ao. Univ.-Prof. Dr. Horst Eidenberger
Allgemein beeideter und gerichtlich zertifizierter Sachverständiger für Informatik und Signalverarbeitung
Institut für Softwaretechnik und Interaktive Systeme der TU Wien

Software ohne Gewähr*

Wann ist die Qualität von Computer-Software mangelhaft?

Wer glaubt, dass der Markt allein den Preis einer Software bestimmt, der negiert wesentliche Aspekte des Rechtsgeschäfts Auftrags-Softwareentwicklung. Um sein Geld sicher zu bekommen, müssen ein paar Dinge beachtet werden. Dieser Artikel erklärt, welche.

1. Einleitung

In der Praxis des gerichtlichen Software-Sachverständigen ist festzustellen, dass sich das Gewährleistungsrecht als großer Stolperstein für Software-Entwicklung herausstellen kann. Das Gewährleistungsrecht sieht bekanntlich vor, dass der Verkäufer für Qualitätsmängel (Sachmängel) unabhängig vom Verschulden einstehen muss. Mangelhaft ist alles, was nicht dem vernünftigerweise zu erwartenden Standard, dem **Stand der Technik** entspricht. Das ist ein weites Feld. Mängel können zwar durch Gewährleistungsbehelfe geheilt werden: zB Verbesserung, Preisminderung und Vertragsauflösung. Bei Software-Mängeln ist Verbesserung aber oft nicht erwünscht, weil die fragliche Software zum Zeitpunkt des Verfahrens ohnehin nicht mehr verwendet wird. Vertragsauflösung hingegen kann für den Verkäufer Totalverlust der eingesetzten Ressourcen bedeuten. Es besteht also mitunter ein erhebliches Risiko.

Bevor wir uns zwei Beispiele ansehen, was Qualitätsmängel für die Beurteilung von Software bedeuten können, ist es wichtig, festzustellen, dass es hier nicht um inhaltliche Mängel (Verfehlung von Anforderungen) oder Probleme der Benutzer-Dokumentation geht, sondern um die Software in ihrem natürlichen Aggregatszustand: den Quellcode. Inhaltliche Mängel fallen zwar auch unter Gewährleistung, sind aber oft offensichtlich. Die Vertragspartner mögen sie verschieden bewerten, kennen sie aber. Rechtsexperten können sich auch ohne technisches Fachwissen schnell ein Bild machen. Ein Sachverständiger wird nicht benötigt. Mängel im Quellcode hingegen schlummern als technische Schulden oft unbemerkt vor sich hin. Ihr böses Erwachen betreibt üblicherweise der mit der Zahlung säumige Kunde (Mahnklage). Festgestellt werden sie vom Gutachter; dabei hilft ihm, dass bei einem kaputten Programm im Gegensatz zu einer kaputten Maschine die Ursache unversehrt bleibt.

Ein Beispiel: A entwickelt für B eine Online-Plattform und verrechnet dafür einen signifikanten Preis. B zahlt nicht. A klagt. B argumentiert, dass die Software von A nicht dem Stand der Technik entspricht, also mangelhaft ist, und verlangt Preisminderung. Das Gericht setzt einen Sachverständigen ein, der die Software anhand der gängigen Qualitätsempfehlungen untersucht: Tatsächlich hätte man, obwohl die gewünschte Funktionalität gegeben ist, einiges besser machen können. Urteil: Die Software ist mangelhaft, Preisminderung. **A ist überrascht**.

Dieses Beispiel ist der typische Fall in der Praxis des Software-Gutachters. Es zeigt, welche Gefahren in mangelnder Software-Qualität stecken. **Technische Schulden** ist hier ein sehr treffender Begriff, können sie doch leicht zu finanziellen werden. Im Folgenden sehen wir uns an, wie Gutachter die Qualität von Quellcodes messen und, darauf basierend, wie man diese „*Lege-artis*-Falle“ entschärfen kann.

2. Was ist Stand der Technik?

Eines ist sicher: **Stand der Technik** ist ein interessanter Begriff. Manche Experten sehen im Stand der Technik ausschließlich die jetzt gerade allerbeste Methode. Wer davon abweicht, befindet sich nicht auf der Höhe der Zeit. Die das so sehen, machen zum Glück dann meist auch gleich die vollständige Induktion und erklären den Stand der Technik zur Unmöglichkeit und zum Nicht-Begriff, für die tägliche Arbeit nicht einsetzbar.

Das hilft dem Praktiker natürlich wenig. Für die tägliche Arbeit nimmt der Sachverständige daher meist die **gute Ingenieurs-Praxis** als Stand der Technik an. Das ist so ziemlich alles, was sich bewährt hat und nicht klar von einer anderen Methode dominiert wird. So ist zB der Rational Unified Process ebenso in seinem Bereich noch Stand der Technik wie ein Scrum-Prozess; das Wasserfall-Modell aber nicht mehr, weil es von iterativen Modellen klar dominiert wird. Mangelhaft ist nun alles, was diesem (großzügigen) *Lege-artis*-Begriff nicht genügt. Um das zu beurteilen, müssen wir messen.

Auch das Messen ist ein weites Feld. Um es einzugrenzen, wurden zahlreiche Methoden vorgeschlagen. Etwa

* Abdruck eines populärwissenschaftlichen Zeitschriftenartikels des Autors. Zuerst erschienen in iX - Magazin für professionelle Informationstechnik 2/2013, 132 bis 136.

das Factors-Criteria-Metrics-Modell, bei dem Metriken von Schlüsselfaktoren und deren Teilkriterien abgeleitet werden. Ein ähnliches Modell, das außerdem einen Messablauf definiert, ist das in Abbildung 1 dargestellt Goals-Questions-Metrics-Modell (GQM). Es definiert den Weg zu aussagekräftigen Metriken top-down. Im ersten Schritt wird das Ziel klargemacht. Dazu benötigt es eine allgemeine, aber verbindliche Vorstellung vom Erwünschten (gute Qualität). Aus dieser Vorstellung werden konkrete Ziele abgeleitet. Die Ziele operationalisieren die Vision. Aus den Zielen lassen sich quantifizierbare Metriken ableiten. In der Praxis funktioniert das oft so, dass für die Ziele geeignete Bündel von Metriken aus den Katalogen der einschlägigen ISO-Normen ausgewählt werden.

Sind die Metriken festgelegt, ist der nächste Schritt die Planung und Durchführung der Evaluierung. Auch hier helfen oft die Normen, weil sie zu den Maßen meist auch eine Operationalisierung mitliefern: zB in Form von Fragen. Die eigentliche Bedeutung der Metriken tritt aber erst im letzten Schritt zu Tage: Die Skalen der Maße erlauben deren Bewertung. Die quantitativen Ergebnisse werden per Reflexion in Verständnis umgemünzt, das zur Erreichung der Vision genutzt wird.

Das GQM impliziert, wie jedes gute Qualitäts-Modell, einen Verbesserungsprozess. Im letzten Teil dieses Artikels werden wir uns im Detail mit diesem Aspekt auseinandersetzen. Zunächst ist aber festzustellen, dass es im Rahmen eines Gerichtsverfahrens *keinen* iterativen technischen Prozess gibt: Es geht nur um die einmalige Beurteilung. Metriken werden einmal angewandt, Aussagen einmal getroffen, Schlussfolgerungen einmal gezogen. All das bezieht sich auf den Quellcode zum Zeitpunkt der Übergabe. Weil Sachverständigen-Arbeit erhebliche Kosten nach sich ziehen kann, liegt dabei der Fokus auf jenen Aspekten, die am lohnendsten (das heißt am ehesten mangelhaft) erscheinen. Es obliegt der Partei, die sich davon einen Vorteil verspricht, diese zu nennen. Dem vorgestellten Prozess folgend, wird der Sachverständige diese Aspekte durch Metriken ausdrücken und dem GQM-Ablauf bis zur Analyse folgen. Er schneidet also aus dem Gesamtbild sowohl Funktionen (Operationalisieren, Evaluieren, Analysieren) als auch Kriterien (Metriken) aus, um möglichst schnell zu einer Ja/Nein-Aussage über die vermeintlichen Mängel zu kommen. Wie der Finanzprüfer sucht er gezielt versteckte Schulden, hier eben technische.

3. Software-Qualitätsmodelle gibt es viele

Damit wird es Zeit für mehr Details: Warum werden welche Metriken angewandt? Grundsätzlich gibt es eine Reihe von Software-Qualitätsmodellen, die Kataloge von Metriken definieren. Zwei Standardisierungsversuche sind dabei hervorzuheben, weil sie eine Systematik bieten, die über **Key Performance Indicators** hinausgehen: ISO-9126 und die erweiternde Neufassung ISO-25000. ISO-9126 definiert 71 Metriken, die in sechs Gruppen organisiert sind. Wichtige Gruppen sind Funktionalität, Zuverlässigkeit und

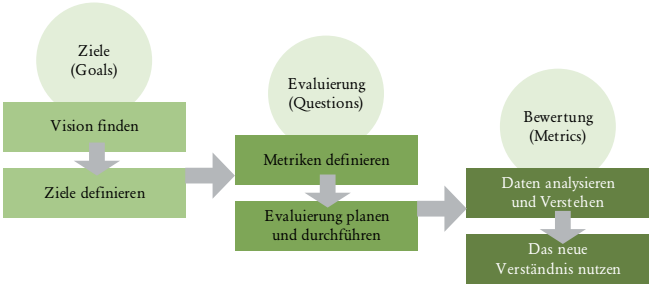

Abbildung 1: Das GQM-Modell führt top-down von Vorstellungen über Messungen zum besseren Verständnis

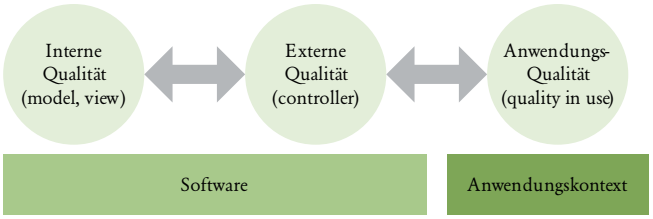

Abbildung 2: Das dreiteilige Qualitätsmodell der ISO-9126-Norm betont insbesondere die Anwendungs-Qualität

Benutzbarkeit. Ein typisches Maß für die Benutzbarkeit ist die **Vollständigkeit der Benutzerdokumentation**, operationalisiert als das **Zählen der Anzahl der Funktionen, die mit Hilfe und/oder Dokumentation implementiert wurden, relativ zur gesamten Anzahl der Funktionen**. Das Ergebnis ist ein Indikator, der im Intervall [0,1] misst, wobei *1* das gute Ende ist.

Die ISO-9126-Norm liefert aber mehr als nur eine Festlegung gängiger Qualitätsaspekte und Metriken. Abbildung 2 illustriert das dem Standard zugrunde liegende Qualitätsmodell, das auch für den Nachfolger ISO-25000 bestimmend ist. Qualität hat aus dieser Sicht drei wesentliche Zonen: Die interne entspricht dem, was im MVC-Schema Model und View sind. Dazu gehört auch der Quellcode. Externe Qualität bezieht sich vor allem auf die Benutzerschnittstelle, also den Controller. Hier gibt es, nebenbei bemerkt, auch noch die ISO-Norm 9241 zur Gestaltung von Benutzerschnittstellen, wobei Teil 151 recht praktische Richtlinien zur Gestaltung und Bewertung von Web-Benutzerschnittstellen definiert. Anwendungs-Qualität fasst alle Aspekte der praktischen Nutzung zusammen. Dazu gehört zB die Effektivität des Einsatzes.

Die Mangelsuche des Gerichtsgutachters lässt sich in diesem Modell relativ klar verorten: Sie findet hauptsächlich bei der internen Qualität statt. Externe Qualität können Benutzer heute oft selbst recht gut beschreiben und beurteilen; ganz vernachlässigt darf sie trotzdem nicht werden: Bei strittigen Fragen hilft die Norm. Anwendungsqualität ist selten relevant, weil etwaige Mängel dort meist klar zu Tage treten.

Die ISO-9126-Norm wurde in jüngster Zeit von der Normgruppe ISO-25000 abgelöst. Dabei definiert ISO-25010 (basierend auf SQALE) das Modell und ISO-25020 die

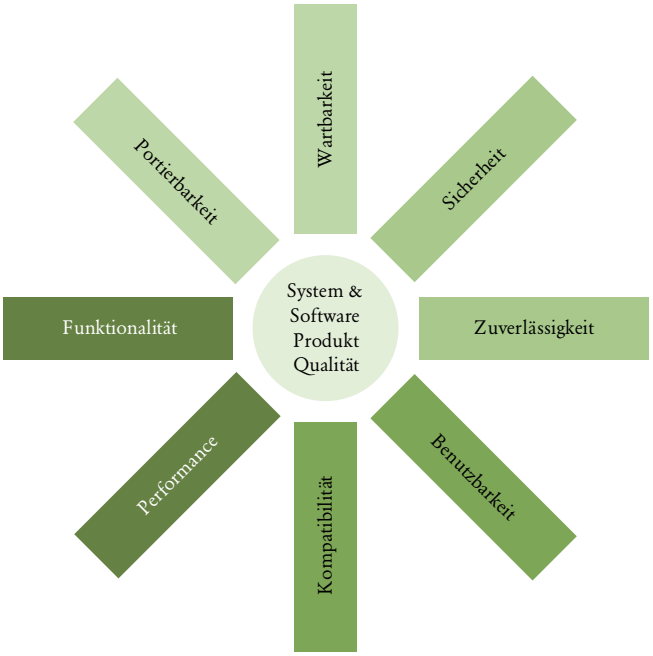

Abbildung 3: Für die Autoren der Normengruppe ISO-25000 setzt sich gute Qualität aus acht Funktionen zusammen

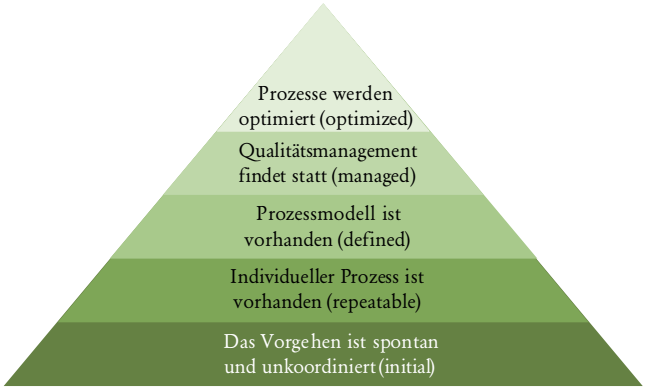

Abbildung 4: Capability Maturity Model: In fünf Stufen zum spitzenmäßigen Vorgehen

Maße. Bemerkenswert ist, dass der Teil ISO-25012 ein Qualitätsmodell für die Datensicht (Modell) definiert, das über die Grundpfeiler Normalisierung, Indexierung und Abfrageoptimierung hinausgeht. Die Metriken basieren oft noch auf der ISO-9126. Neu ist aber die Einteilung der internen und externen Qualität in die in Abbildung 3 dargestellten acht Funktionen. Dazu gekommen sind gegenüber der alten Norm Kompatibilität und Sicherheit, letztere Funktion ist in der Tat ein Leitmotiv der Informatik im letzten Jahrzehnt.

Eine andere Neuerung ist die Differenzierung der Anwendungsqualität in sechs Funktionen: Effektivität, Effizienz, Zufriedenheit, Risikofreiheit und Kontext-Abdeckung. Zur letzten Funktion gehört unter anderem die Flexibilität des Einsatzes einer Software: je flexibler, umso besser. Wiederum gilt, dass Anwendungsqualität für das beschriebene Problem nur von nachgeordneter Bedeutung ist. Sie wird eher durch die Willenserklärungen der Parteien determiniert und dokumentiert, als durch einen Sachverständigen *ex post* bestimmt.

4. Der Mix des Praktikers

Bei der geschilderten Fülle an Möglichkeiten stellt sich die Frage, welche Funktionen und Metriken nun tatsächlich häufig zur Mangelsuche verwendet werden und welche eher von akademischem Interesse sind. Das ist – im wahrsten Sinne des Wortes – von Fall zu Fall verschieden. Über die Zeit aggregiert lassen sich jedoch zumindest die in den folgenden Absätzen beschriebenen Häufungen feststellen. Dabei geht es primär um den Quellcode, sekundär um die Benutzerschnittstelle, die Datenbank und die Laufzeitumgebung.

Das fundamentale Kriterium guten Quellcodes ist eines, das eigentlich unsichtbar sein sollte: der Entwicklungsprozess. Wir haben oben festgestellt, dass zumindest alles, was nicht iterativ ist, nicht evaluiert und nicht lernt, auch nicht Stand der Technik ist. Über alles andere kann man reden: Agile Methoden haben ihre Tücken, wenn es um die Dokumentation geht, nicht agile scheitern oft an schwammigen oder schlicht falschen Anforderungen. Messen kann man den Entwicklungsprozess auf vielfache Weise. Eine Möglichkeit ist die Anwendung der ISO-15504-Norm (SPICE). Für die gerichtliche Anwendung ist das Ergebnis aber oft zu kompliziert zu lesen. Leichter zu verstehen ist das Capability Maturity Model (CMM), das in Abbildung 4 dargestellt ist. Es bestimmt, wie stark ein (Entwicklungs-) Prozess formalisiert ist. Das ist hier etwas Positives: Es bedeutet überlegt und optimiert. Der CMM-Wert (initial, wiederholbar etc) lässt sich leicht bestimmen und ist aussagekräftig. Er eignet sich daher gut als erster **Key Performance Indicator**. Besonders weit kann er uns naturgemäß aber nicht bringen. Ein Projekt im initialen Zustand wird häufiger mangelhaft sein, die Mangelhaftigkeit ist jedoch im Detail zu bestimmen.

In der Quellcode-Analyse gibt es zahlreiche Kriterien für Mangelfreiheit. Dabei ist oft die Zeilenzahl (Standardized Lines of Code, LOC) die Referenz. Im Verhältnis zur LOC kann man zB die Anzahl der Methoden bestimmen (Spaghetti-Code?), die Anzahl der Patterns (ein schöner **Code Smell** für mangelnde Wiederverwendbarkeit?), die Anzahl der Tests (Fehlerfreiheit durch ausreichende Testüberdeckung?) und so weiter. Wichtige Kriterien der Dokumentation lassen sich so bestimmen: Wie viele Zeilen Modul-, API- und Inline-Dokumentation gibt es relativ zur Anzahl der Klassen, Methoden, Ressourcen? Kriterien der Lesbarkeit sind die Einhaltung von Kodier-Konventionen und allgemeinen Best Practices.

Auf formaler Ebene können Kriterien für Software-Defekte zur Anwendung gebracht werden. Etwa die Indikatoren von *McCabe*, die Komplexität aufgrund des Graphenmodells eines Algorithmus beurteilen. Freilich liegt die Relevanz

solcher Methoden meist nicht im Feld der Business-Software, das wiederum für die überwiegende Mehrzahl der Software-Mahnklagen relevant ist.

Benutzbarkeit und Datenqualität sind Klassiker der Anwendungssoftware-Entwicklung. Im ersteren Feld gibt es zur Beurteilung von Mängeln die bereits erwähnte ISO-9241-Norm, deren Praxisnähe der gerichtlichen Nutzung sehr entgegenkommt. Für die Datenqualität verspricht die neue ISO-25012-Norm eine Verbesserung. Was Datenqualität sein soll, ist zwar seit langer Zeit unstrittig, die praktische Definition von Metriken sollte die Beurteilung adäquater Abstraktion, Indexierung, Abfragen-Optimierung etc aber spürbar vereinfachen. Nebenbei sei festgehalten, dass man als Gutachter interessanterweise nur selten auf eine größere Anwendung trifft, deren Datenmodell sauber normalisiert ist. Hier weicht die betriebliche Praxis stark von der reinen Lehre ab, weshalb sich die Frage stellt, ob man das dann überhaupt noch mangelhaft nennen darf.

Schließlich sei hier noch auf die Bedeutung guter Wartbarkeit und Sicherheit hingewiesen. Relevante Aspekte sind Ausmaß und Form der Verwendung von Dritt-Software, die Ausformung der Laufzeitumgebung und natürlich Sicherheitsmechanismen aller Art auf den verschiedensten Ebenen. Insbesondere die Wartbarkeit lässt sich leider oft nur schwer in Metriken fassen. Auch die Standards bieten hier nicht viel mehr als Gemeinplätze. Trotzdem wird dieser Bereich für die Beurteilung potenzieller Mangelhaftigkeit (dann eben qualitativ) immer wichtiger.

5. Mit dem Verbesserungsprozess zur Mangelfreiheit

Soweit die Vorgangsweise des Sachverständigen. Wie kann man sich nun vor den Gefahren versteckter Software-Mängel schützen? Hier gilt der alte Grundsatz, dass Vorbeugen besser als Heilen ist. Wer vorweisen kann, dass er das vernünftigerweise Machbare getan hat, der wird im Fall des Falles gute Karten haben. Mängel mögen trotzdem existieren: Vollständige Mangelfreiheit ist ein lebensfremdes Ideal. Aber sie werden von geringerer Bedeutung und vielleicht auch schwieriger festzustellen sein – und damit steigt das Prozessrisiko für den, der sie behauptet.

Was also benötigt wird, ist ein periodischer Verbesserungsprozess, der die wesentlichen Aspekte guter Software-Qualität misst, dokumentiert und etwaige Verbesserungen urgiert. Leider gibt es derzeit keine ISO-Zertifizierung für die Qualität von Software. Experten können jedoch diese Leistung erbringen, indem sie die oben beschriebenen Methoden im Rahmen periodischer Audits anwenden. Abbildung 5 skizziert einen Verbesserungsprozess, der gedanklich auf der ISO-9000-Qualitätsnorm basiert und dem SPICE-Modell, aber auch SCAMPI im CMM ähnelt. Die Qualitätsbeurteilung kann hier zB auf dem oben beschriebenen GQM-Modell basieren. Ergebnis kann ein Qualitäts-Zertifikat sein. Dass dieses **unabhängig von einem Rechtsstreit periodisch** geprüft und

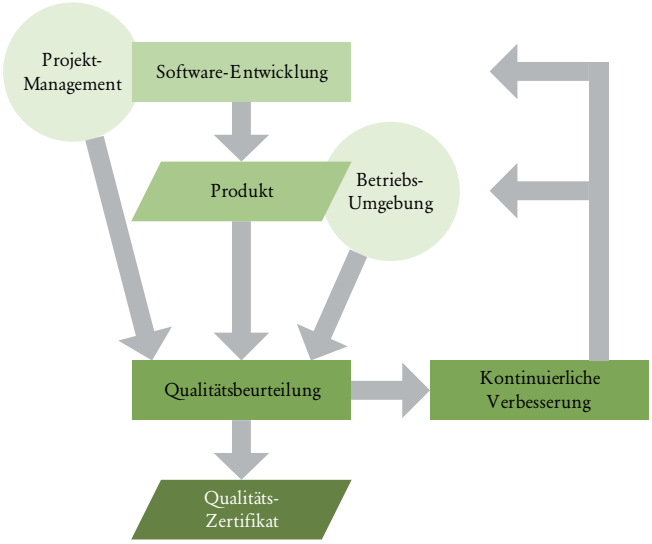

Abbildung 5: Mangelfreiheit so gut wie möglich mit einem Standard-Verbesserungsprozess des Qualitätsmanagements

erneuert wird, stellt einen besonderen Wert in Form von Glaubwürdigkeit dar.

Zur Qualitätsbeurteilung müssen auf jeden Fall auch Aspekte des Projekt-Managements und der Laufzeitumgebung berücksichtigt werden. Die Messung dieser Kriterien lässt sich kaum automatisieren. Viele Metriken, die beschreiben, wie hoch die interne Qualität ist, lassen sich jedoch relativ leicht berechnen. Unix-Shell-Werkzeuge und Skriptsprachen mit regulären Ausdrücken eignen sich für solche Aufgaben hervorragend. So lassen sich auch äußerst umfangreiche Projekte schnell und billig beurteilen.

6. Fazit und Ausblick

Es kommt, was den Mangel betrifft, eine alte Phrase zu Ehren: Der Teufel steckt im Detail. Im vorliegenden Fall können Details viel Geld kosten und große Mühen verursachen. Nicht allein der Preis bestimmt den Wert einer Software. Auch die Freiheit von Mängeln **so gut als möglich** muss dokumentiert sein. Der Weg dahin führt über die periodische Beurteilung der wichtigsten Software-Qualitäts-Indikatoren, wie sie die einschlägigen Normen definieren.

Was wird die Zukunft bringen? Grundsätzlich ist davon auszugehen, dass es eher noch mehr als weniger solcher Fälle geben wird. Software ist allgegenwärtig, ihre Bedeutung wird in allen Bereichen der Gesellschaft weiter steigen, die Systeme werden immer komplexer, was dem Mangel neue Räume eröffnet. Auf technischer Ebene bringt die ISO-25000-Normgruppe eine signifikante Konsolidierung der Software-Qualitätsmessung. Zu ihrer Anwendung bedarf es aber der Übung und Glaubwürdigkeit. Mit diesen technischen Werten lassen sich nach und nach technische Schulden abbauen.

Online-Quellen

[1] Capability Maturity Model (CMM): http://www.sei.cmu.edu/cmmi/

[2] Goal Question Metrics Model (GQM): ftp://ftp.cs.umd.edu/pub/sel/papers/gqm.pdf

[3] ISO-9126/2003: Software-Produkt-Qualität

[4] ISO-9241/2008: Gestaltung von Webschnittstellen

[5] ISO-15504/2004 (SPICE): Prozessbewertung in der Informationstechnologie

[6] ISO-25010/2011: System und Software Qualitäts-Anforderungen

[7] ISO-25012/2008: Daten-Qualitäts-Modell

[8] Software Quality Assessment based on Lifecycle Expectations (SQALE): http://www.sqale.org/

Alle zuletzt überprüft am 11. 10. 2013.

Zum Autor:

Dr. Horst Eidenberger arbeitet als außerordentlicher Universitätsprofessor am Institut für Softwaretechnik und Interaktive Systeme der TU Wien an Modellen zur maschinellen Medienwahrnehmung. Daneben ist er allgemein beeideter und zertifizierter Sachverständiger für Informatik und Signalverarbeitung.

Korrespondenz:
ao. Univ.-Prof. Dr. Horst Eidenberger
Institut für Softwaretechnik und Interaktive Systeme, TU Wien
Favoritenstrasse 9/1882, 1040 Wien
E-Mail: eidenberger@tuwien.ac.at