Inhalt

Unterschreiben Sie links unten! *

„… Sie erkennen an, dass keine Software fehlerfrei ist, und wir raten Ihnen dringend, Ihre Dateien regelmäßig zu sichern. Sie erkennen an, dass die vorstehende Garantie Ihre einzige von dem Hersteller gewährte Garantie in Bezug auf die SOFTWARE ist. …“

Da will Sie doch jemand aufs Glatteis führen!? Unverschämt! Da kommt so eine kleine dahergelaufene Softwareklitsche und schwatzt der Kundschaft fehlerhafte Software auf.

Weit gefehlt! Tatsächlich handelt es sich um ein Zitat aus der EULA (ENDBENUTZER-LIZENZVERTRAG) für Microsoft Software.
[Ziffer 24. ;HERSTELLERGARANTIE]

Wie kann das sein? *

Software wird von Menschen geschrieben. Menschen machen bekanntlich Fehler. Je komplexer eine Aufgabe, desto höher die Fehlerrate. Typische Werte für Fehler pro 1000 Zeilen Quellcode (*Text aus dem mit Hilfe von Computerprogrammen ein Computerprogramm erzeugt wird*) sind:

– Bei industrieller Produktion gefordert: 0.3 bis 2 Fehler pro 1000 Zeilen

– Realistisch: Ein Fehler pro 100 Zeilen Quellcode

Windows 2000, XP, Vista und Windows 7 bestehen aus mehr als 50.000.000 Zeilen Quellcode. Man geht von einer Fehlerrate von 1.2 Fehlern pro 1000 Zeilen Quellcode aus. Damit dürfte die Zahl an Fehlern im Betriebssystem bei ca. 60.000 (sechzig tausend) liegen!!!!

Das Dilemma der Softwarehersteller *

Für Softwarehersteller ist das Betriebssystem ein Fundament, auf dem aufgebaut werden muss. Ein **Haufen Mikado-Stäbchen**, auf den man weitere Mikados so legen muss, dass das Gebilde nicht zusammenbricht. Bei sehr systemnaher Software (*z.B. AntiViren-Software, Firewall etc.*) kommt ein weiterer Umstand zum tragen. Diese Art von Software modifiziert das Fundament selbst. Um beim Bild vom Bau zu bleiben, setzt diese Software eigene Bohrungen an, reißt Wände ein, zieht neue Streben. Bei diesen Aktionen wird das Fundament nicht stabiler, sondern brüchiger und wackliger. Es entsteht ein äußerst empfindliches Gebilde bestehend aus zahlreichen Einzelkomponenten, die für sich betrachtet ausreichend stabil und verlässlich, im Zusammenspiel oft jedoch instabil und unzuverlässig sind.

Anwendungsprogrammierer sind keine besseren Menschen *

Zudem sind die Programmierer von Anwendungs- und Systemsoftware nicht (viel) besser, als die Programmierer eines Betriebssystems. Alle haben es mit Rahmenbedingungen zu tun, die es nahezu unmöglich machen, eine Software fehlerfrei zu erzeugen. Im Bereich Luft- und Raumfahrt, ein Gebiet, dem ich selbst entstamme, wird bekannter Maßen getestet, „bis die Balken glühen“. Und dennoch …

1. Ariane 5 (4. Juni 1996 Explodiert 40 Sekunden nach dem Start – Software stammte von Ariane 4. Ariane 5 erreichte jedoch höhere Geschwindigkeiten. Beim Rechnen mit den größeren Zahlenwerten kam es zum Speicherüberlauf und in der Folge zur Einleitung der Selbstzerstörung. Die Entwicklungskosten betrugen ca. 7 Milliarden US-Dollar).

2. Mars Climate Orbiter (23.09.1999 – 2 Gruppen waren an der Entwicklung der Steuersoftware beteiligt. Eine Gruppe rechnete mit der Einheit Meter, die andere in der Einheit inch. Die angesteuerte Umlaufbahn lag dadurch ca. 170 KM

unterhalb der geplanten. Die Folge: Absturz.

3. Mars Sojourner (1997 – Eine Art Deadlock [Zustand in dem das Programm endlos wartet] veranlasst eine Watchdog Komponente [überwacht den Zustand eines Systems "von außen"] zum Neustart des Bordcomputers, wodurch noch nicht zur Erde übertragene Daten verloren gehen.

Gibt es einen Ausweg? Leider ist hier für reale und in der Praxis eingesetzte Programme nichts in Sicht. Jeder, der sich von der wissenschaftlichen Seite dem Feld Programmierung (*Informatik*) nähert, kommt im Studium mit dem s.g. Korrektheitsbeweiß in Kontakt. Mühsam und über Seiten hinweg wird die Korrektheit einfachster Schleifenkonstrukte bewiesen. Man rechnet sich für wenige Zeilen Quellcode bereits einen "Wolf". Neben dem formalen Beweis der Korrektheit gibt es allerdings die s.g. Verifikation (*man testet die Korrektheit eines Programms anhand bestimmter Testfälle*). Mit einer solchen Verifikation wird also nicht die Korrektheit bewiesen, sondern man überprüft stichprobenartig, ob ein Programm für bestimmte Eingaben die gewünschten Ausgaben liefert. **Durch Testen kann immer nur die Anwesenheit von Fehlern** (*wenn man einen Fehler entdeckt, hat man seine Existenz bewiesen*)**, nie jedoch deren Abwesenheit bewiesen werden.**

Als Hersteller muss man sich also s.g. Testfälle erzeugen und diese „durchspielen". Hier stößt man schnell an die Grenzen des Machbaren. Es gibt unzählige Testfälle (***kombinatorische Explosion*** *– Hat man eine einfache Funktion, die 2 Byte [jeweils 8 BIT] Werte verarbeitet, gibt es bereits 2 ^(8+8) – entspricht 65536 – theoretisch mögliche Testfälle – werden 2 Ganzzahlen verarbeitet, liegt man bereits bei 4 Milliarden*), die man unmöglich alle verifizieren kann. Daher gibt es den Begriff der **Testabdeckung**, der, vereinfacht ausgedrückt, das Verhältnis der Anzahl tatsächlich durchgeführter Tests zur Anzahl theoretisch durchführbarer Tests angibt. Die Testabdeckung ist also ein Maß für die Qualität eines Produktes. Die Kunst besteht allerdings nicht darin, möglichst viele Testfälle abzuarbeiten, sondern vielmehr darin, wichtige Testfälle zu finden und sich auf diese zu beschränken. Als wichtigen Testfall bezeichne ich Testfälle, bei denen die Eingangswerte für eine bestimmte Funktion die Funktion am ehesten aus dem Tritt bringen (*Grenzfälle*).

Geld, Zeit, Marketing *

Softwarehersteller sehen sich oft durch viele äußere Zwänge veranlasst, eine Software nahezu ungetestet zu veröffentlichen. Eine wichtige Messe steht vor der Tür, Geld muss in die leeren Kassen gespült werden, die Marketingabteilung sitzt einem im Nacken. Natürlich muss man an einem bestimmten Punkt den Entschluss fassen, die Software zu veröffentlichen, da man mit aller Zeit der Welt die Software nie fehlerfrei bekommt. Allerdings rächt es sich schnell, wenn man gar zu nachlässig wird. Presse, Kunden und der eigene Support freuen sich über jede fehlerstrotzende Software.

Verständnis *

Sicher werden Sie jetzt zumindest verstehen, warum selbst einfachste kleine Programme immer Fehler enthalten. Fehlerfreie Programme werden auf lange Sicht ein Wunschtraum bleiben. Dennoch gibt es gute und schlechte Software. Gute Software hat ausreichend getestete Hauptfunktionen (*eine Hauptfunktion wäre zum Beispiel in einem Schreibprogramm die Schriftart auf fett zu setzen*) und läuft auf 99 von 100 Rechnern im Alltag ohne Probleme. Gute Software zeichnet sich auch durch guten Support und reaktionsschnelle Fehlerbeseitigung bei kritischen Fehlern aus.

Zum Abschluss *

Angenommen Sie haben eine Software, die Ihnen in den Einstellungen 7 Schalter bietet, bei denen Sie ein Häkchen setzen können. Was denken Sie, wie viele verschiedene Kombinationen es gibt, die Häkchen zu setzen? Wenn Sie eine Viertelstunde Zeit haben, probieren Sie es doch einfach mal aus. Pech, dass Sie je Sekunde ca. 6 der insgesamt 5040 verschiedenen Möglichkeiten testen müssen, um in 15 Minuten fertig zu sein!

Ein toller Freizeittipp um an verregneten Sommertagen die Zeit zu verbringen ist auch der, einfach mal die 10 Bücher im Regal in allen möglichen Reihenfolgen aufzustellen. Nach 420 Tagen könnte man es geschafft haben, vorausgesetzt, man stellt alle 10 Sekunden eine der 3.628.800 möglichen Reihenfolgen auf. Schlaf ist natürlich tabu!