Ausarbeitung Softwarequalität

Software-Qualitätsbewertung mittels Software Metriken

Softwarequalität
Duale Hochschule Baden-Württemberg
Stuttgart

von
Paul Walker und Leon Kampwerth

Matrikelnummer: 3610783, 5722356
Abgabedatum: 22.03.2023

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abstract

Softwarequalität zu messen kann sich mitunter als schwierig herausstellen. Diese Arbeit hat das Ziel einfache Metriken für Qualitätscharakteristika zu erarbeiten. Diese Metriken sollen einfach automatisiert messbar sein und so einen kontinuierlichen Indikator für die Softwarequalität geben, der auch in Continuous Integration / Continuous Delivery (CI/CD) Prozesse genutzt werden kann. Der Fokus wird auf die Qualitätscharakteristika Wartbarkeit und Sicherheit gesetzt.

1 Einleitung

Die meisten Menschen arbeiten jeden Tag mit verschiedener Software. In der Arbeit mit Office-Software oder je nach Beruf spezialisierter Software für bestimmte Aufgaben und/oder Maschinen. Zu Hause nutzen viele Menschen Unterhaltungssoftware wie Spiele, oder Videoplattformen. Software ist heutzutage allgegenwärtig und die meisten Menschen, werden auch schon einmal Software schlechter Qualität genutzt haben. Das kann durch abruptes Schließen einer Applikation, unintuitives Design oder langer Ladezeiten für den Nutzer erkennbar sein. Es kann recht einfach sein, schlechte Qualitäten in Software zu erkennen, allerdings lässt sich Softwarequalität so nicht Objektiv vergleichen.

Diese Arbeit beschäftigt sich damit Metriken zu finden, mit denen Software Qualität Objektiv gemessen werden kann. Softwaremetriken werden hauptsächlich in der Softwareentwicklung genutzt. Während der Softwareentwicklung können Software Metriken Fortschritte erkennbar machen und vor Rückschritten warnen. Software Metriken werden von Entwicklern genutzt, um die Auswirkung von Änderungen im Quellcode auf zum Beispiel die Softwarequalität erkennen zu können und reagieren zu können. Für die Stakeholder, also Kunden oder Investoren, sind Softwaremetriken interessant, um die Arbeit der Entwickler nachvollziehen zu können und um die Softwareentwicklung in bestimmte Richtungen zu leiten.

2 Softwarequalität

Um Softwarequalität messen zu können muss zuerst festgelegt werden, was Softwarequalität überhaupt ist. Viele Aspekte von Softwarequalität sind für den Endnutzer gar nicht erkennbar oder sind nur dann erkennbar, wenn dieser Aspekt eine schlechte Qualität hat.

In dieser Arbeit wird die Definition von Softwarequalität nach International Organization for Standardization (ISO) 25010 [6] verwendet. In dem ISO 25010 Standard wird ein Produktqualitätsmodell für System und Softwarequalität. Nach diesem Modell wird die Qualität eines Softwaresystems oder Produkts durch acht Charakteristika definiert. Diese Charakteristika werden in Abbildung 1 dargestellt.

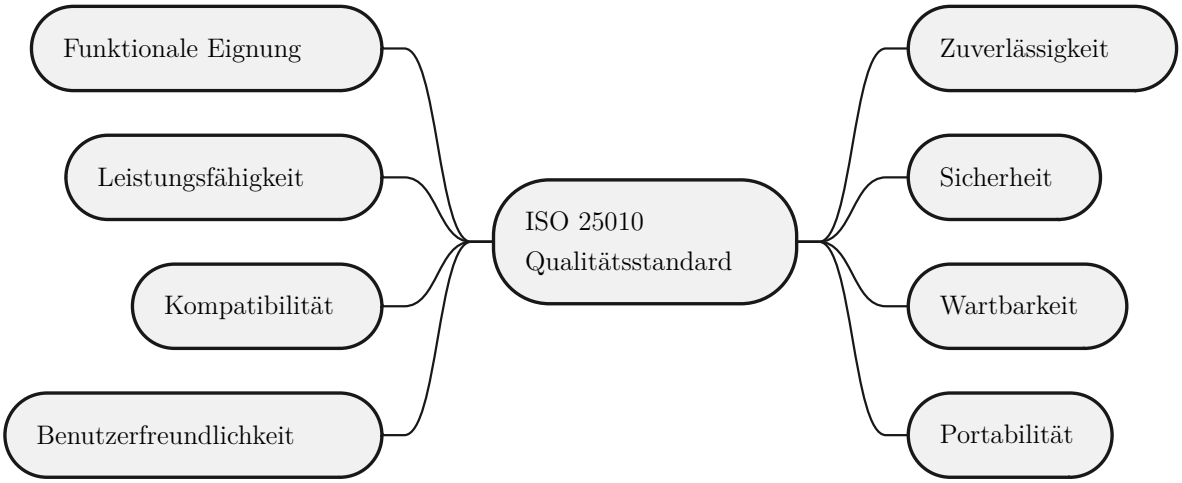

Abbildung 1: Softwarequalität nach ISO 25010 [6]

Funktionale Eignung

Die funktionale Eignung hängt davon ab wie gut funktionale Anforderungen erfüllt werden, da gemessen wird, zu welchem Grad die vom Nutzer genannten und implizierten Anforderungen und Bedürfnisse erfüllt werden [6].

Leistungsfähigkeit

Die Leistungsfähigkeit beschreibt die Leistung von Software abhängig von den vorhandenen Ressourcen [6]. Dabei meinen Ressourcen sowohl Hardwareressourcen wie Rechenleistung oder Speicher als auch Softwareressourcen wie Konfigurationen oder genutzte Bibliotheken.

Kompatibilität

Der Grad zu dem ein Softwareprodukt oder System mit anderen Softwareprodukten oder Systemen Informationen austauschen kann wird durch die Kompatibilität beschrieben [6]. Außerdem beschreibt die Kompatibilität inwiefern verschiedene Softwaresysteme oder Produkte die gleiche Hardware oder Softwareumgebung nutzen können, ohne dass dies zu Problemen führt.

Benutzerfreundlichkeit

Der Grad der Benutzerfreundlichkeit ist nach ISO, wie effektiv und effizient ein definiertes Ziel von bestimmten Nutzern mit einem Softwareprodukt oder System erreicht werden kann und wie Zufrieden die Nutzer dabei sind [6].

Teile der Benutzerfreundlichkeit sind von der ISO 9241-210 Norm zur Ergonomie von Mensch-System-Interaktionen adaptiert.

Zuverlässigkeit

Zuverlässigkeit ist die Fähigkeit eines Softwareprodukts oder Systems die erforderlichen Funktionen des Produkts oder Systems über einen bestimmten Zeitraum und unter bestimmten Bedingungen zu erfüllen [6]. Unzuverlässige Software könnte beispielsweise durch Fehler in der Software in ihrer Ausführung unterbrochen werden, wodurch die erforderlichen Funktionen dann nicht mehr in dem bestimmten Zeitraum ausgeführt werden können, sondern beispielsweise durch einen Neustart der Software länger Zeit benötigen.

Sicherheit

Die Sicherheit von Software ist der Grad zu dem die Software Informationen und Daten schützt. Informationen und Daten sind dann geschützt, wenn andere Personen oder Produkte nur Zugriff auf Daten entsprechend ihrer Berechtigung haben [6]. Dabei darf sich die Sicherheit der Daten weder beim Transport der Daten noch währen eines Angriffs auf die Software ändern [6].

Auf die Sicherheit wird in Abschnitt 2.1 weiter eingegangen.

Wartbarkeit

Der Grad der Wartbarkeit beschreibt die Effektivität und die Effizienz mit der ein Softwareprodukt oder System modifiziert werden kann.

Auf die Wartbarkeit wird in Abschnitt 2.2 weiter eingegangen.

Portabilität

Die Portabilität ist ein Maßstab dafür, wie effektiv und effizient ein Produkt oder System von einer Hardware-, Software- oder einer bestimmten Betriebs- oder Nutzungsumgebungen auf andere übertragen werden kann [6]. Portabilität kann auch als die Eigenschaft eines Systems bezeichnet werden, die es erlaubt, das System einfach auf verschiedene Konfigurationen anpassen zu können [10].

Jedes dieser Charakteristika wird weiter in Subcharakteristika aufgeteilt. In dieser Arbeit wird der Fokus auf zwei bestimmte Charakteristika gesetzt und für jedes eine Metrik vorgestellt, mit der die Qualität der Charakteristik gemessen werden kann.

Eine einzelne Metrik für Softwarequalität zu erstellen ist schwer, da die Softwarequalität durch so viele Charakteristika beeinflusst wird. Außerdem würde eine einzelne Metrik für die Softwarequalität nur bedingt nützlich sein. Es wäre zwar möglich die generelle Qualität einer Software zu quantifizieren, allerdings lässt sich dann nicht herausfinden, in welchen Bereichen Mängel an der Qualität der Software vorliegen. Es ist sinnvoller einzelne Metriken für die Qualitätscharakteristika zu erstellen, wie es auch in dieser Arbeit gemacht wird, da so die Ursache für potenzielle Qualitätsmängel einfacher gefunden werden kann.

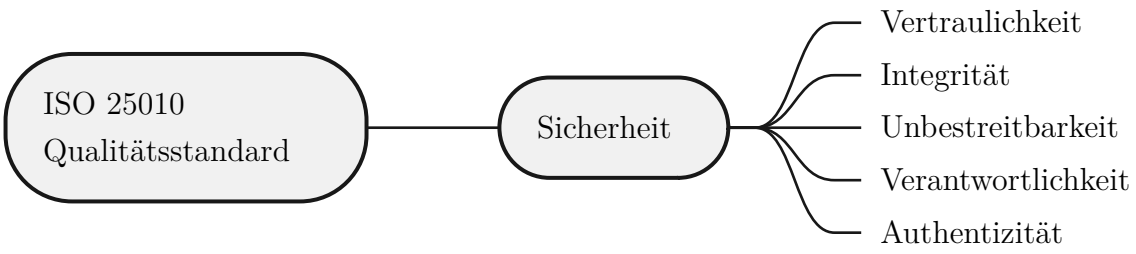

Abbildung 2: Sicherheit nach ISO 25010 [6]

2.1 Sicherheit

Sicherheit beschreibt nach ISO, wie gut ein Produkt oder ein System die Informationen und Daten schützt. Ziel ist es sicherzustellen, dass Personen und andere Systeme nur den Grad an Zugriff auf die Daten haben, für den sie berechtigt sind [6]. Dies bezieht sich sowohl auf die Daten, die vom System gespeichert werden, als auch auf die Daten, die vom System übertragen werden [6]. Sicherheit ist ein wichtiges Qualitätsmerkmal für Software und verbessert das Vertrauen der Benutzer und der Stakeholder in die Software [6]. Die Sicherheit ist nach ISO in weitere Subcharakteristika aufgeteilt. Diese Subcharakteristika sind in Abbildung 2 dargestellt.

Ähnlich zur Wartbarkeit ist auch Sicherheit ein sehr wichtiges Qualitätscharakteristika. Sicherheit wird jedoch nur dann bemerkt wird, wenn sie abwesend ist und die Folgen fehlender Sicherheit zeigen sich meist erst einige Zeit nach der Entwicklungsphase. Daher hat die Sicherheit der Software für viele Entwickler eher eine geringere Priorität und die Erfahrung mit dem Themenbereich ist eher gering [11] [12]. Trotz einem starken Anstieg an Hackerattacken (insbesondre durch den Einsatz von Ransomware) in den letzten Jahren [2] [3] ist Sicherheit weiterhin kein festes Kriterium in der Design- und Entwicklungsphase vieler Teams. Stattdessen wird die Sicherheit meist kurz vor dem Ende der Entwicklung bedacht [11], was die Möglichkeit zum Beheben von architekturellen oder groben Entwicklungsfehlern sehr schwer macht. Um die Qualität der Sicherheit einer Software zu erhöhen, muss sie schon vor der eigentlichen Entwicklung, in der Designphase, mit bedacht werden. Ähnlich zur Wartbarkeit gilt es Regeln und Vorschriften zu definieren, die über die Entwicklung hinweg eingehalten werden.

2.1.1 Vertraulichkeit

Vertraulichkeit misst, wie gut das System sicherstellen kann, dass ein Zugriff auf Daten nur von berechtigten Benutzern durchgeführt werden kann [6].

2.1.2 Integrität

Die Integrität eines Systems beschreibt wie gut es den unauthorisierten Zugriff oder die Modifikation auf seine Programme oder Daten verhindern kann [6].

2.1.3 Unbestreitbarkeit

Der Grad der Unbestreitbarkeit beschreibt, wie gut ein System beweisen kann, dass eine Aktion oder ein Ereignis stattgefunden hat. So soll es einer Entität nicht möglich sein, dies zu bestreiten [6].

Nach der ISO gibt es zwei Möglichkeit dies zu erreichen. Der Ursprungsnachweis ist die erste Möglichkeit. Hier erhällt das System (der Empfänger) einen Nachweis, wer die Aktion oder das Ereignis ausgelöst hat. Dies verhindert, dass eine Entität bestreiten kann, dass sie die Aktion oder das Ereignis ausgelöst hat [8]. Die zweite Möglichkeit ist der Zustellnachweis. Hier erhält die Entität, die die Aktion oder das Ereignis ausgelöst hat, einen Nachweis, dass das System (der Empfänger) die Aktion oder das Ereignis durchgeführt hat. Dies verhindert, dass das System bestreiten kann, dass es die Anweisungen erhalten hat [8]. Beide Möglchkeiten können einzeln eingesetzt werden oder auch kombiniert werden.

2.1.4 Verantwortlichkeit

Verantwortlichkeit gibt an wie gut die Aktionen einer Entität (z.B. Person, System) auf dieses zurückgeführt werden können [6]. Die Verantwortlichkeit kann durch die *Unnachgibigkeit* und die *Integrität* des Systems beeinflusst werden.

2.1.5 Authentizität

Der Grad der Gewissheit, mit dem die Identität einer Entität (z.B. Person, System) bewiesen werden kann wird durch die Authentizität definiert [6]. Das Ziel ist es mit Sicherheit nachweisen zu können, das eine Entität auch die ist, die sie angibt zu sein [7].

2.2 Wartbarkeit

Der Grad der Wartbarkeit ist nach ISO, wie effektiv und effizient das Produkt oder System verändert werden kann [6]. Diese Veränderungen können Korrekturen, Verbesserungen oder Anpassungen der Software auf Änderungen des Einsatzgebiets oder auf Änderungen der Anforderungen sein [6]. Die Wartbarkeit bezieht sich auch auf die Installation von Updates [6]. Die Wartbarkeit ist nach ISO in weitere Subcharakteristika aufgeteilt. Diese Subcharakteristika sind in Abbildung 3 dargestellt.

Die Wartbarkeit ist eine der wichtigsten Qualitätscharakteristika für Entwickler. Viele andere Qualitätscharakteristika wie die Benutzerfreundlichkeit oder die Leistungsfähigkeit bewerten äußere Eigenschaften, die für den Entwicklungsprozess nicht generell eine Rolle spielen. So ist die Leistungsfähigkeit von Software meist nur in kritischen Bereichen der Software relevant. Die Benutzerfreundlichkeit von Software spielt in der Regel nur in Bereichen, mit denen ein Benutzer interagiert eine Rolle. Dagegen ist die Wartbarkeit von Software immer dann relevant, wenn an und mit bestehendem Quellcode gearbeitet werden muss. Softwareprodukte oder Systeme, die eine hohe Wartbarkeit aufweisen, werden in der Entwicklung und der Wartung einfacher sein als Softwareprodukte oder Systeme, die eine geringe Wartbarkeit aufweisen.

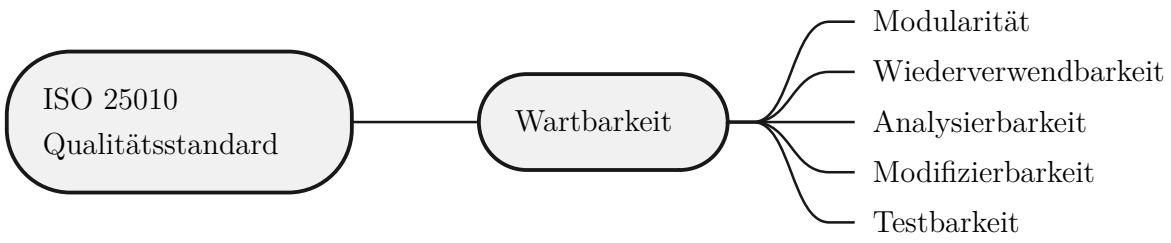

Abbildung 3: Wartbarkeit nach ISO 25010 [6]

Außerdem muss während des ganzen Entwicklungsprozess darauf geachtet werden die potenziell geltenden Regeln und Vorschriften zur Wartbarkeit von Software einzuhalten.

2.2.1 Modularität

Die Modularität wird daran gemessen, inwieweit das System oder Programm aus einzelnen Teilen besteht und Änderungen an einem Teil der Software andere Teile nur minimal oder gar nicht beeinflussen [6]. Module sind Softwareattribute, die Software in unabhängige Einzelteile trennen [10].

2.2.2 Wiederverwendbarkeit

Die Wiederverwendbarkeit ist der Grad, zu dem ein Teil des Systems in mehr als einem System oder beim Erstellen von weiteren Teilen des Systems verwendet werden kann [6]. Die Wiederverwendung bezieht sich dabei nicht nur auf Software, sondern auch auf Dokumentation, Software-Design, Applikationstests und weitere Teile des Systems [5].

2.2.3 Analysierbarkeit

Wie effektiv und effizient es möglich ist, den Einfluss auf ein Produkt oder System abzuschätzen, wenn ein oder mehrere Teile verändert werden, ist durch die Analysierbarkeit definiert [6]. Außerdem wie gut der Ursprung von Fehlern im Produkt oder System diagnostiziert werden kann [6].

2.2.4 Modifizierbarkeit

Die Modifizierbarkeit ist ein Maßstab dafür wie effektiv und effizient sich ein Produkt oder System ändern lässt, ohne dabei Defekte einzuführen oder die Produktqualität zu verschlechtern [6].

Die Modifizierbarkeit kann durch die *Modularität* und die *Analysierbarkeit* beeinflusst werden [6].

2.2.5 Testbarkeit

Der Grad der Testbarkeit ist, wie effektiv und effizient Testkriterien für ein Produkt oder System definiert werden können und wie effektiv und effizient Tests für das

Produkt oder System durchgeführt werden können [6]. Zusätzlich zur Erstellung der Testkriterien und der Durchführung der Tests, spielt es auch eine Rolle wie gut objektive und messbare Tests erstellt werden können, die feststellen, ob ein bestimmtes Kriterium erfüllt wurde [10].

3 Software Metriken

Software Metriken sind quantitative Maße, die zur Bewertung von Softwareprodukten und Softwaresystemen genutzt werden können. Diese Metriken können verschiedene Aspekte der Software Bewerten. Hier werden Metriken vorgestellt, mit denen die Wartbarkeit und die Sicherheit analysiert werden könne.

Software Metriken geben sowohl den Entwicklern als auch den Stakeholdern der Software Informationen über Eigenschaften der Software. Diese Informationen können nützlich sein, um Risiken frühzeitig zu erkennen, Verbesserungen und Verschlechterungen zu quantifizieren und verschiedene Software vergleichen zu können. Software Metriken können in der Kommunikation zwischen Entwicklern und Stakeholdern helfen, da sie helfen Eigenschaften der Software zu objektifizierten und sich so beispielsweise bestimmte Entwicklungstätigkeiten, wie das Refactoring von Software besser gegenüber den Stakeholdern rechtfertigen lassen. Stakeholder können mithilfe von Softwaremetriken den Entwicklungsprozess besser nachvollziehen und so besser und gezielter reagieren. Wenn beispielsweise die Entwicklung eines neuen Features die Softwarequalität stark verschlechtert, kann mehr Zeit für die Entwicklung dieses Features eingeplant werden, damit Qualitätsstandards eingehalten werden können.

Mit Softwaremetriken die, die Softwarequalität quantifizieren können Regeln eingeführt werden, die bestimmte Mindestanforderungen an die Softwarequalität stellen. So kann in der Theorie eine konstante objektiv gute Qualität der Software gewährleistet werden. Um diese Regeln umzusetzen ist es hilfreich die nötigen Metriken automatisch zu generieren und Änderungen an der Software nur dann zu erlauben, wenn die Regeln eingehalten werden.

3.1 Sicherheit

3.2 Wartbarkeit

Viele Charakteristika der Wartbarkeit lassen sich über die Dauer oder Effizienz von Wartungsaufgaben messen. Es kann beispielsweise die Zeit gemessen werden, die benötigt wird, um eine Änderung an der Software durchzuführen, um damit die Modifizierbarkeit zu charakterisieren [4].

Diese Metriken lassen sich allerdings nur bei konkreten Wartungsaufgaben messen. Das ist problematisch, da so die Qualität der Software nur durch großen Aufwand bestimmt werden kann. Ziel hier ist es daher eine Metrik zu finden, mit der sich die Wartbarkeit ohne großen Aufwand bestimmen lässt. Das Paper *A Practical Model for Measuring Maintainability* von Ilja Heitlager, Tobias Kuipers und Joost Visser [4] stellt eine

Möglichkeit dar, die Wartbarkeit ohne großen Aufwand zu messen. Allerdings basiert diese Paper auf dem älteren ISO 9126 Qualitätsmodell [4, 9]. In dieser Arbeit wird die Metrik, die in [4] daher erweitert und auf das neuere ISO 25010 Qualitätsmodell angepasst.

Für diese Metrik werden einige einfache Kennzahlen der Software kombiniert, um Kennwerte für die Jeweiligen Subcharakteristika der Wartbarkeit zu berechnen. Diese Kennwerte können dann verwendet werden, um einen Gesamtwert für die Wartbarkeit der Software zu bekommen. Die Kennzahlen die Genutzt werden sind.

Quellcodevolumen

Die Zahl der effektiven Quellcodezeilen im gesamten Bereich der analysiert werden soll. Je geringer das Quellcodevolumen ist, desto einfacher ist es eine Übersicht über den Bereich zu bekommen, der analysiert werden soll. Also ist niedriger besser.

Komplexität der Einheiten

Die zyklomatische Komplexität einer Einheit sagt aus wie Komplex eine Einheit ist, und damit auch wie komplex Wartungsaufgaben sein können. Eine Einheit ist dabei das kleinste ausführbare Stück Code [4]. Je nach Programmiersprache kann eine Einheit eine Funktion, eine Klasse oder aber das gesamte Programm sein [4]. Eine geringere Komplexität spricht für besser wartbare Software. Die zyklomatische Komplexität von Software kann mit Tools wie *scc* [1] automatisiert gemessen werden. Hier ist also niedriger besser.

Quellcode Duplikation

Quellcode Duplikation ist definiert durch die Anzahl Quellcode Zeilen, die eins zu eins an verschiedenen Stellen zu finden sind. Duplikation spricht für Probleme in der Software. Also ist für eine höhere Qualität der Software eine niedrige Duplikation besser.

Einheitengröße

Eine geringe Einheitengröße sorgt für eine höhere Softwarequalität, da kleinere Einheiten einfacher zu warten sind.

Unit-Testabdeckung

Die Testabdeckung durch Unit-Tests wird häufig verwendet, um Aussagen über die Softwarequalität zu machen. Hier wird diese Metrik auch verwendet. Allgemein gilt, je mehr Unit-Tests vorhanden sind, desto besser.

Modulanzahl

Die Modulanzahl gibt Aussage über die Modularisierung von Software. Je höher die Modulanzahl, desto modularisierter ist die Software. Hier ist also höher besser.

Wiederverwendung von Einheiten

Die Wiederverwendung von Einheiten sagt aus, wie oft eine Einheit an anderer Stelle im Quellcode wiederverwendet wird. Das kann gemessen werde, indem gezählt wird wie oft Klassen genutzt werden oder wie oft Funktionen aufgerufen werden. Hier gilt, höher ist besser.

Diese Kennzahlen werden wie folgt kombiniert, um Metriken für die Subcharakteristika zu erhalten.

Modularität

Eine einfache Metrik für die Modularität ist die Modulanzahl in dem betrachteten Teil der Software. Nachteil dieser Metrik ist, dass sie nicht aussagt, inwieweit Änderungen an einem Modul die anderen Module beeinflussen. Nach ISO 25010 sollen Änderungen an einem Modul, die anderen Module möglichst wenig beeinflussen [6].

Eine andere Metrik ist die Einheitengröße. Diese gibt auch eine gewisse objektive Aussage über die Modularität. Für eine hohe Modularität ist eine geringe Kopplung und hohe Kohäsion, also hohe Zusammengehörigkeit, innerhalb der Einheiten nötig um zu gewährleisten, dass Änderungen an einem Modul, die anderen Module minimal oder gar nicht beeinflussen. Viele kleine Einheiten sprechen für eine hohe Kohäsion, da in einer kleinen Einheit nur wenig Funktionalität, die dann aber stark zusammenhängt, implementiert werden kann.

Sowohl die Modulanzahl als auch die Einheitengröße werden für die Modularität-Metrik verwendet.

Wiederverwendbarkeit

Die Wiederverwendbarkeit lässt sich nicht direkt messen, allerdings kann gemessen werden wie oft Softwarekomponenten tatsächlich wiederverwendet werden, also die Wiederverwendung. Bei hoher Wiederverwendung muss auch die Wiederverwendbarkeit hoch sein. Eine niedrige Wiederverwendung ist ein starker Indikator für niedrige Wiederverwendbarkeit.

Eine weitere Metrik für die Wiederverwendung ist die Code-Duplikation und die Einheitengröße. Wenn hohe Code-Duplikation vorhanden ist, ist die Wiederverwendung in der Regel gering, da duplizierter Code oft wiederverwendbarer implementiert werden kann. Bei geringer Code-Duplikation kann davon ausgegangen werden, dass eine hohe Wiederverwendung des Codes gegeben ist. Für eine hohe Wiederverwendbarkeit ist es wichtig, dass Einheiten nur wenige Aufgaben haben. Eine geringe Einheitengröße deutet daher auf eine höhere Wiederverwendbarkeit.

Analysierbarkeit

Nach [4] werden Code-Volumen, Code-Duplikation, Einheitengröße und Unit-Testabdeckung verwendet, um die Analysierbarkeit zu quantifizieren. Geringes Code-Volumen und geringe Einheitengröße hilft schnell einen Überblick über den

Quellcode zu bekommen, da weniger Quellcode gelesen werden muss, um die Funktion der Software zu verstehen und die Auswirkungen von Änderungen abzuschätzen. Unit-Tests helfen die Funktionsweise des Quellcodes zu verstehen. Außerdem helfen Unit-Tests dabei Fehler in der Software schneller finden zu können und den Ursprung dieser Fehler zu finden und zu analysieren. Eine hohe Quellcode-Duplikation erschwert hingegen das Verständnis.

Modifizierbarkeit

In [4] wird die Komplexität der Einheiten und die Code-Duplikation als Metrik für die Modifizierbarkeit genannt. Code-Duplikation verringert die Modifizierbarkeit, da an Stellen mit dupliziertem Code, alle Teile des duplizierten Codes gleichermaßen geändert werden müssen. Das erhöht den Modifikations-Aufwand stark und erhöht die Chance, dass bei Modifikationen vergessen wird Teile des duplizierten Codes zu ändern und so Fehler im Programm entstehen. Eine hohe Komplexität der Einheiten sorgt dafür, dass es schwieriger ist diese Einheiten zu modifizieren, ohne Fehler in der Software zu erzeugen.

Testbarkeit

Die Testbarkeit von Software hängt nach [4] unter anderem von der Komplexität der Einheiten, der Einheitengröße und der Zahl der Unit-Testabdeckung ab. Eine hohe Unit-Testabdeckung ist ein Indikator für gute Testbarkeit, denn das bedeutet, dass die Software im allgemeinen testbar ist. Kleine Einheitengröße sorgt auch für eine bessere Testbarkeit, da es einfacher ist für kleine Einheiten, die weniger Aufgaben haben, Testfälle zu spezifizieren. Die Komplexität der Einheiten spielt auch eine Rolle, da für komplexe Einheiten voraussichtlich mehr und komplexere Testfälle spezifiziert werden müssen.

In Tabelle 1 wird die Zugehörigkeit der zuvor vorgestellten Quellcode Kennzahlen zu den Wartbarkeit-Subcharakteristika dargestellt. Diese Tabelle erweitert die in [4] vorgestellte Tabelle um die Änderungen die der neuere ISO 25010 Qualitätsstandard gegenüber dem alten ISO 9126 Standard bringt. Damit kann nun Software verglichen werden und Qualitätsänderungen durch Änderungen an der Software gemessen werden. Im Gegensatz zu [4] wird hier keine konkrete Kennzahl für die Wartbarkeit vorgestellt. Vielmehr können mit dieser Metrik Softwarestände und Softwaresysteme relativ zueinander verglichen werden. Dafür werden die relativen Unterschiede in den einzelnen Metriken eins zu eins nach der Tabelle verrechnet, um einen Wert für die Jeweilige Subcharakterstik der Wartbarkeit zu erhalten. Um daraus einen Wert für die Wartbarkeit zu erhalten, werden die Werte der Subcharakteristika eins zu eins verrechnet.

Um diese Metrik zu verbessern, kann die Verrechnung der einzelnen Quellcode-Kennzahlen gewichtet werden. Diese Entscheidung kann je nach Anwendung der Metrik getroffen werden.

	Quellcodevolumen	Komplexität der Einheiten	Quellcode Duplikation	Einheitengröße	Unit-Testabdeckung	Modulanzahl	Wiederverwendung von Einheiten
Modularität				x		x	
Wiederverwendbarkeit			x	x			x
Analysierbarkeit	x		x	x	x		
Modifizierbarkeit		x	x				
Testbarkeit		x		x	x		

Tabelle 1: Zugehörigkeit der Kennzahlen zu den Wartbarkeit-Subcharakteristika

Es ist denkbar diese Metrik automatisch durch Softwaretools zu generieren. Das ermöglicht es diese Metrik, als Auswertung von Software in ein CI/CD System zu integrieren. Damit können Änderungen an der Software ausgewertet werden und gewarnt werden, wenn die Wartbarkeit durch Änderungen an der Software verschlechtert wird. Im extremsten Fall, kann eine Änderung auch abgelehnt werden, wenn die Wartbarkeit verschlechtert wird.

4 Ausblick

Die hier vorgestellte Metrik zur Wartbarkeit macht es zwar einfach die Wartbarkeit von Softwareprodukten und Softwaresystemen zu messen, allerdings hat diese Metrik auch einige Nachteile. So kann diese Metrik keine direkte Aussage über die Dauer von Wartungsaufgaben an der Software machen. Außerdem wurde nicht verglichen wie sich diese Metrik im Vergleich mit anderen Wartbarkeit-Metriken verhält. Es ist denkbar, dass die hier vorgestellte Metrik völlig andere Werte als andere Wartbarkeit-Metriken liefert. In einer weiteren Arbeit könnte darauf eingegangen werden und die hier vorgestellte Wartbarkeit-Metrik mit andere Wartbarkeit-Metriken verglichen werden, um festzustellen, inwieweit sich diese unterscheiden.

Außerdem bietet die hier vorgestellte Wartbarkeit-Metrik die Möglichkeit Gewichte zu setzen, um die Metrik weiter zu verbessern und anzupassen. In einer weiteren Arbeit könnte diese Metrik durch die Gewichtungen an andere Wartbarkeit-Metriken angepasst werden.

In einer weiteren Arbeit ist es außerdem Denkbar weitere Metriken zu finden, mit denen die übrigen Qualitätscharakteristika einfach quantifiziert werden können.

Literatur

[1] Ben Boyter. *Sloc Cloc and Code (scc)*. https://github.com/boyter/scc. 2022.

[2] BSI. *Die Lage der IT-Sicherheit in Deutschland 2022*. de. Die Lage der IT-Sicherheit in Deutschland BSI-LB22/511. Bonn, Okt. 2022, S. 115. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2022.html.

[3] Statista Research Department. *Beschwerden über Internetkriminalität in den USA 2021*. de. Dez. 2022. URL: https://de.statista.com/statistik/daten/studie/154433/umfrage/beschwerden-ueber-internetkriminalitaet-seit-2000/.

[4] Ilja Heitlager, Tobias Kuipers und Joost Visser. „A Practical Model for Measuring Maintainability". In: *6th International Conference on the Quality of Information and Communications Technology (QUATIC 2007)*. IEEE. 2007, S. 30–39. DOI: 10.1109/QUATIC.2007.8.

[5] „IEEE Standard for Information Technology–System and Software Life Cycle Processes–Reuse Processes". In: *IEEE Std 1517-2010 (Revision of IEEE Std 1517-1999)* (2010), S. 1–51. DOI: 10.1109/IEEESTD.2010.5551093.

[6] ISO 25010. *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models*. Standard. Geneva, CH: International Organization for Standardization, 2011.

[7] ISO 27000. *Information technology — Security techniques — Information security management systems — Overview and vocabulary*. Standard. Geneva, CH: International Organization for Standardization, 2018.

[8] ISO 7498-2. *Information processing systems -Open Systems lnterconnection -Basic Reference Model. Part 2: Security Architecture*. Standard. Geneva, CH: International Organization for Standardization, 1989.

[9] ISO 9126-1. *Information technology — Software product quality. Part 1: Quality model*. Standard. Geneva, CH: International Organization for Standardization, 2000.

[10] „ISO/IEC/IEEE International Standard - Systems and software engineering–Vocabulary". In: *ISO/IEC/IEEE 24765:2017(E)* (2017), S. 1–541. DOI: 10.1109/IEEESTD.2017.8016712.

[11] Muhammad Danish Roshaidie u. a. „Importance of Secure Software Development Processes and Tools for Developers". In: (2020). DOI: 10.48550/ARXIV.2012.15153. URL: https://arxiv.org/abs/2012.15153.

[12] Mohammad Tahaei u. a. „"I Don't Know Too Much About It": On the Security Mindsets of Computer Science Students". en. In: *Socio-Technical Aspects in Security and Trust*. Hrsg. von Thomas Groß und Theo Tryfonas. Bd. 11739. Cham: Springer International Publishing, 2021, S. 27–46. ISBN: 9783030559571 9783030559588. DOI: 10.1007/978-3-030-55958-8_2. URL: https://link.springer.com/10.1007/978-3-030-55958-8_2.