Über den Software-Qualitätsbegriff - Teil 2: Anforderungen und Merkmale am Beispiel Software-Ergonomie (Usability)

Roland Petrasch
Technische Fachhochschule Berlin, Fachbereich Informatik
petrasch@tfh-berlin.de

Abstract

Dieser Beitrag führt die Diskussion um den Begriff *Software-Qualität* fort [Petr99b]. Ausgangspunkt war die Feststellung, dass es nicht genügt, den Qualitätsbegriff zu definieren, um ein einheitliches Verständnis für Qualität bei den Beteiligten zu erreichen. Vielmehr sind qualitätsbestimmende Begriffe wie *Forderung* und *Merkmal* zu klären. Auch stellt sich die Frage nach der Verwendung dieser Begriffe im Rahmen von Entwicklungsprozessen und Vorgehensmodellen (vgl. [Petr99a]).

In die Kritik geraten dabei Normen und Standards wie beispielsweise die ISO 9001 oder das V-Modell 97, die kaum zur Klärung beitragen. Die Problematik des Qualitätsbegriffes auch in Verbindung mit Prozessstandards sei daher nochmals kurz erläutert. Weiterhin seien einige kontroverse Argumente vorgestellt, und um das Problem mit der Software-Qualität zu veranschaulichen, ist ein Beispiel aus dem Bereich *Software-Ergonomie* gegeben. Allerdings kann und will dieser Beitrag keine „Küchenrezepte" liefern, sondern hauptsächlich auf die Schwierigkeiten beim Umgang mit Qualität hinweisen und die Diskussion beleben.

1. Was ist Software-Qualität?

Das Begriffspaar *Forderung* und *Merkmal* nehmen eine zentrale Stellung bei vielen Definitionen für *Qualität* ein, z.B. bei der ISO 8204: „Gesamtheit von Merkmalen [...] einer Einheit bezüglich ihrer Eignung, festgelegte und vorausgesetzte Erfordernisse zu erfüllen" [DIN8204].

Allerdings finden sich in der Literatur unterschiedliche Bezeichnungen für *Forderung* und *Merkmal*. So werden Forderungen oftmals auch als *Anforderungen, Requirements, Needs* oder *Bedürfnisse* bezeichnet. Merkmale finden sich in der englischsprachigen Literatur als *Characteristics*. Nun ist leicht einsehbar, dass sich hinter Forderungen und Merkmale außerordentlich komplexe Sachverhalte verbergen, so dass z.B. eine Strukturierung, Verfeinerung und Hierarchisierung notwendig wird, damit Anforderungen operationalisierbar und prüfbar werden, denn Requirements wie *Aufgabenangemessenheit* (als Beispiel für eine ergonomische Anforderung als Teil der nicht-funktionalen Anforderungen) können kaum direkt im Software-Entwurf bzw. der Implementierung umgesetzt werden.

Wichtig erscheint insbesondere die Prüf- oder Testbarkeit: Es genügt bekanntlich nicht, von „ausreichender Antwortzeit" zu sprechen, da diese Anforderung nicht testbar ist. Auch die Quantifizierbarkeit bei der Bewertung kann ein Problem darstellen, denn das Ergebnis einer Prüfung lautet nicht immer „erfüllt" oder „nicht erfüllt", d.h. das Skalenniveau ist zuweilen recht unterschiedlich. Weitere Aspekte tragen zur Komplexität bei, z.B. die *externe* und *interne Qualität* sowie die *Quality in use* [DIN9126], die quasi „Sichtweisen" auf eine Software darstellen. Auch ist zwischen *Prozess-* und *Produktqualität* zu unterscheiden.

Zusammenfassend lässt sich feststellen, dass erst die Zuordnung von Merkmalen zu den Forderungen eine Qualitätsaussage ermöglicht. Nur wenn geklärt ist, was die Anforderungen sind und welche Merkmale einer Software Relevanz für deren Erfüllung haben, ist Software-Qualität ermittelbar.

2. Stand der Diskussion

Angesichts des o.g. leicht verständlichen Konzeptes für Qualität stellt sich die Frage, was am Qualitätsbegriff bzw. an dessen Verwendung diskussionswürdig sein soll. Zwei Punkte sind hier in erster Linie zu nennen:

- Obwohl mit der ISO 8204 eine Definition für Qualität vorliegt und einige Normen diese Festlegung auch übernehmen, ist dennoch eine einheitliche Verwendung weder in der Standardliteratur, noch in den einschlägigen Normen und Standards zu beobachten. Nicht einmal innerhalb der ISO scheint es eine klare und konsistente Linie zu geben: Die ISO 9126 [DIN9126] beispielsweise verwendet den Begriff *Quality (Sub-)Characteristic*, z.B. in Verbindung mit *Usability*, während die ISO 9241-10 [ISO9241] von *Ergonomic Requirements* spricht, welche ebenfalls die *Usability* betreffen. Dies ist zumindest auf der terminologischen Ebene verwirrend.
- Viele Prozessnormen bzw. Vorgehensmodelle greifen das Konzept von Anforderungen und Merkmalen überhaupt nicht auf. Dies betrifft z.B. die ISO 9001 [DIN9001], die ISO 15504 [ISO15504] oder das V-Modell 97 [Bund97]. Es ist zwar oftmals von Anforderungen (Requirements) die Rede, aber auf Merkmale und deren Prüfung gegen die Anforderungen findet sich nichts oder kaum etwas. Dabei wäre dies zwingend notwendig – nicht nur um Prüfungen durchführen zu können, sondern beispielsweise auch für das *Requirements Tracing*.

Die begriffliche Vielfalt wiegt sicherlich nicht so schwer, wie die mangelhafte Unterstützung durch die Prozessnormen und Vorgehensmodelle: Um die Behauptung zu stützen, dass sich eine hohe Prozessqualität auch positiv auf die Produktqualität auswirkt, wäre eine prozessorientierte und detaillierte Beschreibung notwendig, wie Anforderungen als Ausgangspunkt mit Merkmalen des Produktes zu verbinden sind, so dass sich Qualität auch auf

der nicht-funktionalen Ebene (z.B. bei software-ergonomischen Anforderungen) nachweisen lässt[1].

Bevor näher auf das Thema Software-Ergonomie eingegangen wird, seien einige Gegenargumente bzgl. der o.g. Kritikpunkte genannt[2], wobei zu beachten ist, dass die Autoren der Argumente nicht namentlich genannt werden, hauptsächlich aber Vertreter von Prozessnormen und Vorgehensmodelle im positiven Sinne sind[3]:

- Gegenargument #1: Einige Normen (z.B. ISO 9001) unterscheiden nicht zwischen Anforderungen und Merkmalen, weil diese Differenzierung einen zu hohen Detaillierungsgrad bewirken würde. Wenn also in den Werken von Qualität bzw. Anforderungen gesprochen wird, dann ist damit der Qualitätsdefinition ausreichend genüge getan. Eine genauere Betrachtung bzgl. Anforderungen und Merkmale ist nicht notwendig.
- Gegenargument #2: Einige Standards wie das V-Modell 97 enthalten einige Verweise auf andere Normen, was völlig ausreichend ist. Es ist nicht das Ziel von Standards bzw. Vorgehensmodellen, genaue Angaben über die Anforderungen und Merkmale zu machen.
- Gegenargument #3: Um methodenunabhängig zu sein, können Prozessnormen und Vorgehensmodelle nun einmal nicht detaillierter sein. Schließlich soll nur der Prozess, nicht das Produkt beschrieben werden.
- Gegenargument #3: Es ist gar nicht möglich, zu jedem Merkmal eine Anforderung zu definieren. Im Falle von ergonomischen Merkmalen, z.B. einem Cancel-Button, müsste es dann zu jedem Element eine separate Anforderung geben, was nicht realistisch ist.

Bei genauer Betrachtung, sind die Gegenargumente nicht schlüssig: Bei, Argument #1 stellt sich die (Gegen-)Frage, warum *Qualität* erst mit den beiden Begriffen *Anforderungen* und *Merkmalen* definiert wird und dann genau diese beiden Begriffe praktisch keine Rolle mehr spielen. Würde also das Argument stimmen, dann sollte auch die Qualitätsdefinition abstrakter werden, was allerdings wenig hilfreich wäre, da die Software-Entwicklung einen Bedarf nach einer möglichst konkreten Unterstützung hat. Ähnliches gilt für das Argument #2: Wieso ist ein Vorgehensmodell wie das V-Modell 97 nicht in der Lage, methodenunabhängig das Vorgehen zur Transformation von Anforderungen zu Merkmalen sowie die Prüfung von Merkmalen beschreiben[4]? Das ist nicht nachvollziehbar. Auch das Argument #3 ist bereits durch die Realität widerlegt: Praktisch jeder *Style Guide* für das User Interface, der technische Anforderungen an eine Benutzungsoberfläche definiert, gilt für eine beliebige Anzahl von Dialogen und Interaktionselementen, d.h. es muss durchaus nicht für jedes Merkmale eine separat definierte Anforderung existieren.

Dass sich das Qualitätskonzept mit Anforderungen und Merkmalen durchaus in einer allgemeinen Vorgehensweise unterbringen lässt, zeigt das *Quality Function Deployment* (QFD): In einer Art Matrix, dem *House of Quality*, ist das *WAS* (Kundenanforderungen) und das *WIE* (Merkmale) abgebildet, wobei einerseits die *Korrelation* der Merkmale und andererseits der *Unterstützungsgrad* der Merkmale in Bezug auf die Anforderungen anzugeben sind ([Zoll02, S. 124], [DGQ], s. Bild 1).

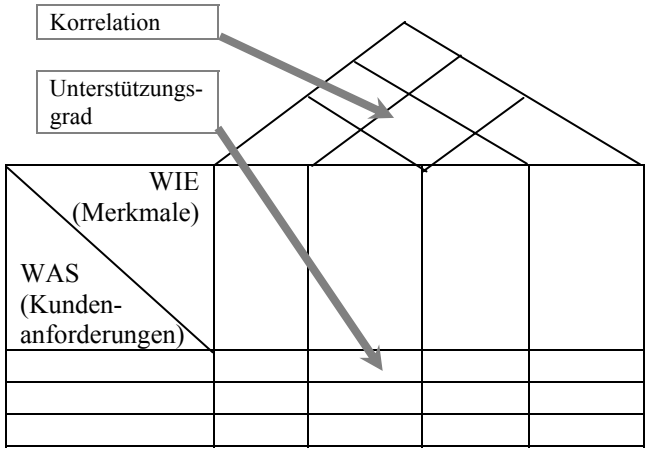

Bild 1: House of Quality (QFD)

Auch wenn die QFD-Bezeichnungen, z.B. „House of Quality“, zuweilen etwas eigenwillig sind, vermag der Ansatz eines zu zeigen: Eine Lösung des „Qualitätsproblems“ für Prozessnormen und Vorgehensmodelle ist möglich, d.h. die methodenunabhängige Verwendungen des Konzeptes für Anforderungen und Merkmale. Es ist zu hoffen, dass sich der momentane Zustand dieser Werke in naher Zukunft verbessern wird.

3. Erfolgfaktor *Software-Ergonomie*

Software-Ergonomie gewinnt z.B. mit der zunehmenden Nutzung von Software als Kommunikations- und Arbeitsmittel zwischen Anbieter und Kunden im Internet erheblich an Bedeutung, wird jedoch in der Praxis oftmals unterschätzt. Immerhin sind im Bereich *Software-Ergonomie*, bzw. *Usability*[5], erhebliche Fortschritte erzielt worden:

- Produktorientierte Normen, z.B. die ISO 9241-10, beschreiben allgemeine ergonomische Anforderungen.
- Style Guides und Guidelines, z.B. die Java Look and Feel Design Guidelines [Sun01a][6], machen auf der technischen Ebene für eine konkrete Plattform oder eine Technologie entsprechende Vorgaben, z.B. für den Abstand von Interaktionselementen.
- Prozessorientierte Normen, z.B. die ISO 13407

[1] Interessante Kritikpunkte bzgl. Normen finden sich auch in [Balz95].
[2] Diese Gegenargumente erreichten mich als Feedback auf den ersten Artikel „Über den Software-Qualitätsbegriff“ [Petr99b].
[3] Außerdem sind die Argumente nicht wörtlich, sondern nur sinngemäß wieder gegeben, da z. T. nur mündliche Gespräche geführt wurden.
[4] Zur Erinnerung: Finden sich im Submodell SE (Systemerstellung) teilweise noch die Begriffe „Anforderung“ und „Merkmal“, kommt der Begriff „Merkmal“ im gesamten Regelungsteil des Submodells QS (Qualitätssicherung) überhaupt nicht mehr vor.
[5] Usability ist lt. ISO 9241 das Ausmaß, in dem die Software von einem bestimmten Benutzer effektiv, effizient und zufrieden stellend verwendet werden kann. Daher seien die Begriffe Software-Ergonomie und Usability synonym verwendet - auch wenn dies formal gesehen nicht ganz korrekt ist.
[6] In den Advanced Topics finden sich sogar weiterführende Vorgaben [Sun01b].

[ISO13407], definieren eine Vorgehensweise zur Konstruktion interaktiver Systeme[7].

- Standardliteratur unterstützen die Software-Entwicklung durch Heuristiken (vgl. [Schn97], [Niel93]), und Methoden (vgl. [Baye+98]). Insbesondere benutzerzentrierte Ansätze (vgl. [Vred+02]), dominieren das Spektrum.

Dennoch ist die systematische und nachvollziehbare Entwicklung ergonomischer Software-Systeme nach wie vor eine Herausforderung. Offenbar gelingt es nicht, software-ergonomische Anforderungen so umzusetzen, dass entsprechende Merkmale zu einer klaren Qualitätsaussage führen so wie dies bei funktionalen Anforderungen der Fall ist. Die Gründe sind vielfältig:

- Die Teildisziplin *Software-Ergonomie* hinkt verständlicherweise anderen Bereichen, z.B. den Programmiersprachen, hinterher. In der Praxis halten die Erkenntnisse daher erst allmählich Einzug. Aber auch in der Aus- und Weiterbildung gab und gibt es Defizite, z.B. wird das Thema „Ergonomie von Software-Systemen“ im Studium oftmals nicht oder nur am Rande behandelt.
- Mit der Gestaltung der Oberfläche verbinden Entwickler teilweise eine Art „letzte Bastion der Kreativität“: Die eigene Ästhetik gepaart mit Unwissenheit ergeben zuweilen skurrile Gebilde, die von Usability weit entfernt sind. Eine emotional geführte Diskussion um die „richtige“ Oberfläche verstärkt das Problem.
- Die Qualitätssicherung beschränkt sich häufig auf die Prüfung der Einhaltung von Style Guides und führt Benutzerbefragungen durch. Konstruktive Ansätze, z.B. Prototyping, sind zwar vorhanden, aber konzeptionell eher intuitiv und mangelhaft in den Entwicklungsprozess integriert.
- Formale Methoden sind entweder nicht vorhanden oder kommen kaum zum Einsatz. Auch die Werkzeugunterstützung ist gering: Kaum ein CASE-Produkt bietet mehr als einen *User Interface Builder* an.

Nun soll an dieser Stelle keine vollständige Ursachenforschung betrieben werden, einige Punkte seien dennoch aufgegriffen: Das Problem der fehlenden Integration des *Usability Engineering*[8] in den Entwicklungsprozess scheint es zunächst nicht zu geben: Prototyping ist selbst in den frühen Phasen einplanbar und Usability Tests (wie Benutzerbefragungen, heuristische Evaluationen etc.) können ebenfalls während des gesamten Prozesses durchgeführt werden. Problematisch aber ist die Verbindung der Konzepte: Welche UML-Diagramme geben Hinweise auf die ergonomische Gestaltung? Wie kann im Vorfeld geprüft werden, ob die Menü-Struktur des Prototyps auch wirklich zu den Funktionen (z.B. Use-Cases) „passt“? Wie sollte die Entwicklungsdokumentation für das User Interface aussehen, damit die Entscheidungen auch später noch nachvollziehbar sind?

Besonders die Verbindung zwischen *Usability Engineering* und *Software Engineering* ist zu verbessern. So gibt es im Bereich *Usability* sehr gute methodische Ansätze wie z.B. das *Contextual Design*, welches u.a. mit Sequence Model und Affinity Diagram [Beye+98, S. 98, 155] arbeitet, den Sprachstandard UML jedoch außer Acht lässt, obwohl sich durchaus für die Software-Ergonomie relevante Sachverhalte damit abbilden lassen, z.B. der Navigationsraum [Henn+00], [Petr00].

4. Software-Ergonomische Anforderungen und Merkmale

Ein problematischer Bereich sind die ergonomischen Anforderungen, wobei das Problem etwas „überzeichnet“ ist, d.h. es seien zur Veranschaulichung zwei Extreme beschrieben:

- Extrem #1: Die Requirements befinden sich auf einer sehr hohen Abstraktionsebene (z.B. ISO 9241: *Selbstbeschreibungsfähigkeit*), bei der die direkte Umsetzbarkeit kaum geben ist, so dass eine Operationalisierung bzw. Konkretisierung erfolgen muss. Der Vorteil bei diesen ergonomischen Konzepten ist deren Mächtigkeit und Allgemeinheit sowie die Benutzerorientierung: Es ist leicht einsehbar, dass ein *selbstbeschreibungsfähiger* Dialog im Sinne des Anwenders und damit ergonomisch ist, d.h. zur ergonomischen Qualität beiträgt.
- Extrem #2: Die Requirements sind auf der technischen Ebene und geben konkrete Vorgaben für die Gestaltung des User Interface. Ihre ergonomische Sinnhaftigkeit lässt sich jedoch nicht immer sofort erkennen, z.B. soll der Abstand bei Command Buttons 5 Pixel betragen, ohne dass sofort klar wird, warum dies ergonomisch ist (s. Bild 2).

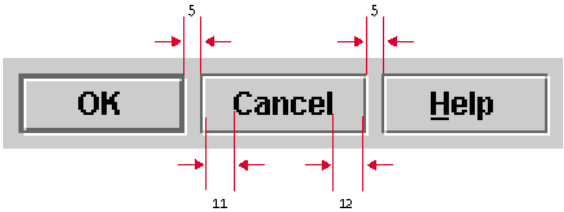

Bild 2: Beispiel aus den Java Look and Feel Design Guidelines “Spacing and Padding in Command Button Groups” [Sun01a, Kap 4]

Die o.g. Kritik ist – obgleich im Ansatz berechtigt – natürlich abzuschwächen: Die ISO 9241 beispielsweise gibt durchaus auch Hinweise für die Umsetzung und führt Beispiele auf. Dennoch besteht das Problem in der Praxis durchaus: Die Software-Entwicklung braucht nachvollziehbare Richtlinien für das Usability, die zwar direkt anwendbar sind, deren ergonomischer Hintergrund jedoch auch deutlich sein muss. Das trifft für viele Werke leider nicht zu: Style Guides, die (vereinfacht ausgedrückt) nur die Pixelabstände fordern, jedoch nicht erklären, warum

[7] Es existieren weiterhin Normen, die zwar nicht spezifisch auf Software-Ergonomie abzielen, aber dennoch von Bedeutung sind, z.B. die DIN ISO 12119 für die Prüfung [DIN12119].

[8] Begriffe wie *User Centered Design*, *Human Factors Engineering* oder *Usability Engineering* wurden bereits in den frühen 80er Jahren geprägt (vgl. [Good+86]). Eine Definition liefert Woodson: "…the practice of designing products so that users can perform required use, operation, service and supportive tasks with a minimum of stress and maximum of efficiency." [Wood91]

dies ergonomisch ist, gefährden ihre Akzeptanz bei den Nutzern. Zumindest sollten Richtlinien entsprechende Referenzen auf weitergehende Literatur enthalten.

Ein abschließendes Beispiel soll auf eine weitere Problematik hinweisen. Es geht um die Frage: Wie können die Eigenschaften des Menschen adäquat beim Usability Engineering berücksichtigt werden? Sie müsste eigentlich durch die o.g. Normen, Standards und Style Guide beantwortet sein. Dies ist jedoch nicht immer der Fall.

Ein Beispiel aus der Wahrnehmungspsychologie soll dies verdeutlichen (vgl. [Gold02]): Das „Gesetz der guten Fortsetzung“ (Gestaltgesetz) besagt, dass das Gehirn sich schneidende Konturen so interpretiert, dass möglichst wenig Biegungen oder Knicke entstehen. Und das „Gesetz der Schließung“ besagt, dass bei nahezu geschlossen Konturen eine Tendenz besteht, diese ganz zu schließen.

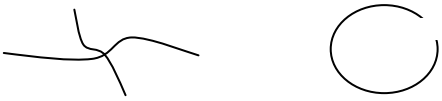

Bild 4: Das „Gesetz der guten Fortsetzung“ und das „Gesetz der Schließung“

Bei der Kontur auf der linken Seite „sieht“ der Betrachter zwei Linien, die sich „gut fortsetzen“, d.h. vertikal und horizontal verlaufen und nicht zwei abgeknickte Linien mit einem Verbindungspunkt. Die Kontur rechts wird mental geschlossen, so dass ein Kreis (und keine gekrümmte Linie) entsteht.

Solche Effekte sind nicht zwingend negativ – nur wenn diese ungewollt entstehen. So tritt bei überlagerten modalen Dialogen teilweise das Problem auf, dass die Navigationselemente nicht richtig erkannt werden. Bild 5 zeigt diesen Effekt: Ein Dialog wird von einer Einstellungsmaske überlagert, wobei die jeweiligen Command Buttons (OK und Abbrechen) sichtbar sind. Wählt der Anwender nun einen der unteren Buttons an, wird die Eingabe abgelehnt (da der obere Dialog modal ist). Dieser ungewünschte Effekt ist zu vermeiden: Die Begrenzungen sollten beispielsweise keine „gemeinsame“ Linie ergeben und die nicht aktivierbaren Buttons des hinteren Dialogs sollten nicht sichtbar sein, da der Anwender evtl. Schwierigkeiten hat, beide Dialoge auseinander zu halten und daher eine unerwünschte Wahrnehmung entsteht, d.h. die Kontur eines einzigen Dialogs (statt zweier Dialoge).

Bild 5: Einstellungen für den eMail-Client beim Netscape™ Navigator™ 7.0

Es wird deutlich, dass die ergonomische Anforderung „Vermeidung von ungewollten Effekten durch die Gestaltgesetze“ nur schwer direkt unsetzbar ist und auf der anderen Seite durch einen Style Guide nur ansatzweise operationalisiert werden kann, z.B. könnte es die Regel geben: „Überlagerte modale Dialogen sollten gegeneinander versetzt platziert werden. Nur die anwählbaren Interaktionselemente für die Navigation sollten dabei sichtbar sein“. Diese Vorgaben sind dann hinreichend konkret und testbar.

5. Fazit

Dieser Beitrag stellte hauptsächlich die Probleme im Umgang mit Qualität heraus:

Prozessnormen und Vorgehensmodelle könnten mehr Unterstützung bei der Software-Entwicklung hinsichtlich der Entstehung von (Produkt-)Qualität geben, wenn die Begriffe *Anforderungen* und *Merkmale* sowie die damit verbundenen Konzepte berücksichtigt würden.

Die Software-Ergonomie stellt darüber hinaus eine weitere Herausforderung dar, weil die Anforderungen oftmals wenig „greifbar“ und die technischen Richtlinien (Style Guides) nicht alle Aspekte der Ergonomie abdecken.

Die Fortsetzung dieses Beitrages (Titel „Über den Software-Qualitätsbegriff - Teil 3“), die demnächst hier erscheinen wird, soll aufzeigen, wie die beschriebenen Ansätze, z.B. das Quality Function Deployment (QFD), mit software-ergonomischen Anforderungen verbunden werden können, um der Software-Entwicklung eine bessere prozessorientierte Unterstützung geben zu können, als dies bei vielen Vorgehensmodellen zur Zeit der Fall ist.

Literatur

[Balz95] Balzert, H.: *Was nützen der Software-Technik die DIN-Normen ?* In: *GI-Softwaretechnik-Trends*, Gesellschaft für Informatik e.V., Band 15, Heft 1, Juni 1995

[Baye+98] Bayer, H.; Holtzblatt, K.: *Contextual Design.* Morgan KaufmannPublishers, 1998

[Bund97] Bundesministerium des Innern: *Entwicklungsstandard für IT-Systeme des Bundes: Vorgehensmodell, Teil 1: Regelungsteil*, Juni 1997

[DGQ] Deutsche Gesellschaft für Qualität (Hrsg): *QFD – Quality Function Deployment*, Beuth Verlag, 2001

[DIN8204] DIN EN ISO 8402: *Qualitäts-management Begriffe*. Beuth Verlag Berlin, 1995

[DIN9001] DIN EN ISO 9001: 2000: *Qualitätsmanagementsysteme - Anforderungen*. Beuth Verlag Berlin, 2000

[DIN9126] DIN ISO 9126:1991: *Informationstechnik - Beurteilen von Software-Produkten, Qualitätsmerkmale und Leitfaden zu deren Verwendung*. Beuth Verlag Berlin, 1991

[DIN12119] DIN ISO/IEC 12119: *Software-Erzeugnisse: Qualitätsanforderungen und Prüfbestimmungen*. Beuth Verlag Berlin, 1995 (früher: DIN 66285)

[Gold02] Goldstein, E.B.: *Wahrnehmungspsychologie. Eine Einführung*, 2002

[Good+86] Good, M.; Spine, T. M.; Whiteside, J.; George, P.: *User derived impact analysis as a tool for usability engineering*. In: Conference Proceedings *CHI '86 Human Factors in Computing Systems*, Boston, April 1986, ACM, New York, S. 241-246

[Herz+97] Herzwurm, G.; Schockert, S.; Mellis, W.: *Qualitätssoftware durch Kundenorientierung*, Vieweg, 1997

[Henn+00] Hennicker, R.; Koch, N.: *A UML-based Methodology for Hypermedia Design*. Ludwig-Maximilian-Universität, München, 2000

[ISO9241] ISO 9241-10: *Ergonomic Requirements for Office Work with Visual Display Terminals - Part 10: Dialogue principles*. International Organization for Standardization, 1996

[ISO13407] ISO 13407: *Human centred design processes for interactive systems.* International Organization for Standardization, 1999

[ISO15504] ISO/IEC 15504-1: *Information technology -- Software process assessment -- Part 1: Concepts and introductory guide* . International Organization for Standardization, 2003

[Niel93] Nielson, J.: *Usability Engineering*, Academic Press, 1993

[Petr99a] Petrasch, R.: *The Definition of ‚Software Quality': A Practical Approach*. In: Proceedings of the *10th International Symposium on Software Reliability Engineering*, Florida, Nov. 1999, IEEE Computer Society, 1999, S. 33-34

[Petr99b] Petrasch, R.: *Über den Software-Qualitätsbegriff*. In: *GI-Softwaretechnik-Trends*, Gesellschaft für Informatik e.V., Band 19 Heft 3, Nov. 1999, S. 36-39

[Petr00] Petrasch, R.: *Development of Ergonomic Software: Transformation from Requirements to Features using a Web Application's Navigation Space as an Example*. In: Conference Proceedings *CONQUEST 2000 – Conference on Quality Engineering in Software Technology*, Arbeitskreis Software-Qualität Franken e.V., 2000, S. 118-127

[Petr01] Petrasch, R.: *Einführung in das Software-Qualitätsmanagement. Mit Gastbeiträgen von Thomas Blum, Ralf Kneuper, Tim Koomen, Martin Pol und Andreas Spillner.* Logos Verlag, Berlin, 2001

[Schn97] Shneiderman, B.: *Designing the User Interface. Strategies for Effective Human-Computer Interaction*, 3. Auflage, Addison Wesley Longman, 1997

[Sun01a] Sun Microsystems Inc. (Editor): *Java(TM) Look and Feel Design Guidelines*, 2. Auflage, Addison Wesley, 2001

[Sun01b] Sun Microsystems Inc. (Editor): *Java(TM) Look and Feel Design Guidelines: Advanced Topics*, 1. Auflage, Addison Wesley, 2001

[Vred+02] Vredenburg, K.; Isensee, S.; Righi, C.: *User Centered Design,* Prentice Hall PTR, 2002

[Wood81] Woodson, W. E.: *Human Factors Design Handbook*, McGraw-Hill, 1981

[Zoll02] Zollondz, H.-D..: *Grundlagen Qualitätsmanagement*. R. Oldenbourg Verlag, München Wien, 2002