

Software Safety: Risiken bei Informatiksystemen

Prof. Dr. Lutz Prechelt, Christopher Özbek

Freie Universität Berlin
Institut für Informatik
<http://www.inf.fu-berlin.de/w/SE/>

- Begriffsdefinitionen
- Arten von Gefahren/Risiken
 - Technik
 - Menschen
 - Ereignisse/Kopplungen
- Beispiele
- Vorgehen

- Geboren in Homburg/Saar
- Informatik-Studium
 - 2000-2003 Universität Karlsruhe
 - 2003-2004 Georgia Institut of Technology, Atlanta, USA
 - Fulbright Stipendiat
 - Abschluss: Master of Science in Computer Science
- Seit 2004 Wissenschaftlicher Mitarbeiter
 - Freie Universität Berlin
 - Institut für Informatik
 - Arbeitsgruppe Software Engineering unter Prof. Lutz Prechelt
 - Doktorand im Bereich Open Source Software Engineering
 - Disserationsthema: Innovationseinführung in Open Source Projekten

- Wir betrachten die Auswirkungen von Informatiksystemen
- spezieller:
 - Wir interessieren uns für Bedrohungen der Sicherheit
 - Insbesondere Gefahren für "Leib und Leben"
 - aber auch schwächere Arten
(z.B. plötzliche Freiheitseinschränkungen)

~~Software~~-Safety

System-Safety

- Unfall (*accident*):
 - Unerwünschtes, ungeplantes (aber nicht zwingend ganz unerwartetes) Ereignis, das zu einem Schaden oder Verlust (Zeit, Geld, Menschenleben, Kunden,) führt
- Bedrohung, Gefahr (*threat, hazard*):
 - Eine mögliche Einwirkung auf das System, die alleine oder zusammen mit anderen zu einem Unfall führen kann
- Versagen (*failure*):
 - Eine untolerierbare Abweichung eines Systems von seiner Spezifikation
 - Kann zu einem Unfall führen, muss es aber nicht
- Sicherheit (*safety*):
 - Zustand des Geschütztseins vor Unfällen

Definitionen (2)

- Risiko (*risk*):
 - Produkt aus Eintrittswahrscheinlichkeit eines Unfalls und der dadurch entstandenen Schadenshöhe (quantitative Größe)
 - Solche Berechnungen sind aber meist nicht zielführend
 - Umgangssprachlich: Möglichkeiten, wie ein System einen Schaden hervorrufen kann
- Zuverlässigkeit (*reliability*):
 - Qualitativ: das Funktionieren eines Systems gemäß der Spezifikation unter **allen** vorgesehenen(!) Bedingungen
 - Quantitativ: Wahrscheinlichkeit des Nichtversagens
- Robustheit (*robustness*):
 - Fähigkeit eines Systems, auch unter *unvorhergesehenen* Bedingungen Unfälle auszuschließen

Definitionen (3)

- Mangel, Defekt (*defect*):
 - Eine strukturelle Unzulänglichkeit in einem System, die zu Versagen führt oder führen kann
- Fehler (*error*):
 - Ein Ereignis beim Bau eines Systems, das zu einem Mangel führt oder führen kann
- Schutz (*security*):
 - Die Widerstandsfähigkeit eines Systems gegen absichtliche Angriffe. Das System ist sicher (geschützt, *secure*), wenn die Angriffe ohne Unfall überstanden werden
 - Teilaspekt von Sicherheit

1. Software enthält Defekte
2. Hardware und technische Einrichtungen versagen
3. Menschen sind
 - fehlbar
 - insbesondere unter Stress
 - unvorsichtig
 - insbesondere wenn sie sich sicher fühlen
 - oftmals ignorant
 - manchmal böswillig
4. Auch unwahrscheinliche Ereignisse treten gelegentlich ein
 - auch mehrere zugleich
5. Scheinbar unverbundene Ereignisse sind tatsächlich oftmals eng gekoppelt

Beispiele folgen (Einordnung ist of mehrdeutig)

1. Software enthält Defekte

- Bekanntlich gibt es in umfangreichen Softwareprogrammen auch nach gründlichem Test noch mehrere Entwurfs- oder Programmiermängel
- Viele davon lassen sich nur extrem schwer mit Tests entdecken
- Man kann sie im Prinzip mit Durchsichten entdecken. Das gelingt aber nur, wenn man genau weiß, wonach man sucht
 - d.h. die Anforderungen sind bekannt und verstanden
 - Das ist aber nicht immer der Fall
- Beispiele folgen
 - Achtung: Häufig passen sie auch in andere Kategorien

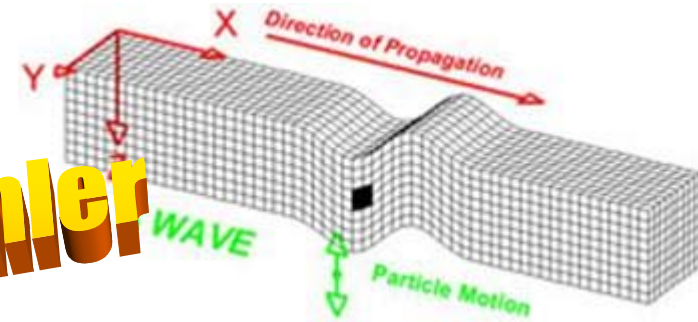
Bsp: Simulations-Software für Space-Shuttle-Missionen

- Zur Vorbereitung auf die zweite Mission des Space-Shuttle übten die Astronauten mit einem **Simulator**
- In einer bestimmten Situation entschieden sie sich, die Mission abubrechen, überlegten es sich dann aber doch anders und brachen das Abbruch ab
 - Die Simulation wurde weiter
- Nach einer Erdumrundung brachen sie die Mission allerdings erneut ab
- Das Programm geriet in dieser Situation in eine Endlosschleife
- **Die Entwerfer hatten die Möglichkeit zweier Abbrüche nie bedacht**
 - Software Engineering Notes (SEN) 8(3), 1983



Bsp: Erdbeben-Simulation 1979

- In einem Programm zur Simulation der Wirkung von Erdbeben auf Gebäude wurde ein kleiner Fehler entdeckt:
 - An einer Stelle hätte man $\sum_k (|x_k|)$ berechnen müssen, es wurde aber $\sum_k (x_k)$ berechnet
 - Wurde nur entdeckt, weil ähnliche Läufe unerklärlich unterschiedliche Resultate ergaben
- Das **typischer Programmierfehler** wurde entdeckt, um die Seismotestfestigkeit von Reaktorgebäuden für Kernkraftwerke sicherzustellen
 - Laut Gesetzgebung müssen diese den stärksten je in der Region erlebten Beben stand halten
- 5 Kernkraftwerke mussten deshalb dauerhaft abgeschaltet werden:
 - in Maine, New York, Pennsylvania, Virginia, Virginia



- Kritische Frage hier:
 - Darf man sich für kritische Entscheidungen auf eine Simulation verlassen, die man nicht experimentell überprüfen kann?



Bild: http://de.wikipedia.org/wiki/Kernkraftwerk_Krümmel

Therac-25

- Die Software des Krebs-Strahlentherapie-Gerätes Therac-25 hatte zwei kritische Defekte
- Dadurch kam es bei Behandlungen mehrfach zu Überdosen
 - Mehrere Menschen starben daran
 - oder erlitten schwere Schäden
- Das trat insbesondere bei besonders kompetenten Bedienern auf
 - Sehr schnelle Eingaben brachten das Gerät zum Versagen
- Das Vorgängermodell Therac-20 hatte ungefähr die gleichen Softwaredefekte, blieb aber unfallfrei
 - Denn es besaß noch simple elektromechanische Sicherheitseinrichtungen



2. Hardware und technische Einrichtungen versagen oder gehen kaputt

- Auch die zuverlässigsten technischen Systeme versagen zumindest hin und wieder oder gehen kaputt
 - z.B. können selbst intakte Computer in der Tat irren, wenn kosmische Strahlung den Inhalt einer Speicherzelle ändert
 - Meistens sind die Ursachen aber viel banaler

Bsp: Telefonausfall New York 1991

- **Notstromgenerator** für eine Vermittlungsstelle sprang nicht an
- System lief 6 Stunden lang auf Notbatterie, bis diese leer war
- Die vorgesehenen Signale, um Personal zu Hilfe zu holen, sprangen nicht an
 - weil sie außer Betrieb genommen worden waren:
 - sie hatten zuvor mehrfach wegen Bauarbeiten falschen Alarm gegeben
- Hätten auch nicht zwingend geholfen, denn beide zuständigen Notfallbearbeiter waren abwesend
 - sie besuchten einen Kurs über Stromversorgungs-Notfälle
- **Resultat:**
 - ca. 5 Mio. ausgefallene Telefongespräche,
 - 4 Stunden Ausfall aller drei New Yorker Flughäfen



Bsp: Geldautomaten-Versagen 1993

- Am 13. März 1993 versagten ca. 5.000 Geldautomaten in allen Gegenden der USA
- Grund: Das **Dach** des zuständigen Rechenzentrums war eingestürzt
 - Grund: zu hohe Schneelast auf dem Dach
- Das Ersatz-Rechenzentrum war nicht verfügbar, weil es wegen des Terroristenangriffs auf das World Trade Center (einen Monat zuvor) bereits belegt war



3.a Menschen sind fehlbar

- insbesondere unter Stress
- Fast jedes noch so gut gestaltete System kann durch Bedienfehler in unerwünschte Zustände geraten
- Entscheidend ist dann, ob es zusätzliche Sicherungen gibt, die das höchstwahrscheinlich bemerken und zu reparieren erlauben

Bsp: Frontaler Zugcrash Berlin 1993

- Während Bauarbeiten zur Elektrifizierung der Strecke war Wochentags nur einspuriger Betrieb bei Wannsee
- Am Karfreitag konnten beide Spuren benutzt werden
 - Der **Fahrdienstleiter** beließ den Schalter versehentlich trotzdem auf "einspurig"
 - Der Computer setzte daraufhin korrekt das Signal auf "Halt"
 - Der **Fahrdienstleiter** hielt dies für ein Versagen des Computers, fragte aber nirgends zurück. Er stellte das Zusatzsignal (extra wegen der Bauarbeiten aufgestellt!) auf "Weiterfahren"
 - Er übersah einen entgegenkommenden, nicht fahrplanmäßigen Zug
- Es gab "nur" 3 Tote und 20 Verletzte
 - Denn beide Züge fuhren (wegen einspurig) nur Tempo 30



Bsp: Beinahe-Katastrophe Space Shuttle 1986

- Übermüdete **Techniker** interpretierten die Anzeige eines Ventilversagens falsch
- Sie ließen deshalb eine Sauerstoffleitung offen
 - Dadurch flossen 5 Minuten vor dem Start 68.000 Liter flüssiger Sauerstoff aus dem externen Tank ab
- Im Startfall hätte das Shuttle seine Umlaufbahn nicht erreichen können
- Aber: Der Countdown wurde bei -31 sec abgebrochen, nachdem ein Temperaturfühler durch den Abfluss unerlaubt niedrige Werte angezeigt hatte



3.b Menschen sind unvorsichtig

- Das Paradebeispiel hierfür ist der Untergang der Titanic
 - er kam nur zustande durch extremen Leichtsinn
 - dieser wurde möglich, weil das Schiff als unsinkbar galt:
 - 4 von 16 Abteilungen durften vollaufen, ohne dass das Schiff sinken würde. So ein Fall war noch nie vorgekommen
 - Das späte Ausweichen vor dem Eisberg führte zum Aufschlitzen von 5 Abteilungen
 - Man hatte keine Sicherheitsübungen gemacht und zuwenig Boote an Bord
- Oft ist die Wirkung des Leichtsinnns aber subtiler



Bsp: Osprey-Schwenkflügelflugzeug

- Die Konstruktion enthält eine computerunterstützte Steuerung
- Diese benutzt z.B. Rollraten-Sensoren
- Zur Ausfallsicherheit gibt es diese Sensoren dreifach
- Ein Prototyp stürzte ab, weil 2 der 3 Sensoren verpolt angeschlossen waren
 - Bei 2 weiteren Prototypen entdeckte man daraufhin je 1 verpolten Sensor
 - **Entwurfsfehler** (keine Warnung bei Sensorfehler)
 - Schlamperei der **Monteure**
 - Fehlender Hardware-schutz vor Verpolung



Bsp: Chernobyl 1986

- Aufgrund einer Kernschmelze im Reaktor Chernobyl 4 traten große Mengen Radioaktivität aus
 - Mehrere Tausend Tote, ca. 0,5 Millionen Verseuchte
- Der Unfall geschah während der Durchführung von Experimenten zum Thema "Wie weit reicht Notstrom zum Kühlen aus?"
 - Für die Experimente hatten die **Techniker** mehrere Sicherheitssysteme abgeschaltet: Notkühlsystem, Leistungsregelg., automatische Notabschaltung
- Die **Konstruktion des Reaktors** war inhärent gefährlich und schwer zu kontrollieren
 - z.B. Verwendung von Grafit (brennbar!) für die Moderatorstäbe



3.c Menschen sind oftmals ignorant

- Es fehlt ihnen an Wissen
 - selbst an entscheidenden Stellen
- ohne dass sie das merken

- Daraus resultieren sowohl Entwurfsfehler als auch Benutzungsfehler

Bsp: Abkühlautomatismus im Stahlwerk 1993

- Deutsches Stahlwerk benutzt Computersteuerung für den Abkühlprozess des frischen Stahls und zum Start der Weiterverarbeitung
- Die **Programmierer** hatten die Zeitmessung auf die Normalzeituhr der PTB abgebildet
 - PTP: Physikalisch-technische Bundesanstalt in Braunschweig
- Umschaltung auf Sommerzeit verkürzte die Kühlzeit um 1 Stunde
- Riesige Sachschäden am Stahlwerk
- 2 Ignoranz-Fehler:
 - Eigentlich kommt es auf die Stahl-Temperatur an, nicht auf Zeit
 - Uhrzeit ist nicht immer der korrekte Indikator für die abgelaufene Zeit



Bsp: U-Bahn ohne Fahrer 1993

- Eine U-Bahn-Tür klemmte und schloss nicht richtig.
- Der **Fahrer** stieg aus, um das zu beheben.
- Sobald die Tür schloss, fuhr der Zug los – ohne den Fahrer
- Der **Fahrer** hatte den Knopf zum Losfahren mit Klebeband in "gedrückt"-Stellung festgeklebt
 - Bahn stoppte automatisch und fuhr auf Knopfdruck automatisch los, sobald (aber erst wenn) die Türen wieder geschlossen waren
 - Fahrer hatte die Betriebsvorschrift verletzt, niemals die Führerkabine zu verlassen
- **Entwerfer** hatte die Faulheit und Achtlosigkeit des Fahrers versäumt vorherzusehen

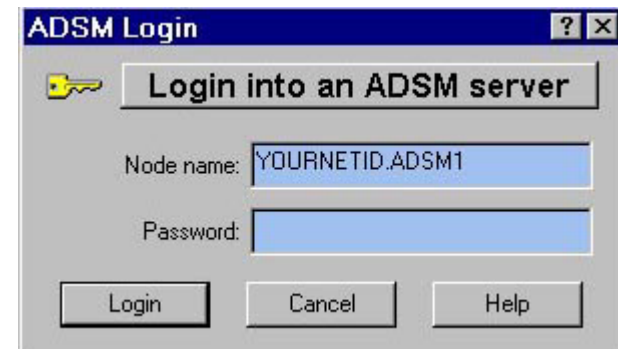


3.d Menschen sind manchmal böswillig

- Schlimmstenfalls (also fast immer) muss ein System nicht nur gegen zufällige und versehentliche Gefahren geschützt werden
- sondern zusätzlich gegen absichtliche Angriffe
- Dieser Schutz ist besonders schwierig,
 - weil die Möglichkeiten für solche Angriffe unübersehbar vielfältig sind

Bsp: Geld als Weihnachtsgeschenk

- Am 24.12.1987 überwies ein holländischer **Bankangestellter** sich selbst 8,4 Millionen Dollar und 6,7 Millionen Dollar auf ein Schweizer Konto
- Das Sicherheitssystem der Bank sah vor, dass Überweisungen von zwei Personen autorisiert werden müssen (4-Augen-Prinzip)
- Der Mann kannte jedoch das Passwort eines anderen Kollegen
- Die zweite Überweisung gelang wegen eines technischen Fehlers nicht
 - nur dadurch flog das Ganze am nächsten Tag auf



Bsp: Terry Dean Rogan

- US-Bürger Terry Dean Rogan verlor seine Brieftasche
 - samt Führerschein und Kreditkarten
 - der **Finder** nahm seine Identität an und verübte zwei Morde und zwei Raubüberfälle
 - Es wurde ein Haftbefehl auf Terry Dean Rogan ausgegeben
- Er wurde binnen 14 Monaten 5 mal von der Polizei festgenommen,
 - obwohl er seit dem ersten Mal beständig versuchte, die Datenbankeinträge korrigieren zu lassen
 - Ihm wurden vor Gericht \$55.000 Schadenersatz zugesprochen, zahlbar von der Polizei von Los Angeles



4. Auch unwahrscheinliche Ereignisse treten gelegentlich ein

- Manchmal funktionieren Systeme für lange Zeit ganz sicher, bis ein sehr seltenes Ereignis eintritt
- Deshalb kann Testen als Sicherheitsgarantie nicht ausreichend sein

Bsp: Kontoüberziehung Bank of New York

- In der Buchungssoftware lief unbemerkt ein **Zähler** über
 - Dadurch konnte die Bank einlaufende Gutschriften nicht verbuchen
 - Die Federal Reserve Bank buchte hingegen die Belastungen von Zahlungsausgängen normal weiter
- Es entstand bei der Bank ein Soll von 32 Milliarden Dollar
- Die BoNY musste sich für einen Tag 24 Milliarden Dollar leihen
 - Die Zinskosten dafür betragen 5 Millionen Dollar



In the 1990s The Bank of New York moved its headquarters to One Wall Street.

Bsp: McMuffin

- McDonalds hatte ein seit Jahren korrekt funktionierendes Zeiterfassungssystem
- Plötzlich liefen dessen Uhren weitaus zu schnell
 - Die Angestellten bekamen 2 bis 4 Stunden zu viel Arbeitszeit zugerechnet
 - und erhielten entsprechend höhere Lohnzahlungen
- Ursache waren die Zeitgeber in den neuen Toastern für das neue Produkt McMuffin
 - Diese erzeugten Impulse im Stromnetz, die die Uhren beeinflussten
 - Zugleich kam es zusätzlich zu Phantombestellungen in den Kassen, die das Inventar- und Kassensystem ebenfalls durcheinander brachten



5. Scheinbar unverbundene Ereignisse sind tatsächlich oftmals eng gekoppelt

- Auch wenn ein System entwurfsseitig scheinbar keine einzelnen Versagenspunkte enthält, kommen Versagen durch nur ein Ereignis vor
- Denn manchmal sind die mehreren nötigen Ereignisse gar nicht wirklich verschieden

Bsp: ARPAnet-Versagen 1986

- Das ARPAnet (direkter Vorläufer des Internet) war stets für Fehlertoleranz entworfen
- Deshalb war z.B. die Region Neuengland an den Rest des ARPAnet nicht nur über 1 Leitung, sondern über 7 getrennte Leitungen angebunden
- Alle diese 7 Leitungen wurden am 12.12.1986 gleichzeitig bei Baggerarbeiten durchtrennt
- Denn sie liefen alle über das selbe Glasfaserkabel
 - SEN 12(1), 1987



Ende der Beispiele

Die Rolle der Informatik

- Im Prinzip ist die Rolle von Computern und Informatik bei Sicherheitsproblemen keine besondere
- Aber Computer haben die Entwicklung höchst komplizierter und riskanter Systeme extrem verstärkt:
 - Größere solche Systeme als je zuvor
 - Stärkere Wechselwirkungen zwischen ihnen
 - Mehr und schnellere Veränderungen an ihnen
 - Manchmal ein naives Vertrauen in ihre Verlässlichkeit

Hierarchische Sicht von Unfällen

Zum Verstehen eines Unfalls und der konkret gewordenen Bedrohungen sollte man drei Ebenen unterscheiden:

- Mechanismen:
 - konkreter Hergang beim Unfall. Rein beschreibend.
- Bedingungen:
 - Zustand des Systems und seiner Umgebung bei Beginn des Unfalls (Zum Verstehen des Hergangs).
- Urgründe (*root causes*):
 - Allgemeine Bedingungen im Umfeld des Systems, die zu den konkreten Bedingungen bei Unfallbeginn geführt haben (Zum Vermeiden ähnlicher Unfälle in der Zukunft).
- Urgrund-Analyse wird leider selten gemacht, weil Sie fast immer einige Beteiligte beschämt.

Was man aus den Beispielen lernen kann

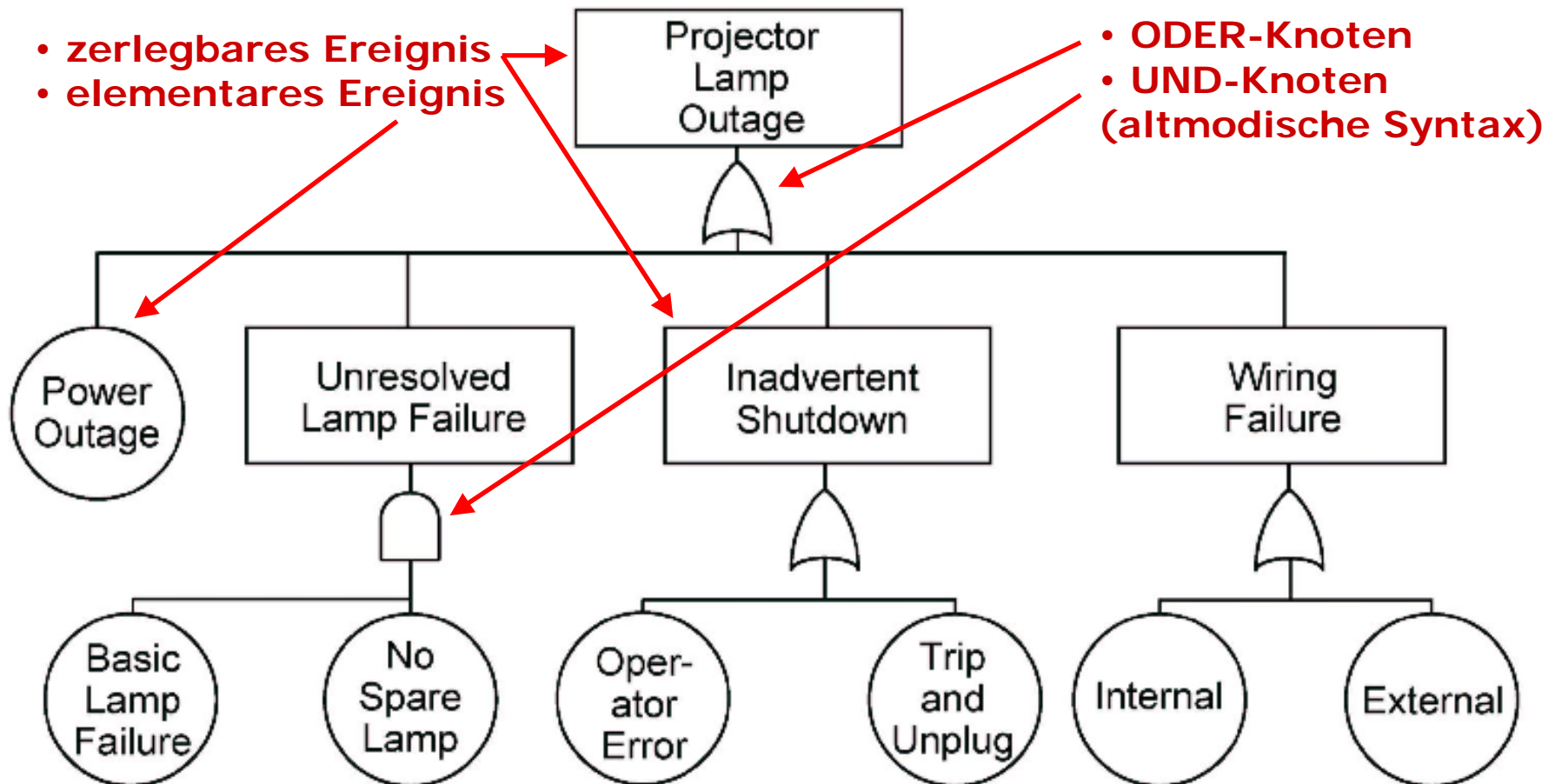
1. Vollständigkeit von Anforderungen ist kritisch
2. Menschliches Versagen ist allgegenwärtig
 - auch da, wo man es nicht dulden will
3. Technisches Versagen ist häufiger als uns lieb ist
 - und kann auch indirekt von Mensch, Tier oder Natur verursacht sein
4. Für schwere Unfälle gibt es meist mehr als eine Ursache
5. Wir lassen uns von Vorsichtsmaßnahmen gern einlullen
6. Je komplexer ein System wird, desto mehr Risiken hat es
 - leider steigern Gegenmaßnahmen gegen Risiken oft die Komplexität noch weiter...
 - Klarheit und Einfachheit sind hohe, aber schwierige Ziele

Vorgehen zum Bau risikoarmer Systeme

- Gefahrenbestimmung
 - Welche Gefahren gibt es?
- Risikoanalyse
 - Zu jeder Gefahr: Wie hoch ist das Risiko? Kann man es ganz umgehen? Lohnen sich Vorbeugung oder Abwehr?
- Entwurf
 - Gegenmaßnahmen erdenken, definieren und umsetzen. Vorrangfolge beachten (siehe unten).
- Entwurfssicherheitsprüfung
 - Ähneln Gefahrenbestimmung, aber viel konkreter. Nötigenfalls Entwurf nachbessern und Prüfung wiederholen.
- Bau
- Abnahmesicherheitsprüfung
 - Noch konkreter. Bau bewirkt stets Entwurfsänderungen.

Gefahrenbestimmung: z.B. Fehlerbaumanalyse

- Idee: Gefahrereignisse so lange hierarchisch in Bedingungen zerlegen, bis Gefahr verstanden ist



Risiken der Risikoanalyse

Es gibt nur drei Möglichkeiten, alle sind problematisch:

1. Risiken werden überschätzt

- Vermeidbare Kosten beim Systembau treten ein
 - Eine teilweise Niederlage

2. Risiken werden unterschätzt

- Vermeidbare Schäden beim Systembetrieb treten ein
 - Die klassische Niederlage

3. Risiken werden richtig eingeschätzt

- Gerade weil nun Unfälle vermieden werden, kann der Wert der Risikoanalyse nicht bewiesen werden
- Das kann die nächste Risikoanalyse beeinträchtigen
 - Ggf. eine indirekte Niederlage

Vorgehen: Grundregeln

- Baue Sicherheit von vornherein ein
 - Explizite nicht-funktionale Anforderung
- Betrachte das System als Ganzes, nicht seine Teile
 - Sicherheit ist eine Eigenschaft des Systems, nicht der Teile
 - Selbst wenn alle Teile versagensfrei sind, kann das System vielfältig unsicher sein
- Verlasse Dich nicht allein auf Erfahrungen und Standards
 - Jedes System ist anders. Analysiere es!
- Verwende qualitative und quantitative Methoden
 - Zahlen können in die Irre führen (falsche Prioritäten)
- Gestehe ein, dass Abwägungen nötig sind und Konflikte auftreten

Umgang mit Gefahren

Beachte die Vorrangfolge beim Umgang mit Gefahren:
(z.B.: Datenqualität: Gefahren durch Fehler in Datenbeständen)

- Gefahr eliminieren ("intrinsische Sicherheit")
 - z.B. eine Information, die nicht gespeichert wurde, kann auch nicht falsch sein
- Gefahr reduzieren
 - z.B. falsche Dateneingabe durch Plausibilitätsprüfungen unwahrscheinlicher machen
- Gefahr beherrschen (passiv, aktiv)
 - z.B. passiv: Schadenersatzhöhe begrenzen
 - z.B. aktiv: Datenqualität regelmäßig überwachen
- Schaden verringern
 - z.B. potentiell falsche Daten nur für mäßig wichtige Zwecke verwenden

Probleme beim Entwurf sicherer Systeme

- Sicherheit kostet zusätzliches Geld
 - "Amortisation" ist unklar
- Risikobetrachtungen haben negativen Geruch
 - sind deshalb schwer zu vermitteln
- Sicherheitsmaßnahmen sind nicht sexy
 - Nicht so imponierend wie Funktionalität.
Schwer zu verkaufen.
- Oft wird das Vorhandensein von Sicherheitsmaßnahmen mit vollständiger Sicherheit gleichgesetzt
 - "Titanic-Effekt"
- Oder es herrscht die Haltung vor
"Man bekommt es ja sowieso nicht sicher hin"

Quellen für viele der Beispiele in diesem Foliensatz:

- Peter G. Neumann: *Computer-related Risks*, Addison-Wesley 1995
- ACM Sigsoft Software Engineering Notes (SEN)
 - <http://www.acm.org>
- The RISKS Forum
 - <news:comp.risks>
 - <http://www.risks.org>

Peter G. Neumann



Zusammenfassung

- Fast alle Informatiksysteme bergen Risiken
 - möglicherweise bis hin zu Leib und Leben
- Die Risiken entstammen vielen verschiedenen Quellen
 - die meisten davon sind nicht technischer Natur
- Risiken lassen sich nur beherrschen, wenn das System im Ganzen betrachtet wird
 - und nicht nur die Software oder nur das technische System oder nur der Mensch
- Unfälle entstehen fast immer durch das Zusammenwirken mehrerer Gefahren
 - Ihre Vermeidung verlangt also entsprechend vernetztes Denken in allen Abschnitten des Lebens eines Systems

Danke!

Vorgehen: Gefahrenbestimmung

- Nutze historische Daten
 - Sicherheitserfahrungen, Problem- und Unfallberichte
- Nutze öffentliche Checklisten, Standards, Vorgehensbeschreibungen
- Für physische Gefahren: Untersuche Energiequellen, Energieflüsse, gefährliche Materialien
- Untersuche die Mensch-Maschine-Schnittstelle
 - Ist sie verständlich? Für jeden? Ist sie robust?
- Untersuche den Prozess des Systembaus (insbes. SW)
 - Solide Ingenieurpraktiken?
- Untersuche alle normalen/besonderen Betriebszustände
 - insbes. Übergänge (Start, Stopp, Zusammenbruch, Reparatur, etc.)

- Untersuche denkbare ungewöhnliche Umstände des Betriebs
 - Umwelteinflüsse (Wetter, Tiere, Erdbeben, ...)
 - Infrastrukturversagen: Strom, Kühlung, ...
 - Hardwareversagen (endgültig, vorübergehend)
 - Missbenutzung: Fehlinstallation, Fehlbedienung, Gebrauch zu anderem Zweck, Missachtung von Meldungen, vorsätzlicher Missbrauch
- Beachte die langfristige Perspektive
 - Änderungen am System; Änderungen am technischen, organisatorischen oder sozialen Umfeld
- Gehe schrittweise durch Gesamtprozess des Systems
 - Was kann schief gehen? Wie kann man es vermeiden?
 - Was ist zu tun, wenn der schlimmste Fall eintritt?



Relevant bei Ersatz elektromechan. Sicherheitsmechanismen durch SW-Mechanismen:

- Mythos: *"Computer sind billiger als analoge oder elektromechanische Sicherheitssysteme"*
 - Stimmt nur bei hohen Stückzahlen, weil die Entwicklung sicherer SW sehr teuer ist
- Mythos: *"Software ist leicht zu ändern"*
 - Aber sehr schwer korrekt zu ändern
- Mythos: *"Computer sind zuverlässiger als analoge oder elektromechanische Systeme"*
 - Hardwareseitig ja, aber es kommen häufig Softwaredefekte hinzu, die die Sicherheit beeinträchtigen
 - z.B. Space Shuttle: Höchste denkbare Qualitätsansprüche bei der Entwicklung. Extrem konservativer Ansatz. 1980–95 wurden dennoch 16 gefährliche Defekte aufgedeckt.

- Mythos: *"Erhöhung der SW-Zuverlässigkeit steigert die Sicherheit"*
 - Stimmt oft, aber nicht immer, weil viele Sicherheitsprobleme in Anforderungen oder Entwurf begründet sind und nicht von mangelnder Korrektheit herrühren
 - Stimmt tendenziell nicht, weil ein Bewusstsein hoher SW-Zuverlässigkeit die Vorsicht untergraben kann
- Mythos: *"SW-Wiederverwendung steigert die Sicherheit"*
 - kann stimmen wegen höherer Zuverlässigkeit (siehe oben)
 - kann aber auch falsch sein: Ein sicherheitskritisches System wird so entworfen, dass die Sicherheitsargumente möglichst einleuchtend werden.
Das ist bei Wiederverwendung oft nicht mehr möglich.

- Risiken müssen ernst genommen werden
 - Wo keine ausdrücklichen gesetzlichen Vorschriften bestehen (und selbst dort), ist das leider oft nicht der Fall
- Stillschweigende Annahmen müssen sichtbar gemacht werden
 - damit sie überprüft werden können
 - und dann müssen sie überprüft werden!
- Sorgfältige Implementierung ist unverzichtbar
 - denn selbst gute Fehlertoleranzmaßnahmen sind begrenzt
- Ganzheitliches Denken ist unverzichtbar
 - Nicht nur die SW, sondern das ganze System betrachten
 - Nicht nur das System, sondern auch sein Umfeld betrachten