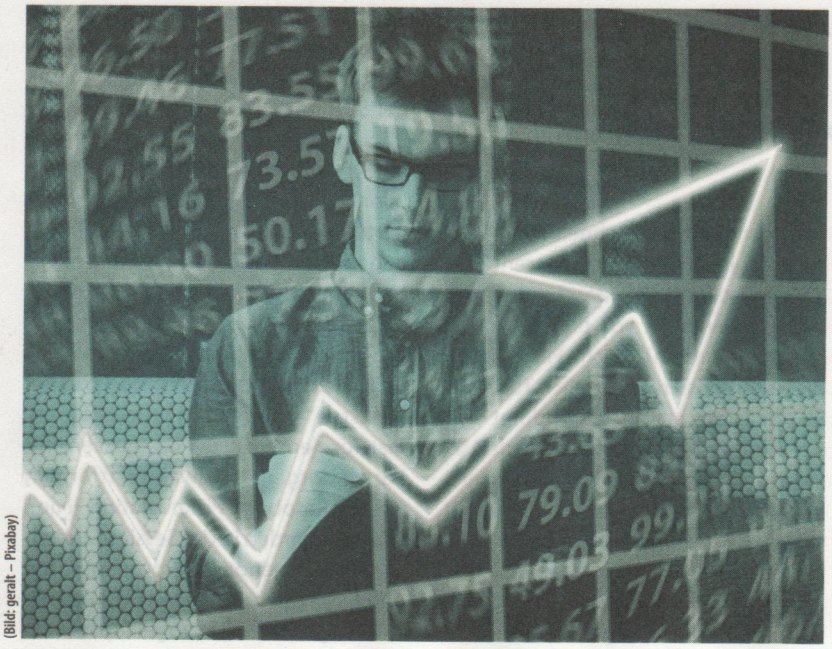
Software-Entwicklung:

Sieben Regeln erfolgreicher Programmierer



(Bild: geralt – Pixabay)

Testen von Software hat viel mit sportlicher Betätigung zu tun: Es ist mühsam, anfällig für Übertreibungen und stets lauert die Versuchung, es ganz auszusetzen. Wie also vorgehen, um die Software-Qualität dauerhaft und kontinuierlich zu sichern?

Dass Tests für die Produktion von qualitativ hochwertiger Software außerordentlich nützlich sind, ist unumstritten. Aber das Testen ist auch eine Herausforderung: Man muss immer am Ball bleiben und am wirkungsvollsten ist es in kleinen Dosen, denn ausgiebiges Testen ist sehr ermüdend. Täglich zehn Minuten sind also besser als einmal pro Woche zwei Stunden. Würden Sie beim Sport alles auf die letzte Woche im Jahr schieben und dann am Stück trainieren? Das Beispiel mag lächerlich erscheinen, aber genauso gehen Software-Entwickler und Prüfer überall auf der Welt vor. Ohne Pflege verlieren Test-Suites ihre Genauigkeit, ihr Zustand verschlechtert sich zusehends, sie werden nutzlos und dadurch ignoriert. Dasselbe passiert mit der statischen Analyse: Setzt man sie als Test-Tool ein, erzielt sie ein sehr ungünstiges Verhältnis zwischen Aufwand und Nutzen, und wird letztendlich nicht weiter genutzt. Dasselbe gilt für andere wichtige Praktiken, die Teil der regelmäßigen Code-Prüfung sein sollten, um die Zuverlässigkeit, Qualität und Sicherheit zu verbessern, egal in welchen Industrien die Entwickler arbeiten. Die nachfolgenden Best Practices haben sich bei erfolgreichen Entwicklern aller Branchen immer wieder bewährt.

1. Peer Reviews

Eine andere oder auch mehrere Personen einen Blick auf den Code werfen zu lassen, ist ein effektives Mittel zum Aufdecken gravierender Probleme, bevor sie bis zum finalen Anwender gelangen. Will man diese Praktik zum Bestandteil der täglichen Routine machen, sollte man sich vor Übertreibungen hüten. Versuchen Sie nicht, eine ganze Code-Basis überprüfen zu lassen, sondern beschränken Sie sich auf die Änderungen, die mit einer Funktion oder Nachbesserung zu tun haben. Bei der mühsamen Arbeit in Sachen Programmierstil, optimale Verfahrensweisen und Sicherstellung der Konformität ist die Nutzung der statischen Analyse empfehlenswert.

2. Statische Code-Analyse

Erstellen Sie einen guten präventiven Regelsatz, der sich direkt auf Probleme anwenden lässt, die in der Vergangenheit vorgefunden wurden oder die man in Zukunft vermeiden möchte. Es ist sinnvoll, die statische Analyse nicht in der Qualitätssicherung anzusiedeln, sondern unbedingt an den Arbeitsplätzen der Entwickler, damit diese frühzeitig Rückmeldung erhalten und etwaige Probleme beheben können. Jegliche Konformitätsprobleme sollten Bestandteil der Konfiguration für die statische Analyse sein.

3. Schreiben von Modultests

Schreiben Sie laufend Modultests – einen für jede Datei oder einen für jede Änderung. Diese Tests gleich beim Programmieren zu schreiben, kostet weniger Zeit, denn man weiß ja bereits, was der Code leisten soll. Die Tests sollten außerdem so robust werden, dass sie auch auf einer anderen Maschine oder an einem anderen Tag noch funktionieren.

4. Ausführung von Modultests

Je seltener man Tests ausführt, umso ungenauer werden sie – getreu dem zweiten Hauptsatz der Thermodynamik. Es ist empfehlenswert, die Ungenauigkeiten bei jedem Release sukzessive zu verringern. Niemand hat die Zeit, alles sofort zu erledigen. Man sollte deshalb versuchen, im aktuellen Release etwas besser zu sein als im vorhergehenden.

5. Bessere Code-Überdeckung

Zusätzlich zum Erstellen neuer Modultests sollte man durch Messen der Code-Überdeckung sicherstellen, dass der Umfang des Codes, den man testet, immer größer wird. Arbeiten Sie daran, die Maßzahl schrittweise zu steigern, vielleicht im aktuellen Release um 5 % gegenüber dem letzten. Moderne Prüftechnologien wie die Servicevirtualisierung erzielen eine Steigerung dieser Zahl, damit man auch komplizierte Systeme prüfen kann.

6. Messen

Was man nicht misst, bekommt man auch nicht in den Griff. Wie lässt sich sonst feststellen, ob sich die Situation

verbessert oder verschlechtert? Wertvoll sind Daten zu Entwicklungsaktivitäten, wie etwa die Zahl der Check-ins, der gefundenen Fehler oder der durch statische Analyse gefundenen Regelverstöße, Informationen über Code-Überdeckung usw. Im ersten Durchgang geht es nur um das reine Sammeln der Daten, bis sich nach einer Reihe von Releases abzeichnet, ob die Zahlen nach oben oder nach unten wandern. Weitere Informationen über Praktiken zu Maßzahlen enthält die Präsentation „Metrics that matter".

7. Durchführen von Post-Mortem-Analysen

Nehmen Sie sich nach einem Release die Zeit für eine Analyse, wie es gelaufen ist. Nach 90 Tagen hat man bereits ein gutes Gefühl für die anfängliche Qualität und kann eine Einschätzung abgeben. Was ist schiefgegangen? Wie lässt sich so etwas künftig vermeiden? Der Einsatz der statischen Analyse unterstützt dabei, bestimmte Probleme zu vermeiden. Wenn es um das Verbessern der eigenen Praktiken bei der Software-Prüfung geht, sollte man daran denken, wie schwer es ist, die gefassten Vorsätze einzuhalten. Man sollte sich nicht zu viel auf einmal zumuten. Entscheiden Sie sich, womit Sie beginnen wollen, und gehen Sie eines nach dem anderen an. Weniger ist hier mehr. Kleine Gewohnheiten, konsequent umgesetzt, haben die größte Wirkung.

Arthur Hicken, Product Evangelist bei Parasoft / jk

Plattform für Elektronik Expertenwissen

EMV beherrschen – von den Grundlagen in die Praxis

Müssen Sie sich in Entwicklung, Fertigung oder Qualitätssicherung mit EMV-Problemstellungen befassen? Dann könnte unser „Training for Professionals" für Sie geeignet sein.

Vom 22. bis 23. November erklärt Dr. Heinz Zenkner von Bluechips Microhouse in Haar bei München die Grundlagen und Werkzeuge der EMV. An konkreten Beispielen aus der Praxis wird das Erlernte so vertieft, dass die Teilnehmer das erlernte Wissen auf ihre Produkte projizieren können.

Der Inhalt umfasst:

- ➜ Warum EMV: Störemission, Störfestigkeit, Normung, Signalintegrität, Qualität der Produktentwicklung, EMV-Qualität in der Produktion
- ➜ Phänomene der EMV: Koppelarten, Signalzusammensetzung, Störmodelle mit Faustformeln, Störpfade und Ausbreitungscharakteristik, Konzeptmatrix
- ➜ Gehäusekonstruktion in der EMV: Materialien, Schirmdämpfung, Konstruktionsmaßnahmen, Abstrahlungseigenschaft von Öffnungen und Spalte
- ➜ Bauelemente und Filter in der EMV: Funktionsweise, Anwendung, Layout
- ➜ Layout und Grundschaltungen in der EMV: Schaltungsbeispiele (µC, Clock, Schnittstellen, Netzeile, …)
- ➜ Precompliance-Messungen in der EMV: Störspannung, Feldstärke, ESD, Sonden, Antennen
- ➜ Entstörbeispiele

Weitere Informationen zum Programm und die Anmeldungen finden Sie unter www.training-for-professionals.de