Formale Grundlagen der Fehlertoleranz in verteilten Systemen

Felix Gärtner

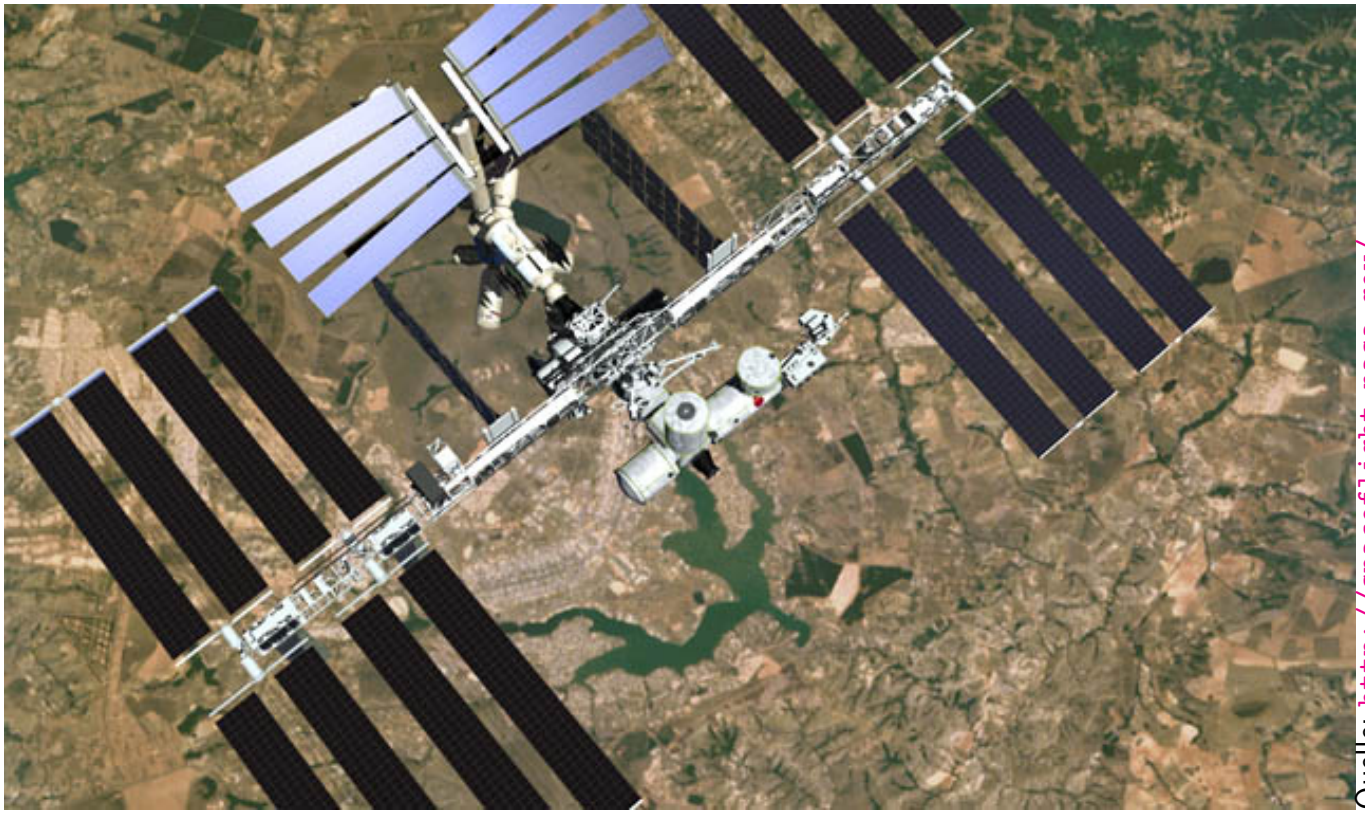
Technische Universität Darmstadt

Kritische Infrastrukturen

Quelle: http://www.cs.virginia.edu/~survive/

- Computersysteme schaffen Flexibilität, bergen aber auch Risiken.
  - Wir alle machen Fehler — auch Computer!
  - Fehler tolerieren durch Risikomanagement und Technologie

Grenzen des Risikomanagements

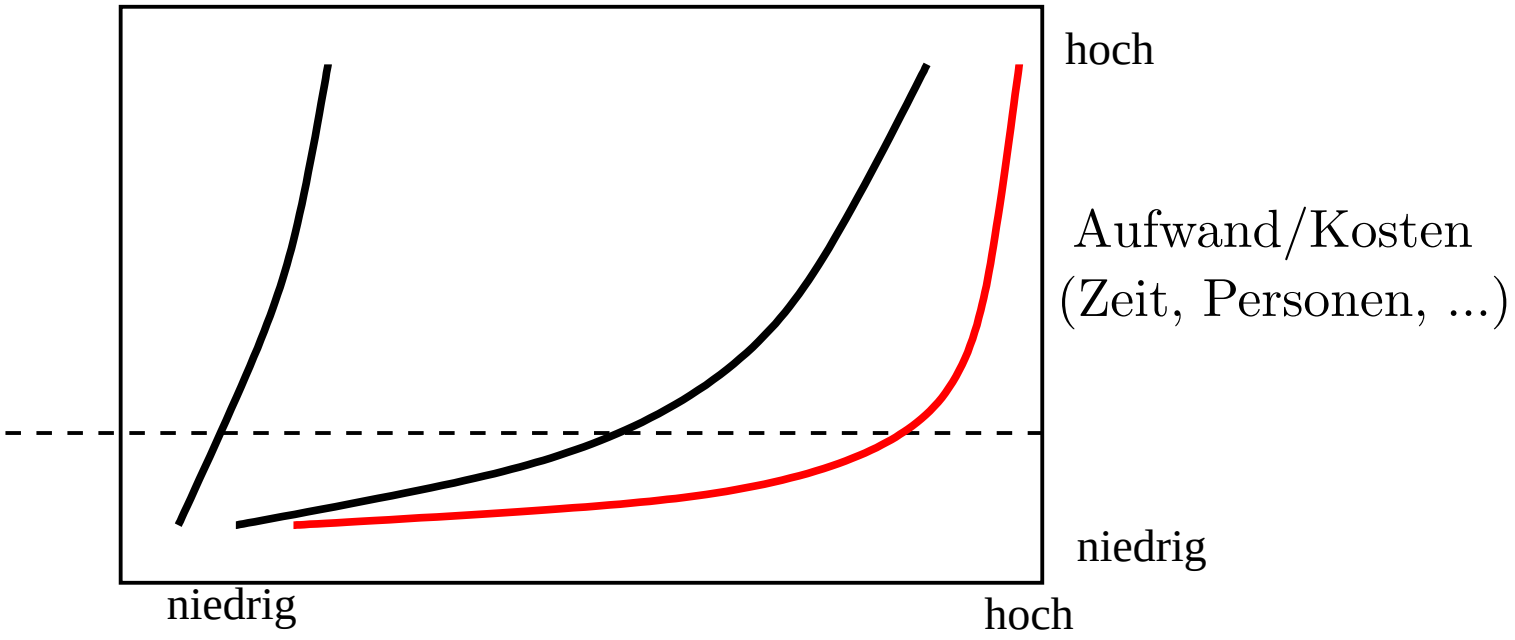


Quelle: http://spaceflight.nasa.gov/

- Soft-/Hardwarefehler unvermeidbar — wer versichert die ISS?

Wissenschaftliche vs. industrielle Interessen

- Erfahrung: Mehr Rigorosität $\approx$ mehr Verläßlichkeit.
- Erfahrung: Mehr Rigorosität $\approx$ höhere Kosten.



Forschungsagenda

- Ziel: Theoretisch abgesicherte Ingenieursmethodik
  - Analyse durch mathematische Modellierung.
    - Beitrag: Systemmodelle der Fehlertoleranz.
  - Verständnis/Intuition entwickeln für Wirkungsweise von existierenden Methoden der Fehlertoleranz.
    - Beitrag: Begrifflichkeit der Redundanz.
  - Grundlagen legen für die Ingenieurspraxis.
    - Beitrag: Methoden der Beobachtungen.
- "Angebote an die Ingenieursintuition."
- Hier: Konzentration auf "Redundanz"

Eigenschaften verteilter Systeme

- System Σ: nichtdeterministische Zustandsmaschine
- Eigenschaft P: Menge von Abläufen $\sigma = s_1, s_2, \ldots$ ($sem(\Sigma) =$ alle Abläufe des Automaten)
- Sicherheitseigenschaft S ("immer . . . "):

$$\forall \sigma : \sigma \notin S \Rightarrow (\exists i : \forall \beta : (\sigma|_i \cdot \beta \notin S))$$

- Lebendigkeitseigenschaft L ("schlußendlich . . . "):

$$\forall \sigma : \forall i : \exists \beta : (\sigma|_i \cdot \beta \in L)$$

Fehlertoleranzspezifikationen [Arora and Kulkarni 1998; Gärtner 1999a]

- Eigenschaftsklassen sind fundamental [Alpern and Schneider 1985]: $\forall P : \exists S, L : (P = S \cap L)$

- Fehler können zu einer Verletzung von S oder L führen.

bei Fehlern. . .	S erfüllt	$\Diamond S$ erfüllt
L erfüllt	maskierend	nicht-maskierend
L nicht erfüllt	*fail-safe*	keine

- Klassifikation analog zu Erfahrungen aus der Praxis

Fehlertoleranzkomponenten

- Fehlertolerantes Programm =
  fehler-intolerantes Programm + Fehlertoleranzkomponenten
- Detektor: erkennt Systemzustand
- Korrektor: erzwingt Systemzustand
- Abstraktionen von vielen bekannten Fehlertoleranzmechanismen, gute Strukturierungsmöglichkeit [Gärtner 1998].

Fehlertoleranztheorie

- Theoreme [Arora and Kulkarni 1998]:
  - Detektoren sind notwendig und hinreichend für Sicherheitseigenschaften.
  - Korrektoren sind notwendig und hinreichend für Lebendigkeitseigenschaften

- Funktionsprinzip: Redundanz [Gärtner and Völzer 2001]
  - Platzredundanz = nicht-erreichbarer Zustand
  - Zeitredundanz = nicht-ausführbare Transition

Notwendigkeit von Platzredundanz

- $\Sigma = (C, I, T, A)$, Fehlermodell F, Eigenschaft P.
- Σ erfüllt P, $F(\Sigma)$ verletzt P.
- Baue Σ' aus Σ so daß $F(\Sigma')$ erfüllt P.
- Dann hat Σ' nicht erreichbare Zustände.
- Beweis: Σ' besitzt eine Transition, die auch in Σ zur Verletzung von P geführt hätte.

Anwendung der Theorie

- Alle gängigen Fehlertoleranzverfahren können als Instanzen der allgemeinen Theorie angesehen werden.

- *triple modular redundancy* (TMR):
  - System wird dreifach repliziert, Ausgaben durch einen *voter* abgeglichen.
  - Toleriert den Totalausfall einer Komponente.
  - Platzredundanz liegt im Kreuzprodukt der Zustandsräume.
  - Zeitredundanz bei der Fehlerbehandlung im *voter*.

- Fehlererkennende Codes:
  - *parity bit* bei der Datenübertragung
  - Platzredundanz durch vergrößerung des Zustandsraumes.

Fragen für Ingenieure

- Welche Art von Fehlertoleranz soll erreicht werden (maskierend, nichtmaskierend, *fail-safe*)?
- Welche Auswirkungen auf die Systemeigenschaften haben diese Fehlermodelle (Lösbarkeit, Menge der Redundanz)?
- Wo benötige ich Detektoren, wo Korrektoren?
- Wie baue ich Detektoren und Korrektoren schrittweise in das System ein?
- Wie arbeite ich redundanzoptimal?

Weitere Beiträge

- Untersuchung der Systemmodelle (asynchron, partiell synchron, zeitbeschränkt asynchron, Fehlerdetektormodell, . . .):
  - Welches Systemmodell sollte verwendet werden?
- Andere Systemsemantik (*Halbordnung*):
  - Neu: Definition sinnvoller Detektionssemantiken zum Beobachten in fehlerbehafteten Systemen.
  - Neu: Entsprechende Beobachtungsalgorithmen.
  - Nutzen: Verwendbar als Detektionsmodule in Fehlertoleranzverfahren.

Zusammenfassung

- Solide, theoretisch fundierte methodologische Basis notwendig um verläßliche Systeme zu konstruieren.
- Arora/Kulkarni-Theorie erlaubt feingranularen Entwurf von Fehlertoleranzmethoden.
- Neu: Erweiterung der Theorie um Begrifflichkeit der Redundanz.
  - Begrifflichkeit der Redundanz erlaubt Redundanz zu messen (z.B. Zählen von redundanten Zuständen/Transitionen).
  - Nutzen: Systeme werden bezüglich Redundanz vergleichbar.
  - Nutzen: Sowohl theoretische [Kulkarni and Arora 2000] als auch didaktische [Gärtner 1999b] Weiterentwicklungen des Gebietes.

Danksagungen

- Folien produziert unter Verwendung von pdfLaTeX und Klaus Guntermanns PPower4.

References

Alpern, B. and Schneider, F. B. 1985. Defining liveness. *Information Processing Letters 21*, 181–185.

Arora, A. and Kulkarni, S. S. 1998. Component based design of multitolerant systems. *IEEE Transactions on Software Engineering 24*, 1 (Jan.), 63–78.

Gärtner, F. C. 1998. Fundamentals of fault tolerant distributed computing in asynchronous environments. Technical Report TUD-BS-1998-02 (July), Darmstadt University of Technology, Darmstadt, Germany. To appear in *ACM Computing Surveys*, 31(1), March 1999.

Gärtner, F. C. 1999a. An exercise in systematically deriving fault-tolerance specifications. In *Proceedings of the Third European Research Seminar on Advances in Distributed Systems (ERSADS)* (Madeira Island, Portugal, April 1999).

Gärtner, F. C. 1999b. Fundamentals of fault-tolerant distributed computing in asynchronous environments. *ACM Computing Surveys 31*, 1 (March), 1–26.

Gärtner, F. C. and Völzer, H. 2001. Defining redundancy in fault-tolerant computing. In *Brief Announcement at the 15th International Symposium on DIStributed Computing (DISC 2001)* (Lisbon, Portugal, Oct. 2001).

Kulkarni, S. S. and Arora, A. 2000. Automating the addition of fault-tolerance. In M. Joseph Ed., *Formal Techniques in Real-Time and Fault-Tolerant Systems, 6th International Symposium (FTRTFT 2000) Proceedings*, Number 1926 in Lecture Notes in Computer Science (Pune, India, Sept. 2000), pp. 82–93. Springer-Verlag.