D02-QM Qualitätskriterien: Aufbau, Messgrößen und Bewertung

Editor: Siniša Đukanović (Fraunhofer SIT)

Prüfung: Jens Kubieziel (AGETO)

Typ: [TECHNISCHER BERICHT]

Projekt: „PersoApp“

Version: 1.0

Datum: 19.06.2013

Status: [FREIGABE]

Klasse: [ÖFFENTLICHKEIT]

Datei: D02_QM_Qualitätskriterien-Aufbau, Messgrößen und Bewertung.docx

Zusammenfassung

Dieses Dokument dient dazu die Qualitätskriterien festzulegen und zu beschreiben welche im Entwicklungs- und Releaseprozess der Open-Source-Software PersoApp angewendet werden. Hierzu werden zunächst die einzuhaltenden Qualitätskriterien bestimmt. Für diese Qualitätskriterien werden Messgrößen zur Bewertung der Softwarequalität aufgestellt, sowie geeignete Methoden zur Überprüfung der Kriterien.

Konsortialleitung:
Prof. Dr. Ahmad-Reza Sadeghi und Dr. Sven Wohlgemuth
System Security Lab, TU Darmstadt/CASED, Mornewegstr. 32, 64293 Darmstadt
Tel.: +49-6151-16-75561 Fax: +49-6151-16-72135
E-Mail: persoapp@trust.cased.de Web: https://www.persoapp.de

Nutzungslizenz

Mitglieder des Konsortiums

1. **AGETO Service GmbH**, Deutschland
2. **Center for Advanced Security Research Darmstadt (CASED)**, Deutschland
3. **Fraunhofer Institut für Sichere Informationstechnologie (SIT)**, Deutschland
4. **Technische Universität (TU) Darmstadt**, Deutschland

Versionen

Version	*Datum*	*Beschreibung (Editor)*
0.1	2013-04-26	Erste Version (Siniša Đukanović)
0.2	2013-06-13	Überarbeitete Version für Freigabe (Jens Kubieziel)
1.0	2013-06-17	Reviews von AGETO eingearbeitet (Siniša Đukanović)

Autoren

Autoren	*Beiträge*
Philipp Holzinger (Fraunhofer SIT)	Gesamtes Dokument
Stefan Triller (Fraunhofer SIT)	Gesamtes Dokument

[1] http://creativecommons.org/licenses/by-sa/3.0/

Inhaltsverzeichnis

1 Ziel und Zweck des Dokumentes

Dieses Dokument dient dazu die Qualitätskriterien festzulegen und zu beschreiben welche im Entwicklungs- und Releaseprozess der Open-Source-Software PersoApp angewendet werden. Hierzu werden zunächst die einzuhaltenden Qualitätskriterien bestimmt. Für diese Qualitätskriterien werden Messgrößen zur Bewertung der Softwarequalität aufgestellt, sowie geeignete Methoden zur Überprüfung der Kriterien.

Die Aufstellung und Systematisierung der Qualitätskriterien wird basierend auf der Norm ISO/IEC 9126, dem Internationalen Standard zu ``Software Engineering - Product Quality'' vorgenommen. Unter diese Norm fallen zum Beispiel die folgenden Kriterien: Funktionalität, Zuverlässigkeit und Fehlertoleranz, Effizienz und Wartbarkeit. Um diese Kriterien beurteilen zu können definieren wir Messgrößen, beispielsweise Anzahl der Kommentarzeilen im Verhältnis zum Quellcode um die Wartbarkeit beurteilen zu können, und beschreiben wie die Messgrößen zu bewerten sind.

Für die Reviewprozesse werden die Qualitätskriterien aus diesem Dokument als Grundlage dienen.

2 Abkürzungen und Begriffsdefinitionen

Begriff	*Abkürzung*	*Definition*
Depth of Inheritance Tree	DIT	Absolute Anzahl der Stufen in der Vererbungshierarchie
Technische Richtlinien	TR	Das Bundesamt für Sicherheit in der Informationstechnik spezifiziert diverse Technische Richtlinien, die für die Umsetzung der PersoApp verpflichtend zu berücksichtigen sind.[2]

3 Zuständigkeiten und Verantwortlichkeiten

Das Fraunhofer SIT als Qualitäts- und Sicherheitsverantwortlicher in diesem Projekt stellt in diesem Dokument Qualitätskriterien für die Entwicklung des Open-Source-Projekts PersoApp auf. Überprüft werden sie vom Kernentwicklerteam während der Reviewprozesse. Die anderen Projektteilnehmer haben an diesem Dokument keine

[2] Siehe https://www.bsi.bund.de/DE/Themen/ElektronischeAusweise/TechnRichtlinien/trundschutzprofile_node.html

Anteile, wurden aber in Form eines Reviews dieses Dokuments mit der Möglichkeit für Kommentare und Änderungswünsche, indirekt daran beteiligt.

4 Qualitätskriterien

Ein ausgewiesenes Ziel des Projekts ist der Aufbau einer Open-Source-Community, die die Entwicklung der PersoApp langfristig vorantreibt. Um über den gesamten Projektverlauf eine hohe Softwarequalität gewährleisten zu können, werden basierend auf ISO/IEC 9126 nachfolgend zunächst die Qualitätskriterien festgelegt, anhand derer eine entsprechende Bewertung vorgenommen werden kann.

Funktionalität

Zu den wesentlichen Aspekten bei der Entwicklung der Software zählen die Plattformunabhängigkeit und die Interoperabilität. Durch die Plattformunabhängigkeit wird ein breites Spektrum unterschiedlicher Benutzer und Entwickler angesprochen, die nicht an ein bestimmtes System gebunden sind. Die Interoperabilität nimmt eine zentrale Rolle ein, da die Software mit einer Vielzahl unterschiedlicher Fremdsysteme, wie z.B. eID-Server diverser Hersteller, interagieren muss. Daher gilt es einerseits sicherzustellen, dass die Software auf allen Zielsystemen den vollständigen Funktionsumfang zur Verfügung stellt. Zum anderen muss gewährleistet werden, dass die Interaktion mit Fremdsystemen zuverlässig und einheitlich erfolgt.

Zuverlässigkeit

Eine Software, die für den Produktiveinsatz entwickelt wird, muss einen zuverlässigen Systembetrieb erlauben. Dies umfasst, dass die Anzahl der auftretenden Fehler im Rahmen der Benutzung möglichst niedrig ist. Erst hierdurch wird der Einsatz der Software planbar und praktikabel.

Sicherheit

Die Software verarbeitet sensible Daten, wie z.B. Name und Geburtsdatum eines Bürgers, und ist aufgrund dessen ein potentielles Angriffsziel unterschiedlichster Akteure. Ihre allgemeine Akzeptanz hängt in hohem Maße davon ab, dass zu jedem Zeitpunkt ein akzeptables Sicherheitsniveau gewährleistet werden kann. Um ein akzeptables Sicherheitsniveau zu erreichen, ist es nötig bereits früh im Entwicklungsprozess einer Software Sicherheitsaspekte zu berücksichtigen. Sicherheit als Qualitätsmerkmal kann nur durch Transparenz und gute Dokumentationen über getroffene sicherheitskritische Entscheidungen erreicht werden. Sicherheitsrelevante Dokumentationen, wie etwa eine Bedrohungsanalyse oder die Beschreibung der Sicherheitsarchitektur lassen sich nur schwer auf Metriken abbilden, sind aber Bestandteil des Projektes und werden im noch folgenden Dokument „D03-QM Entwurfs- und Entwicklungsprozess von sicheren Open Source Softwaremodulen der AusweisApp“ beschrieben.

Modularität

Die Weiterentwicklung der Software erfolgt durch eine Open-Source-Community. Um einer Vielzahl von Entwicklern die Arbeit an einem komplexen Softwareprodukt zu ermöglichen, muss es modular aufgebaut sein. Einzelne Teilkomponenten kommunizieren über wohldefinierte Schnittstellen und können parallel entwickelt und getestet werden. Darüber hinaus bietet ein modulares System die größtmögliche Flexibilität um Teilkomponenten austauschen und ändern zu können.

Übertragbarkeit und Wiederverwendbarkeit

Zur Vermeidung von Mehrfachaufwänden sollen möglichst Konzepte und Lösungen erarbeitet werden, die so generisch sind, dass sie wiederverwendet werden können. Letztlich kann hierdurch der Gesamtaufwand für Test und Entwicklung gesenkt werden.

Effizienz und Performance

Die Software soll möglichst ökonomisch mit verfügbaren Hardware-Ressourcen umgehen. Im Zuge der Erstellungsphase gilt es akzeptable Vorgaben für die Performance zu definieren, deren Einhaltung über die Projektlaufzeit überprüft werden muss.

Wartbarkeit

Die Komponenten, die im Laufe des Projekts weiterentwickelt werden, müssen langfristig an sich ändernde Anforderungen angepasst werden können. Darüber hinaus sollen Funktionen hinzugefügt und Fehlerkorrekturen effizient durchgeführt werden können. Dieser kontinuierliche Wartungsprozess wird von einer Vielzahl unterschiedlicher Entwickler getragen und stellt insbesondere daher Anforderungen an eine umfangreiche Dokumentation und die Lesbarkeit von Quelltext.

Projektzugänglichkeit *

**Vgl. „Barrierefreiheit“ im Angebotsschreiben*

Ein erklärtes Projektziel ist der Aufbau einer aktiven Community, die die Weiterentwicklung der Software nachhaltig vorantreibt. Um dies erreichen zu können, müssen die Einstiegshürden für interessierte Entwickler und Organisationen möglichst niedrig gehalten werden.

Verständlichkeit

Die einzelnen Module der Software sollen so strukturiert und dokumentiert sein, dass sie für andere Entwickler verständlich und nachvollziehbar sind. Dies umfasst unter anderem die Einhaltung einer noch auszuarbeitenden Programmierrichtlinie in „D04-QM Programmierrichtlinien zur Erstellung von AusweisApp Softwaremodulen“.

Benutzerfreundlichkeit

Entwickler, Benutzer und interessierte Besucher sollen durch benutzerfreundliche Systeme an die Software herangeführt und bei der Projektdurchführung unterstützt werden. Bei diesen Systemen kann es sich um gängige Lösungen für die Code- und Dokumentenverwaltung handeln, beispielsweise aber auch um ein Wiki, oder eine Projektwebseite.

Integrationsfähigkeit

Die im Projektverlauf weiterentwickelte Software, oder einzelne Komponenten hiervon, sollen grundsätzlich in Software anderer Hersteller integriert werden können. Eine solide Grundlage dafür ist nur gegeben, wenn eine Vielzahl der oben aufgeführten Qualitätskriterien erfüllt wird. Insbesondere betrifft dies die ordnungsgemäße und zuverlässige Funktionalität, ein modularer Aufbau, die Wiederverwendbarkeit und Wartbarkeit des Quelltextes sowie die Verständlichkeit. Um dies zu erreichen ist es nötig gute Dokumentationen, u.a. zum Quelltext, der (Sicherheits-)architektur und den externen Schnittstellen, zu erstellen. Dadurch wird die Arbeit der Entwickler anderer Hersteller deutlich erleichtert.

5 Messgrößen und Bewertungsmaßstäbe

Nachdem in Kapitel 4 die Qualitätskriterien festgelegt wurden, anhand derer die Softwarequalität der PersoApp bemessen werden soll, widmet sich dieses Kapitel der Auswahl geeigneter Messgrößen, um die festgelegten Kriterien quantifizieren zu können. Darüber hinaus werden Bewertungsmaßstäbe definiert, die für die Auswertung der Messgrößen erforderlich sind.

In der nachfolgenden Auflistung wird für jedes beschriebene Qualitätskriterium einzeln festgelegt, welche Messgrößen erfasst werden müssen und in welcher Weise diese zu bewerten sind. Einige Messgrößen sind Indikatoren für mehrere Qualitätskriterien und werden daher ggf. mehr als einmal aufgeführt.

QS1: Funktionalität

Messgröße: Kompatibilität zu gängigen Windows-Plattformen

Bewertung: Um eine breite Nutzerbasis ansprechen zu können, ist es notwendig, dass die Software auf allen gängigen Windows-Plattformen uneingeschränkt lauffähig ist. Hierfür werden manuelle Tests durchgeführt, um etwaiges Fehlverhalten aufzudecken. Dabei gilt jede Unzulänglichkeit als Qualitätsmangel.

Messgröße: Anzahl der bekannten Softwarefehler

Bewertung: Jeder bekannte Softwarefehler wird in einem Issue Tracker als “Defect” erfasst und priorisiert. Gezählt werden all jene Einträge, deren Bearbeitung noch nicht abgeschlossen ist. Eine steigende Anzahl von “Defects” stellt einen Qualitätsverlust der Software dar und impliziert Handlungsbedarf. Dies gilt insbesondere dann, wenn ein erheblicher Anteil der Issues hoch priorisiert wurde (d.h. starke Einschränkung der Funktionalität). Insgesamt wird eine möglichst geringe Anzahl unbearbeiteter “Defects” angestrebt.

Messgröße: Anzahl der Abweichungen von den Technischen Richtlinien

Bewertung: Die vollständige Einhaltung der Technischen Richtlinien ist zwingend erforderlich um die Funktionalität, Interoperabilität und Sicherheit der Anwendung gewährleisten zu können. Daher gilt jede Abweichung als inakzeptabel.

Messgröße: Testabdeckung

Bewertung: Durch automatisierte Softwaretests kann die erwartungsgemäße Funktionsweise von Softwarekomponenten geprüft werden. Die Testabdeckung in Prozent kann einen Eindruck davon vermitteln, wie umfangreich getestet wurde. Hierdurch können Komponenten ermittelt werden, die gar nicht, oder nur unzureichend getestet wurden. Insgesamt sollte eine zeilenweise Testabdeckung des Quelltextes von 15 % nicht unterschritten werden.

QS2: Zuverlässigkeit

Messgröße: Anzahl neuer “Defects” pro Monat

Bewertung: Jeder bekannte Softwarefehler wird in einem Issue Tracker als “Defect” erfasst und priorisiert. Gezählt werden all jene Einträge, die im Laufe eines Monats erstellt wurden. Eine stetig steigende Anzahl neuer Softwarefehler kann auf einen Qualitätsverlust hindeuten. Insgesamt wird eine Senkung der Fehlerfunde bei gleichbleibendem oder steigendem Testaufwand über den Projektverlauf angestrebt.

Messgröße: Testabdeckung

Bewertung: Durch automatisierte Softwaretests kann die erwartungsgemäße Funktionsweise von Softwarekomponenten geprüft werden. Die Testabdeckung in Prozent kann einen Eindruck davon vermitteln, wie umfangreich getestet wurde. Hierdurch können Komponenten ermittelt werden, die gar nicht, oder nur unzureichend getestet wurden. Insgesamt sollte eine zeilenweise Testabdeckung des Quelltextes von 15 % nicht unterschritten werden.

QS3: Sicherheit

Messgröße: Anzahl durchgeführter Reviews pro Klasse

Bewertung: Durch ein manuelles Review kann für eine Komponente detailliert festgestellt werden, welche Qualitätsanforderungen sie erfüllt. Für die Bewertung einer Komponente, die noch keinem Review unterzogen wurde, können lediglich automatisch auswertbare Messgrößen herangezogen werden. Im Zuge des Projektverlaufs gilt es die Kernkomponenten mindestens einem Review zu unterziehen.

Messgröße: Dauer bis zum Fix eines hoch priorisierten Softwarefehlers

Bewertung: Jeder bekannte Softwarefehler wird in einem Issue Tracker als “Defect” erfasst und priorisiert. Gemessen wird der Zeitraum zwischen dem Eintrag des Softwarefehlers in den Issue Tracker und seiner vollständigen Abarbeitung. Hierbei berücksichtigt werden nur solche Softwarefehler, denen die höchste Priorität beigemessen wurde. Es wird eine möglichst kurze Abarbeitungszeit angestrebt. Eine steigende Fixdauer hingegen wird als Qualitätsverlust gewertet.

Messgröße: Anzahl der bekannten Softwarefehler

Bewertung: Jeder bekannte Softwarefehler wird in einem Issue Tracker als “Defect” erfasst und priorisiert. Gezählt werden all jene Einträge, deren Bearbeitung noch nicht abgeschlossen ist. Eine steigende Anzahl von “Defects” stellt einen Qualitätsverlust der Software dar und impliziert Handlungsbedarf. Dies gilt insbesondere dann, wenn ein erheblicher Anteil der Issues hoch priorisiert wurde (d.h. starke Einschränkung der Funktionalität). Insgesamt wird eine möglichst geringe Anzahl unbearbeiteter “Defects” angestrebt.

Messgröße: Anzahl der “Defects” pro Komponente

Bewertung: Jeder bekannte Softwarefehler wird in einem Issue Tracker als “Defect” erfasst. Gezählt werden alle Einträge pro Komponente, unabhängig von Priorität und Status. Eine Komponente, die eine signifikant höhere Anzahl von Softwarefehlern aufweist als der Durchschnitt über alle Komponenten, kann auf Qualitätsmängel im

Zuge der Entwicklung hindeuten. Als signifikant erhöht gilt die Anzahl der Softwarefehler, wenn der Durchschnittswert über alle Komponenten um das Doppelte oder mehr überschritten wird.

Messgröße: Anzahl der Abweichungen von den Technischen Richtlinien

Bewertung: Die vollständige Einhaltung der Technischen Richtlinien ist zwingend erforderlich um die Funktionalität, Interoperabilität und Sicherheit der Anwendung gewährleisten zu können. Daher gilt jede Abweichung als inakzeptabel.

Messgröße: Testabdeckung

Bewertung: Durch automatisierte Softwaretests kann die erwartungsgemäße Funktionsweise von Softwarekomponenten geprüft werden. Die Testabdeckung in Prozent kann einen Eindruck davon vermitteln, wie umfangreich getestet wurde. Hierdurch können Komponenten ermittelt werden, die gar nicht, oder nur unzureichend getestet wurden. Insgesamt sollte eine zeilenweise Testabdeckung des Quelltextes von 15% nicht unterschritten werden.

QS4: Modularität

Messgröße: Anzahl Codezeilen pro Methode

Bewertung: Eine sehr umfangreiche Methode ist potentiell schlechter lesbar und legt nahe, dass eine sinnvolle Aufteilung in mehrere Methoden möglich ist. Die Größe einer Methode sollte daher 50 Zeilen nicht überschreiten. Gezählt werden alle physikalischen Zeilen, die mindestens ein Zeichen enthalten, das weder Kommentar noch Leerraum (nicht-druckbare Zeichen) ist.

Messgröße: Anzahl Methoden pro Klasse

Bewertung: In einem qualitativ hochwertigen Design ist gewährleistet, dass jede Klasse eine wohldefinierte Einheit bildet und sie der Erfüllung einer konkreten Aufgabe gewidmet ist. Eine Untersuchung von über 2000 Open-Source-Java-Applikationen zeigte, dass die Klassen durchschnittlich jeweils 3,5 Methoden umfassten [2]. Für die Entwicklung der PersoApp könnte daher eine Klasse, die signifikant mehr Methoden umfasst, auf einen strukturellen Qualitätsmangel hinweisen. Von der Zählung ausgeschlossen werden reine Getter/Setter-Methoden.

QS5: Übertragbarkeit und Wiederverwendbarkeit

Messgröße: Anzahl der eingehenden Abhängigkeiten pro Klasse

Bewertung: Gezählt werden alle Klassen und Interfaces, die die zu bemessende Klasse verwenden. Unter Verwendung wird die Implementierung, die Erbung, der Methodenaufruf, die Referenzierung und der Zugriff auf Variablen verstanden. Je

häufiger eine Klasse verwendet wird, umso mehr spricht dies für ihre Wiederverwendbarkeit.

Messgröße: Anzahl der ausgehenden Abhängigkeiten pro Klasse

Bewertung: Gezählt werden alle Klassen, Interfaces, etc., die von der zu bemessenden Klasse verwendet werden. Unter Verwendung wird die Implementierung, die Erbung, der Methodenaufruf, die Referenzierung und der Zugriff auf Variablen verstanden. Je höher die Anzahl der ausgehenden Abhängigkeiten ist, umso größer könnte der Aufwand für die Wiederverwendung und Wartung sein.

Messgröße: Anzahl der Kinder pro Klasse

Bewertung: Für eine Klasse wird die Anzahl der Klassen gezählt, die direkt oder indirekt von ihr erben. Je höher die Anzahl der Kinder ist, umso mehr spricht dies für ihre Wiederverwendbarkeit.

QS6: Effizienz und Performance

Messgröße: Depth of Inheritance Tree (DIT) pro Klasse/Interface

Bewertung: Für eine Klasse/Interface wird die Anzahl der Vererbungsstufen bis zur Erreichung des Wurzelknotens gezählt. Der Aufwand für die Instantiierung und Verwaltung von Objekten könnte mit steigender Vererbungstiefe zunehmen, daher sollten hochkomplexe Vererbungsstrukturen vermieden werden. Eine Untersuchung von über 2000 Open-Source-Java-Applikationen zeigte, dass die maximale Vererbungstiefe bei 5 Stufen (respektive 6 inkl. java.lang.Object) lag [2]. Für die Entwicklung der PersoApp sollte daher dieser Wert nicht signifikant überschritten werden.

Messgröße: Maximal benötigter Arbeitsspeicher in Megabyte

Bewertung: Für einen oder mehrere wohldefinierte Anwendungsfälle wird der maximal benötigte Arbeitsspeicher in Megabyte auf einer Referenzmaschine ermittelt. Hierbei gilt ein niedrigerer Speicherbedarf als effizienter.

Messgröße: Durchschnittliche CPU-Auslastung in Prozent

Bewertung: Für einen oder mehrere wohldefinierte Anwendungsfälle wird die durchschnittlich verursachte CPU-Auslastung in Prozent auf einer Referenzmaschine ermittelt. Hierbei gilt eine niedrigere Auslastung als effizienter.

QS7: Wartbarkeit

Messgröße: Anteil der undokumentierten Schnittstellen

Bewertung: Es werden alle als “public” deklarierten Klassen, Interfaces, Methoden und Konstruktoren gezählt, die nicht mit einem Javadoc-Kommentar versehen sind. Von der Zählung ausgeschlossen sind Getter/Setter-Methoden, leere Konstruktoren sowie überschreibende Methoden, die durch “@Override” annotiert sind. Es wird angestrebt, dass der Anteil der undokumentierten Schnittstellen bei 0 % liegt.

Messgröße: Anzahl der ausgehenden Abhängigkeiten pro Klasse

Bewertung: Gezählt werden alle Klassen, Interfaces, etc., die von der zu bemessenden Klasse verwendet werden. Unter Verwendung wird die Implementierung, die Erbung, der Methodenaufruf, die Referenzierung und der Zugriff auf Variablen verstanden. Je höher die Anzahl der ausgehenden Abhängigkeiten ist, umso größer könnte der Aufwand für die Wiederverwendung und Wartung sein.

Messgröße: Anzahl Kommentarzeilen im Verhältnis zu Codezeilen pro Klasse

Bewertung: Durch Kommentarzeilen soll die Verständlichkeit und Lesbarkeit des Codes verbessert werden. Wenn diese nur unzureichend vorhanden sind, entspricht dies einem Qualitätsmangel, der sich besonders im Zuge der Einarbeitung und Wartung auswirkt. Eine Untersuchung von mehr als 5000 Open-Source-Projekten zeigte, dass der durchschnittliche Anteil der Kommentarzeilen am Quelltext bei 19% liegt [1]. Daher gilt für die Entwicklung der PersoApp ein deutlich darunter liegender Anteil als Qualitätsmangel. Als Codezeile wird hier jede physikalische Zeile gezählt, die mindestens ein Zeichen enthält, das weder Kommentar noch Lerraum (nicht-druckbare Zeichen) ist.

Messgröße: Anzahl der Abweichungen von der Programmierrichtlinie pro Klasse/

Interface

Bewertung: Jede Abweichung von der Programmierrichtlinie (auszuarbeiten in „D04-QM Programmierrichtlinien zur Erstellung von AusweisApp Softwaremodulen“) gilt als Qualitätsmangel. Als inakzeptabel gelten mehr als fünf Abweichungen pro Klasse.

Messgröße: Zyklomatische Komplexität pro Methode

Bewertung: Die zyklomatische Komplexität nach McCabe [3] erlaubt eine Einschätzung, ob der Kontrollfluss einer Methode als unübersichtlich gilt. Die Berechnung des Komplexitätswertes beruht auf der Anzahl der Verzweigungen, die den Kontrollfluss der Methode steuern, wie z.B. If-Abfragen, bedingte Schleifen, etc. Die Festlegung eines oberen Limits für die Komplexität einer Methode im Rahmen

eines Softwareentwicklungsprojekts wird empfohlen, daher sollte für die Entwicklung der PersoApp ein Wert von 10 nicht überschritten werden [4, Seite 15 f.].

QS8: Projektzugänglichkeit*

*Vgl. „Barrierefreiheit“ im Angebotsschreiben

Messgröße: Anzahl aktiver Entwickler

Bewertung: Eine hohe Anzahl aktiver Entwickler impliziert, dass der Zugang zu dem Projekt weitgehend barrierefrei ist. Ein aktiver Entwickler zeichnet sich dadurch aus, dass er in regelmäßigen Abständen Commits beiträgt. Für das Projekt wird im Zuge des Community-Buildings eine steigende Anzahl von Entwicklern angestrebt.

Messgröße: Anzahl Kommentarzeilen im Verhältnis zu Codezeilen pro Klasse

Bewertung: Durch Kommentarzeilen soll die Verständlichkeit und Lesbarkeit des Codes verbessert werden. Wenn diese nur unzureichend vorhanden sind, entspricht dies einem Qualitätsmangel, der sich besonders im Zuge der Einarbeitung und Wartung auswirkt. Eine Untersuchung von mehr als 5000 Open-Source-Projekten zeigte, dass der durchschnittliche Anteil der Kommentarzeilen am Quelltext bei 19 % liegt [1]. Daher gilt für die Entwicklung der PersoApp ein deutlich darunter liegender Anteil als Qualitätsmangel. Als Codezeile wird hier jede physikalische Zeile gezählt, die mindestens ein Zeichen enthält, das weder Kommentar noch Leeraum (nicht-druckbare Zeichen) ist.

Messgröße: Anzahl der Abweichungen von der Programmierrichtlinie pro Klasse/

Interface

Bewertung: Jede Abweichung von der Programmierrichtlinie (auszuarbeiten in „D04-QM Programmierrichtlinien zur Erstellung von AusweisApp Softwaremodulen“) gilt als Qualitätsmangel. Als inakzeptabel gelten mehr als fünf Abweichungen pro Klasse.

QS9: Verständlichkeit

Messgröße: Anteil der undokumentierten Schnittstellen

Bewertung: Es werden alle als “public” deklarierten Klassen, Interfaces, Methoden und Konstruktoren gezählt, die nicht mit einem Javadoc-Kommentar versehen sind. Von der Zählung ausgeschlossen sind Getter/Setter-Methoden, leere Konstruktoren sowie überschreibende Methoden, die durch “@Override” annotiert sind. Es wird angestrebt, dass der Anteil der undokumentierten Schnittstellen bei 0 % liegt.

Messgröße: Anzahl Kommentarzeilen im Verhältnis zu Codezeilen pro Klasse

Bewertung: Durch Kommentarzeilen soll die Verständlichkeit und Lesbarkeit des Codes verbessert werden. Wenn diese nur unzureichend vorhanden sind, entspricht dies einem Qualitätsmangel, der sich besonders im Zuge der Einarbeitung und Wartung auswirkt. Eine Untersuchung von mehr als 5000 Open-Source-Projekten zeigte, dass der durchschnittliche Anteil der Kommentarzeilen am Quelltext bei 19 % liegt [1]. Daher gilt für die Entwicklung der PersoApp ein deutlich darunter liegender Anteil als Qualitätsmangel. Als Codezeile wird hier jede physikalische Zeile gezählt, die mindestens ein Zeichen enthält, das weder Kommentar noch Leerraum (nicht-druckbare Zeichen) ist.

Messgröße: Anzahl der Abweichungen von der Programmierrichtlinie pro Klasse/

Interface

Bewertung: Jede Abweichung von der Programmierrichtlinie (auszuarbeiten in „D04-QM Programmierrichtlinien zur Erstellung von AusweisApp Softwaremodulen“) gilt als Qualitätsmangel. Als inakzeptabel gelten mehr als fünf Abweichungen pro Klasse.

Messgröße: Zyklomatische Komplexität pro Methode

Bewertung: Die zyklomatische Komplexität nach McCabe [3] erlaubt eine Einschätzung, ob der Kontrollfluss einer Methode als unübersichtlich gilt. Die Berechnung des Komplexitätswertes beruht auf der Anzahl der Verzweigungen, die den Kontrollfluss der Methode steuern, wie z.B. If-Abfragen, bedingte Schleifen, etc. Die Festlegung eines oberen Limits für die Komplexität einer Methode im Rahmen eines Softwareentwicklungsprojekts wird empfohlen, daher sollte für die Entwicklung der PersoApp ein Wert von 10 nicht überschritten werden [4, Seite 15 f.].

QS10: Benutzerfreundlichkeit

Messgröße: Anzahl Downloads pro Monat

Bewertung: Eine zunehmende Anzahl von Downloads pro Monat spricht dafür, dass das System ausreichend benutzerfreundlich ist, um interessierten Besuchern einen Einstieg zu ermöglichen.

Messgröße: Anzahl aktiver Entwickler

Bewertung: Eine hohe Anzahl aktiver Entwickler impliziert, dass der Zugang zu dem Projekt weitgehend barrierefrei ist. Ein aktiver Entwickler zeichnet sich dadurch aus, dass er in regelmäßigen Abständen Commits beiträgt. Für das Projekt wird im Zuge des Community-Buildings eine steigende Anzahl von Entwicklern angestrebt.

Messgröße: Anzahl der Besucher auf der Projektwebseite pro Monat

Bewertung: Eine steigende Anzahl an Webseitenbesucher spricht für ein benutzerfreundliches System, das die Bedürfnisse der Interessenten widerspiegelt.

Notiz: Aufwendige Usability-Tests stehen nicht im Fokus des Projekts und können aufgrund der dafür notwendigerweise aufzubringenden Ressourcen nicht durchgeführt werden.

QS11: Integrationsfähigkeit

Messgröße: Anteil der undokumentierten Schnittstellen

Bewertung: Es werden alle als “public” deklarierten Klassen, Interfaces, Methoden und Konstruktoren gezählt, die nicht mit einem Javadoc-Kommentar versehen sind. Von der Zählung ausgeschlossen sind Getter/Setter-Methoden, leere Konstruktoren sowie überschreibende Methoden, die durch “@Override” annotiert sind. Es wird angestrebt, dass der Anteil der undokumentierten Schnittstellen bei 0 % liegt.

Messgröße: Anzahl der Produkte/Dienstleistungen, die die Software integrieren

Bewertung: Eine steigende Anzahl von Produkten und Dienstleistungen von Partnern und Drittherstellern, die die Software integrieren, legen nahe, dass der Integrationsfähigkeit ausreichend Rechnung getragen wurde.

Messgröße: Anzahl der Abweichungen von den Technischen Richtlinien

Bewertung: Die vollständige Einhaltung der Technischen Richtlinien ist zwingend erforderlich um die Funktionalität, Interoperabilität und Sicherheit der Anwendung gewährleisten zu können. Daher gilt jede Abweichung als inakzeptabel.

6 Abläufe

Die in diesem Dokument erwähnten Qualitätskriterien dienen als Grundlage für die Reviewprozesse vor einem Release einer neuen Version der Software. Die Überprüfung der Qualitätskriterien wird weitestgehend automatisiert ablaufen. Hierzu wird der Quellcode, bzw. dessen Änderungen zur Vorversion, mit Hilfe einer Software eingelesen und ausgewertet. Verschiedene Qualitätskriterien können verschiedene Auswertungssoftware erfordern. Welche Auswertungssoftware konkret eingesetzt wird, kann den noch folgenden Dokumenten "D08-QM Review-Konzept „AusweisApp“", "D08-QM-2 Operative Planung und Durchführung von Reviews und Release-Updates" und "D08-QM-3 Prozessbeschreibung zur Durchführung von Code-Reviews und Sicherstellung der Dokumentationsqualität" entnommen werden. Darin wird der Reviewprozess spezifiziert werden. Das Ergebnis der Auswertungssoftware wird mit den Entwicklern diskutiert und ggf. werden die Entwickler aufgefordert nachzubessern. Weiterhin werden die Ergebnisse der Auswertungssoftware im Wiki der Projekthomepage dokumentiert.

7 Mitgeltende Dokumente

Die folgenden Dokumente sind für die Erhebung und Bewertung der Messgrößen zu beachten:

- „D04-QM Programmierrichtlinien zur Erstellung von PersoApp Softwaremodulen „
- „BSI TR-03112 „Das eCard-API-Framework”
- „BSI TR-03127 Architektur Elektronischer Personalausweis“
- „BSI TR-03128 EAC-PKI'n für den elektronischen Personalausweis“
- „BSI TR-03130 eID-Server“

Literatur

[1] Arafat, Oliver und Riehle, Dirk. *The comment density of open source software code*. In: Companion to the Proceedings of the 31st International Conference on Software Engineering (ICSE 2009), S. 195–198, 2009.

[2] Grechanik, Mark, McMillan, Collin, DeFerrari, Luca, Comi, Marco, Crespi, Stefano, Poshyvanyk, Denys, Fu, Chen, Xiw, Qing und Ghezzi, Carlo. *An empirical investigation into a large-scale Java open source code repository.* In: Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2010), Artikel 11, 10, 2010.

[3] McCabe, Thomas J. *A complexity measure*. In: Proceedings of the 2nd international conference on Software engineering (ICSE 1976), 1976.

[4] Watson, Arthur H., McCabe, Thomas J. und R Wallace, Dolores. *Structured testing: A testing methodology using the cyclomatic complexity metric*. NIST special Publication, 500(235), S. 1–114, 1996.