Softwarequalität - Qualitätsmodelle

Proseminar
„IT-Kennzahlen und Codemetriken“

Clara Lange

17.05.2010

TU München

Inhalt

1. Was ist Softwarequalität?
2. Sichten auf Softwarequalität
3. Messen von Qualität
4. Qualitätsmodelle
5. Qualitätsmanagement

1. Was ist Softwarequalität?

1. Definitionen von Qualität
2. Der Begriff „Softwarequalität“
3. Unterscheidung von Softwarequalität

1.1. Definitionen von Qualität

Qualität

=

„Gesamtheit der Eigenschaften von Erzeugnissen, die den Grad ihrer Eignung für einen bestimmten Verwendungszweck bestimmen“

[Quelle: Fremdwörterbuch]

1.1. Definitionen von Qualität

Qualität

=

„Bezeichnung einer potentiell wahrnehmbaren Zustandsform von Systemen und ihrer Merkmale, welche in einem bestimmten Zeitintervall anhand bestimmter Eigenschaften des Systems in diesem Zustand definiert wird“

[Quelle: wikipedia.de]

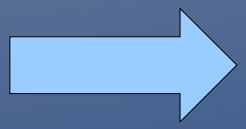
1.2. Der Begriff „Softwarequalität“

- Wie kann man Softwarequalität messen?
- Wer ist verantwortlich für die Qualität von Software?
- Lohnen sich Verbesserungen an der Softwarequalität?
- Wie beeinflusst die Qualität den Gebrauch vom Benutzer?
- Führt hohe Qualität auch zu höherem Profit und hält sich das Produkt lang auf dem Markt?

1.2. Der Begriff „Softwarequalität“

Good Software = Good Business???

Softwarequalität bedeutet auch, dass die festgelegten Anforderungen erreicht werden.



Softwarequalität ist ein

komplexes und vielschichtiges Konzept,

das aus verschiedenen Perspektiven betrachtet werden kann

und durch Qualitätsmanagement erreicht werden kann.

1.3. Unterscheidung von Softwarequalität

Produktqualität
= Grad der Qualität eines Produktes

Prozessqualität
= Qualität der Herstellungsprozesse für ein Produkt

Innere Qualität
= Entwicklerperspektive
(→ Lesbarkeit des Quellcodes, Erweiterbarkeit, Wartbarkeit, ...)

Äußere Qualität
= Kundenperspektive
(→ Einfache Verwendbarkeit, Zuverlässigkeit, Funktionsumfang, ...)

2. Sichten auf Softwarequalität

1. Transzendente Sicht
2. Benutzerorientierte Sicht
3. Herstellerorientierte Sicht
4. Produktorientierte Sicht
5. Wertorientierte Sicht

2.1. Transzendente Sicht

- Qualität kann nicht präzise definiert und gemessen werden.
- Qualität kann nur durch Erfahrung beurteilt werden.
- Qualität entspricht einem Ideal, das man anstreben will, welches aber wahrscheinlich nie erreicht werden kann.

2.2. Benutzerorientierte Sicht

- Qualität ist ausgerichtet auf die Bedürfnisse und Anforderungen des Benutzers. → Problem: unterschiedliche Benutzergruppen, somit unterschiedliche Bedürfnisse
- Benutzer entscheidet subjektiv über die Qualität von Produkten.

2.3. Herstellerorientierte Sicht

- Qualität entspricht der Einhaltung von Anforderungen.
- Sicht bezieht sich auf die Qualität während der Produktion und nach der Auslieferung.
- Qualität wird durch Prozesskontrolle erreicht, um Kosten für Nacharbeit zu verhindern.

2.4. Produktorientierte Sicht

- Qualität stellt eine exakt definierbare und messbare Größe dar.
- Ordnung der Produkte ist hinsichtlich ihrer Qualität und ihrer Eigenschaften möglich.
- Sicht ermöglicht eine objektive und kontextfreie Beurteilung von Qualität.

2.5. Wertorientierte Sicht

- Qualität wird über die Herstellungskosten und einen angemessenen Einkaufspreis definiert.
- Qualität entspricht einem guten Preis-Leistungs-Verhältnis.

3. Messen von Qualität

1. Allgemeines zum Messen
2. Messen aus der Sicht des Benutzers
3. Messen aus der Sicht des Herstellers

3.1. Allgemeines zum Messen

- Messen von Softwarequalität ist sehr problematisch, da es nicht einmal eine einheitliche Definition von Softwarequalität gibt.
- Gegensätzliche Softwarequalitätsziele: Funktionsumfang ↔ Benutzbarkeit, Portabilität ↔ Sicherheit
- Messen von Softwarequalität ist abhängig von der Sicht.

3.2. Messen aus der Sicht des Benutzers

Zuverlässigkeit:

Messbar durch die Zeit zwischen Ausfällen des Systems oder durch die Anzahl der Ausfälle über eine gewisse Zeitspanne

Benutzbarkeit:

Messbar durch die Zeit, die verstrichen ist, bis der Benutzer sich eine gewisse Kompetenz angeeignet hat, um mit dem Produkt gut umgehen zu können

3.3. Messen aus der Sicht des Herstellers

Anzahl der Defekte/Ausfälle:

- Normalisierung:

  Defektrate = Anzahl der Defekte hinsichtlich der Größe des Softwareprodukts

- Auswirkung auf das Testen:

$$\frac{\text{Anzahl der Fehler während einer bestimmten Software Engineering Aktivität}}{\text{Gesamtanzahl der Fehler}}$$

3.3. Messen aus der Sicht des Herstellers

Überarbeitungs-/Nacharbeitungskosten:

- Nacharbeitungszeit = zusätzlicher Arbeitsaufwand, um Fehler nach der Fertigstellung der Dokumentation und des Codes zu finden und zu beheben
- Nacharbeitungskosten = zusätzliche Kosten, die in der Nacharbeitungszeit entstehen
- Normalisierung:

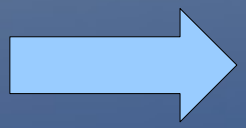
  Anteil der Nacharbeitungszeit an der gesamten Entwicklungszeit

4. Qualitätsmodelle

1. Allgemeines zu Qualitätsmodellen
2. Qualitätsmodell von McCall
3. Qualitätsmodell nach der Norm ISO 9126
4. Probleme von Softwarequalitätsmodellen
5. Mögliche Lösungen für die Problematik der Qualitätsmodelle

4.1. Allgemeines zu Qualitätsmodellen

Qualitätsmodelle zeigen, wie Qualitätseigenschaften miteinander verbunden sind und wie man diese messen kann.



Qualitätsmodelle

dienen als Hilfsmittel für konkrete Planungs-

und Bewertungsaufgaben.

4.2. Qualitätsmodell von McCall

1. Bestandteile des Modells
2. Aufbau des Qualitätsmodells
3. Teilausschnitt aus dem Qualitätsmodell nach McCall

4.2.1. Bestandteile des Modells

Das Modell nach McCall ist ein dreistufiges Qualitätsmodell, zusammengesetzt aus Qualitätsfaktoren („factors“), mehreren Merkmalen („criteria“) zu jedem Faktor und Kenngrößen („metrics“) zu jedem Merkmal.

4.2.1. Bestandteile des Modells

- Korrektheit = Umfang, in dem die Software ihre Spezifikation und damit die Anforderungen erfüllt
- Zuverlässigkeit = Leistungsfähigkeit eines Systems, die angeforderte Funktionalität unter festgelegten Bedingungen und für einen festgelegten Zeitraum zu erfüllen
- Effizienz = Größenordnung, in dem ein System seine Leistung mit einem minimalen Ressourcenverbrauch erbringt

4.2.1. Bestandteile des Modells

- Integrität = Umfang, in dem ein System nicht berechtigte Zugriffe auf Programme und Daten und deren unberechtigte Veränderung verhindert
- Benutzbarkeit = Leichtigkeit, mit der ein Benutzer die Bedienung eines Systems erlernen kann
- Wartbarkeit = Leichtigkeit, mit der ein System verändert werden kann, um Fehler zu beheben

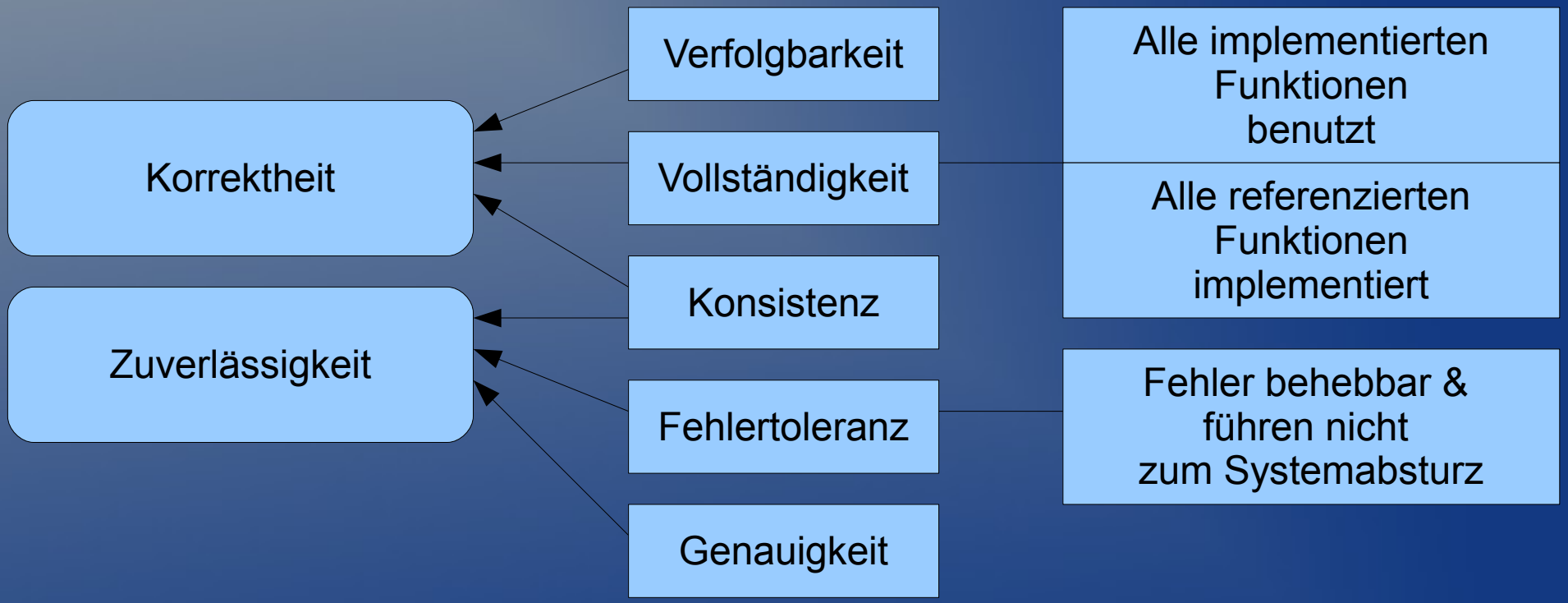
4.2.1. Bestandteile des Modells

- Testbarkeit = Aufwand, der in einem System für die Durchführung von Tests und die Erstellung von Testfällen notwendig ist
- Flexibilität = Bequemlichkeit, mit der ein System abgeändert werden kann, um neue Anforderungen umzusetzen
- Portabilität = Leichtigkeit, mit der ein System von einer Hardwareumgebung oder Softwareumgebung in eine andere übergeführt werden kann

4.2.1. Bestandteile des Modells

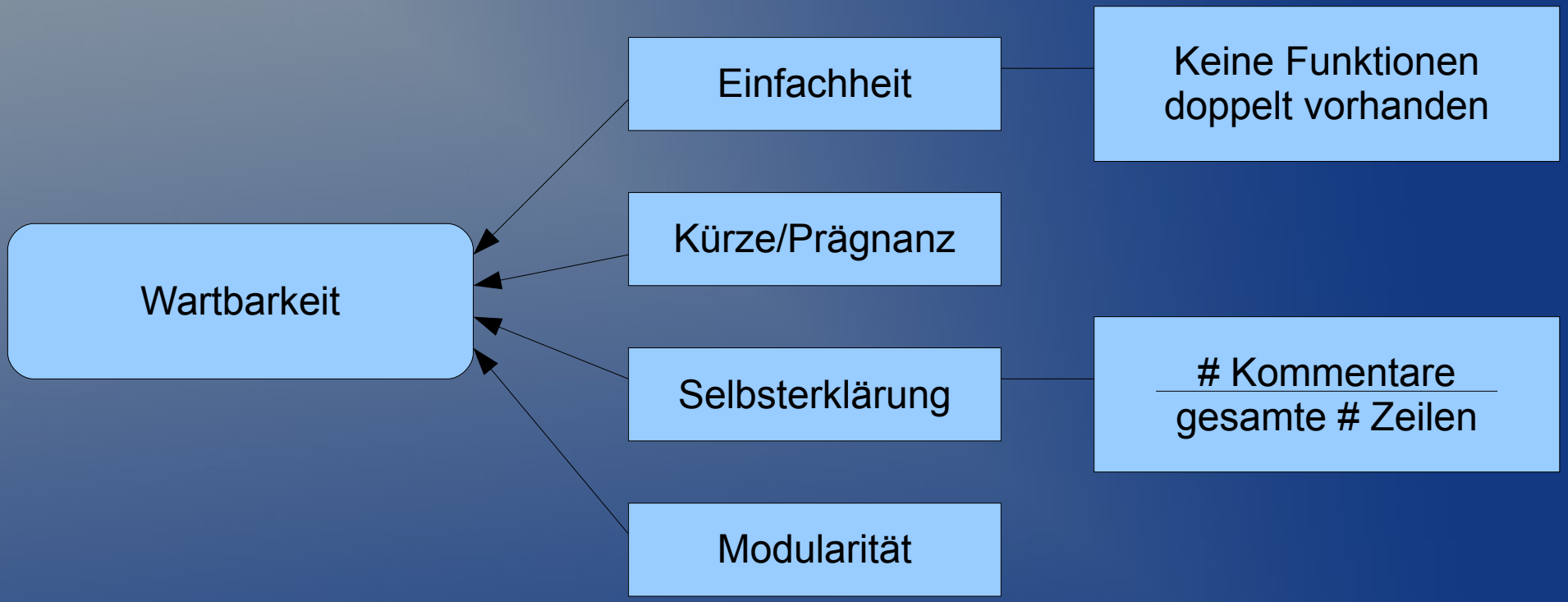
- Wiederverwendbarkeit = Größenordnung, in der ein Teil der Software in mehr als einem Programm oder System erneut verwendet werden kann
- Verknüpfbarkeit = Bequemlichkeit, mit der mehrere Systeme Informationen austauschen und mit diesen arbeiten können

4.2.2. Aufbau des Qualitätsmodells

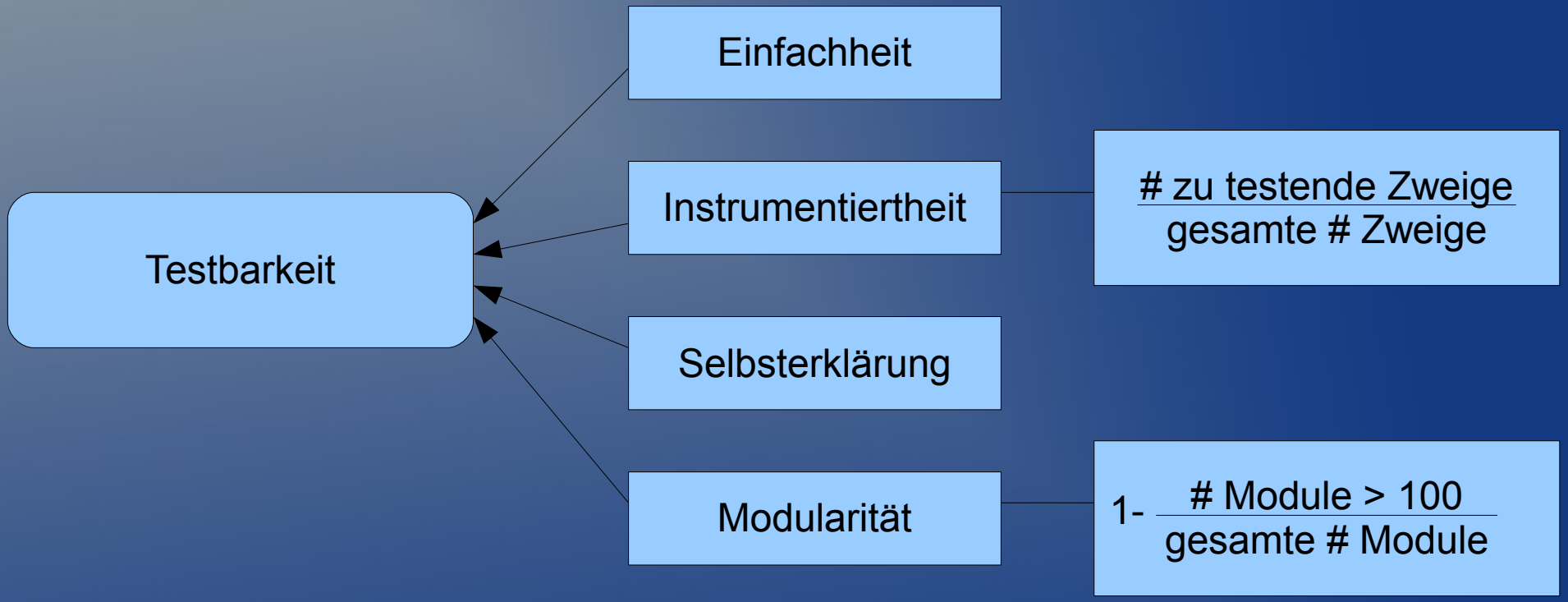


4.2.2. Aufbau des Qualitätsmodells

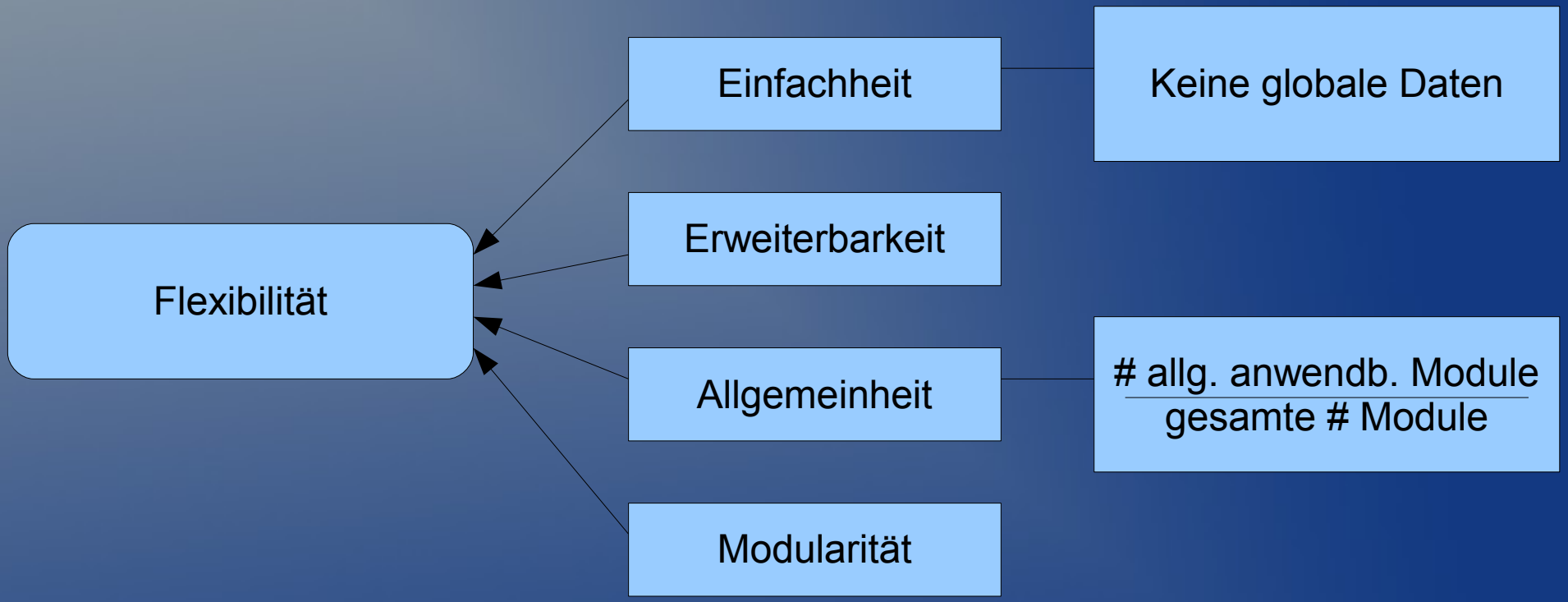
4.2.2. Aufbau des Qualitätsmodells



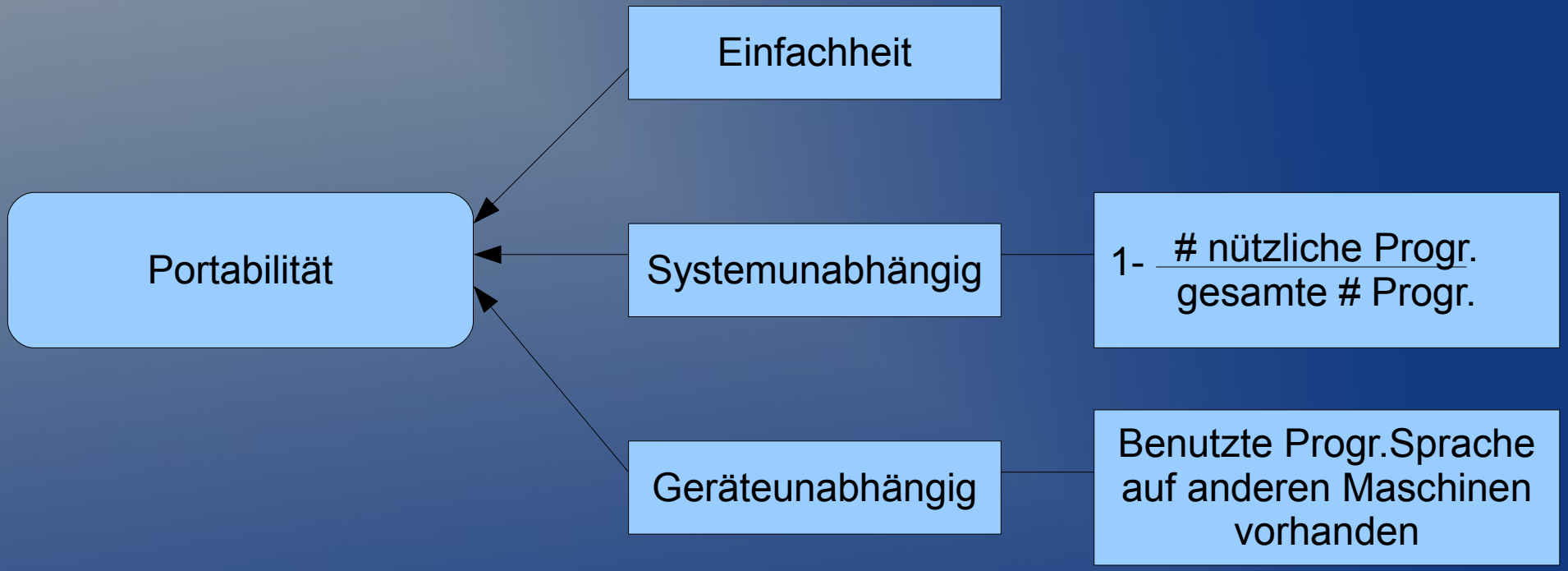
4.2.2. Aufbau des Qualitätsmodells



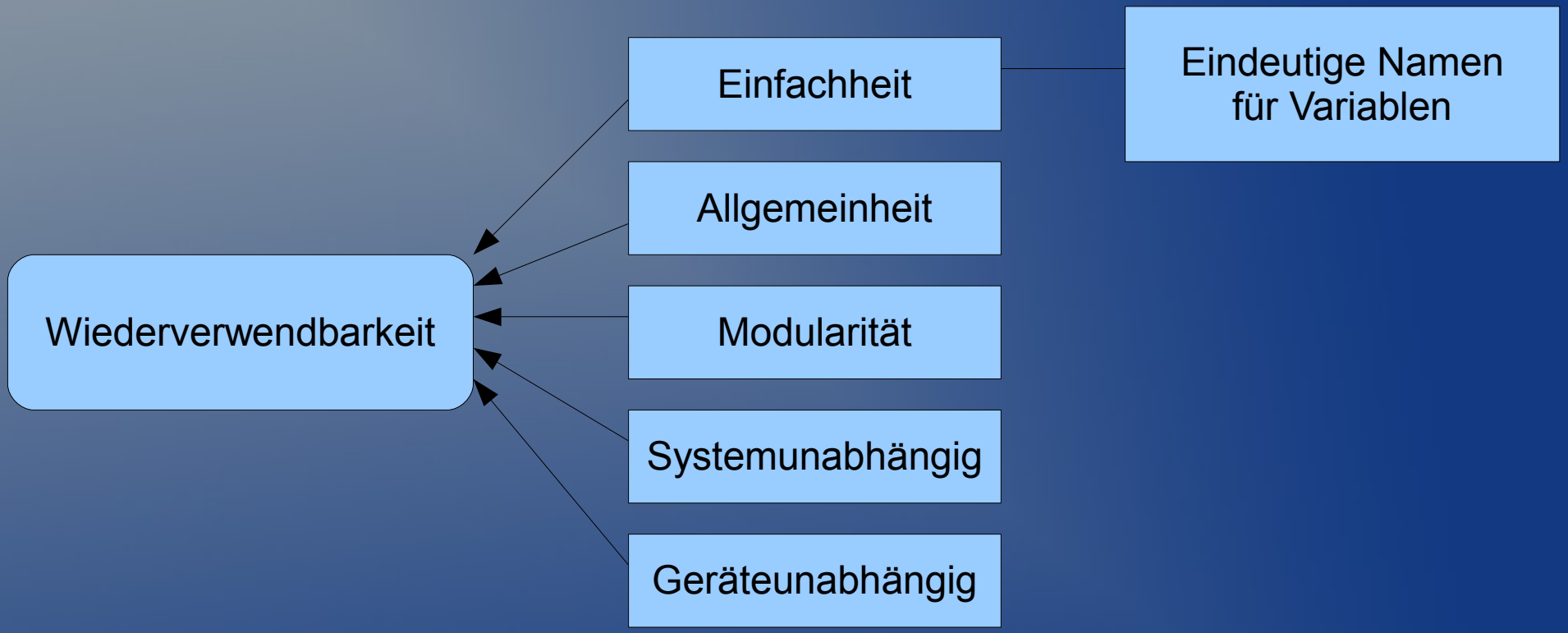
4.2.2. Aufbau des Qualitätsmodells



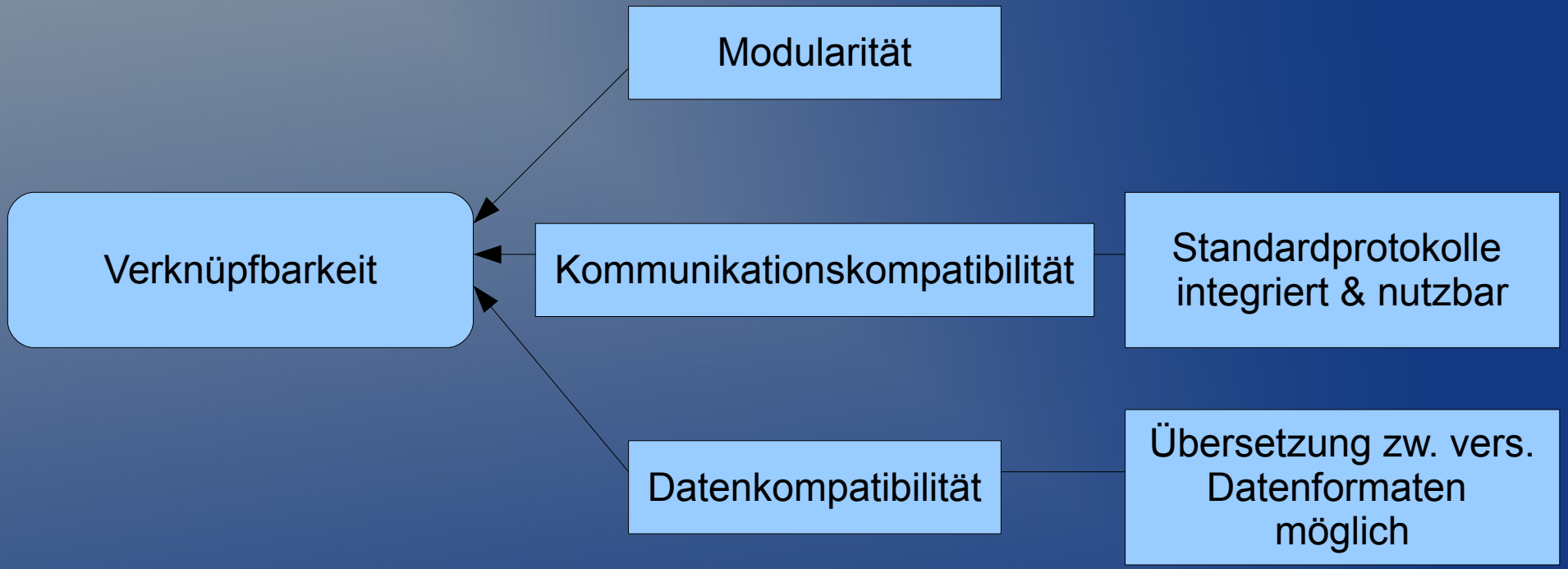
4.2.2. Aufbau des Qualitätsmodells



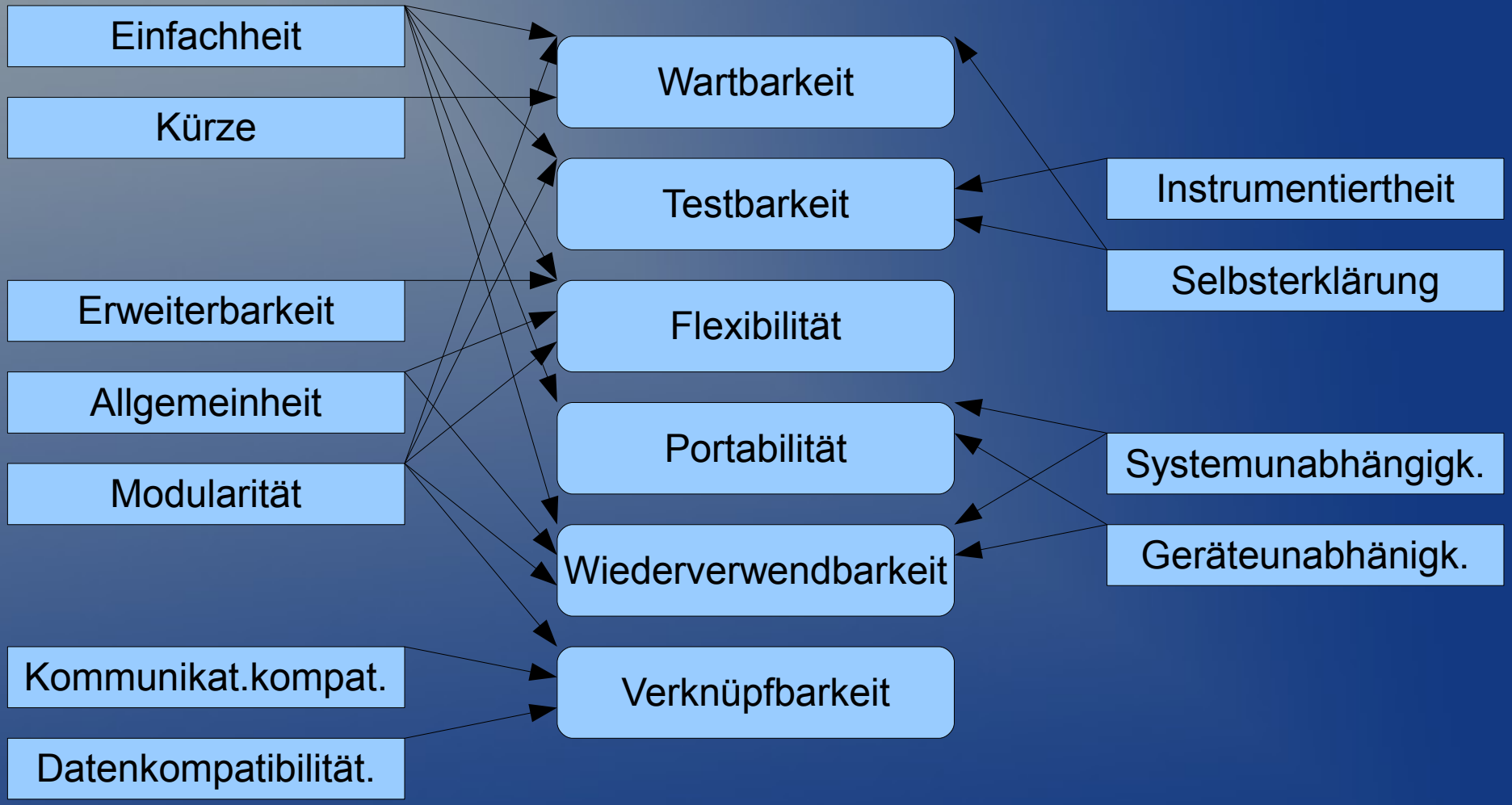
4.2.2. Aufbau des Qualitätsmodells



4.2.2. Aufbau des Qualitätsmodells



4.2.3. Teilausschnitt aus dem Qualitätsmodell nach McCall



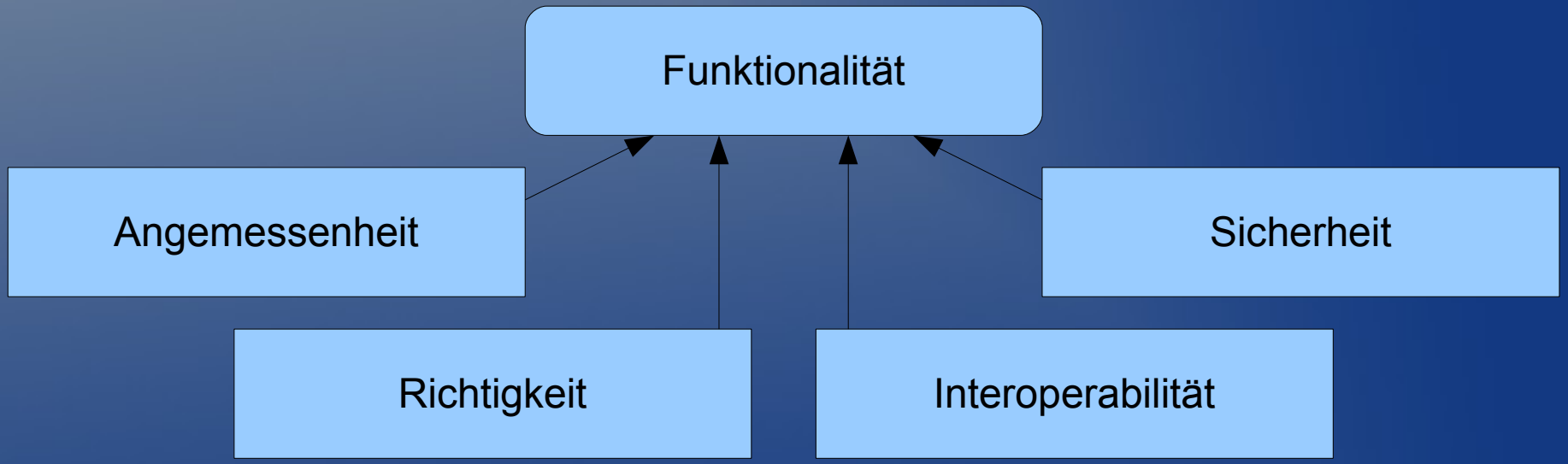
4.3. Qualitätsmodell nach der Norm ISO 9126

1. Bestandteile des Qualitätsmodells
2. Gesamtüberblick über ISO 9126

4.3.1. Bestandteile des Qualitätsmodells

Funktionalität:

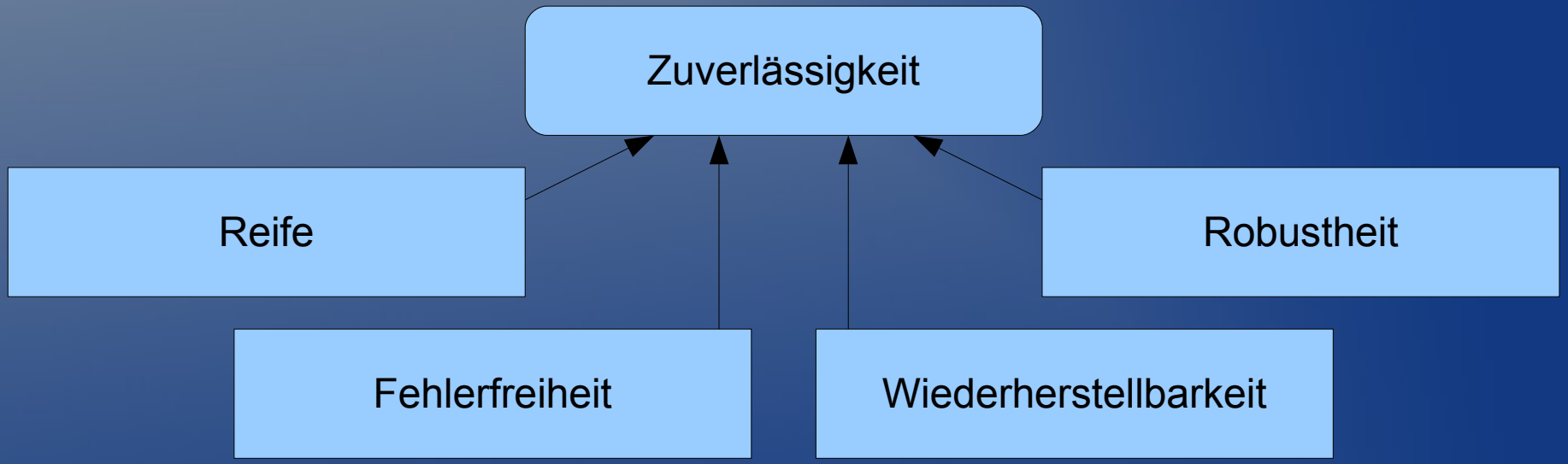
Inwieweit wurden die geforderten und erforderlichen Funktionen implementiert? Sind diese ausführbar?



4.3.1. Bestandteile des Qualitätsmodells

Zuverlässigkeit:

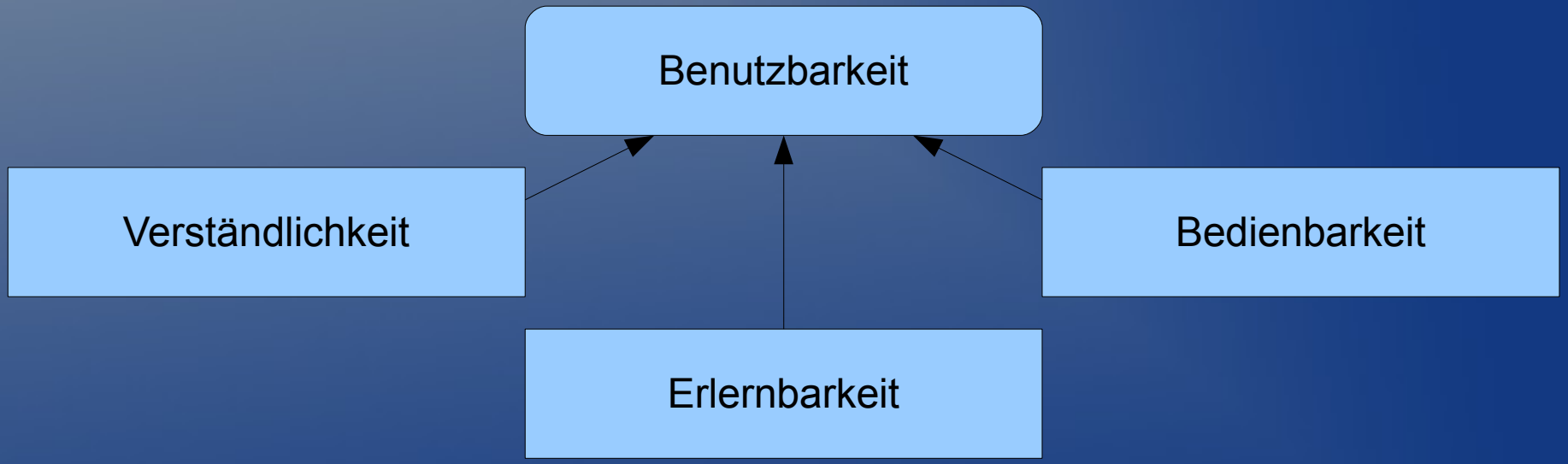
Ist eine angemessene Freiheit von Fehlern garantiert? Kann die Software ein bestimmtes Leistungsniveau halten?



4.3.1. Bestandteile des Qualitätsmodells

Benutzbarkeit:

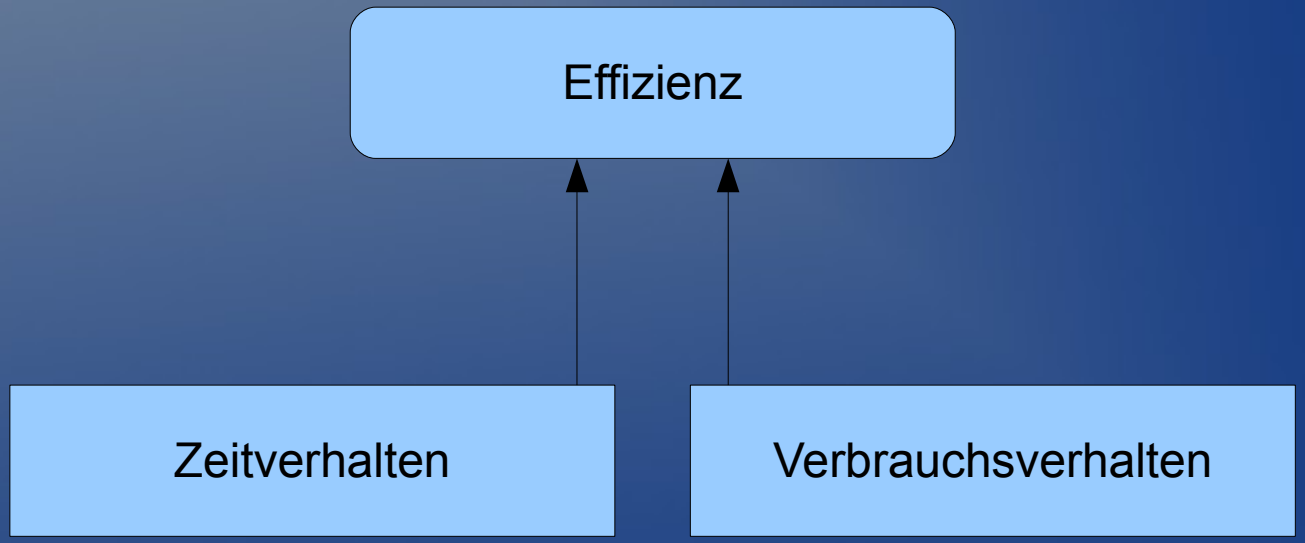
Welcher Aufwand ist nötig, dass der Benutzer das Programm leicht bedienen kann und die Bedienung schnell erlernen kann?



4.3.1. Bestandteile des Qualitätsmodells

Effizienz:

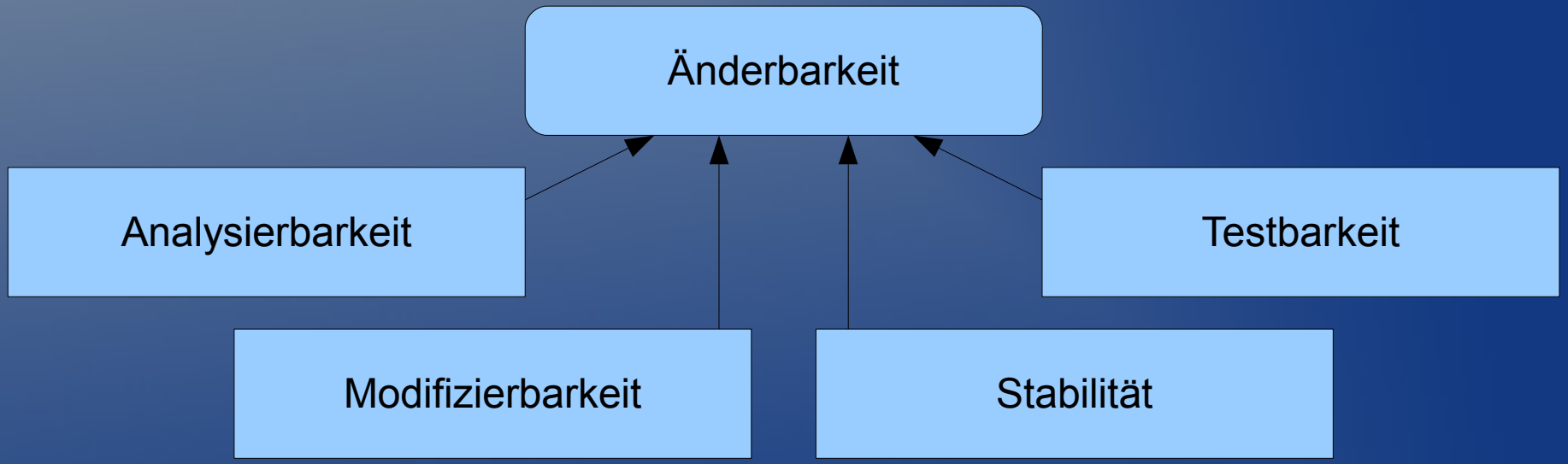
Steht das zeitliche Verhalten im Verhältnis zu den verbrauchten Ressourcen und eingesetzten Betriebsmitteln?



4.3.1. Bestandteile des Qualitätsmodells

Änderbarkeit:

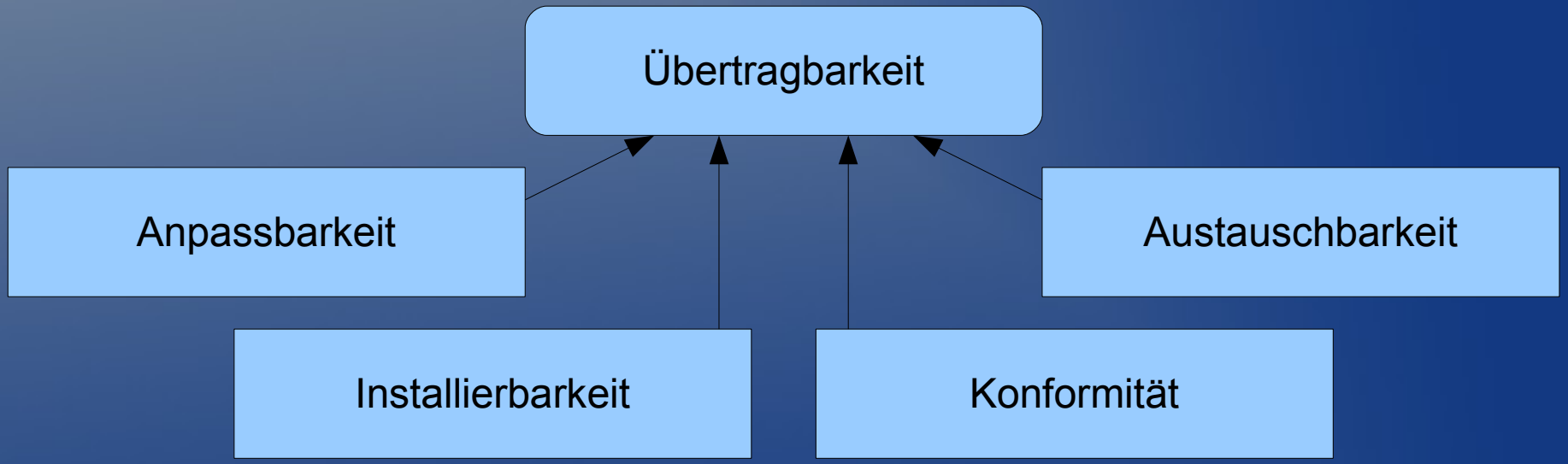
Wie leicht lässt sich das System verbessern oder an eine neue Umgebung anpassen?



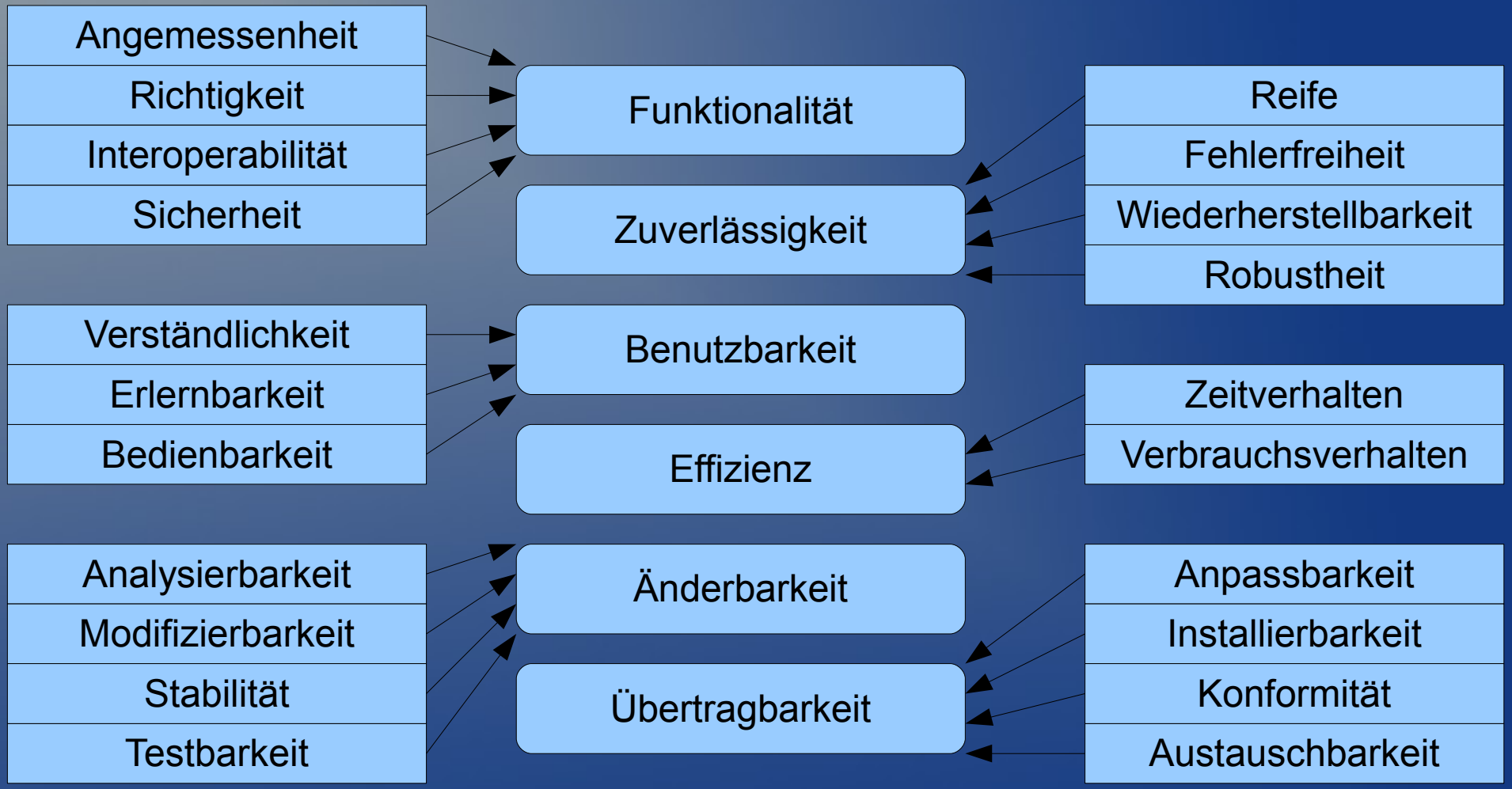
4.3.1. Bestandteile des Qualitätsmodells

Übertragbarkeit:

Ist es realisierbar, die Software auf anderen Hard- oder Softwaresystemen einzusetzen?



4.3.2. Gesamtüberblick über ISO 9126



4.4. Probleme von Softwarequalitätsmodellen

- Standardisierte Qualitätsmodelle → Keine Rücksicht auf individuelle Qualitätsanforderungen von Produkten oder Projekten & fehlender Bezug zu Aktivitäten im Entwicklungsprozess, somit auch zu Kosten und Nutzen
- Zusammenhang zwischen Merkmalen und Teilmerkmalen meist nicht eindeutig
- Entscheidung, welche Eigenschaften zu einer guten Definition von Softwarequalität gehören
- Unpräzise Messungen & Ermittlung von Kenngrößen nicht aussagekräftig genug → Schwierigkeit, die Kriterien meist direkt messen oder bewerten zu können

4.5. Mögliche Lösungen für die Problematik der Qualitätsmodelle

Aktivitätenbasierte Qualitätsmodelle

http://mediatum2.ub.tum.de

/doc/737380/737380.pdf

Wartung

	Modifizieren	Testen	Verstehen (Analysieren)
Strukturiertheit	x	x	x
Kommunikativität		x	
Selbsterklärbark.		x	x
Kürze/Prägnanz			x

4.5. Mögliche Lösungen für die Problematik der Qualitätsmodelle

- Individuelle Festlegung der Qualitätsziele anhand der Anforderungen eines Projektes, d.h. Qualitätsmodelle können angepasst und erweitert werden
- Blick auf den Prozess während der Herstellung eines Produktes, d.h. Prozesskontrolle

Qualitätsmanagement

5. Qualitätsmanagement

- Qualitätsmanagement = alle organisatorischen und technischen Maßnahmen, die der Erschaffung und der Erhaltung von Softwarequalität dienen
- Qualitätsplanung
- Qualitätssteuerung
- Qualitätskontrolle bzw. -prüfung

Literatur

- [KP96] Kitchenham, B., Pfleeger, S.L., Software Quality: The Elusive Target, IEEE Software, v.13 n.1, p.12-21, January 1996
- http://de.wikipedia.org/wiki/Softwarequalit%C3%A4t
- http://de.wikipedia.org/wiki/ISO/IEC_9126
- http://www.ifi.uzh.ch/rerg/fileadmin/downloads/teaching/courses/swq_ss07/kapitel_06_Q_d_e.pdf
- http://user.cs.tu-berlin.de/~sepradm/ws9900/projekt/referate/SQS_folien.pdf
- http://wwwmatthes.in.tum.de/file/Teaching/EIST/public/Teil2_1.pdf
- http://wirtschaftslexikon.gabler.de/Definition/qualitaetssicherung.html
- http://pi.informatik.uni-siegen.de/lehre/2007s/LM/lm_sqmo_20070505_a5.pdf
- http://books.google.de/books?id=7zbSZ54JG1wC&pg=PA10&lpg=PA10&dq=kitchenham+pfleeger+mccall&source=bl&ots=fifrjv61nf&sig=YphXrUXqaFONCuMdZEHt0zg3D_8&hl=de&ei=CpfRS9zpFd-JOOzO4asO&sa=X&oi=book_result&ct=result&resnum=3&ved=0CBEQ6AEwAg#v=onepage&q=kitchenham%20pfleeger%20mccall&f=false
- [MRW77] McCall, Jim, Richards, Paul, Walters, Gene, Factors in Software Quality, November 1977

Schluss

Vielen Dank

für die

Aufmerksamkeit!