

Abb. 3.20 Homogene und heterogene Redundanz

sich die dynamische Redundanz im Software-Bereich nicht negativ auf die Laufzeit aus.

Insgesamt ergeben sich aus den obigen Überlegungen vier verschiedene Möglichkeiten der Mehrfachauslegung, die in Abb. 3.20 grafisch gegenübergestellt sind.

In der Praxis kommen Systeme zum Einsatz, die eine Mischung aus dynamischer und statischer Redundanz forcieren. In Abb. 3.21 ist als Beispiel eines solchen *Hybridsystems* die Redundanztopologie des Space Shuttles skizziert. Die sicherheitskritischen Systeme werden durch insgesamt fünf baugleiche Rechner gesteuert. Vier davon sind mit identischer Software ausgestattet und bilden zusammen einen vierfach ausgelegten homogenen Redundanz-Cluster. Alle Rechner des Clusters laufen parallel und werden durch einen Voter kontrolliert (statische Redundanz). Fällt einer der Rechner aus, werden die verbleibenden als *2-aus-3-System* weiter betrieben. Zusätzlich steht ein fünfter Rechner zur Verfügung, der mit separat entwickelter Software ausgestattet ist und zur Übernahme der Shuttle-Steuerung manuell aktiviert werden kann. Insgesamt entsteht hierdurch eine zweite Ebene der Absicherung, die auf dem Prinzip der dynamischen heterogenen Redundanz beruht.

3.4.2 Selbstüberwachende Systeme

Zur Erhöhung der Fehlertoleranz führen viele sicherheitskritische Systeme verschiedene Arten der Selbstdiagnose durch. Die durch einen entdeckten Fehler ausgelösten Systemreaktionen unterscheiden sich untereinander erheblich und hängen nicht

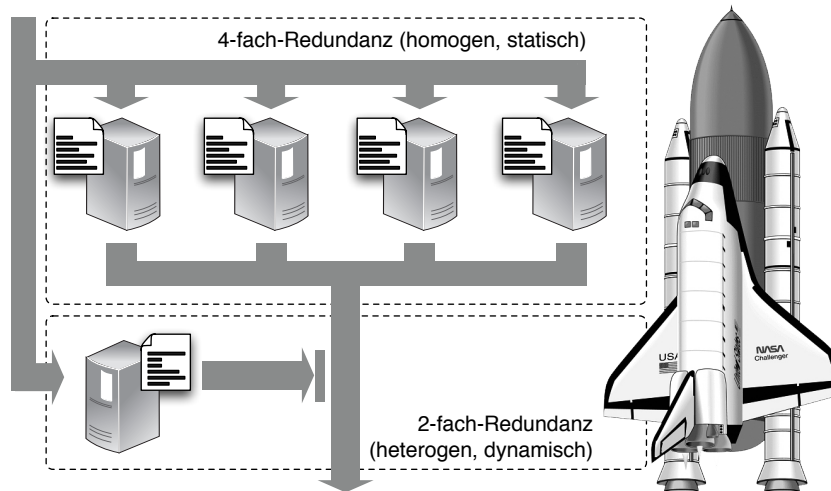


Abb. 3.21 Software-Redundanz am Beispiel des Space Shuttles

zuletzt von der Schwere der aufgetretenen Fehlfunktion ab. Die folgenden Szenarien stellen typische Reaktionsstrategien dar:

■ Fail-Safe-Reaktion

Auf einen Fehlerzustand wird durch den Wechsel in einen sicheren Zustand reagiert (*fail safe state*). Einige Systeme erreichen ihren sichersten Zustand, indem sie sich schlicht deaktivieren (*Selbstabschaltung*), andere setzen im Fehlerfall spezielle Sicherheitsmechanismen in Gang. Klassische Beispiele aus diesem Bereich sind die selbstaktivierende Notbremse eines Personenaufzugs oder die Sitzplatzverriegelung einer Achterbahn.

An dieser Stelle werfen wir einen erneuten Blick auf die Bordelektronik moderner Kraftfahrzeuge. Für jedes an den CAN-Bus angeschlossene Steuergerät sieht die Spezifikation einen zweistufig ausgelegten Fail-Safe-Mechanismus vor, der den Umgang mit defekten Busknoten auf systematische Weise regelt. Wie in Abb. 3.22 gezeigt, befindet sich jedes Steuergerät zu jedem Zeitpunkt in genau einem von drei Zuständen:

– Zustand 1: Error active

Dieser Zustand wird im regulären Betrieb eingenommen. Ein Steuergerät darf sowohl lesend als auch schreibend auf den CAN-Bus zugreifen und quittiert jeden detektierten Übertragungsfehler durch das Senden einer speziellen Fehlerbotschaft (*error frame*).

– Zustand 2: Error passive

Häufen sich die Übertragungsfehler, so geht der Busknoten in einen passiven Betriebsmodus über. In diesem Zustand darf das Steuergerät immer noch lesend und schreibend auf den Bus zugreifen, erkannte Übertragungsfehler wer-

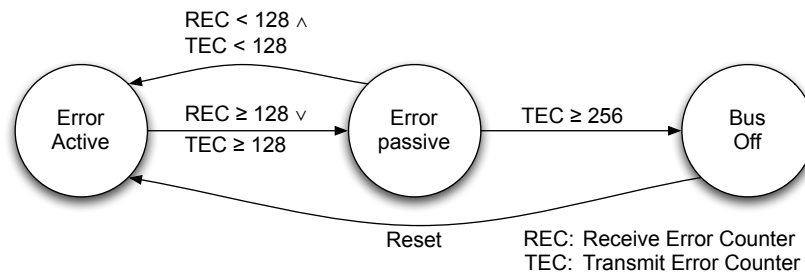


Abb. 3.22 Fail-safe-Verhalten am Beispiel der Kommunikation CAN-Bus-basierter Kraftfahrzeugsteuergeräte

den jedoch nicht mehr durch das Senden einer Fehlerbotschaft quittiert. Sollte der betreffende Busknoten selbst für die gemessenen Fehler verantwortlich sein, so wird auf diese Weise verhindert, dass der Kommunikationskanal dauerhaft mit einer Flut von Fehlermitteilungen überschwemmt wird.

– **Zustand 3: Bus off**

Häufen sich die Übertragungsfehler weiter, so trennt sich das Steuergerät selbstständig vom Bus ab und stellt jegliche Kommunikation ein (*fail-safe state*). Im laufenden Betrieb ist die Einnahme dieses Zustands irreversibel und das Gerät kann ausschließlich durch einen vollständigen Reset wieder aktiviert werden.

Intern werden die Zustandsübergänge über die zwei Zähler REC (*receive error counter*) und TEC (*transmit error counter*) gesteuert, die jedes Steuergerät selbstständig verwaltet. Beide dienen der Buchführung über die Fehlerhistorie und werden während des Betriebs ständig aktualisiert. Der REC wird erhöht, falls eine Nachricht fehlerhaft empfangen wurde und ansonsten erniedrigt. In analoger Weise wird der TEC erhöht, falls eine Nachricht fehlerhaft gesendet wurde und ansonsten erniedrigt.

■ **Selbstreparatur**

Systeme dieser Bauart versuchen, die im Rahmen einer Selbstdiagnose ermittelten Inkonsistenzen selbstständig zu beseitigen. Die Möglichkeiten der Selbstreparatur sind vielfältig. Neben dem Löschen inkonsistenter Datenstrukturen oder dem nachträglichen Einsetzen von Standardwerten sind sämtliche Spielarten komplexer Algorithmen denkbar. Diese reichen bis hin zur semi-autonomen Reparatur von Datenbeständen unter Beteiligung des Benutzers. Das Prinzip der Software-Selbstreparatur wird heute bis hinauf zur Betriebssystemebene angewendet. Beispiele sind die automatische Wiederherstellung der Dateisystemintegrität nach einem Systemabsturz oder die automatische Neuinstallation versehentlich gelöschter Systemkomponenten.