Schichtenbetrachtung bei der Genehmigung von Software aufgrund von Betriebserfahrung

Wolfgang Ehrenberger

Fachbereich Angewandte Informatik, Fachhochschule
36039 Fulda
Wolfgang.Ehrenberger@informatik.fh-fulda.de

Abstract: Um aus der Betriebserfahrung mit Software gewonnene Zuverlässigkeitsangaben von einer Anwendung für eine andere übernehmen zu können, müssen bestimmte Voraussetzungen erfüllt sein. Insbesondere muss bekannt sein, welche *Betriebszeit* oder welche Anzahl von *Abläufen* bezüglich einer bestimmten Anforderungsart stattgefunden hat. Weiterhin ist wichtig, zu wissen, welche *Module* an der Abarbeitung einer bestimmten Art von Anforderungen beteiligt waren. Daraus kann man errechnen, unter welchen Grenzen die Versagensraten einzelner Schichten von Modulen liegen und auf das Verhalten des Gesamtsystems auch in einer neuen Anwendung schließen.

1 Einleitung

Bei der Genehmigung von Software ist es wichtig, der Sicherheitsrelevanz angemessene Anforderungen zu stellen. Daher hat der Standard [IEC 61508], Teil 1 vier Sicherheitsstufen (SILs) eingeführt. Dem lag auch das Bestreben nach bestmöglicher Austauschbarkeit von Softwareteilen zugrunde: Es sollte möglich sein, Software von einer Anwendung in eine andere zu übernehmen, ohne eine aufwändige Neu-Qualifizierung ihrer Module durchführen zu müssen. Ein weiterer wichtiger Gesichtspunkt war der Wunsch, die Qualifikation nach einer bestimmten Sicherheitsstufe aufgrund von Betriebserfahrung durchführen zu können. Besonders bei viel verwendeten Programmteilen sollte es möglich sein, sie aus einer Betriebsumgebung ohne umfangreiche Nacharbeit in eine andere zu übernehmen. Deshalb wurden die SILs auch probabilistisch quantitativ und nicht, wie bis dahin oft üblich, qualitativ definiert.

Dieser Beitrag geht der Frage nach, wie man das in der Vergangenheit angewandte Eingangsdaten-Profil in ein anderes, das in neuer Umgebung zu verwendende, umrechnen könne. Hierfür führen wir den Begriff der Schicht ein. Eine Schicht besteht aus den bei einer Art von Eingangsdatenkombinationen aufgerufenen Modulen. Deren Aufrufhäufigkeit und -Art muss vermerkt werden. Damit ist auch die Umrechnung von Erfahrung, die in einem *Zeitabschnitt* gewonnen wurde, auf Software möglich, die als *auf Anforderung arbeitend* betrachtet wird und umgekehrt.

2 Berücksichtigung des Anforderungsprofils

Wir befassen uns zunächst damit, worauf man sinnvoller Weise Betriebserfahrung beziehen solle. Hierzu gibt es verschiedene Möglichkeiten. Aus Platzgründen beschränken wir uns auf das Folgende und definieren bzw. nehmen an:

(1) Betriebserfahrung bezieht sich auf Software-Module.
(2) Ein Software-Modul ist ein Unterprogramm.
(3) Ein Software-Modul ist in sich abgeschlossen.
(4) Ein Modul hat eine überschaubare Größe.
(5) Eine Schicht besteht aus den Modulen, die an der Abarbeitung einer Anforderung oder einer Art von Anforderungen (bzw. Eingangsdatenkombinationen) beteiligt sind.
(6) Während des Sammelns von Betriebserfahrung hält ein Zähler fest, wie oft der Modul aufgerufen (benutzt, angefordert) wird, oder es liegt eine konservative Schätzung der Zahl der Aufrufe (Benutzungen, Anforderungen) vor.
(7) In gleicher Weise wird gezählt oder konservativ geschätzt, wie oft eine Schicht im Laufe des Betriebs aufgerufen wurde. Daraus wird errechnet oder geschätzt, wie viel Betriebszeit in einer Schicht verbraucht wurde.

Wenn bei der Abarbeitung von zunächst als gleich angesehenen Anforderungen oder Anforderungsarten bzw. Eingangsdaten unterschiedliche Folgen von Modulen zum Zuge kommen, hat man die Schichten zu allgemein definiert. Dies kann zu Schwierigkeiten bei späteren Berechnungen führen. Dann empfiehlt sich eine Neudefinition der jeweiligen Schicht in Form einer Aufspaltung in mehrere Schichten.

Nachdem sicherheitsrelevante Programme mit großer Sorgfalt entwickelt zu werden pflegen, darf man davon ausgehen, dass:

(8) nur solche Programme einem Genehmigungsverfahren unterzogen werden, von denen kein Programmversagen beobachtet wurde.

Wir nehmen also an, dass jeder Fehler, der einem beobachteten Programmversagen zugrunde lag, beseitigt oder maskiert worden sei, *bevor* die für das Genehmigungsverfahren erforderliche Datensammlung nach (7) begonnen wurde.

Aus den Testanzahlen oder der zeitlichen Inanspruchnahme der einzelnen Schichten lassen sich die Versagenswahrscheinlichkeiten oder Versagensraten der einzelnen Schichten errechnen. Nach [IEC 61508] Teil 7 ergibt sich die Obergrenze $\lambda_{i,\alpha}$ der Versagensrate der Schicht i zu einem Signifikanzgrad α für $\alpha = 5\%$ zu:

$$\lambda_{i,\alpha} = - \ln\alpha / T_i = 3 / T_i.$$

T_i ist dabei die Zeit, z.B. in Stunden, über die Schicht i beansprucht oder getestet wurde. Die Gesamtzeit des Tests oder Probebetriebs ist ΣT_i. Die relative Häufigkeit, mit der die Schicht Nr. i während des Tests oder Probebetriebs zum Zuge gekommen ist, ergibt sich als $\pi_i = T_i / \Sigma T_i$. Die Versagensrate des Gesamtprogramms läßt sich durch

$$\lambda_{ges,\alpha} = - \ln\alpha / \Sigma T_i$$

zum Signifikanzgrad α nach oben beschränken. Alle π_i zusammen ergeben das Anforderungsprofil.

Tabelle 1 zeigt ein Beispiel für eine akkumulierte Betriebserfahrung von je 3000 Stunden in 1000 Anwendungen. Obige Formel führt zu einer oberen Schranke von 10^{-6} für die Versagensrate und qualifiziert damit das Beispielprogramm zu SIL 2.

Tabelle 1: Betriebserfahrung in Stunden in drei Schichten, insgesamt $3*10^6$ Betriebsstunden; Aufteilung nach Anforderungshäufigkeit der Schichten.

Schicht	**1**	**2**	**3**
π_i	1/3	1/2	1/6
T_i/h	10^6	$1,5*10^6$	$0.5*10^6$
$\lambda_{i,\alpha}$	$3*10^{-6}$	$2*10^{-6}$	$6*10^{-6}$

3 Umrechnung in ein neues Anforderungsprofil

Bei bekannten bisherigen π_i und $\lambda_{i,\alpha}$ und geschätzten neuen $\pi_{i,neu}$ besteht die Möglichkeit, von einem alten Profil auf ein anderes umzurechnen, also die bisherige obere Schranke für die Versagensrate auf eine neue obere Schranke für die künftig zu erwartende Versagensrate zu übertragen.

Für beliebige $\pi_{i,neu}$ wird i. a. die Gesamtzeit ΣT_i der bisherigen Betriebserfahrung nicht mehr der zu erwartenden Beanspruchung des neuen Systems entsprechen, so daß die bisherige Formel für $\lambda_{ges,\alpha}$ noch anzupassen ist.

Es stellt sich heraus, daß der naheliegende Ansatz $\lambda_{ges,\alpha,neu} = \Sigma\ \pi_i * \lambda_{i,\alpha}$ infolge der Kombination einseitiger Vertrauensbereiche i. a. zu einer nur beschränkten Aussagekraft führt. Wie sich zu gleichem Signifikanzgrad stärkere Aussagen herleiten lassen, wird in [Ehr02] vorgestellt. Für das Beispiel aus Tabelle 2 ergibt dieser Ansatz eine neue obere Schranke von ca. $2,25*10^{-6}$ für die Versagensrate, was nur noch der SIL 1 entspricht.

Tabelle 2: Neue Betriebsumgebung; zu erwartende Häufigkeiten von Schicht i im neuen System

Schicht	**1**	**2**	**3**
$\pi_{i,neu}$	49/100	1/100	1/2

4 Ergänzende Bemerkungen

Falls in einem sicherheitsrelevanten Programm die Module einiger Schichten mit Hilfe deterministischer Verfahren als stets korrekt arbeitend bewiesen worden sind, dürfen ihre Versagensraten jeweils mit 0 angenommen werden. Sind nicht alle Module einer Schicht als korrekt nachgewiesen worden, gilt der Grundsatz: „Eine Kette ist so stark, wie ihr schwächstes Glied."

Manchmal soll die Software nicht im Zeitbereich beurteilt werden, sondern als auf Anforderung arbeitend. Es geht also bei der Genehmigung nicht darum, eine Oberschranke $\lambda_{ges,\alpha}$ für die Versagensrate nachzuweisen, sondern eine entsprechende Oberschranke der Versagenswahrscheinlichkeit $p_{ges,\alpha}$ pro Anforderung aufzuzeigen. Beide Fälle lassen sich einfach ineinander umrechnen. Es gilt mit der jeweiligen Betriebszeit T und der in dieser Zeit angefallenen bzw. anfallenden Anforderungsanzahl n:

$$\lambda * T = n * p,$$

weil die Anzahl der Versagensfälle in einem Zeitraum davon unabhängig ist, ob das System kontinuierlich oder auf Anforderung arbeitet.

Literaturverzeichnis

[IEC 61508] Functional safety of electrical/electronic/programmable electronic safety-related systems 7 parts, 1997, new committee draft of parts 1, 2, 3, 4 November 2005
[Ehr02] Ehrenberger, Software-Verifikation, Carl Hanser Verlag, 2002, SS 120..129