Prüfungszusammenfassung

Modul Betriebssysteme (bsys)

Simon Wächter

2016

Inhalt

[1 Synchronisierung 2](#_Toc452927186)

[1.1 Mittel 2](#_Toc452927187)

[1.2 Deadlocks 2](#_Toc452927188)

[2 Verteilung 2](#_Toc452927189)

[2.1 Definition 2](#_Toc452927190)

[2.2 Verteilungsstrategien 2](#_Toc452927191)

[2.3 Anforderungen 2](#_Toc452927192)

[2.3.1 Transparenz 2](#_Toc452927193)

[2.3.2 Flexibilität 3](#_Toc452927194)

[2.3.3 Zuverlässigkeit 3](#_Toc452927195)

[2.3.4 Performance 3](#_Toc452927196)

[2.3.5 Skalierbarkeit 3](#_Toc452927197)

[2.4 Mach OS 3](#_Toc452927198)

[2.5 Synchronisation von verteilten Systemen 3](#_Toc452927199)

[2.5.1 Zeitsynchronisation 3](#_Toc452927200)

[2.5.2 Gegenseitiger Ausschluss über Systemgrenzen 4](#_Toc452927201)

[2.5.3 Vergleich der drei Algorithmen 4](#_Toc452927202)

[2.6 Gemeinsam genutzter Speicher 4](#_Toc452927203)

[2.7 Vergleich der Funktionalität von verteilten und monolithischen Systemen 5](#_Toc452927204)

[3 Sicherheitsaspekte 5](#_Toc452927205)

[3.1 Übersicht 5](#_Toc452927206)

[3.2 Unix 6](#_Toc452927207)

[3.3 Windows 6](#_Toc452927208)

[3.4 Schützenswerte Güter 6](#_Toc452927209)

[3.5 Härtung 6](#_Toc452927210)

[3.6 Security Monitoring 7](#_Toc452927211)

[3.7 Securityarchitektur 7](#_Toc452927212)

[3.8 Firmenweite End-to-End Security 8](#_Toc452927213)

[3.9 Sicherheitsmanagement 8](#_Toc452927214)

[3.10 Risikomanagement 8](#_Toc452927215)

[3.11 Operationelle IT Risiken 9](#_Toc452927216)

[3.12 IT Sicherheit im Spannungsfeld 9](#_Toc452927217)

[4 Modellierungen 9](#_Toc452927218)

# Synchronisierung

## Mittel

* Blocking Call
* Mutex
* Dijikstra Semaphore
* Additive Semaphore
* Semaphore Sets
* Barrier
* Rendevous

## Deadlocks

Auftreten:

1. Ressource ist nur exklusiv nutzbar
2. Ressource kann nicht abgegeben/entzogen werden
3. Belegte Ressourcen werden erneut belegt
4. Zyklische Ketten von Prozessen, bei welcher jeder Prozess eine Ressource der nächsten Kette benötigt

Gegenmassnahmen:

1. Sicherstellen, dass immer eine der Deadlock-Bedingungen nicht erfüllt ist
2. Ressourcenbedarf analysieren
3. Eingetretene Deadlocks erkennen und auflösen

# Verteilung

## Definition

Ein verteiltes System ist ein Zusammenschluss unabhängiger Computer, der sich für den Benutzer als ein einzelnes System präsentiert.

## Verteilungsstrategien

* Hardware / Software / Benutzung
* Client-Server-System: Viele Clients greifen auf einen oder mehrere Server bzw. Services zu.
* Verteiltes Dateisystem (z.B. NFS): Ein über mehrere Server verteiltes virtuelles Dateisystem steht Clients zur transparenten Benutzung zur Verfügung.
* Netzwerk-fähiges Betriebssystem (lose Kopplung): das Betriebssystem stellt system-übergreifende Funktionalität (z.B. rcp, rlogin, rsh unter Unix) zur Verfügung.
* Verteiltes Betriebssystem (enge Kopplung): Das Betriebssystem selbst ist verteilt, für Benutzer und Anwendungen ist dies nicht sichtbar.
* Verteilte Anwendung: Durch die Programmierung der Anwendung wird das verteilte System erstellt – das Programm muss in der Regel die Verteillogik kennen.

## Anforderungen

### Transparenz

* Ortstransparenz: Der Ort der genutzten Ressourcen / erbrachten Dienste ist für den Anwender nicht sichtbar.
* Migrationstransparenz: Ressourcen können verlagert werden, ohne dass sich ihr Name bzw. ihre Nutzung verändert.
* Replikationstransparenz: Anwender können nicht erkennen, wie viele Instanzen es gibt.
* Nebenläufigkeitstransparenz: Mehrere Anwender können die Ressourcen automatisch gemeinsam und unabhängig voneinander nutzen.
* Parallelitätstransparenz: Aktivitäten können parallel bzw. nebenläufig ausgeführt werden, ohne dass der Anwender es bemerkt.

### Flexibilität

* Services sollen dynamisch an den Bedarf anpassbar sein, entweder durch administrative Eingriffe oder durch Eigenkonfiguration zur Laufzeit.
* Änderungen sollen keinen kompletten Neustart des verteilten Systems erfordern.

### Zuverlässigkeit

* Erhöhte Zuverlässigkeit gegenüber Einzelsystemen trotz additiver Ausfallwahrscheinlichkeiten (Redundanz).
* End-zu-End Verfügbarkeit statt Komponentenverfügbarkeit.
* Sicherheit – gleiche Richtlinien & Umsetzung im gesamten verteilten System.
* Fehlertolerenz (versus Overhead).
* Automatisches Recovery von Komponenten.

### Performance

* Verteilungs- und Kommunikationsmehraufwand muss den Aufwand wert sein (Granularität, Fehlertolerenz).
* Wiederholbarkeit / Determinismus von Leistungsindikatoren.
* Abhängigkeit von nicht direkt kontrollierbaren Komponenten (z.B. LAN).
* End-zu-End Performance statt Komponenten- Performance.

### Skalierbarkeit

* Angebotsseite:
  + Statische oder dynamische Zufügung / Wegnahme von Servern.
  + Verrechnung: Durchschnitt oder „Peaks“?
  + Vermeiden von „Flaschenhälsen“ durch zu starke Serialisierung (Applikationskomponenten, Datenstrukturen, Algorithmen).
* Dienstnehmerseite:
  + Nicht vorhersagbare Anzahl Dienstnehmer.
  + Einhaltung von Dienstgütegarantien

## Mach OS

* Microkernel Architektur
* Komponenten:
  + Task: Umgebung zum Ausführen von Prozessen
  + Thread: Einzelner Ablauf innerhalb eines Tasks
  + Port: Kommunikationskanal
  + Nachrichten: Zwischen Tasks versendete Daten, welche über einen Port fliessen
  + Memory Objects: Speicher für den Task

## Synchronisation von verteilten Systemen

### Zeitsynchronisation

* Absolut
  + Zeitserver (z.B. Network Time Protocol, NTP)
  + Eigene Referenz-Uhr (z.B. Funkempfänger)
* Relativ
  + Median mehrerer interner oder externer Quellen
  + Neuester Zeitstempel gilt
  + Maximale Abweichung begrenzen, sonst Lokalzeit nutzen
* Vorstellen ist einfacher als Rückstellen (Backup, „make etc.)

### Gegenseitiger Ausschluss über Systemgrenzen

* Zentraler Algorithmus (mit Redundanz)
  + Zuverlässig, berechenbar
  + Starke Serialisierung
* Verteilter Algorithmus
  + Zeitliche Ordnung im Gesamtsystem (setzt synchrone Uhrzeiten voraus)
  + Getimte Nachricht an alle Prozesse und Warten auf Bestätigung (setzt zuverlässige Nachrichtenübermittlung voraus)
* Token Ring Algorithmus
  + Explizite Erlaubnis zum Zugriff in vordefinierter Reihenfolge
  + Suboptimale Ressourcennutzung / Wartezeiten

### Vergleich der drei Algorithmen

* Zentraler Algorithmus (mit Redundanz)
  + Nachrichten pro Eintritt/Verlassen: 3
  + Verzögerung vor dem Eintritt (in Nachrichten): 2
  + Probleme: Absturz des zentralen Koordinators
* Verteilter Algorithmus
  + Nachrichten pro Eintritt/Verlassen: 2 \* (n - 1)
  + Verzögerung vor dem Eintritt (in Nachrichten): 2 \* (n - 1)
  + Probleme: Absturz / Nicht-Antwort von Prozessen
* Token Ring Algorithmus
  + Nachrichten pro Eintritt/Verlassen: 1 bis unendlich
  + Verzögerung vor dem Eintritt (in Nachrichten): 0 bis n -1
  + Probleme: Verlorener Token, Prozessabsturz

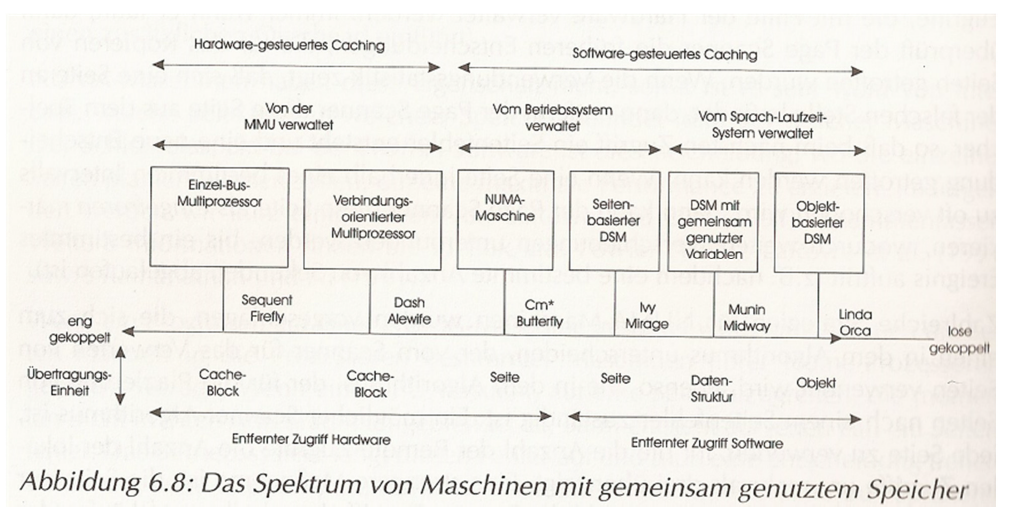
## Gemeinsam genutzter Speicher

Eng gekoppelt:

* (Non-Uniform Memory Access) NUMA: Hardware- Multiprozessoren mit diversen Prozessoren am gleichen Bussystem

Lose gekoppelt:

* Distributed Shared Memory (DSM): Ein Seitenladefehler wird dadurch aufgelöst, dass eine Seite nicht von der lokalen Disk geladen wird, sondern von einer entfernten Disk
* Schlechte Performance, lange Netz-Wartezeiten
* Optimierung durch Anlage nur der gemeinsam benutzten Datenstrukturen auf gemeinsam genutzten Seiten, alle anderen Seiten bleiben lokal.



## Vergleich der Funktionalität von verteilten und monolithischen Systemen

* Ein monolithisches Betriebssystem präferiert Effizienz vor Robustheit. Obwohl das Design die Trennung funktionaler Blöcke vorsieht, werden Privilegien nicht auf Teile des Betriebssystems beschränkt (z.B. „root“ Ausführungsrechte im Datei- und Prozesssystem). Die verwendeten Kommunikationsmechanismen (Signale, Pipes, Share Memory, Sockets usw.) sind überall identisch.
* Eine Schichtenarchitektur des Betriebssystems fördert die Robustheit durch die Anordnung von Funktionalität in gegeneinander abgegrenzte Schichten mit unterschiedlichen Privilegien. Der höchstprivilegierte Block behandelt Interrupt- Behandlung und Kontextwechsel, die darüber liegenden Schichten beinhalten Gerätetreiber, Speicherverwaltung, das Dateisystem und die Benutzerschnittstelle. Am oberen Ende folgt schliesslich die am wenigsten priorisierte Applikationsschicht.
* Eine Microkernel-Architektur präferiert Robustheit vor Effizienz. Die den einzelnen Teilen des Betriebssystems zugeordneten Privilegien sind so restriktiv wie möglich und die Kommunikation zwischen den Blöcken basiert auf spezialisierten Kommunikationsmechanismen, die auf die jeweiligen Zugriffsrechte begrenzt sind (nur wenige Teile des Betriebssystems benötigen mehr Rechte als die Applikationen). Somit kann das Betriebssystem oberhalb des Microkernels auf wenige Service begrenzt werden, die die Basisfunktionen des Betriebssystems realisieren.

# Sicherheitsaspekte

## Übersicht

* Typische Schwachstellen:
  + Lieferwege (physisch, aber vor allem elektronisch)
  + Software-Voreinstellungen (Defaults)
  + Funktionale Fehler der Software / Ausnutzbarkeit von Nebeneffekten
  + Nicht getestete oder nicht autorisierte Änderungen
  + Fremd-/Fernzugriffe oder Auslagerung aus dem Sicherheits-Kontext
  + Der Benutzer
  + Der Administrator
* Ursachen:
  + Komplexes Zusammenwirken verschiedener Effekte
  + Ungerechtfertigtes Vertrauen („Es sah aber echt aus“)
  + Gutwilligkeit der Beteiligten („Wird schon stimmen“)
  + Fehlerhafte Ausführung und/oder Kontrolle (insbes. unter Zeitdruck)
  + Böswilligkeit / Vorsatz (Innen- oder Aussentäterschaft)

## Unix

* Fehlerhafte Default-Konfigurationen
* Unnötige (offene) Services / Ports
* Zu wenig restriktive Zugriffsrechte
* Technische Benutzer ohne Passworte
* Brute Force Passwort-Angriffe auf Dienste wie telnet, ftp, ssh (Würmer, Botnets)
* Ausnutzung von Software-Schwachstellen (meist Authentisierung, Speicherüberlauf, Escape-Shells)

## Windows

* Angriffe auf „Service Control“ Programme (SCP) unter Kontrolle des „Service Control Manager“ (SCM, „services.exe“)
* Ausnutzung von Software-Schwachstellen („Buffer Overflow“) 🡪 aufgrund der schwachen betriebssysteminternen Abgrenzung des Speicher-Management
* Nicht benötigte Services mit schwachem Schutz
* Lokale Administratorenrechte

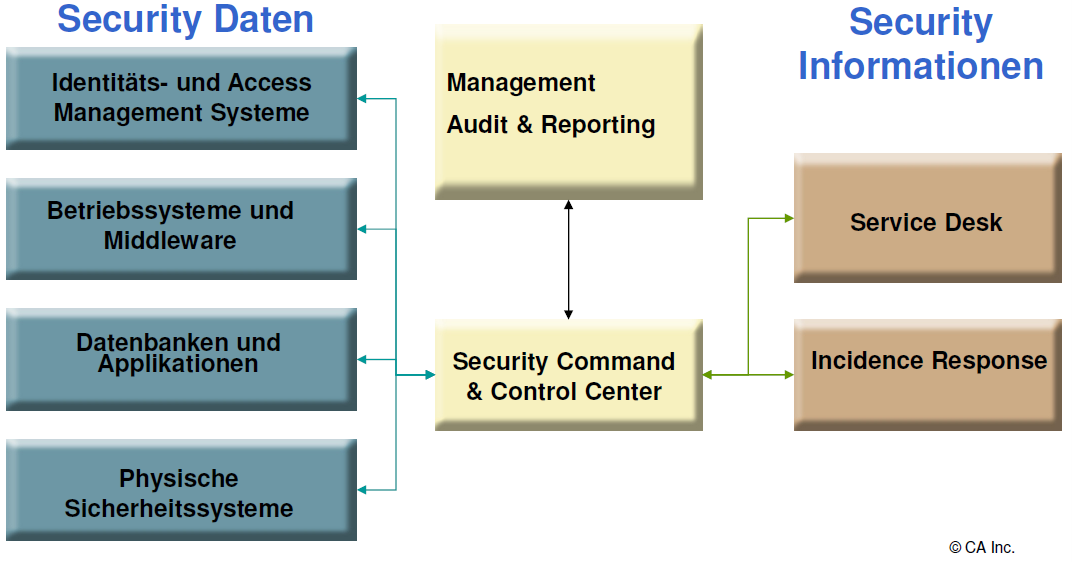
## Schützenswerte Güter

* Verfügbarkeit
* Vertraulichkeit
* Korrektheit
* Zeitliche Abfolge
* Bereitstellungstermin
* Identität / Autorisierung
* Handlungsfähigkeit
* Guter Ruf / Image

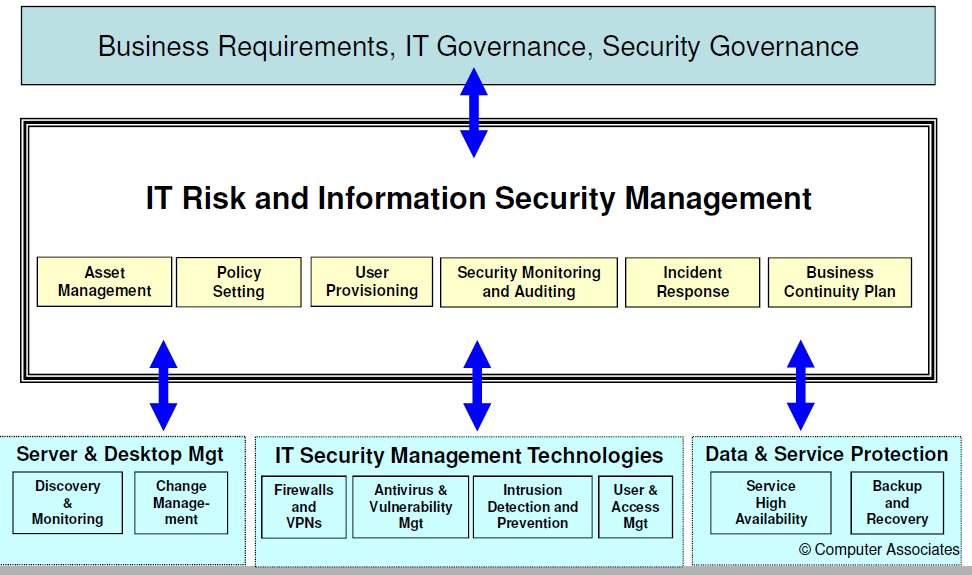
## Härtung

* Identifikation relevanter Systeme oder Applikationen
* Technische Überprüfung (Security Scan, Penetration Testing, Application Security Audit, Ethical Hacking)
* Organisatorische Überprüfung (ICT Bedrohungs- und Risikoanalyse)
* Definition des Massnahmenkatalog zur Erhöhung des Sicherheitsniveaus:
  + Default-Installationen kritisch hinterfragen 🡪 ggf. eigener Default
  + Konfigurationsänderungen (Rechte, Rollen, Prozesse etc.)
  + Einspielen von Security-relevanten Updates
  + Entfernen nicht benötigter HW/SW-Komponenten oder Schnittstellen
  + Installation von sicherheitsspezifischer Zusatzsoftware
  + Zyklische Neuüberprüfung von aussen (Scan) oder innen (z.B. Secunia)
  + Veränderungen / Verbesserungen im Betriebsablauf oder Überwachung
  + Auswertung „echter“ Angriffsversuche oder Kollateralinformation

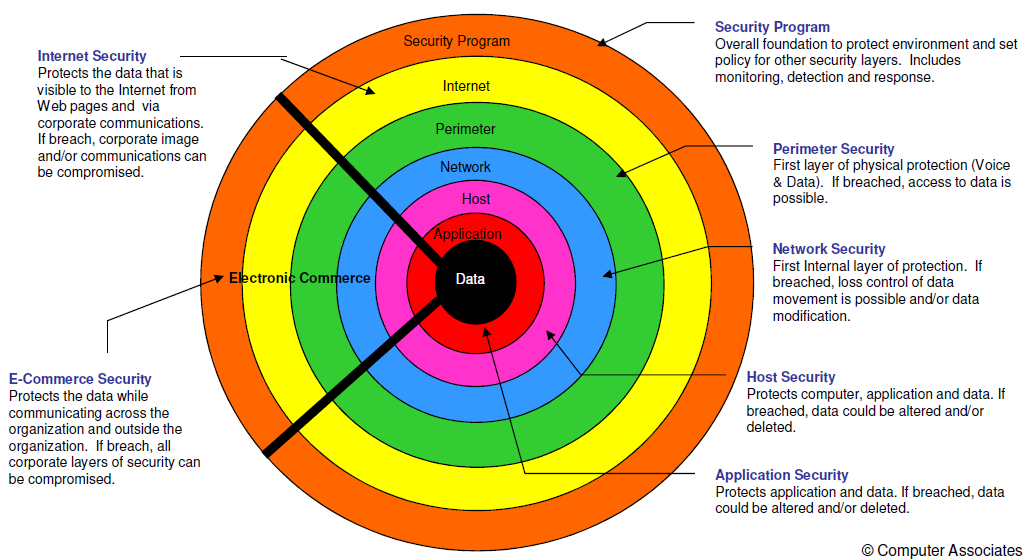
## Security Monitoring



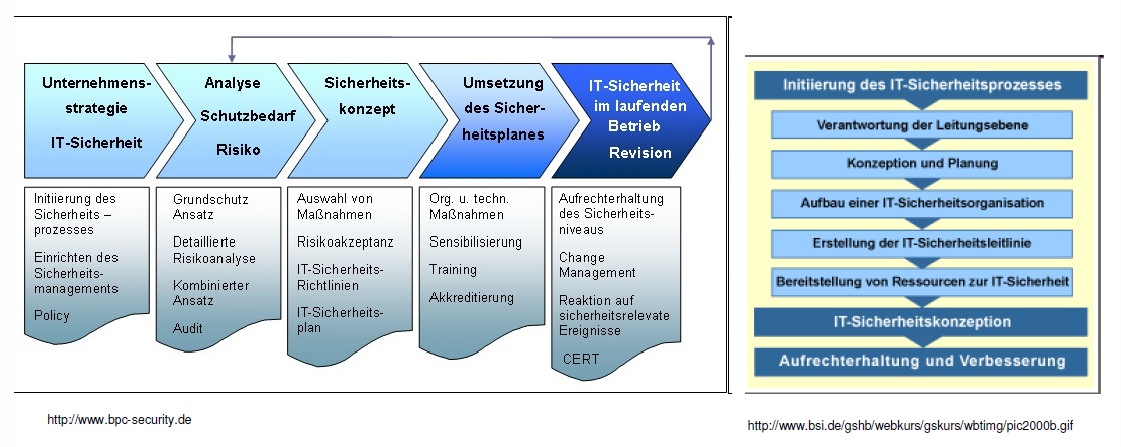
## Securityarchitektur



## Firmenweite End-to-End Security



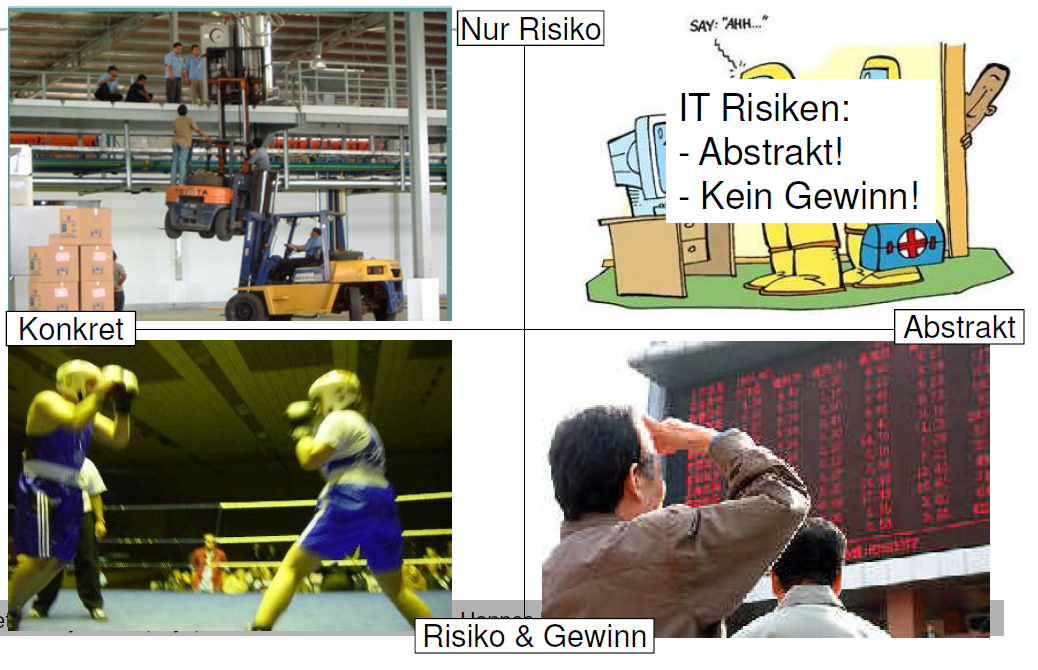
## Sicherheitsmanagement



## Risikomanagement

* Die wissentliche oder unwissentliche Akzeptanz einer Verlustwahrscheinlichkeit und möglichen Schadenhöhe.
* Risiko-Management durch:
  + Analyse
  + Vermeidung (nur bedingt möglich etc.)
  + Übertragung (Versicherung, Werkschutz etc.)
  + Begrenzung (präventive Massnahmen etc.)
  + Akzeptanz (formaler, dokumentierter Willensakt)
  + Ignoranz (wegschauen etc.)
* Meist gibt es eine Mischung der Massnahmen, abhängig vom Risikotyp, der Risikokultur, den Kostenfolgen usw.

## Operationelle IT Risiken



## IT Sicherheit im Spannungsfeld

* “Die sicherste Bank hat keine Türen, aber auch keine Kunden” - IT-Sicherheit kann im Widerspruch zu Elementen der Geschäftstätigkeiten stehen.
* Auch bei einer funktionierenden Zusammenarbeit zwischen IT und Business kann es zu Zielkonflikten kommen, die systematisch gelöst werden müssen.
* Das Business trägt die abschliessende Verantwortung für Risiko-Entscheide.
* Die IT und die IT-Sicherheit müssen das Business wesentlich besser verstehen.

# Modellierungen

