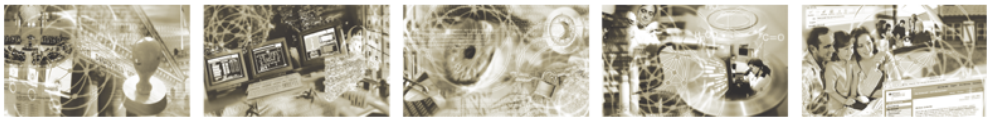
Bundesamt
für Sicherheit in der
Informationstechnik



Band G, Kapitel 7: Prinzipien

Prinzipien der Verfügbarkeit

Hinweis zur Aktualität:

Das HV-Kompendium war letztmalig im Jahr 2013 überarbeitet und aktualisiert worden. Es entsprach in vielen Teilen nicht mehr dem Stand der Technik und ist daher zurückgezogen worden.

Der Grundlagenband – Band G – ist einer der ursprünglichen 4 Bände des HV-Kompendiums. Er wird wegen seiner grundlegenden und immer noch zutreffenden Inhalte zu Informationszwecken weiterhin veröffentlicht.

In den einzelnen Kapiteln wird an einigen Stellen auf die ehemaligen Bände "Band B: Bausteine", "Band M: Maßnahmen" und "Band AH: Architekturmodelle und Hilfsmittel" des zurückgezogenen HV-Kompendiums verwiesen. Auf diese Bände kann nicht mehr zugegriffen werden.

Ebenso wird an einigen Stellen auf Informationen und Dokumente aus dem Internet verwiesen, die über die angegebene URL nicht mehr erreichbar sind.

Bundesamt für Sicherheit in der Informationstechnik
Postfach 20 03 63
53133 Bonn
Tel.: +49 22899 9582-0
E-Mail: hochverfügbarkeit@bsi.bund.de
Internet: https://www.bsi.bund.de

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Prinzipien der Verfügbarkeit

Der vorliegende Beitrag beschreibt grundlegende Prinzipien, die bei der Bereitstellung von IT-Dienstleistungen mit hohen Anforderungen an die Verfügbarkeit eine leitende Rolle spielen. Die Prinzipien werden zunächst in einer übergreifenden und generischen Sichtweise dargestellt. In den weiteren Beiträgen des HV-Kompendiums (Band 2) werden diese Prinzipien in konkreten Technologien und Services umgesetzt. Einige Realisierungsbeispiele verdeutlichen bereits in diesem Dokument den Stellenwert der jeweiligen Prinzipien innerhalb einer HV-Lösung. Die Prinzipien, die Entwurf und Evolution leiten, leisten in ihrem Zusammenspiel einen gemeinsamen Beitrag zur Gestaltung und Optimierung der Verfügbarkeit in HV-Umgebungen.

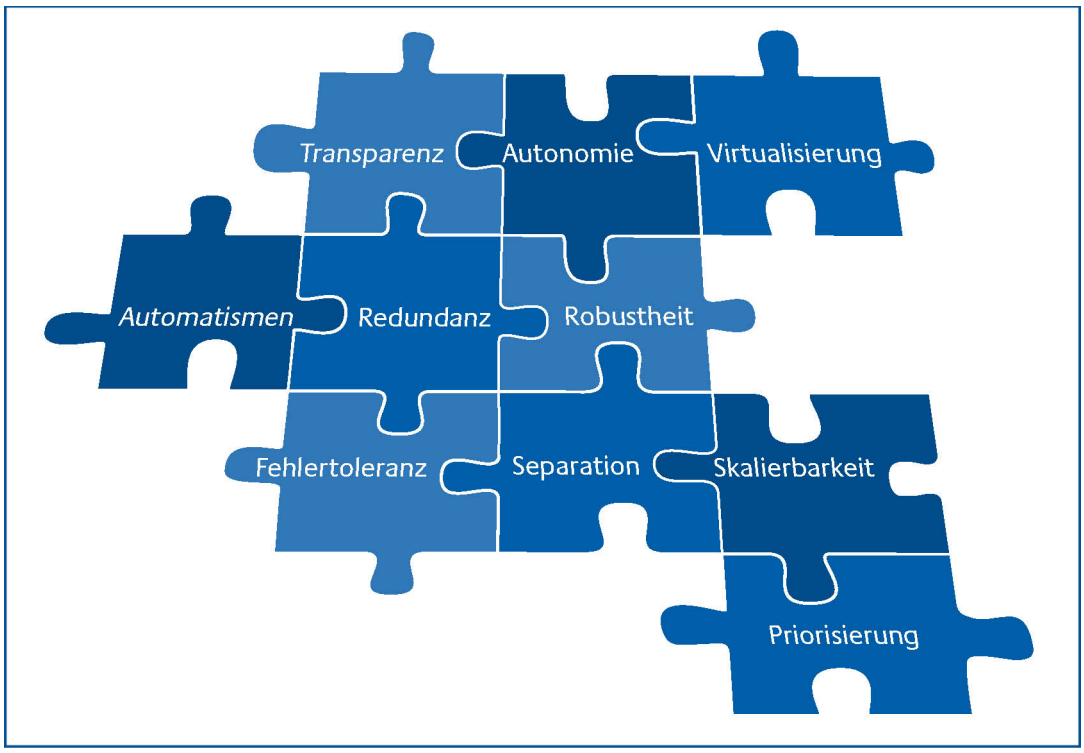


Abbildung 1: Prinzipien für die Realisierung hochverfügbarer IT-Systeme

In den „Standards und Architekturen für E-Government“ (SAGA) sind für die IT-Steuerung des Bundes einige Anforderungskriterien an die IT-Architektur der Bundesverwaltung zusammengestellt [SAGA08]. Diese Anforderungen sind entweder direkt als Prinzip der Verfügbarkeit aufgenommen worden (z. B. Skalierbarkeit) oder sie beeinflussen die hier aufgeführten Prinzipien in einem für die Verfügbarkeit positiven Sinn (z. B. Wartbarkeit → Robustheit). Die Zusammenhänge zwischen den Prinzipien der Verfügbarkeit und den Anforderungskriterien aus SAGA werden im Abschnitt 1.11 erläutert. Die Abbildung 1 liefert einen Überblick über die Prinzipien, die in den folgenden Abschnitten erläutert werden:

- Fehlertoleranz
- Redundanz
- Robustheit
- Separation
- Virtualisierung

- Transparenz
- Skalierbarkeit
- Automatismen
- Priorisierung
- Autonomie

1.1 Redundanz

Die Redundanz stellt ein wesentliches Prinzip zur Realisierung hochverfügbarer Architekturen und IT-Infrastrukturen dar. Unter Redundanz wird allgemein das mehrfache Vorhandensein (*redundare (lat.) = „im Überfluss vorhanden sein, überfließen“*) von gleichartigen Objekten verstanden. Dies können technische Komponenten, Informationen, Services oder auch Personal sein. Redundanz im Umfeld der Hochverfügbarkeit ist normalerweise dadurch gekennzeichnet, dass mehr Betriebsmittel vorhanden sind, als zur Erfüllung der spezifizierten Funktionen eines IT-Systems erforderlich sind.

1.1.1 Redundanzverfahren

Hinsichtlich ihrer Aktivierung und ihrer Merkmale wird zwischen den verschiedenen Redundanzverfahren, wie in Abbildung 2 dargestellt, unterschieden. In der Praxis werden Kombinationen und Mischformen dieser Redundanzverfahren eingesetzt.

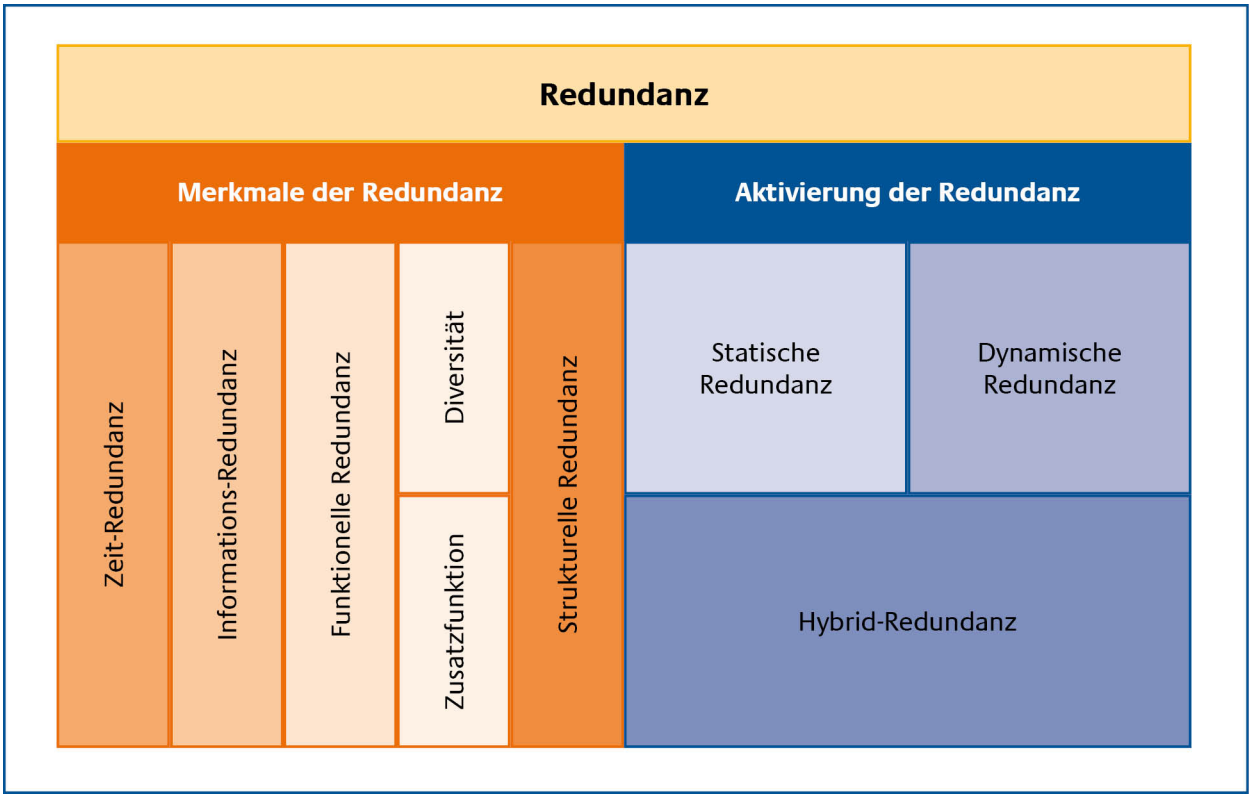


Abbildung 2: Redundanzverfahren [Echt90]

1.1.1.1 Merkmale der Redundanz

Redundanzverfahren lassen sich durch die Merkmale Struktur, Funktion, Information und Zeit charakterisieren. Das Merkmal, welches durch seine besondere Bedeutung das Verfahren entscheidend prägt, wird typischerweise für die Einordnung des Verfahrens herangezogen. Eine Software, die Testfunktionen beinhalten (z. B. Prüfsummenverfahren), wird beispielsweise der funktionellen Redundanz zugeordnet. Die Bereitstellung von funktioneller Redundanz erfordert meist auch strukturelle Redundanz. Die oben genannten Testfunktionen werden durch weitere Komponenten des Systems realisiert. Eine HV-Lösung, die nach dem Redundanzprinzip konzipiert wurde, kann also durchaus mehrere der nachfolgend beschriebenen Merkmale aufweisen.

Strukturelle Redundanz:

Strukturelle Redundanz bezeichnet das mehrfache Vorhandensein von Ressourcen (z. B. Netz- oder Speicherkomponenten), die bei vorausgesetzter Fehlerfreiheit des Systems nicht notwendig wären. Strukturell redundante Komponenten dienen grundsätzlich der gleichen Aufgabenerfüllung und liefern deshalb keine weitere Funktionalität. Es kann sich bei den zusätzlichen Komponenten um aktive oder passive, um baugleiche oder auch andersartige Komponenten handeln.

Funktionelle Redundanz

Die funktionelle Redundanz realisiert zusätzliche, für den Normalbetrieb nicht erforderliche Funktionen. Häufig handelt es sich hierbei um spezielle Fehlertoleranz- oder Zusatzfunktionen, die im Falle eines Fehlerereignisses greifen. Im Unterschied zu den bisher aufgeführten Redundanzmechanismen handelt es sich hierbei nicht um weitere funktional gleichwertige Betriebsmittel, sondern um Zusatzfunktionen, die speziell zur Realisierung von Redundanz eingesetzt werden. Beispiele hierfür sind Algorithmen zur Erzeugung von Paritätsbits, Synchronisationsmechanismen oder Steuerkomponenten für Cluster-Systeme.

Informationsredundanz

Informationsredundanz bezeichnet das Vorhandensein zusätzlicher, über den Nutzinhalt hinausgehender, Informationen. Informationsredundanz dient der Fehlererkennung und häufig auch der Fehlerbehandlung bei der Informationsverarbeitung. Paritätsbits beispielsweise erlauben die Erkennung verfälschter Bits. Über spezielle Kodierverfahren lassen sich auch fehlertolerante Codes (wie z. B. bei Datenträgern) realisieren, die eine Reparatur defekter Informationen erlauben. Informationsredundanz lässt sich grundsätzlich nur zusammen mit einer funktionellen Redundanz (zur Erzeugung der Informationsredundanz) sowie ggf. mit der strukturellen Redundanz (zum Ablegen der zusätzlichen Information) realisieren.

Zeitredundanz

Ebenso wie die Informationsredundanz ist auch die Zeitredundanz eng mit der funktionellen Redundanz verknüpft. Als Zeitredundanz wird die Zeit bezeichnet, die für die Erbringung der funktionellen Redundanz zur Verfügung steht. Beispielsweise die Zeit, die erforderlich ist, einen Fehlercode zu errechnen und abzulegen.

1.1.1.2 Aktivierung der Redundanz

Neben den zuvor aufgeführten Merkmalen der Redundanz ist die Art der Aktivierung eine weitere Eigenschaft durch die Redundanzausprägungen beschrieben werden können. Betrachtet man den Zeitpunkt, zu dem die redundanten Mittel aktiviert werden, so kann man Redundanzverfahren einteilen in statische und dynamische Redundanz.

Statische Redundanz

Die statische Redundanz bezeichnet das Vorhandensein aktiver Betriebsmittel, die bereits während des regulären Betriebes aktiv sind und zur Erfüllung der geforderten Funktionalität beitragen (obwohl sie nicht erforderlich sind). Sie verfügen in der Regel über freie Ressourcen, die im Fehlerfall hinzugezogen werden, um ausgefallene Betriebsmittel (Hot- / Warm-Standby-Server) zu ersetzen. Auch Cluster-Architekturen sind typische Anwendung statischer Redundanz.

Dynamische Redundanz

Die dynamische Redundanz wird häufig auch als Reserveredundanz bezeichnet, da diese Betriebsmittel im Normalbetrieb deaktiviert sind und erst im Fehlerfall aktiv eingebunden werden. Im Fall der dynamischen Redundanz werden Primär- und Ersatzkomponenten (auch Sekundär- oder Reservekomponenten genannt) unterschieden. Ein typisches Einsatzszenario ist beispielsweise das Vorhandensein eines Servers im Cold-Standby (Backup als Ausweichmöglichkeit), der bei Ausfall des Primärservers aktiviert wird.

1.1.1.3 Spezielle Ausprägungen mit besonderer Bedeutung

In der Praxis kommen sowohl Mischformen (z. B. Hybridredundanz) der oben genannten Redundanzverfahren als auch spezielle Ausprägungen vor. Wegen ihrer besonderen Bedeutung in hochverfügbaren IT-Systemen werden in den folgenden Abschnitten drei weitere Verfahren vorgestellt.

Pfadredundanz

Wenn nicht nur Einzelkomponenten (Knoten) in einem komplexen System hochverfügbar ausgelegt sind, sondern auch die Verbindungen (Kanten) unter diesen, so bietet diese Architekturvariante die Möglichkeit, Datenwege dynamisch umzuleiten. Diese Eigenschaft wird häufig als Pfadredundanz oder Mehrpfadigkeit *(engl. multi-path)* bezeichnet.

Geografische Redundanz

Zur Realisierung geografischer Redundanz werden redundante Mittel räumlich ausreichend weit, beispielsweise auf zwei Unternehmensstandorte, verteilt. Geographische Redundanz ist aufgrund der Erweiterung eines Systems um weitere Ressourcen, der strukturellen Redundanz zuzuordnen. Mit lokal ausreichend weit verteilten Ressourcen *(engl. geographically dispersed resources)* tritt man Gefahren entgegen, die mit extrem geringer Wahrscheinlichkeit an beiden Orten gleichzeitig auftreten. In der Praxis wird geographische Redundanz etwa durch ein Geo-Cluster oder ein Ausweichrechenzentrum realisiert. Der Ansatz eignet sich zur Vorsorge gegen Bedrohungen mit lokaler Auswirkung.

Diversität

Diversität bezeichnet eine Redundanzausprägung, die auf funktional ähnlichen, aber unterschiedlich implementierten Komponenten basiert. Diversitär redundante Komponenten tragen in gleicher Weise zur Erbringung der Nutzfunktionen in einer hochverfügbaren Architektur bei, basieren aber auf unterschiedlichen Varianten der Realisierung. Dies können sowohl verschiedene Implementierungen ein und desselben Algorithmus als auch Hardware-Komponenten unterschiedlicher Hersteller mit verschiedenen Betriebssystemen oder alternative Medien zur Datenübertragung (z. B. Glasfaser und Funk) sein. Diversität kann in gewisser Weise auch als Designredundanz betrachtet werden, da sie auf unterschiedlichen Systementwürfen beruht. Insgesamt kann durch den Einsatz diversitärer Komponenten die Anfälligkeit einer Architektur für Totalausfälle durch Komponenten- spezifische Fehler oder Bedrohungen reduziert werden. Hierbei ist aber zu beachten, dass die Anfälligkeit insgesamt steigt, da diversitäre Komponenten auch diversitären Bedrohungen ausgesetzt sind. Beispielhaft kann hier eine redundante Serverarchitektur genannt werden, die mit zwei diversitären Betriebssystemen A und B ausgerüstet ist. Tritt ein Virus für Betriebssystem A auf, so bleibt die Nutzfunktion durch Betriebssystem B insgesamt erhalten. Andererseits ist die Architektur aber zusätzlich für Virenangriffe auf Betriebssystem B anfällig [LiPS01].

1.1.1.4 Redundante Komponenten

Eine grundlegende Designempfehlung für hochverfügbare Architekturen ist die redundante Auslegung von Hardware- und Software-Komponenten. Da in einem IT-System bereits der Ausfall nur einer Komponente zum Systemausfall führen kann, wird bereits auf Komponentenebene für Redundanz gesorgt.

Hardware

Fehlertolerante Hardware ist auf Ebene der Bauteile in der Regel vollständig redundant ausgelegt. Die Abbildung 3 zeigt eine hochverfügbare IT-Komponente, die über mehrfach ausgelegte, redundante Bauteile (z. B. Netzteil, CPU, Festplatten oder Hauptspeicher) verfügt, welche ständig synchronisiert werden.

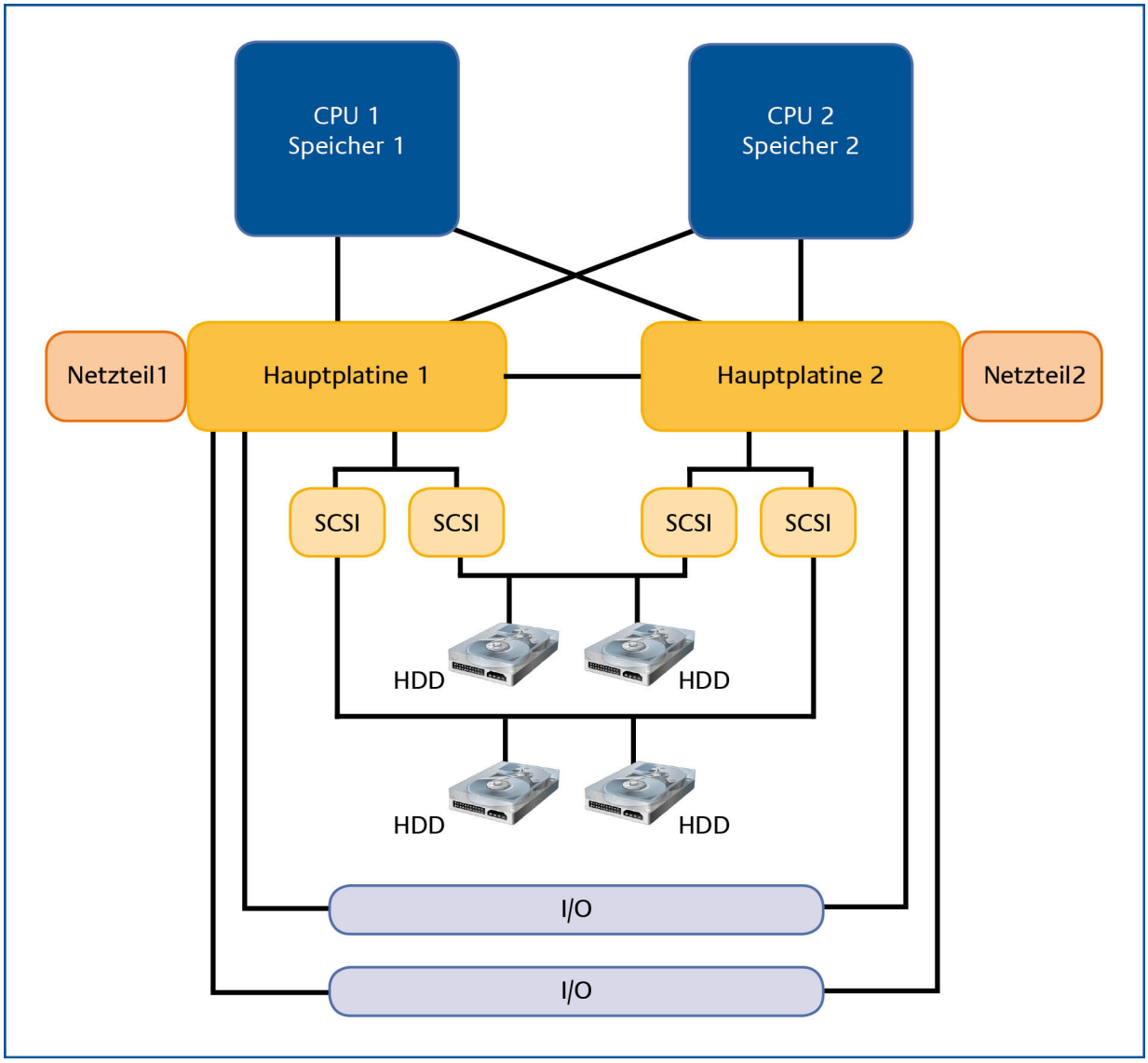


Abbildung 3: IT-Komponente mit redundanten Bauteilen

Bei einem Defekt kann das redundante Bauteil die Aufgabe übernehmen. Die Funktionsfähigkeit und Verfügbarkeit der Komponente bleibt erhalten. Benachbarte Systeme und laufende Anwendungen werden nicht beeinträchtigt.

Software

Eine auf Redundanz basierende Robustheit ist grundsätzlich auch für Software-Komponenten realisierbar. Fehlerhafte Software-Zustände werden durch Hardware-Fehler, falsche Benutzereingaben oder Programmierfehler hervorgerufen. In diesen Fällen führt die Aktivierung einer gleichartigen, redundanten Software-Komponente nicht zum gewünschten Erfolg, weil diese bei gleichen Umgebungsbedingungen das gleiche fehlerhafte Ergebnis liefern würde. Wird die gleiche Funktion durch andersartige Implementierungen (Software-Diversität) bereitgestellt, so kann ggf. trotz auftretenden Fehlers die Funktionsfähigkeit der Software (SW) aufrecht erhalten werden. Software-Diversität bedeutet, dass eine SW-Funktion über zwei oder mehr unterschiedliche Wege (z. B. Programmiersprachen, Entwurfsmethoden oder Projektteams) realisiert wird. Techniken der Software-Diversität werden aufgrund des hohen Aufwands selten eingesetzt. Das Mitführen redundanter Informationen durch die Software ist dagegen ein häufig eingesetztes

Verfahren, um im Fehlerfall auf die zunächst für die Funktion nicht benötigten redundanten Informationen zurückgreifen zu können oder um deren Korrektheit zu prüfen (z. B. zusätzliche Bitpositionen).

Werden abnorme Systemzustände, fehlerhafte Eingaben oder Integritätsverluste von Daten durch die Implementierung von Redundanz in Verbindung mit „intelligenter Software“ ausreichend abgefangen, so wird bei Software-Komponenten ein hohes Maß an Robustheit und Fehlertoleranz erreicht.

Hybrid

In der Praxis sind bei komplexen Architekturen Hybrid-Verfahren im Einsatz, bei denen die erforderliche Qualität aus einer Kombination von Hardware- und Softwaretechnologien realisiert wird. Beispielhaft können hier RAID-Festplatten genannt werden, die einen Ausfall der Hardware durch Softwaretechniken (Redundanz der Daten) ausgleichen. Ein weiteres Beispiel ist ein Festplatten-Betriebssystem, welches auf Fehler in einzelnen Sektoren automatisch mit einer Sperrung dieser Sektoren für den weiteren Zugriff reagiert.

1.1.1.5 Redundanz bei IT-Dienstleistungen und -Personal

Redundanz kann auf unterschiedlichen Ebenen hergestellt werden. Im letzten Abschnitt wurden Einsatz und Nutzen von Redundanz auf der Ebene der Komponenten erläutert. Auch auf den höheren Ebenen des HV-Schichtenmodells ist die Schaffung von Redundanz möglich.

IT-Dienstleistungen

Betrachtet man die Ebene der IT-Dienstleistungen als Schnittstelle zu den Geschäftsprozessen, so ergibt sich hier eine weitere effektive Möglichkeit, den Qualitätsstandard bei der Diensterbringung zu verbessern, indem nicht nur eine, sondern mehrere (möglichst diversitäre) IT-Services oder IT-Dienstleistungen für eine bestimmte Geschäfts-Funktion vorgesehen werden. Für den Kommunikationsanteil eines Geschäftsprozesses kann z. B. sowohl Telefonie als auch E-Mail vorgesehen werden, welche idealerweise nicht auf die gleichen technischen Systeme aufsetzen. Besonders im Rahmen der Notfallvorsorge stellt Redundanz auf Ebene der IT-Services oder der Dienstleistungen wirkungsvolle präventive Maßnahmen dar. Für die Aktivierung der redundanten Services wird im Rahmen der Notfallvorsorge und Notfallübungen das Know-How an den Personenkreis vermittelt, der beim Eintritt der besonderen Lage die Geschäftsprozesse auf der Grundlage der alternativen Service wahrnehmen soll. Die redundante Auslegung von Komponenten ist dabei eine weitere Möglichkeit, einen positiven Effekt auf die Einzelverfügbarkeit der IT-Services bzw. IT-Dienstleistungen zu erreichen.

Personal

Für die Zuständigkeit und Ausführung von Aktivitäten in einem Prozess werden Rollen definiert. Handelt es sich hier um Schlüsselrollen, beispielsweise der Systemadministrator oder der Service Desk beim Incident Management, so muss diese Rolle personell ausreichend ausgestattet sein. Durch eine entsprechende organisatorische Regelung muss geklärt sein, welche Person beim Ausfall einer anderen deren Rolle im Prozess übernehmen kann. Insbesondere vor dem Hintergrund einer hohen Quote bei Personalausfällen, z. B. durch Krankheit, Epidemien, Fluktuation oder Tod, muss dieser Einflussfaktor auf die Gesamtverfügbarkeit von IT-Services immer berücksichtigt und entsprechende Maßnahmen, wie „Schaffung von Redundanz beim Personal“, umgesetzt werden.

1.1.2 Redundanzarchitekturen

Werden redundante Teilarchitekturen zu redundanten Gesamtsystemen kombiniert, entsteht eine Redundanzhierarchie. Die einzelnen Redundanzebenen bauen hierbei aufeinander auf. Die Hierarchie sollte hinsichtlich ihrer HV-Eigenschaften lückenlos sein, d. h., sie sollte alle funktionalen Teilbereiche einbeziehen. Die Umsetzung der Redundanzhierarchie kann entweder vollredundant oder strangredundant erfolgen.

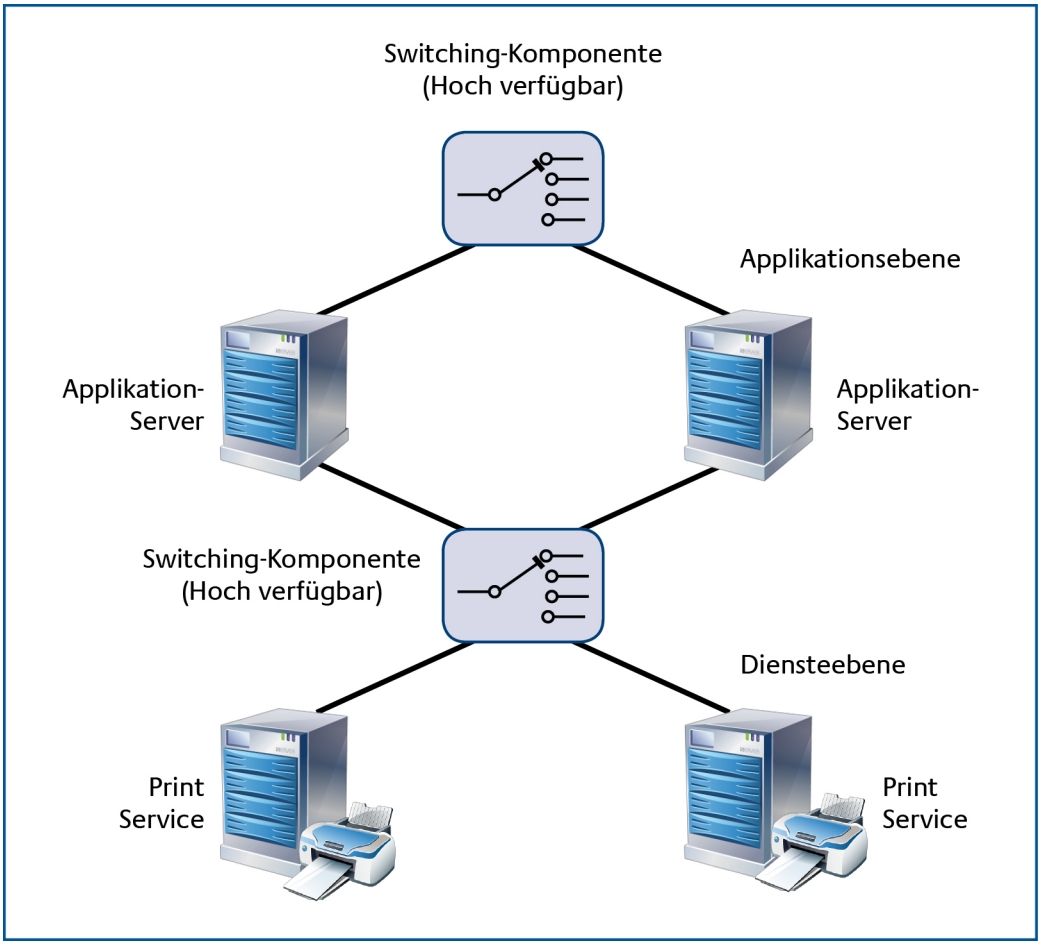


Abbildung 4: Beispielrealisierung der Vollredundanz

Unter einer *vollredundanten* Hierarchie (auch als horizontale Redundanz bezeichnet) ist eine Redundanzhierarchie zu verstehen, bei der jede Ebene für sich eigenständig volle Redundanz bietet. Diese Hierarchie bietet Fehlertoleranz für jeweils einen Ausfall pro Hierarchieebene. Mehrfachausfälle, welche die gleiche Ebene betreffen, können von dieser Architektur nicht aufgefangen werden. Eine entsprechende Architektur ist in der Abbildung 4 dargestellt. Neben der redundanten Auslegung der Server ist hier auch auf Redundanz bei den Umschaltkomponenten zu achten.

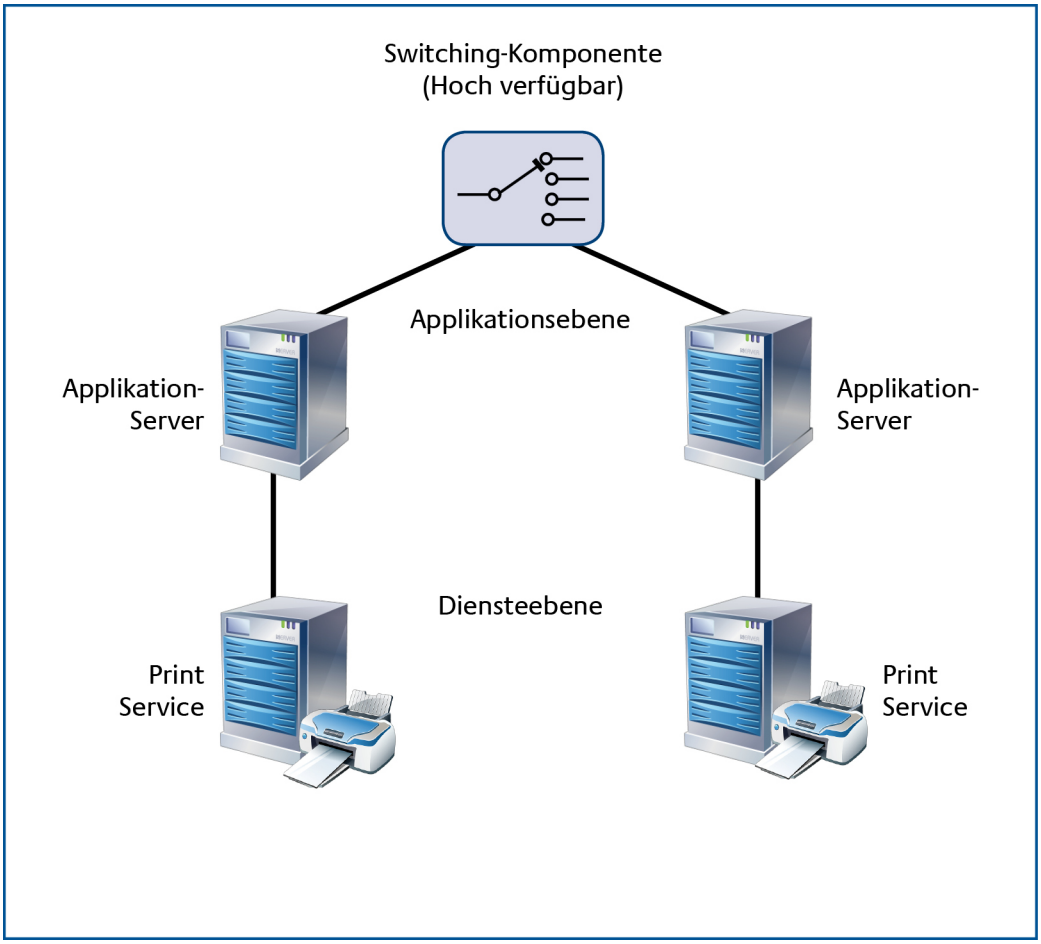


Abbildung 5: Beispielrealisierung für Strangredundanz

Eine *strangredundante Hierarchie* (auch als vertikale Redundanz bezeichnet) ist dadurch gekennzeichnet, dass die einzelnen redundanten Komponenten auf jeweils korrespondierende Komponenten der untergeordneten Ebene zugreifen. Diese Hierarchie bietet Fehlertoleranz nur innerhalb eines Stranges. Betrifft der Ausfall mehr als einen Strang, so ist keine Fehlertoleranz mehr gegeben. In der Abbildung 5 ist Strangredundanz exemplarisch realisiert.

Ein vollredundant ausgelegtes System (mit hochverfügbaren Komponenten zum Umleiten der Transaktionen) kann im Vergleich zu einem strangredundant ausgelegten System Ausfälle besser kompensieren. Theoretisch sollte daher auf möglichst jeder Ebene Vollredundanz angestrebt werden. Praktisch ist dies jedoch oftmals nur mit sehr großem Aufwand oder gar nicht realisierbar, etwa wenn zwei Teilkomponenten nur als Einheit angeboten werden (z. B. bei einer Festplatte und einem normalen Festplatten-Controller). In diesen Fällen muss besonders auf die sorgfältige Auswahl robuster Komponenten und auf die Wartung geachtet werden, um Ausfälle zu vermeiden.

1.2 Fehlertoleranz

Unter Fehlertoleranz versteht man allgemein die Eigenschaft eines Systems, die vorgesehene Funktion trotz eines die Funktion beeinträchtigenden Ereignisses weiterhin zu erfüllen. Bei technischen Systemen stellt die Fehlertoleranz eine wichtige Voraussetzung zur Erreichung einer hohen Verfügbarkeit dar. Die Systeme müssen auftretende Fehler in Hard- und Software sowie unerwartete Eingaben ohne Beeinträchtigung der Funktion verarbeiten.

In Bezug auf das Auftreten von Fehlern wird zwischen transienten und permanenten Fehlern [Jalo94] unterschieden.

- **Transiente Fehler** treten vorübergehend auf und sind häufig in der Hardware-Umgebung zu finden, da es hier zu den meisten unkalkulierbaren Einflüssen (z. B. Temperatur- und Energieschwankungen) kommen kann. Transiente Fehler sind generell schwer aufzuspüren, weil sie nur sporadisch auftreten und deshalb bei der Diagnose nicht immer entdeckt werden. Ihre Ursache ist schwer zu reproduzieren und daher kaum detektierbar.
- **Permanente Fehler** bleiben nach dem Eintreten dauerhaft bestehen. Für die Systemadministration sind es regelmäßig reproduzierbare Fehler, und in der Regel können diese Fehler sehr gut analysiert und dadurch auch sehr gut behoben werden.

Ein fehlertolerantes System gerät im Fehlerfall nicht in einen undefinierten Zustand (siehe Abbildung 6), sondern verbleibt in einem kontrollierbaren Zustand. Es verhält sich aufgrund implementierter Automatismen selbstheilend oder geht in einen definierten Notbetrieb über, der anschließend etwa durch einen manuellen Eingriff wieder in den Normalbetrieb überführt werden kann. Im optimalen Fall ist das Gesamtsystem in dem Maße fehlertolerant, dass der Anwender den aufgetretenen Fehler in einem Teilsystem nicht bemerkt (Transparenz). Die Reaktion des Systems erfolgt hierbei für den Anwender ohne Beeinträchtigung der Arbeitsprozesse.

Während der Begriff der Fehlertoleranz nur die vorweg beschriebene Fähigkeit eines technischen Systems beschreibt, muss bei der Konzeption auf Methoden zurückgegriffen werden, die eben diese Fehlertoleranz gewährleisten. Hier existieren verschiedene Ansätze, die nachfolgend beschrieben werden. Alle verfolgen das Ziel, Systeme nach einem aufgetretenen Fehlerereignis in einen, dem System bekannten und vom System erreichbaren Zustand zu versetzen, der die Konsistenz der geforderten Funktion sicherstellt.

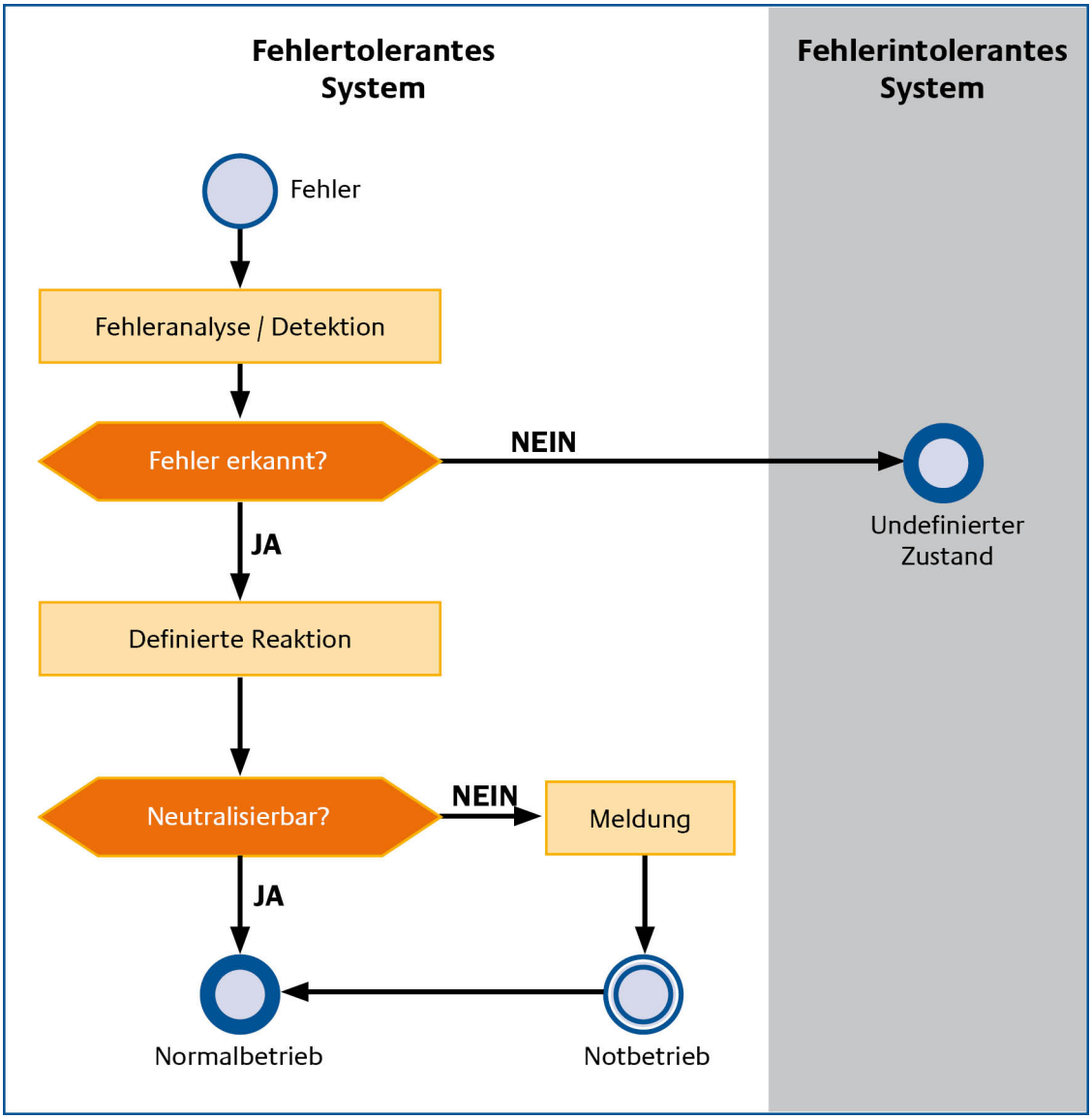


Abbildung 6: IT-System mit Mechanismen zur Erreichung von Fehlertoleranz

1.2.1 Fehlerbehebung

Voraussetzung für die Fehlerbehebung *(engl. error recovery)* ist die Fehlererkennung. Nachdem ein Fehler im System erkannt wurde, wird bei der Fehlerbehebung die gegenwärtig vom System ausgeführte Funktion abgebrochen und der fehlerhafte Zustand in einen fehlerfreien Zustand überführt. Dies geschieht entweder durch das Vorsetzen oder Zurücksetzen in einen fehlerfreien Zustand (z. B. in der Prozessortechnik). Ziel ist es dabei immer einen fehlerfreien Zustand zu finden, von dem aus der Prozess fehlerfrei weitergeführt werden kann.

Ein typisches Beispiel für rückwärts gerichtete Fehlerbehebung ist das Rollback-Verfahren eines Datenbank-Management-Systems (DBMS). Das DBMS stellt im laufenden Betrieb fest, dass eine bestimmte Transaktion zurückgesetzt werden muss. Mithilfe der Informationen im Transaktionsprotokoll wird ein Zustand aus der Vergangenheit gesucht, der nicht fehlerbehaftet ist. Die Transaktionen werden dann zurückgesetzt und das System bietet an der Benutzerschnittstelle einen fehlerfreien Zustand an.

Da die Ursache des Fehlers (z. B. eine fehlerhafte Komponente) bei Verfahren dieser Art nicht behoben wird, ist die Fehlerbehebung insbesondere bei transienten Fehlern angebracht, weil hier die Fehlerursache „flüchtig“ ist und die Überführung in einen fehlerfreien Zustand bereits eine effektive Fehlerbehandlung darstellen kann. Fehlerbehebung belässt also fehlerhafte Komponenten im System, überführt jedoch deren Fehlzustand in einen fehlerfreien Zustand. In dieser Eigenschaft unterscheidet es sich grundsätzlich von der Fehlerkompensierung.

1.2.2 Fehlerkompensierung

Voraussetzung für die Fehlerkompensierung *(engl. error compensation)* sind redundante Mittel, d. h. das System enthält genügend Redundanz, um den fehlerhaften Zustand zu überwinden (siehe auch Abschnitt 1.1) und die fehlerfreie Funktion des Systems wiederherzustellen. Bei der Fehlerkompensierung wird nur das Ergebnis fehlerhafter Komponenten beeinflusst. Fehlerhafte Komponenten oder fehlerhafte Zustände werden durch die Fehlerkompensierung in ihrer Substanz nicht verändert. Ein Verfahren zur Fehlerkompensierung ist beispielsweise die Fehlermaskierung. Sie lässt durch eine „Maske“ nur die Ergebnisse durch, die fehlerfrei sind. Verglichen mit der Fehlerbehebung benötigt die Fehlerkompensation in der Regel mehr Systemressourcen, sie hat jedoch üblicherweise einen geringeren Zeitbedarf bei der Fehlerbehandlung. Die Fehlerkompensierung ist sowohl bei transienten als auch bei permanenten Fehlern ein wirksames Verfahren.

1.2.3 Fehlertoleranz durch Mehrfachauslegung der Komponenten

Die redundante Auslegung von Systemen gewinnt besondere Bedeutung, wenn die Korrektheit der spezifizierten Funktionen als weitere Forderung auftritt. In diesen Fällen ist es erforderlich, neben einem (trivial zu erkennenden) Ausfall einer Komponente auch fehlerhafte Funktionsergebnisse zu detektieren und zu korrigieren. Beispielsweise im Flugzeugbau, der Raumfahrt oder dem Betrieb von Kraftwerken ist es zwingend erforderlich, dass die benötigte Funktion sowohl korrekt (in ihrem Ergebnis) als auch unterbrechungsfrei im erwarteten Zeitraum zur Verfügung steht. Hierzu sind über die redundante Auslegung hinausgehende Mechanismen notwendig, die eine Fehlererkennung und auch eine Fehlerkorrektur ermöglichen.

In der Abbildung 7 wird eine redundante Auslegung der Komponenten dargestellt. Fällt die Komponente A aus, so ist über die Komponente A' die Funktion des Systems weiterhin gegeben. In der Regel wird bei dieser Art Systemdesign die redundante Komponente nicht dazu verwendet ein zweites Ergebnis zu ermitteln. Die redundante Komponente übernimmt vielmehr die Funktion der primären Komponente nach deren Ausfall.

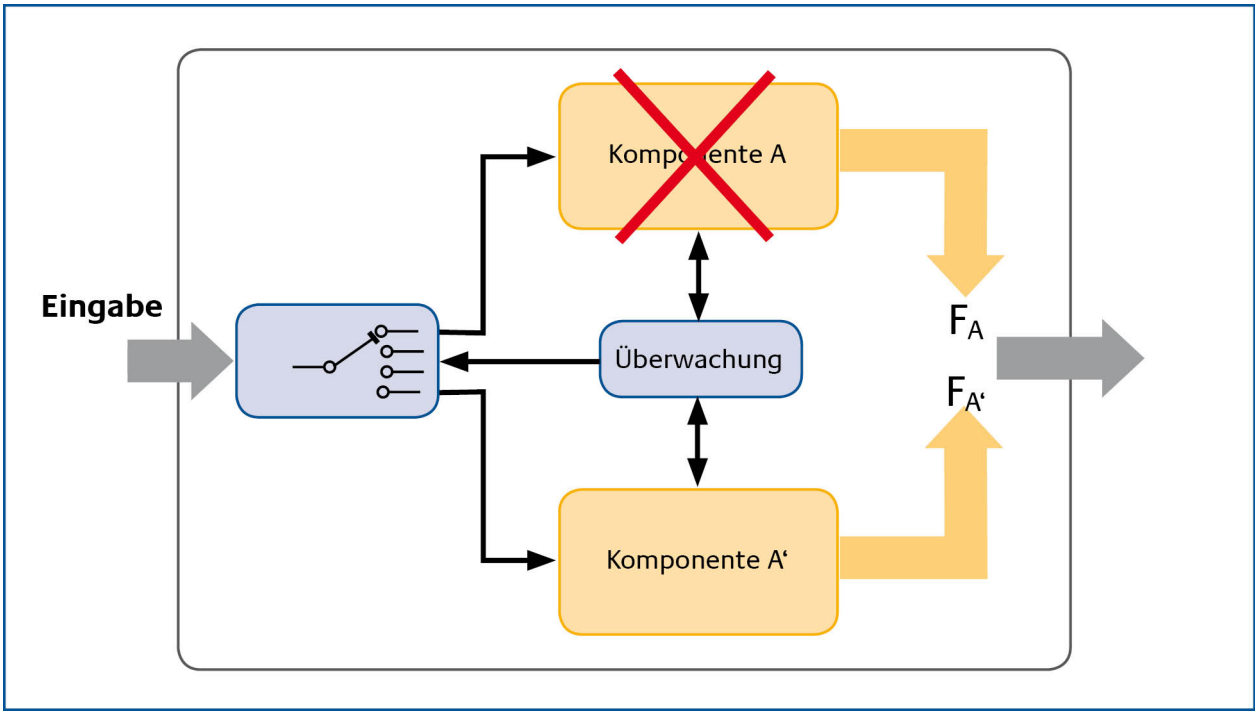


Abbildung 7: Einfache Redundanz der Komponenten

Erst der Parallelbetrieb bei gegebener Redundanz ermöglicht die gleichzeitige Bearbeitung eines Vorgangs und den Vergleich der Ergebnisse, um Fehler zu erkennen. Im Normalbetrieb können Fehler anhand zweier unterschiedlicher Ergebnisse erkannt werden. Im Störfall, d. h. beim Ausfall einer Komponente ist die Fehlererkennung (nur noch ein Ergebnis wird ermittelt) nicht mehr möglich.

Die Abbildung 8 zeigt eine zweifache Auslegung. Beim Ausfall einer Komponente ist die Fehlererkennung immer noch möglich. Die Voraussetzungen für eine Fehlerkorrektur sind jedoch bei diesem Systemdesign nach einem Ausfall nicht mehr gegeben.

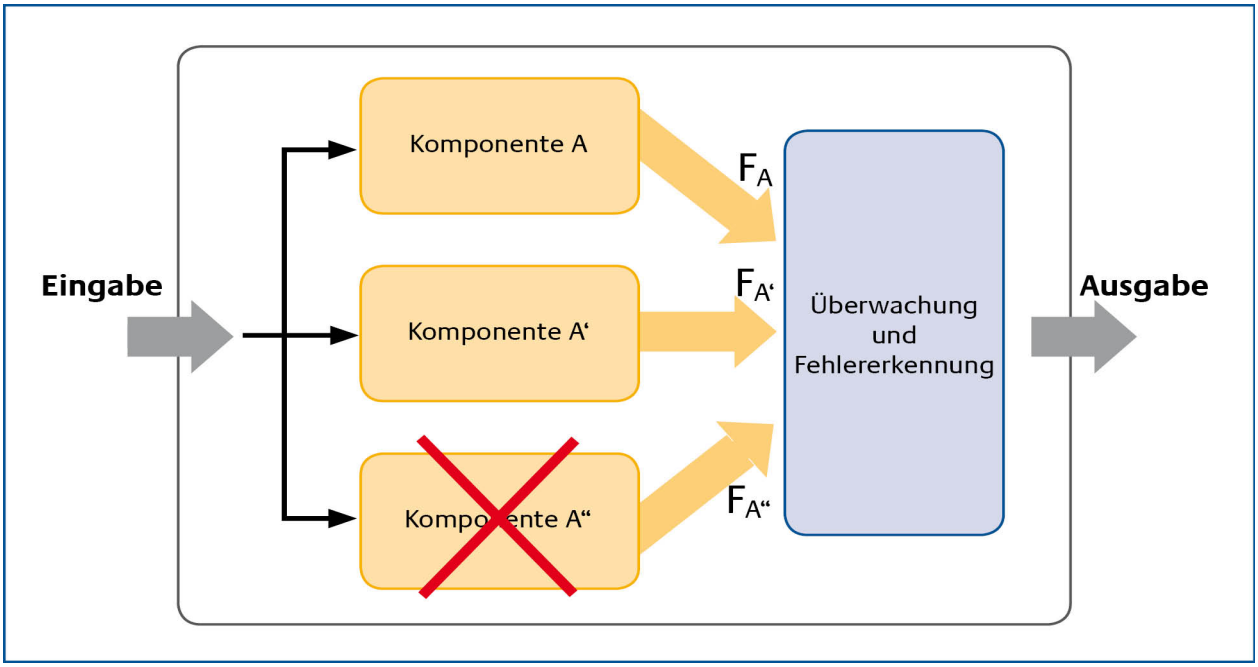


Abbildung 8: Zweifache Auslegung zur Fehlererkennung

Für eine Fehlerkorrektur im Sinne der funktionalen Korrektheit, auch beim Ausfall einer Komponente, ist daher eine dreifache Auslegung der Komponenten erforderlich. Die Abbildung 9 zeigt, dass selbst beim Ausfall einer Komponente immer noch drei Ergebnisse bereitstehen und eine Fehlerkorrektur mittels „Voting“ durchgeführt werden kann [Prad96].

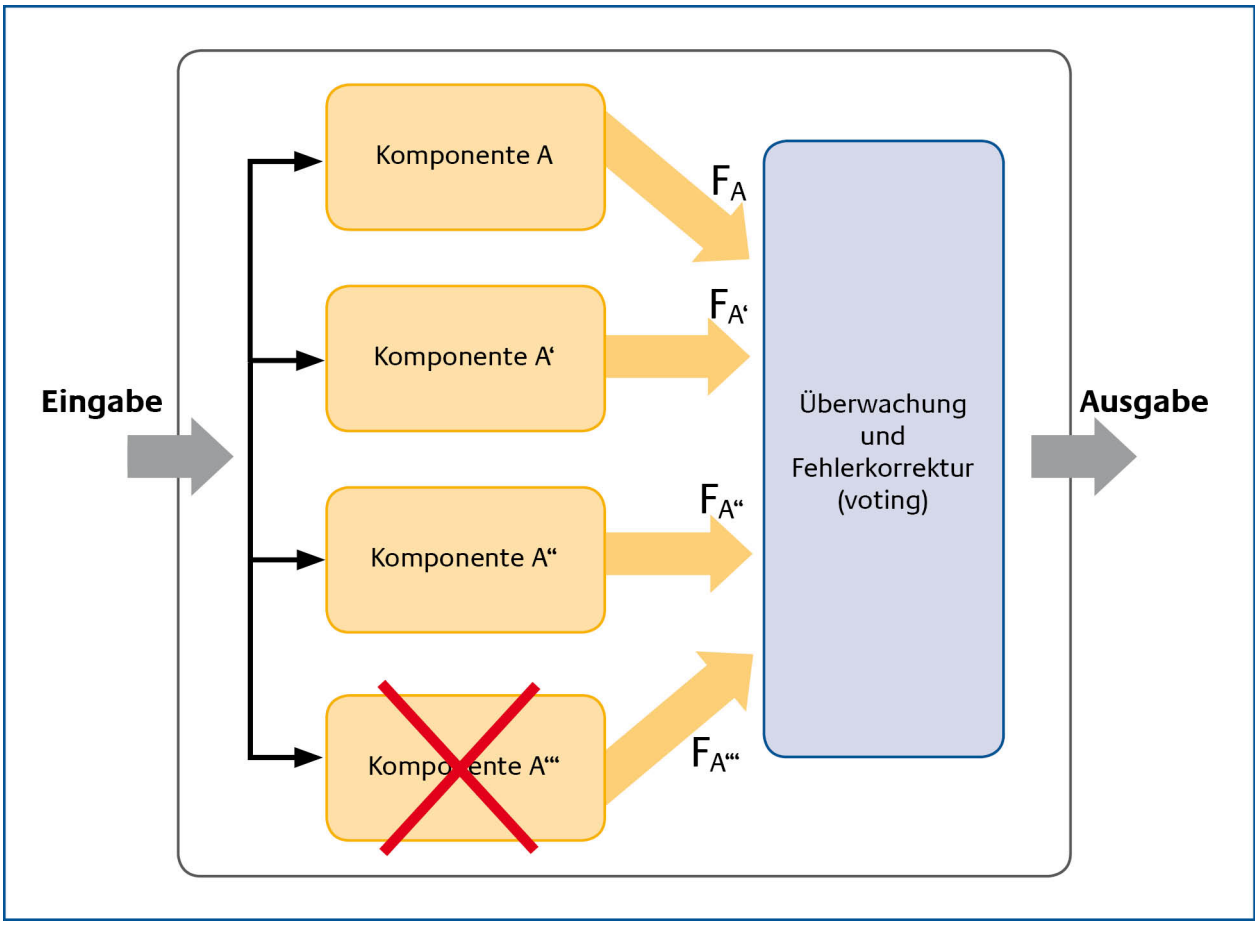


Abbildung 9: Dreifache Auslegung zur Fehlererkennung und -korrektur

Bei dieser Auslegung ist eine Fehlertoleranz ohne Leistungsverminderung *(engl. fail operational)* realisiert.

1.3 Robustheit

Robustheit beschreibt die Fähigkeit eines Systems oder einer Komponente auf Störeinflüsse von außen so zu reagieren, dass der funktionelle Fortgang im System hiervon nicht wesentlich betroffen ist. Die Umsetzung dieses Prinzips in einer HV-Umgebung bedeutet, deren Betrieb durch präventive Maßnahmen gegen Störeinflüsse von außen zu schützen. Robuste Komponenten schützen das System nicht vor Fehlern, deren Ursachen innerhalb des Systems zu finden sind.

Komponenten erhalten ihre Robustheit etwa weil hochwertige Materialien verwendet wurden oder spezielle Vorrichtungen zum Schutz gegen Umwelteinflüsse (z. B. Staub, Wasser oder Überhitzung) integriert wurden. Die Komponenten werden entsprechend des Einsatzbereichs und der Einsatzbedingungen konzipiert. So sind beispielsweise die Komponenten eines Industrie-PCs, aufgrund der speziellen Anforderung an Stoßfestigkeit und Temperaturbeständigkeit, anders auszulegen als die eines herkömmlichen Arbeitsplatz-PCs. Auch die besondere Belastung viel fordert eine besondere Sorgfalt bei der Auswahl der Komponenten.

Besondere Beanspruchung einzelner Komponenten in der Betriebsphase kann, auch bei Verwendung hochwertiger Materialien, zu Verschleißerscheinungen führen. Durch regelmäßige Wartung der Komponenten und einem rechtzeitigen Austausch stark beanspruchter Komponenten kann die Robustheit aufrechterhalten werden (siehe Beitrag „Software“ in Band II des HV-Kompendiums).

Neben robusten Hardware-Komponenten ist auch für Software-Komponenten in HV-Umgebungen ein hohes Maß an Robustheit erstrebenswert. Störeinflüsse von außen sind bei der Software beispielsweise andere Software-Komponenten, die einen fehlerbehafteten Input (quantitativ und qualitativ) liefern. Ebenso ist der Mensch, der mit der Software arbeitet, eine potenzielle Fehlerquelle. Robuste Software-Module müssen auf entsprechende Einflüsse von außen, die nicht den Normalfall darstellen, angemessen reagieren können. Die Robustheit der Software muss in geeigneten Testverfahren nachgewiesen werden.

Die Robustheit von Komponenten gegen Umgebungs- und Störeinflüsse kann in Testverfahren erprobt werden. CPUs und Speichereinheiten können beispielsweise in der Entwicklungs- und Fertigungsphase in sogenannten Stresstests auf ihre Belastbarkeit überprüft werden, indem sie äußeren Einflüssen ausgesetzt werden, die weit über die Anforderungen im Normalbetrieb hinausgehen [Schm06].

Der Begriff „Robustheit“ ist typischerweise technisch geprägt, trotzdem lässt sich das Prinzip in gewisser Weise auch auf das Personal übertragen. Das IT-Personal muss als Teil der gesamten HV-Umgebung auch bei auftretenden Störungen (z. B. Lärm oder Gefahrensituation) die seiner Rolle entsprechende Funktion ausüben können. In der Personalentwicklung wird diese Eigenschaft des Personals nicht mit „Robustheit", sondern mit „Belastbarkeit“ bezeichnet. Das Personalmanagement verwendet als Mittel zur Auswahl besonders belastbarer Bewerber ebenfalls Stresstests, in denen sich zeigen soll, wie ein Bewerber in belastenden Situationen reagiert. Der Umgang mit störenden Einflüssen von außen kann weitgehend trainiert werden. Regelmäßig sollten daher im Rahmen von Notfallübungen die Mitarbeiter mit derartigen Situationen konfrontiert werden, um Routine und Handlungssicherheit im Umgang mit derartigen Situationen zu erreichen.

Konzeptionell besteht meist die Möglichkeit, auf robuste Einzelkomponenten zugunsten eines robusten Teilsystems zu verzichten. Die Robustheit bezieht sich dann nicht mehr auf einzelne Komponenten, sondern auf Teilsysteme oder Module. In diesem Fall wird der Ausfall von Einzelkomponenten bewusst in Kauf genommen und durch redundante Komponenten aufgefangen.

1.4 Separation

Das Verfügbarkeitsprinzip *Separation* beschreibt im hier betrachteten Kontext die bedarfsgerechte Trennung von IT-Services oder Ressourcen. IT-Services mit unterschiedlichen Anforderungsprofilen werden beispielsweise auf getrennte Ressourcen (IT-Systeme) aufgesetzt, damit potenzielle Fehlerquellen und Gefährdungen besser eingegrenzt und isoliert werden. Die Qualitätsmerkmale der verwendeten Systeme können so optimal auf das Anforderungsprofil zugeschnitten werden. Ein Prozess der durchgängig mit „offenen“ Daten operiert benötigt keine aufwendigen Verfahren, welche die Vertraulichkeit dieser Daten sichern. Liegen dagegen hohe Anforderungen an die Verfügbarkeit vor, so müssen geeignete (IT-)Ressourcen zur Abwicklung des IT-Servicesvorhanden sein, damit die vereinbarten Zeiten eingehalten werden. Die Realisierung einer HV-Architektur ist oftmals mit einem hohen Aufwand verbunden und sollte daher auch dediziert für kritische Geschäftsprozesse genutzt werden. Ein weiterer, die Qualität begünstigender

Effekt, ist die Reduktion der Komplexität bei den separierten Einzelsystemen, was wiederum zu mehr Transparenz und besserer Skalierbarkeit führen kann.

Praktische Beispiele für eine Umsetzung der Separation in Bezug auf IT-Ressourcen sind die modulare Software-Entwicklung *(Separation of Concerns)* [HüLo95] und im Bereich der Speicherverwaltung das Aufteilen der Daten nach Speicherhierarchien.

Die Separation erfordert im Vorfeld der Umsetzung eine detaillierte Planung und Bedarfserhebung auf Grundlage des erforderlichen Schutzbedarfs im Hinblick auf die Verfügbarkeit. Es ist detailliert zu ermitteln, welche Geschäftsprozesse und IT-Dienstleistungen als kritisch hinsichtlich ihrer Aufgabenerfüllung anzusehen sind. Diese werden im Rahmen der Anforderungsanalyse in der Phase S identifiziert und analysiert. Die Analyse schließt die Ermittlung ein, welche Komponenten der HV-Umgebung zur Erbringung der Dienstleistungen erforderlich sind.

Je nach ermitteltem Bedarf und zur Verfügung stehenden Ressourcen können unterschiedliche Separationsverfahren zum Einsatz kommen.

Geographische Separation

Die Auslagerung kritischer IT-Services und der zur Erbringung erforderlichen Ressourcen in spezielle Betriebsumgebungen wird als Geographische Separation bezeichnet. Dies können beispielsweise Rechenzentren spezialisierter Dienstleister oder besonders geschützte Gebäude (Gebäudeteile) sein.

Separation von Infrastruktur und Netzen

Dieser Punkt beinhaltet die Abspaltung und physikalische Trennung der Netz- und Infrastrukturkomponenten der HV-Architektur. Hierbei werden den kritischen Systemen dedizierte Netzwerke und Infrastrukturkomponenten zugeordnet. Der Betrieb sollte physikalisch und logisch von der bestehenden Infrastruktur getrennt unter Einsatz besonderer Sicherheitstechnik (z. B. USV, Netzersatzanlage, Klimatisierung etc.) erfolgen.

Services mit dedizierter HV-Infrastruktur

Eine vollständig eigenständige Ausgestaltung von IT-Services mit besonderer Bedeutung sowie der zugehörigen Infrastruktur wird als dedizierte HV-Infrastruktur bezeichnet. Neben eigenständigen Netzen und Versorgungseinrichtungen werden dedizierte, entsprechend der HV-Anforderungen ausgestaltete IT-Systeme in besonderen HV-Umgebungen eingesetzt.

Separation von Teams

Bei der Entwicklung technischer Systeme werden Entwicklungs- und Testteam zum Zweck einer optimierten Güte des entwickelten Produktes (Systems) voneinander getrennt. Test- und Entwicklungsaufgaben werden personell getrennt, um Annahmen, Gedankenmodelle und versteckte Voraussetzungen, die die Programmierer getroffen oder verwendet haben, kritisch hinterfragen zu können. Die auf das Testen spezialisierten Personen oder Teams bekommen Routine und lernen die relevanten Testmethoden zunehmend besser kennen und beherrschen. Das macht das Testen wirksamer und erhöht zudem die Produktivität. Die Separation von Teams wird auch in anderen Bereichen der IT-Organisation angewandt, um eine Überdeckung der Aufgaben und eine gegenseitige Beeinflussung bewusst zu unterbinden.

Bei besonders hohen Anforderungen an die Qualität der Software werden zusätzlich mehrere voneinander getrennte Entwicklungsteams mit der Herstellung von Software-Modulen beauftragt,

welche die gleichen Funktionen abdecken[1]. Man nennt dieses Verfahren auch N-Versionen-Programmierung. Die N Versionen des Moduls laufen parallel und bei nicht identischen Resultaten findet eine Mehrheitsabstimmung statt [GrRo01].

Separation der Daten

Die Separation von Daten zum Schutz der Vertraulichkeit ist ein probates Mittel. Eine Separation der Daten kann auch zum Zweck ihrer besseren Verfügbarkeit erfolgen. Daten aus dem Berichtswesen oder der Geschäftsanalyse werden vom Datenpool des operativen Geschäfts separiert. Dadurch wird sichergestellt, dass die Anfragen der operativen Geschäftsprozesse nicht von den, oftmals Ressourcen belastenden, Analyseprozessen beeinflusst werden.

1.5 Virtualisierung

Virtualisierung bezeichnet die Auflösung der statischen Beziehung zwischen den Ressourcen, die eine IT-Umgebung zur Verfügung stellt und den Geschäftsprozessen, die diese Ressourcen nutzen. Damit ist es möglich, Anwendungen auf nahezu beliebigen Umgebungen und unter flexibler Nutzung der zur Verfügung stehenden Ressourcen laufen zu lassen. Das Prinzip der Virtualisierung gewinnt weiterhin an Bedeutung und kann in mehrfacher Hinsicht zur Unterstützung bei der Entwicklung hochverfügbarer Architekturen eingesetzt werden.

Die Virtualisierung gestattet eine effektive Nutzung und Auslastung der bestehenden Ressourcen. Dies spart Kosten und kann zu einer Verringerung der Komplexität der IT-Infrastruktur beitragen. In herkömmlichen IT-Architekturen werden für nahezu jede Fachanwendung eine spezielle Serverkomponente sowie Applikationssoftware benötigt. Dies führt insbesondere im Umfeld hochverfügbarer Systeme zu erhöhten Kosten und Aufwänden, da jedes dieser Systeme redundant ausgelegt werden muss. Die für die Ausfallsicherheit zusätzlichen Ressourcen werden, bei einer Standby-Architektur im Normalbetrieb nicht genutzt. Das Prinzip der Virtualisierung erlaubt eine dynamische und gleichmäßige Verteilung der Lasten auf die verfügbaren Betriebsmittel.

Virtualisierung ist auf mehreren Ebenen der IT-Funktionalität möglich. Neben der Virtualisierung beim physikalischen Speicher (Zoning) können darüber hinaus Hardware, Betriebssysteme, Software oder vollständige Serverumgebungen virtualisiert werden [BITK10]. Der Einsatz der zur Verfügung stehenden Virtualisierungsprinzipien erlaubt vor allem ein flexibles und dynamisches Management sowohl der Ressourcen als auch der IT-Services. Dies bietet insbesondere im HV-Umfeld umfangreiche Möglichkeiten zur Fehlerreaktion und Fehlerbehandlung. Nach dem Ausfall einer Komponente kann der hierauf laufende Anwendung ein anderer virtueller Server zugeteilt werden. Diese dynamische Anpassung kann darüber hinaus automatisiert und ohne eine für den Anwender spürbare Zeitverzögerung ablaufen.

1.5.1 Hardware-Virtualisierung

Bei der Hardware-Virtualisierung werden beliebige Hardware-Komponenten nachgebildet und der Software zur Verfügung gestellt. Aus Sicht des Betriebssystems sowie der Applikation verhält sich die virtuelle Hardware wie eine reale Hardware-Komponente, ist aber durch eine zusätzliche „Pufferschicht“ von der realen Hardware getrennt.

1 Es gibt andere Verfahren, wie beispielsweise das „Extreme Programming“, die bewusst die Rollentrennung aufheben, damit ein optimaler Wissenstransfer gewährleistet wird.

1.5.1.1 Virtualisierung des Speichers

Bei der Speicher-Virtualisierung nutzen Betriebssystem und Applikation einen virtuellen Speicherbereich, der unabhängig von der physikalischen Ausprägung verwendet werden kann. Die Nutzung des virtuellen Speicherbereichs ist insbesondere unabhängig von der Art, der tatsächlichen Größe und Standort des physikalischen Speichersystems.

1.5.1.2 Virtuelle Maschinen

Virtuelle Maschinen als die umfassende Ausprägung einer Hardware-Virtualisierung bilden vollständige Serversysteme und Hardwareplattformen nach. So können auf einer bestehenden Serverplattform mehrere Server-Anwendungen realisiert werden, die sich die bestehenden Ressourcen teilen.

1.5.2 Software-Virtualisierung

Mittels Software-Virtualisierung lassen sich Betriebssysteme oder Anwendungen simulieren.

1.5.2.1 Betriebssystem-Virtualisierung

Die Betriebssystem-Virtualisierung als Sonderfall der Software-Virtualisierung stellt Anwendungen eine vollständige, virtuelle Laufzeitumgebung bereit. Diese befindet sich in der Regel innerhalb eines geschlossenen Containers mit definierten Grenzen und Ressourcen. Da für die Ausführung der Anwendung kein Betriebssystem auf einer speziellen Hardware gestartet wird, können Container und Anwendung auf beliebige Systeme portiert werden. So werden die vorhandenen Ressourcen optimal genutzt, im Fehlerfall können sowohl der Software-Container als auch die darauf laufende Applikation auf andere Systeme ausgelagert werden.

1.5.2.2 Anwendungsvirtualisierung

Die Anwendungsvirtualisierung erlaubt das lokale Ausführen von Anwendungen, ohne dass diese zuvor installiert werden müssen. Den Anwendungen steht eine virtuelle Betriebssystemumgebung zur Verfügung, die alle benötigten Konfigurationseinträge, Betriebssystemkomponenten und Ressourcen enthält. Die Anwendung befindet sich zur Laufzeit in einem virtuellen Betriebssystem-Container, der gegenüber dem Basis-Betriebssystem sowie anderen Anwendungen abgeschlossen ist. Fehler der Anwendung wirken sich zunächst nur innerhalb dieses Containers aus, andere Anwendungen oder das Basisbetriebssystem sind hiervon nicht betroffen. Ebenso kann bei einem Fehler der zugrunde liegenden Hardware- oder Software-Plattform eine Auslagerung des Containers auf eine andere Ressource erfolgen.

1.5.3 Virtuelle Testumgebung

Soft- und Hardware-Virtualisierung werden im Rahmen von Testaktivitäten genutzt, um die verschiedenen Einsatzumgebungen der Systeme zu simulieren. Ein spezieller Testzustand kann dadurch problemlos „eingefroren“, und zu einem späteren Zeitpunkt für die Fehleranalyse verwendet werden.

1.5.4 Virtuelle Lernumgebung

Das IT-Personal arbeitet üblicherweise in einem sehr dynamischen Arbeitsumfeld, welches kontinuierliches Weiterbilden erfordert. Dies trifft in besonderem Maß auf das Personal zu, welches

an HV-Lösungen arbeitet. Virtuelle Lernumgebungen bieten effiziente Weiterbildungsmöglichkeiten und ermöglichen ein Coaching bei konkreten Problemlösungen direkt am Arbeitsplatz.

1.6 Transparenz

Der Begriff *Transparenz* hat in der IT unterschiedliche Bedeutungen. In Bezug auf IT-Ressourcen bedeutet Transparenz, dass man auf sie zugreifen kann, ohne sie und den Ort ihrer Implementierung zu kennen. Dieser Aspekt wurde bereits im Zusammenhang mit den Prinzipien Fehlertoleranz und Virtualisierung betrachtet.

Bezieht man Transparenz auf die IT-Organisation, so wird dadurch ein weiteres Design-Prinzip für HV-Umgebungen bestimmt. In diesem Kontext wird unter Transparenz die Durchschaubarkeit oder Deutlichkeit (*lat. transparens*) von organisatorischen Zusammenhängen verstanden. In diesem Kontext kommt der Betrachtung und Analyse von Organisationspotentialen(s. HV-Kompendium V1.6, Band B, Kapitel B1 „Meta-Ebene“) eine besondere Bedeutung zu. In den dort betrachteten Dimensionen, welche Organisationspotentiale prägen, wird Transparenz an verschiedenen Stellen gefordert:

- in der Kulturdimension als Ziel-, Entscheidungs- und Ergebnis-Transparenz,
- in dem Prinzip der lernenden Organisation,
- in der Datenbasis als Dokumentation der Wissensbasis und Dokumentation der Prozesse.

Damit sind prägende Merkmale identifiziert, welche das Ogranisationspotential im Sinne einer Entwicklungsstufe beschreiben. In Organisationen auf niedrigen Potentialstufen ist Transparenz regelmäßig gering ausgeprägt, weil z.B. keine dokumentierten Beschreibungen mit dem Umgang von Ereignissen vorliegen. Dabei schafft das Prinzip Transparenz wesentliche Voraussetzungen für:

- Serviceability (Nutzbarkeit, Handhabbarkeit),
- Maintainability (Wartbarkeit),
- Flexibilität,
- Erweiterbarkeit,
- Wiederverwendbarkeit und
- Reproduzierbarkeit.

In dieser technischen Sichtweise wird gemeinhin das Verständnis für die Notwendigkeit einer hinreichenden Transparenz gezeigt, während für den organisatorischen Bereich dieses Verständnis oft fehlt.

Realisiert wird Transparenz wesentlich durch allgemein verständliche Darstellung in der Dokumentation und in der Orientierung von Prozessen an allgemein akzeptierten Standards. So bringt beispielsweise die Orientierung an Standrads der IT-Governance und generischen Prozessmodellen die notwendige Transparenz über alle Phasen des Lebenszyklus hinweg und ermöglicht, dass strategische Aspekte aus Sicht der Geschäftsprozesse sowie organisatorische und technische Aspekte von allen Beteiligten gleichermaßen erkannt und verstanden werden. Dies stellt die notwendige Voraussetzung für den Know-How-Transfer und die kontrollierte Reproduzierbarkeit von Prozessen dar. Durch transparente Standards und Prozessmodelle wird ein Rahmen festgelegt, der die Professionalität des IT-Sevice Management fördert und optimiert.

Ein Garant für die Transparenz der IT sind generische Prozessmodelle für die Serviceerbringung(wie z.B. CobiT oder ITIL), die Bewertungskriterien und Steuerungsinstrumente für die Leistungsfähigkeit und Qualität der Services einführen. An dieser Stelle sei auf die Transparenz als klares Verständnis und eindeutige Erwartungshaltung zwischen Servicenehmer und Servicegeber hingewiesen, die klare Vereinbarungen und Definitionen im Rahmen von Service Level Agreements (SLA) erfordern. SLAs bringen Klarheit über Kosten, Leistungen, Nutzen und Wirtschaftlichkeit der IT und stecken die zu erbringenden und zu erwartenden Leistungen für beide Parteien ab.

1.7 Skalierbarkeit

Der Begriff Skalierbarkeit beschreibt im Allgemeinen die Anpassungsfähigkeit eines Objektes, eines Systems oder eines Verfahrens. Die Skalierbarkeit von HV-Umgebungen stellt ein grundsätzliches Designprinzip bei der Umsetzung hochverfügbarer Architekturen dar. Gut skalierbare IT-Infrastrukturen sind in der Lage, flexibel, effizient, schnell und zuverlässig auf wachsende Lasten und Anforderungen zu reagieren. Bei wachsendem Ressourcenbedarf oder bei zusätzlichen erweiterten Anforderungen muss es möglich sein, weitere Betriebsmittel oder Ressourcen hinzuzufügen, ohne die Zuverlässigkeit der bestehenden Infrastruktur zu beeinträchtigen. Darüber hinaus sollte der Aufwand zur Erweiterung annähernd linear zu den gewachsenen Anforderungen stehen. Einen sinnvollen Ansatz zur Realisierung einer effizienten Skalierbarkeit stellt die Modularisierung dar, auf die bereits im Zusammenhang mit dem Prinzip Separation hingewiesen wurde. Hierbei werden Funktionsmodule verwendet, die bei Veränderung der Anforderungen ohne prinzipielle Veränderung der Architektur eine flexible Anpassung der HV-Infrastruktur erlauben. Weitere technische Verfahren wie verteilte Dateisysteme, Multiprozessorsysteme und Cluster-Architekturen tragen zu einer hohen Skalierbarkeit bei und werden in HV-Umgebungen eingesetzt.

Der Faktor Skalierbarkeit ist essentiell, um auch bei steigenden Benutzerzahlen gleich bleibende Antwortzeiten liefern zu können und einen unterbrechungsfreien Betrieb zu gewährleisten.

Skalierbarkeitsanforderungen sind nicht nur im technischen, sondern auch im organisatorischen Bereich zu berücksichtigen. Skalierbarkeit auf der Ebene der Organisation kann bedeuten, den Behördenaufbau derart zu strukturieren, dass neue Aufgabenfelder, Personalzuwachs und Aufgabenumverteilung im laufenden Betrieb gut kompensiert werden.

1.8 Automatismen

Der Begriff *Automatismen* stammt aus der Verhaltensbiologie und bezeichnet Aktionen von Organismen, die unbeeinflusst vom eigenen Willen automatisiert ablaufen. Beispiele hierfür sind vor allem die so genannten „Vitalfunktionen“ wie Atmung oder Herzschlag, die für die Erhaltung der lebenswichtigen Funktionen zuständig sind.

Überträgt man diese Definition auf IT-Ressourcen im HV-Umfeld, so stellen Automatismen Vorgänge innerhalb der HV-Architektur dar, die ohne menschliches Zutun ablaufen und die Aufrechterhaltung der Verfügbarkeit gewährleisten. Diese automatischen Prozesse sind in der Regel systemimmanent für HV-Komponenten und sorgen innerhalb der Komponente für eine automatisierte, unverzügliche Reaktion auf Fehlersituationen. Da diese Reaktionen innerhalb der HV-Komponente stattfinden, sind sie zunächst für die umliegenden Systeme und die Anwender transparent und nicht spürbar.

Automatismen sorgen innerhalb eines HV-Verbundes durch Regelmechanismen für den Ausgleich der Wirkung von spontanen Ereignissen. In der Regel ist die Wirkung der Automatismen auf einen kurzfristigen und überwiegend auch provisorischen Ausgleich beschränkt, die Behebung von Schäden sowie die Beseitigung bedrohlicher äußerlicher Ereignisse ist Aufgabe des Incident Managements. Die Orientierung des IT-Service-Managements an generischen Prozessmodellen schafft Automatismen innerhalb und zwischen den einzelnen Prozessen, mit denen abhängige Prozesse z.B. bei Incidents automatisch an angestoßen werden (Prozess-Trigger). Voraussetzung für das Funktionieren der Automatismen innerhalb von Prozessketten ist die Orientierung an den Prozessmodellen (ITILoder CobiT) für die Erhöhung der Professionalität in der eigenen Organisation.

Der Wirkansatz von Automatismen lässt sich in präventiver oder reaktiver Weise realisieren. Der präventive Ansatz versucht, auf absehbar bedrohliche Ereignisse zu reagieren und einen Schaden zu verhindern, bevor dieser auftritt. Ein Beispiel für einen präventiven Ansatz im HV-Umfeld stellt die Lastverteilung in einem Cluster über einen Load-Balancing-Mechanismus oder eine Load-Balancing-Komponente dar. Diese führt eine gleichmäßige Verteilung der auftretenden Last auf alle Komponenten durch und verhindert so einen möglichen Ausfall eines einzelnen Systems durch Überlast. Voraussetzung hierfür ist, dass die an den Komponenten anfallende Last nur einen Teil der System-Ressourcen (z. B. CPU) belegt, um Reserven bei einem Ausfall zur Verfügung zu haben.

Ein anderer Ansatz des präventiven Vorgehens findet sich im sog. Monitor, also der Überwachung der IT-Systeme. Ein gut implementiertes Monitoring ermöglicht häufig, drohende Ausfälle schon im Voraus zu erkennen (z. B. durch die Überwachung von Hardwarekomponenten) und eventuell automatisch auf andere Komponenten umzuschalten, ehe es zum Ausfall kommt.

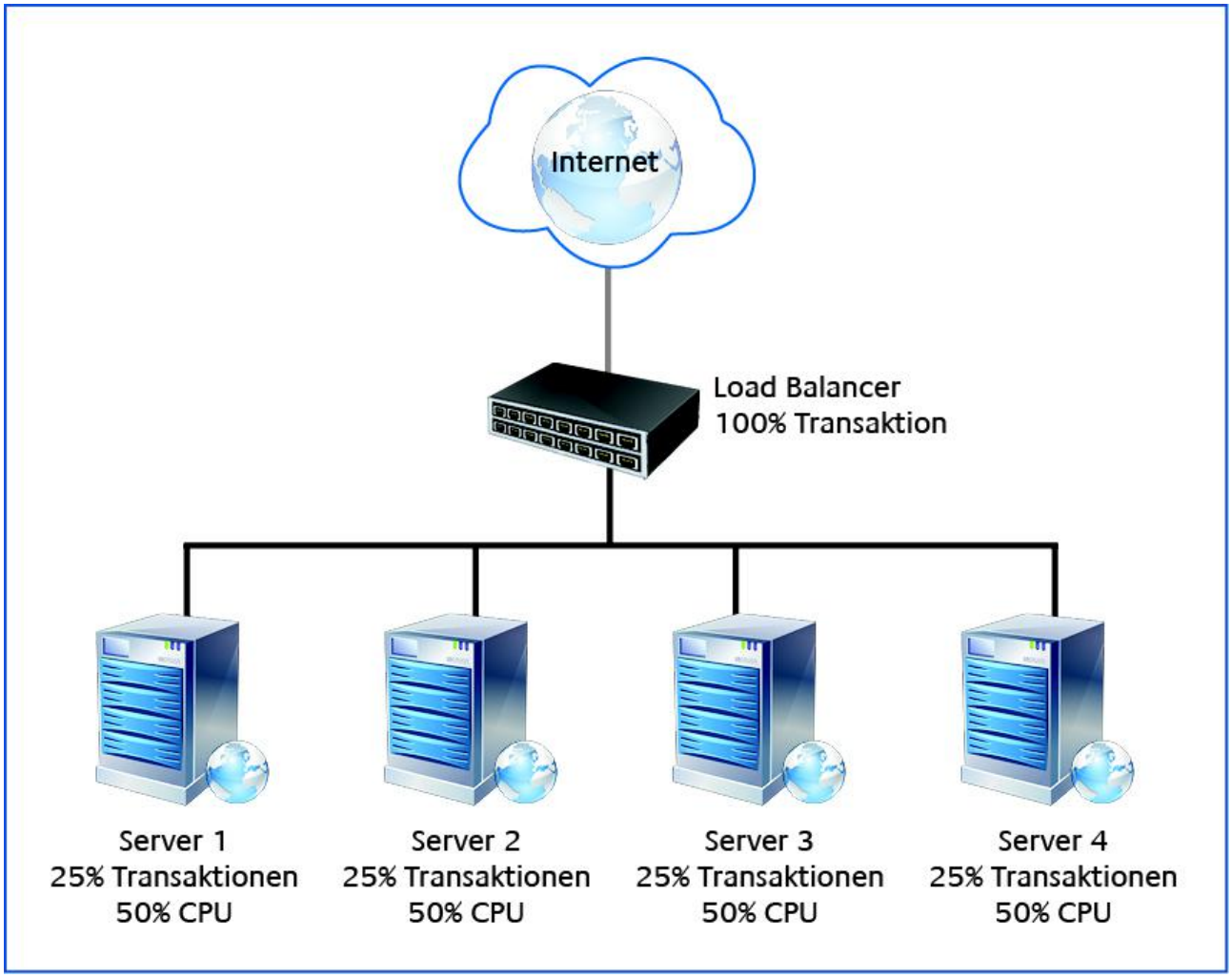


Abbildung 10: Präventive Verteilung der Last durch Load-Balancing-System

Reaktive Automatismen reagieren im Unterschied dazu in der Regel auf bereits aufgetretene Schadensereignisse innerhalb einer HV-Komponente. Ein Beispiel für einen reaktiven Automatismus sind Fail-Over-Mechanismen, die bei einem Ausfall oder Defekt einer Komponente die Nutzfunktionen an die verbleibenden Systeme übertragen und so die Verfügbarkeit der Dienste im Schadenfall sicherstellen.

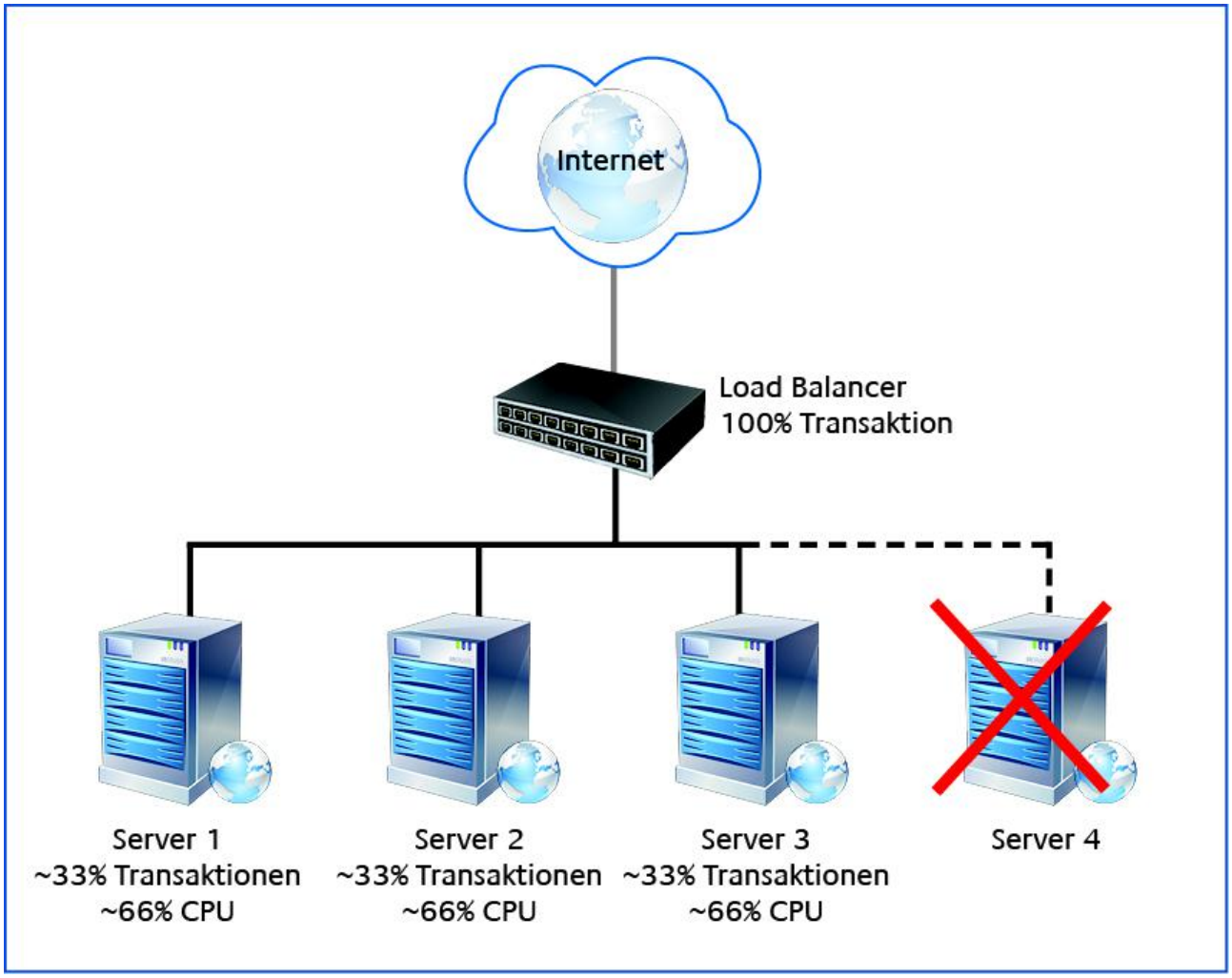


Abbildung 11: Reaktive Verteilung der Last durch Load-Balancing-System

1.9 Priorisierung

Priorisierung regelt die Vorrangigkeit von Prozessen, Aktivitäten, Dienstleistungen, IT-Services oder Ressourcen. In einem Priorisierungsverfahren (z. B. Rangfolgeverfahren oder ABC-Technik) erfolgt eine Einordnung nach gewissen Kriterien wie Bedeutung, Kritikalität oder Wichtigkeit.

Vor dem Design einer hochverfügbaren IT-Architektur muss eine Priorisierung der IT-Dienstleistungen und -Ressourcen im Hinblick auf die Bedeutung der Geschäftsprozesse durchgeführt werden (s. HV-Kompendium Band G, Kapitel G4: „Phase S“). Die daraus resultierenden Informationen bilden die Grundlage für ein anforderungsgerechtes Design der HV-Architekturen und können in vielfältiger Weise in die Ausgestaltung der (Teil-)Architekturen und -Prozesse einfließen.

Im Zusammenhang mit HV-Lösungen wird Priorisierung genutzt, um bei begrenzten Betriebsmitteln den Prozessen mit höherer Priorität mehr Ressourcen zur Verfügung zu stellen. Stehen nicht mehr ausreichend Ressourcen für alle Prozesse zur Verfügung, wird zumindest der Betrieb der Services mit besonderer Bedeutung aufrechterhalten.

Eine Priorisierung ist auch für den Fall eines Systemausfalls und dem anschließenden Wiederanlauf eine wichtige Voraussetzung. Hierzu liefert die in der Phase S erhobenen Größe „Dauer der Verzichtbarkeit“ den notwendigen Input für die Festlegung der Prioritäten. Nach einem Systemausfall kann gemäß der zuvor festgelegten Prioritäten ein geordneter Wiederanlauf erfolgen. Bei einem Teilausfall können die Prioritäten als Grundlage für die Festlegung eines eingeschränkten Betriebes genutzt werden.

1.9.1 Dienstpriorisierung

In der Praxis ist die Dienstpriorisierung im IP-Verkehr ein bekanntes Beispiel. Im Header-Bereich des IP-Packets werden die sog. Precedence Bits gesetzt (z. B. bei der Priorisierung von VoIP) und ausgewertet [Taka91]. Die Priorisierung gestattet es, den kritischen Diensten einen Vorzug zuzuweisen. Dienstpriorisierung ist eine wirkungsvolle Maßnahme, die in und zwischen unterschiedlichen Netzen nutzbar ist (Voraussetzung sind abgestimmte Protokolle und Parameterwerte). Weitere Informationen zu Dienstpriorisierung sind im Beitrag „IT-Dienste und Dienstleistungen“ des HV-Kompendiums enthalten.

1.10 Autonomie

Autonomie bezeichnet in der Regel eine Selbstständigkeit im Hinblick auf Selbstverwaltung und Entscheidungsfreiheit. Autonomie im Umfeld der Hochverfügbarkeit kann als Eigenständigkeit bei der Erfüllung der Verfügbarkeitsanforderungen verstanden werden (HV-Autonomie). Während die Zielvorgaben für das Design und die Ausgestaltung der Architektur durch die geforderte Verfügbarkeitsklasse von außen vorgegeben werden, erfüllen die Teilkomponenten diese Aufgabe autonom und transparent für den Anwender.

An hochverfügbare Systeme (VK 4- 5) werden so hohe Anforderungen im Hinblick auf Ausfallzeiten, Reaktionszeiten und Wiederherstellungszeiten etc. gestellt, dass diese manuell (d. h. durch Wartungs- oder Betriebspersonal) kaum zu erfüllen sind. Darüber hinaus steigt der Aufwand für die zu implementierenden Redundanzmechanismen mit steigender Verfügbarkeit unverhältnismäßig.

Zurzeit ist das autonome Systemverhalten noch Gegenstand der Forschung, mit praktischen Ergebnissen ist im Lauf der nächsten Jahre zu rechnen. Das Ziel ist die Schaffung einer weitestgehenden Systemautonomie auf verschiedensten Ebenen der Architektur, bis hin zu einer autonomen HV-Architektur.

Die „HV-Autonomie“ sieht folgende Eigenschaften für Teilsysteme vor:

Adaptivität

Ein System ist adaptiv bezüglich einer Menge von Eingaben, wenn es in der Lage ist, für diese Eingaben seine Zielstellung zu erfüllen.

Selbstmanagement / Selbstverwaltung

Ein System ist selbstverwaltend (selbstmanagend), wenn es adaptiv ist, ohne von außen kontrolliert zu werden.

Selbstkonfiguration

Ein System ist selbstkonfigurierend, wenn es seine Konfiguration ohne externen Kontrolleingriff anpasst, um seine Adaptivität sicherzustellen.

Selbstorganisation

Ein selbstorganisierendes System ist ein selbstverwaltendes System, welches zusätzlich strukturadaptiv ist und eine dezentrale Kontrolle hat.

Selbststabilisierung

Ein System ist selbststabilisierend, wenn es im fehlerfreien Fall,

1. ausgehend von einem beliebigen Zustand einen legalen Zustand in beschränkter Zeit erreicht und
2. ausgehend von einem legalen Zustand seinen Zustand in der Menge der legalen Zustände hält.

Das System führt eine eigenständige, automatisierte Behebung von Fehlerzuständen durch.

1.11 Prinzipien und das SOA-Referenzmodell

Die zuvor unter dem Aspekt der Verfügbarkeit dargestellten Prinzipien spiegeln sich, wenn auch mit anderer Schwerpunktsetzung, in anderen Design-Methoden wieder.

Im Architekturkonzept des Bundes [SAGA08] werden zu diesem HV-Kompendium kongruente Architektur- und Integrationsprinzipien definiert, welche die Interoperabilität, Wiederverwendbarkeit und Skalierbarkeit der verschiedenen Dienste und Komponenten im Fokus haben. Das Potenzial dieser Prinzipien, IT-Infrastrukturen evolutionär und flexibel an sich schnell ändernde Anforderungen anzupassen und weiterzuentwickeln, ist auch ein Potenzial, welches hochverfügbaren Architekturen entgegenkommt.

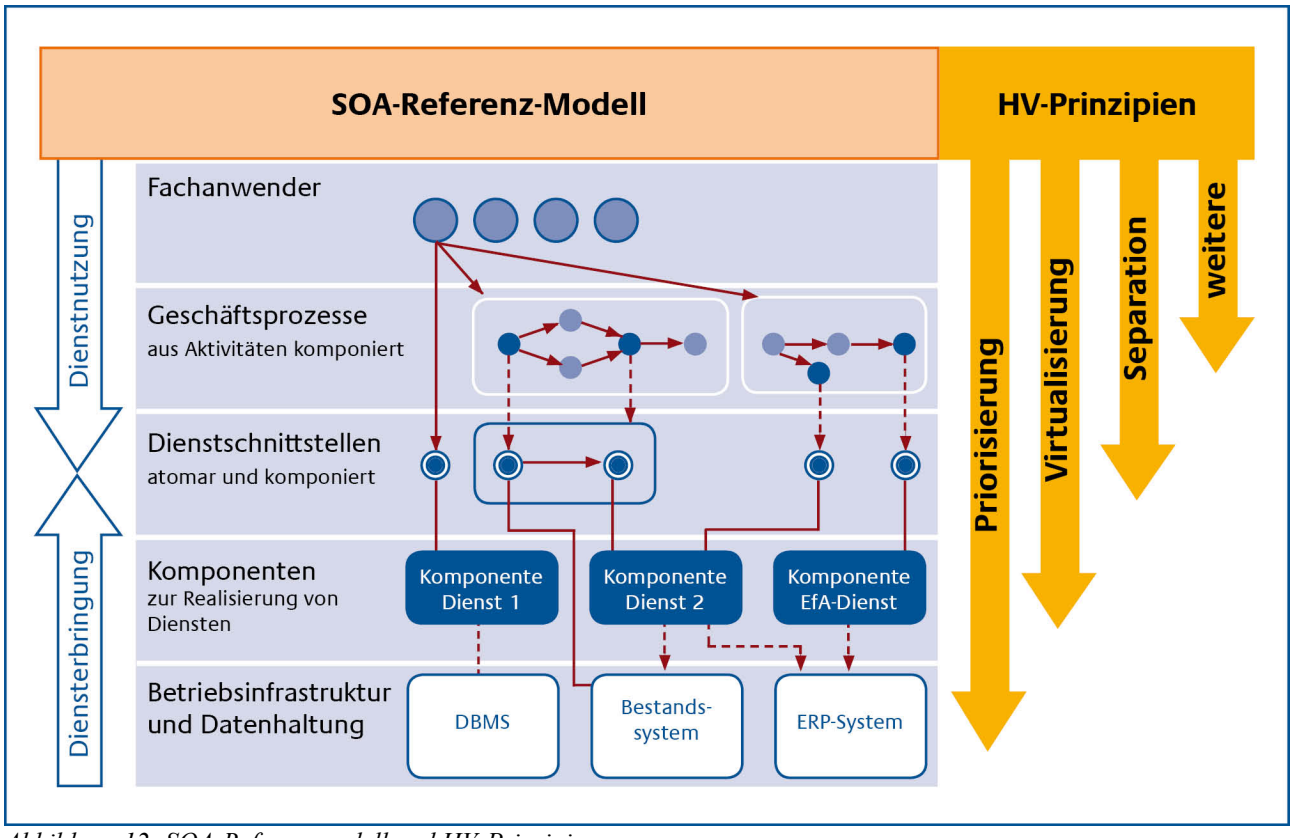


Abbildung 12: SOA-Referenzmodell und HV-Prinzipien

Die Abbildung 12 veranschaulicht die Integration der HV-Prinzipien in das SOA-Referenzmodell aus dem IT-Architekturkonzept der Bundesverwaltung.Nach dem Referenzmodell soll jeder Dienst in sich abgeschlossen und damit eigenständig nutzbar sein. In der Abbildung 12 korrespondiert jeder Teilprozess mit genau einer Schnittstelle. Neben dieser Forderung nach Separation wird für jeden Service auch die Virtualisierung empfohlen. Danach besitzt jeder Dienst eine wohl definierte Schnittstelle und allein deren Kenntnis reicht für die Nutzung. Kenntnisse über die Implementierung oder gar über Details sind dafür nicht erforderlich.

Die bisherige Diskussion zeigt exemplarisch die Parallelitäten in den der SOA und dem HV-Kompendium zugrunde liegenden Design-Prinzipien. Während SOA die Eigenschaften von Diensten eher aus technischer Sicht beschreibt, steht für die Orchestrierung der Dienste nach dem Architekturkonzept des Bundes die Geschäftsprozess-Logik im Vordergrund. Im HV-Kompendium werden diese beiden Sichten zu einer ganzheitlichen Sichtweise zusammengeführt.

1.12 Zusammenfassung

Die in diesem Beitrag vorgestellten Prinzipien sind im IT-Bereich bereits hinlänglich bekannt, wie auch die Integration in das Architekturkonzept des Bundes zeigt [SAGA08]. Sie bekommen jedoch aufgrund ihrer Anwendung in hochverfügbaren Systemen eine neue Bedeutung. Die Umsetzung der Prinzipien durch technische und organisatorische Maßnahmen dient überwiegend der Vermeidung oder Reduzierung von Ausfallzeiten. Eine bedarfsgerechte Umsetzung der Prinzipien im Sinne einer hohen Verfügbarkeit verändert nicht die Standardfunktionalität des Systems, sondern sichert diese auf dem vereinbarten Servicelevel.

Für die einzelnen Prinzipien werden nachfolgend nochmals die Kernaussagen zusammengefasst:

- **Redundanz**
  Im Umfeld der HV-Technologie ist die Redundanz ein wesentliches Prinzip. Redundante Mittel in Kombination mit leistungsfähigen Mechanismen führen zu Fehlertoleranz und Ausfallsicherheit.
- **Fehlertoleranz**
  Fehlertolerante Systeme gewährleisten die Diensterbringung auch im Fall von Störungen und erlauben die Fortführung der Prozesse bei Ausfällen von Teilsystemen.
- **Robustheit**
  Robustheit beschreibt die Fähigkeit auf äußere Störeinflüsse so zu reagieren, dass diese nicht zu Ausfällen führen. Robustheit wird meist durch Verwendung hochwertiger Materialien und robustem Design der Hardware- und Software-Architekturen erreicht.
- **Separation**
  Separation ermöglicht eine Trennung von Services und Ressourcen insbesondere von denen mit geringerer Bedeutung. Dadurch werden spezielle und anforderungsgerechte „HV-Umgebungen“ geschaffen.
- **Virtualisierung**
  Virtualisierung führt eine logische Trennung von Funktion und Struktur durch. Die Bindungen zwischen Hardware und Software, bzw. Applikation und Umgebung werden aufgelöst. Durch Virtualisierung erreicht man mehr Unabhängigkeit und Flexibilität bei der Umsetzung hochverfügbarer Prozesse.

- **Transparenz**
  Transparenz der IT-Ressourcen sorgt gegenüber dem Anwender für ein „Verbergen“ von partiellen Fehlern (Fehlertransparenz) und der Komplexität (siehe auch Virtualisierung). Transparenz in der IT-Organisation schafft Durchschaubarkeit oder Deutlichkeit von organisatorischen Zusammenhängen.
- **Automatismen**
  Automatismen sorgen innerhalb eines HV-Systems durch automatisch ablaufende Regelmechanismen für den Ausgleich von potenziellen Störungen.
- **Priorisierung**
  Priorisierung bedeutet eine Bevorzugung der unternehmenskritischen Prozesse bzw. deren unterstützenden Ressourcen und stellt ein Mittel zur bedarfsgerechten Planung und Umsetzung der benötigten HV-Umgebung dar.
- **Skalierbarkeit**
  Skalierbarkeit ist eine wichtige Voraussetzung bei der Planung und Umsetzung hochverfügbarer IT-Infrastrukturen, da sie die effiziente Implementierung auch bei steigenden Anforderungen sicherstellt.
- **Autonomie**
  Autonome Teilsysteme führen ihre Aufgabe selbstständig und transparent aus und stellen einen wesentlichen Schritt hin zur Realisierung fehlertoleranter Systeme dar.

Neben der Anwendung der Verfügbarkeitsprinzipien auf technischer Ebene ist auch das reibungslose Ineinandergreifen der organisatorischen Prozesse eine wichtige Voraussetzung für eine hohe Verfügbarkeit. Die Maßnahmen und Architekturvorschläge in den weiteren Beiträgen des HV-Kompendiums setzen diese Prinzipien auf technischer und organisatorischer Ebene um. Nur das Zusammenwirken der Prinzipien auf allen Ebenen einer HV-Architektur ermöglicht in der Praxis hohe Verfügbarkeit der IT-Dienstleistungen.

Anhang: Verzeichnisse

Abkürzungsverzeichnis

Ein komplettes Verzeichnis hierzu findet sich in Band AH, Kapitel 5

Glossar

Ein komplettes Verzeichnis hierzu findet sich in Band AH, Kapitel 6

Literaturverzeichnis

Ein komplettes Verzeichnis hierzu findet sich in Band AH, Kapitel 7