

Software-Qualität

Kapitel 2

Qualität fassbar machen

Inhalte

- Fehler finden: Grundeinstellung zum Testen
- Qualitätsanforderungen konkretisieren und prüfen
- Beispiel Usability: Gut bedienbare Software gestalten
- Qualitätsmodelle: Was genau soll Qualität bedeuten?

Prof. Dr. Kurt Schneider



Software-Qualität

Kapitel 2.1

Qualität fassbar machen

Inhalte

- Fehler finden: Grundeinstellung zum Testen
- Qualitätsanforderungen konkretisieren und prüfen
- Beispiel Usability: Gut bedienbare Software gestalten
- Qualitätsmodelle: Was genau soll Qualität bedeuten?

Prof. Dr. Kurt Schneider



Erste Aufgabe: beim Testen helfen



K. Schneider / J.Greenyer

SWQ 2016 - 48



Was ist Testen?

=Def „Ausführen eines Programms mit dem Ziel, Fehler zu finden“

- Das heißt also:
 - das Programm muss ausgeführt werden
 - jeder gefundene Fehler ist ein Erfolg für den Tester
 - Für den Entwickler nur indirekt
- Testen ist also nicht:
 - Wunsch zu zeigen, dass das Programm in Ordnung ist
 - Herumprobieren-Ändern-wieder Probieren
 - ein unmittelbar konstruktiver Verbesserungsschritt
- Fazit: Testen erscheint zunächst destruktiv
 - verlangt gewisse Schadenfreude: Autor hat die nicht!
 - enthält viele kreative Elemente (was könnte schiefgehen?)

K. Schneider / J.Greenyer

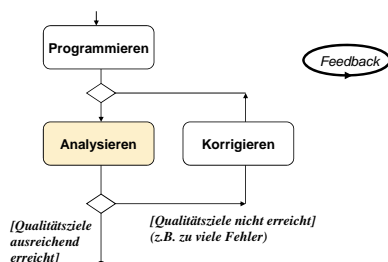
SWQ 2016 - 49



Analytische Qualitätssicherung

wie trägt Test zu besserer SW-Qualität bei?

- Analytische Maßnahme alleine verbessert die Qualität nicht
- Sie ist aber in einen Q-Prozess eingebunden, der es tut



K. Schneider / J.Greenyer

Höhere Qualität, besseres Vertrauen

SWQ 2016 - 50



Tätigkeiten und Dokumente beim Testen

Vorgehensweise

- Test planen
 - Resultat: Testplan
 - Vorgehen: wer tut wann was?
 - Testgeschirr bzw. Testrahmen
- Testfälle erstellen
 - Resultat: Testfallbeschreibung
 - Setup, Voraussetzungen
 - Eingaben und Sollausgaben
- Tests nach Plan durchführen
 - Resultat: Testberichte
- Gefundene Fehler zusammenfassen
 - Resultat: Fehlerberichte

Wichtig:
Tests dokumentieren

Resultate: Testplan bis Fehlerbehebungsberichte

K. Schneider / J.Greenyer

Siehe auch: Zister, Bittl, Grechenig, Köhle (2001): Software Engineering mit UML und dem Unified Process, Pearson Studium

SWQ 2016 - 51

SE Einige Definitionen

Prinzip

Prüfling =_{Def} zu prüfende Software (nicht: deren Autor!)

Autor =_{Def} Verfasser des Prüfings

Laufzeitversuch =_{Def} Entwickler spielt herum, um zu sehen, „ob Programm läuft“
selbstverständliche Voraussetzung für Test - nicht mehr

Test =_{Def} Ausführung eines Programmes mit dem Ziel, Fehler zu finden
„Test“ wird von manchen auch anders verstanden: oft Missverständnisse

Testvorschrift: Anweisung, wie Test abzuwickeln ist (inkl. Setup)

Testprotokoll: standardisiertes Formular zur Doku. über jeden gefundenen Fehler

SWQ K. Schneider / J.Greenyer SWQ 2016 - 52

SE Was ist ein Fehler?

Prinzip

„Wenn sich Soll- und Ist-Ergebnis (bzw. -verhalten) unterscheiden, liegt ein Fehler vor“

Symptom/Anzeichen für Fehler also:

- Unterschied zwischen Soll und Ist
- Kein-Unterschied heißt aber nicht: Kein-Fehler

Fehler finden: nur falls ein Unterschied feststellbar

- Also falls Soll bekannt, Ist messbar und Abweichung erkennbar

Rumprobieren (auch von Fachleuten)
ohne definiertes Soll ist kein Test

Debugging (Tüfteln zur Fehlerbeseitigung)
folgt auf das Testen, ist aber selbst kein Test

- Programm muss natürlich vor Test ablauffähig sein
- Test beseitigt (zunächst) keinen Fehler

Nicht alle Programme sind gleich gut testbar (Qualitätsaspekt „Testbarkeit“)

SWQ K. Schneider / J.Greenyer SWQ 2016 - 53

SE Fehler in Tests

Hintergrund

Abweichung von Soll und Ist bedeutet Fehler in

falschem	Ist	(„falsche Ausgabe“)
oder falschem	Soll	(„falsche Vorgabe“)
oder ungeeignetem	Vergleich	(„falscher Test“)

Nicht jede Abweichung ist daher Indiz für **Programmf Fehler**
obwohl man die eigentlich sucht!

Andere Möglichkeiten

- Testfall war falsch Programm korrekt, Soll falsch aus Anf. ermittelt
- Anforderung falsch interpretiert Soll korrekt ermittelt, Anf. missverstanden
- Vergleich entspricht nicht den Erfordernissen
Toleranz, Kriterien oder Typ falsch

Problem: woher weiß man, welcher Fall vorliegt?

SWQ K. Schneider / J.Greenyer SWQ 2016 - 54

SE Beispiel: Parkscheinautomat

Vorschau auf eine spätere Testaufgabe

„Wer parken will, kauft sich einen Parkschein für max. 2 Stunden und legt ihn unter die Windschutzscheibe“

Parkplatz Schlossgasse
Ihre Parkzeit endet
Mi 11:04

• Auf dem Parkschein soll stehen, wann die Parkzeit endet
– Ein Programm soll das ausrechnen

• Machen Sie dafür gute Testfälle!

• Übrigens:
Testfälle kann man schon vor dem Programm selbst entwickeln!

SWQ K. Schneider / J.Greenyer SWQ 2016 - 55

SE Anforderungen muss man genau kennen

Vorschau auf das Beispiel Parkscheinautomat

- Welches Geld nimmt der Automat?
– Euro-Münzen
- Welche Stückelung?
– 1€, 50c, 20c, 10c
- Wieviel kostet das Parken?
– „sagen wir: 1,10€ pro Stunde“
Wg. Rundung: 1,20€ pro Stunde
- Was passiert bei Einwurf über 2,40€?
– Geld verfällt, keine Rückgabe
- Wann ist der Automat in Betrieb?
– 24h
– Wann muss man für das Parken zahlen?
– Zwischen 9 und 19 Uhr

Man kann sich beim SOLL-Wert durchaus verrechnen

Hilfreich: ähnlichen Parkschein beschaffen (z.B. aus anderer Stadt)

Unklarheiten über Requirements führen zu unklaren/falschen Testfällen

SWQ K. Schneider / J.Greenyer SWQ 2016 - 56

SE Testplanung

Vorgehensweise

- Im Prinzip: Wer führt wann welche Testfälle durch
 - Testmethoden (Black-/White-Box-Test, Regressionstest usw.)
 - Konkrete Testfälle (Eingabe, Soll-Ausgabe)
 - Konfiguration der Testplattform (auch Testdaten in Datenbank)
 - Bereitstellung des Testgeschirrs („Hilfsgerüst für die Testfälle“)
- Kriterien für den Start der Testdurchführung
 - Version für den Test freigegeben
- Kriterien für Testunterbrechung
 - Normalerweise wird bei Fehlern weitergetestet
 - Kriterien, wann das nicht mehr sinnvoll ist
- Kommunikationswege
 - Wer erfährt in welcher Form von Testergebnissen?
 - Was tun diese Rollen daraufhin (selbständig); korrigieren sie?

SWQ K. Schneider / J.Greenyer Siehe auch: Ziser, Bittl, Grechting, Köhle (2001): Software Engineering mit UML und dem Unified Process, Pearson Studium. SWQ 2016 - 57

Erinnerung: NoRisk-App

Ein Beispiel; diese Folie haben Sie schon gesehen

- FunGate entwickelt seit Jahren Software für Versicherungsgesellschaft
 - Großrechner in Zentrale
 - Desktop-SW NoRisk für Kunden-Akquise
 - Geplant: App-Version von NoRisk
 - Ziel: Kundenberater mit Smartphones flexibler machen!
- FunGate hat guten Ruf zu verlieren - das wird kein Kinderspiel!




K. Schneider / J. Greenyer SWQ 2016 - 58

Testfälle erstellen

Prinzip

- Black-Box-Test: Aus der Spezifikation**
 - zu allem, was in Spezifikation verlangt ist (**und nur dazu!**), muss geprüft werden, ob es wirklich so funktioniert
 - Tut die SW, was sie soll - und sonst nichts?
 - Spezifikation muss dazu die **Soll-Werte** vorgeben
 - Soll ist prinzipiell nicht aus dem Code ableitbar!
 - SW-Verhalten muss Spezifikation genügen
 - Egal, was „intern“ passiert



K. Schneider / J. Greenyer SWQ 2016 - 59

BlackBox-Tests

Beispiele von NoRisk

- Spezifikation enthält Tariftabelle für Versicherungspolizen
 - Bereich: Risikosportarten
 - Große Tabelle, daraus zu entnehmen:

Risikosportarten: Privat-Unfallversicherung	Tarif: NoRisk Daring		
Alter	15-24	25-35	Senioren (ab 36)
Luftsportarten*	279,00 €	250,00 €	200,00 €
Wassersportarten*	99,00 €	200,00 €	289,00 €
Kombi SuperDaring	349,00 €	399,00 €	444,00 €

* im Glossar ist erklärt:

- Luft-Sportarten:** Fallschirmspringen, Paragliding, Segelflug (ohne Hilfsmotor)
- Wasser-Sportarten:** Rafting, Canyoning, Wasserski, Tauchen. Nicht: Schnorcheln, Turmspringen, Schwimmen

K. Schneider / J. Greenyer SWQ 2016 - 60

Black-Box-Sicht:

Sichtbares Verhalten korrekt?



K. Sc 1

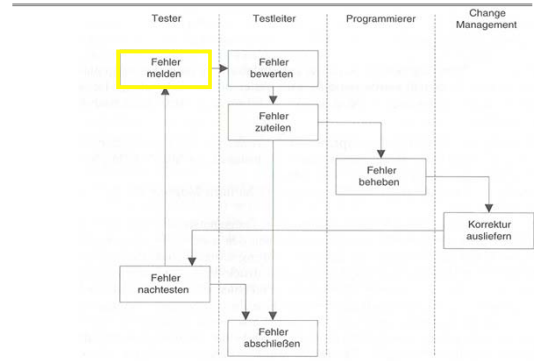
Konsequenz: Ohne Spezifikation kein Test

- Soll-Werte nur aus Spezifikation
- Oft sind Spezifikationen aber zu vage, unvollständig, unklar
- Das führt oft zu absurden Aussagen (z.B. Sommerville):
 - „In den meisten ... Fällen jedoch ist das Testen ein intuitiverer Vorgang, weil zum Schreiben detaillierter Spezifikationen ... die Zeit nicht ausreicht.“
 - „Das Testen beruht gewöhnlich auf einem intuitiven Verständnis der gewünschten Arbeitsweise der Komponente.“ **Bitte nicht!**
- Denn viele neigen zur „selbstherrlichen Verfeinerung“
 - Testen gegen „selbst-erfundene“ Anforderungen (der Tester!) ist sinnlos
 - Intuition der Entwickler ist hier einmal nicht gefragt
- Bei Spezifikationsmängeln ist allein der Kunde die Referenz!
 - Nur der Kunde darf Anforderungen präzisieren
 - Gute Idee: Kunden mit „intuitiv“ entstandenen Testfällen konfrontieren!

K. Schneider / J. Greenyer SWQ 2016 - 62

Test durchführen

Ablauf und Rollen



K. Schneider / J. Greenyer SWQ 2016 - 63

SWQ

SE

Sehr wichtig, aber ganz anders: Abnahmetest

- Ein wichtiger Tag! Juristisch und wirtschaftlich.
- Sehr aufregend für alle Beteiligten.
Muss gut vorbereitet sein. Man hofiert den Kunden.
- Zuvor: Kunde hatte Abnahmetestfällen zugestimmt.
- Beim Abnahmetest: Sie werden durchgeführt
- Ergebnis: Je nach Verlauf...
 - Software „wird abgenommen“, bezahlt, Wartung beginnt
 - **ODER nicht abgenommen:**
 - Noch keine Bezahlung; Kosten laufen aber weiter
 - Vertrag nicht erfüllt; Projektleiter nicht entlastet
 - Schnell Nacharbeit, oft abends und am Wochenende
 - Oft sehr großer Druck durch Image- und finanz. Schaden!

Abnahme ist eines der wichtigsten Ereignisse in einem Projekt

SWQ

K. Schneider / J. Greenyer SWQ 2016 - 64

SWQ

Software-Qualität

Kapitel 2.2

Qualität fassbar machen

Inhalte

Fehler finden: Grundeinstellung zum Testen

Qualitätsanforderungen konkretisieren und prüfen

Beispiel Usability: Gut bedienbare Software gestalten

Qualitätsmodelle: Was genau soll Qualität bedeuten?

SE

Prof. Dr. Kurt Schneider

SWQ

Q arbeitet sich ein: Qualitäts-Anforderungen

Alle reden von „guter Qualität“.
Aber was meinen die eigentlich?
Meinen die Kunden dasselbe,
und woher wissen wir das?

SWQ

K. Schneider / J. Greenyer SWQ 2016 - 66

SWQ

SE

Qualität definiert

EN ISO 8402
„Qualität ist die Gesamtheit von Merkmalen einer Einheit bezüglich ihrer Eignung, festgelegte und vorausgesetzte Erfordernisse zu erfüllen.“

DIN 55350-11 95
„... festgelegte oder abgeleitete Erfordernisse (Qualitätsanforderungen)...“

- **Einheit:**
 - Produkt, Dienstleistung, Tätigkeit, Prozess, Organisation - Software
- **Vorausgesetzte/abgeleitete Erfordernisse:**
 - Schwerer zu fassen, implizit. Wer beurteilt Erfüllung im Konfliktfall?
- **Qualität ist damit nur in Bezug auf Ziele definiert.**
 - Gleiches Produkt/Prozess/Projekt, andere Ziele => Q. ändert sich!
 - Nachbarschaft zu Req. Engineering: Q-Anforderungen gehören dazu

SWQ

K. Schneider / J. Greenyer, Martin Glitz: Vorlesung Software-Engineering, Kap. 19, Universität Zürich, 2007 SWQ 2016 - 67

SWQ

SE

Schlechte SW-Qualität im Alltag

persönliche Erfahrungen

- Automatic Teller Machine
– lästige Präzision
- United Airlines Auskunftssystem
– Angriff aus dem Telefon
- Planungssystem
– Tanz ums Datum
- On-line Überweisung
– Wann ist der Fehler passiert?
- Essenskarte aufladen
– auch SW soll höflich sein

Seite zurück-gesetzt.
(und gebucht?)

SWQ

K. Schneider / J. Greenyer SWQ 2016 - 68

SWQ

SE

Q-Begriffe

- **Qualitätsaspekt bzw. -merkmal** =_{Def} „Eigenschaften einer Einheit, anhand derer ihre Qualität beschrieben und beurteilt wird.“
 - Jedoch keine Aussage über Grad der Ausprägung
 - Kann über mehrere Stufen von Teilmerkmalen verfeinert werden
- **Qualitätsziel** =_{Def} Angestrebtes Qualitätsmerkmal
 - Mit oder ohne konkret angestrebtem Erreichungsgrad/Ausprägung
- **Qualitätsanforderung** =_{Def} Ein oder mehrere Qualitätsziele
- **Qualitätsmetrik** =_{Def} Maß für Ausprägung eines Q-Aspekts

SWQ

K. Schneider / J. Greenyer SWQ 2016 - 69



ISO 9126 → ISO 25 000

Bekannte Norm ist durch neue ersetzt



- **Funktionalität**
 - Angemessenheit
 - Richtigkeit
 - Interoperabilität
 - Ordnungsmäßigkeit
 - Sicherheit
- **Zuverlässigkeit**
 - Reife
 - Fehlertoleranz
 - Wiederherstellbarkeit
- **Benutzbarkeit**
 - Verständlichkeit
 - Erlernbarkeit
 - Bedienbarkeit

- **Effizienz**
 - Zeitverhalten
 - Verbrauchsverhalten
- **Änderbarkeit**
 - Analysierbarkeit
 - Modifizierbarkeit
 - Stabilität
 - Prüfbarkeit
 - Austauschbarkeit
- **Übertragbarkeit**
 - Anpassbarkeit
 - Installierbarkeit
 - Konformität

K. Schneider / J. Greenyer SWQ 2016 - 70

Software-Qualität

Kapitel 2.3


Usability Engineering


ein Beispiel für einen Qualitätsaspekt

Inhalte


Bedeutung guter Bedienbarkeit
 Grundkonzepte von Usability
 Aspekte der Benutzerfreundlichkeit
 ISO 9241, Teil 10
 Konstruktives Usability Engineering
 Experten-Evaluationen

Prof. Dr. Kurt Schneider






Übersicht




- **Bedienbarkeit, Usability, Ergonomie: was ist das?**
 - Definitionen
 - Qualitätsaspekte und Charakterisierungen
 - Messbare Größen
- **Bedienbare Software gestalten: konstruktiv**
 - Referenzmodell Usability Engineering (Offergeld, nach Mayhew)
 - Worauf muss man achten?
 - Beispiele für Gestaltungsregeln
 - Direkte Hilfe für die Entwicklung
- **Zusätzlich angesprochen:**
 - Nutzen
 - Häufige Probleme
 - Forschungsfragen

K. Schneider / J. Greenyer SWQ 2016 - 72




Wo Usability besonders wichtig ist

Beispiele Internet und eBusiness




- **Immer mehr kommerzielle, webbasierte Systeme**
 - eCommerce für Endkunden (Banken, Bücher etc.)
 - eBusiness (B2B, B2C, internal Business)
- **Konsequenzen**
 - **User Interface ist das "Gesicht des Unternehmens" zum Kunden**
 - Prägt Image, Unternehmen wird damit identifiziert
 - Weiche Bindungs-/Ablehnungsfaktoren durch "Digitalen Verkäufer"
 - "Konkurrenz ist nur einen Mausklick entfernt"
 - Auftritte sind leichter vergleichbar
 - Wechsel zur Konkurrenz ist viel einfacher
- **Fazit**
 - Im Web hängt Geschäftserfolg stärker von Usability ab
 - Schlecht bedienbare Programme gefährden Firmenbestand

K. Schneider / J. Greenyer SWQ 2016 - 73

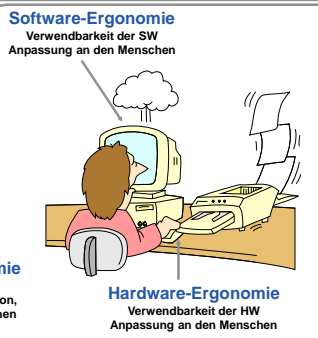


Bedienbarkeit und Ergonomie




Ergo (gr.): „Arbeit“


Ergonomie:
(nach Offergeld; unvollständig): „Wissenschaft von der Arbeit“
(nach Nievergelt): „Wissenschaft von der Anpassung der Arbeit an den Menschen“



K. Schneider / J. Greenyer SWQ 2016 - 74



Software-Ergonomie



Ergonomie
(Anpassung der Arbeit an den Menschen)

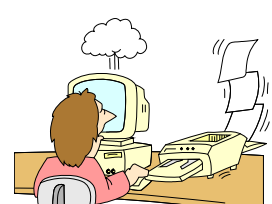
durch

- Schnittstellen („Benutzbarkeit“)
- Funktionalität („Nützlichkeit“)
- Korrektheit („Verlässlichkeit“)

Ziel:
Zusammenspiel aller Komponenten optimieren, die die Arbeitssituation von Computernutzern bestimmen:

- Mensch
- Aufgabe
- Technik
- Org. Rahmen

Nicht nur
Präsentationsaspekte interaktiver Software



K. Schneider / J. Greenyer SWQ 2016 - 75

Usability („Gebrauchstauglichkeit“)

Definition laut ISO 9241, Teil 11

Usability = *Def*
Der Grad, zu dem ein Produkt oder System durch *definierte Benutzer* verwendet werden kann, um *spezielle Ziele*, nämlich

- Effektivität
- Effizienz
- Zufriedenheit

in einem *bestimmten Nutzungskontext* zu erreichen.

Achtung: Diese Usability-spezifischen Definitionen weichen vom allg. Sprachgebrauch etwas ab

Effektivität = *Def*
„Genauigkeit u. Vollständigkeit, mit der Benutzer best. Ziele erreichen.“

Effizienz = *Def*
„Die von den Benutzern aufgewendeten Mittel im Verhältnis zur Genauigkeit und Vollständigkeit, mit der die Benutzer ihre Ziele erreichen.“

Zufriedenheit = *Def*
„Die Freiheit von Unbehagen und Beschwerden sowie die positive Einstellung zur Nutzung des Produkts oder Systems.“

K. Schneider / J. Greenyer SWQ 2016 - 76

Checkliste „Gutes Design“

Wann ist etwas gut bedienbar/usable?

- Kann die Funktion des Gerätes leicht bestimmt werden?
- Ist klar, welche Aktionen ausgeführt werden können?
- Ist klar, wie eine Absicht in physische Aktionen bzw. Kommandos umgesetzt wird?
- Ist erkennbar, wie die Aktion tatsächlich ausgeführt wird?
- Ist leicht zu erkennen, ob das Gerät im gewünschten Zustand ist?
- Kann der Benutzer leicht den wahrgenommenen Zustand interpretieren?
- Wird der Systemzustand so dargestellt, dass ein Vergleich mit den Zielen des Benutzers leicht fällt?

K. Schneider / J. Greenyer SWQ 2016 - 77

Kriterien menschengerechter Arbeit

Menschengerechte Arbeit

Benutzerfreundlichkeit

Individuum

Benutzerfreundlichkeit

Dialogsystem Benutzer

Funktionalität Benutzung

Sozialverträglichkeit

Gruppe

Sozialverträglichkeit

Dialogsysteme Team

Sicherheit Kooperation

Dafür gibt es definierte Kriterien

Dafür eher nicht

K. Schneider / J. Greenyer Siehe: Vorlesung von Michael Offergeld an der Universität Ulm, WS 2002 SWQ 2016 - 78

Kriterien der Benutzerfreundlichkeit

Qualitätsaspekte und Charakterisierungen

Verfügbarkeit

Aufgabenangemessenheit

Orientierung

Beeinflussung

Verfügbarkeit

Nützlichkeit

Komfort

Effektivität

Effizienz

Übersichtlichkeit

Selbstbeschreibungsfähigkeit

Erwartungskonformität

Fehlertoleranz

Lernförderlichkeit

Individualisierbarkeit

Steuerbarkeit

Funktionalität

Benutzbarkeit

K. Schneider / J. Greenyer Siehe: Vorlesung von Michael Offergeld an der Universität Ulm, WS 2002 SWQ 2016 - 79

EN ISO 9241, Teil 10 - Übersicht

„Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten“

Grundsätze

- Aufgabenangemessenheit
- Selbstbeschreibungsfähigkeit
- Steuerbarkeit
- Erwartungskonformität
- Fehlertoleranz
- Individualisierbarkeit
- Lernförderlichkeit

und ihre Anwendung

- Bei Gestaltung und Bewertung von Dialogsystemen.
- Nur allgemeine Leitlinien.
- Umsetzung hängt ab von
 - den Merkmalen des Benutzers,
 - den Arbeitsaufgaben,
 - der Arbeitsumgebung
- eine Vorentscheidung bildet oft die Wahl der **Dialogtechnik/Dialogführung**:
 - mittels Menüs
 - mittels Kommandosprachen
 - mittels direkter Manipulation
 - mittels Bildschirmformularen

K. Schneider / J. Greenyer SWQ 2016 - 80

Kriterien der Benutzerfreundlichkeit

Beispiel: Steuerbarkeit

Aufgabenangemessenheit

Selbstbeschreibungsfähigkeit

Steuerbarkeit

Erwartungskonformität

Fehlertoleranz

Individualisierbarkeit

Lernförderlichkeit

Ein Dialog ist **steuerbar**, wenn = *Def* der Benutzer in der Lage ist, den Dialogablauf zu starten sowie seine Richtung und Geschwindigkeit zu beeinflussen bis das Ziel erreicht ist.

Mögliche Teilaspekte:

- Darstellungsformate
- Dialogfluß
- Interaktionsgeschwindigkeit
- Feedback der Anwendung
- Verwendung von Werkzeugen
- Verhalten in kritischen Situationen

K. Schneider / J. Greenyer SWQ 2016 - 81

Kriterien der Benutzerfreundlichkeit

Beispiel: Erwartungskonformität

- Aufgabenangemessenheit
- Selbstbeschreibungsfähigkeit
- Steuerbarkeit
- Erwartungskonformität
- Fehlertoleranz
- Individualisierbarkeit
- Lernförderlichkeit

Ein Dialog ist **erwartungskonform**, wenn =_{Def}

- er konsistent ist,
- den Merkmalen des Benutzers entspricht, z.B.
 - seinen Kenntnissen aus dem Arbeitsgebiet,
 - seiner Ausbildung und
 - seiner Erfahrung, sowie
- den anerkannten Konventionen entspricht, z.B.
 - anerkannten Styleguides (z.B. Windows-Styleguide),
 - allgemein verwendeten Gestaltungsregeln (Links im Internet).

Mögliche Teilaspekte:

- einheitliches Dialogverhalten,
- Ähnlichkeit von Dialogen bei ähnlichen Aufgaben
- Rückmeldungen über den Bearbeitungsstand

K. Schneider / J. Greenyer SWQ 2016 - 82

Kriterien der Benutzerfreundlichkeit

Beispiel: Individualisierbarkeit

- Aufgabenangemessenheit
- Selbstbeschreibungsfähigkeit
- Steuerbarkeit
- Erwartungskonformität
- Fehlertoleranz
- Individualisierbarkeit
- Lernförderlichkeit

Ein Dialog ist **individualisierbar**, wenn =_{Def}

- das Dialogsystem Anpassungen an die Erfordernisse der Arbeitsaufgabe und an die individuellen Fähigkeiten und Vorlieben des Benutzers zulässt.

Anpassungen aber nur innerhalb bestimmter Grenzen (Benutzer nicht beeinträchtigen).

Mögliche Teilaspekte:

- Alternative Bedien- u. Darstellungsformen anbieten
 - Navigation für Laien- vs. Expertenbenutzer
 - grafische vs. textuelle Darstellungen
- Individueller Dialogeinstellungen erlauben
 - Layout / Bildschirmaufteilung
 - Farben und Schriften

K. Schneider / J. Greenyer SWQ 2016 - 83

Kriterien der Benutzerfreundlichkeit

Beispiel: Lernförderlichkeit

- Aufgabenangemessenheit
- Selbstbeschreibungsfähigkeit
- Steuerbarkeit
- Erwartungskonformität
- Fehlertoleranz
- Individualisierbarkeit
- Lernförderlichkeit

Ein Dialog ist **lernförderlich**, wenn er den Benutzer beim Erlernen des Dialogsystems unterstützt und anleitet.

Mögliche Teilaspekte:

- Regeln und zugrundeliegende Konzepte dem Benutzer zugänglich machen
 - Aufbau von Ordnungsschemata,
 - Aufbau von Merkregeln
- Unterstützt relevante Lernstrategien
 - Learning by Doing,
 - Lernen am Beispiel
- Unterstützt das Wiederauffrischen von Gelerntem

K. Schneider / J. Greenyer SWQ 2016 - 84

Klassifikation von Usability-Zielen

Alle quantitativen Usability-Ziele (incl. ease-of-use, ease-of-learning)

Absolutes Maß objektiv Relatives Maß subjektiv

Performance-Ziele
quantifizieren die **Leistung des Benutzers** bei Erledigung einer Aufgabe. Dabei werden Zeit und Fehler gemessen.

Preference-Ziele
Auf Basis der **Vorliebe des Benutzers** für eine von mehreren Oberflächenalternativen (Voraussetzung: Benutzer hat damit Erfahrung).

absolut subjektiv


Satisfaction-Ziele
Vom Benutzer geäußerte (absolute) Zufriedenheit mit einer bestimmten Oberfläche.

Bei den **Preference-/Satisfaction-Zielen**:
subjektive Reaktionen des Benutzers sind zwar nicht objektiv, aber messbar.

K. Schneider / J. Greenyer SWQ 2016 - 85

Usability-Ziele messbar machen

- Mögliche Metriken für Performance-Ziele:
 - Zeit, um eine Aufgabe abzuschließen
 - Zeit, um eine Aufgabe abzuschließen, nachdem das System eine bestimmte Zeit nicht verwendet wurde
 - Anzahl und Arten von Fehlern pro Aufgabe
 - Anzahl an Fehlern innerhalb einer Zeitspanne
 - Anzahl an Zugriffen auf Online-Hilfe oder Benutzerhandbücher
 - Anzahl an Benutzern, die einen besonderen Fehler machen
 - Anzahl an Benutzern, die eine Aufgabe erfolgreich abgeschlossen haben
- Erfahrener Qualitätsexperte entgegnet:
„Ok, das ist messbar.
Was (davon) interessiert uns und wieso?“



K. Schneider / J. Greenyer SWQ 2016 - 86

Q will Usability gleich anfangs einbauen



K. Schneider / J. Greenyer SWQ 2016 - 87

Konstruktiv: Gestalten

- Bisher: Qualitätsaspekt „Usability / Bedienbarkeit“ geklärt
 - Ergonomie ist verwandt
 - Kann man für analytische oder konstruktive Zwecke einsetzen
- Jetzt: Bedienbare Software gestalten (=konstruktiv)
- Usability bei
 - Vorgehensweise
 - GUI-Auswahl
 - Styleguide
- Nutzen: Positive Auswirkungen
- Wichtig: Usability von Projektanfang an mit-gestalten!

K. Schneider / J.Greenyer SWQ 2016 - 88

„Was kann man überhaupt gestalten?“ Schalenmodell

The Schalenmodell (Shell Model) illustrates the layers of design in software engineering. It consists of concentric semi-circular layers representing different levels of abstraction and design focus:

- Hardware:** Arbeitsplatz, Arbeitsumgebung
- Ein-/Ausgabe:** Dialog
- Werkzeug:** Mensch-Rechner, Funktionsabteilung
- Mensch-Mensch:** Funktionsabteilung, Gestaltung der Arbeitsabläufe

The outermost layer is labeled **Gestaltungsebene**. A large arrow points from the inner layers towards the outer layers, indicating the progression of design. The right side of the diagram is labeled **Gestaltungsrichtung** and **Gestaltungsspielraum**. The bottom right corner indicates the **Software** and **organisatorischer Bereich**.

K. Schneider / J.Greenyer Siehe: Vorlesung von Michael Offergeld an der Universität Ulm, WS 2002 SWQ 2016 - 89

Häufige Probleme mit Usability

- Am schlimmsten: UE findet gar nicht statt
- Nicht viel besser: UE soll gegen Projektende Bedienprobleme lösen, „Feuer löschen“
- Oder zu ambitioniertes UE: entwickelt Eigenleben im Projekt
 - Usability - war da sonst noch was?
 - Projekt leidet unter schlecht synchronisierten Teilprozessen
- Daher wichtig
 - Verzahnung von UE und Entwicklung von Anfang an planen
 - Durch Maßnahmen unterstützen

K. Schneider / J.Greenyer SWQ 2016 - 90

Diagnoseorientierung

„Analytische QS“ ist meist das höchste der Gefühle

The diagram illustrates the **Systementwicklungsprozess** (System Development Process) with five phases:

- Phase 1: Projekt-vorbereitung
- Phase 2: Anforderungs-analyse
- Phase 3: Entwurf
- Phase 4: Entwicklung Integration und Tests
- Phase 5: Überleitung in die Nutzung

A feedback loop labeled **Software-ergonomische Evaluation** leads from Phase 5 back to Phase 1, with a note: **teures Redesign in späten Entwicklungsphasen**. The final outcome is labeled **Endprodukt mit mangelhafter ergonomischer Güte**.

> Keine durchgängige Berücksichtigung ergonomischer Aspekte
> Keine Beteiligung der späteren Benutzer

K. Schneider / J.Greenyer Siehe: Vorlesung von Michael Offergeld an der Universität Ulm, WS 2002 SWQ 2016 - 91

Konstruktive Usability-Qualitätssicherung durch iteratives Vorgehen

The diagram shows the **Systementwicklungsprozess** (System Development Process) and the **User-Interface-Prozessmodell** (User Interface Process Model) integrated into it. The system development process includes phases: Projekt-vorbereitung, Anforderungs-analyse, Entwurf, Entwicklung, and Installation/betriebnahme. The UI process model includes: Projekt-vorbereitung, Anforderungs-analyse, Entwurf, Evaluation & Test, and Überleitung in die Nutzung. The process is iterative, with **Evaluation 1**, **Evaluation 2**, and **Evaluation n** leading to **Anpassung** (adaptation) and back into the development cycle. The final outcome is **Endprodukt mit hoher ergonomischer Güte**.

> Durchgängige Berücksichtigung ergonomischer Aspekte
> Kontinuierliche Beteiligung der späteren Benutzer

K. Schneider / J.Greenyer Siehe: Vorlesung von Michael Offergeld an der Universität Ulm, WS 2002 SWQ 2016 - 92

Referenzmodell Usability Engineering

Die Phasen der Projektbegleitung

Version: 02/2001-1

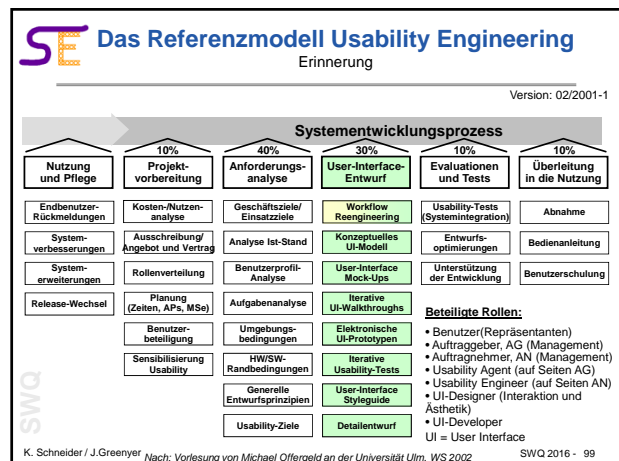
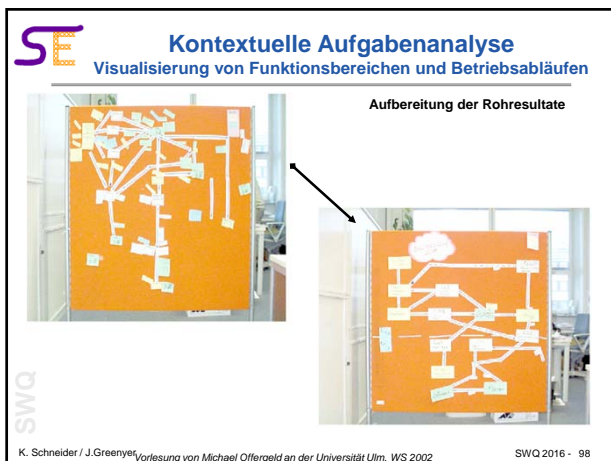
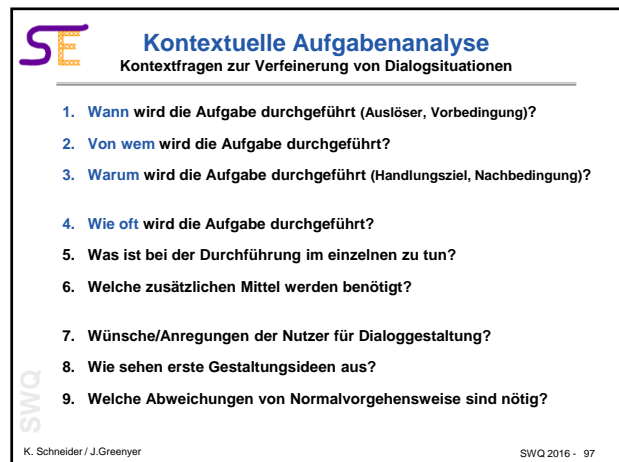
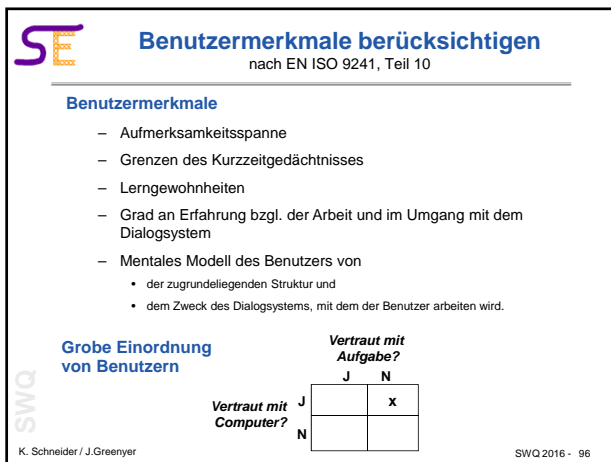
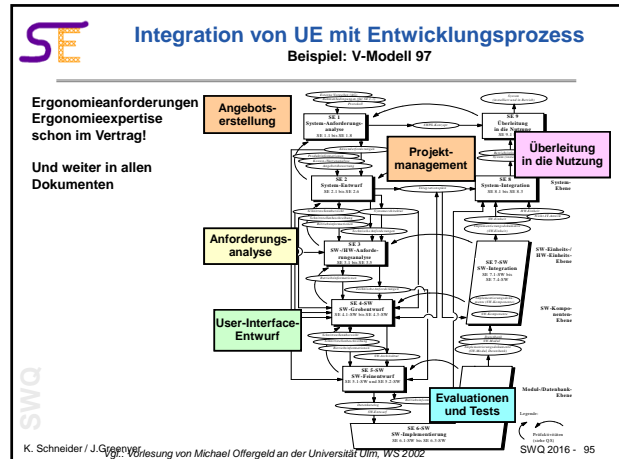
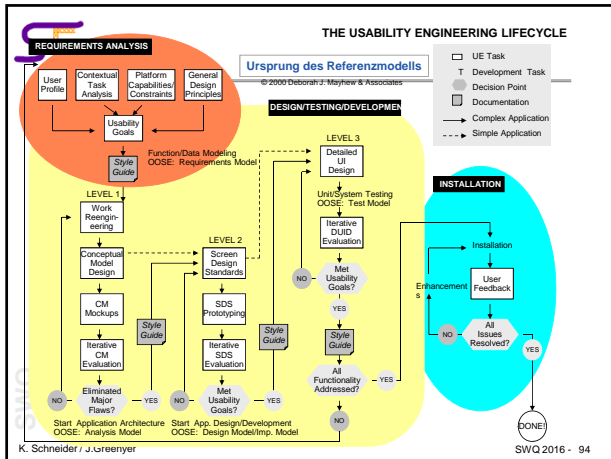
The diagram details the **Systementwicklungsprozess** (System Development Process) with five phases and their associated tasks:

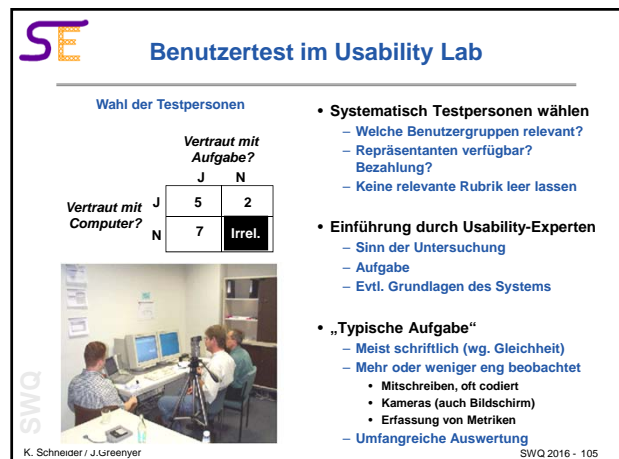
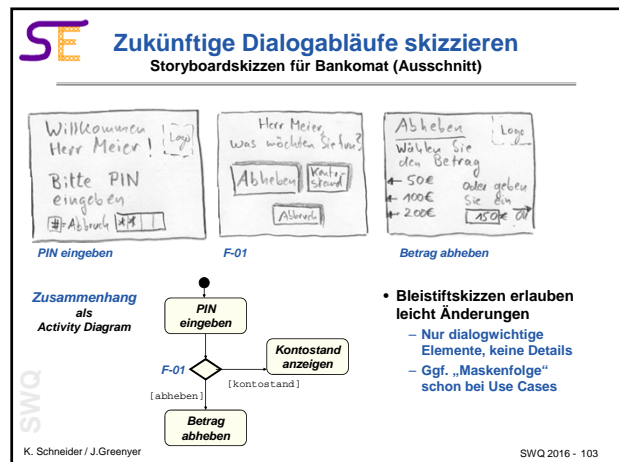
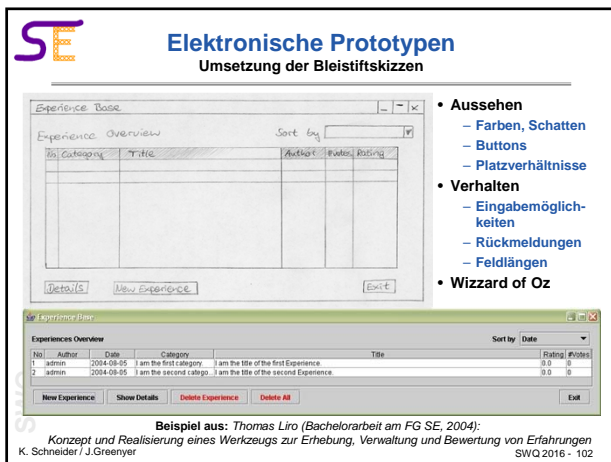
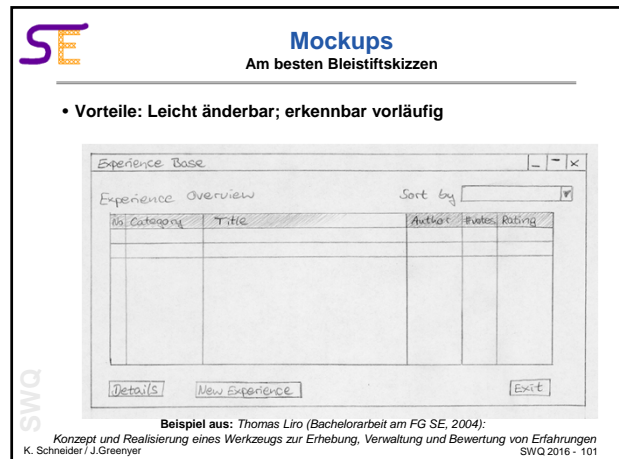
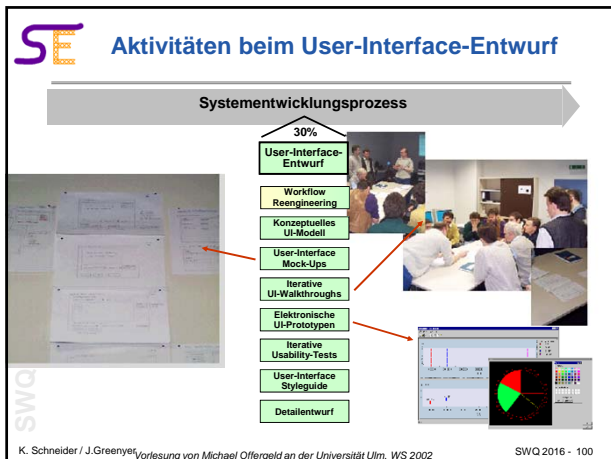
- 10% Projekt-vorbereitung:** Kosten-Nutzen-analyse, Ausschreibung/Angebot und Vertrag, Rollenverteilung, Planung (Zeiten, APs, MSe), Benutzerbeteiligung, Sensibilisierung Usability
- 40% Anforderungs-analyse:** Geschäftsziele, Analyse Ist-Stand, Benutzerprofil-Analyse, Aufgabenanalyse, Umgebungsbedingungen, HW/SW-Randbedingungen, Generelle Entwurfsprinzipien, Usability-Ziele
- 30% User-Interface-Entwurf:** Workflow Reengineering, Konzeptuelles UI-Modell, User-Interface Mock-Ups, Iterative UI-Walkthroughs, Elektronische UI-Prototypen, Iterative Usability-Tests, User-Interface Styleguide, Detailentwurf
- 10% Evaluationen und Tests:** Usability-Tests (Systemintegration), Entwurfs-optimierungen, Unterstützung der Entwicklung
- 10% Überleitung in die Nutzung:** Abnahme, Bedienanleitung, Benutzerschulung

Beteiligte Rollen:

- Benutzer(Repräsentanten)
- Auftraggeber, AG (Management)
- Auftragnehmer, AN (Management)
- Usability Agent (auf Seiten AG)
- Usability Engineer (auf Seiten AN)
- UI-Designer (Interaktion und Ästhetik)
- UI-Developer
- UI = User Interface

K. Schneider / J.Greenyer Siehe: Vorlesung von Michael Offergeld an der Universität Ulm, WS 2002 SWQ 2016 - 93





SWQ

Experten-Evaluationen

Usability-Experten untersuchen anders

- Orientierung im System**
 - Erster Eindruck: Anmutung, Navigation
- Grundlegende Regeln (z.B. „Acht Goldene Regeln“)**
 - kennen die Experten; oft Checklisten
 - Verstöße gegen Farben-, Größen-, Konsistenzregeln?
 - Richtige GUI-Elemente, Dialogabläufe?
 - Tote Links, unpassende Überschriften, irreführende Metaphern
- Einfache Aufgabe (oft vorher selbst-gestellt/-ausgewählt)**
 - Erprobung im Kontext
- Beobachtung**
 - Manchmal zu zweit durchgeführt
 - Mitschrift: Notizblock; allenfalls Bildschirmaufzeichnung
- Präsentation**
 - Streng objektiv, keine Geschmacksaussagen: darum geht es nicht
 - Alle Befunde mit Begründung, evtl. negativen Folgen (Konkurrenz)
 - Dokumentation: Evaluationsbericht mit Screenshots

SWQ

K. Schneider / J. Greenyer SWQ 2016 - 107

SWQ

UE-Aktivitäten und ihr Nutzen

Einfache Varianten

- Beschreibung Nutzerschnittstelle**
 - Input für Benutzerhandbuch, Online-Hilfe, Schulungen
- Beschreibung der Betriebsabläufe, Interaktionsszenarien**
 - Input für Abnahmetests
- Nutzerbeteiligung (z.B. Ergo-Audit)**
 - Sensibilisierung der Nutzer für Bedienprobleme
 - Schon früh bessere Akzeptanz durch AG

SWQ

K. Schneider / J. Greenyer SWQ 2016 - 108

Software-Qualität

Kapitel 2.4

Qualität fassbar machen

Inhalte

Fehler finden: Grundeinstellung zum Testen

Qualitätsanforderungen konkretisieren und prüfen

Beispiel Usability: Gut bedienbare Software gestalten

Qualitätsmodelle: Was genau soll Qualität bedeuten?

Prof. Dr. Kurt Schneider

„Wie prüft man Qualitäts-Anforderungen?“

Auch für andere Qualitäts-Aspekte als Usability

SWQ

K. Schneider / J. Greenyer SWQ 2016 - 110

SWQ

Boehms „Qualitätenbaum“

Geordneter Katalog möglicher Q.-Anforderungen

SWQ

Siehe z.B. in Boehm, Barry (1981): Software Engineering Economics. Prentice Hall, Englewood Cliffs, NJ.

K. Schneider / J. Greenyer SWQ 2016 - 111

SWQ

ISO 9126, Operationalisierung

Originalformulierungen aus der Norm

SWQ

Jede Sub-Characteristic wird in Attribute konkretisiert und messbar gemacht.

Dies muss aber jedes Projekt selbst machen.

K. Schneider / J. Greenyer http://www.sce.umkc.edu/~burnes/pl/software_quality_management/iso9126-1and2.pdf (3-43) SWQ 2016 - 112

Eine Interpretation der Qualitätsaspekte in Anlehnung an ISO 9126

Funktionalität Vorhandensein von Funktionen mit festgelegten Eigenschaften, diese Funktionen erfüllen die definierten Anforderungen. • Angemessenheit • Richtigkeit • Interoperabilität • Ordnungsmäßigkeit • Sicherheit Zuverlässigkeit Fähigkeit der Software, ihr Leistungsprofil unter festgelegten Bedingungen über einen festgelegten Zeitraum zu bewahren. • Rolle • Fehleranzahl • Wiederherstellbarkeit Benutzbarkeit Aufwand, der zur Benutzung erforderlich ist, und individuelle Bereitstellung der Benutzung durch eine festgelegte oder vorausgesetzte Benutzergruppe; hinweislich der Bereich Software-ergonomie. • Verständlichkeit • Erlernbarkeit • Bedienbarkeit	Effizienz Verhältnis zwischen dem Leistungsniveau der Software und dem Umfang der eingesetzten Betriebsmittel unter festgelegten Bedingungen. • Zeitverhalten • Verbrauchsverhalten Änderbarkeit Aufwand, der zur Durchführung vorangegebener Änderungen notwendig ist. • Änderbarkeit • Stabilität • Prüfbarkeit Übertragbarkeit Eignung der Software, von einer Umgebung in eine andere übertragen zu werden. • Anpassbarkeit • Installierbarkeit • Kompatibilität • Austauschbarkeit
--	--

K. Schneider / J. Greenyer

SWQ 2016 - 113

Standards sind nicht umsonst

K. Schneider / J. Greenyer

SWQ 2016 - 114

Qualitätsaspekte definiert

Prinzip: Definition nach IEEE 610.12-1990 (SE-Glossary)

- **Diese Definitionen sollten Sie kennen!**
Hier ähneln sich ISO 9126, 25 000 und IEEE 610.12-1990 sehr
- **Flexibilität (Flexibility)**
– Leichtigkeit, mit der ein System abgeändert werden kann, um es in Anwendungen oder Umgebungen zu benutzen, für die es nicht entwickelt worden ist.
- **Portabilität (Portability)**
– Leichtigkeit, mit der ein System von einer HW- bzw. SW-Umgebung in eine andere transferiert werden kann
- **Wiederverwendbarkeit (Reusability)**
– Ausmaß, in dem SW in mehr als einem Programm oder SW-System verwendet werden kann

K. Schneider / J. Greenyer

SWQ 2016 - 115

Qualitätsaspekte definiert

Nach IEEE 610.12-1990 (SE-Glossary)

- **Korrektheit (Correctness)**
– Ausmaß, in dem SW ihre Spezifikation erfüllt (*Qualitätsmerkmal?*)
- **Effizienz (Efficiency)**
– Ausmaß, in dem ein System seine Leistungen mit einem Minimum an Ressourcenverbrauch erbringt
- **Zuverlässigkeit (Reliability)**
– Fähigkeit eines Systems, die verlangte Funktionalität unter gegebenen Randbedingungen für eine gegebene Zeit zu erfüllen
- **Testbarkeit (Testability)**
– Ausmaß, in dem ein System das Erstellen von Testbedingungen erleichtert, sowie die Durchführung der Tests, ob die Bedingungen erfüllt sind

K. Schneider / J. Greenyer

SWQ 2016 - 116

Qualitätsaspekte definiert

Nach IEEE 610.12-1990 (SE-Glossary)

- **Verwendbarkeit (Bedienbarkeit: Usability)**
– Leichtigkeit, mit der ein Benutzer die Bedienung eines Systems, die Dateneingabe und die Interpretation seiner Ergebnisse lernen kann
- **Wartbarkeit (Maintainability)**
– Leichtigkeit, mit der ein System geändert werden kann. Ziel:
 - Fehler beheben
 - Fähigkeiten zu erhöhen
 - An eine veränderte Umgebung anpassen
- **Integrität (Integrity)**
– Ausmaß, in dem ein System unberechtigte Zugriffe auf Programme und Daten und deren unberechtigte Veränderung verhindert
 - „Security“: Sicherheit der Daten vor unberechtigtem Zugriff und Verlust durch Angriffe
 - Dagegen „safety“ = Def Abwesenheit von Risiken, insb. für Menschen d.h. „richtet keinen Schaden an“

K. Schneider / J. Greenyer

SWQ 2016 - 117

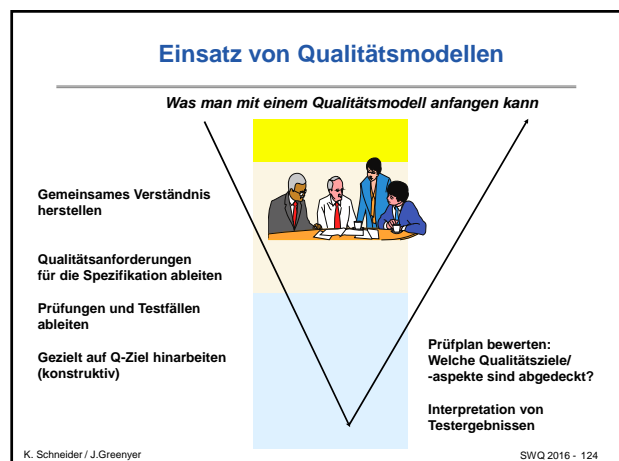
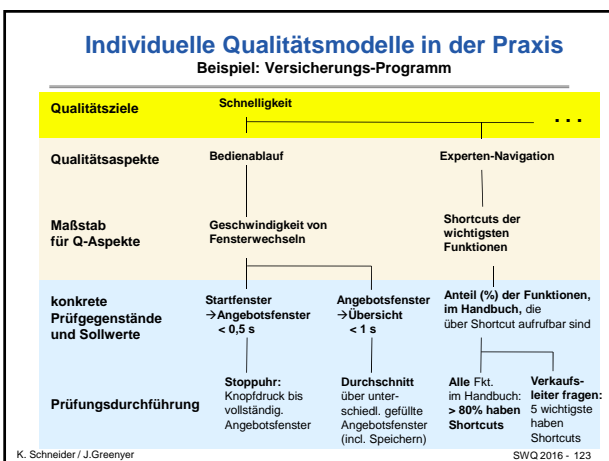
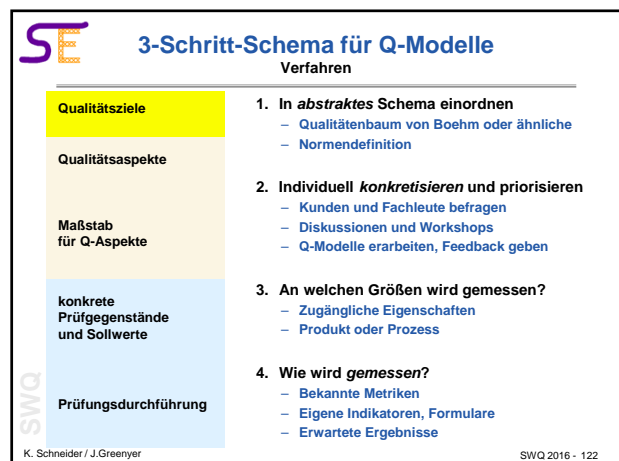
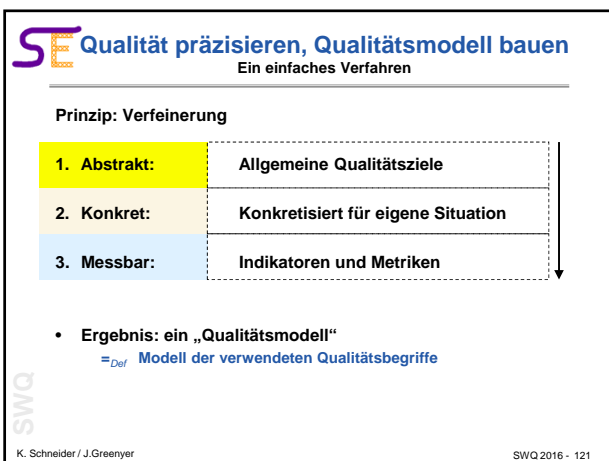
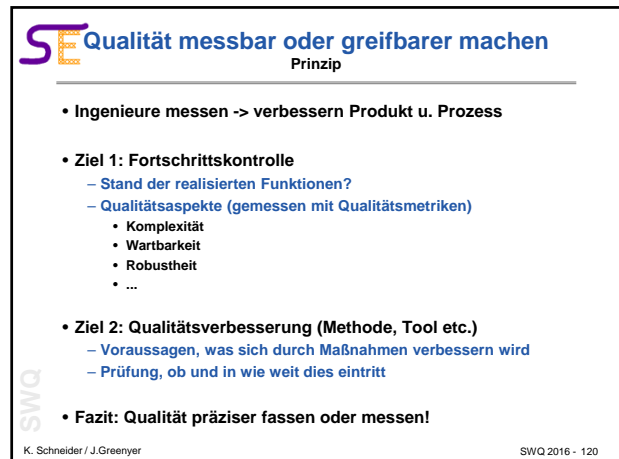
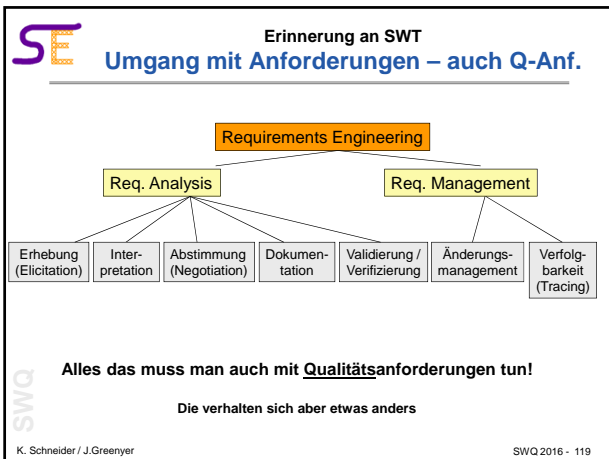
Was nützen generische Q.-Definitionen?

Vorgehensweise

- Checkliste, um keine Aspekte zu vergessen
- Anregung, zeigen Beziehungen zwischen Q.-Aspekten
- Auch vage Qualitätsaspekte können *priorisiert* werden
- Reichen aber *nicht* aus, um
 - Qualitätsanforderungen genau zu verstehen
 - Zielerreichungsgrade festzustellen
 - Die Entwicklung oder Prüfung anzuleiten
- Teils normierte/im Unternehmen einheitliche Terminologie
- Nächster Schritt daher: *Spezifische* Q-Anforderungen

K. Schneider / J. Greenyer

SWQ 2016 - 118



Zitate aus einem SW-Qualitätshandbuch

Ausgangspunkt: Liste von Qualitätsmerkmalen

Dazu Zitate/Erläuterungen:

- „Qualitäts-Merkmale müssen mit den Anforderungen des Pflichtenhefts in Einklang gebracht werden.“
- „... ist es erforderlich, zu jedem ausgewählten Merkmal *messbare Größen* ... zu definieren.“
- „... zu berücksichtigen, dass die *Erhebung* wirtschaftlich und praktikabel sein muss.“

Sie sehen:
noch viel Konkretisierungsbedarf für jedes Projekt!

SWQ K. Schneider / J. Greenyer SWQ 2016 - 125

Beispiele für Qualitätsanforderungen

Smartphone-App NoRisk

- Welche Anforderungen haben Sie an ein neues Smartphone?
- An eine App?
- Spezielle Anforderungen der Versicherung an NoRisk-App?

Funktionalität
Zuverlässigkeit
Bedienbarkeit
Portabilität
Wartbarkeit
Effizienz
„Schnelligkeit“

K. Schneider / J. Greenyer SWQ 2016 - 126

Zur Diskussion gestellt:

Aus dem Pflichtenheft für NoRisk

- „Qualitätsziele des Projekts“
- ...
- NoRisk muss einfach und intuitiv benutzbar sein, das heißt konkret: Es soll Bearbeiter unterstützen und darf den Arbeitsflow mit dem Kunden keinesfalls unterbrechen oder aufhalten!
- Da NoRisk auf keinen Fall den Workflow des Bearbeiters mit dem Kunden stören darf, wird hoher Wert auf Schnelligkeit gelegt. Seitenwechsel und Berechnungen müssen schnell vonstatten gehen und Reaktionszeiten müssen sehr gering sein.
- ...
- NoRisk schickt die Kundendaten an den firmeneigenen Server. Diese Daten müssen mit großer Vorsicht behandelt werden (Datenschutzgesetz beachten). NoRisk muss also eine hohe Sicherheit bei der Übertragung gewährleisten. Da NoRisk Kundendaten auch auf dem mobilen Endgerät speichert, muss die App auch dabei den datenschutzrechtlichen Anforderungen und Standards genügen.

Usability
Schnelligkeit
Sicherheit, Datenschutz

SWQ K. Schneider / J. Greenyer SWQ 2016 - 127

Pflichtenheft zu NoRisk

Priorisierung

Priorisierung der Q-Aspekte

Q-Aspekt	Priorisierung (0-10)
Effizienz	7
Robustheit	7
Stabilität	7
Flexibilität	2
Wiederverwendbarkeit	6
Erweiterbarkeit	7
Wartbarkeit	7
Portabilität	6
Sicherheit, Integrität	10
Schnelligkeit	8
Usability	8

K. Schneider / J. Greenyer SWQ 2016 - 128

Qualitätsmodell für NoRisk-App

Folienform mit Farben

- Für Diskussionen
- Ein Qualitätsziel pro Folie
- Sollte gut vorbereitet sein

Textform, eingerückt

- Für Überblick (Fachleute)
- Doku. aller Qualitätsziele
- Kann schnell geändert werden

Qualitätsmodell

4. Fehlertoleranz

4.1 Datenerhalt trotz Absturz

4.1.1 Recovery von eingegebenen Daten nach Absturz

Angebotsmaske teilweise ausgefüllt. Wurde nicht explizit gespeichert.

Alle Felder werden ausgefüllt. Batterie entnehmen; warten; Wieder einsetzen. Wie viel geht im schlimmsten Fall verloren?

Soll: maximal letztes Feld

4.1.2 Recovery bei Verbindungsabbruch

...

SWQ K. Schneider / J. Greenyer SWQ 2016 - 129

Oberflächliche Qualitätsziele – wieso?

Hintergrund

- Die Zeit ist knapp
 - Eigentlich soll Software entwickelt werden
- Mangelndes Problembewusstsein
 - „Wir sind Profis, Qualität entsteht sowieso“ (leider nicht)
 - Zweck von Q-Modellen ist unbekannt (daher keine Engagement)
 - Ziele werden vage genannt (zu abstrakt: nicht handlungsleitend)
 - „Unser Handbuch gibt doch diese Abstraktion vor, also reicht sie“
- Unvermögen
 - „Wir finden keine tiefer-gehenden Q-Modelle“
 - Wie man Q-Modelle effizient einsetzt, ist unklar
 - „Wenn man da einmal anfängt, hört man nicht mehr auf“ (Qualitätsziele als echte Falle!)

SWQ K. Schneider / J. Greenyer SWQ 2016 - 130

SWQ

Q-Modelle von der Stange?

Tipps und Tricks

- (+) **Wiederverwendung spart Arbeit, Zeit und Geld**
 - Sonst oft Q-Ziele vergessen -> Boehm, ISO 9126 oder Checkliste!
 - Es gibt schon zahllose Q-Modelle - wieso sie nicht verwenden?
- (-) **Aber: jedes Projekt ist anders**
 - Andere Q-Anforderungen
 - Andere Maßstäbe für einen Q-Aspekt
 - Andere Sollvorgaben
- Fazit: Teile wiederverwenden!**
 - Abstrakte Begriffsbäume
 - Vorhandene Metriken oder Indikatoren
- Andere Teile NICHT:**
 - Konkretisierung und Priorisierung abstrakter Q-Ziele
 - Zusätzliche Indikatoren
 - Spezifische Einschränkung des Suchraums

K. Schneider / J. Greenyer
SWQ 2016 - 131

SWQ

Zusammenfassung

Umgang mit Qualitätsanforderungen

- Das Problem beginnt schon bei den Begriffen
 - Was ist SW-Qualität - und was bedeuten ihre Aspekte?
 - Genormte Definitionen, aber noch interpretierbar

Vorgehen:

- Q-Anforderungen erheben
- Priorisieren
- Konkretisieren
 - Qualitäts-Modell mit 3-Schritt-Schema
 - Obere und untere Teile wiederverwenden, mittlere individuell neu machen
- Passende Präsentationsform
- Später als Referenz nutzen
 - Entwicklung, Reviews, Abnahme

K. Schneider / J. Greenyer

Qualitätsmodell

4. Fehlertoleranz
4.1 Datenverlust trotz Absturz
4.1.1 Recovery von eingegebenen Daten nach Absturz
4.1.2 Recovery bei Verbindungsabbruch