Software-Qualität

Kapitel 4

Messen von Software-Qualität

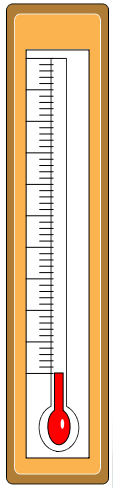
Inhalt

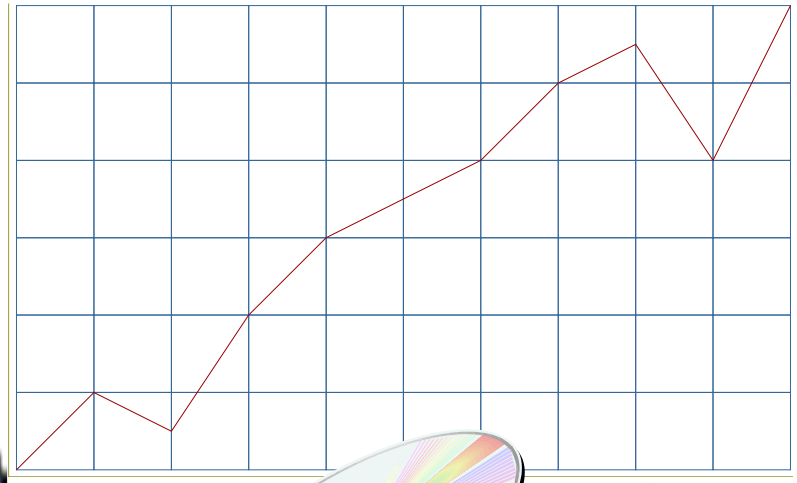
Software-Metriken

Metriken nach Maß: GQM

Prof. Dr. Kurt Schneider

Vage, konkret, messbar





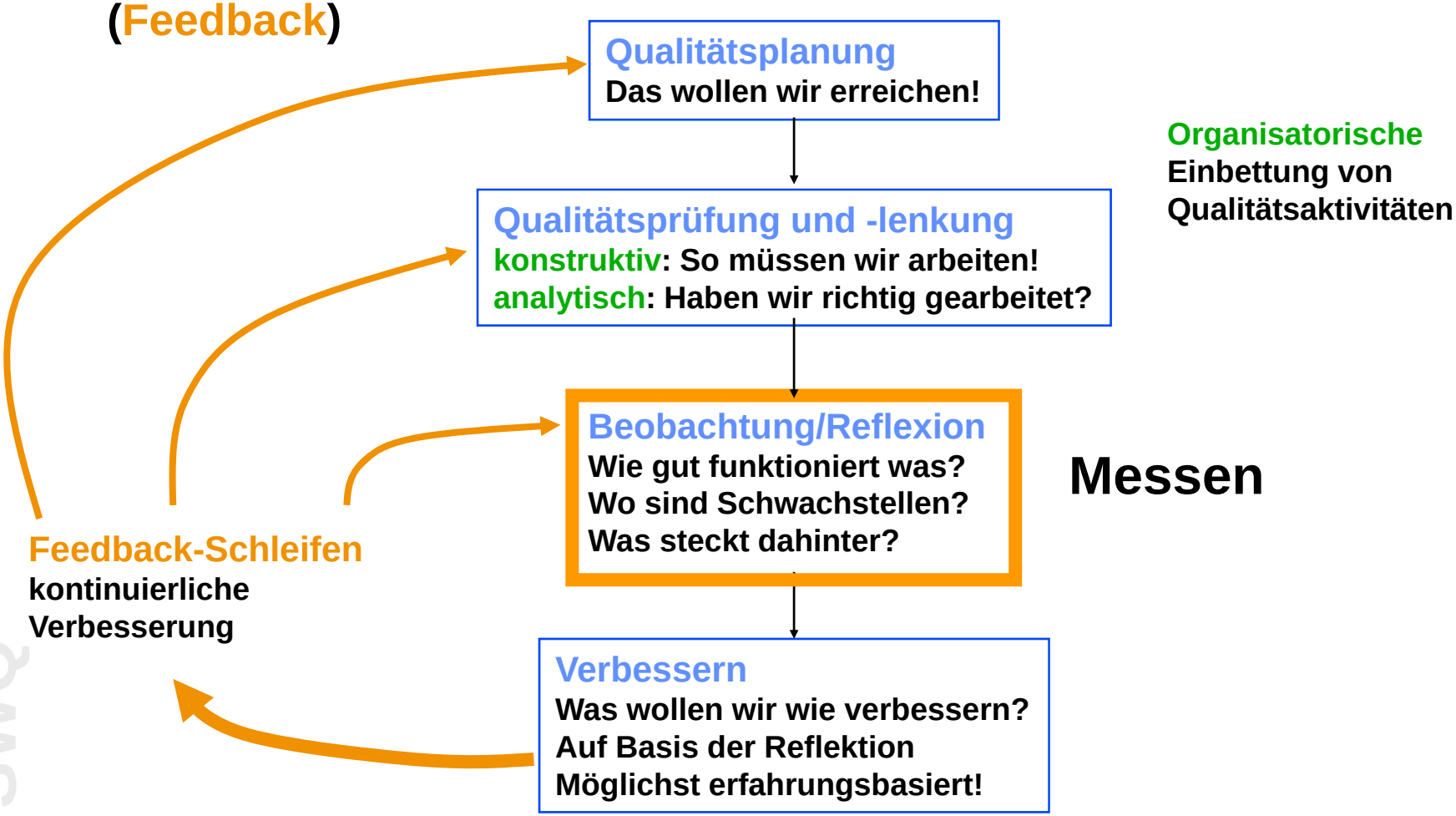
Qualität (oder so)

Überblick

- **Was wird gemessen und wozu?**
- **In welchem Zusammenhang steht Messung?**
  - **Zusammenhang mit anderen Qualitäts-Maßnahmen**
  - **Voraussetzung für Planung und Kontrolle**
  - **Vorhersage von Fehlern, Risiken früh erkennen**
  - **Spezialfall Analytische Qualitätssicherung**
- **Metriken und Maße in der Software-Qualität**
- **„Metriken maßgeschneidert“: Goal-Question-Metric (GQM)**

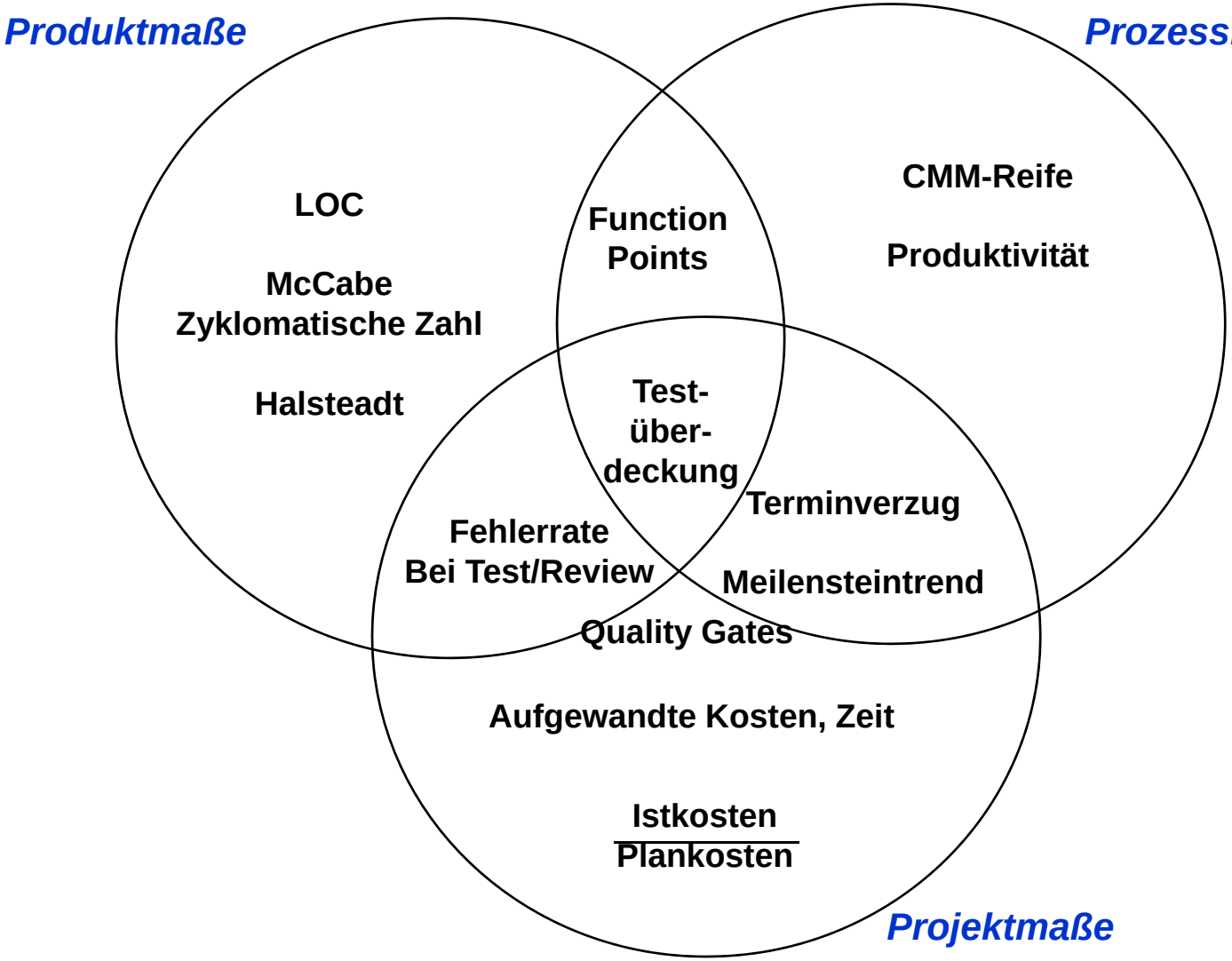
Zusammenspiel der Maßnahmen

- **Q-Management: Planen-lenken-beobachten-verbessern (Feedback)**



Was und woran misst man?

Beispiele



Was wird gemessen?

Prinzip

Direkt	Indirekt am Produkt	Prozess
Codeumfang	**Funktionalität**	**Produktivität**
Laufzeit	**Lesbarkeit**	**Nachvollzieharkeit**
Speicherbedarf	**Wartbarkeit**	**Reife**
Aufwand (PM)	**Usability**	**Zertifizierbarkeit**
Fehler	**Komplexität**	**Agilität**

aber wie misst man das?

Softwaremaße und -metriken

Definitionen

Messen

- Eigenschaften der realen Welt Zahlen oder Zeichen zuordnen

Maß

- Zuordnung einer Zahl oder Zeichens und einer Einheit (z.B. 1 m)

Metrik [gr.: „Kunst des Messens“] in der Mathematik

- Abstand zwischen zwei Dingen (axiomatisch definiert)

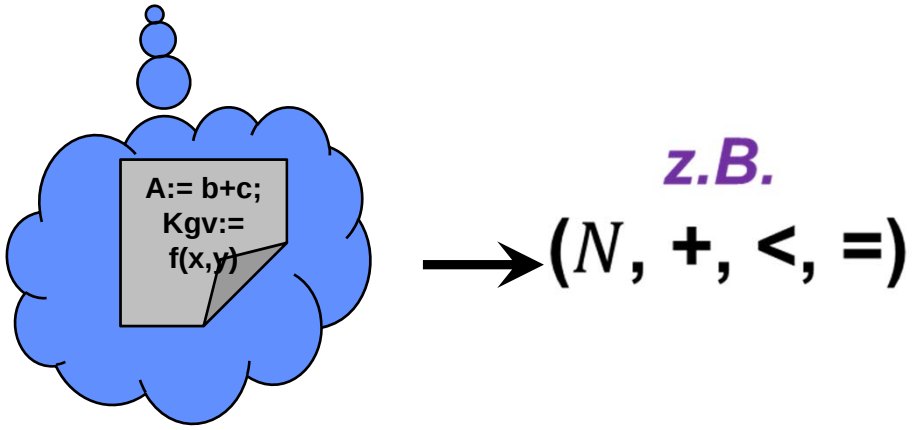
Softwaremetrik

Funktion, die eine Software-Einheit in einen Zahlenwert abbildet. Dieser Wert ist interpretierbar als der Erfüllungsgrad einer Qualitätseigenschaft der Software-Einheit.
(IEEE Standard 1061)

Messen und Skalen

Grundlagen

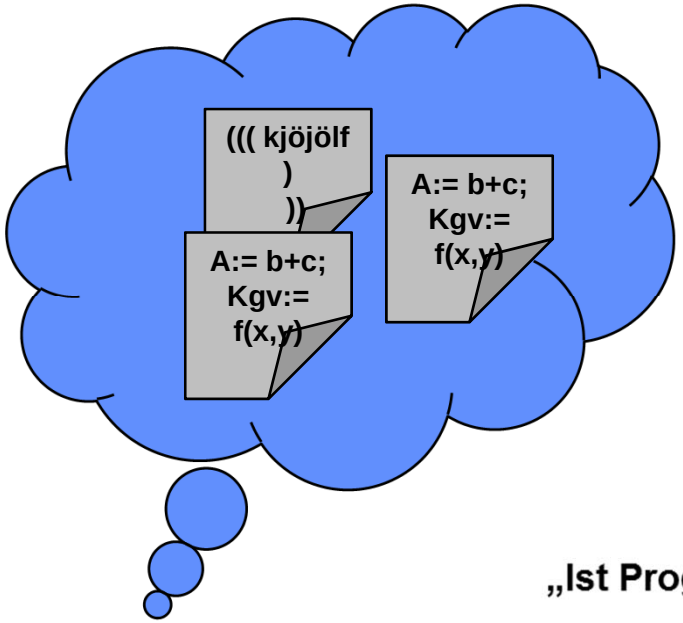
- **Messen: Abbilden eines (Software-) Objekts auf Skala**
- **Skala: Mathem. Struktur aus**
  - **Grundmenge**
  - **Operationen**
  - **Relationen (Vergleiche)**



- **Aussagekraft der Metrik hängt von Skala ab**
  - **Welche Operationen sind darauf zulässig?**
  - **ACHTUNG: meist weniger als auf Grundmenge definiert!**
- **Übler Fehler: Anwendung unzulässiger Operationen/Vgl.**

Nominalskala

- **Objekte werden von Metrik *m* „mit Namen versehen“ (nomen)**
  - **also klassifiziert**
  - **bzw. Schubladen zugeordnet**
  - **Es gibt nur eine Äquivalenzrelation ≅**



Programme

Metrik *m*: „in Sprache geschrieben“

({Java, C, Scheme}, ≅)

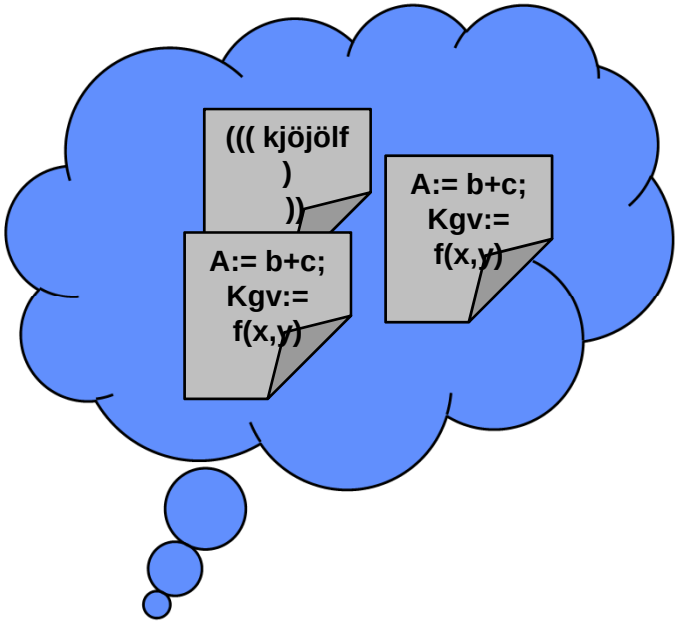
Erlaubt ist nur:

„Ist Programm A in gleicher Sprache geschrieben wie B?“

$m(A) \cong m(B)$?

Ordinalskala

- **Objekte werden durch *m* klassifiziert**
  - **Es gibt eine Äquivalenzrelation $\cong$**
  - **Zusätzlich ist eine Ordnung definiert : „besser“ $\overset{*}{<}$**



Studentische Programme

Metrik *m*: „wird benotet “

$$\rightarrow (\{1,2,3,4,5\}, \cong, \overset{*}{<})$$

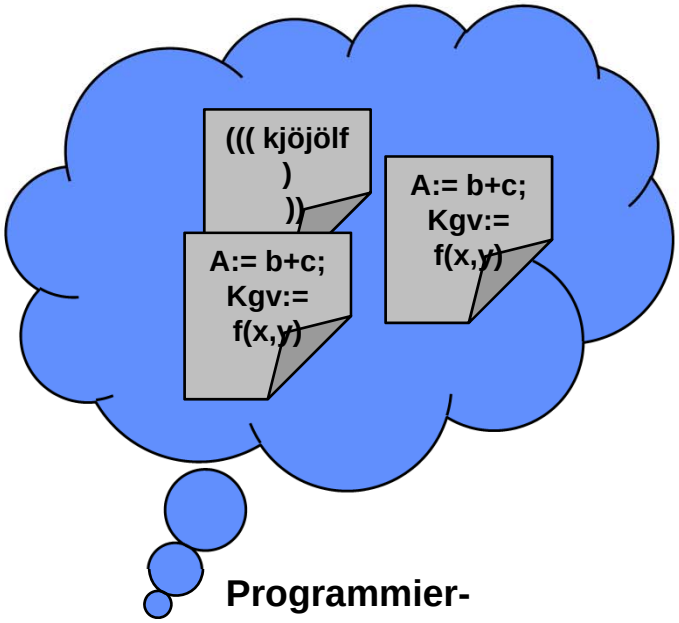
Erlaubt ist nun zusätzlich:

„Programm A wird besser benotet als B“

$$m(A) \overset{*}{<} m(B)$$

Intervallskala

- **Abstände (Intervalle) sind bedeutungsvoll**
  - **Intervall/Abstand zwischen aufeinanderfolgenden Werten ist „gleich groß“,**
    z.B. (3→4 = 7→8) oder (1.5.11 → 7.5.11 = 3.8.12 → 9.8.12)
  - **Rechnen mit Intervallen ist erlaubt, mit Werten nicht**
    $(7.5.11 \neq 7 * 1.5.11)$



Metrik *m* : „Termin“

({TageDifferenz},+, -, , /)

Erlaubt ist nun zusätzlich:

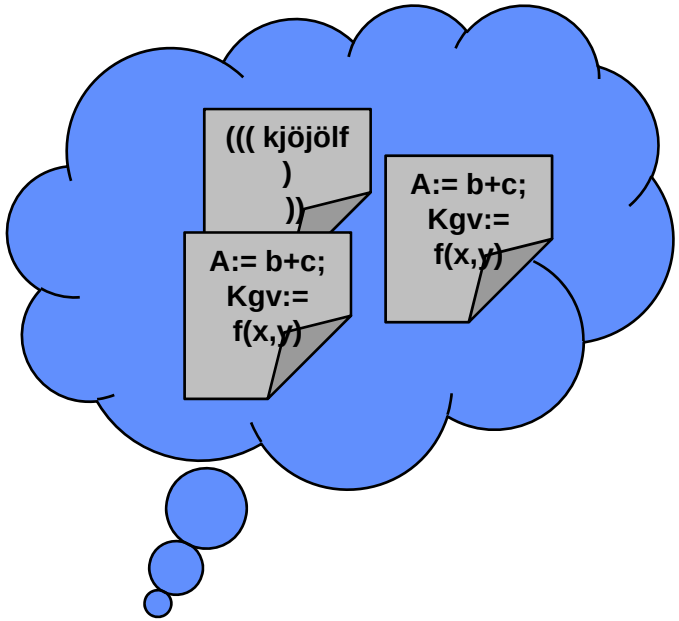
Intervalle ins Verhältnis setzen
Projekt ist *halb so stark verspätet* wie letztes

(d.h.: *Intervall* [t.Fertig-t.Geplant] ist *halb so groß* wie dort)

SWQ

Rationalskala

- **Definierter Nullpunkt: man darf auch Verhältnisse bilden**
  - **Also zusätzlich multiplizieren, dividieren**



Programme

Metrik *m*: „Anzahl Getter“

$$(R^+, 0, \ldots, *, /)$$

Erlaubt ist nun zusätzlich:

„Programm A hat doppelt so viele Getter wie B“
Mittelwerte, Durchschnitte usw.

***m* (A) = 2* *m* (B)**

Skalen-Beispiele um Software

- **CMM Maturity Level (1-5)**
- **Beliebtheit einer App im AppStore (nach Downloads)**
- **Gefundene Fehler**
- **Zertifizierter Betrieb (ISO 9000)**
- **Bisher erreichte Leistungspunktzahl (Transcript of records)**
- **Wertung in Vorlesungsevaluierung**

Klassische Metrik: Lines Of Code

Beispiel

LOC zählt die Codezeilen eines Programms.

LOC ist somit ein scheinbar einfaches Maß.

ABER:

Was wird gezählt? Und: was impliziert das?

- **Nur Zeilen mit ausführbarem Code**
- **Ausführbarer Code und Datendefinition**
- **Ausführbarer Code, Definitionen und Kommentare**
- **Anzahl physikalischer Zeilen**
- **Begrenzung der Konstrukte zum Zählen verwenden**

Klassische Metrik: Lines Of Code

Genauer hingesehen

Wie viele LOC hat dieses Programm?

```
public void buyTicket()
{
  // Adults pay more
  if ( isAdult )
  {
    // Payment delegated
    payFullFee();
  }
  else
  {
    // all others pay less
    payReducedFee();
  }

  //In all cases print the ticket
  printTicket();
}
```

alle Zeilen	17
ausführbarer Code	12
+Kommentare	16
Zeilen mit Semikolon	3

Versuche, die Qualität zu messen

Bekanntestes Beispiel McCabe: Cyclomatic Complexity

Abgleitet aus dem „Programmablaufgraph G“

V(G) = e - n + 2

edges, nodes

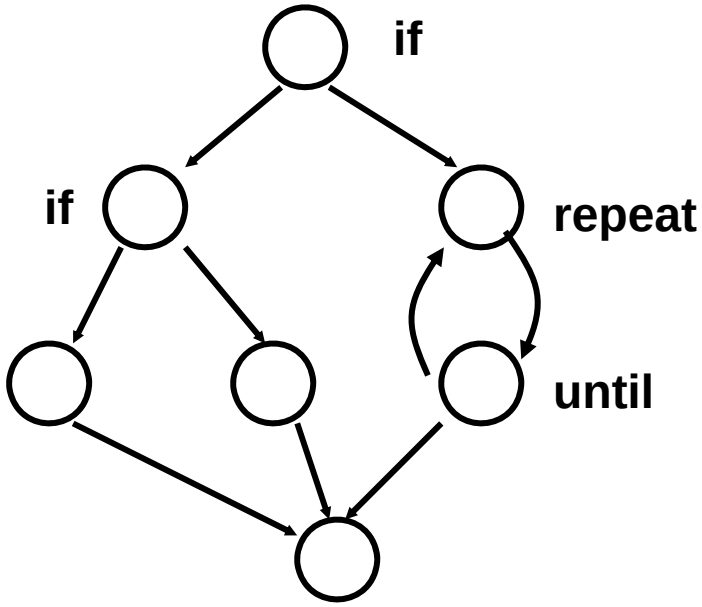
Algorithmus

gilt für strukt. Sprachen (Pascal, Cobol)

v(G) = Anzahl (IFs, Schleifen)

+ CASE-Fälle (je Anw. -1)

+ 1



Fehler-Risiko abschätzen

V(G) > 10 : mittel

V(G) > 20 : hoch

V(G) > 50 : unbeherrschbar

In diesem Beispiel:

9 edges – 7 nodes + 2

bzw.

2 ifs+ 1 Schleife + 0 CASE + 1

= 4

Beispiel für McCabe-Berechnung

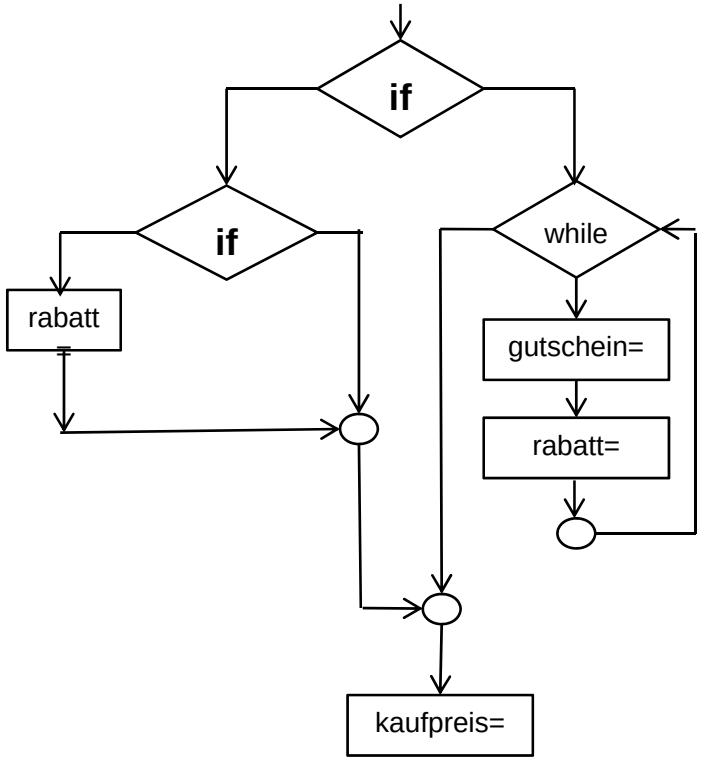
Auszug aus Beispielprogramm

```
…
EuroWert kaufpreis=0;
EuroWert rabatt=0;
…
// Waren aussuchen, Kaufpreis ermitteln, dann:

if (kunde.istMitarbeiter()) {
  // Mitarbeiterrabatt, aber keine Gutscheine

  if (kunde.istRabattberechtigt()) {
    rabatt=kaufpreis*kunde.rabattSatz();
  }
}
else {
   // externer Kunde, hat evtl. Gutscheine

   while (kunde.hatGutschein()){
      gutschein = kunde.gibtGutschein();
      rabatt = rabatt+gutschein.wert();
   }
}
kaufpreis=kaufpreis-rabatt;
…
```

Programmablaufgraphen ableiten:

Zwischenergebnis:

Programmablaufplan (DIN 66 001)

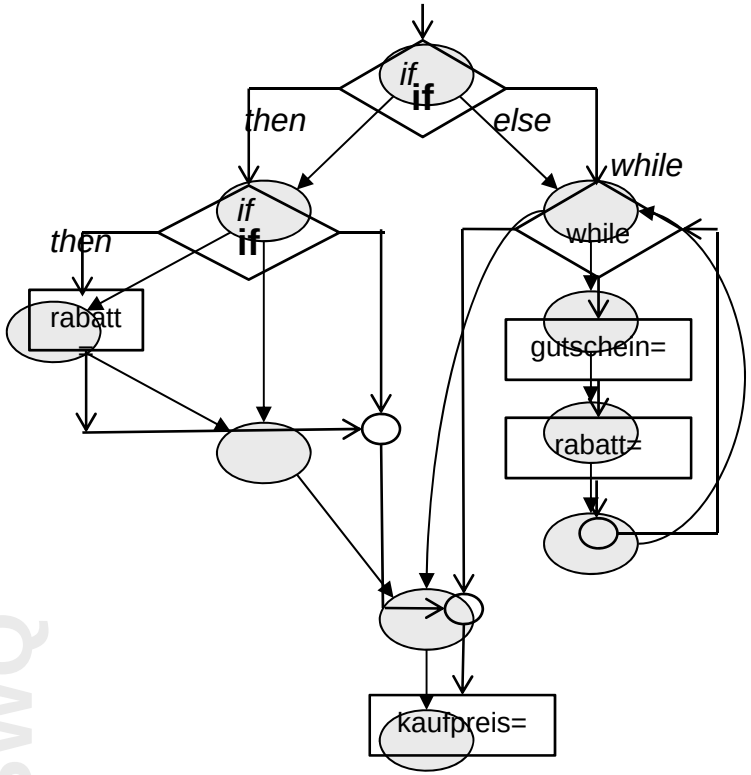


Beispiel für McCabe-Berechnung

Programmablaufgraphen ableiten:

Zwischenergebnis:

Programmablaufplan (DIN 66 001)



Schritt 2 (Abstrahieren):

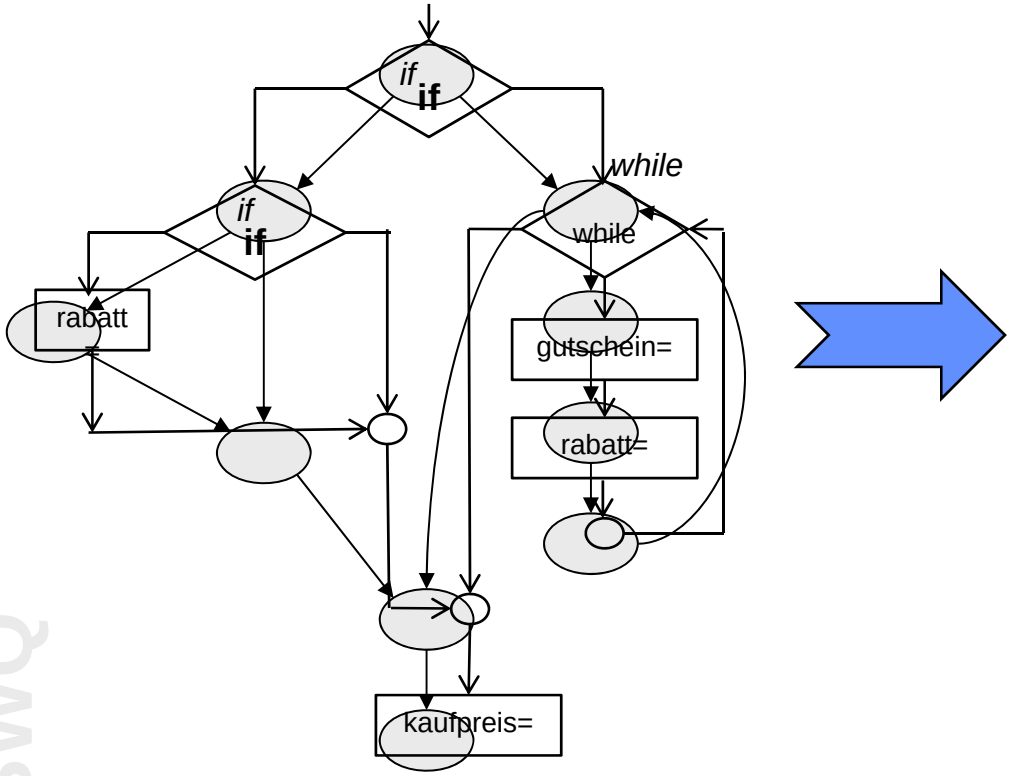
Programmablaufgraph

Beispiel für McCabe-Berechnung

Programmablaufgraphen ableiten:

Zwischenergebnis:

Programmablaufplan (DIN 66 001)



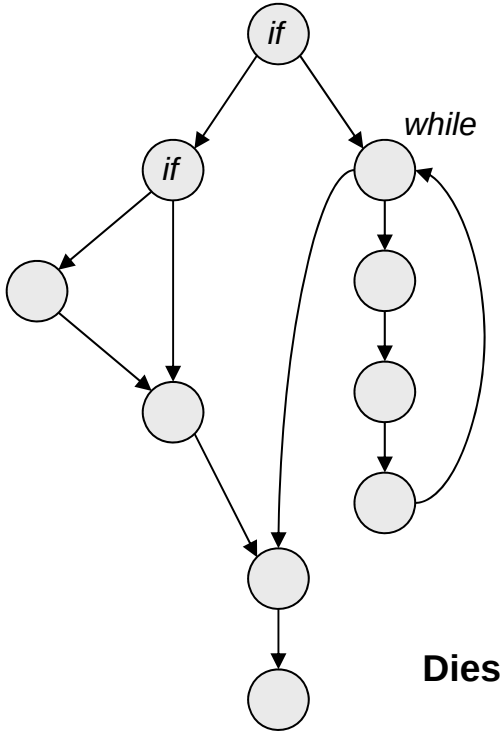
Schritt 2 (Abstrahieren):

Programmablaufgraph

Beispiel für McCabe-Berechnung

Programmablaufgraph

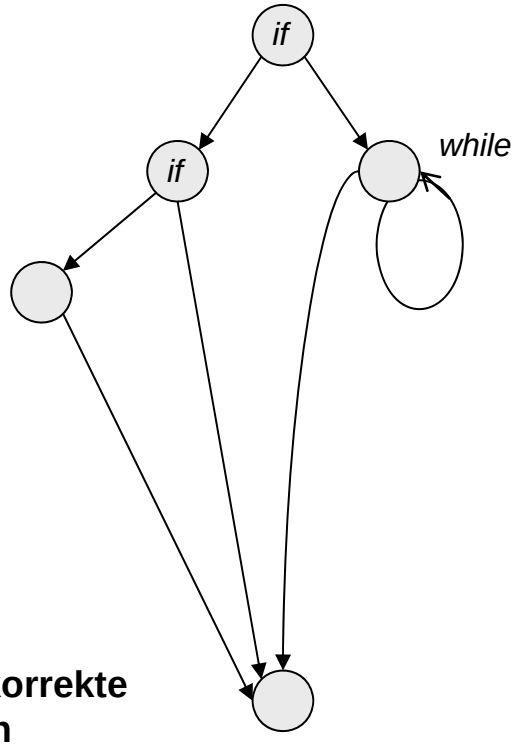
mit unnötigen Knoten



G1

Schritt 3:
Reduzierter Programmablaufgraph

ohne unnötige Knoten



G2

Dies sind zwei gleichermaßen korrekte Programmablaufgraphen

Beide mit V(G1)=V(G2)=4

Es gibt noch viel mehr Metriken

Beispiel: Metriken für OO

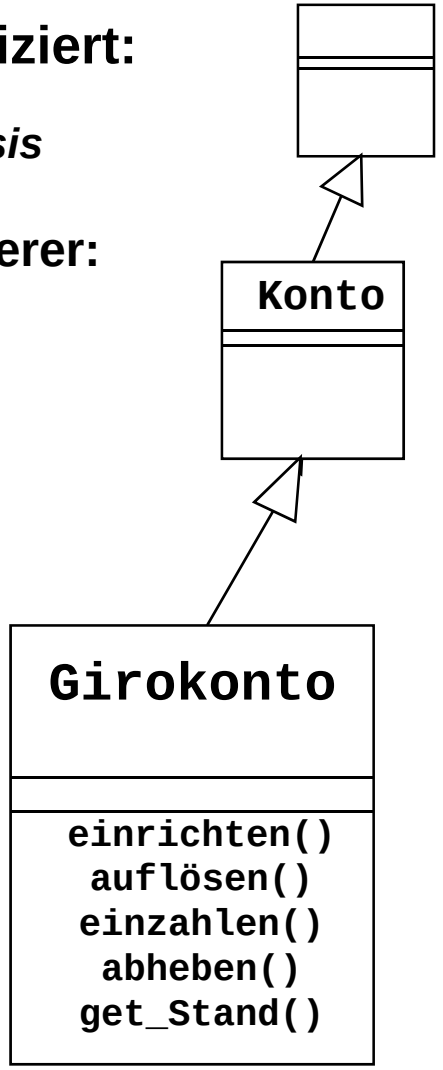
Traditionelle Maße für Objektorientierung modifiziert:

Klassen, Objekte, Vererbung, Polymorphie als Basis

Objektorientierte Metriken nach Chidamber und Kemerer:

- **Weighted Methods per Class (WMC)**
  - **McCabe für alle Methoden der Klasse, addiert.**
- **Depth in Inheritance Tree (DIT)**
  - **Wie viele Oberklassen darüber?**
  - ***Je mehr, desto fehlerbehafteter.***
- **Coupling Between Objects (CBO)**
  - **Anz. Klassen, mit denen kommuniziert wird.**
  - ***Je mehr, desto höhere Kopplung.***

Diese Metriken werden auf Klassen angewendet.



Beispiele für Anforderungs-Metriken

- Korrektheit und Verständlichkeit von Texten
  - Fehlerfreiheit (Rechtschreibhilfe)
  - Anteil Passivsätze
  - Flesch-Kincaid und Flesch Reading Ease (Lesbarkeit)
  - Durchschnittliche Länge der Sätze
- Sprachliche Defekte von Anforderungen
- *Beispiel: Eindeutigkeit*

$$Eindeutigkeit = \frac{\sum Anforderungen\_Ohne\_Defekte}{\sum Anforderungen}$$

- *Beispiel: Testbarkeit*

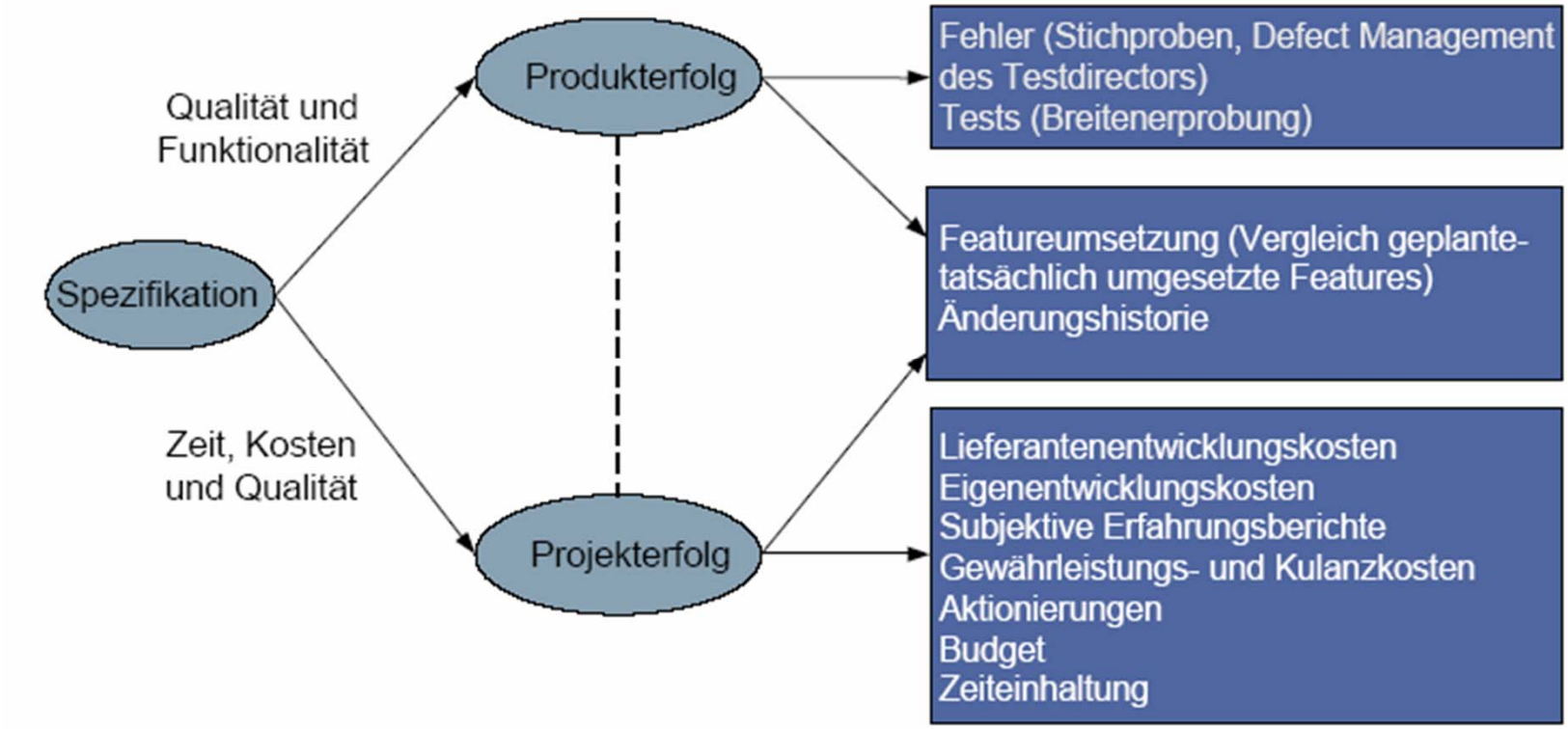
$$Testbarkeit = \frac{\sum Anforderungen\_\triangleright\_Zwei\_Abnahmekriterien}{\sum Anforderungen} \bullet Eindeutigkeit$$

  - Hat die Anforderung mind. zwei Abnahmekriterien?
  - Ist sie möglichst eindeutig formuliert?
  - Verweist sie auf die Abnahmekriterien?

M.Recknagel, C. Rupp: Metriken für Anforderungen. Wie gut sind Ihre Anforderungen wirklich? SQM, 11.5.06, Düsseldorf. SQS

Kriterien für Projekt- und Produkterfolg

quasi ein Qualitätsmodell für die Spezifikation

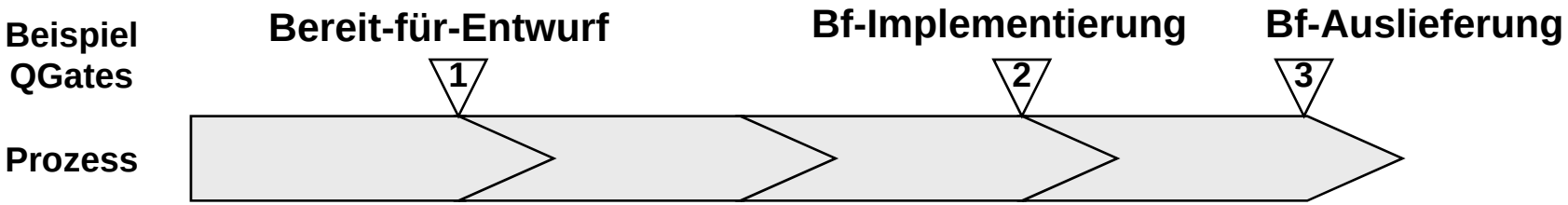


M.Recknagel, C. Rupp: Metriken für Anforderungen. Wie gut sind Ihre Anforderungen wirklich? SQM, 11.5.06, Düsseldorf. SQS

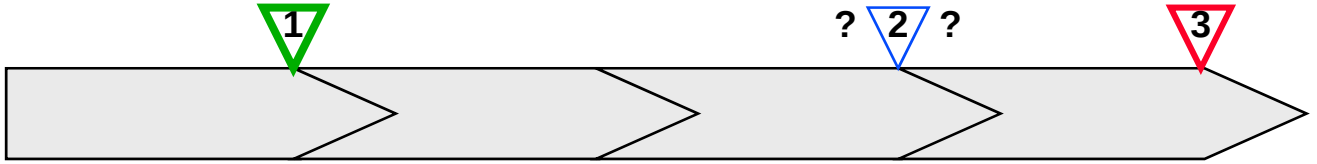
Fortschrittsmessung mit Quality Gates

komplexe Metrik auf simpler Ordinalskala

- **Idee**
  - **Kurze, scharfe Prüfung an definierten *Prozess*-Stellen**



  - **Prüfkriterien: essenzielle Fortschrittsindikatoren**
    - **Dokumente vorhanden, zugänglich?**
    - **Wichtige inhaltliche Prüfungen bestanden?**
  - **Fortschrittsmessung: Passierte Quality Gates (hier: 1)**



- **Mehr dazu: bei Reviews (methodisch ähnlich)**

„Fallen“ bei Metriken

- **Was misst man *wirklich*?**
  - **(Inwiefern) ist Metrik ein Modell für die gemessene Eigenschaft?**
  - **Bezug zu Mensch erfasst (z.B. Benutzbarkeit, Wartbarkeit)?**
- **Beispiel: „cyclomatic complexity is used as a**
  - **quantitative measure of *testability* [siehe bei Testen!] and an**
  - **indication of ultimate *reliability*“**
- **Was kann man aus den Ergebnissen schließen?**
  - **Auf welcher Skala liegen die Messungen?**
- **Generell: Vorsicht beim Interpretieren!**
  - **Gefährlicher Ansatz: „was können wir denn leicht messen?“**
  - **Ganz anderes Prinzip: Goal-Question-Metric**

Q misst

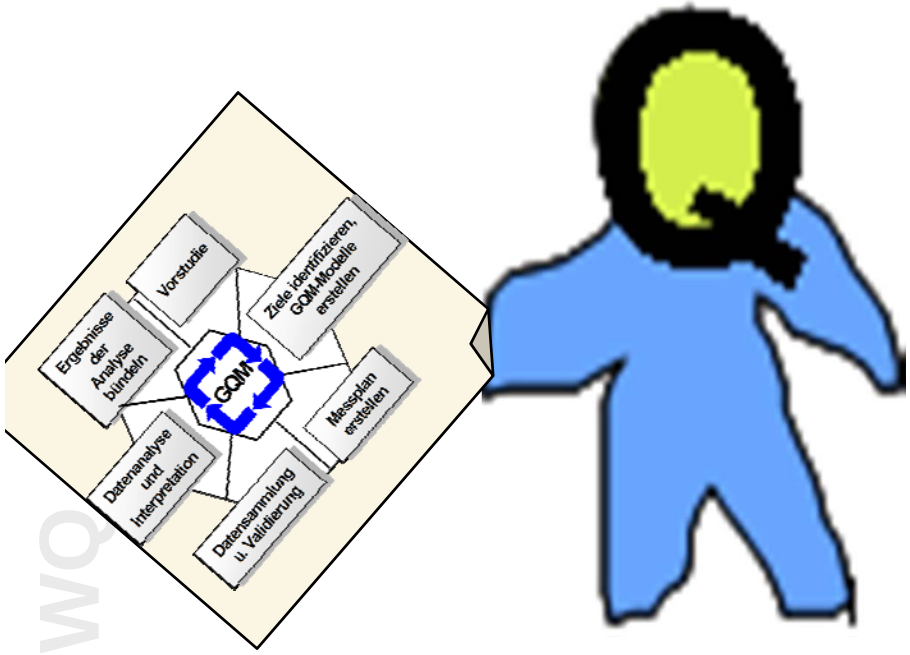
Ziele

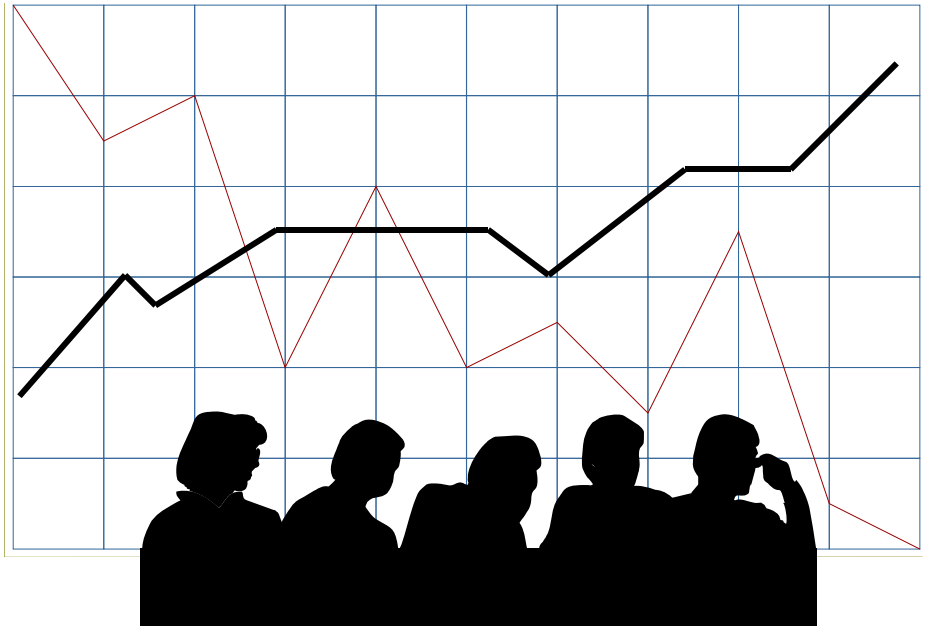
⇩

Fragen

⇩

Metriken





Wie findet man die richtigen Metriken?

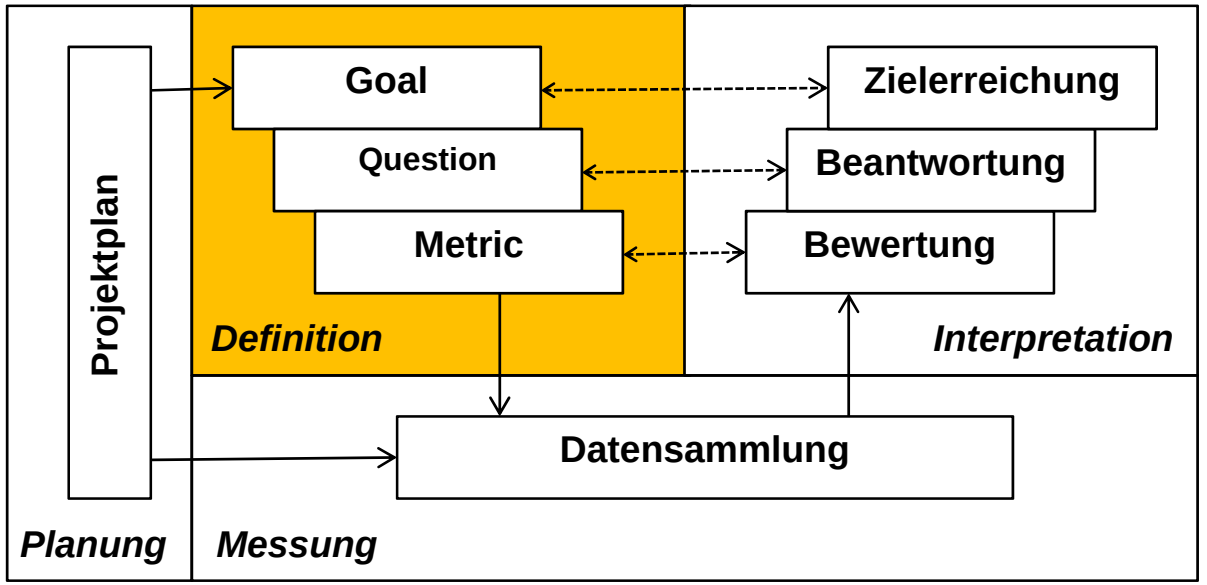
- **Situation:**
  - **Es gibt ein Problem**
    - **zu viele Fehler, zu viel Aufwand oder zu lange Entwicklungszeiten**
  - **Unternehmen oder Projekt möchte/muss sich verbessern**
    - **Dazu muss man etwas ändern**
    - **Aber was? Und wird dadurch wirklich etwas besser?**
  - **Idee: Man müsste messen!**
    - **Nur was?**
    - **Die „normalen Metriken“ wie LoC, McCabe usw.?**
    - **Mit einem Wort: „was sich leicht messen lässt“?**
  - **Das ist oft nicht die beste Lösung!**
    - **Geringe Aussagekraft für spezielles Problem**
    - **Oft fehlen in der Analyse wichtige Daten**

Was tun?

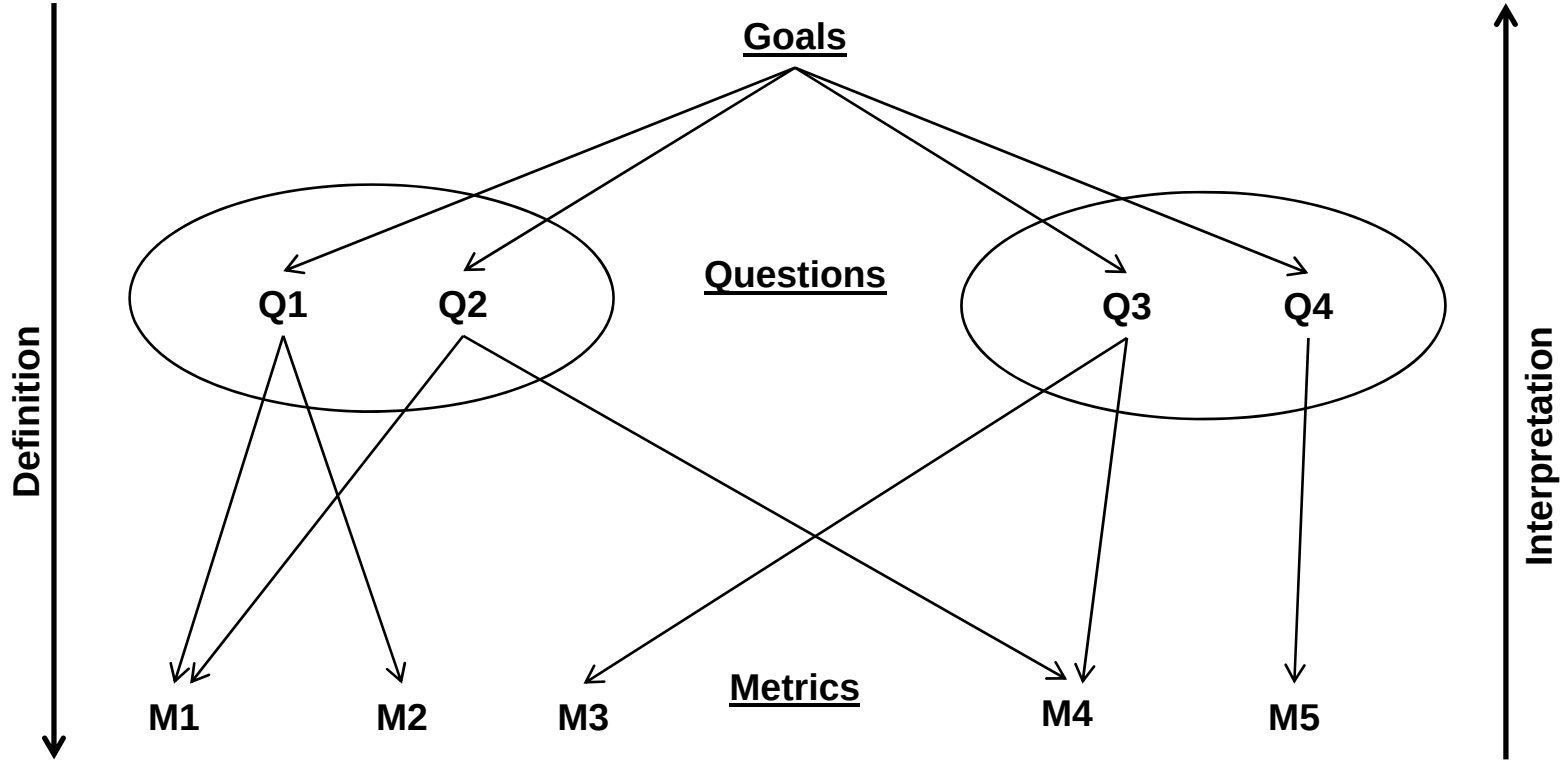
Goal-Question-Metric (GQM)-Methode

Überblick

- **Systematisches Vorgehen beim Messen von SW und Prozessen**
- **Top-Down: Ziele aufstellen, dazu passende Metriken ableiten**
- **GQM wurde erfolgreich in Industriebprojekten eingesetzt**



GQM-Modell



Vorgehen bei GQM

Verfahren

Prinzip ist einfach:

Einige Ziele, mehr Fragen, möglichst wenige Maße

1. **Ziele erheben und verfeinern (Goal): Zielbaum**
2. **Ziele mit Facetten genauer beschreiben (Goal)**

Aspekte, um die es geht	**- zum Beispiel beim Testen...**
Zweck	Verstehen; verbessern
Qualitätsaspekt	Effizienz, Effektivität, Kostenwirksamkeit
Betrachtungsgegenstand	Testprozess, Testplan
Perspektive u. Umgebung	Projekt/Bereich xyz

3. **Ableitung von Fragen zu den Zielen (Questions): Ein Abstraction Sheet pro Ziel**
4. **Ableitung von zugehörigen Metriken (Metric)**
5. **Messplan für Datenerhebung erstellen (Metric)**
6. ***Datenerhebung und Auswertung***

Messziele mit Facetten

Beispiel: Verständlichkeit des Codes

Ziel	Zweck	Q-Aspekt	Beobachtungsgegenstand	Perspektive
3.1	Untersuche	Lesbarkeit	Kommentare im Code	Entwickler
3.2	Verbessere	Lesbarkeit	Kommentare im Code	Tester
...	...	...	...	...
5.1	Steuere	Effizienz	Ablauf Modultest	Projektleitung
...	...	...	...	...

Facetten führen zu Nachfragen, Umformulierung
hier als Spalten notiert

Abstraction Sheet

Beispiel

Abstraction Sheet

Ziel: G 3.2 **Ausgefüllt von: Q Datum: 14.2.**

Zweck der Messung	Qualitätsaspekt	Betrachtungsgegenstand	Perspektive
Verbessere	Lesbarkeit	Kommentare im Code	Tester

Qualitätsfaktoren

- a- Kommentardichte
- b- sprachlich verständlich
- c- Bezug zum Anwendungsglossar
- d- mit Begründungen (Rationale)

1

Einflussfaktoren

- Forderungen in Programmierrichtlinien
- Englischfähigkeiten
- Schulung
- Moderierter Erfahrungsworkshop zum Kommentarstil

4

Ausgangshypothese: wie ist es jetzt?

- a- unter 5% der Zeilen sind Kommentare
- b1- ca.70% enthalten nur Stichwörter, aber keine vollständigen Sätze
- b2- schlechtes Englisch
- c- keine Referenzen auf Glossar (<1%)
- d- ca. ¾ der Kommentare beziehen sich darauf, wie es funktioniert – nicht, wieso es so gemacht wird

2

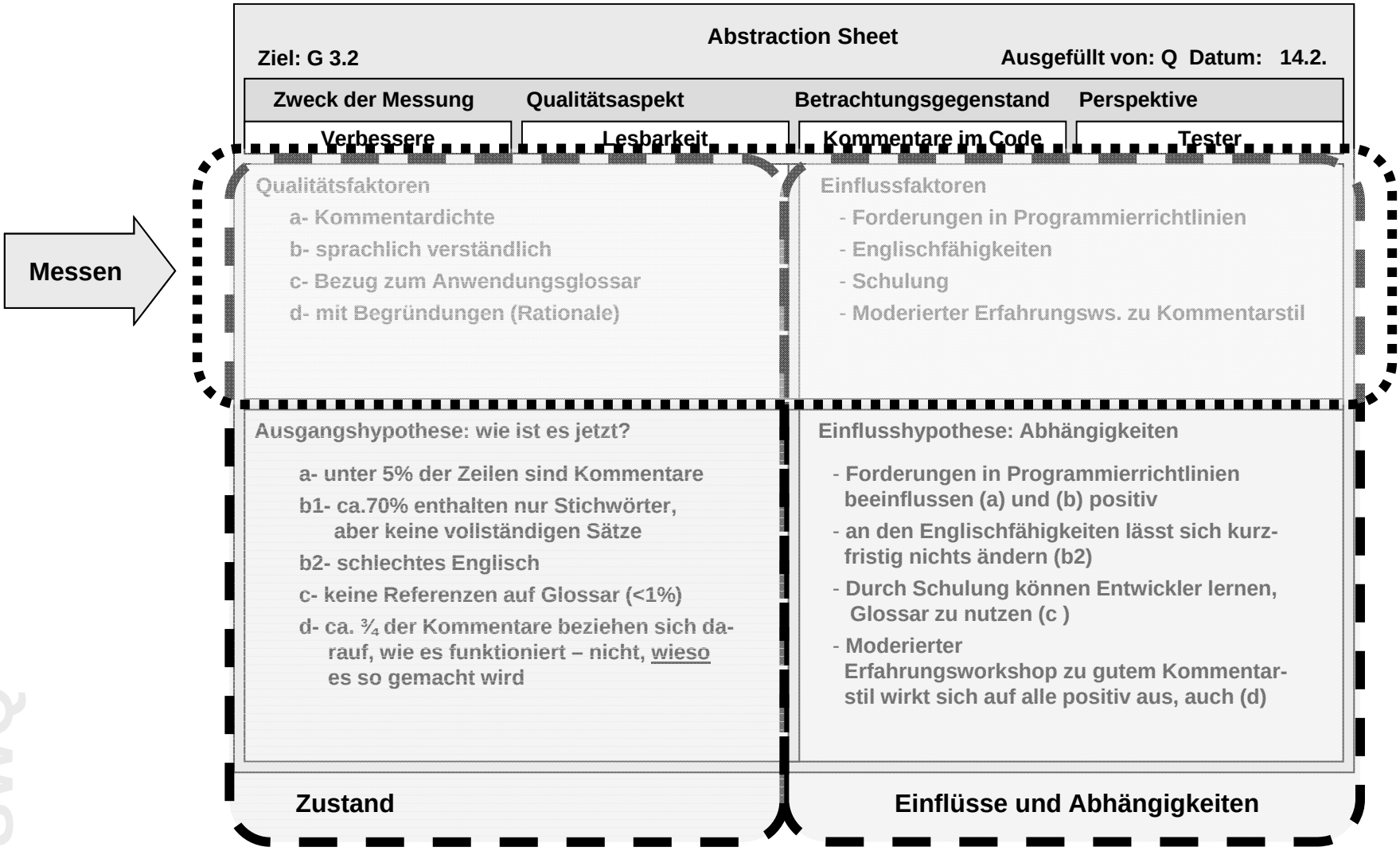
Einflusshypothese: Abhängigkeiten

- Forderungen in Programmierrichtlinien beeinflussen (a) und (b) positiv
- an den Englischfähigkeiten lässt sich kurzfristig nichts ändern (b2)
- Durch Schulung können Entwickler lernen, Glossar zu nutzen (c)
- Moderierter Erfahrungsworkshop zu gutem Kommentarstil wirkt sich auf alle positiv aus, auch (d)

3

Abstraction Sheet

Zusammenhang



Ableitung einer Fragestellung und Metrik

Beispiel

- **Ableitung von Fragestellungen aus dem Abstraction Sheet sowie zugehörige Metriken**

Question: Führt ein *moderierter Erfahrungsworkshop zum Kommentarstil* dazu, dass Entwickler auch Begründungen (Rationale) in den Kommentaren dokumentieren?

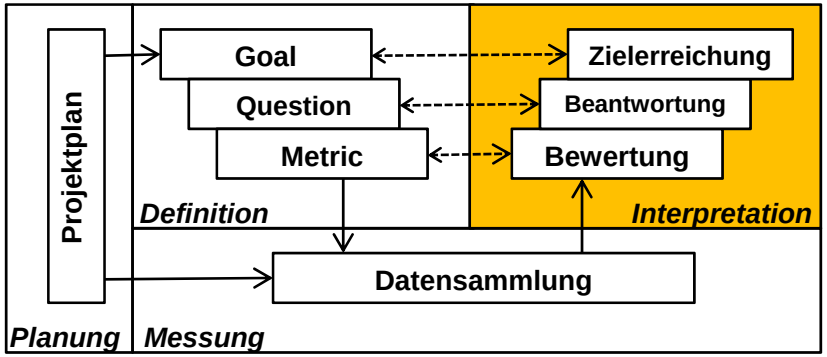
Metric: *Anteil der neuen Kommentare, die eine Begründung enthalten.* Vor und nach der Durchführung eines Erfahrungsworkshops, zum Vergleich.

Auswertung der Messungen

- **Messungen werden in einer *Feedback Session* diskutiert**
  - **Beispiel**: Nach Durchführung des Erfahrungsworkshops enthalten 25% der neuen Kommentare *keine* Begründung (Rationale). Ausgangshypothese war 75%.
    Ist das eine *Verbesserung der Lesbarkeit von Kommentaren*?

- **Rückschluss aus dem Ergebnis der *Feedback Session* auf Fragestellung und jeweiliges Ziel (Goal)**

- **Dokumentierte Ergebnisse der Auswertungsphase**
  - **Beobachtungen**
  - **Interpretierte Messergebnisse**
  - **Schlussfolgerungen**
  - **Erforderliche Aktivitäten**



Zusammenfassung GQM

Messen, was man wissen will,
nicht das, was leicht zu messen ist!

- **Mit GQM findet man systematisch geeignete Metriken (vom Ziel zur Metrik). Manchmal denkt man sich neue aus.**
- **Weglassen ist die Kunst:**
  Wenige prägnante Fragen, wenige Metriken
- **Das Ergebnis ist dann leicht interpretierbar:**
  Einsetzen ins Abstraction Sheet, mit Erwartung vergleichen
- **Die Messergebnisse sind selten statistisch signifikant, aber sehr häufig aussagekräftig und nützlich**