ICS 35.080 VDI-RICHTLINIEN September 2019

VEREIN DEUTSCHER INGENIEURE

Softwarezuverlässigkeit

VDI 4011

Entwurf

Software reliability

Einsprüche bis 2019-12-31

- *vorzugsweise über das VDI-Richtlinien-Einspruchsportal http://www.vdi.de/4011*
- *in Papierform an*
  VDI-Gesellschaft Produkt- und Prozessgestaltung
  Fachbereich Sicherheit und Zuverlässigkeit
  Postfach 10 11 39
  40002 Düsseldorf

Inhalt Seite

Vorbemerkung ... 2

Einleitung ... 2

1 Anwendungsbereich ... 3

2 Begriffe ... 3

3 Abkürzungen ... 4

4 Methoden und Kriterien zur Entwicklung zuverlässiger Software ... 4

4.1 Lebenszyklusmodell ... 5

4.2 Prozess- und Vorgehensmodelle ... 5

4.3 Softwarespezifikation ... 8

4.4 Entwicklungsmethodik ... 10

4.5 Wiederverwendung ... 11

5 Methoden und Kriterien zur Bewertung von Softwarezuverlässigkeit ... 12

5.1 Probabilistische Verfahren ... 12

5.2 Nutzungsprofilunabhängige Verfahren ... 16

6 Methoden und Kriterien zur Bewahrung der Softwarezuverlässigkeit ... 18

6.1 Tätigkeiten der Modifikation ... 19

6.2 Tätigkeiten der Rekonfiguration ... 22

6.3 Tätigkeiten der Parametrierung ... 22

6.4 Konfigurationsmanagement ... 22

Anhang ... 23

A1 Beschreibung der Entwicklungsphasen ... 23

A2 Einflussfaktoren der Softwarezuverlässigkeit ... 24

A3 Schritte zur Ermittlung der funktionalen Anforderungen ... 24

A4 Schritte für die Softwarewartbarkeit ... 25

Schrifttum ... 26

VDI-Gesellschaft Produkt- und Prozessgestaltung (GPP)

Fachbereich Sicherheit und Zuverlässigkeit

VDI-Handbuch Zuverlässigkeit

Vorbemerkung

Der Inhalt dieser Richtlinie ist entstanden unter Beachtung der Vorgaben und Empfehlungen der Richtlinie VDI 1000.

An der Erarbeitung dieser Richtlinie waren beteiligt:

Prof. Dr. *Yongjian Ding*, Magdeburg-Stendahl

Dipl.-Inf. *Thomas Eisenbarth*, Stuttgart

Dr. *Mario Hellmich*, Salzgitter

Dr.-Ing. *Jazdi Nasser*, Stuttgart

Dr.-Ing. *Hubert Keller*, Karlsruhe

Dr.-Ing. *Camelia Maga*, Ludwigsburg

Dipl.-Ing. *Horst Miedl*, Hallbergmoos

Priv.-Doz. Dr. *Jörg R. Müller*, Koblenz

Dipl.-Ing *Tobias Nelke*, Hamburg

Prof. Dr. *Francesca Saglietti*, Erlangen-Nürnberg

Dr. *Freddy Seidel*, Zeuthen

Allen, die ehrenamtlich an der Erarbeitung dieser Richtlinie mitgewirkt haben, sei gedankt.

Eine Liste der aktuell verfügbaren Blätter dieser Richtlinienreihe ist im Internet abrufbar unter www.vdi.de/4011.

Einleitung

Es kann davon ausgegangen werden, dass jede hinreichend komplexe Software eine unbekannte Anzahl von Fehlern beinhaltet. Es existieren zwar Verfahren zur Schätzung der Anzahl dieser Fehler aus Erfahrungswerten, allerdings kann sie nicht mit Sicherheit bestimmt werden. Ebenso können nicht mit Sicherheit alle Fehler in einer Software lokalisiert werden. Um Zuverlässigkeitsanforderungen an Software gewährleisten zu können, ist daher in den meisten Fällen eine Kombination verschiedener Maßnahmen und Methoden notwendig. Diese Richtlinie gibt einen Überblick über entsprechende Methoden und Kriterien zur Entwicklung zuverlässiger Software, zur Bewertung der Softwarezuverlässigkeit sowie zur Bewahrung der Softwarezuverlässigkeit bei Modifikationen.

Das Verhalten einer Software bei der Ausführung, und damit auch die Art und Häufigkeit, mit der sich latent vorhandene Fehler manifestieren, ist abhängig von den Betriebsbedingungen (Nutzerverhalten, Eingabeparameter, Umgebung wie das Betriebssystem, Speicherverlauf, parallel laufende Anwendungen etc.). Diese hängen im Detail von Zufallsgrößen (Daten, Anforderungen, Benutzereingaben und deren zeitliche Abfolge etc.) ab. Daher wird Softwarezuverlässigkeit im Rahmen dieser Richtlinie aufgefasst als die Wahrscheinlichkeit, dass Software unter festgelegten Betriebsbedingungen und innerhalb eines festgelegten Zeitintervalls ihre geforderte Funktion erfüllt. Eine Nichterfüllung der geforderten Funktion wird als ein Softwareversagen bezeichnet.

Die in einer Software vorhandenen Fehler können u.a. logische Fehler hinsichtlich der Anforderungen, Spezifikationsfehler (das heißt, Anforderungen wurden falsch beschrieben), systematische Fehler im Design oder der Implementierung, Entwurfsfehler in der Lösung, Implementierungsfehler im Einzelnen, Dokumentationsfehler oder auch Fehler im Laufzeitsystem sein. Hierauf wird im Folgenden im Detail eingegangen.

Eine Zuverlässigkeitsanalyse von Software ist schwierig, da sich zum einen Software, im Gegensatz zu physikalischen bzw. Hardwaresystemen, diskontinuierlich verhält und zum anderen in vielen Fällen keine genauen, auf empirischen Untersuchungen beruhenden Daten und Aussagen über Zuverlässigkeitskenngrößen, z.B. die Ausfallrate λ, von Software erhoben werden können. Hierbei spielt u.a. die Schwierigkeit, den Betriebsbedingungen entsprechende Testfälle unabhängig voneinander generieren zu können, eine Rolle.

Eine hohe Zuverlässigkeit von Softwaresystemen wird durch die Anwendung und das Zusammenspiel verschiedener Maßnahmen erreicht. Diese sind:

a) Organisatorische Maßnahmen (wie Normen, Vorgehensmodelle etc.):
   Organisatorische Maßnahmen zur Erhöhung der Softwarezuverlässigkeit haben das Ziel, definierte und kontrollierbare Prozesse zu gestalten
b) Konstruktive Maßnahmen (wie Modularisierung, Wiederverwendung, Verwendung von Beschreibungsverfahren wie UML und entsprechenden Werkzeugen sowie fehlervermeidende Programmiersprachen und zugehörige Laufzeitsysteme etc.):
   Konstruktive Maßnahmen zur Erhöhung der Softwarezuverlässigkeit haben das Ziel, eine strukturierte Vorgehensweise in der Entwicklung zu ermöglichen.
c) Analytische Maßnahmen (wie Softwareprüfung, Code-Review etc.):
   Analytische Maßnahmen zur Erhöhung der Softwarezuverlässigkeit haben das Ziel, mög-

lichst viele in der Software enthaltenen Fehler und Mängel vor Inbetriebnahme zu entdecken.

Die Betrachtung der Fehlerratenentwicklung (Reifegrad) im Betrieb allein erlaubt keine Aussage über zukünftige Fehlereintritte, da Änderungen in der Nutzung oder von Eingabeparametern und Updates die Eintrittswahrscheinlichkeit von Fehlern unvorhersehbar ändern können. Allerdings ist eine solche Analyse geeignet, die Kritikalität von Softwarebausteinen zu bewerten und Rückschlüsse für die Entwicklung im Sinne von Aufwandsplanung für die Entwicklung und den Test zu erhalten.

Diese Maßnahmen beeinflussen sich gegenseitig und verfolgen die folgenden Ziele:

a) Fehlerabwehr:
   Entstehung von Softwarefehler über einen verbesserten Entwicklungsprozess und konstruktive Maßnahmen möglichst vermeiden.
b) Fehleroffenbarung und -korrektur:
   Vor der Inbetriebnahme möglichst viele der enthaltenen Softwarefehler über ein sicherheitsbezogenes Vorgehensmodell aufdecken.
c) Fehlerauswirkungsausschluss:
   Gefährliche Auswirkungen von Softwarefehlern während des Betriebs verhindern.

1 Anwendungsbereich

Diese Richtlinie ist branchenübergreifend anwendbar, für alle Domänen gültig und bezieht sich nicht ausschließlich auf eingebettete Systeme. Generell wurde sie konzipiert mit dem Ziel, auf Systeme angewendet zu werden, bei denen Software eine Wertschöpfung darstellt.

Sie richtet sich grundsätzlich an Ingenieure, insbesondere an Personen, die an der Entwicklung von Software beteiligt sind (beispielsweise Softwareentwickler, Systemarchitekten, (technische) Projektleiter) aber auch an Gutachter sowie Anwender, Auftraggeber, Nutzer oder Kunden.

Die Richtlinie ist als Empfehlung gedacht, die Softwarezuverlässigkeit planbarer und greifbarer macht.

Die funktionale Sicherheit und IT-Sicherheit werden in dieser Richtlinie nicht berücksichtigt. Zudem ist es nicht Ziel dieser Richtlinie Lehrbücher oder vertiefende Fachartikel zu ersetzen. Eine allgemeine Übersicht über die bestehende Normenlandschaft und fehlende Aspekte soll nicht gegeben werden.