Blockdiagramm

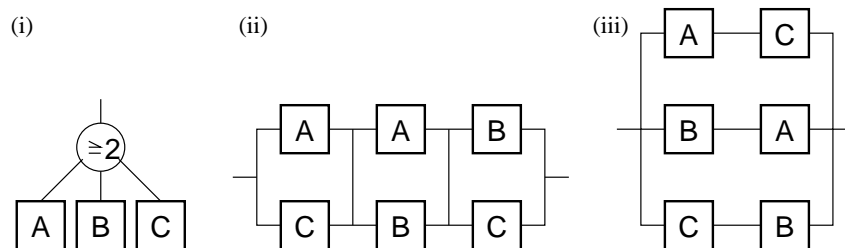


Abbildung 1.4: Das dreistrahlige Düsenflugzeug aus Beispiel 3 besitzt Teilredundanz: (i) ein Zuverlässigkeitsdiagramm, (ii), (iii) Blockdiagramme

1.4 Hierarchische Verfügbarkeitsmodelle

Heutzutage setzen sich geschichtete Modelle (wie beispielsweise das wohlbekannte ISO/OSI-Referenzmodell) verstärkt im Bereich der Beschreibung von (Rechen-) Systemen durch. Dies ist eine sehr vorteilhafte Sichtweise eines Systems als „Hierarchie virtueller Maschinen“ (vgl. etwa [Bei88]), die funktionale Abstraktion unterstützt.

In diesem Abschnitt wird deshalb eine Darstellungsform für die Beschreibung der Verfügbarkeit von Rechensystemen vorgestellt, die diese Sichtweise mit der Darstellungsform der Zuverlässigkeitsdiagramme kombiniert.

Die Grundbausteine von hierarchischen Modellen sind Komponenten, ein Modell besteht aus mindestens einer Komponente. Komponenten können atomar sein oder Unter-Komponenten enthalten. Jede Komponente kann nur in einer anderen Komponente enthalten sein, der Graph, der die *enthalten-sein*-Relation repräsentiert, bildet also einen Baum.

Eine Komponente hat (i.a. nur) zwei Zustände: entweder sie ist verfügbar, d.h. sie kann ihr gestellte Aufgaben bearbeiten, oder sie ist un verfügbar, also ausgefallen. Der Unverfügbarkeitszustand wird auch mit 0, *unavailable* oder *down* bezeichnet. 1, *available* oder *up* bezeichnet den Verfügbarkeitszustand.

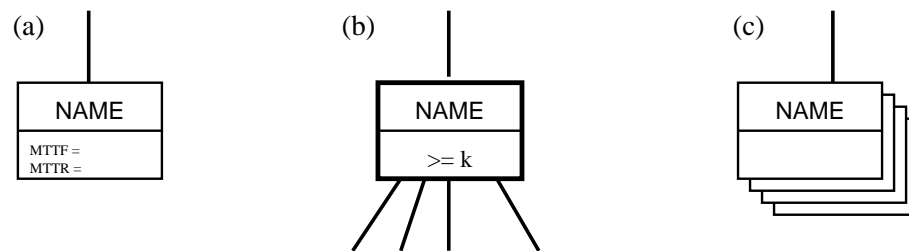


Abbildung 1.5: Die Grundbausteine der Modelle: (a) atomare Komponente, (b) nicht-atomare Komponente, (c) Array von identischen Komponenten

1.4.1 Grundbausteine

Atomare Komponenten

Für atomare Komponenten (Abb. 1.5 a) muß der Grad ihrer Verfügbarkeit durch Angabe der *MTTF*, *MTTR* spezifiziert werden. Sie enthalten keine Unter-Komponenten.

Nicht-atomare Komponenten

Die Verfügbarkeit von nicht-atomaren Komponenten (Abb. 1.5 b) ergibt sich durch die Verfügbarkeit der Unter-Komponenten und die Form der Abhängigkeit von den Unter-Komponenten.

Sei im Folgenden mit n die Anzahl der Unter-Komponenten einer nicht-atomare Komponente bezeichnet und k eine ganze Zahl $1 \leq k \leq n$.

Die Notation „ $\geq k$ “ (vgl. Abb. 1.5 b) beschreibt die Form der Abhängigkeit einer Komponente von ihren Unter-Komponenten: die Komponente ist verfügbar, wenn mindestens k ihrer Unter-Komponenten verfügbar sind. Offensichtlich ist für $k = n$ die Komponente nur verfügbar, wenn *alle* Unter-Komponenten verfügbar sind. Im Fall $k = 1$ ist die Komponente verfügbar, wenn

mindestens eine der Unter-Komponenten verfügbar ist. Auch der Fall $k = 0$ ist vorstellbar, dann ist die Komponente *immer* verfügbar, sie ist also unabhängig von ihren Unter-Komponenten.

Und-Abhängigkeit: Abhängigkeiten der Form „ $\geq k$ “ mit $k = n =$ Anzahl Unter-Komponenten werden als *und*-Abhängigkeiten bezeichnet, da sie der logischen Funktionalität von Und-Bausteinen entsprechen.

Die Komponente ist verfügbar \Leftrightarrow alle Unter-Komponenten sind verfügbar \Leftrightarrow die erste Unterkomponente ist verfügbar *und* die zweite Unterkomponente ist verfügbar *und* ... *und* die n -te Unterkomponente ist verfügbar.

Man schreibt auch häufig „&“ bzw. „^“ statt „ $\geq n$ “.

Oder-Abhängigkeit (Redundanz): Analog nennt man Abhängigkeiten der Form „ ≥ 1 “ *oder*-Abhängigkeiten bzw. *Redundanz*. Die Komponente ist verfügbar, wenn mindestens eine ihrer Unter-Komponenten verfügbar ist. Hier ist auch „ \vee “ statt „ ≥ 1 “ gebräuchlich.

k-aus-n-Abhängigkeit (Teilredundanz): Eine Abhängigkeit der Form „ $\geq k$ “ mit $1 < k < n$ bezeichnet man als *Teilredundanz* oder (nicht-triviale) *k-aus-n*-Abhängigkeit.

Beispiele findet man u.a. in der Luftfahrt: ein vierstrahliges Düsenflugzeug kann noch sicher landen, wenn nur zwei der vier Triebwerke funktionieren. Es handelt sich also um eine 2-aus-4-Abhängigkeit.

Array-Komponenten

Identische Unter-Komponenten, die nicht unterschieden werden, können zu Arrays zusammengefaßt werden (Abb. 1.5 c). Eine Array-Komponente besitzt die Zustände $0, 1, \dots, n$ (und nicht $0, 1, \dots, 2^n - 1$), da bei nicht-unterscheidbaren Unter-Komponenten die Anzahl der verfügbaren Unter-Komponenten zur Charakterisierung des Gesamtzustands der Array-Komponente genügt.

Welche dieser Zustände Verfügbarkeitszustände sind, wird durch die Spezifikation der Art der Abhängigkeit („ $\geq k$ “) in der die Array-Komponente enthaltenden Komponente festgelegt (vgl. Abb. 1.6). Die Verwendung von Array-Komponenten ermöglicht eine kompaktere Darstellung und kann auch bei der Lösung des Markov-Modells zur Zustandsraumreduktion ausgenutzt werden.

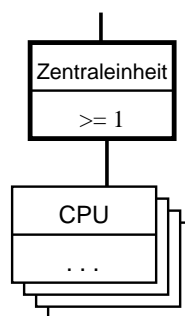


Abbildung 1.6: Eine aus vier identischen CPUs bestehende Zentraleinheit kann kompakt und übersichtlich durch die Verwendung eines Arrays von identischen Komponenten dargestellt werden. Die Array-Komponente besitzt die Zustände $0, 1, 2, 3, 4$. Die Komponente 'Zentraleinheit' ist verfügbar, wenn die Array-Komponente in einem Zustand ≥ 1 ist, also mindestens eine der CPUs verfügbar ist.

1.4.2 Erweiterungen

Die Grundbausteine reichen leider noch nicht aus, um alle Systeme, die sich durch Zuverlässigkeitsdiagramme, Fehlerbäume oder Blockdiagramme modellieren lassen, auch als hierarchische Modelle beschreiben zu können. In hierarchischen Modellen wird jede Komponente durch genau einen Modellbaustein dargestellt, wohingegen sich die Variablen in den Blättern von Fehlerbäumen beliebig häufig wiederholen dürfen.

Erst durch die Erweiterung der Modellwelt um *geteilte Komponenten* werden die Darstellungsformen bzgl. der Mächtigkeit der Modellklassen äquivalent. Eine darüber hinausgehende Erweiterung erlaubt, Abhängigkeiten zwischen Komponenten im Ausfall- und Verfügbarkeitsverhalten, wie Restriktionen und Folgefehler, graphisch darzustellen.

Es ist schon jetzt abzusehen, daß bei der Verwendung aller Modellelemente (Grundbausteine und Erweiterungen) Einschränkungen notwendig sein werden, damit die Modelle analytisch oder numerisch behandelbar sind. Die Semantik bestimmter Kombinationen von Modellelementen (z.B. geteilte Unter-Komponenten von Array-Komponenten) bedarf sicherlich auch noch einer genaueren Erklärung. Es sei an dieser Stelle auf die entsprechenden späteren Kapitel, welche die Lösung von Modellen beschreiben, verwiesen.

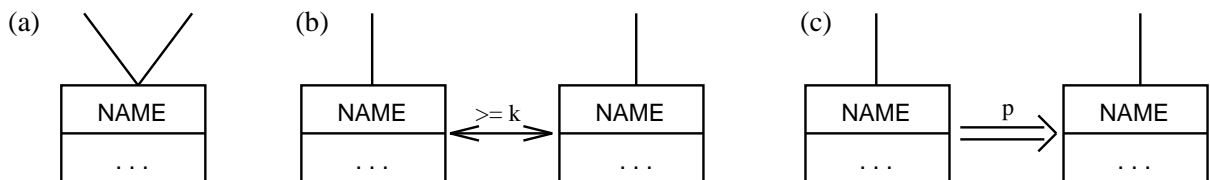


Abbildung 1.7: Erweiterungen der Modellwelt: (a) geteilte Komponente, (b) Restriktion, (c) Folgefehler.

Geteilte Komponenten

In Zuverlässigkeitsdiagrammen bezeichnet jedes Vorkommen derselben Variable dieselbe Komponente. In hierarchischen Verfügbarkeitsmodellen besteht dagegen eine Eins-zu-Eins-Abbildung zwischen Modellbausteinen und Komponenten.

Sind in einem hierarchischen Verfügbarkeitsmodell zwei identische Bausteine mit demselben Namen enthalten, so handelt es sich um verschiedene Komponenten. (Der Name einer Komponente entspricht also eher einer Typbezeichnung als einer eindeutigen Identifizierung.) Die mit den Grundbausteinen erzeugbaren Modelle sind also äquivalent zu Zuverlässigkeitsdiagrammen, bei denen jede bool'sche Variable in höchstens einem der Blätter vorkommt.

Eine geteilte Komponente (Abb. 1.7 a) ist gleichzeitig in mehreren nicht-atomaren Komponenten enthalten, entspricht also dem mehrfachen Vorkommen derselben bool'sche Variable im Zuverlässigkeitsdiagramm.

Restriktionen

Abhängigkeiten zwischen Komponenten können in Form von Restriktionen beschrieben werden. Der Pfeil „ \longleftrightarrow “ mit der Beschriftung $\geq k$ zwischen den zwei Komponenten in Abb. 1.7 b) bedeutet, daß zu jedem Zeitpunkt höchstens k der Komponenten im Unverfügbarkeitszustand sein können.

Für $k = 1$ bedeutet dies z.B. daß die linke Komponente auf jeden Fall verfügbar bleibt, falls die rechte Komponente ausgefallen ist, unabhängig von der für die linke Komponente spezifizierten Verfügbarkeit (und umgekehrt).

Folgendes wichtiges Anwendungsbeispiel motivierte die Einführung von Restriktionen.

Während der Durchführung einer Softwareänderung ist eine bestimmte Komponente nicht verfügbar. Softwareänderungen werden aber geplant durchgeführt und es wird vermieden, Softwareänderungen an verschiedenen Komponenten gleichzeitig durchzuführen. Also kann zu jeder Zeit nur eine Komponente wegen der Durchführung einer Softwareänderung ausfallen.

Folgefehler

Eine andere Form der Abhängigkeit zwischen Komponenten sind Folgefehler, d.h. der Ausfall einer Komponente zieht mit einer bestimmten Wahrscheinlichkeit den Ausfall einer anderen Komponente nach sich. Folgefehler werden durch einen Folgerungspfeil „ \implies “ zwischen den betroffenen atomaren Komponenten, der mit einer festen Wahrscheinlichkeit versehen ist, beschrieben.

In Abb. 1.7 c) hat der Folgerungspfeil zwischen den beiden Komponenten also die Bedeutung „ein Ausfall der linken Komponente zieht mit Wahrscheinlichkeit p einen Ausfall der rechten Komponente nach sich, unabhängig von der für die rechte Komponente spezifizierten Verfügbarkeit“.

Als Beispiel für einen Folgefehler sei folgendes Szenario betrachtet. Die Spannungsversorgung eines Rechners habe eine $MTTF$ von 2 Jahren und eine $MTTR$ von einer halben Stunde. Eine kurzzeitige Unterbrechung der Spannungsversorgung kann unter ungünstigen Umständen zu einem „Head-Crash“ des Plattenlaufwerks führen. Davon unabhängig habe ein Plattenlaufwerk eine $MTTF$ von 3 Jahren und eine $MTTR$ von einer Stunde. Untersuchungen haben ergeben, daß die Wahrscheinlichkeit für einen durch einen Ausfall der Spannungsversorgung verursachten „Head-Crash“ $p = 0.01$ ist (Abb. 1.8).

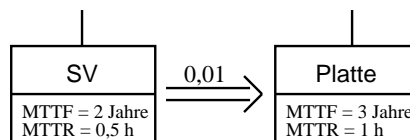


Abbildung 1.8: Beispiel für einen Folgefehler: Ein Ausfall der Spannungsversorgung (SV) zieht mit Wahrscheinlichkeit 0.01 einen Ausfall des Plattenlaufwerks nach sich.

Die Ausfallwahrscheinlichkeit der Spannungsversorgung beträgt somit

$$\frac{0.5}{2 \cdot 365.25 \cdot 24 + 0.5} = 2.85185 \cdot 10^{-5}$$

und die Ausfallwahrscheinlichkeit des Plattenlaufwerks ohne Berücksichtigung des Folgefehlers ist

$$\frac{1}{3 \cdot 365.25 \cdot 24 + 1} = 3.80243 \cdot 10^{-5}.$$

Bei Berücksichtigung des Folgefehlers ergibt sich dagegen für das Plattenlaufwerk eine Ausfallwahrscheinlichkeit von

$$2.85185 \cdot 10^{-5} \cdot 0.01 + 3.80243 \cdot 10^{-5} - 2.85185 \cdot 10^{-5} \cdot 0.01 \cdot 3.80243 \cdot 10^{-5} = 3.83095 \cdot 10^{-5}.$$

1.5 Eine Grammatik für hierarchische Verfügbarkeitsmodelle

In diesem Abschnitt wird eine Grammatik für hierarchische Verfügbarkeitsmodelle vorgestellt. Sie wird in *Erweiterter Backus-Naur Form, EBNF*, (siehe etwa [Se89]) beschrieben.

Nicht-Terminale beginnen mit Großbuchstaben und sind kursiv gesetzt, etwa *Model* oder *AtomicComponent*. Terminale sind fett gedruckt und in Kleinbuchstaben gesetzt, beispielsweise **compound** oder **array** und zusätzlich von Hochkommata umschlossen, falls sie spezielle Symbole sind, etwa '=' oder '('. Ein senkrechter Strich | beschreibt eine Auswahl. Runde Klammer () werden für Gruppierungen, geschweifte Klammern { } für Wiederholungen (auch Null-mal) und eckige Klammern [] für optionale Konstrukte verwendet.

```

    Model ::= CompoundComponent
    Component ::= AtomicComponent | CompoundComponent
    AtomicComponent ::= atomic component ComponentName
                        [ influences Influence { ',' Influence } ]
                        mttf = Expression
                        mtr = Expression
                        end ComponentName
    CompoundComponent ::= compound component ComponentName
                        [ constants ConstantDefinition { ',' ConstantDefinition } ]
                        [ influences Influence { ',' Influence } ]
                        [ restriction ListOfPaths ':' Dependency ]
                        [ defines shared Component { Component } ]
                        dependency Dependency
                        consists of ( ListOfComponents | Array )
                        end ComponentName
    ComponentName ::= Identifier
    Dependency ::= ( '>=' Expression ) | and | or
    ConstantDefinition ::= ConstantName '=' Expression
    ConstantName ::= Identifier
    Influence ::= Probability ':' Path
    Probability ::= Expression
    ListOfPaths ::= Path { ',' Path }
    Path ::= ComponentName [ '[' IndexList ']' ] [ '->' Path ]
    IndexList ::= IndexRange { ',' IndexRange }
    IndexRange ::= Expression | ( Expression ':' Expression )
    ListOfComponents ::= ( Component | SharedBinding ) { ( Component | SharedBinding ) }
    SharedBinding ::= shared component ComponentName
    Array ::= array '[' Expression ']' of Component
  
```

Abbildung 1.9: EBNF für hierarchische Verfügbarkeitsmodelle

Eine Spezifikation für ein hierarchisches Verfügbarkeitsmodell ist die Spezifikation einer zusammengesetzten Komponente, die also Unter-Komponenten enthält.

Spezifikationen für Unter-Komponenten sind textuell in der Spezifikation ihrer eindeutigen Vaterkomponente enthalten (Ausnahme: geteilte Komponenten, siehe unten). Dadurch ist es auf einfache Weise möglich, komplette Modelle wiederum als Unter-Komponenten für größere Modelle zu verwenden.

$$\begin{aligned}
\textit{Expression} & ::= \textit{Term} \{ ('+' | '-') \textit{Term} \} \\
\textit{Term} & ::= \textit{Factor} \{ ('*' | '/' | \mathbf{mod} | \mathbf{div}) \textit{Factor} \} \\
\textit{Factor} & ::= \textit{Base} \{ '**' \textit{Exponent} \} \\
\textit{Base} & ::= '(\textit{Expression})' | \textit{ConstantName} | \textit{RealConstant} \\
\textit{Exponent} & ::= '(\textit{Expression})' | \textit{ConstantName} | \textit{RealConstant} \\
\textit{RealConstant} & ::= [('+' | '-')] \textit{UnsignedInteger} \\
& \quad ['.' [\textit{UnsignedInteger}]] \\
& \quad [(\mathbf{e} | \mathbf{E}) [('+' | '-')] \textit{UnsignedInteger}] \\
\textit{UnsignedInteger} & ::= \textit{Digit} \{ \textit{Digit} \} \\
\textit{Identifizier} & ::= \textit{Alpha} \{ \textit{AlphaNumeric} \} \\
\textit{AlphaNumeric} & ::= \textit{Alpha} | \textit{Digit} \\
\textit{Alpha} & ::= \mathbf{A} | \mathbf{B} | \dots | \mathbf{Z} | \mathbf{a} | \mathbf{b} | \dots | \mathbf{z} | _ \\
\textit{Digit} & ::= \mathbf{0} | \mathbf{1} | \dots | \mathbf{9}
\end{aligned}$$

Abbildung 1.10: EBNF für arithmetische Ausdrücke und Bezeichner

Jede Komponente hat innerhalb des sie unmittelbar umschließenden Blocks einen eindeutigen Namen. Konstanten, die in der Vaterkomponente (oder deren Vaterkomponente usw.) definiert sind, sind sichtbar, können aber auch neu definiert werden (lexical scoping).

Für eine geteilte Komponente ist stets der Block, in dem sie definiert wird – und nicht der Block, indem sie gebunden wird – maßgeblich für die Sichtbarkeit von Konstanten.

1.5.1 Atomare Komponenten

Für atomare Komponenten wird die *MTTF* und *MTTR* direkt angegeben.

Zusätzlich kann noch eine Liste von Folgefehlern hinter dem Schlüsselwort **influences** angegeben werden. Diese Liste besteht aus mit Kommata separierten Paaren $p : K$, die spezifizieren, daß ein Ausfall der atomaren Komponente A mit Wahrscheinlichkeit p einen Ausfall der Komponente K auslöst.

K muß eine atomare Komponente sein und der arithmetische Ausdruck für p muß eine reelle Zahl zwischen 0 und 1 ergeben.

Ist die Komponente K nicht im selben Block wie A definiert, so muß der Pfad zu dieser Komponente angegeben werden. Dieser Pfad beginnt mit dem Namen einer gemeinsamen Vater-Komponente von A und K und endet bei K , wobei für auf dem Pfad liegende Unter-Komponenten von Array-Komponenten die jeweiligen Indizes angegeben werden müssen: die Indizes werden durch Kommata separiert aufgezählt und in eckige Klammern '[' ']' eingeschlossen. $\mathbf{m} : \mathbf{n}$ ist dabei als Abkürzung für $\mathbf{m}, \mathbf{m}+1, \dots, \mathbf{n}-1, \mathbf{n}$, wobei $m \leq n$ gelten muß, verwendbar.

1.5.2 Zusammengesetzte Komponenten

Eine zusammengesetzte Komponente enthält hinter dem Schlüsselwort **dependency** eine Beschreibung der Abhängigkeit von ihren Unter-Komponenten, $\geq e$, was ausdrückt, daß die Komponente genau dann verfügbar ist, wenn mindestens e ihrer Unter-Komponenten verfügbar sind. Statt ≥ 1 und $\geq n$ (falls n die Anzahl Unter-Komponenten ist) kann **or** bzw. **and** geschrieben werden.

1.5.4 Beispiel

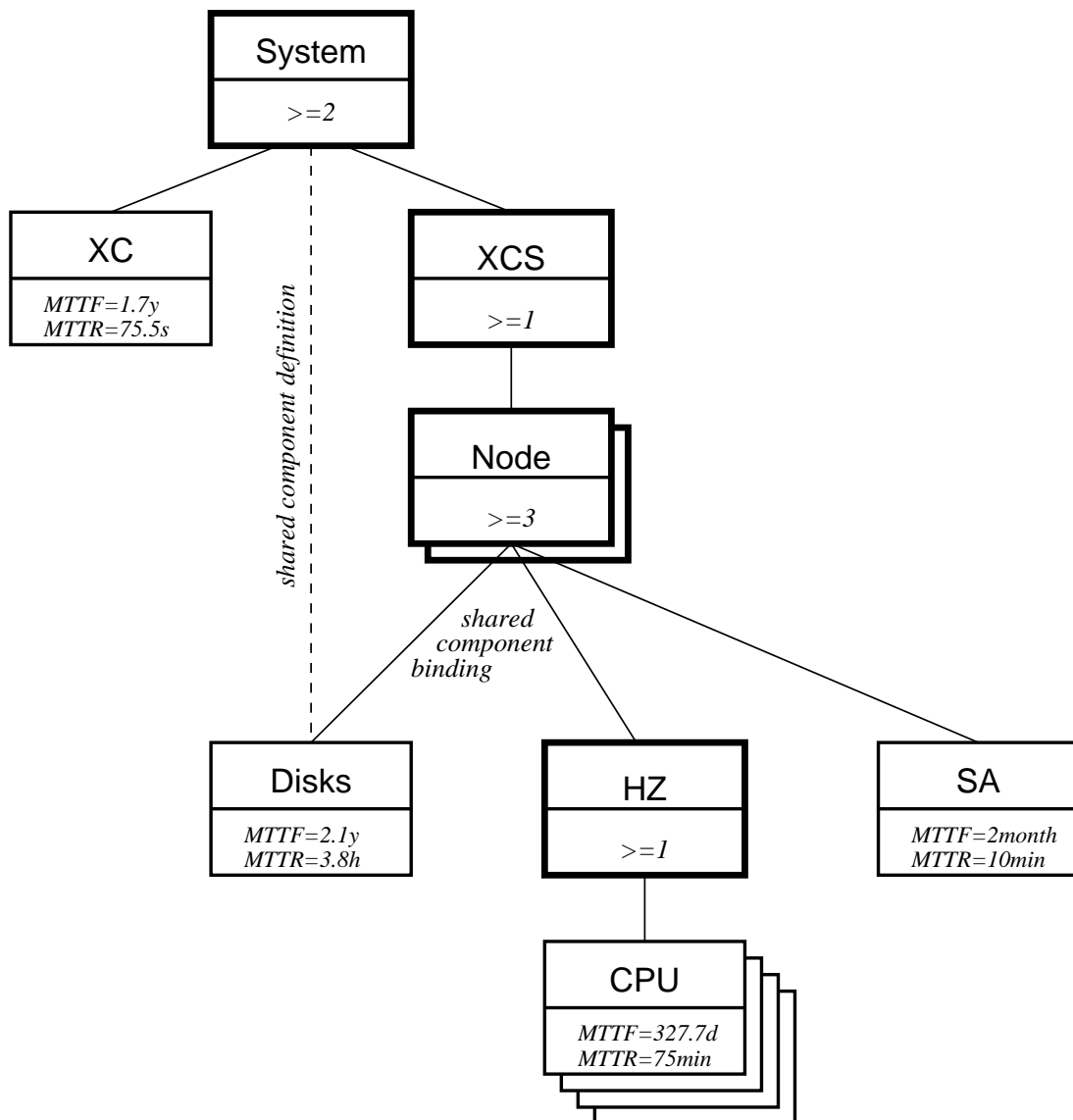


Abbildung 1.11: Beispiel eines hierarchisches Verfügbarkeitsmodells

Das folgende Beispiel ist eine textuelle Spezifikation des in Abb.1.11 dargestellten hierarchischen Verfügbarkeitsmodells.

```

compound component System
  constants
    minute = 60,
    hour   = 60 * minute,
    day    = 24 * hour,
    month  = 30.42 * day,
    year   = 12 * month,
    n_nodes = 2
  defines shared

    atomic component disks
      mttf = 2.1 * year
      mtrr = 3.8 * hour
    end XC

  dependency AND
  consists of

    atomic component XC
      mttf = 1.7 * year
      mtrr = 75.5 * second
    end XC

    compound component XCS
      restriction Node[1:2] -> SA : OR
      dependency OR
      consists of array [n_nodes] of

        compound component Node
          constants n_cpus = 4
          dependency AND
          consists of

            shared component disks

            atomic component SA
              mttf = 2 * month
              mtrr = 10 * minute
            end SA

            compound component HZ
              dependency >=1
              consists of array [n_cpus] of

                atomic component cpu
                  mttf = 327.7 * day
                  mtrr = 75 * minute
                end cpu

              end HZ

            end Node

          end XCS

        end System

```


Kapitel 2

Lösungskonzepte

Die vorgestellte Spezifikationstechnik der hierarchischen Verfügbarkeitsmodelle erlaubt eine saubere und klare Spezifikation eines Rechensystems, auf der verschiedene Lösungstechniken zur Ermittlung interessierender Verfügbarkeitsmaße aufsetzen können. Dabei können – je nach Komplexität des Modells und je nach gewünschten Ergebnisgrößen – verschiedene Lösungsmethoden zum Einsatz kommen. Hierfür kommen in Frage:

Analytische Vorgehensweise: Unter Auslassung der unter 1.4.2 genannten Erweiterungen ist es möglich – unter Annahme der Unabhängigkeit sämtlicher atomarer Komponenten – die Verfügbarkeitswahrscheinlichkeit (Availability) direkt auszurechnen (Abschnitt 2.1).

Geht man noch einen Schritt weiter, so kann man unter der Voraussetzung Exponentialverteilter Verfügbarkeits- und Reparaturzeiten der atomaren Komponenten auch die entsprechenden Zeiten für das Gesamtmodell näherungsweise berechnen (Abschnitt 2.2).

Markovsche Lösung: Sollen zusätzliche Abhängigkeiten atomarer Komponenten berücksichtigt werden, so ist eine analytische Lösung nicht mehr (von Spezialfällen abgesehen) möglich. Sämtliche Modelle, die mit der in Abschnitt 1.4 beschriebenen Technik spezifiziert sind, lassen sich jedoch mittels numerischer Lösung über einen Markov-Prozeß lösen. Auch hier läßt sich neben der Availability die Ausfallrate des Systems bestimmen, allerdings muß weiter die Voraussetzung Exponentialverteilter (bzw. Phasenverteiler) Verfügbarkeits- und Reparaturzeiten erfüllt sein. (Abschnitt 2.3).

Simulation: Während die beiden vorher genannten Verfahren die Ausfall- und Verfügbarkeitswahrscheinlichkeiten betrachten und davon ausgehend die mittleren Ausfall- und Verfügbarkeitszeiten ermitteln, sind sie jedoch zur Untersuchung verschiedener Ausfallzeitverteilungen nicht oder nur bedingt geeignet.

Ist eine solche Verteilung von Interesse, so kann diese auf simulativem Weg ermittelt werden. Voraussetzung dafür ist allerdings, daß entsprechende Angaben über die Verteilungen von Ausfall- und Verfügbarkeitszeiten der einzelnen atomaren Komponenten vorliegen. (Abschnitt 2.4).

Mögliche Realisierung der Konzepte

Zu diesen Konzepten wurden bereits erste Untersuchungen durchgeführt, um deren Realisierbarkeit und den Aufwand abzuschätzen. So gibt es zur analytischen Vorgehensweise einige Literatur, die daraus abgeleiteten für uns relevanten Formeln werden in Abschnitt 2.1 und 2.2 vorgestellt. Die Gesamt-Verfügbarkeit des Systems läßt sich hierbei Schritt für Schritt von den Blättern ausgehend bis zur Wurzel des Hierarchie-Baumes berechnen.

Des Weiteren wurde prototypisch untersucht, wie eine Umsetzung des Modells in einen Markov-Prozeß durchzuführen wäre. Die entsprechenden Konzepte werden in Abschnitt 2.3 vorgestellt.

Als Objekt der Modellierung wurde das XCS-Modell [Zä95] gewählt. Es zeigte sich, daß beide Prototypen (analytisches Modell und numerisches Modell) tatsächlich die gleichen Ergebnisse lieferten (unter Auslassung der vorhandenen Abhängigkeiten zwischen den atomaren Komponenten). Dies ist natürlich kein überraschendes Ergebnis, es deutet jedoch darauf hin, daß die unterschiedlichen Umsetzungen funktionieren. Abschnitt 3.1 gibt einen Überblick über die ermittelten Ergebnisse, wobei dort die Abhängigkeiten der atomaren Komponenten im numerischen Modell berücksichtigt sind.

Eine Lösung per Simulation wurde nicht weiter in Erwägung gezogen, zum einen wegen des hohen Zeitaufwands während der Laufzeit und zum anderen aufgrund der meist ungenauen Ausgangsdaten bezüglich der Ausfallzeitverteilungen der atomaren Komponenten.

Prinzipiell würden sich die Mechanismen in einem simulativen Modells an denen des numerischen Modells orientieren, wobei allerdings die komplexeren Ausfallzeitverteilungen zur Anwendung kämen. Eine Implementierung in HIT ([Bei88]) würde darauf hinauslaufen, daß die Hierarchie der Spezifikation umgekehrt würde: die Blätter würden als aktive Komponenten auf der obersten Hierarchie des HIT-Modells agieren und ihre Ausfall- und Reparaturereignisse an die sie enthaltenen Komponenten weiter versenden. Auf der untersten Ebene der HIT-Hierarchie befindet sich die Komponente, die das Gesamt-System repräsentiert und deren Ausfall- und Verfügbarkeitszeiten ausgewertet werden sollen.

Aufwand

Erste Untersuchungen haben gezeigt, daß eine Implementierung eines allgemeinen analytischen und numerischen Löser im Bereich des Möglichen liegt. Insbesondere die Implementierung des analytischen Löser scheint wegen der kurzen Laufzeiten auch für komplexe Modelle von Interesse, auch wenn hier keine Abhängigkeiten zwischen atomaren Komponenten untersucht werden können.

Da das Konzept für die Lösung bereits prototypisch für ein Modell implementiert wurde, wird der Hauptaufwand für ein generisches Tool darin liegen, eine geeignete Benutzerschnittstelle für die Modellspezifikation zur Verfügung zu stellen. Diese sollte sinnvollerweise eine textuelle Schnittstelle sein, damit das Tool weitgehend systemunabhängig eingesetzt werden kann. Diese textuelle Schnittstelle kann später als Schnittstelle zu einem graphischen Front-End eingesetzt werden.

2.1 Wahrscheinlichkeiten-Modell

Als Generalvoraussetzung bei der Analyse von Wahrscheinlichkeiten-Modellen muß die Unabhängigkeit von (Unter-) Komponenten gefordert werden. Daraus ergibt sich eine Einschränkung der Modellwelt: Restriktionen, Folgefehler und geteilte Komponenten können nicht modelliert werden.

Ist die Annahme (bzw. Voraussetzung) Exponential-verteilter Verfügbarkeits- und Reparaturzeiten nicht zu rechtfertigen (bzw. nicht erfüllt), oder sind von den atomaren Komponenten $MTTF$ und $MTTR$ erst gar nicht bekannt, so kann für das Gesamtsystem lediglich die Verfügbarkeit p und Ausfallwahrscheinlichkeit q berechnet werden. Allerdings legen diese schon das Verhältnis zwischen $MTTF$ und $MTTR$ des Systems fest, da

$$p + q = 1, \quad \frac{MTTF}{p} = \frac{MTTR}{q} = MTBF \quad \frac{p}{q} = \frac{MTTF}{MTTR}$$

gilt. Zu ausführlicheren Herleitungen der folgenden Formeln sei u.a. auf [Gö88], Abschnitt 4.1.1 ff. verwiesen.

2.1.1 Bezeichnungen

Es sei eine nicht-atomare Komponente mit n Unter-Komponenten gegeben. Für die Unter-Komponenten seien die benötigten Verfügbarkeitsmaße bereits berechnet oder (im Falle atomarer Unter-Komponenten) einfach vorgegeben.

Tabelle 2.1 listet die im folgenden verwendeten Bezeichnungen auf. Mit $MTTF$, λ , ... (also ohne Indizierung) werden die zu berechnenden Werte für die aus den Unter-Komponenten zusammengesetzte Komponente bezeichnet. $MTTF_i$, λ_i , ... (also mit Indizierung) sind die entsprechenden Werte der i -ten Unter-Komponente, $i = 1, \dots, n$. Im Falle identischer Unter-Komponenten haben natürlich alle Unter-Komponenten identische Werte, diese werden durch den Index 0 gekennzeichnet, etwa $MTTF_i = MTTF_0$ für alle $i = 1, \dots, n$.

2.1.2 Und-Abhängigkeit

Die Komponente ist verfügbar, wenn alle Unter-Komponenten verfügbar sind, sie fällt aus, sobald eine der Unter-Komponenten ausfällt. Somit ergibt sich für die Verfügbarkeit und Ausfallwahrscheinlichkeit der zusammengesetzten Komponente

$$p = \prod_{i=1}^n p_i, \quad q = 1 - \prod_{i=1}^n (1 - q_i) \quad (2.1)$$

was sich bei identischen Unter-Komponenten vereinfacht zu

$$p = (p_0)^n, \quad q = 1 - (1 - q_0)^n. \quad (2.2)$$

2.1.3 Oder-Abhängigkeit (Redundanz)

Die Komponente ist verfügbar, wenn mindestens eine der Unter-Komponenten verfügbar ist, sie fällt aus, sobald alle Unter-Komponenten ausgefallen sind. Somit ergibt sich für die Verfügbar-

	Für die gesamte Komponente:
p	Verfügbarkeit
q	Ausfallwahrscheinlichkeit
$MTTF$	mittlere Zeit bis zum Ausfall
$MTTR$	mittlere Reparaturdauer
λ	Ausfallrate ($= 1/MTTF$)
μ	Reparaturrate ($= 1/MTTR$)
$F_F(t)$	Verteilungsfunktion der Verfügbarkeitszeiten
$f_F(t)$	Dichtefunktion dieser Verteilung ($= \frac{d}{dt}F_F(t)$)
$F_R(t)$	Verteilungsfunktion der Reparaturzeiten
$f_R(t)$	Dichtefunktion dieser Verteilung ($= \frac{d}{dt}F_R(t)$)
n	Anzahl Unter-Komponenten dieser Komponente
$\vec{z}(t) = (z_1(t), \dots, z_n(t)) \in \{0, 1\}^n$	Zustand der Komponente zum Zeitpunkt t : $z_i(t) = 0 \Leftrightarrow$ die i -te Unterkomponente ist ausgefallen
	Für die i-te Unterkomponente:
p_i	Verfügbarkeit
q_i	Ausfallwahrscheinlichkeit
$MTTF_i$	mittlere Zeit bis zum Ausfall
$MTTR_i$	mittlere Reparaturdauer
λ_i	Ausfallrate ($= 1/MTTF_i$)
μ_i	Reparaturrate ($= 1/MTTR_i$)
$F_{F,i}(t)$	Verteilungsfunktion der Verfügbarkeitszeiten
$f_{F,i}(t)$	Dichtefunktion dieser Verteilung ($= \frac{d}{dt}F_{F,i}(t)$)
$F_{R,i}(t)$	Verteilungsfunktion der Reparaturzeiten
$f_{R,i}(t)$	Dichtefunktion dieser Verteilung ($= \frac{d}{dt}F_{R,i}(t)$)

Tabelle 2.1: Zusammenfassung der verwendeten Bezeichnungen

keit und Ausfallwahrscheinlichkeit der zusammengesetzten Komponente

$$q = \prod_{i=1}^n q_i, \quad p = 1 - \prod_{i=1}^n (1 - p_i) \quad (2.3)$$

was sich bei identischen Unter-Komponenten vereinfacht zu

$$q = (q_0)^n, \quad p = 1 - (1 - p_0)^n. \quad (2.4)$$

2.1.4 k-aus-n-Abhängigkeit (Teilredundanz)

Der Zustand $\vec{s} = (s_1, \dots, s_n)$ einer zusammengesetzten Komponente ist durch das kartesische Produkt der Zustände $s_i \in \{0, 1\}$, $i = 1, \dots, n$, der n Unter-Komponenten definiert. Da jede der n Unter-Komponenten verfügbar oder ausgefallen sein kann, besitzt die zusammengesetzte Komponente 2^n Zustände.

Bei Teilredundanzen ist der Fall „identische Unter-Komponenten“ von größerer Relevanz, da er realistischer ist als der Fall „verschiedene Unter-Komponenten“. Im letzteren Fall werden die Formeln zur Berechnung sehr komplex und unübersichtlich, sie bestehen im wesentlichen aus einer Aufzählung und Summation über die 2^n einzelnen Zustände.

Die Wahrscheinlichkeit $P[\vec{z} = \vec{s}]$, daß sich die zusammengesetzte Komponente im Zustand $\vec{s} =$

$(s_1, s_2, \dots, s_n) \in \{0, 1\}^n$ befindet, ist wegen der vorausgesetzten Unabhängigkeit der Unter-Komponenten gleich

$$\begin{aligned} P[\vec{z} = \vec{s}] &= P[z_1 = s_1 \mid z_2 = s_2 \mid \dots \mid z_n = s_n] \\ &= P[z_1 = s_1] \cdot P[z_2 = s_2] \cdot \dots \cdot P[z_n = s_n]. \end{aligned}$$

Man beachte, daß $(p_i)^{s_i} = p_i$, falls $s_i = 1$ und $(p_i)^{s_i} = 1$, falls $s_i = 0$, andere Werte nehmen die s_i hier nicht an, also

$$P[z_i = s_i] = (p_i)^{s_i} \cdot (q_i)^{1-s_i} = \begin{cases} p_i, & \text{falls } s_i = 1 \\ q_i, & \text{falls } s_i = 0 \end{cases}.$$

Identische Unter-Komponenten

Sind alle Unter-Komponenten identisch, so besitzen sie dieselbe Verfügbarkeit $p_i = p_0$ und Ausfallwahrscheinlichkeit $q_i = q_0$, $i = 1, \dots, n$. Somit ist $P[z_i = 1] = p_0$, wenn die i -te Unterkomponente verfügbar ist und $P[z_i = 0] = q_0 (= 1 - p_0)$, wenn sie ausgefallen ist, also

$$P[z_i = s_i] = (p_0)^{s_i} \cdot (q_0)^{1-s_i}, \quad s_i \in \{0, 1\}.$$

Insgesamt folgt somit für $\vec{s} \in \{0, 1\}^n$ aus (2.5)

$$\begin{aligned} P[\vec{z} = \vec{s}] &= (p_0)^{s_1} (q_0)^{1-s_1} \cdot (p_0)^{s_2} (q_0)^{1-s_2} \cdot \dots \cdot (p_0)^{s_n} (q_0)^{1-s_n} \\ &= (p_0)^{s_1+s_2+\dots+s_n} \cdot (q_0)^{(1+1+\dots+1)-(s_1+s_2+\dots+s_n)} \\ &= (p_0)^{|\vec{s}|} \cdot (q_0)^{n-|\vec{s}|}, \end{aligned}$$

wobei die Betrags-Norm des Zustandsvektors $|\vec{s}| := \sum_{i=1}^n |s_i|$ gleich der Gesamtzahl von verfügbaren Unter-Komponenten ist. Insgesamt gibt es jeweils $\binom{n}{k}$ Möglichkeiten, daß genau k der n Unter-Komponenten verfügbar sind.

Somit folgt für die Wahrscheinlichkeit $P[|\vec{z}| = k]$, daß genau k der Unter-Komponenten verfügbar sind

$$P[|\vec{z}| = k] = \binom{n}{k} (p_0)^k \cdot (q_0)^{n-k} \quad (2.5)$$

und

$$p = P[|\vec{z}| \geq k] = \sum_{i=k}^n \binom{n}{i} (p_0)^i \cdot (q_0)^{n-i} \quad (2.6)$$

ist die Wahrscheinlichkeit, daß mindestens k der Unter-Komponenten verfügbar sind, also die Verfügbarkeit der zusammengesetzten Komponente. Die Ausfallwahrscheinlichkeit der zusammengesetzten Komponente ist schließlich

$$q = 1 - p = \sum_{i=0}^{k-1} \binom{n}{i} (p_0)^i \cdot (q_0)^{n-i}. \quad (2.7)$$

Verschiedene Unter-Komponenten

Bei verschiedenen Unter-Komponenten müssen die einzelnen Unter-Komponenten unterschieden werden, es kommt nun nicht mehr alleine auf die Anzahl ausgefallener Unter-Komponenten an, sondern auch darauf, welche ausgefallen sind. Deshalb müssen die Einzelwahrscheinlichkeiten über alle entsprechenden Zustände summiert werden. Am übersichtlichsten notiert man dies

mit den Zustandsvektoren. Dazu definiert man aus den Verfügbarkeiten und Ausfallwahrscheinlichkeiten der Unter-Komponenten die folgenden Vektoren:

$$\vec{p} := (p_1, \dots, p_n), \quad \vec{q} := (q_1, \dots, q_n) \in [0, 1]^n.$$

Der Zustandsvektor

$$\vec{s} := (s_1, \dots, s_n) \in \{0, 1\}^n$$

gibt weiterhin für alle $i = 1, \dots, n$ an, ob die i -te Komponente ausgefallen ($s_i = 0$) bzw. verfügbar ($s_i = 1$) ist und die Betrags-Norm $|\vec{s}| = \sum_{i=1}^n s_i$ gibt wieder die Anzahl verfügbarer Unter-Komponenten zu einem Zustandsvektor an. Weiter definiert man nun den Einsvektor

$$\vec{1} := (1, 1, \dots, 1) \in \mathbf{N}^n$$

und das Potenzieren von reellen Vektoren mit geeigneten Zustandsvektoren, welches komponentenweise mit anschließendem Aufmultiplizieren der Potenzen geschieht:

$$(\vec{p})^{\vec{s}} := \prod_{i=1}^n (p_i)^{s_i} \in \mathbf{R}.$$

Addition und Subtraktion von (Zustands-) Vektoren sind wie üblich komponentenweise definiert. Die Wahrscheinlichkeit für einen Zustand $\vec{z} \in \{0, 1\}^n$ läßt sich mit den definierten Vektoren und Operationen kompakt notieren durch

$$P[\vec{z} = \vec{s}] = (\vec{p})^{\vec{s}} \cdot (\vec{q})^{\vec{1} - \vec{s}}. \quad (2.8)$$

Also berechnet sich die Verfügbarkeit der gesamten Komponente durch:

$$p = P[|\vec{z}| \geq k] = \sum_{k \leq |\vec{s}| \leq n} (\vec{p})^{\vec{s}} \cdot (\vec{q})^{\vec{1} - \vec{s}} \quad (2.9)$$

Die Ausfallwahrscheinlichkeit ist wieder

$$q = 1 - p = \sum_{|\vec{s}| < k} (\vec{p})^{\vec{s}} \cdot (\vec{q})^{\vec{1} - \vec{s}}. \quad (2.10)$$

2.1.5 Numerische Stabilität

Dieser Abschnitt soll mit ein paar Überlegungen zur numerischen Stabilität der vorgestellten Berechnungen abgeschlossen werden.

Fehler bei Berechnungen mit endlicher Genauigkeit

Bekanntermaßen werden reelle Zahlen in Rechnern nur mit endlicher Genauigkeit dargestellt und berechnet. Eine reelle Zahl $x \in \mathbf{R}$ wird im Rechner durch $M(x) \approx x$ repräsentiert. Die Menge aller Maschinenzahlen $\{M(x) | x \in \mathbf{R}\}$ ist nur endlich, deshalb werden zwei verschiedene reelle Zahlen, die sehr nahe beisammen sind, möglicherweise durch dieselbe Maschinenzahl dargestellt. Zudem kommt es bei der Durchführung von Rechenoperationen zu Ungenauigkeiten, wie das unten angegebene Beispiel 2) zeigt. Genaugenommen führt ein Rechner statt einer Rechenoperation „+“ eine ähnliche Operation „ \oplus “ aus, die die gewünschte Operation in vielen Fällen nur annähert. Besonders gefährlich sind die Addition und Subtraktion von Zahlen, die sich in der Größenordnung sehr stark unterscheiden. Dabei kann es zur Auslöschung von Stellen kommen, d.h. das angenäherte Ergebnis besitzt noch nicht einmal die mögliche Genauigkeit der Zahlendarstellung.

Beispiele:

1. **Darstellungsfehler:**

z.B. würde die Zahl 100001 bei 5-stelliger Genauigkeit durch $M(100001) = 100000 \neq 100001$ dargestellt,

2. **Stellenauslöschung** bei Addition oder Subtraktion:

z.B. $(M(100000) \oplus M(1)) \ominus M(100000) = (100000 \oplus 1) \ominus 100000 = 100000 \ominus 100000 = 0 = M(0)$. Das korrekte Ergebnis wäre natürlich $1 = M(1)$, welches sich auch bei einer günstigeren Anordnung der Schritte selbst mit den gestörten Operatoren berechnen läßt: $(M(100000) \ominus M(100000)) \oplus M(1) = (100000 \ominus 100000) \oplus 1 = 0 \oplus 1 = 1 = M(1)$ (alles wieder bei 5-stelliger Genauigkeit).

Fehler bei der Berechnung von Wahrscheinlichkeiten

Bei der numerischen Auswertung von Ausdrücken der Form

$$1 - \prod_{i=1}^n (1 - x_i)$$

und

$$1 - (1 - x)^n$$

mit $x \in (0, 1)$ und $n \in \mathbf{N}$ kann es zu numerischen Instabilitäten kommen, da Teilausdrücke $1 - \varepsilon$ mit sehr kleinem $\varepsilon > 0$, die sehr nahe bei 1 sind, wegen der endlichen Zahlendarstellung des Rechners fälschlicherweise als 1 dargestellt werden. Zumindest kommt es bei der Berechnung von $1 - (1 - \varepsilon)$ zu Stellenauslöschungen, so daß statt $1 - (1 - \varepsilon) = \varepsilon$ fälschlicherweise 0 berechnet wird.

Es empfiehlt sich deshalb, Ausdrücke der obigen Form auszumultiplizieren und die einzelnen Summanden in der Reihenfolge aufsteigender Beträge zu summieren, also bei den vom Betrag kleinsten Summanden zu beginnen.

Beispielsweise ist die Berechnung von $1 - (1 - x)^n$ durch

$$\left(\left(\dots \left((-x^n + \binom{n}{n-1} x^{n-1}) - \binom{n}{n-2} x^{n-2} \right) + \dots \right) - x \right)$$

der direkten Berechnung, etwa durch

$$1 - (\dots((1 - x) \cdot (1 - x)) \cdot \dots) \cdot (1 - x),$$

vorzuziehen. Die einzelnen Summanden werden dann (mit alternierendem Vorzeichen!) im wesentlichen in der Reihenfolge aufsteigender Größenordnungen summiert, was zu weniger Stellenauslöschungen und somit höherer Genauigkeit führt.

2.2 MTTF/MTTR-Approximation

Die Generalvoraussetzung bei der Analyse von Wahrscheinlichkeiten-Modellen war die Unabhängigkeit von Unter-Komponenten. Sind zudem alle Verfügbarkeits- und Reparaturzeiten der atomaren Unter-Komponenten einer *nur aus atomaren Unter-Komponenten zusammengesetzten* Komponente „gedächtnislos“ verteilt, so können auch die *MTTF* und *MTTR* für diese zusammengesetzte Komponente exakt berechnet werden.

Leider sind die Ausfall- und Reparaturzeiten einer aus atomaren Komponenten mit Exponentialverteilten Ausfall- und Reparaturzeiten zusammengesetzten Komponente aber im allgemeinen Hyperexponential- und nicht mehr Exponentialverteilt.

Exponential-Verteilung: [BFS87, Seite 168]

$$\text{Dichtefunktion } f(t) = \beta e^{-\beta t},$$

$$\text{Verteilungsfunktion } F(t) = 1 - e^{-\beta t},$$

Parameter $\beta > 0$,

Erwartungswert: $1/\beta$

n-phasige Hyperexponential-Verteilung: [BFS87, Seite 169]

$$\text{Dichtefunktion } f(t) = \sum_{i=1}^n \alpha_i \beta_i e^{-\beta_i t},$$

$$\text{Verteilungsfunktion } F(t) = \sum_{i=1}^n \alpha_i (1 - e^{-\beta_i t}) = 1 - \sum_{i=1}^n \alpha_i e^{-\beta_i t},$$

Parameter $\alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_n > 0$, wobei $\sum_{i=1}^n \alpha_i = 1$,

Erwartungswert: $\sum_{i=1}^n \alpha_i / \beta_i$

Deshalb ist die Annahme Exponentialverteilter Verfügbarkeits- und Reparaturzeiten aller Unter-Komponenten *nicht* erfüllt, wenn eine der Unter-Komponenten selbst weitere Unter-Komponenten enthält. Eine Berechnung der *MTTF* und *MTTR* von den Blättern (= atomaren Komponenten) des Hierarchie-Baumes bis zur Wurzel (= System), die auf der Annahme einer Exponential-Verteilung sämtlicher Verfügbarkeits- und Reparaturzeiten beruht, ist daher nicht korrekt, selbst wenn alle Verfügbarkeits- und Reparaturzeiten der *atomaren* Komponenten tatsächlich Exponentialverteilt sind.

Andererseits stellt sich bei näherer Betrachtung der entstehenden Hyperexponential-Verteilungen heraus, daß diese sich in vielen Fällen nur sehr geringfügig von Exponential-Verteilungen unterscheiden und deshalb kein allzugroßer Fehler bei der Berechnung der *MTTF* und *MTTR* durch die Approximation der Verfügbarkeits- und Reparaturzeiten durch Exponential-Verteilungen befürchtet werden muß.

Leider stehen keine Abschätzungen zur Verfügung, wie groß der Approximationsfehler im konkreten Fall sein kann, daher sind die in diesem Kapitel beschriebenen Methoden nur mit Vorsicht anzuwenden und die Anwendung sollte auf grobe Voranalysen beschränkt sein, denen stets noch genauere Untersuchungen folgen sollten. Für solche verfeinerten Folgeuntersuchungen können die hier beschriebenen Methoden allerdings mit sehr einfachen Mitteln und geringem Ressourcenverbrauch schon gute Näherungen ermitteln, etwa als Startwerte für iterative Verfahren, bei denen ein ungünstiger Startwert zwar die Performanz des Verfahrens, nicht aber die Genauigkeit der Ergebnisse negativ beeinflusst.

2.2.1 Beispiel: Und-Abhängigkeit von zwei atomaren Komponenten

Die Problematik und das prinzipielle Vorgehen bei der Approximation läßt sich am besten vorab an einem einfachen Beispiel verdeutlichen, bevor die einzelnen Formeln zur Approximation hergeleitet werden.

Es wird eine Komponente betrachtet, die aus zwei identischen Unter-Komponenten besteht und genau dann verfügbar ist, wenn keine der Unter-Komponenten ausgefallen ist (Abb. 2.1). Die Unter-Komponenten haben eine $MTTF$ von einer Stunde, $MTTR$ von 36 Sekunden und ihre Ausfall- und Reparaturzeiten seien Exponential-verteilt, also „gedächtnislos“.

$$\begin{array}{llll} MTTF_0 = 1 h & \lambda_0 = 1 h^{-1} & F_{F,0}(t) = 1 - e^{-t} & f_{F,0}(t) = e^{-t} \\ MTTR_0 = 0.01 h & \mu_0 = 100 h^{-1} & F_{R,0}(t) = 1 - e^{-100t} & f_{R,0}(t) = 100 e^{-100t} \end{array}$$

Gesucht sind $p, q, MTTF$ und $MTTR$ für die aus den zwei Unter-Komponenten zusammengesetzte Komponente „SYSTEM“.

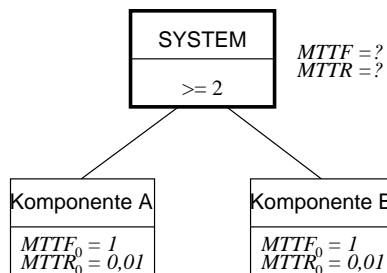


Abbildung 2.1: Und-Abhängigkeit von zwei atomaren Unterkomponenten

Modellierung durch einen Markov-Prozeß

Das Ausfall- und Reparaturverhalten der zusammengesetzten Komponente kann durch einen Markov-Prozeß mit folgenden Zuständen beschrieben werden:

- 0 = keine Unter-Komponente ausgefallen,
- 1 = eine der beiden Unter-Komponente ausgefallen und
- 2 = beide Unter-Komponenten ausgefallen.

Das Zustandsübergangsdiagramm dieses ergodischen Markov-Prozesses ist in Abb. 2.2 dargestellt, die Zustandsübergangsraten q_{ij} von Zustand i in Zustand j sind durch folgende Ratenmatrix gegeben:

$$Q = (q_{ij}) = \begin{pmatrix} -2\lambda_0 & 2\lambda_0 & 0 \\ \mu_0 & -(\lambda_0 + \mu_0) & \lambda_0 \\ 0 & 2\mu_0 & -2\mu_0 \end{pmatrix} = \begin{pmatrix} -2 & 2 & 0 \\ 100 & -101 & 1 \\ 0 & 200 & -200 \end{pmatrix}$$

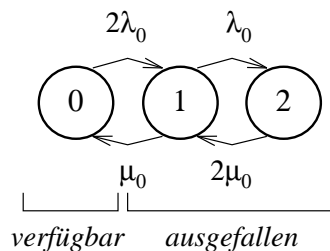


Abbildung 2.2: Zustandsübergangsdiagramm des ergodischen Markov-Prozesses

Berechnung der Verfügbarkeit und Ausfallwahrscheinlichkeit

Durch Lösen des linearen Gleichungssystems (*Steady-State-Analysis*), [Bu94, Abschnitt 3.1, S. 27-28]

$$(\pi_0, \pi_1, \pi_2) \cdot Q = 0, \quad \sum_i \pi_i = 1$$

lassen sich die stationären Zustandsaufenthaltswahrscheinlichkeiten $\pi_i =$ „Wahrscheinlichkeit für Zustand i “ dieses Markov-Prozesses und somit die Verfügbarkeit p und Ausfallwahrscheinlichkeit q bestimmen:

$$(\pi_0, \pi_1, \pi_2) = \frac{1}{(\lambda_0 + \mu_0)^2} \cdot (\mu_0^2, 2\lambda_0\mu_0, \lambda_0^2) \approx (9.8030 \cdot 10^{-1}, 1.96059 \cdot 10^{-2}, 9.8030 \cdot 10^{-3})$$

$$p = \pi_0 = \frac{\mu_0^2}{(\lambda_0 + \mu_0)^2} \approx 9.8030 \cdot 10^{-1}, \quad q = \pi_1 + \pi_2 = \frac{2\lambda_0\mu_0 + \lambda_0^2}{(\lambda_0 + \mu_0)^2} \approx 1.9704 \cdot 10^{-2}.$$

Es kann natürlich zur Berechnung von p und q völlig auf die Modellierung als Markov-Prozeß verzichtet werden, da die Formeln (1.1), (1.2) und (2.2) verwendet werden können:

$$\begin{aligned} q_0 &= \frac{MTTR_0}{MTBF_0} = \frac{1/\mu_0}{1/\lambda_0 + 1/\mu_0} = \frac{\lambda_0}{\lambda_0 + \mu_0} \\ p_0 &= \frac{MTTF_0}{MTBF_0} = \frac{1/\lambda_0}{1/\lambda_0 + 1/\mu_0} = \frac{\mu_0}{\lambda_0 + \mu_0} \\ p &= (p_0)^2 = \frac{\mu_0^2}{(\lambda_0 + \mu_0)^2} \approx 9.8030 \cdot 10^{-1} \\ q &= 1 - p = \frac{2\lambda_0\mu_0 + \lambda_0^2}{(\lambda_0 + \mu_0)^2} \approx 1.9704 \cdot 10^{-2}. \end{aligned}$$

Verteilung der Verfügbarkeitszeiten und MTTF

Die Aufenthaltsdauern in den *einzelnen* Zuständen eines Markov-Prozesses sind stets Exponentialverteilt, und da Zustand 0 der totalen Verfügbarkeit entspricht, ist leicht einzusehen, daß

$$\begin{aligned} F_F(t) &= P(\text{TTF} \leq t) \\ &= P(\min\{\text{TTF}_1, \text{TTF}_2\} \leq t) \\ &= 1 - P(\min\{\text{TTF}_1, \text{TTF}_2\} > t) \\ &= 1 - P(\text{TTF}_1 > t \text{ und } \text{TTF}_2 > t) \\ &= 1 - P(\text{TTF}_1 > t) \cdot P(\text{TTF}_2 > t) \\ &= 1 - (1 - P(\text{TTF}_1 \leq t)) \cdot (1 - P(\text{TTF}_2 \leq t)) \\ &= 1 - (1 - F_{F,1}(t)) \cdot (1 - F_{F,2}(t)) \\ &= 1 - (1 - F_{F,0}(t))^2 \\ &= 1 - (1 - (1 - e^{-\lambda_0 t}))^2 \\ &= 1 - e^{-2\lambda_0 t}, \end{aligned}$$

also die Verfügbarkeitszeiten gemäß

$$F_F(t) = 1 - e^{-2\lambda_0 t} = 1 - e^{-2t}$$

und somit Exponentialverteilt sind. Die MTTF ist demnach $MTTF_0/2 = 1/2h$.

Verteilung der Reparaturzeiten und MTTR

Im Gegensatz zu *einzelnen* Zuständen sind die Aufenthaltsdauern in *Zustandsmengen* im allgemeinen *nicht* Exponential-verteilt.

Es sei zunächst bemerkt, daß die Verteilung der Reparaturdauern sich *nicht* analog zur Berechnung der Verteilung der Verfügbarkeitszeiten als *Maximum* der einzelnen Reparaturdauern berechnen lassen:

$$F_R(t) \neq P(\max\{\text{TTF}_1, \text{TTF}_2\} \leq t) = 2 \cdot e^{-\mu_0 \cdot t} - e^{-2\mu_0 \cdot t},$$

da dies dem Anfangszustand „beide Unter-Komponenten sind ausgefallen“ entspricht, die gesamte Komponente aber schon bei Ausfall von nur einer Unter-Komponenten unverfügbar wird.

Um die Verteilung der Reparaturdauern zu ermitteln, betrachte man den *absorbierenden* Markov-Prozeß mit dem in Abb. 2.3 gezeigten Zustandsübergangsdiagramm und Zustandsübergangsmatrix

$$\bar{Q} = (q_{ij}) = \begin{pmatrix} 0 & 0 & 0 \\ \mu_0 & -(\lambda_0 + \mu_0) & \lambda_0 \\ 0 & 2\mu_0 & -2\mu_0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 100 & -101 & 1 \\ 0 & 200 & -200 \end{pmatrix}.$$

Der Markov-Prozeß startet im Zustand 1 (= eine Unter-Komponente ist ausgefallen) und erreicht irgendwann (u.U. nach einigen Wechseln zwischen den Zuständen 1 und 2 (= beide Unter-Komponenten sind ausgefallen)) den absorbierenden Zustand 0 (= keine Unter-Komponente ist ausgefallen).

Die zeitabhängige Zustandsaufenthaltswahrscheinlichkeit $\pi_0(t)$ = „Wahrscheinlichkeit für Zustand 0 zum Zeitpunkt t “ entspricht somit der Verteilung der Reparaturdauern der zusammengesetzten Komponente „SYSTEM“. Die *MTTR* der zusammengesetzten Komponente ist schließlich der Erwartungswert dieser Verteilung.

Um die Zustandsverteilungen $\pi_i(t)$ zu ermitteln, ist die lineare Differentialgleichung (*Chapman-Kolmogorov-Vorwärts-Gleichung*)

$$\frac{d}{dt}(\pi_0(t), \pi_1(t), \pi_2(t)) = (\pi_0(t), \pi_1(t), \pi_2(t)) \cdot \bar{Q}$$

mit dem Anfangswert

$$(\pi_0(0), \pi_1(0), \pi_2(0)) = (0, 1, 0)$$

(genau eine Unter-Komponente ist ausgefallen) zu lösen [Bu94, Abschnitt 3.2, S. 54-56].

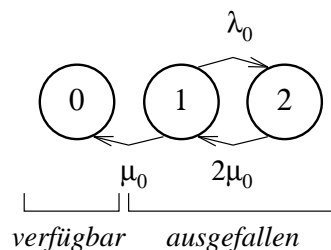


Abbildung 2.3: Zustandsübergangsdiagramm des absorbierenden Markov-Prozesses: Startzustand ist 1 = „eine Unter-Komponente ist ausgefallen“, der Zustand 0 = „keine Unter-Komponente ist ausgefallen“ ist absorbierend, da kein Übergang von Zustand 0 in einen anderen Zustand möglich ist.

Die allgemeine Lösung (in Abhängigkeit von λ_0, μ_0) ist schon bei diesem einfachen Beispiel (nur zwei Unter-Komponenten!) sehr komplex, bei Lösung für $\lambda_0 = 1$ und $\mu_0 = 100$ ergibt sich die 2-phasige Hyperexponential-Verteilung

$$F_R(t) = \pi_0(t) = 1 - \underbrace{0.9904762258e^{-99.01942115t}}_{\text{1. Phase}} - \underbrace{0.009523774625e^{-201.9805789t}}_{\text{2. Phase}}.$$

Deren Erwartungswert ist

$$MTTR = \left(\frac{0.9904762258}{99.01942115} + \frac{0.009523774625}{201.9805789} \right) h \approx 1.005 \cdot 10^{-2} h.$$

Auch zur Berechnung der *MTTR* kann auf die Modellierung als Markov-Prozeß (und Lösung der Differentialgleichung) verzichtet werden, da p, q und *MTTF* bekannt sind und somit wegen (1.4) die *MTTR* eindeutig bestimmt ist:

$$MTTR = \frac{q}{p} \cdot MTTF = \frac{2\lambda_0\mu_0 + \lambda_0^2}{\mu_0^2} \cdot \frac{MTTF_0}{2} = \frac{2\mu_0 + \lambda_0}{2\mu_0^2} \approx 1.005 \cdot 10^{-2} h.$$

Die Formel (1.4) liefert normalerweise nur eine Approximation der *MTTR*, da aber hier die exakten Werte für p, q und *MTTF* verwendet werden (die Verfügbarkeitszeiten der zusammengesetzten Komponente haben sind ja in diesem Beispiel tatsächlich als Exponential-verteilt herausgestellt) wird sogar der exakte Wert für die *MTTR* berechnet.

Approximation

Die Verteilung der Reparaturzeiten ist also definitiv nicht Exponential-verteilt. Andererseits fällt auf, daß die zweite Phase

1. zur Verteilung nur wenig Beitrag liefert, da $0.009523774625 \ll 0.9904762258$ und
2. sehr schnell abklingt, da $-201.9805789 \ll -99.01942115$.

Insofern besteht Grund zur Hoffnung, daß der Fehler nicht zu groß sein wird, wenn man in einem Modell, daß die zusammengesetzte Komponente „SYSTEM“ als Unter-Komponente enthält, die Verteilung der Reparaturzeiten durch eine Exponential-Verteilung mit gleichem Erwartungswert approximiert, also den Parameter $\beta = 1/0.01005 = 99.5025$ wählt:

$$F_R(t) \approx 1 - e^{-99.5025t}.$$

Abb. 2.4 zeigt den absoluten Fehler $F_R(t) - (1 - e^{-99.5025t})$, der bei dieser Approximation der Reparaturzeiten-Verteilung entsteht.

Zusammenfassung und Interpretation der Ergebnisse

Für die aus zwei identischen Komponenten zusammengesetzte Komponente wurden unter der Annahme von Exponential-verteilten Verfügbarkeits- und Reparaturzeiten die *exakten* Werte für $p, q, MTTF$ und *MTTR* berechnet. Die Reparaturdauern der zusammengesetzten Komponente sind nicht Exponential-verteilt, insofern stellen Berechnungen, in denen die für diese Komponente ermittelten Werte *MTTF* und *MTTR* zusammen mit einer Exponential-Verteilungs-Annahme eingehen, lediglich Annäherungen dar.

Im gewissen Sinne ist das dargestellte Beispiel ein „best case“, denn

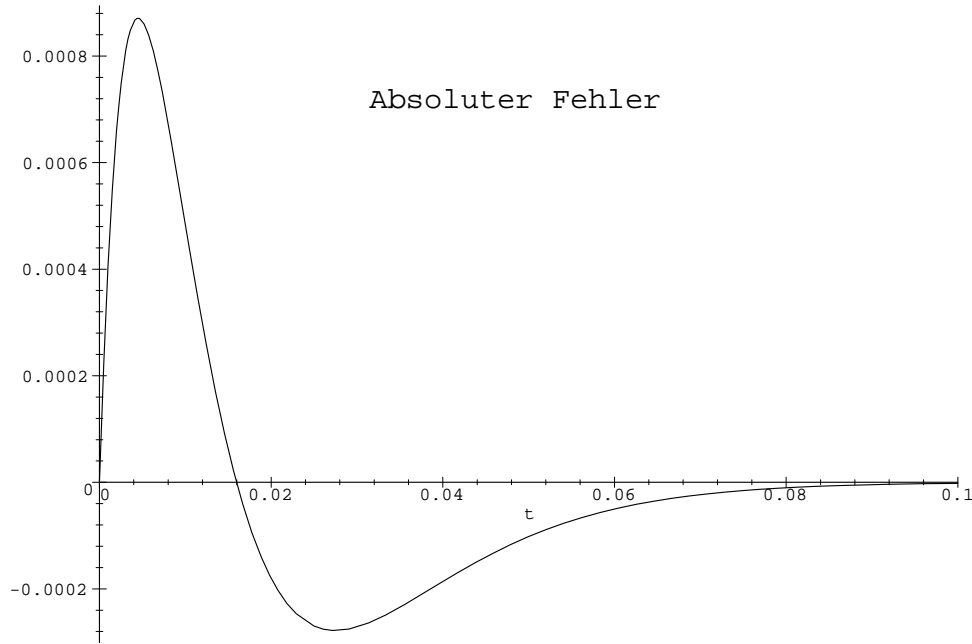


Abbildung 2.4: Absoluter Fehler der Approximation: $F_R(t) - (1 - e^{-99.5025t})$

1. die exakte Verteilung der Reparaturzeiten weist nur zwei Phasen auf. Werden mehr als zwei Unter-Komponenten betrachtet, ergeben sich auch entsprechend mehr Phasen für die Hyperexponential-Verteilung der Reparaturzeiten. Die Approximation durch eine 1-phasige Hyperexponential-Verteilung (= Exponential-Verteilung) wird dann ungenauer.
2. das Gewicht (0.9904762258) der langsamer abklingenden ersten Phase ist ca. 100-mal größer als das Gewicht der zweiten Phase. Das hat zwei Gründe:
 - (a) die Ausfallrate ist 100-mal kleiner als die Reparaturrate,
 - (b) es besteht eine Und-Abhängigkeit der zusammengesetzten Komponente von den beiden Unter-Komponenten.

Auch bei der Dichte spielt die zweite Phase deshalb keine große Rolle:

$$f_R(t) = 98.0764 \cdot e^{-99.01942115 \cdot t} + 1.92362 \cdot e^{-201.9805789 \cdot t}.$$

Wird das Beispiel dahingehend abgeändert, daß unter Beibehaltung sämtlicher Parameter lediglich die Und-Abhängigkeit in eine Oder-Abhängigkeit abgewandelt wird, so ergeben sich

$$MTTR = MTTR_0/2 = 0.005h$$

$$MTTF = 50.78518579h$$

$$p = 0.99990$$

$$q = 9.8444 \cdot 10^{-5}$$

$$F_R(t) = 1 - e^{-2\mu \cdot t} = 1 - e^{-200 \cdot t}$$

$$F_F(t) = 1 - 0.9906465197 \cdot e^{-0.01942115000 \cdot t} - 0.009412930394 \cdot e^{-102.9805789 \cdot t}$$

$$\approx 1 - e^{-0.01960784314 \cdot t}.$$

Die Reparaturzeiten sind nun also Exponential-verteilt und die Verfügbarkeitszeiten sind Hyperexponential-verteilt. Die Dichte der Verfügbarkeitszeiten-Verteilung (bzw. der approximativen

Exponential-Verteilung) ist

$$\begin{aligned} f_F(t) &= 0.0192395 \cdot e^{-0.01942115000 \cdot t} + 0.969349 \cdot e^{-102.9805789 \cdot t} \\ &\approx 0.01960784314 \cdot e^{-0.01960784314 \cdot t}. \end{aligned}$$

Die (schnell abklingende) zweite Phase hat hier einen im Vergleich zur zweiten Phase beim Originalbeispiel ($98.0764 \cdot e^{-99.01942115 \cdot t} + 1.92362 \cdot e^{-201.9805789 \cdot t}$) großen Anteil. Die Approximation ist also deutlich schlechter und würde natürlich bei Betrachtung von mehr Unter-Komponenten Genauigkeit verlieren.

Bei Teilredundanz (z.B. drei identische Unter-Komponenten: das System ist verfügbar, wenn höchstens eine Komponente ausgefallen ist) sind weder die Verfügbarkeits- noch die Reparaturzeiten der zusammengesetzten Komponente Exponential-, sondern mehr-phasig Hyperexponential-verteilt.

Schlußfolgerungen

Trotz der Risiken durch die nicht vorhersagbare Genauigkeit der Approximation der Verteilungsfunktionen ist diese Technik von Nutzen, da die Mittelwerte (bei einer Hierarchie-Stufe) korrekt berechnet werden und empirische Untersuchungen gezeigt haben, daß die Genauigkeit der Mittelwerte auch bei mehrstufigen Hierarchien (bei Vergleich mit der doch häufig geringen Genauigkeit der geschätzten Parameter und im Vergleich mit der Rechengenauigkeit) ausreichend für erste Grob-Analysen ist.

In den folgenden drei Abschnitten 2.2.2 – 2.2.4 werden die entsprechenden Formeln zur näherungsweise Berechnung der *MTTF* und *MTTR* für die drei Fälle Und-Abhängigkeit, Oder-Abhängigkeit und Teilredundanz hergeleitet.

2.2.2 Und-Abhängigkeit

Wegen der (tatsächlichen bzw. näherungsweise angenommenen) Exponential-Verteilung der Zeiten bis zu einem Ausfall bilden die Ausfallereignisse der Unter-Komponenten Poisson-Ströme mit Raten λ_i (vergleiche Abb. 2.5 und 2.6).

Sobald eine der Unter-Komponenten ausfällt, fällt die gesamte Komponente aus. Die Ausfälle der zusammengesetzten Komponente werden also durch den Strom beschrieben, der durch Mischen der Ströme der einzelnen Unter-Komponenten entsteht. Das Mischen von Poisson-Strömen ergibt wieder einen Poisson-Strom, die Ausfallraten werden einfach addiert:

$$\lambda = \lambda_1 + \dots + \lambda_n.$$

Bei identischen Unter-Komponenten: $\lambda = n \cdot \lambda_0$.

Für die mittlere Zeit bis zu einem Ausfall ergibt sich mit $MTTF = 1/\lambda$, $MTTF_i = 1/\lambda_i$ für $i = 1, \dots, n$ durch Kehrwertbildung und Erweitern der Brüche auf den Hauptnenner

$$\frac{1}{MTTF} = \frac{1}{MTTF_1} + \dots + \frac{1}{MTTF_n} = \frac{\sum_{j=1}^n \prod_{i \neq j} MTTF_i}{\prod_{i=1}^n MTTF_i}$$

und schließlich

$$MTTF = \frac{\prod_{i=1}^n MTTF_i}{\sum_{j=1}^n \prod_{i \neq j} MTTF_i}, \quad (2.11)$$

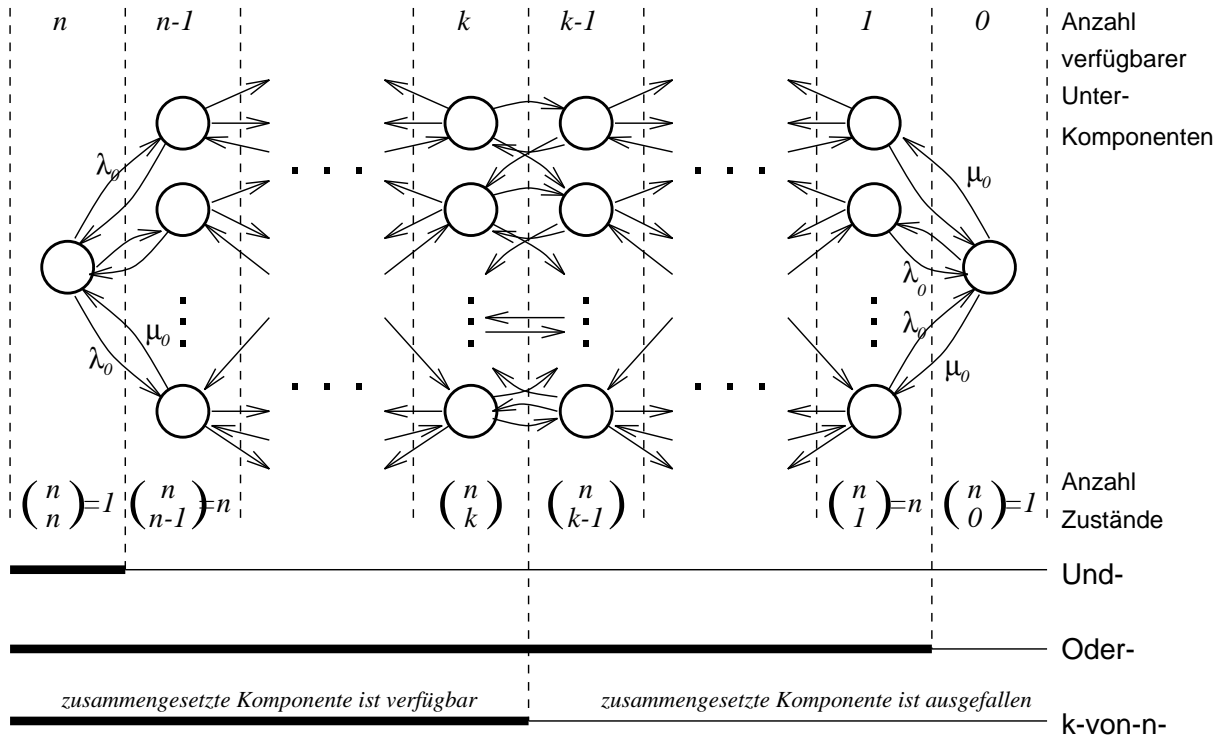


Abbildung 2.5: Die Zustände und Zustandsübergänge einer aus n identischen Unter-Komponenten zusammengesetzten Komponente

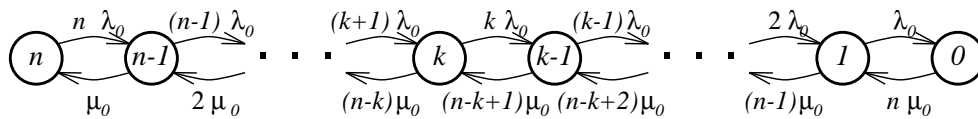


Abbildung 2.6: Werden die jeweils $\binom{n}{k}$ Zustände mit gleicher Anzahl k von verfügbaren Unter-Komponenten zusammengefaßt, ergibt sich ein Geburts-Todes-Prozeß

bzw. bei identischen Unter-Komponenten:

$$MTTF = \frac{MTTF_0}{n}$$

Letztlich läßt sich die mittlere Reparaturzeit approximieren durch

$$MTTR = \frac{q}{p} \cdot MTTF$$

Sind die Unter-Komponenten atomare Komponenten mit Exponential-verteilten Verfügbarkeits- und Reparaturzeiten, dann sind p, q und $MTTF$ für die zusammengesetzte Komponente exakt (die Verfügbarkeitszeiten der zusammengesetzten Komponente sind dann tatsächlich auch Exponential-verteilt), und somit ist auch der mit obiger Formel berechnete Wert für die $MTTR$ exakt, obwohl natürlich die Reparaturzeiten der zusammengesetzten Komponente i.a. nicht Exponential-verteilt sein werden.

2.2.3 Oder-Abhängigkeit (Redundanz)

Sei angenommen, die zusammengesetzte Komponente ist ausgefallen (und somit alle Unter-Komponenten ebenso). Die Komponente wird verfügbar, sobald eine Unter-Komponente repariert ist. Wegen der (tatsächlichen bzw. näherungsweise angenommenen) Exponential-Verteilung

der Reparaturzeiten der Unter-Komponenten bilden diese Reparaturereignisse einen Poisson-Strom (siehe wieder Abb. 2.5 und 2.6).

Die Reparatur der zusammengesetzten Komponente wird also durch den Strom beschrieben, der durch das Mischen der Ströme der einzelnen Unter-Komponenten entsteht. Das Mischen von Poisson-Strömen ergibt wieder einen Poisson-Strom, die Reparaturraten werden einfach addiert:

$$\mu = \mu_1 + \dots + \mu_n.$$

Bei identischen Unter-Komponenten: $\mu = n \cdot \mu_0$.

Für die mittlere Reparaturdauer ergibt sich (analog zu entsprechender Formel für die Approximation der *MTTF* bei Und-Abhängigkeit):

$$MTTR = \frac{\prod_{i=1}^n MTTR_i}{\sum_{j=1}^n \prod_{i \neq j} MTTR_i}, \quad (2.12)$$

bzw. bei identischen Unter-Komponenten:

$$MTTR = \frac{MTTR_0}{n}.$$

Schließlich läßt sich die mittlere Zeit bis zu einem Ausfall approximieren durch

$$MTTF = \frac{p}{q} \cdot MTTR.$$

Sind die Unter-Komponenten atomare Komponenten mit Exponential-verteilten Verfügbarkeits- und Reparaturzeiten, dann sind p, q und *MTTR* für die zusammengesetzte Komponente exakt (die Reparaturzeiten der zusammengesetzten Komponente sind dann tatsächlich auch Exponential-verteilt), und somit ist auch der mit obiger Formel berechnete Wert für die *MTTF* exakt, obwohl natürlich die Verfügbarkeitszeiten der zusammengesetzten Komponente i.a. nicht Exponential-verteilt sein werden.

2.2.4 k-aus-n-Abhängigkeit (Teilredundanz)

Der Zustand \vec{z} einer zusammengesetzten Komponente ist durch das kartesische Produkt der Zustände der n Unter-Komponenten definiert. Da jede der n Unter-Komponenten verfügbar oder ausgefallen sein kann, besitzt die zusammengesetzte Komponente 2^n Zustände.

Bei Teilredundanzen ist der Fall „identische Unter-Komponenten“ von größerer Relevanz als der Fall „verschiedene Unter-Komponenten“. Im letzteren Fall werden die Formeln zur Berechnung sehr komplex und unübersichtlich, sie bestehen im wesentlichen aus einer Aufzählung und Summation über die 2^n einzelnen Zustände.

Es sei noch einmal deutlich darauf hingewiesen, daß bei echter Teilredundanz („ $\geq k$ “ mit $1 < k < n = \text{Anzahl Unter-Komponenten}$) weder die Verfügbarkeits- noch die Reparaturzeiten der zusammengesetzten Komponente Exponential-, sondern im allgemeinen mehr-phasig Hyper-exponential-verteilt sind.

Um *MTTF* und *MTTR* zu approximieren, muß der Begriff des *kritischen* Zustands eingeführt werden:

Ein Zustand heißt *kritisch-verfügbar*, wenn das System (hier: die zusammengesetzte Komponente) verfügbar ist und der nächste Ausfall einer (Unter-) Komponente zum Ausfall des Systems führen würde. [Gö88]

Da die zusammengesetzte Komponente nur in kritisch-verfügbaren Zuständen ausfallen kann, ist die Übergangsrate von Verfügbarkeits- zu Unverfügbarkeitszuständen gleich der Wahrscheinlichkeit, daß ein kritisch-verfügbare Zustand eingenommen ist, multipliziert mit der Übergangsrate von diesem Zustand in einen Unverfügbarkeitszustand.

Sei $P[\vec{z} = \vec{s}]$ die Wahrscheinlichkeit, daß sich die zusammengesetzte Komponente im Zustand $\vec{s} = (s_1, s_2, \dots, s_n) \in \{0, 1\}^n$ befindet (vgl. Abschnitt 2.1.4).

Identische Unter-Komponenten

Kritisch-verfügbare Zustände sind im Fall der Teilredundanz bei identischen Unter-Komponenten die Zustände \vec{z} mit $|\vec{z}| = k$. Die einzelnen Zustände \vec{z} mit $|\vec{z}| = k$ brauchen nicht unterschieden werden, da es bei identischen Unter-Komponenten nur auf die Anzahl verfügbarer bzw. ausgefallener Unter-Komponenten ankommt.

Die Wahrscheinlichkeit, daß ein Zustand kritisch-verfügbar ist, ist nach Formel (2.5)

$$P[|\vec{z}| = k] = \binom{n}{k} (p_0)^k \cdot (q_0)^{n-k}.$$

Die Übergangsraten von diesen Zuständen in Unverfügbarkeitszustände sind $k \cdot \lambda_0$. Somit ist die Übergangsrate von einem Verfügbarkeitszustand in einen Ausfallzustand der zusammengesetzten Komponente gleich

$$\begin{aligned} \lambda &= P[|\vec{z}| = k] \cdot k \lambda_0 \\ &= \binom{n}{k} (p_0)^k \cdot (q_0)^{n-k} \cdot k \lambda_0 \end{aligned}$$

Die mittlere Zeit bis zu einem Ausfall ist dann gleich dem Kehrwert der Ausfallrate:

$$MTTF = \frac{MTTF_0}{\binom{n}{k} \cdot k \cdot (p_0)^k \cdot (q_0)^{n-k}}. \quad (2.13)$$

Die *MTTR* der zusammengesetzten Komponente könnte analog durch Betrachtung der *kritisch ausgefallenen* Zustände hergeleitet werden, einfacher geht es jedoch durch die fundamentale Beziehung

$$MTTR = \frac{q}{p} \cdot MTTF.$$

Verschiedene Unter-Komponenten

Bei *nicht*-identischen Unter-Komponenten werden zur Approximation der Ausfallrate für die zusammengesetzte Komponente mit „ $\geq k$ “-Abhängigkeit die verschiedenen Raten der Unter-Komponenten über alle möglichen kritischen Verfügbarkeitszustände mit den entsprechenden Wahrscheinlichkeiten gewichtet summiert. Am übersichtlichsten notiert man dies wieder mit den Zustandsvektoren wie in Abschnitt 2.1.4.

Die Wahrscheinlichkeit für einen bestimmten Zustand $\vec{s} \in \{0, 1\}^n$ (es war $s_i = 1$, falls die i -te Unter-Komponente verfügbar ist, 0 sonst) ist nach Formel (2.8)

$$P[\vec{z} = \vec{s}] = (\vec{p})^{\vec{s}} \cdot (\vec{q})^{\vec{1}-\vec{s}} \left(= \prod_{i=1}^n (p_i)^{s_i} \cdot \prod_{i=1}^n (q_i)^{1-s_i} \right).$$

Kritisch-verfügbare Zustände sind alle $\vec{s} \in \{0, 1\}^n$ mit $|\vec{s}| = k$. Der nächste Ausfall einer beliebigen (noch verfügbaren) Unter-Komponente bringt die zusammengesetzte Komponente in

einen Unverfügbarkeitszustand. Die Rate $\lambda(\vec{s})$ von einem kritisch-verfügbaren Zustand \vec{s} in einen beliebigen Unverfügbarkeitszustand ist deshalb die Summe aller λ_i , für die gilt, daß die i -te Komponente noch verfügbar ist, also $s_i = 1$ gilt.

$$\lambda(\vec{s}) = \sum_{\substack{i=1,\dots,n \\ s_i=1}} \lambda_i = \sum_{i=1,\dots,n} s_i \cdot \lambda_i$$

Die gesuchte Ausfallrate für die zusammengesetzte Komponente ergibt sich nun durch eine mit den entsprechenden Wahrscheinlichkeiten gewichtete Summation über alle möglichen kritischen Verfügbarkeitszustände.

$$\lambda = \sum_{|\vec{s}|=k} P[\vec{z} = \vec{s}] \cdot \lambda(\vec{s}) = \sum_{|\vec{s}|=k} \left((\vec{p})^{\vec{s}} \cdot (\vec{q})^{\vec{1}-\vec{s}} \cdot \sum_{i=1}^n s_i \cdot \lambda_i \right).$$

Die mittlere Zeit bis zu einem Ausfall *MTTF* der zusammengesetzten Komponente ist der Kehrwert dieser Rate λ .

$$MTTF = \left(\sum_{|\vec{s}|=k} \left(\left(\prod_{i=1}^n (p_i)^{s_i} \cdot (q_i)^{1-s_i} \right) \left(\sum_{i=1}^n \frac{s_i}{MTTF_i} \right) \right) \right)^{-1} \quad (2.14)$$

Die mittlere Reparaturzeit berechnet sich schließlich wieder durch die fundamentale Beziehung

$$MTTR = \frac{q}{p} \cdot MTTF.$$

2.3 Markov-Modell

Mit einem Markov-Modell lassen sich im Gegensatz zu den in Abschnitt 2.1 und 2.2 vorgestellten Techniken die *Mittelwerte* der Verteilungen von Verfügbarkeits- und Reparaturzeiten – also die *MTTF* und *MTTR* – nicht nur näherungsweise, sondern exakt berechnen. In Markov-Modellen können auch Abhängigkeiten zwischen Komponenten und Folgefehler berücksichtigt werden. Allerdings müssen die Verfügbarkeits- und Reparaturzeiten sämtlicher atomarer Komponenten auch weiterhin als Exponential-verteilt (oder zumindest Phasen-verteilt) angenommen werden.

Von Nachteil ist die hohe Komplexität der Lösung von Markov-Modellen, denn das zu lösende lineare Gleichungssystem hat eine Größe, die dem Produkt der Anzahlen von Zuständen der atomaren Komponenten entspricht. Die Laufzeit kann im schlimmsten Fall exponentiell mit der Anzahl atomarer Komponenten wachsen, da die Ergänzung des Modells um eine weitere atomare Komponente mit zwei Zuständen zu einer Verdopplung der Größe des linearen Gleichungssystems führen kann.

2.3.1 Struktur und Spezifikation des Markov-Modells

Ein Markov-Modell zur Zuverlässigkeitsanalyse besteht aus

1. **Zustandsraum** S ,
2. **Initialzustand** $s^0 \in S$,
3. **Transitionen** $t : S \times S \rightarrow \mathbf{R}_0^+$,
4. **Partition** des Zustandsraums $P = \{P_0, \dots, P_{k-1}\}$,

und kann kurz durch das 4-Tupel $M = (S, s^0, t, P)$ notiert werden.

Zustandsraum

Der Zustandsraum ist das kartesische Produkt der Zustandsmengen S_i aller n atomaren Komponenten im Modell:

$$S := S_1 \times S_2 \times \dots \times S_n = \{s^0, s^1, s^2, \dots\}, \quad 1 < |S| \leq \infty,$$

Es genügt hier, endliche Zustandsräume zu betrachten, da jede atomare Unter-Komponente im allgemeinen nur endlich viele verschiedene Zustände kennt.

Ein System-Zustand (Zustand des gesamten Modells) $s \in S$ ist also ein n -Tupel $s = (s_1, \dots, s_n)$, wobei $s_k \in S_k$ dem Zustand der k -ten atomaren Komponente entspricht. Ein hochgestellter Index, wie beispielsweise in s^0 , kennzeichnet einen speziellen System-Zustand, hier den Initialzustand, der sich aus den einzelnen Zuständen (s_1^0, \dots, s_n^0) der atomaren Komponenten zusammensetzt. Tiefgestellte Indizes bezeichnen also Zustände von atomaren Komponenten, es ist etwa s_1^0 der Initialzustand der ersten atomaren Komponente.

Die Zustandsmenge S_k der atomaren Komponente k enthält die zwei Zustände „ausgefallen“ (= un verfügbar, unavailable, 0, ...) und „verfügbar“ (= nicht ausgefallen, available, 1, ...). Die Zustandsmengen werden auch mit $\{0, 1\}$ identifiziert. Es ist möglich, feinere Abstufungen der Verfügbarkeit zu modellieren, indem beispielsweise für eine Komponente k die Zustände

$$s_k \in S_k := \{0, 1, 2\}$$

einem Total-Ausfall, einem Teil-Ausfall (Komponente nur noch eingeschränkt verfügbar) bzw. der vollen Verfügbarkeit entsprechen.

Die Spezifikation des Zustandsraumes wird durch eine Aufzählung der Zustandsmengen der atomaren Komponenten vorgenommen:

$$\begin{aligned} S_1 &:= \{0, 1, \dots\} \\ &\vdots \\ S_n &:= \{0, 1, \dots\} \end{aligned}$$

Initialzustand

Der Initialzustand $s^0 \in S$ ist der Zustand, in dem sich das System zu Beginn befindet. Im allgemeinen wird der Zustand gewählt, in dem alle Unter-Komponenten vollständig verfügbar sind, also keine Ausfälle vorhanden sind.

Die Spezifikation des Initialzustands wird durch die Angabe der Initialzustände der einzelnen atomaren Komponenten vorgenommen:

$$s^0 = (s_1^0, s_2^0, \dots, s_n^0) \in S, \quad s_k^0 \in S_k$$

Transitionen

Transitionen, also Zustandswechsel, sind immer mit einer Rate versehen. Die Zustandsübergangsfunktion

$$t : S \times S \longrightarrow \mathbf{R}_0^+$$

beschreibt die Häufigkeit der Zustandswechsel:

$$t(s^i, s^j) = \lambda_{i,j} \begin{cases} = 0, & \text{falls kein Übergang von Zustand } s^i \\ & \text{nach Zustand } s^j \text{ möglich ist,} \\ > 0, & \text{falls ein solcher Zustandswechsel im Mittel nach} \\ & 1/\lambda_{i,j} \text{ Zeiteinheiten im Zustand } s^i \text{ erfolgt.} \end{cases}$$

Es genügt, die Wechselraten für die *möglichen* Zustandswechsel festzulegen, wenn die übrigen als Null angenommen werden.

Zustandswechsel finden immer dann statt, wenn eine atomare Komponente k ihren Zustand wechselt, also ausfällt oder repariert worden ist. Dadurch ändert sich genau das k -te Element des den Zustand repräsentierenden Tupels $s^j = (s_1^j, \dots, s_k^j, \dots, s_n^j)$. Hat eine Komponente eine mittlere Zeit bis zum Ausfall von $MTTF$, so ist die Rate für einen entsprechenden Zustandswechsel durch den Kehrwert der $MTTF$ gegeben.

Anstatt sämtliche mögliche Zustandswechsel einzeln aufzuzählen ist es problemnäher und kompakter, Zustandswechsel, die demselben Ereignistyp (Ausfall bzw. Reparatur) zugrundeliegen und dieselbe Komponente k betreffen, zusammenzufassen, beispielsweise für den Ausfall einer atomaren Unter-Komponente:

Ausfall der k -ten Komponente

Für alle Zustände $s = (s_1, \dots, s_n) \in S$ mit $s_k = 1$ definiere

$$s' := (s_1, \dots, s_{k-1}, s_k - 1, s_{k+1}, \dots, s_n)$$

$$t(s, s') := 1/MTTF_k$$

„ $s_k = 1$ “ ist die *Vorbedingungen* für das Ereignis, sie charakterisiert alle Zustände, in denen die k -te atomare Unter-Komponente verfügbar ist und somit ausfallen kann, d.h. für die ein Übergang zum *Nachfolgezustand* $s' := (s_1, \dots, s_{k-1}, s_k - 1, s_{k+1}, \dots, s_n)$ möglich ist.

Analog kann man die Reparatur dieser Komponente wie folgt beschreiben:

Reparatur der k -ten Komponente
 Für alle Zustände $s = (s_1, \dots, s_n) \in S$ mit $s_k = 0$ definiere
 $s' := (s_1, \dots, s_{k-1}, s_k + 1, s_{k+1}, \dots, s_n)$
 $t(s, s') := 1/MTTR_k$

Dies ist eine sehr kompakte und problemnahe Darstellung, die auch die in Abschnitt 1.4.2 vorgeschlagenen Erweiterungen der Modellwelt unterstützt:

Restriktionen, z.B. Ausfall der Komponente k ist nur möglich, wenn Komponente l nicht ausgefallen ist (vgl. Abschnitt 1.4.2), können in die Vorbedingungen eingebaut werden. Im Beispiel oben genügt die Ergänzung von „ $s_l \neq 0$ “.

Folgefehler, z.B. Ausfall von Komponente k zieht mit Wahrscheinlichkeit p einen Ausfall von Komponente l nach sich (vgl. Abschnitt 1.4.2), werden modelliert, indem der Ausfall von Komponente k durch zwei verschiedene Transitionen modelliert wird:

1. Die erste Transition spezifiziert den Ausfall von Komponente k *ohne* Konsequenzen für Komponente l , die Rate ist $(1 - p)/MTTF_k$.
2. Die zweite Transition spezifiziert den Ausfall von Komponente k *mit* Folgeausfall von Komponente l , die Rate ist $p/MTTF_k$.

Der Nachfolgezustand ist geeignet zu definieren, im Beispiel wird bei der zweiten Transition zusätzlich noch das l -te Element dekrementiert.

Für **Array-Komponenten** aus identischen Komponenten, die deshalb nicht unterschieden werden, (vgl. Abschnitt 1.4.1) wird der Zustandsraum $S_k := \{0, 1, \dots, A\}$ definiert, wobei A die Anzahl identischer Komponenten dieser Array-Komponente ist. Zustand 2 bedeutet dann beispielsweise, daß noch 2 der A Komponenten verfügbar sind. Vorbedingung für einen (Teil-) Ausfall ist dann, daß mindestens eine der identischen Komponenten noch verfügbar ist. Die Ausfallrate entspricht der mit der Anzahl noch verfügbarer Unter-Komponenten multiplizierten Ausfallrate einer einzelnen Unter-Komponente:

(Teil-) Ausfall der Array-Komponente k
 Für alle Zustände $s = (s_1, \dots, s_n) \in S$ mit $s_k > 0$ definiere
 $s' := (s_1, \dots, s_{k-1}, s_k - 1, s_{k+1}, \dots, s_n)$
 $t(s, s') := s_k/MTTF_k$

Die Spezifikation für die Transitionen eines Reparatur-Ereignisses in dieser Array-Komponente werden analog (mit Vorbedingung $s_k < A$, $s_k + 1$ statt $s_k - 1$ und Reparaturrate $(A - s_k)/MTTR_k$) vorgenommen.

Zusammenfassend:

Die Spezifikation der Transitionen wird durch eine Aufzählung der verschiedenen Ereignistypen in folgender Notation vorgenommen:

Name

Für alle Zustände $s = (s_1, \dots, s_n) \in S$ mit Vorbedingungen an s definiere

$s' :=$ Nachfolgezustand für s

$t(s, s') :=$ Rate für Wechsel von s nach s'

Partition

Durch eine Partition

$$P = \{P_0, \dots, P_{k-1}\}, \quad k \geq 1, \quad P_i \cap P_j = \emptyset, \text{ falls } i \neq j, \quad S = \bigcup_{P_i \in P} P_i$$

werden die Systemzustände in Klassen P_i eingeteilt. Diese Klassen werden auch *Makro-Zustände* oder *globale Zustände* genannt. Im einfachsten Fall gibt es zwei Klassen, P_0 für die Menge aller Zustände, in denen das System ausgefallen ist und P_1 für die Zustände, in denen es nicht ausgefallen, also verfügbar ist.

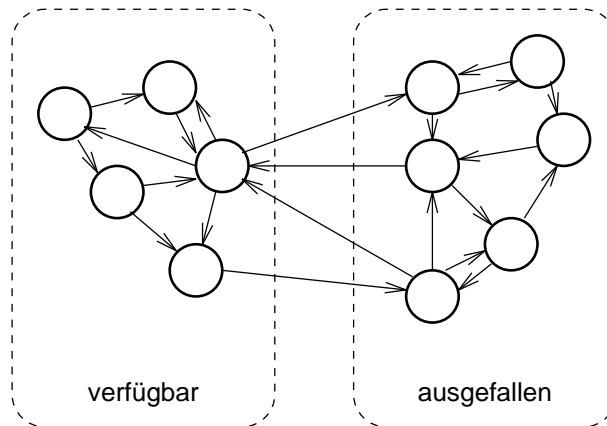


Abbildung 2.7: Partition des Zustandsraums in zwei Makrozustände

Soll feiner differenziert werden, also zusätzlich noch zwischen globalen Zuständen mit verschiedenem Grad an Verfügbarkeit unterschieden werden, so wird die Klasse P_1 noch weiter unterteilt. Die Klasse P_0 sollte stets der totalen Unverfügbarkeit entsprechen.

Die Mengen P_i können auf verschiedenste Weisen spezifiziert werden, beispielsweise durch Prädikate (also Bool'sche Ausdrücke) über den Zuständen der atomaren Komponenten

$$\text{Pr}_i : S \longrightarrow \{\text{wahr}, \text{falsch}\},$$

die dann die Mengen P_i durch

$$P_i = \{(s_1, \dots, s_n) \in S \mid \text{Pr}_i(s_1, \dots, s_n) = \text{wahr}\}$$

charakterisieren.

Die Spezifikation der Partition (Makro-Zustände) wird durch eine Aufzählung der sie charakterisierenden Prädikate vorgenommen:

$$\begin{aligned} \text{Pr}_0(s) &:= \dots \\ &\vdots \\ \text{Pr}_{k-1}(s) &:= \dots \end{aligned}$$

2.3.2 Lösung

Von Interesse sind letztlich natürlich die Verfügbarkeit p (= Wahrscheinlichkeit für einen Zustand $s \in P_1$), Ausfallwahrscheinlichkeit q (= Wahrscheinlichkeit für einen Zustand $s \in P_0$), $MTTF$ und $MTTR$ des Systems.

Mit dem Markov-Modell lassen sich (unter geeigneten Voraussetzungen, die hier aber im allgemeinen erfüllt sind) durch Aufstellen und Lösen des „Gleichungssystems des globalen Gleichgewichts“

- die Aufenthaltswahrscheinlichkeiten für die Makrozustände P_i und
- die Übergangsraten zwischen den Makrozuständen

ermitteln. Die Verfügbarkeit p und Ausfallwahrscheinlichkeit q liegen somit also unmittelbar nach der Lösung des Markov-Modells vor. Die ebenfalls ermittelte Rate r für den Übergang von P_1 = „verfügbar“ nach P_0 = „ausgefallen“ läßt eine Berechnung von $MTTF$ und $MTTR$ durch Little's Law zu:

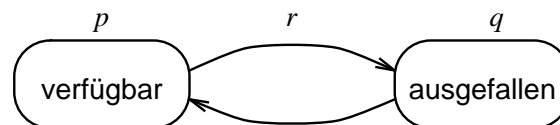


Abbildung 2.8: Makrozustände

Bei Interpretation der Zustandsaufenthaltswahrscheinlichkeiten als mittlere Populationen, der $MTTF$ und $MTTR$ als mittlere Verweilzeiten und der Zustandsübergangsraten als Durchsätze lassen sich durch Little's Law

$$\begin{array}{rclcl}
 \text{mittlere Population} & = & \text{mittlere Verweilzeit} & \times & \text{Durchsatz} \\
 p & = & MTTF & \times & r \\
 q & = & MTTR & \times & r
 \end{array}$$

die mittleren Verfügbarkeits- und Reparaturzeiten berechnen durch

$$MTTF = p/r,$$

$$MTTR = q/r.$$

Ähnliche Formeln gelten für den (hier nicht weiter behandelten) Fall, daß die Makrozustände feiner in Zustände mit verschiedenem Grad von Teilverfügbarkeit unterteilt werden.

2.3.3 Tool-Unterstützung

Zur rechnergestützten Spezifikation und Lösung des Markov-Modells eignet sich das Software-Tool **USENUM** von M. Sczittnick [Bu94], da es eine Spezifikation des Modells in der hier dargestellten Form (vgl. Anhang A) erlaubt und eine Lösung (u.a. auch) auf Multi-Prozessorsystemen¹ mittels verschiedener numerischer Verfahren ermöglicht.

¹es existieren unter anderem Versionen für SOLARIS (SunOS 4.1 und 5.3). USENUM läßt sich auf viele weitere Systeme portieren, da es in C unter Benutzung von Standard-Bibliotheken implementiert ist.

2.4 Simulatives Modell

Ein simulatives Modell bietet im Unterschied zu den bisher vorgestellten Analysetechniken den Vorteil, beliebige Verteilungen für Ausfall- und Reparaturzeiten einsetzen zu können. Auch das Problem einer Zustandsraumexplosion wie bei komplexen Markov-Modellen tritt bei einer Simulation nicht auf.

Als nachteilig hingegen kann sich die Laufzeit der Modelle herausstellen, die benötigt wird, um statistisch gesicherte Ergebnisse zu erhalten. Hier spielt einerseits die Komplexität der Modelle eine Rolle, andererseits aber auch das Verhältnis der Anzahlen verschiedener Ereignisse zueinander, welches i.d.R. durch die Größenordnungen der verschiedenen im Modell vorkommenden Zeiten oder Raten gegeben ist.

Nachfolgend wird ein Ansatz vorgestellt, wie sich die eingeführten hierarchischen Verfügbarkeitsmodelle auf ein simulatives Modell abbilden lassen, welches mit HIT [Bei88] gelöst werden kann. Dieser Ansatz bildet die Baumstruktur der hierarchischen Modelle auf ein HIT-Modell ab, welches die ursprüngliche Struktur wiedergibt. Es sind sicher auch Alternativen denkbar, welche – in Analogie zu den Markovschen Modellen – die hierarchische Struktur auf ein flaches Modell abbilden. Der vorgestellte Ansatz bietet jedoch den Vorteil, daß sich Ergebnisse nicht nur für das Gesamtmodell, sondern auch für die im Modell enthaltenen zusammengesetzten Komponenten im selben Simulationslauf ermitteln lassen, falls dies von Interesse ist.

Die – nicht unbedingt triviale – Umsetzung ist maßgeblich durch die Erweiterungen der Modellwelt (Folgefehler, etc.) bedingt. Eine Umsetzung der analytisch lösbaren Modellklasse würde bedeutend einfacher aussehen.

2.4.1 Struktur und Spezifikation des HIT-Modells

Das zu erzeugende HIT-Modell ist derartig strukturiert, daß sich auf oberster Ebene des Modells *alle* atomaren Komponenten wiederfinden. Das bedeutet insbesondere, daß auch mehrfach vorkommende identische Komponenten (d.h. solche, die durch Array-Bildung zusammengesetzter Komponenten entstehen) jeweils einzeln umgesetzt werden müssen. Die einzige Ausnahme von dieser Regel besteht bei Arrays atomarer Komponenten.

Atomare Komponenten

Jede atomare Komponente K wird durch zwei *Services* repräsentiert: seien sie K_{up} und K_{down} genannt. Für jede atomare Komponente wird bei Simulations-Start genau ein Prozeß K_{up} erzeugt (per `create`), so daß die Simulation im Zustand vollständiger Verfügbarkeit gestartet wird.

Jeder dieser Prozesse setzt zu Auswertungszwecken einen Dienstaufwurf an seine zugehörige zusammengesetzte Komponente ab, um dieser mitzuteilen, daß die entsprechende atomare Komponente verfügbar ist. Anschließend wartet der Prozeß eine gemäß der Verfügbarkeitsdauer verteilte Zeitspanne, um nach Ablauf dieser Dauer einen entsprechenden Prozeß K_{down} zu erzeugen und zu terminieren.

Dieser Prozeß vom Typ K_{down} setzt seinerseits wieder einen Dienstaufwurf an seine übergeordnete zusammengesetzte Komponente ab, um dieser den Eintritt der Unverfügbarkeit mitzuteilen. Nach Ablauf der entsprechenden Reparaturdauer erzeugt der Prozeß dann wieder den zugehörigen Prozeß K_{up} . Zur Veranschaulichung wird dieser Mechanismus in Abb. 2.9 nochmals graphisch dargestellt.

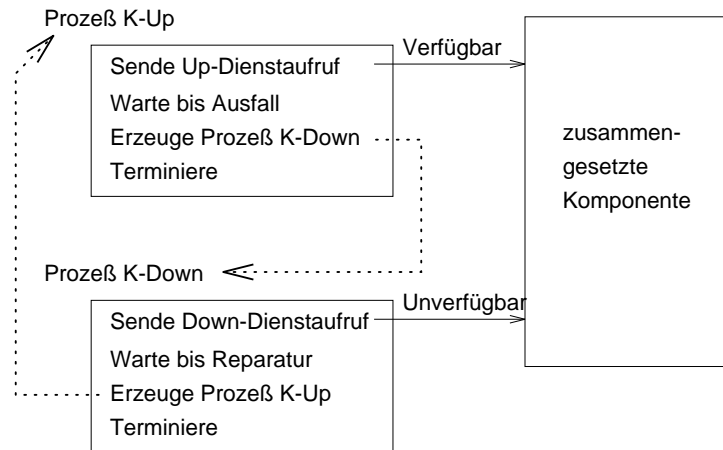


Abbildung 2.9: Simulation atomarer Komponenten mittels zweier Dienste

Ist eine atomare Komponente K nicht in Modellwelt-Erweiterungen wie Folgefehler oder Restruktionen involviert, so kann der Mechanismus der wechselseitigen Prozeßerzeugung entfallen und durch einen einzigen zyklischen Prozeß ersetzt werden, der im Wechsel die Dienstaufrufe $K - Up$ und $K - Down$ an seine übergeordnete Komponente absetzt. Dieses Vorgehen wird für die Implementierung einer automatischen Umsetzung aus Effizienzgründen auf jeden Fall empfohlen.

Atomare Array-Komponenten

Bei einer atomaren Array-Komponente A läßt sich der Vorteil ausnutzen, daß die Dienste A_{up} und A_{down} identische Eigenschaften haben und daher nur einmal spezifiziert werden müssen. Bei der initialen Prozeß-Erzeugung muß dann darauf geachtet werden, daß der Prozeß A_{up} entsprechend der Anzahl Array-Elemente oft erzeugt wird (`create`).

Zusammengesetzte Komponenten

Eine zusammengesetzte Komponente Z bietet für jede in ihr enthaltene Komponente E die beiden Dienste E_{up} und E_{down} an. Diese Dienste werden nach Ausfall bzw. Reparatur der Komponente E aufgerufen, um der zusammengesetzten Komponente Z die Zustandsänderung mitzuteilen.

Um den eigenen Zustand ermitteln zu können, benötigt die Komponente Z für jede in ihr enthaltene Komponente eine Zustandsvariable, um deren Zustand mitzuhaltten. Wird nun ein Dienstaufruf an die Komponente Z abgesetzt, so bedeutet dies eine Änderung der betreffenden Variablen. Anschließend wird der Gesamt-Zustand der Komponente Z überprüft, ob die eingetretene Zustandsänderung von Einfluß auf deren Verfügbarkeit gewesen ist. Wenn das der Fall ist (d.h. es resultiert ein Ausfall der Komponente Z , oder deren Verfügbarkeit ist wieder hergestellt), dann setzt die Komponente Z ihrerseits einen entsprechenden Dienstaufruf an die ihr übergeordnete Komponente ab.

Diese Dienstaufrufe dienen also der Mitteilung von Zustandsänderungen und sind mit keinerlei Zeitverbrauch verbunden. Zeitverbräuche für Ausfall und Reparatur werden lediglich in den atomaren Komponenten realisiert.

Zusammengesetzte Array-Komponenten

Zusammengesetzte Array-Komponenten sind prinzipiell dadurch realisierbar, daß jede Instanz dieser Komponente und jeder ihrer Unterkomponenten einzeln umgesetzt wird. Dabei ist auf eine sinnvolle Namensgebung zu achten, um Konflikte zu vermeiden.

Für den Fall, daß die zusammengesetzte Array-Komponente keine Restriktionen und Folgefehler enthält, ist eine Umsetzung in ein HIT-Modell denkbar, welche über Komponenten-Arrays und parametrisierte Dienste realisiert wird. Dies bedeutet jedoch in diesem Fall lediglich eine Reduzierung des Modelltextes; der Simulationsaufwand nimmt durch die erst bei Laufzeit auszuwertenden Komponenten-Zuordnungen eher zu. Daher wird auf diesen Fall nicht weiter eingegangen.

Geteilte Komponenten

Geteilte Komponente lassen sich auf äußerst triviale Weise realisieren: ändert sich der Zustand einer solchen Komponente (d.h. fällt sie aus, oder ist sie repariert worden), so wird an jede der übergeordneten Komponenten der entsprechende Dienstauftrag abgesetzt.

Solange die bool'schen Funktionen der übergeordneten Funktionen alle monoton sind (im vorgestellten Modellkonzept sind sie dies, es sind jedoch entsprechende Erweiterungen denkbar), entstehen durch den Einsatz geteilter Komponenten keinerlei Hazard-Probleme (vgl. z.B. [OV89]). Sind die Funktionen jedoch nicht monoton (d.h. die Zustandsänderung einer Komponente kann an einer Stelle zur Unverfügbarkeit, an einer anderen zur Verfügbarkeit führen), so kann es zu einem Hazard kommen, d.h., daß sich der Zustand des Gesamtsystems in Nullzeit *mehrfach* ändert.

Dies ist aufgrund der fehlenden Zeitdauer bei einer reinen Verfügbarkeitsanalyse ohne Belang. Sollen jedoch auch MTTF und MTTR bestimmt werden, so kann ein doppelter Wechsel (Verfügbar \rightarrow Unverfügbar \rightarrow Verfügbar), obwohl er in Nullzeit geschieht, dazu führen, daß statt eines langen zwei kürzere Intervalle in die Auswertung der MTTF eingehen.

Sollten tatsächlich nicht-monotone bool'sche Funktionen mit geteilten Komponenten eingesetzt werden, so sind zusammengesetzte Komponenten (zumindest dort, wo Zeiten ausgewertet werden sollen) mit einer zusätzlichen Logik auszustatten, um solche Nullzeit-Hazards herauszufiltern. In der Regel unterliegen die eingesetzten Funktionen jedoch der Monotonie-Eigenschaft, so daß hier eine genauere Untersuchung nicht nötig erscheint.

Restriktionen

Die Eigenschaft der Restriktion soll das Ausfallen einer atomaren Komponente unterbinden, wenn der Modellzustand gerade eine bestimmte Eigenschaft erfüllt. Um dies zu realisieren, benötigt eine solche Komponente Zugriff auf den für sie relevanten Teil des Modellzustands.

Dies läßt sich einfach und performant durch den Einsatz entsprechender global zugreifbarer Variablen realisieren, in denen der Zustand der abzufragenden Komponenten zusätzlich mitgehalten wird und welche von anderen Diensten aus abgefragt werden können. Der üblicherweise existierende Nachteil der Seiteneffekte globaler Variablen ist in diesem Fall nicht gegeben, da genau spezifiziert ist, welche Dienste diese Variablen schreiben und welche sie lesen dürfen und für die Einhaltung dieser Spezifikation durch den automatischen Codegenerator gesorgt wird. Dennoch ist natürlich auch eine funktionale Lösung möglich, obwohl sie aus Performance-Gründen nicht empfohlen wird.

Die Restriktion einer Komponente K wird dann so realisiert, daß das Prozeßmuster des zugehörigen Dienstes K_{up} wie in Abb. 2.10 dargestellt abgeändert wird. Dies bedeutet, daß ein potentielles Ausfallereignis komplett ignoriert wird, solange die Restriktionsbedingung erfüllt ist.

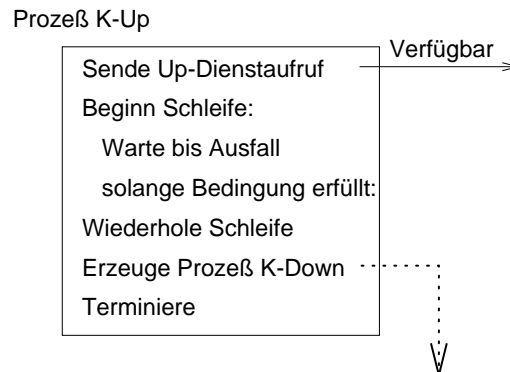


Abbildung 2.10: Berücksichtigung von Restriktionen in atomaren Komponenten

Eine alternative Semantik der Restriktion könnte darin bestehen, daß der Ausfall einer Komponente K durch die Restriktion lediglich solange herausgezögert wird, bis die Bedingung nicht mehr erfüllt ist. Dieses Verhalten läßt sich durch Einsatz eines Semaphors erreichen, wodurch zusätzlich der Einsatz globaler Variablen überflüssig wird.

Hierzu ist es nötig, daß jedesmal, wenn sich die Restriktionsbedingung im Rahmen eines Zustandswechsels ändert, der entsprechende Dienst des Semaphors aufgerufen werden muß. D.h. wechselt der Zustand der Bedingung in „erfüllt“, so wird die P-Operation ausgeführt und somit ein Ausfall von K unterbunden, wechselt der Zustand wieder zurück, so wird die V-Operation ausgeführt und ein Ausfall von K ist wieder möglich. Innerhalb des Dienstes K_{up} ist dann lediglich die Erzeugung des Ausfallprozesses durch die Semaphor-Dienste P und V als kritischer Bereich zu markieren. Dies ist in Abb. 2.11 dargestellt.

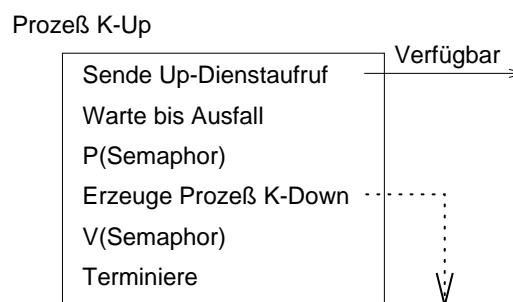


Abbildung 2.11: Berücksichtigung von Restriktionen durch Semaphor

Da innerhalb dieses kritischen Bereiches kein Zeitverbrauch stattfindet, besteht solange keine Deadlock-Gefahr, wie in den verwendeten Ausdrücken für die Restriktion keinerlei Selbstbezug (direkt oder indirekt) auftritt.

Folgefehler

Der Folgefehler ist am komplexesten von allen Erweiterungen der Modellwelt zu realisieren. Dabei ist es möglich, daß das Ausfallereignis einer beliebigen Komponente mit einer bestimmten Wahrscheinlichkeit den Ausfall einer (oder mehrerer) anderen atomaren Komponente induziert.

Die Beschränkung auf atomare Komponenten ist u.a. deshalb nötig, weil es bei zusammengesetzten Komponenten unklar ist, welches Ereignis in diesem Fall die Reparatur der Komponente bewirkt.

Wird ein Folgefehler ausgelöst, so sind zwei wesentliche Punkte zu beachten:

- Falls die betroffene Komponente bereits ausgefallen ist, so zeigt der Folgefehler keinerlei Auswirkung. Dies läßt sich durch eine Abfrage der entsprechenden Zustandsvariable erreichen, ähnlich wie bei der Realisierung der Restriktion.
- Es muß dafür gesorgt werden, daß der noch existente Prozeß K_{up} keinerlei relevante Ereignisse mehr auslöst, da er obsolet geworden ist. Hier sind zwei Mechanismen vorzusehen:
 - a) Solange der neue Prozeß K_{down} aktiv ist, kann der Prozess K_{up} sowohl über die Population der Prozesse K_{down} als auch über die Zustandsvariable der eigenen Komponente K feststellen, ob seine Komponente K bereits aus einem anderen Grund ausgefallen ist. In diesem Fall terminiert der Prozeß K_{up} einfach, ohne einen weiteren K_{down} zu erzeugen.
 - b) Nach Beendigung des neu erzeugten Prozesses K_{down} generiert dieser einen neuen Prozeß K_{up} und terminiert selber. Dennoch muß der alte Prozeß K_{up} die Information erhalten, daß er nicht mehr aktuell ist. Dies läßt sich durch eine in der Komponente enthaltene zyklische Zählvariable erreichen, welche bei jedem `create` von K_{up} weitergezählt wird. Stellt K_{up} vor seiner Beendigung fest, daß diese Zählvariable inzwischen weitergezählt wurde, dann überspringt er die Prozeßgenerierung von K_{down} , da ein anderer Prozeß diese Aufgabe übernommen hat.

Werden diese Punkte beachtet, so ist gewährleistet, daß die Zahl der aktiven Prozeß-Paare konstant bleibt.

2.4.2 Lösung

Bei der Auswertung ist es möglich, sowohl die Verfügbarkeit als auch MTTF und MTTR (und deren Verteilung) an beliebigen Komponenten zu bestimmen. Sinnvollerweise beschränken wir uns hier auf zusammengesetzte Komponenten, da die entsprechenden Werte der atomaren Komponenten bereits als Eingabeparameter vorliegen. Das Gesamtmodell ist in diesem Fall lediglich der Sonderfall einer zusammengesetzten Komponente, die in keiner weiteren Komponente enthalten ist.

Zur Auswertung der Verfügbarkeit ist in der betreffenden Komponente ein STATE-Stream zu deklarieren, zur Bestimmung von Ausfall- und Verfügbarkeits-Dauern benötigt man jeweils einen Stream vom Typ EVENT.

Jedesmal, wenn sich der Zustand der Komponente ändert, muß dies beim Update der entsprechenden Streams berücksichtigt werden. Für den STATE-Stream zur Bestimmung der Verfügbarkeit bedeutet dies:

- Beim Wechsel „Unverfügbar“ \rightarrow „Verfügbar“ muß ein `UPDATE stream BY 1` geschehen,
- beim Wechsel „Verfügbar“ \rightarrow „Unverfügbar“ geschieht ein `UPDATE stream BY -1`.

Dadurch wird erreicht, daß der Stream während verfügbarer Zeiten den Wert 1 annimmt und während der übrigen Zeit den Wert 0.

Die `UPDATE`-Statements für den Event-Stream zur Bestimmung der MTTF, sind wesentlich komplexer angelegt. Hier muß der Zeitpunkt des letzten Zustandswechsels („Unverfügbar“ \rightarrow

„Verfügbar“) in einer Hilfsvariable mitgehalten werden (Zugriff über die Systemvariable `time`). Beim nächsten Wechsel „Verfügbar“ → „Unverfügbar“ kann dann der Update des Streams um die betreffende Zeit-Differenz geschehen.

Zur Bestimmung der MTTR ist analog bei den umgekehrten Zustandsübergängen vorzugehen.

2.4.3 Zusätzliche Erweiterungen der vorgestellten Modellwelt

Existiert ein automatischer Umsetzer der Spezifikationsprache in ein ablauffähiges HIT-Modell, so ist es, sofern der erzeugte Code – z.B. durch entsprechende Einrückungen – lesbar konzipiert wurde, durchaus möglich, den erzeugten Code manuell zu überarbeiten und ursprünglich nicht vorgesehene Eigenschaften in das Modell zu integrieren. Man muß sich dabei jedoch dessen bewußt sein, daß dieses Vorgehen, je umfangreicher die vorgenommenen Änderungen sind, sehr fehleranfällig werden kann. Daher sollte ein solches Vorgehen nur in Ausnahmefällen oder zu Testzwecken eingesetzt werden.

Sind manuelle Änderungen notwendig, insbesondere, wenn sie häufig auftreten, ist es besser, die gewünschten Änderungen als Erweiterung in die Spezifikationsprache aufzunehmen und vom Codegenerator automatisch umsetzen zu lassen. Dies hat insbesondere den Vorteil, daß Modelle einfacher und schneller modifiziert werden können, ohne nach jeder Codegenerierung alle manuellen Änderungen am generierten Code nachvollziehen zu müssen.

Denkbare Änderungen an der vorgestellten Beschreibungssprache ist z.B. die Erweiterung der Angabe von Ausfall- und Reparaturzeiten mit benutzerspezifisierten Verteilungsfunktionen. Möglich wäre auch die Integration nicht-monotoner bool'scher Funktionen, wobei die bereits erwähnte Hazard-Problematik beachtet werden muß. Interessant wäre die Einführung von Reparatureinheiten, wodurch die Reparaturzeiten zusätzlich um durch Blockierungseffekte verursachte Zeiten erhöht würden.

Kapitel 3

Beispielanwendung

3.1 Das XCS als hierarchisches Verfügbarkeits-Modell

Abbildung 3.1 zeigt eine graphische Darstellung des hierarchischen Verfügbarkeits-Modells für das XCS (Cross Coupled System). Das zum Vergleich herangezogene Einzelrechner-Modell (Abb. 3.2) besteht aus nur einem Knoten, der acht statt vier Prozessoren (HZ bzw. HZ1 ... HZ8) hat. Ansonsten ist es mit dem XCS-Modell identisch.

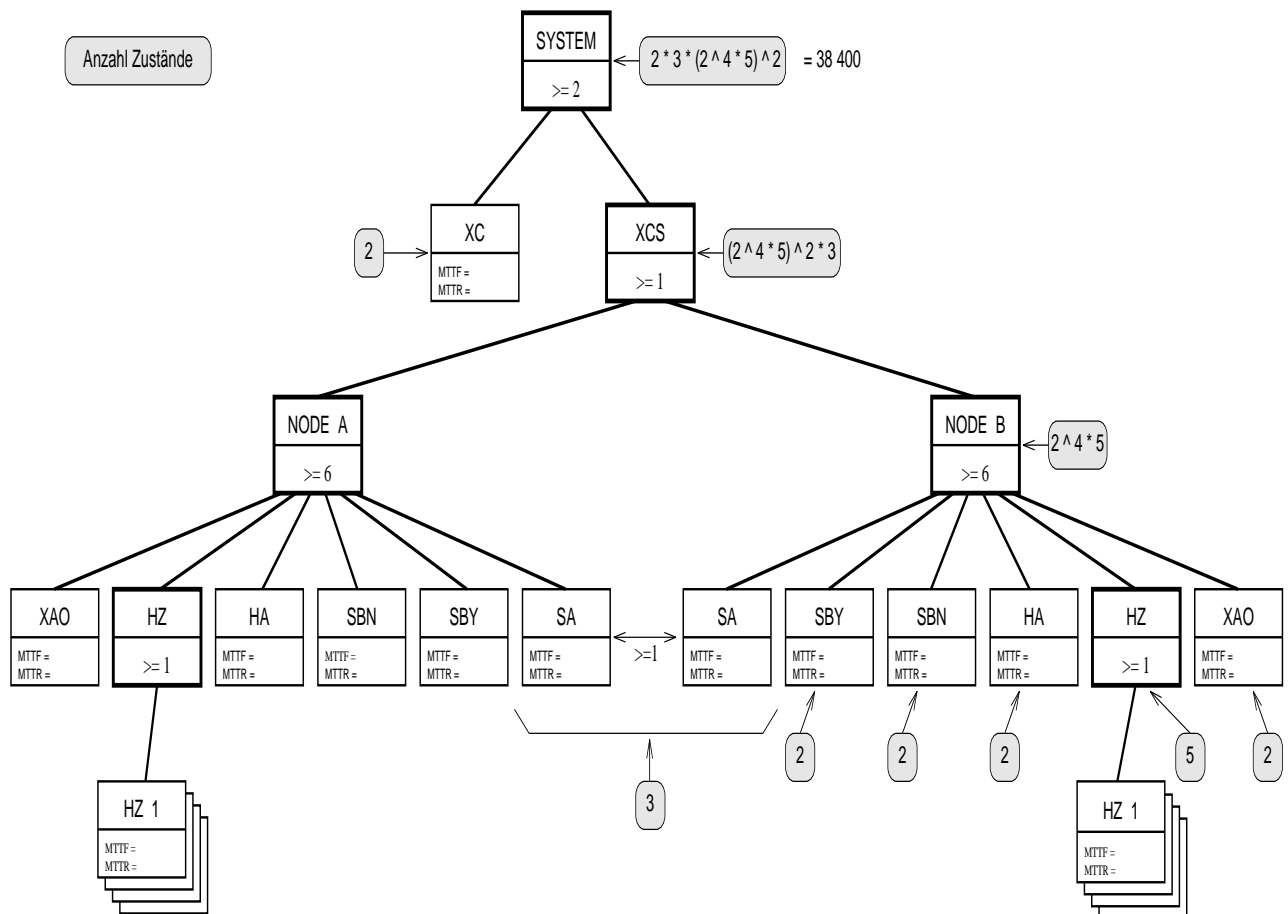


Abbildung 3.1: Das XCS als hierarchisches Verfügbarkeits-Modell

Die grau unterlegten Kästchen neben den Komponenten veranschaulichen die Berechnung der Zustandsraum-Größe im Markov-Modell und stellen *keine* Elemente der Modellwelt dar. Man

beachte, daß die Komponenten SA nicht mit jeweils zwei, sondern mit zusammen drei Zuständen angegeben sind, da beide Komponenten wegen der Restriktion nicht gleichzeitig ausfallen können. Für das XCS-Modell ergeben sich 38400 Zustände, für das Einzelrechner-Modell nur 576.

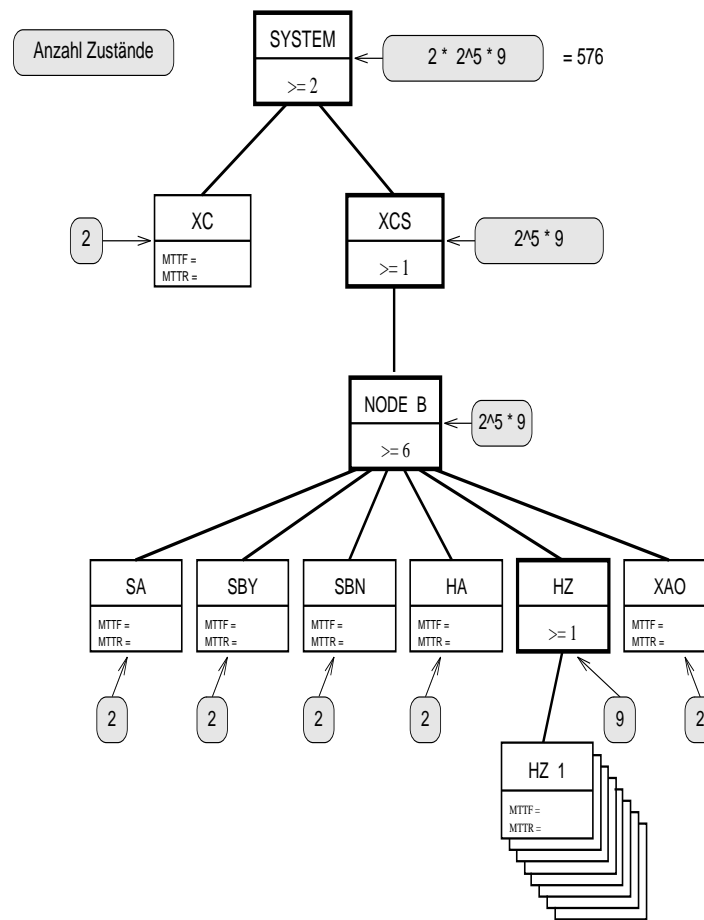


Abbildung 3.2: Der Einzelrechner als hierarchisches Verfügbarkeits-Modell

3.1.1 Die Komponenten des XCS-Modells

SYSTEM: das System ist verfügbar, wenn die Stromversorgung **und** der eigentliche Rechner verfügbar sind.

XC: die Stromversorgung ist eine atomare Komponente mit $MTTF= 2731.49$ Tage und $MTTR= 135.17$ Minuten.
Ausfallursache: Stromausfall

XCS: das XCS ist verfügbar, wenn mindestens einer der beiden Knoten verfügbar ist.

Node A, Node B: ein Knoten ist verfügbar, wenn seine sechs Unter-Komponenten XAO, HZ, HA, SBN, SBY und SA alle verfügbar sind.

XAO: diese atomare Komponente modelliert Anwender / Operatoren.
 $MTTF= 1821.03$ Tage und $MTTR= 32.78$ Minuten.
Ausfallursache: Anwender- / Operatorfehler

HZ: die Zentraleinheit ist eine Array-Komponente, die verfügbar ist, wenn mindestens einer der vier (acht) Prozessoren verfügbar ist.

HZ1 ... HZ4 (HZ8): die einzelnen Prozessoren der Zentraleinheit sind identische atomare Komponenten mit $MTTF= 327.67$ Tage und $MTTR= 175.50$ Minuten.
Ausfallursache: Prozessorausfall

HA: diese atomare Komponente modelliert Hardwareänderungen
 $MTTF= 227.47$ Tage und $MTTR= 233.21$ Minuten.
Ausfallursache: Hardwareänderung

SBN: Fehler der Betriebssystemsoftware werden mit dieser atomaren Komponente modelliert.
 $MTTF= 321.25$ Tage und $MTTR= 160.73$ Minuten.
Ausfallursache: ungeplante Betriebsunterbrechung bei Fehler

SBY: Änderungen der Betriebssystemsoftware werden mit dieser atomaren Komponente modelliert. $MTTF= 455.22$ Tage und $MTTR= 59.56$ Minuten.
Ausfallursache: geplante Betriebsunterbrechung bei Änderung

SA: Softwareänderungen werden mit dieser atomaren Komponente modelliert.
 $MTTF= 186.19$ Tage und $MTTR= 77.26$ Minuten.
Ausfallursache: Softwareänderung. Softwareänderungen werden nicht gleichzeitig an beiden Knoten durchgeführt. Dies wird durch die Restriktion „ ≤ 1 “ zwischen den Komponenten SA der beiden Knoten ausgedrückt.

3.1.2 Ergebnisse

Tabelle 3.1 faßt die mit dem Wahrscheinlichkeiten-Modell und Approximation bzw. mit dem Markov-Modell ermittelten Verfügbarkeitsmaße zusammen, welche offensichtlich (im Rahmen der Rechengenauigkeit) übereinstimmen. Es sei noch einmal bemerkt, daß sich mit Wahrscheinlichkeiten-Modellen Abhängigkeiten zwischen Komponenten nicht modellieren lassen.

Während sich die mittlere Reparaturzeit $MTTR$ des XCS im Vergleich zum Einzelrechner nicht wesentlich verbessert (um weniger als 6 Minuten), sind die Verfügbarkeit p und die mittlere Verfügbarkeitszeit $MTTF$ für das XCS drastisch besser (Faktor 40).

Beim XCS tritt ein Totalausfall im Mittel seltener als einmal alle sechs Jahre auf, d.h. dem Betrieb eines XCS wird bei einer angenommenen Nutzungsdauer von fünf Jahren wahrscheinlich eher durch ein Upgrading des Kunden auf neuere Hardware als durch einen Totalausfall seines XCS ein Ende bereitet. Dagegen wird sich ein Kunde an zwischenzeitliche Totalausfälle seines Einzelrechnersystems gewöhnen müssen, da mit ihnen im Mittel fast sechsmal im Jahr zu rechnen ist.

	Verfügbarkeit p	Ausfallwahrscheinlichkeit q	$MTTF$ [Jahre]	$MTTR$ [Stunden]	Lösungsmethode
Einzelrechner	0.998516157895	$1.483842105492 \cdot 10^{-3}$	0.17132	2.2302	Wk. & Approx.
	0.998516	$1.48384 \cdot 10^{-3}$	0.17132	2.2302	Markov-Modell
XCS mit Restriktion	—	—	—	—	Wk. & Approx.
	0.999964	$3.63820 \cdot 10^{-5}$	6.7110	2.1389	Markov-Modell
XCS ohne Restriktion	0.999963534975	$3.6465025219 \cdot 10^{-5}$	6.6606	2.1277	Wk. & Approx.
	0.999964	$3.64650 \cdot 10^{-5}$	6.6606	2.1277	Markov-Modell

Tabelle 3.1: Ergebnisse

Anhang A

Spezifikation mittels USENUM

Zur Beschreibung und Lösung des Markovmodells wird **USENUM** von M. Sczittnick [Bu94] eingesetzt.

A.1 Umsetzung von atomaren Komponenten

Für jede atomare Komponente werden *MTTF*, *MTTR*, Zustandsvariable mit Startzustand und zwei Transitionen (in der Datei `model.mod`) definiert:

```
CONSTANTS                                # Anfang Konstantendefinition
  component_A_mttr                        # Name der Konstanten
  double                                  # Wertebereich
  77.26 * 60.0                            # Wert
  component_A_mttf                        # Name der Konstanten
  double                                  # Wertebereich
  31.41 * 24.0 * 60.0 * 60.0             # Wert
  #
STATE                                     # Anfang der Zustandsvariablendeklaration
  component_A_state                        # Name der Zustandsvariablen
  short                                   # Wertebereich
  #
INITIAL_STATES                           # Anfang der Definition von Initialzustaenden
  s->component_A_state = 1;               # 's->' bezeichnet den aktuellen Zustand
  #
TRANSITION                                # Definition einer Transition fuer Ausfall
  component_A_failure                     # Name
  s->component_A_state == 1               # Vorbedingung: Komponente ist verfuegbar
  %                                       # Ende eines Blocks, der mehrzeilig sein kann
  next->component_A_state--;              # 'next->' bezeichnet den Folgezustand
  %                                       #
RATE                                      # Definition der Uebergangsrates
  1.0/c->component_A_mttf                 # = Kehrwert der MTTF
  %                                       #
  %                                       # 'c->' bezeichnet die Menge der Konstanten
  %                                       #
TRANSITION                                # Definition einer Transition fuer Reparatur
  component_A_repair                      # Name
  s->component_A_state == 0               # Vorbedingung: Komponente ist ausgefallen
  %                                       # Ende eines Blocks, der mehrzeilig sein kann
  next->component_A_state++;              # 'next->' bezeichnet den Folgezustand
  %                                       #
```

```

RATE                                     # Definition der Uebergangsrates
  1.0/c->component_A_mttr                # = Kehrwert der MTTR
%                                         #
%                                         #

```

A.2 Umsetzung von Restriktionen

Bei atomaren Komponenten, zwischen denen Abhängigkeiten bestehen, müssen die entsprechenden Restriktionen in den Vorbedingungen der Transitionen berücksichtigt werden:

```

TRANSITION                               #
  component_A_failure                     #
  (s->component_A_state == 1) &&          # Vorbedingung: Komponente ist verfuegbar
  (s->component_B_state == 1)            # und andere Komponente ist auch verfuegbar
%                                         #
  next->component_A_state--;              #
%                                         #
RATE                                       #
  1.0/c->component_A_mttf                 #
%                                         #
%                                         #

```

Eine entsprechende Spezifikation für Abhängigkeiten zwischen nicht-atomaren Komponenten steht noch aus.

A.3 Umsetzung von Array-Komponenten

Die Umsetzung von Array-Komponenten unterscheidet sich nicht wesentlich von der Umsetzung atomarer Komponenten. Die Zustandsvariable, die bei atomaren Komponenten lediglich die Werte 0 und 1 annimmt, wird bei Ausfällen in Array-Komponenten entsprechend erniedrigt bzw. erhöht. Zusätzlich sind noch die Vorbedingungen und Raten zu modifizieren:

```

CONSTANTS                                 #
  component_A_mttr                         #
  double                                   #
  77.26 * 60.0                             #
  component_A_mttf                         #
  double                                   #
  31.41 * 24.0 * 60.0 * 60.0              #
  component_A_dim                          # Dimension der Array-Komponente
  short                                    # = Anzahl identischer Unter-Komponenten
  4                                         #
STATE                                     #
  component_A_state                         #
  short                                    #
INITIAL_STATES                            #
  s->component_A_state = c->component_A_dim; #

```

```

TRANSITION                                #
  component_A_failure                      #
  s->component_A_state > 0                 # Vorbedingung: nicht alle ausgefallen
%                                          #
  next->component_A_state--;              #
%                                          #
RATE                                       # Rate multipliziert mit Anzahl
  s->component_A_state/c->component_A_mttf
%                                          # verfuegbarer Komponenten
%                                          #
TRANSITION                                #
  component_A_repair                      # Vorbedingung:
  s->component_A_state < c->component_A_dim
%                                          # mindestens eine ausgefallen
  next->component_A_state++;              #
%                                          #
RATE                                       # Rate multipliziert mit Anzahl
  (c->component_A_dim - s->component_A_state)/c->component_A_mttr
%                                          # ausgefallener Komponenten
%                                          #

```

A.4 Umsetzung von Komponenten mit Unter-Komponenten

Jeder Komponente K mit Unter-Komponenten K_1, \dots, K_n und $\geq k$ -Abhängigkeit wird zunächst ein logischer Ausdruck der Form

$$(A_1(X) + A_2(X) + \dots + A_n(X) \geq k)$$

zugeordnet. Ist K_i eine atomare Komponente oder eine Array-Komponente, so wird A_i direkt durch die Zustandsvariable der Komponente K_i , z.B. $s \rightarrow K_i_state$ (um $A(s \rightarrow)$, einen boolschen Ausdruck über den aktuellen Zustand zu formen) oder $next \rightarrow K_i_state$ (um $A(next \rightarrow)$, einen boolschen Ausdruck über den Nachfolgezustand zu formen) ersetzt. Andernfalls enthält K_i selbst Unter-Komponenten und A_i wird rekursiv durch den entsprechenden Ausdruck für K_i ersetzt.

Auf diese Weise entsteht ein logischer Ausdruck (A), der den Zustandsraum in zwei Makro-Zustände partitioniert und folgendermaßen (in der Datei `model.qua`) verwendet wird:

```

STATEMEASURE                             # Def. eines zu messenden Makrozustands
  available                               # Name fuer den Makrozustands
DISTRIBUTION                             # Verteilung ermitteln
%                                         # logischer Ausdruck beschreibt Makrozustand
  (A(s->))                                # Wert des Makrozustands =
? 1 : 0                                   # 1, falls log. Ausdruck erfuehlt, sonst 0
%                                         #
COUNTMEASURE                             # Def. einer zu messenden Rate
  avail_to_fail_rate                     # Name fuer die Rate
MEAN                                      # Mittelwert ermitteln
%                                         # Messe Zustandsuebergangsrate
  (A(s->))                                # von Makrozustand = logischer Ausdruck erfuehlt
%                                         #
  !(A(next->))                            # nach Makrozustand = logischer Ausdruck falsch

```

```
% #
% #
```

A.5 Weitere Spezifikationen

In der Datei `model.gen` wird die maximale Größe des Zustandsraums beschränkt:

```
MAXSTATENUMBER
100000
```

Die Lösungsmethode – hier: numerische Analyse des stationären Zustands unter zuhilfenahme einer manuellen Strukturierung des Zustandsraums – wird in der Datei `model.sol` festgelegt:

```
numeric
structured
stationary
```

Der Zustandsraum wird zur Unterstützung des Löser manuell in Makrozustände unterteilt, indem einige sich möglichst schnell ändernde Zustände spezifiziert werden. Diese Definitionen werden in der Datei `model.str` vorgenommen:

```
macrostate
m_1
short
s->component_A_state
%
m_2
short
s->component_B_state
%
%
```

Zur Steuerung der Analyse wird schließlich die Datei `model.con` angelegt. Sie bestimmt auch die Dateinamen für abzulegende Zwischenergebnisse:

```
MODEL
  model.mod
GENERATE
  model.gen
SPACE
  /tmp/model.spa
STRUCTURE
  model.str
SOLUTION
  model.sol
VECTORS
  /dev/null
QUANTITATIVE
  model.qua
```


Literaturverzeichnis

- [Bei88] Beilner, H.; Mäter, J.; Weißenberg, N.:
Towards a Performance Modelling Environment: News on HIT:
Proceedings of the 4th International Conference on Modelling Techniques and Tools
for Computer Performance Evaluation; Palma de Mallorca, September 1988.
- [BFS87] Bradley, P.; Fox, B.L.; Schrage, L.E.:
A Guide to Simulation:
Second Edition, Springer, 1987.
- [Ech90] Echte, K.:
Fehlertoleranzverfahren
Springer, 1990.
- [Gö88] Görke, W.:
Zuverlässigkeitsprobleme elektronischer Geräte
Skript zur Vorlesung WS 1988/89, Universität Karlsruhe, 1988.
- [Lap85] Laprie, J.C.:
Dependable Computing and Fault Tolerance: Concepts and Terminology
Proc. 15th Annual Int. Symp. on Fault Tolerant Computing Systems, 1985.
- [OV89] Oberschelp, W.; Vossen, G.:
Rechneraufbau und Rechnerstrukturen
3. Aufl., Oldenbourg Verlag, ISBN 3-486-21248-6, 1989.
- [Sch94] Schneeweiss, W.:
*Computer Systems Dependability Evaluation Using SyRePa (System Reliability
Package)*
Fern-Universität Hagen, 1994.
- [Bu94] Buchholz, P.; Dunkel, J.; Müller-Clostermann, B.; Sczittnick, M.; Zäske, S.:
Quantitative Systemanalyse mit Markovschen Ketten
Teubner-Texte zur Informatik, Band 8, Teubner, 1994.
- [Se89] Sethi, R.:
Programming Languages – Concepts and Constructs
Addison–Wesley, 1989.
- [Zä95] Zäske, S.:
Verfügbarkeitsanalyse eines Mehrrechnersystems
Universität Dortmund, Informatik IV, Projekt OSMOD'94/'95, 1995.

