# Autonomic Computing

## Wie sich Computer selbst konfigurieren, warten und verteidigen.

Thomas Liske

Technische Universität Dresden

2005 / Hauptseminar Rechnernetze

# Gliederung

# Aufwand in heutigen Computer Systemen

- 30% - 40% des IT-Budget zum Schutz vor Ausfällen und zur Wiederherstellung
- 60% - 75% des TCO einer Datenbank entstehen durch Administration
- 40% von Computerausfällen durch Benutzer bzw. Administrator verursacht

# Was ist Autonomic Computing?

Paul Horn (IBM Research) 2001 mit seinem
*Manifest über den Stand der Informationstechnik*:

Einleitung der Forschungsinitiative *Autonomic Computing*

# Was sind Autonomic Computing Systeme?

Computersysteme die:

1. sich “selbst kennen” (Komponenten, Zustand, ...),
2. sich automatisch Konfigurieren,
3. versuchen ihre Effiktivität zu erhöhen,
4. fehlertolerant sind,
5. sich selbst vor Dritten schützen,
6. sich an ihre Umgebung / ihren Kontext anpassen,
7. offene Standards zur Kommunikation verwenden und
8. die Komplexität vor dem Benutzer “verstecken”

# Selbstkonfigurierend

- Heute
  - aufwendige und fehlerträchtige Installation, Konfiguration und Integration verschiedener Systeme
- Autonomic Computing
  - automatische Konfiguration der Komponenten und Systeme entsprechend eine höheren Regelwerkes

# Selbstheilend

- Heute
  - Fehlersuche in großen komplexen Systemen personal- und zeitaufwendig
- Autonomic Computing
  - automatische Erkennen und Beheben von Fehlern in Soft- und Hardware

# Selbstoptimierend

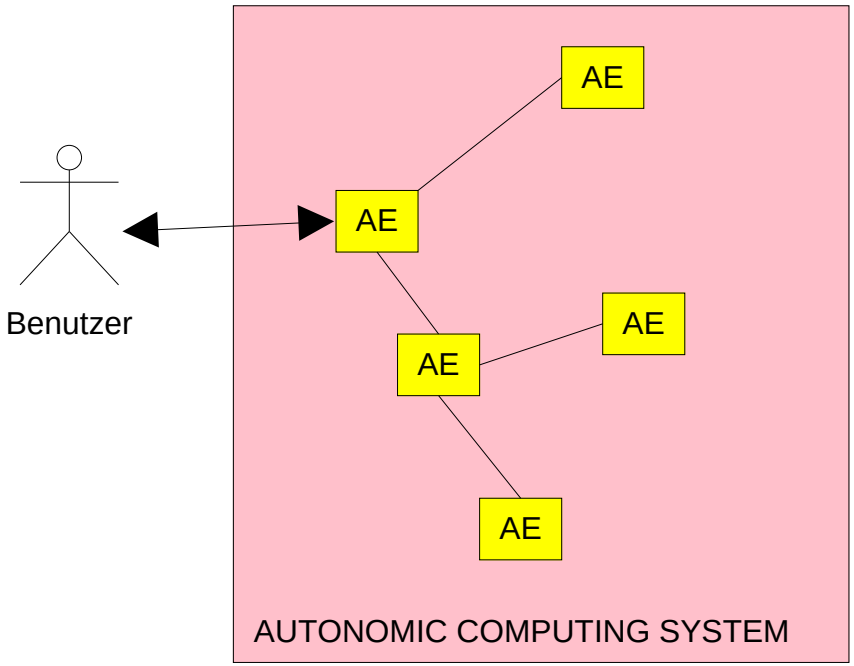
- Heute
  - viele konfigurierbare Parameter in verschiedenen Teilen von Systemen
- Autonomic Computing
  - automatische und kontinuierliche Optimierung von allen Komponenten des Gesamtsystems

# Selbstschützend

- Heute
  - meist nur manuelles reagieren auf Angriffe
  - kein komplettes Frühwarnsystem
- Autonomic Computing
  - System erkennen und bekämpfen Angriff selbständig
  - Frühwarnsystem zur Verhinderung von Komplettausfällen

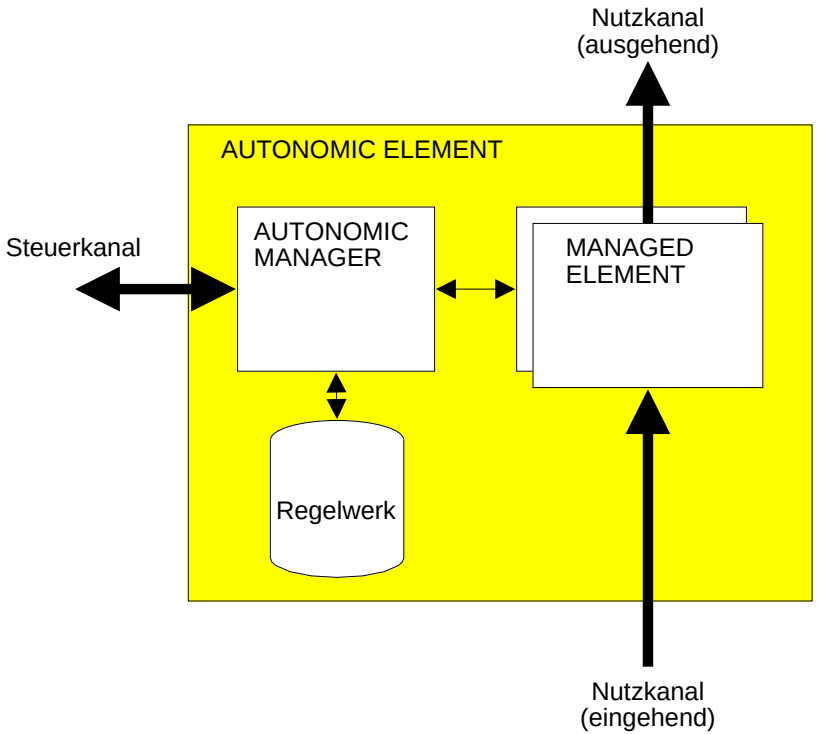
# Gliederung

# Aufbau von Autonomic Computing Systems

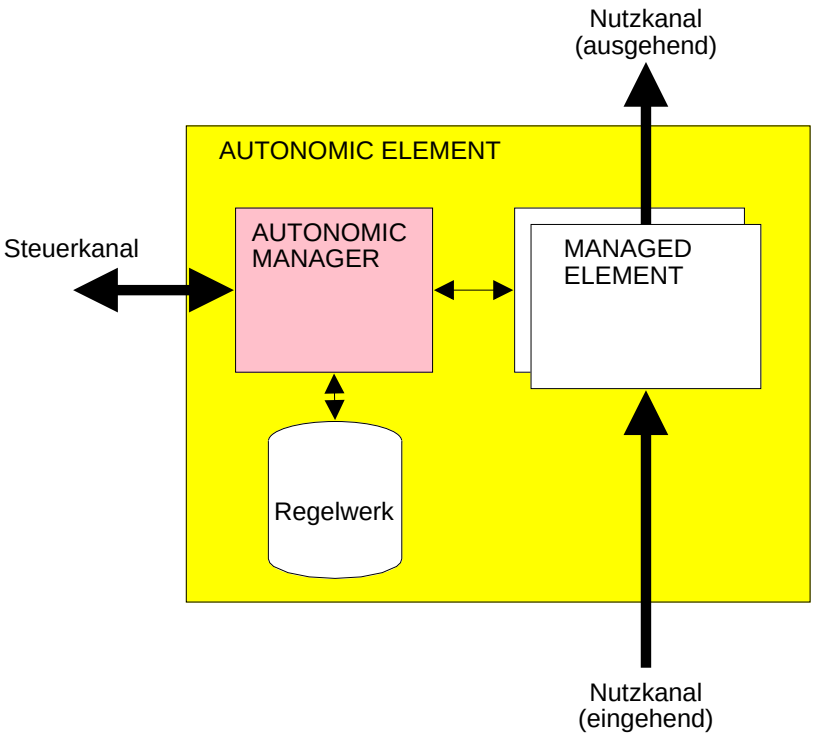



- Verbund von *Autonomic Elements*
- Verhalten wird durch ein *Regelwerk* bestimmt

# Das Autonomic Element

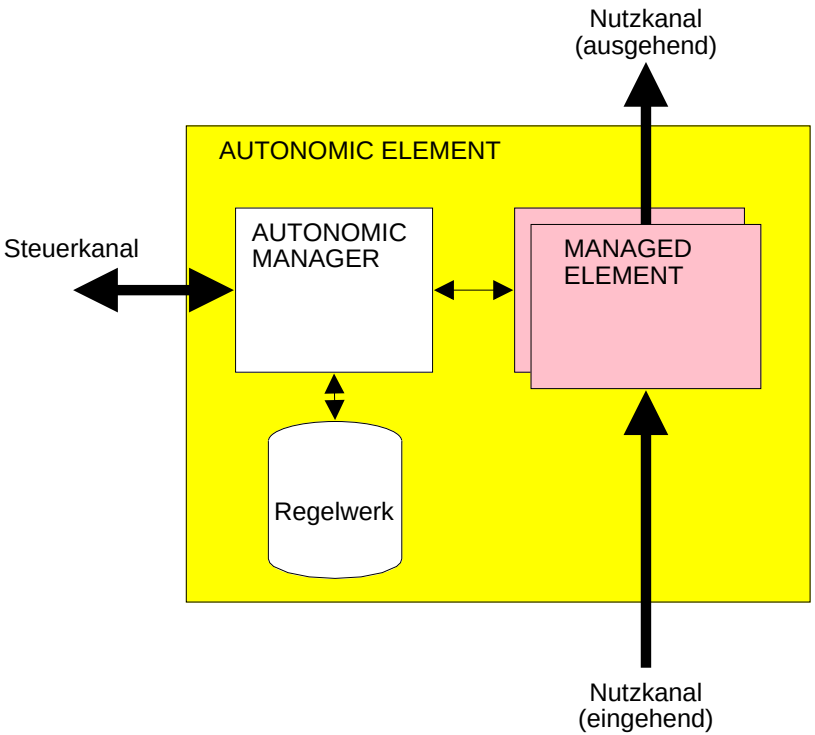


# Der Autonomic Manager



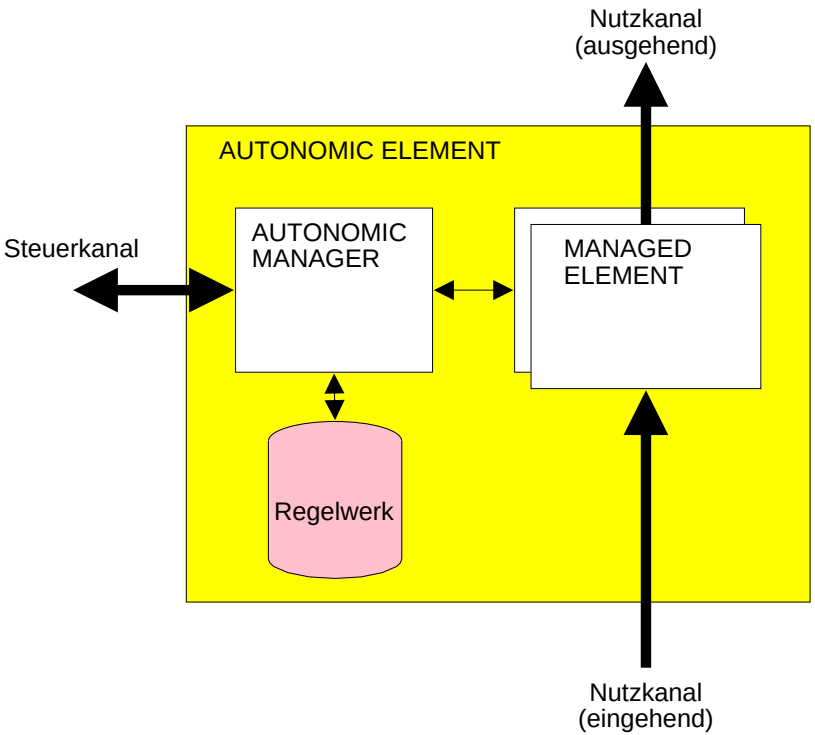

- analysiert den Zusammenhang mit anderen *Autonomic Elements*
- überwacht das *Managed Element*

# Das Managed Element




- enthält nutzbare Resource
- überwacht und gesteuert durch den *Autonomic Manager*

# Das Regelwerk




- definiert die zu erreichenden Ziele des *Automic Elements*

# Phase 1: Spezifikationen

- Registrierung in einem Verzeichnisdienst
- Wissen über angebotene und benötigte Dienste
- standardisierte Ontologien und Beschreibungen für Dienste benötigt

# Phase 2: Lokalisierung

- Auffinden benötigter Dienste
- Entscheidung über Verfügbarkeit und Vertraulichkeit der Anbieter

# Phase 3: Aushandeln von Parametern

- Ableiten der Anforderungen aus Benutzereingaben oder Berechnungen
- Aushandeln von Parametern wie
  - Preis oder
  - Dienstgüte

# Phase 4: Einrichtung

- Reservierung von Resourcen auf den *managed Elements* für ausgehandelte Beziehungen
- evtl. Anfordern von weiteren Diensten von anderen *Autonomic Elements*

# Phase 5: Nutzung

- Überwachung der *managed Elements*
- Überwachung des Dienst-Nutzers

# Phase 6: Auflösung

- Freigabe reservierter Resourcen
- evtl. Speicherung des geleisteten Dienstes für Verrechnungszwecke

# Entwurf, Test und Verifikation

- Werkzeuge zum Erstellen
  - von Beziehungen zu anderen *Autonomic Elements*
  - des *Regelwerkes*
- Test und Verifikation schwierig

# Installation und Konfiguration

- *Autonomic Element* registriert sich selbst bei einem Verzeichnisdienst
- findet andere benötigte *Autonomic Elements* über Verzeichnisdienst

# Überwachung und Problemerkennung

- Überwachen ob eigenes *Regelwerk* eingehalten wird
- Überwachung ist Basis für Optimierung und Problemerkennung

# Updates

- Selbstständiges finden und installieren von Updates
- bei Problemen können Updates rückgängig gemacht werden

# Gliederung

# Sicherheit in Autonomic Computing Systems

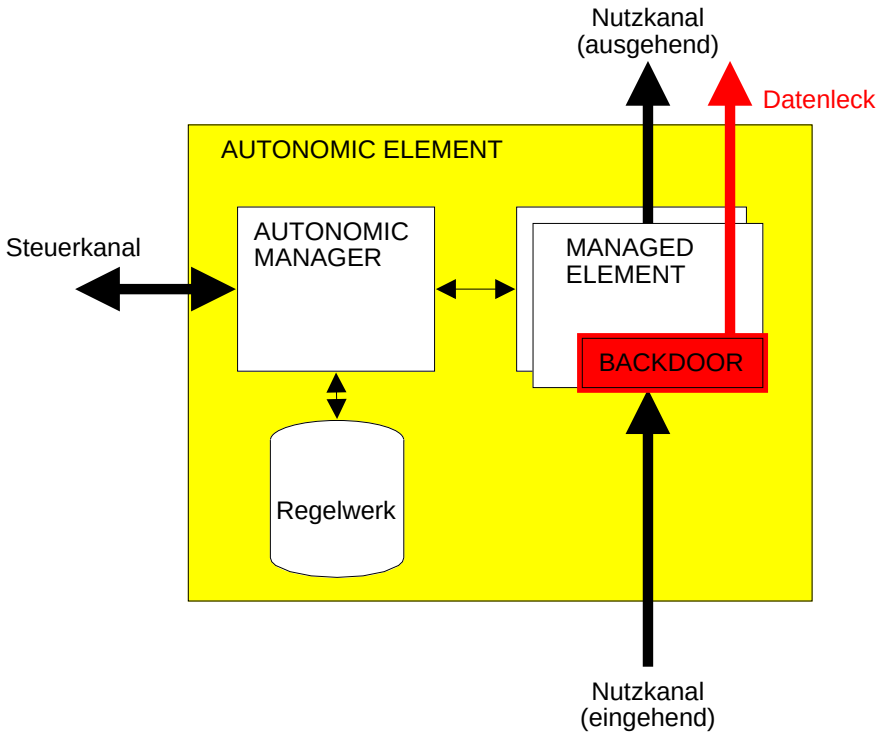
- traditionelle Sicherheitsprobleme und -lösungen bleiben bestehen
- das System muß sicher sein für
  - jede Konfiguration und
  - jeden Zustand

  den es selbständig erreichen kann

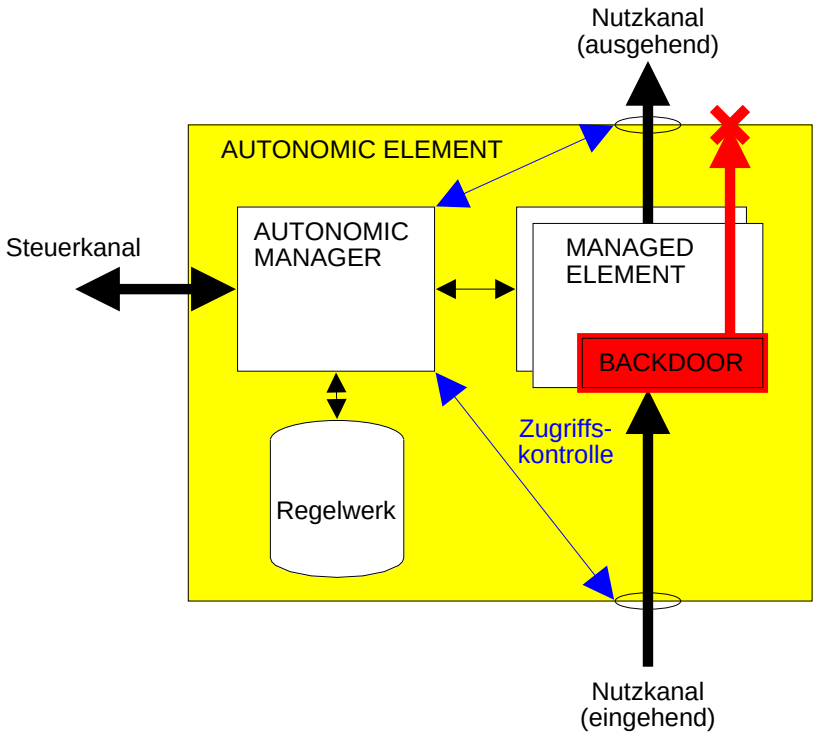
# Sicherheit zwischen Autonomic Elements

- Sicherheit bereits auf Ebene der *Autonomic Elements* durchgesetzt
- Vertrauen der *Autonomic Elements* untereinander durch:
  - web-of-trust
  - Certificate Authorities

# Backdoors in Autonomic Elements



# Backdoors in Autonomic Elements




- blockieren von Kommunikationskanälen die nicht dem Regelwerk entsprechen
- kein Schutz vor Covered Channels

# Ablauf einer Kompromittierung

Unterteilung in drei Phasen:

- Kompromittierung,
- Detektierung und
- Wiederherstellung

# Kompromittierung und Detektierung

- kompromittierte Systeme stellen Zuverlässigkeitsprobleme dar:
  - funktionieren nicht oder nur teilweise
  - liefern verfälschte Ergebnisse
- Detektierung
  - schwierig, nicht immer 100%-tige Sicherheit ob Kompromittierung vorliegt oder nicht
- Ziel: Minimierung der Zeit zwischen Kompromittierung und dessen Detektierung

# Wiederherstellung des Systems

- abschalten / isolieren kompromittierter Elemente
- System wird in zustandslosen und zustandsabhängigen Teil unterteilt
  - zustandslos: einfaches Wiederherstellen möglich (z.B. Neuinstallation)
  - zustandsabhängig: Wiederherstellen z.B. mittels *Secure Distributed Storage*
- System kann wieder freigeben werden

# Vorteile verteilter Datenspeicher

Verteilung von Daten über mehrere Knoten für

- Hochverfügbarkeit
- Korrektheit
- Vertraulichkeit

# Vorgehen

- verschlüsseln der Daten
- aufteilen der Daten in $N$ Teile, verteilen auf $N$ Knoten
- zum Wiederherstellen werden mind. $\frac{N}{2}$ intakte Teile benötigt
- Verfälschungen werden erkannt

# Zusammenfassung

- heutige System werden immer komplexer und damit fehleranfällig und teuer
- noch viele offene Fragen die gelöst werden müssen
- *Autonomic Computing Systeme* bieten Möglichkeiten für eine bessere Verfügbarkeit und höhere Sicherheit als heutige Computer Systeme

# Literatur


- J. O. Kephart und D. M. Chess
  *The Vision of Autonomic Computing*
  IEEE Computer, Januar 2003, Seite 41 - 50
- D. M. Chess, C. C. Palmer und S. R. White
  *Security in an autonomic computing environment*
  IBM SYSTEMS JOURNAL, VOL 42, NO 1, 2003, Seite 107 - 118
- A. G. Ganek und T. A. Corbi
  *The dawning of the autonomic computing era*
  IBM SYSTEMS JOURNAL, VOL 42, NO 1, 2003, Seite 5 - 18