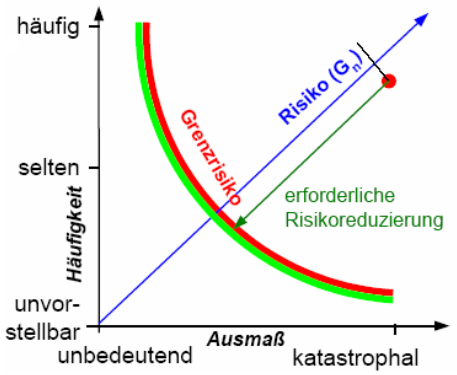
Qualität Eingebetteter Systeme

Prof. Dr.-Ing. Peter Liggesmeyer
Lehrstuhl Software Engineering: Dependability
TU Kaiserslautern
Direktor Fraunhofer Institut für Experimentelles Software Engineering (IESE),
Kaiserslautern

- Qualität Eingebetteter Systeme: Beispiel Sicherheit
- Die Bedeutung von Standards
- Die "Erfahrung"
- Der "Common Sense"

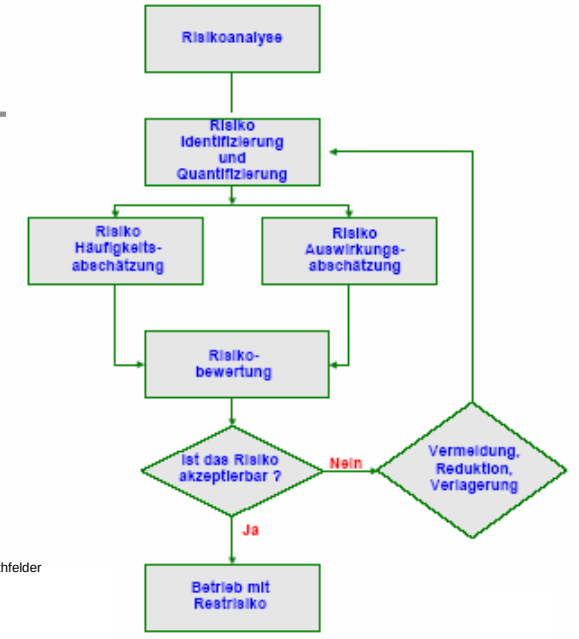
Qualität Eingebetteter Systeme: Beispiel Sicherheit Definition Risiko

- Definition Risiko: R = H * S
  - H zu erwartende Häufigkeit des Eintritts eines Ereignisses, das zu einem bestimmten Schaden führt
  - S das bei Ereigniseintritt zu erwartende Schadenausmaß



Sicherheit zwischen Theorie und Praxis
Übersicht Risikobegriffe

Für den Umgang mit Risiken sind deren Identifikation, Bewertung und Akzeptanz wichtige Schritte. Im folgenden wird die Risikoakzeptanz betrachtet.



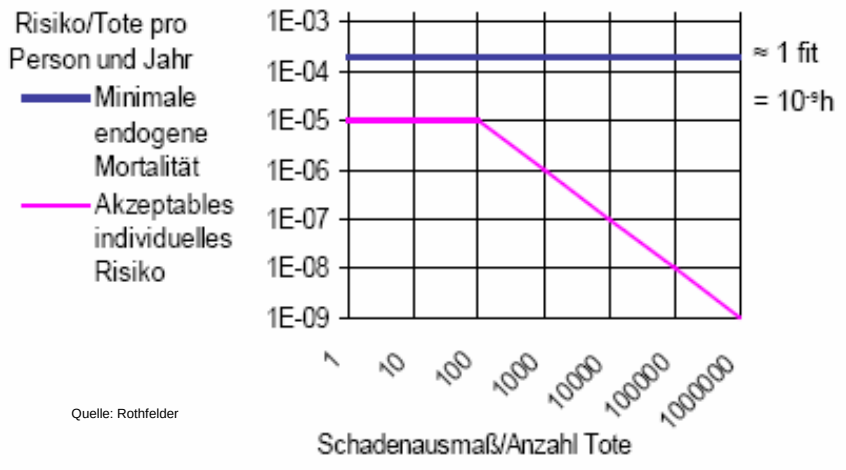
Quelle: Rothfelder

Sicherheit zwischen Theorie und Praxis
Risikoakzeptanz-Verfahren

seda
software engineering dependability

- Einige wichtige Verfahren zur quantitativen Risikoakzeptanz sind:
  - MEM (Minimale Endogene Mortalität)
  - GAMAB (Globalement Au Moins Aussi Bon); GAME (Globalement Aussi Équivalent)
  - ALARP (As Low as Reasonably Practicable)

Sicherheit zwischen Theorie und Praxis: Risikoakzeptanz
Minimale Endogene Mortalität (MEM)

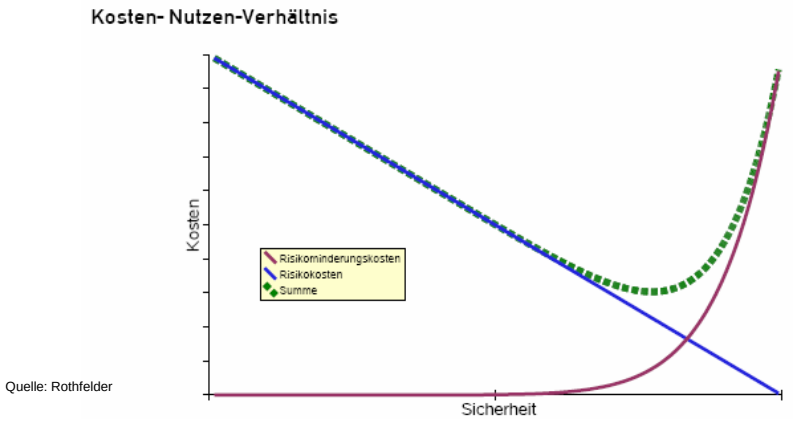


Quelle: Rothfelder

Sicherheit zwischen Theorie und Praxis: Risikoakzeptanz
Einflussfaktoren für Risikoakzeptanz

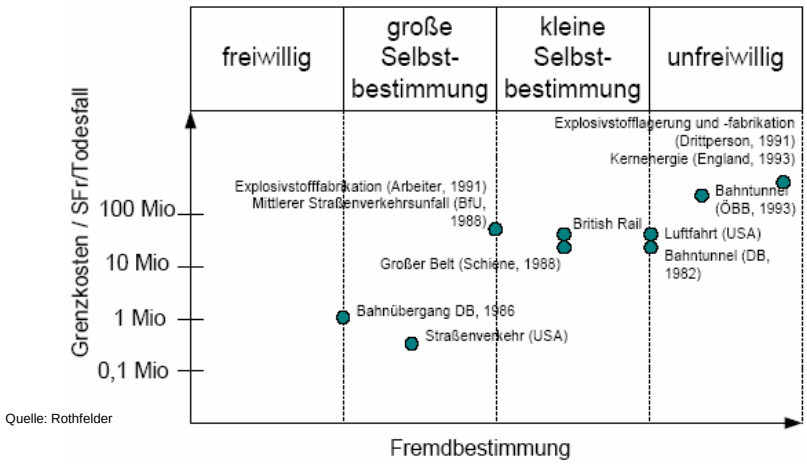
- Welche Risiken akzeptabel sind, ist ebenfalls subjektiv und unter anderem von folgenden Faktoren abhängig:
  - Wie hoch ist der Nutzen? – Große Strecken in der Luftfahrt: Bezieht man die Gefährdung auf die zurückgelegte Strecke oder auf die im Flugzeug verbrachte Zeit?
  - Wer ist gefährdet? – Raumfahrer, Kranke, Bahnpassagiere, Betriebspersonal, unbeteiligte Dritte
  - Wie hoch ist der Grad der Selbstbestimmung? – Autofahrer vs. Aufzug
  - Wie viele Menschen befinden sich in Gefahr? – Auto vs. Kernkraftwerk
    - Schadenausmaß: Tod? Verletzte?

Sicherheit zwischen Theorie und Praxis: Risikoakzeptanz Wie sicher ist sicher genug?



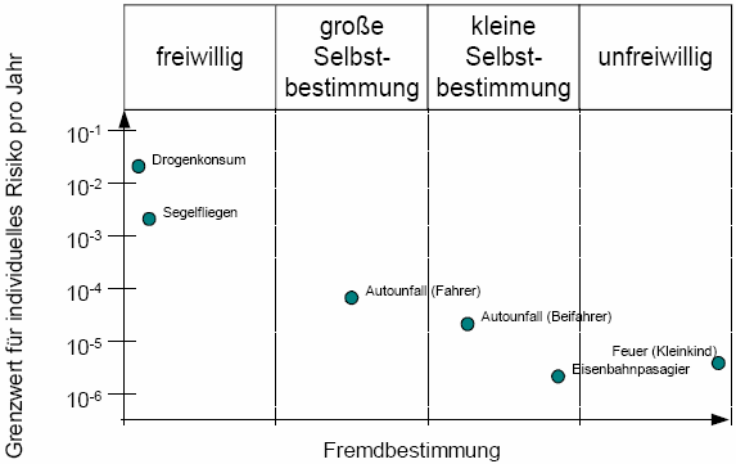
Quelle: Rothfelder

Sicherheit zwischen Theorie und Praxis Grenzkosten vs. Fremdbestimmung



Quelle: Rothfelder

Sicherheit zwischen Theorie und Praxis
Grenzwert für individuelles Risiko pro Jahr vs. Fremdbestimmung



Quelle: Rothfelder

Qualität Eingebetteter Systeme
Kleine Ursache – große Wirkung

```
...
declare
  vertical_veloc_sensor: float;
  horizontal_veloc_sensor: float;
  vertical_veloc_bias: integer;
  horizontal_veloc_bias: integer;
  ...
begin
  declare
    pragma suppress(numeric_error, horizontal_veloc_bias);
  begin
    sensor_get(vertical_veloc_sensor);
    sensor_get(horizontal_veloc_sensor);
    vertical_veloc_bias := integer(vertical_veloc_sensor);
    horizontal_veloc_bias := integer(horizontal_veloc_sensor);
    ...
  exception
    when numeric_error => calculate_vertical_veloc();
    when others => use_irs1();
  end;
end irs2;
```

Qualität Eingebetteter Systeme
Qualitätssicherung

seda software engineering dependability

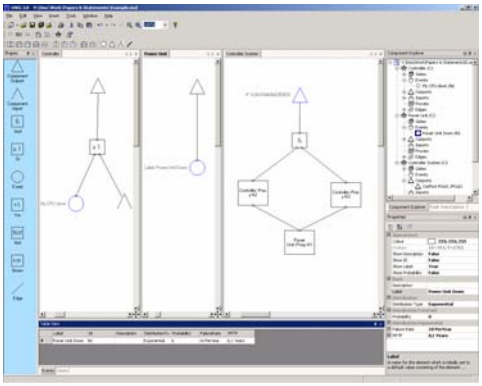
- Testen:
  - Viele alte Techniken, die für viele Entwickler immer wieder neu sind
  - Wenige grundsätzlich neue Theorien in den letzten 15-20 Jahren
  - Leistungsfähigkeit der Techniken theoretisch unklar (praktisch erst recht)
  - Theoretisch klar ist die Schwäche, dass Fehler “durchschlüpfen” können
  - Wird von vielen universitären Forschern als unergiebiges Arbeitsfeld gesehen => wenig Forschung, wenig Lehre dazu => siehe erster Punkt
- Formale Techniken:
  - Theoretisch ziemlich leistungsfähig, praktisch eingeschränkt
  - In der Praxis nur sehr punktuell verwendbar
  - Theoretisch auch nur für eng definierte Anwendungsgebiete
- Inspektionen und Reviews
- Automatische Statische Analysen

Qualität Eingebetteter Systeme
Modellbasierte Sicherheits- und Zuverlässigkeitsanalysetechniken

seda software engineering dependability

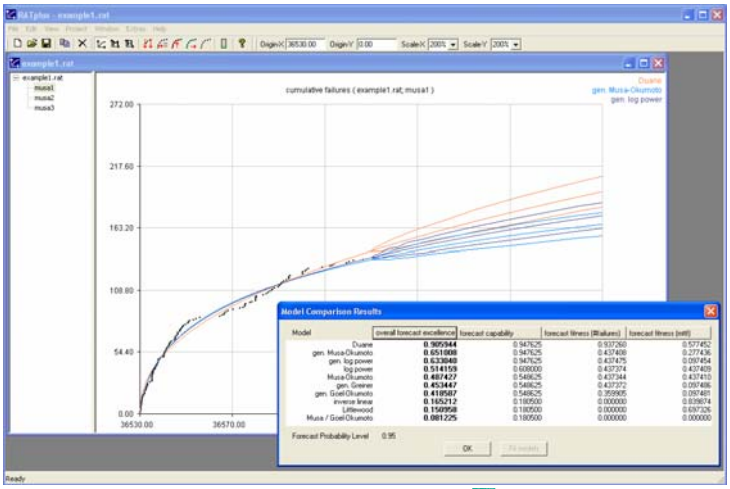
Z.B.:

- Hazard-Analysen (FHA, PHA,...)
- Event Tree Analyse
- FME(C)A (Failure Modes Effects (and Criticality) Analysis) (DIN 25448, IEC 812)
- Zuverlässigkeitsblockdiagramme (IEC 61078)
- Fehlerbaumanalyse (DIN 25424, IEC 61025)
- Markov-Analysen (IEC 61165)



Qualität Eingebetteter Systeme
Zuverlässigkeitswachstumsmodellierung

- Wie zuverlässig ist mein System jetzt?
- Wie zuverlässig wird es zum geplanten Freigabetermin sein?
- Wie viele Fehler werden bis dahin aufgetreten sein?
- ...



Standards
Bedeutung von Standards

- Standards entscheiden im Zweifelsfall, welche Verfahrensweisen, Methoden und Techniken als Stand der Technik bzw. als Stand von Wissenschaft und Technik zu betrachten sind.
- Standards und Normen:
  - Keine Rechtsnorm , aber antizipierte Sachverständigengutachten
- Gesetzliche Regelungen:
  - z.B. Produkthaftungsgesetz, Schadensersatz nach BGB
- Europäische Richtlinien
  - Haben den Charakter eines Gesetzes, weil sie von den Mitgliedsstaaten zwingend in nationales Recht umzusetzen sind
- Verordnungen
  - werden meistens von Behörden – der Exekutive – erlassen und sind in der Regel verbindlich

Standards
Bedeutung von Standards

seda software engineering dependability

- Normung ist in Deutschland die planmäßige, durch die interessierten Kreise gemeinschaftlich durchgeführte Vereinheitlichung von materiellen und immateriellen Gegenständen zum Nutzen der Allgemeinheit. Deutsche Normen werden in einem privatrechtlichen Verein durch interessierte Kreise erstellt (z. B. DIN Deutsches Institut für Normung e.V., Verband Deutscher Elektrotechniker (VDE) e.V.). Standards und Normen sind keine Rechtsnormen. Sie sind – im Unterschied zu Gesetzen – nicht rechtsverbindlich, aber sie können als antizipierte Sachverständigengutachten verstanden werden. Durch Einhaltung der jeweils relevanten Normen kann ein Hersteller sicherstellen, dass der Stand der Technik erreicht ist, und er damit seine Sorgfaltspflicht erfüllt hat.

Standards und Softwareprüfung

seda software engineering dependability

- Alle Standards betonen die Bedeutung des funktionorientierten Testens
- Viele Standards enthalten explizite Regelungen der einzusetzenden Techniken
  - DIN EN 50128 erzwingt die Nutzung von Kodierkonventionen für C
  - RTCA DO-178 B regelt explizit die einzusetzenden strukturorientierten Testtechniken in Abhängigkeit der Kritikalität

Erfahrung
Notwendige, minimale Anforderungen an Tests

- Absolut notwendig entspr. aller maßgeblichen Standards:
  - Funktionsorientierte Testplanung für alle Testphasen
  - Reproduzierbarkeit von Testergebnissen => automatische Regressionstests nach Software-Modifikationen
- Weitgehender Konsens:
  - Ergänzende strukturorientierte Abdeckung (mindestens Zweigüberdeckungstest)
  - In kritischen Anwendungsbereichen – z. B. der Avionik – werden darüber hinaus durch Standards explizit gründlichere strukturorientierte Tests gefordert => RTCA DO 178 B
  - Zusätzlich Leistungs- und Stresstests

Erfahrung
Software-Qualität

	/Fenton, Ohlsson 00/	/Basili, et al. 96/	/Cartwright, Shepperd 00/	/Basili, Perricone 84/	/Abreu, Melo 96/
Wenige Module enthalten die Mehrzahl der Fehler.	++	++	(+)	++	/
Wenige Module erzeugen die meisten Ausfälle.	++	/	/	/	/
Viele Fehler im Modultest bedeuten viele Fehler im Systemtest.	+	/	/	/	/
Viele Fehler im Test bedeuten viele Ausfälle im Feld.	--	/	/	/	/
Fehlerdichten korrespondierender Phasen sind über Releases hinweg konstant.	+	/	/	/	/
Umfangsmaße sind geeignet zur Fehlerprognose.	+	/	+	-	/

++: starke Bestätigung; +: schwache Bestätigung; 0: keine Aussage; -: schwache Ablehnung; -- starke Ablehnung; /: nicht evaluiert; ?: unklar

Erfahrung Software-Qualität

<table>
<tr><th></th><th>/Fenton, Ohlsson 00/</th><th>/Basili, et al. 96/</th><th>/Cartwright, Shepperd 00/</th><th>/Basili, Perricone 84/</th><th>/Abreu, Melo 96/</th></tr>
<tr><td rowspan="7">Code-Komplexitätsmaße sind geeignete Mittel zur Fehler-prognose.</td><td rowspan="7">besser als Umfangs-maße:
-</td><td>WMC: +</td><td>WMC: /</td><td rowspan="7">besser als Umfangs-maße:
-</td><td>MHF: +</td></tr>
<tr><td>DIT: ++</td><td>DIT: ++</td><td>AHF: 0</td></tr>
<tr><td>RFC: ++</td><td>RFC: /</td><td>MIF: +</td></tr>
<tr><td>NOC: ?</td><td>NOC: ?</td><td>AIF: (+)</td></tr>
<tr><td>CBO: ++</td><td>CBO: /</td><td>POF: +</td></tr>
<tr><td>LCOM: 0</td><td>LCOM: /</td><td>COF: ++</td></tr>
</table>

- Objektorientierte Maße:
  - WMC (*Weighted Methods per Class*)
  - DIT (*Depth of Inheritance Tree*)
  - NOC (*Number Of Children*)
  - CBO (*Coupling Between Object-classes*)
  - RFC (*Response For a Class*)
  - LCOM (*Lack of Cohesion on Methods*)
  - MHF: *Method Hiding Factor*
  - AHF: *Attribute Hiding Factor*
  - MIF: *Method Inheritance Factor*
  - AIF: *Attribute Inheritance Factor*
  - POF: *Polymorphism Factor*
  - COF: *Coupling Factor*

Common Sense

- Wir haben eine beachtliche Diskrepanz zwischen Theorie und Praxis bei der Qualitätssicherung eingebetteter Systeme
- Standards unterstreichen bestimmte Praktiken unter Nichtbeachtung der mäßigen Theorie
- Eine Aufarbeitung der theoretischen Seite ist nicht in Sicht
- Die praktische Bedeutung bestimmter Ansätze (z.B. modellbasierte Verfahren) wird eher zunehmen
- Man muss heute zwingend die potentiell relevanten Standards kennen
- Es gibt einige interessante empirische Erkenntnisse, die man nicht ignorieren sollte => viel Unsinn aus der Vergangenheit ist heute widerlegt

Literatur

- /Abreu, Melo 96/
  Abreu F., Melo W., *Evaluating the Impact of Object-Oriented Design on Software Quality*, Proc. Metrics '96, pp. 90 - 99
- /Basili et al. 96/
  Basili V., Briand L.C., Melo W.L., *A Validation of Object-Oriented Design Metrics as Quality Indicators*, IEEE Transactions on Software Engineering, Vol. 22, No. 10, October 1996, pp. 751-761
- /Basili, Perricone 84/
  Basili V.R., Perricone B.T., *Software Errors and Complexity: An Empirical Investigation*, Communications of the ACM, Vol. 27, No. 1, January 1984, pp. 42-52
- /Cartwright, Shepperd 00/
  Cartwright M., Shepperd M., *An Empirical Investigation of an Object-Oriented Software System*, IEEE Transactions on Software Engineering, Vol. 26, No. 8, August 2000, pp. 768-796
- /Fenton, Ohlsson 00/
  Fenton N., Ohlsson N., *Quantitative Analysis of Faults and Failures in a Complex Software System*, IEEE Transactions on Software Engineering, Vol. 26, No. 8, August 2000, pp. 797-814