Proseminar:
IT-Kennzahlen und Softwaremetriken

Grundlagen des Messens

Tobias Riasanow

Gliederung

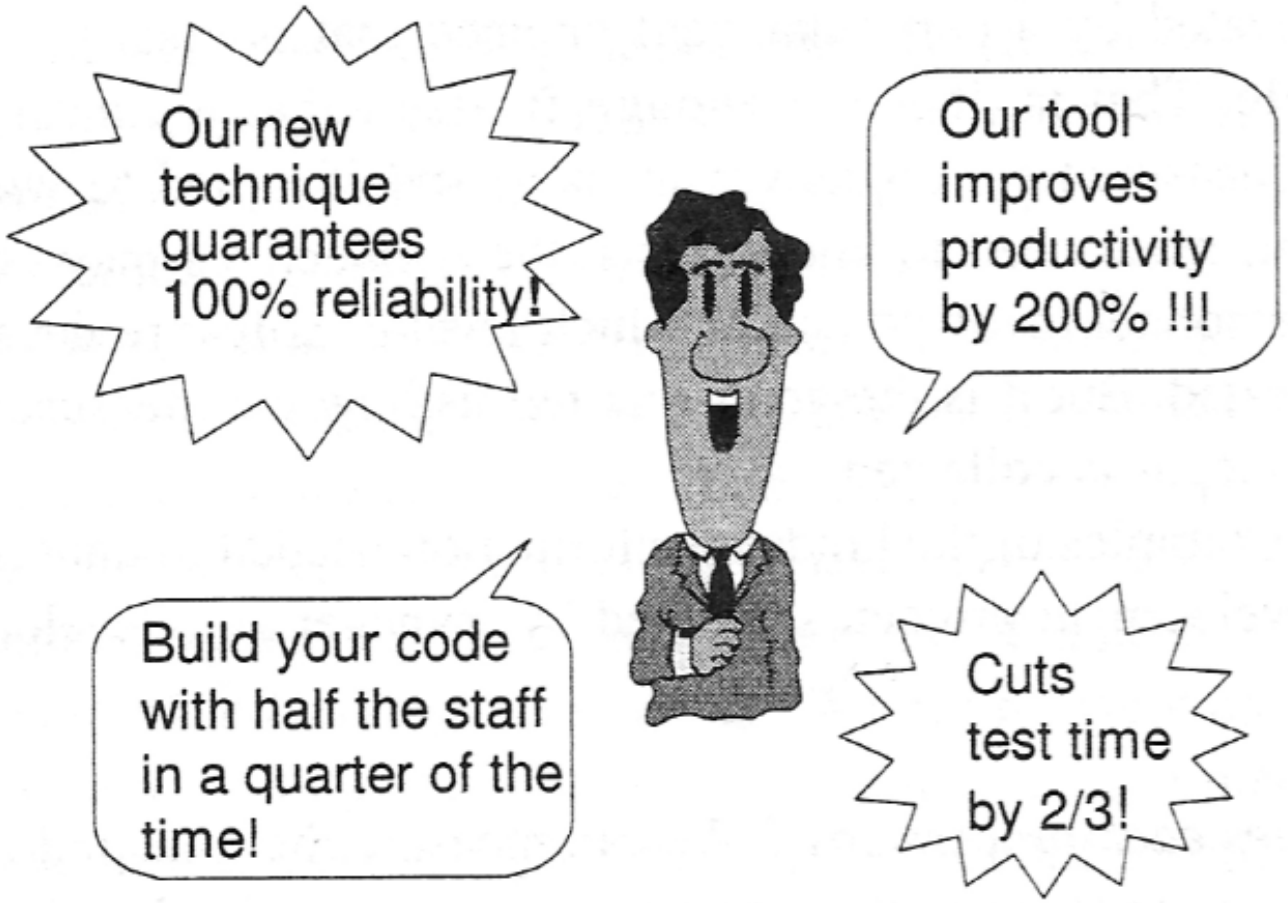
sebis

1. Messen: Was ist das und warum machen wir es?

    1.1 Warum misst man?
    1.2 Was ist eine Messung?
    1.3 Übertragung auf Software Engineering

3. Grundlagen des Messens

    2.1 Representational Theory of Measurement
    2.2 Empirische Zusammenhänge
    2.3 Mapping
    2.4 korrektes Messen
    2.5 Indirektes Messen
    2.6 Prognosen
    2.7 Skalenniveau
    2.8 Simpson Paradoxon


Our new technique guarantees 100% reliability!
Our tool improves productivity by 200% !!!
Build your code with half the staff in a quarter of the time!
Cuts test time by 2/3!

1.1 Warum misst man? [1]

sebis

Beispiel ***Software-Engineering (SE)***:

Ziele: zeitliche Einordnung der Fertigstellung, ausreichendes Budget? etc.

Messcharakteristiken von Software sind z.B.:

Qualität des Designs, geeigneter Code, Vollständigkeit, Anforderungsgerechtigkeit

Daraus entsteht vor allem großer **Kundennutzen**

➔ **Ohne Messungen kann Technologie nicht funktionieren!!!**

➔ Durch Messen werden Konzepte sichtbar, verständlich und kontrollierbar

1.1 Warum misst man? [2]

Generell heißt das:

Measurement is a common and necessary practise for understanding, controlling and improving our environment.
[Fenton, Pfleeger]
„Was nicht messbar ist, wird messbar gemacht!“
[Galileo Galilei (1564-1642)]

Beispiele:

- Wirtschaft
- Medizin
- Technik

➔Jeder misst und vergleicht auch im alltäglichen Leben

1.2 Was ist eine Messung? [1]

***Measurement** is the process by which **numbers** or **symbols** are assigned to attributes of **entities** in the real world in such a way as to describe them according to **clearly defined rules**.*
[Fenton, Pfleeger]

➔ Man misst nicht Dinge oder Attribute, sondern Attribute von Dingen

➔ diese abstrakte Darstellung von Entitäten hilft dabei die menschliche Sichtweise darzustellen und zu konkretisieren

Dennoch :

Measurement is a process whose definition is far from clear-cut
[Fenton, Pfleeger]

1.2 Was ist eine Messung? [2]

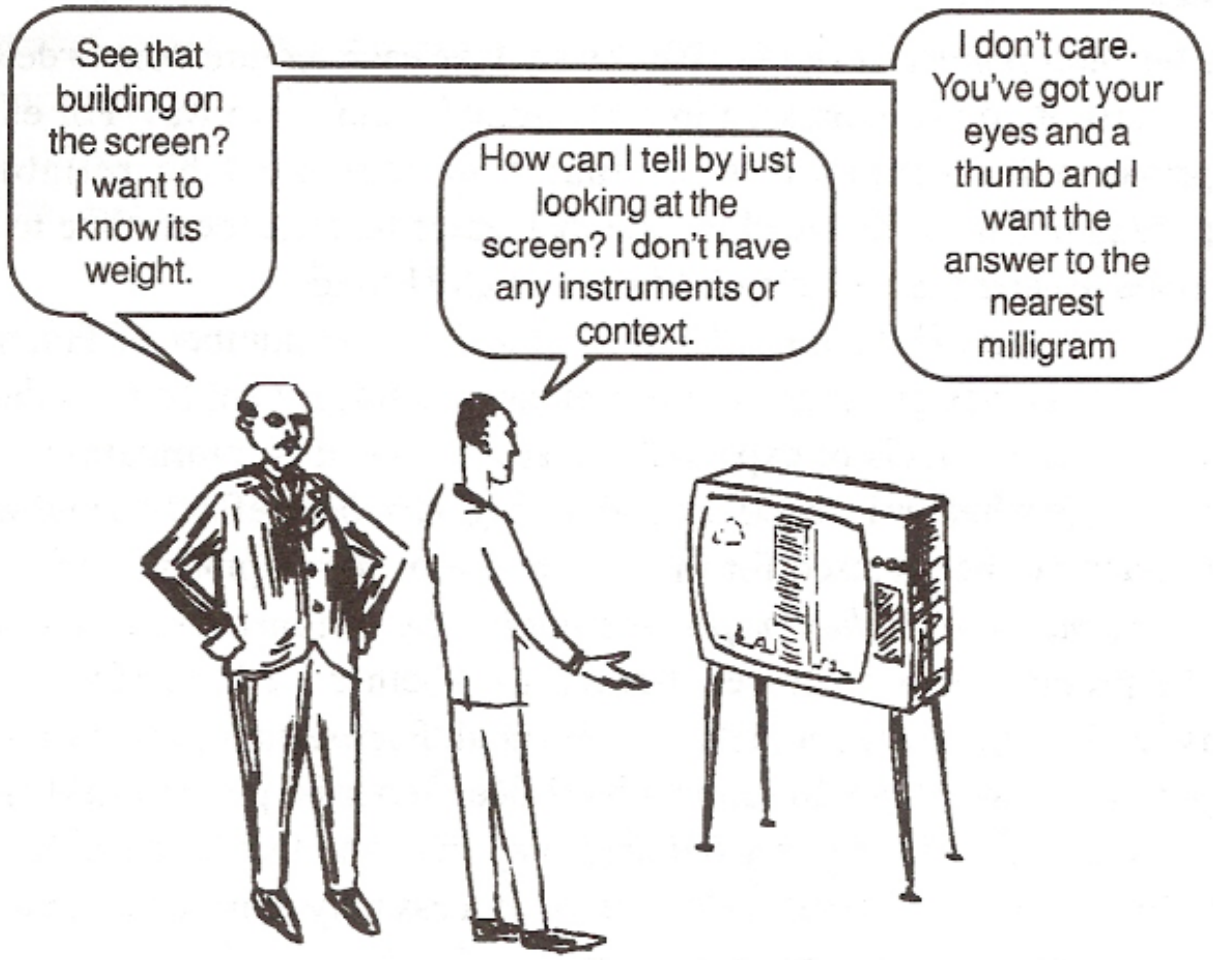


Figure 1.2: Software measurement – resource estimation

1.2 Was ist eine Messung? [3]

Daraus ergeben sich folgende **Grundsatzfragen**:

1) ***Genauigkeit einer Messung*** beruht sowohl auf dem Messinstrument, als auch auf der Definition des Messvorgangs!!

2) Dennoch gibt es immer eine ***potenzielle Fehlerquelle***!!
   ➔ welche Fehlerquote ist akzeptabel?

3) Welcher ***Maßstab*** ist für meine Messung korrekt?

4) Auf welche Weise kann man die gewonnen Ergebnisse ***analysieren/ beurteilen/ vergleichen***?

1.2 Was ist eine Messung? [4]

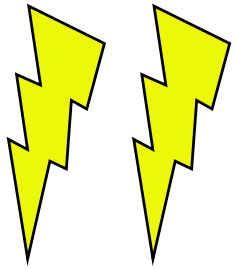
sebis

(beispielhafte) Fragen zur Verdeutlichung:

- die Farbe ist ein Attribut des Raums. Ist „blau“ dann ein Messwert in einem Raum mit blauen Wänden?
- Wie misst man Intelligenz? Bzw. misst eine IQ-Test adäquat die Intelligenz eines Menschen?
- Die Größe eines Menschen kann in Meter/ Zentimeter gemessen werden. Auch in Meilen oder Kilometer?
- Dürfen beim Messen die Schuhe anbehalten werden? In welcher Position misst man?
- A ist doppelt so groß ist wie B, kann man dann auch behaupten gestern war es doppelt so warm wie heute?

1.3 Übertragung auf Software Engineering [1]

„Messen“ ist momentan Luxus im Bereich des SE



Das liegt für die meisten Entwicklungsprojekte an:

1) Festlegen von messbaren Zielen

„projects without clear goals will not archieve their goals clearly“ [Gilb‘s Principle of Fuzzy Targets]

2) Verstehen und Quantifizieren von Kosten d. Komponenten

3) Bewertung und Prognostizierung der Softwarequalität

4) Neue (revolutionäre) Entwicklungstechnologien werden zuvor hinsichtlich Effektivität und Effizienz nicht genügend überprüft

➔ **unregelmäßige, widersprüchliche und unvollständige Messungen**

1.3 Übertragung auf Software Engineering [2]

Messen ist nicht nur für problematische Projekte notwendig!

> ***„You can neither predict nor control what you cannot measure“*** [DeMarco‘s Regel]

➔ Jede Messung wird mit einem bestimmten Ziel/Nutzen ausgeführt

<u>Messen ist wichtig für 3 Basisaktivitäten:</u>

1) **Verstehen** ➔ Sichtbarkeit

2) **Kontrolle** ➔ Vorraussagung

3) **Verbesserung** ➔ Prävention

➔ Dennoch sind die Möglichkeiten oft begrenzt

1.4 Zusammenfassung

- andere Ingenieurswissenschaften gehen mit gutem Beispiel voran
- „Software Measurement“ besteht aus vielen Teilgebieten
- Ziele für Messungen müssen spezifisch und klar formuliert sein
- beim Messen sollte man „mutig“ vorgehen

➔ Bis jetzt noch geringes Verständnis von **Softwareattributen**

➔ keine guten **Tools zum Messen** vorhanden

➔ die Übertragung **bewährter Prozeduren** auf Software gestaltet sich schwierig

2.1 The Representational Theory of Measurement

sebis

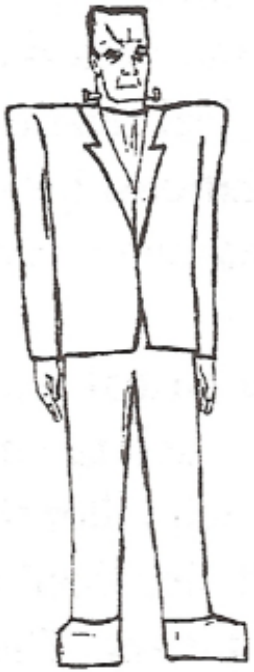
Representation condition:

Asserts that a measurement mapping M must map entities into numbers and empirical relations into numerical relations in such a way that the empirical relations preserve and are preserved by the numerical relations.

- Formalisiert die Welt
- Daten aus den Messungen sollen Attribute von Objekten repräsentieren, mit denen Manipulationen durchgeführt werden können
- Dinge werden aber durch den Vergleich mit Anderen verstanden und nicht wenn man ihnen Nummern gibt

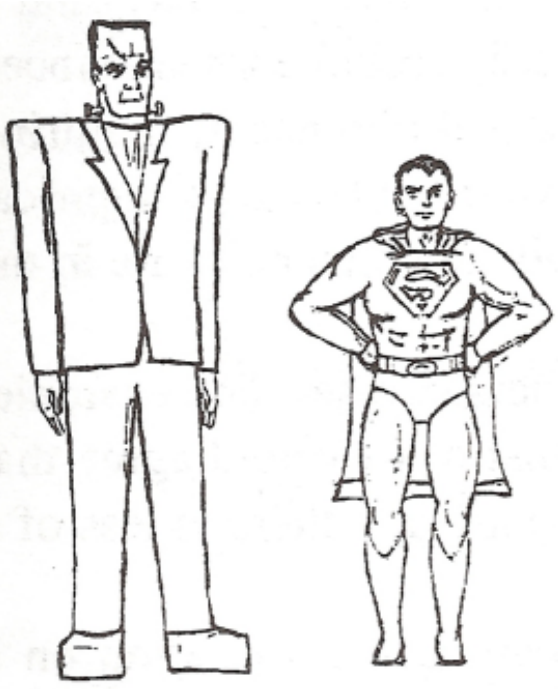
➔ ***Representational Theory of Measurement***

2.2 Empirische Relationen [1]



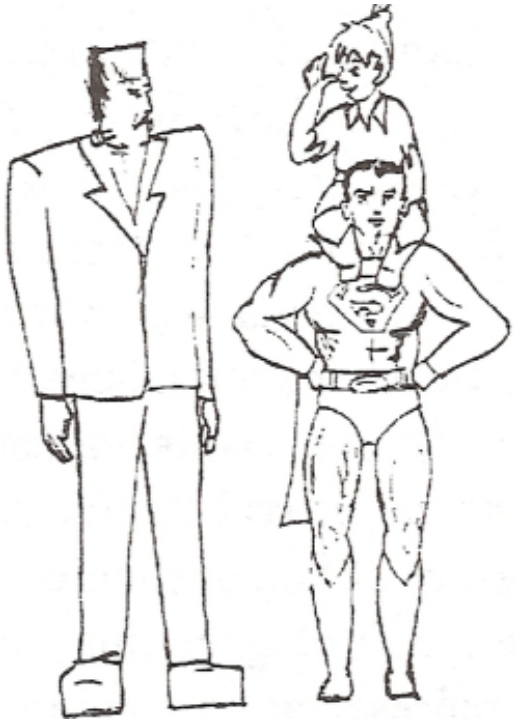
unäre Relation

A ist **groß**



binäre Relation

A ist **größer als** B



ternäre Relation

B+C sind **größer als** A

➔ „größer als“ ist eine **empirische Relation** für Körpergrößen

2.2 Empirische Relationen [2]

Table 2.1: Sampling 100 users to express preferences among products A, B, C, and D

	A	B	C	D
A	—	80	10	80
B	20	—	5	50
C	90	95	—	96
D	20	50	4	—

More functionality

	A	B	C	D
A	—	45	50	44
B	55	—	52	50
C	50	48	—	51
D	54	50	49	—

More user-friendly

- **Funktionalität**:
  keine klare Aussage ob B oder D besser ist möglich

- **Benutzerfreundlichkeit**:
  hier lassen sich gar keine sinnvollen empirischen Relationen feststellen

2.2 Empirische Relationen [3]

oft sind keine objektiven Aussagen möglich, da es unterschiedliche Vorstellungen von deinem Attribut geben kann

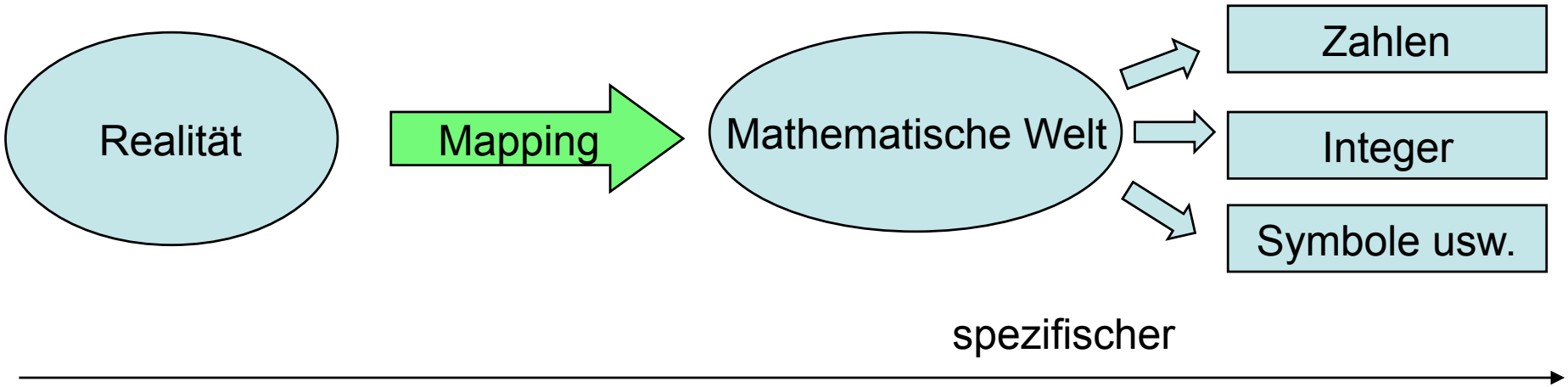
➔ zuerst können nur **subjektive Aussagen** getroffen und gesammelt werden

Likert scale
Give the respondent a statement with which to agree or disagree. For example: This software program is reliable.

Strongly Agree	Agree	Neither agree nor disagree	Disagree	Strongly Disagree

➔ Diese sind aber nun die **Basis für zukünftige empirische Relationen**, da so formales Messen durch Charakterisierung einer Grundübereinstimmung möglich wird!

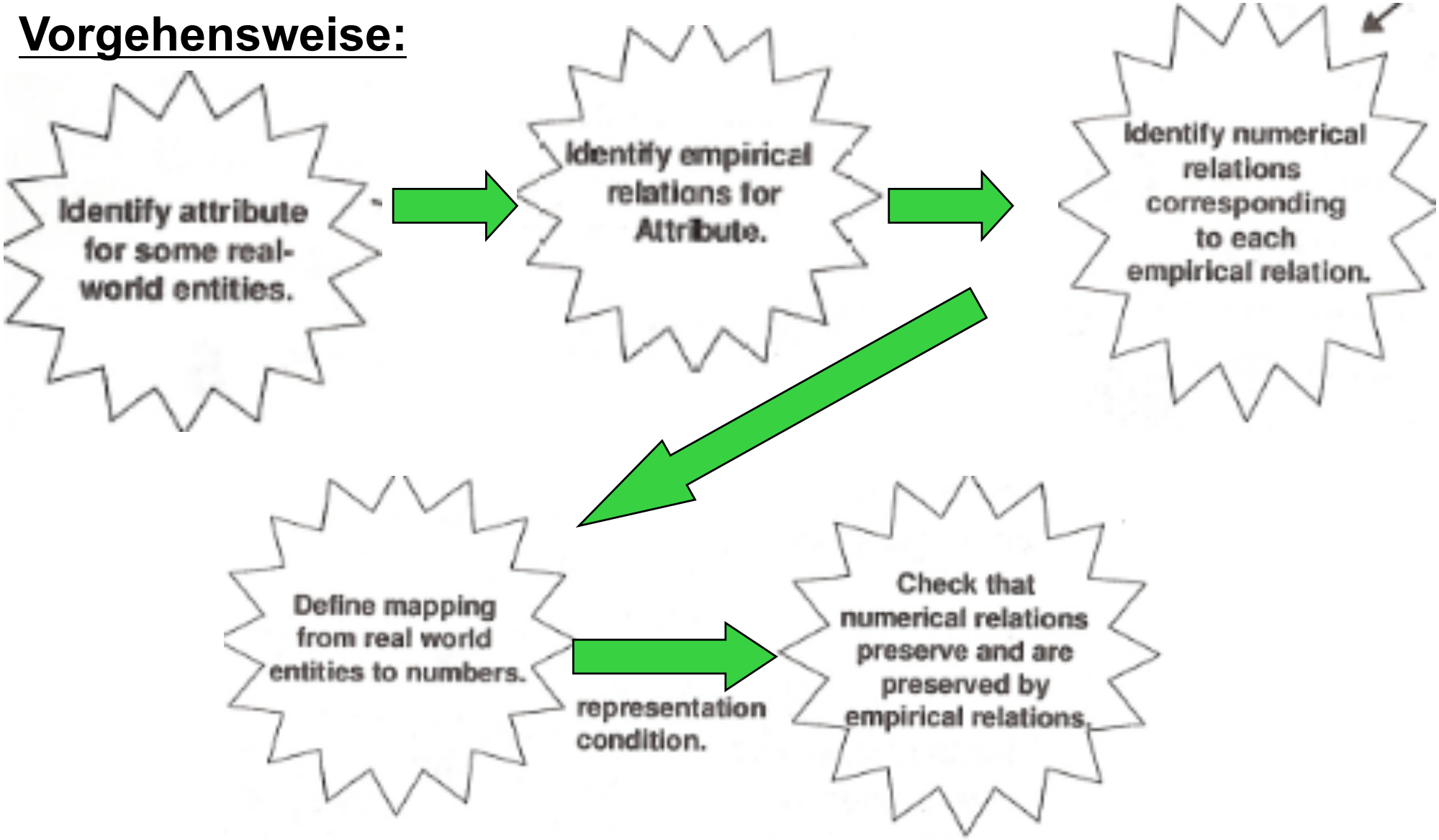
2.3 Mapping [1]



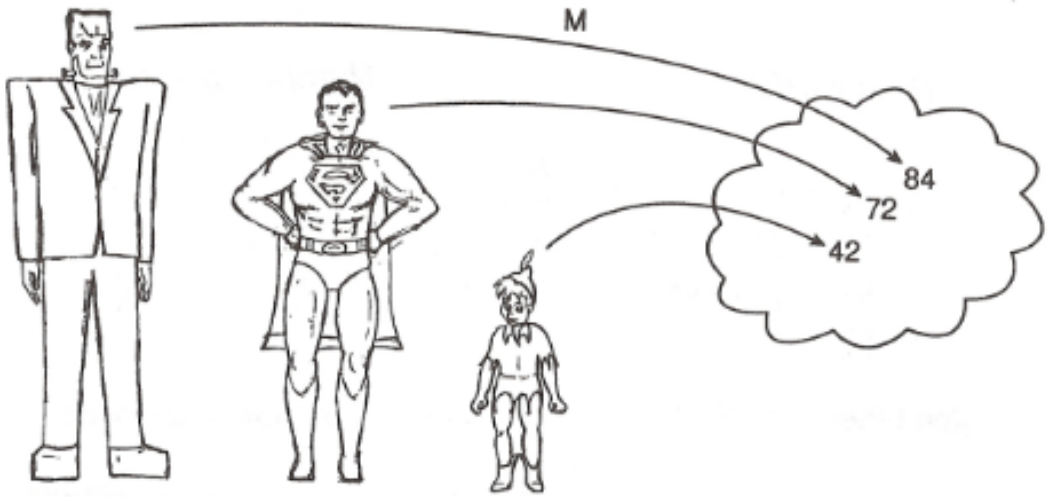
Erweiterung der Definition [von Pfleeger, Fenton]:

Measurement is a mapping from the empirical world to the formal, relational world. Consequently, a measure is the number or symbol assigned to an entity by this mapping in order to characterize an attribute.

Vorgehensweise:



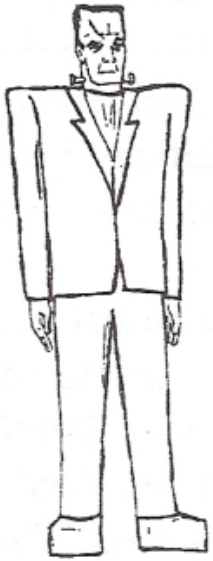
2.3 Mapping [3]



Für jedes A, dass größer als B ist muss **M(A) > M(B)** gelten &
M(A) > M(B) nur wenn A tatsächlich größer als B ist!

- jede Messung, die die Darstellungsbedingung erfüllt ist eine gültige Messung!
- je besser das System der empirischen Relationen, desto häufiger gibt es gültige Messungen
- aber: mehr Bedingungen ➔ mehr muss bei einer Messung beachtet werden

2.3 Mapping [4] - Beispiele

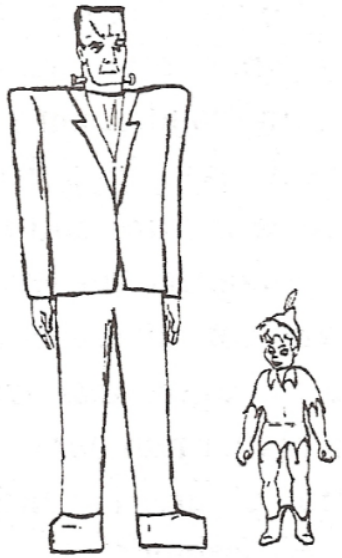


ist groß: x > 70

➔M(x) > 70

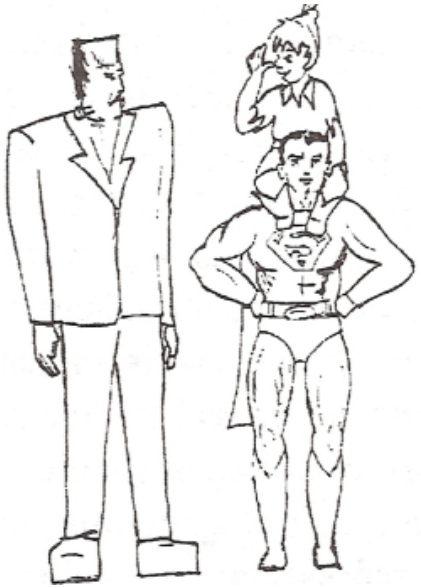
nicht groß: x < 70

➔ M(x) < 70



viel größer als: x > y + 15

➔ M(x) > M(y) + 15



0,7(y) + 0,8(z) > x

➔ 0,7 M(y) + 0,8 M(z) > M(x)

2.4 korrektes Messen [1]

2.4 korrektes Messen [2]

sebis

Grundsatzfragen:

- Wie viel muss man über Attribute wissen um sie messen zu können?
- Woher wissen wir ob wir das richtige Attribut gemessen haben?
- Welche sinnvollen Aussagen kann man nun treffen?
- Welche aussagekräftigen Operationen können durchgeführt werden?

Ziele:

➔ Gleichmäßiges & widerspruchsfreies Messen

➔ Liefert Basis zur Interpretation der Daten

➔ Dienen zum Niederschreiben des Anfangsverständnisses, das dann durch weitere Softwareanalysen erweitert wird

2.5 Indirektes Messen

Indirekte Messung:
kann nur durch Referenz auf ein anderes Attribut gemessen werden!

Beispiel **Softwareentwicklung**:

Programmer productivity	$\frac{\text{LOC produced}}{\text{person months of effort}}$
Module defect density	$\frac{\text{number of defects}}{\text{module size}}$
Defect detection efficiency	$\frac{\text{number of defects detected}}{\text{total number of defects}}$

2.6 Prognosen

Bsp.: Prognose der Kosten einer Fahrt von München nach Berlin

1) mathematisches Modell

2) Parameter für die Prognose definieren

3) Prozedur zur Interpretation

a = Strecke zwischen München/ Berlin in km

b = Preis von 1 Liter Sprit

c = Durchschnittstrecke, die mit 1 Liter gefahren werden kann

➔ **Kosten der Fahrt = (a * b) / c**

2.7 Skalenniveau [1]

Es gibt viele verschiedene Arten des Mapping ➔ verschiedene Messskalen

3 Fragen bezüglich Repräsentationen und Skalen:

1) Wie bestimmt man wann ein numerisches Relationssystem einem anderem vorzuziehen ist?

2) Woher wissen wir ob ein bestimmtes empirisches Relationssystem eine Repräsentation in einem numerischen Relationssystem hat? ➔ ***Repräsentationsproblem***

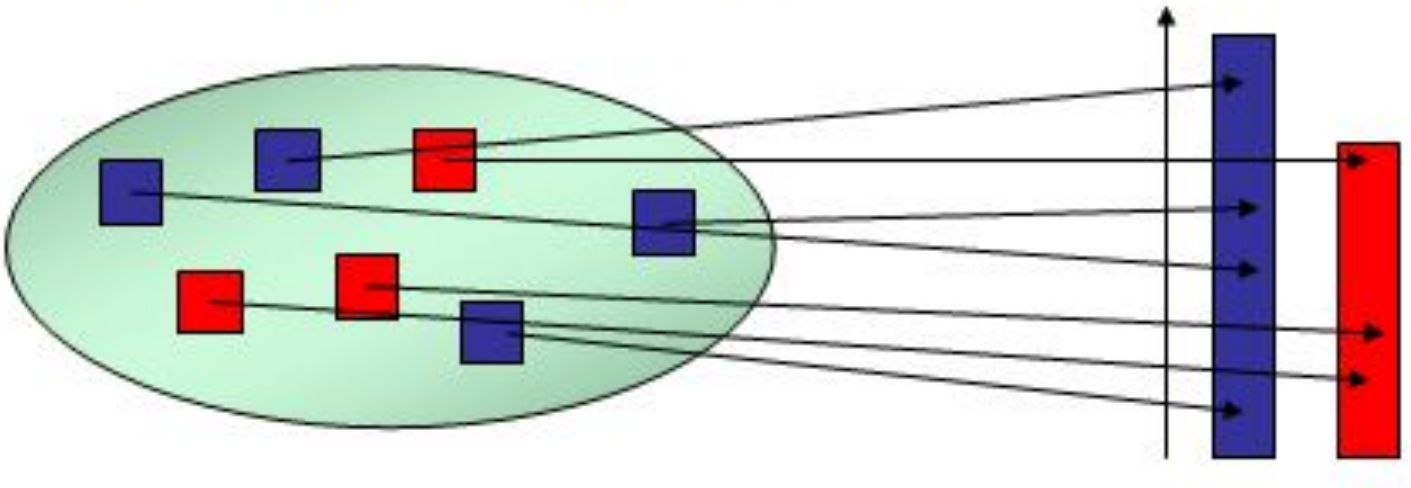
3) Was machen wir, wenn wir unterschiedliche mögliche Repräsentationen (in mehreren Skalen) im gleichen numerischen Relationssystem haben? ➔ ***Eindeutigkeitsproblem***

Mächtigkeit ↑ ➔ Repräsentationsmöglichkeiten ↓ ➔ differenzierte Messskala

2.7 Skalenniveau [2]

1) Nominalskala

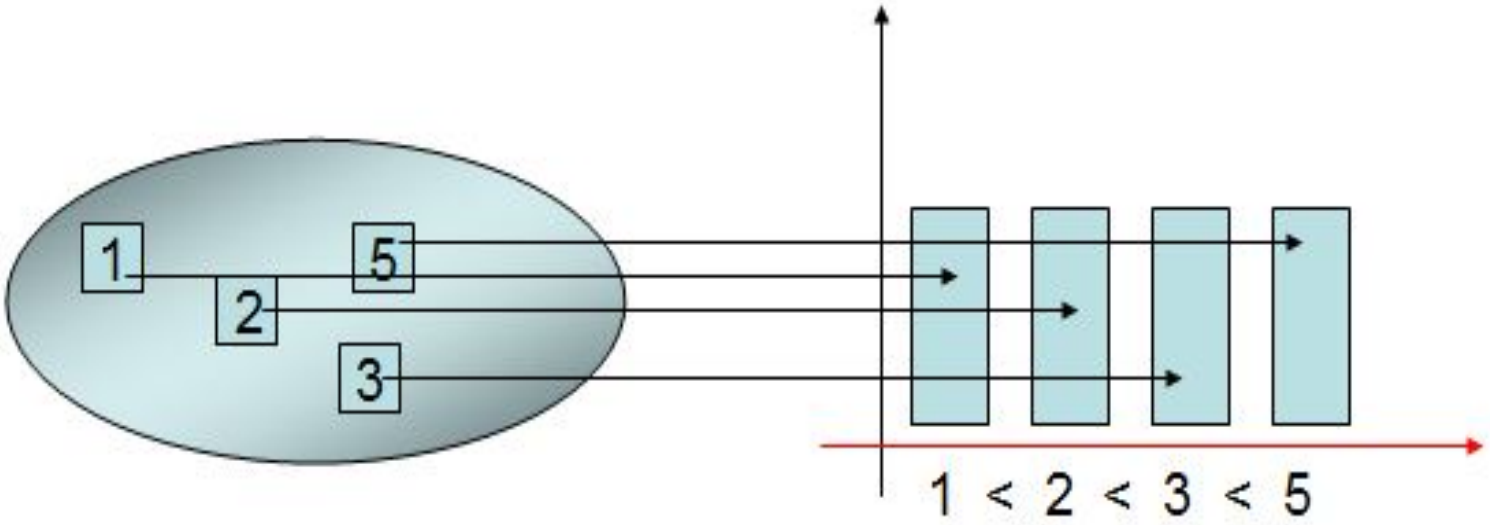
- das empirische Relationssystem besteht nur aus **verschiedenen Klassen**, die nicht geordnet sind
- Jede Nummerierung oder symbolische Darstellung der Klassen ist eine akzeptable Messung. Diese sagen aber nichts über Größe etc aus.
- Jede **1-zu-1 Abbildung von M zu M'** ist erlaubt
- Bsp.: Geschlecht (Mann/ Frau)
- Nur =/≠ Operationen möglich



2.7 Skalenniveau [3]

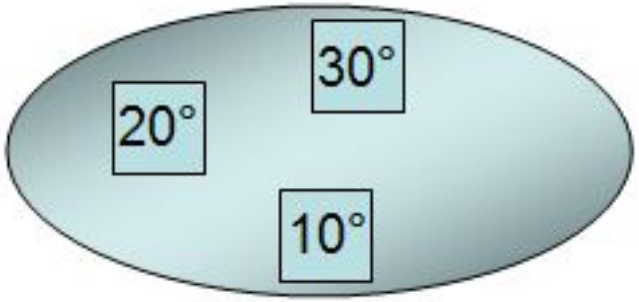
2) Ordinalskala

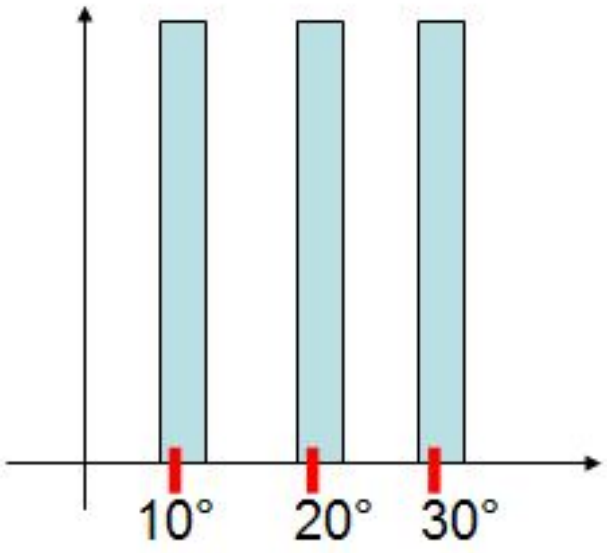
- = Nominalskala mit geordneter **Reihenfolge**
- erlaubte Transformationen: alle monotonen Abbildungen von M zu M‘ mit **M(x) > M(y)** , daraus folgt automatisch, dass **M‘(x) > M‘(y)** ist
- Bsp.: Schulnoten („sehr gut“ bis „ungenügend“)
- nun sind auch noch </> Operationen möglich



3) Intervallskala

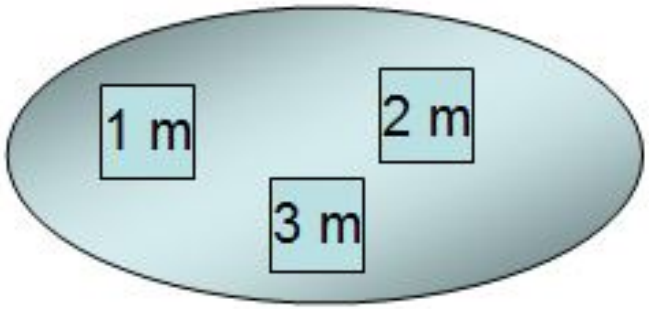
- gibt zusätzlich noch die **Intervallgröße** zwischen den Klassen an
- zeigt Differenzen auf, aber kein Verhältnis
- nur Affintransformationen erlaubt, dh. **M‘ = aM + b (a>0)**
- Bsp.: Zeitskala, Temperatur (in Celsius, Fahrenheit)
- nun sind auch +/- Operationen möglich

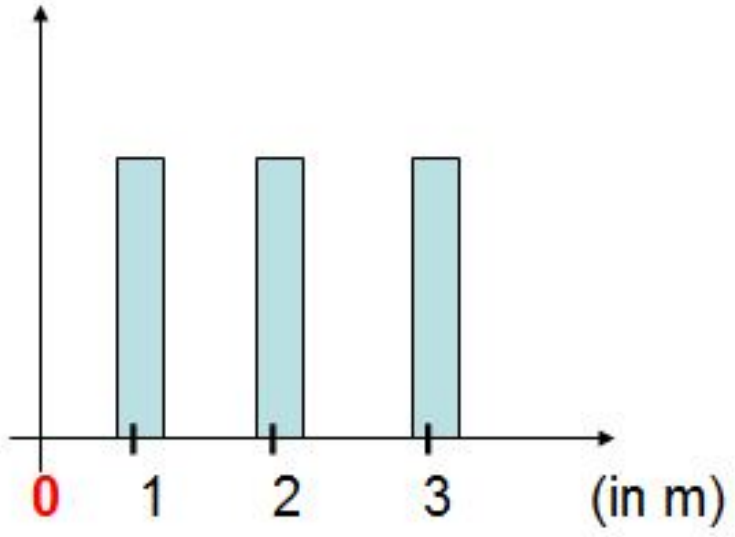




4) Verhältnisskala

- gibt nicht nur Reihenfolge und Intervalle an, sondern auch das **Verhältnis**
- besitzt einen Nullpunkt, der als Bezugspunkt der Verhältnisse fungiert und bei dem die Abbildung beginnt
- alle arithmetischen Operationen sind nun erlaubt
- Bsp.: Temperatur (gemessen in Calvin), Körpergröße, Zeitintervalle
- nur noch Verhältnistransformationen erlaubt: **M‘ = aM (a>0)**





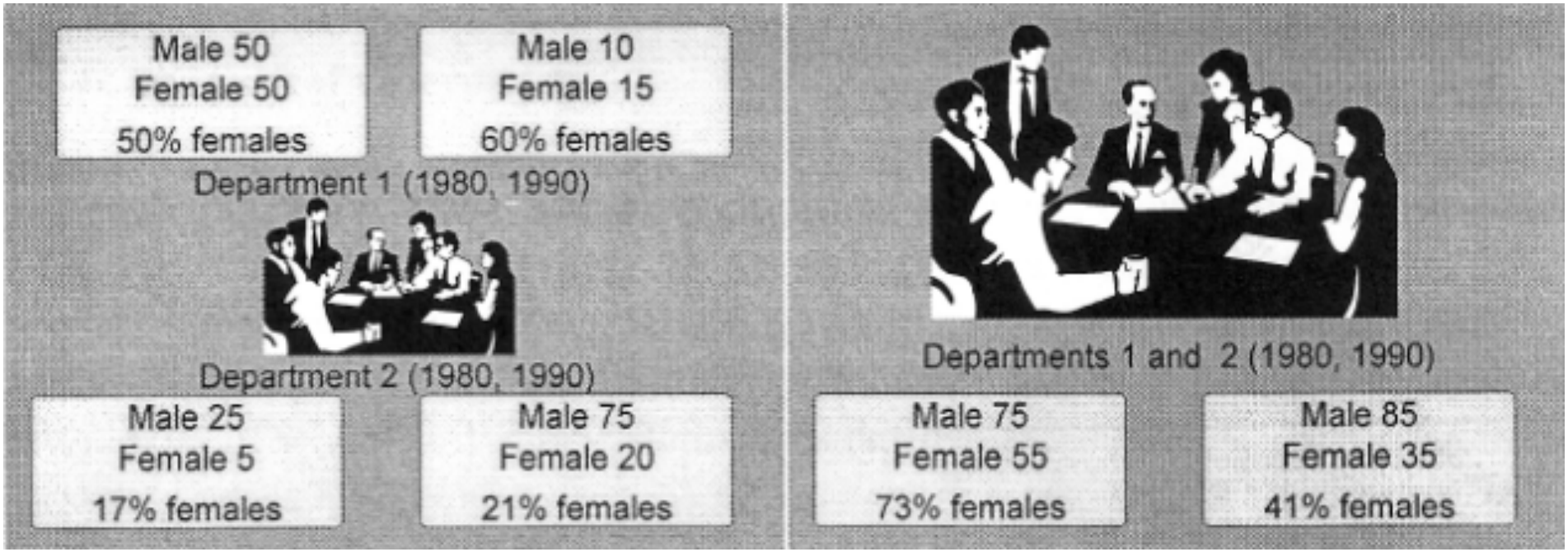
5) Absolute Skala

- Die Messung für eine absolute Skala wird einfach durch das Zählen der Anzahl der Elemente im Entitätsset gemacht
- Das Attribut hat immer die Form „Häufigkeit des Ereignisses x in der Entität“
- deswegen gibt es nur eine mögliche Messabbildung nämlich die tatsächliche Anzahl:

$$M' = M$$

- alle arithmetischen Operationen sind möglich
- Bsp.: LOC für die Codezeilen eines Programms

2.8 Simpson Paradoxon



- separat betrachtet nimmt der weibliche Anteil jeder Abteilung zu, obwohl er im gesamten abnimmt ➔ **Paradoxon**

- Betrachtungsweise/ -art einer „Messung“ beachten!

Vielen Dank

für die

Aufmerksamkeit!