

Automatisierter Test in der Softwareentwicklung

Formale Beschreibungssprache erhöht Automatisierungsgrad

Die Komplexität der mit Hilfe von Software gelösten Aufgaben wächst rasant. In gleichem Maße steigt der Umfang der Software und damit auch der zeitliche Aufwand für die Testzyklen. Gerade in sicherheitsrelevanten Anwendungen in Fahrzeugen, Schiffen oder Flugzeugen lässt sich die erforderliche Softwarequalität ohne automatische Tests kaum noch erreichen.

Die beim Softwaretest zu bewältigenden Probleme scheinen bei wachsender Komplexität geradezu exponentiell zuzunehmen. Idealerweise sollte die Software durch die angelegten Eingabevektoren vollständig getestet werden, wobei diese Tests möglichst ressourcenschonend und in wirtschaftlich vertretbarer Zeit ausgeführt werden sollen. Die Qualitätssicherungsexperten versuchen, eine Reihe von Testfällen zu definieren, die den Quell-Code möglichst vollständig abdecken sollen. Die Durchführung der Tests erfolgt in der Regel manuell, ebenso wie die Dokumentation der Ergebnisse. Eine Gewissheit über das Maß der tatsächlichen Abdeckung des Quell-Codes durch die Testfälle gibt es nicht, und die manuelle Definition und Ausführung der Testsequenzen ist zeit- und personalintensiv.

Die verschiedenen Teststrategien lassen sich in zwei grundsätzliche Kategorien gliedern. Beim Black-Box-Test wird das zu testende System (Implementation unter Test, IUT) ausschließlich von außen betrachtet. Das IUT wird mit einer angemessenen Zusammenstellung von Eingabeereignissen oder Vektoren stimuliert, und die Reaktion wird anhand des von außen sichtbaren Verhaltens beurteilt. Das real gezeigte Verhalten des IUT wird dokumentiert und mit dem gewünschten Verhalten verglichen. Etwaige Abweichungen deuten auf Fehler hin, die ihren Ursprung

z. B. in der gewählten internen Architektur oder in Codierungsfehlern haben können. Beim White-Box-Test wird die interne Struktur des zu testenden Systems betrachtet, um die Ursache eventuell auftretender Fehler im Quell-Code zu lokalisieren. In der Regel erweist sich eine Kombination von Black-Box- und White-Box-Testverfahren am wirkungsvollsten bezogen auf die am Ende erreichte Qualität und die dafür eingesetzten Ressourcen.

Automatisierte Black-Box-Tests

Wenn Softwareprodukte zu testen sind, überwiegen heute noch in weiten Bereichen manuelle Verfahren. Die Testfälle und ihre Abfolge werden manuell definiert und sind meist so spezifisch auf das individuell zu testende Produkt zugeschnitten, dass eine weitere Verwendung für ähnlich gelagerte Applikationen nicht möglich ist. Eine Reihe von Fehlerursachen ist diesem Verfahren zur Testgewinnung inhärent: die manuell erzeugten Testfälle decken möglicherweise nicht den vollständigen Funktionsumfang ab oder sind schlicht falsch definiert.

Die manuelle Ausführung der Tests und der Abgleich der realen Ergebnisse mit dem Zielprofil birgt ebenfalls Risiken: Testreihen können unvollständig ausgeführt werden oder es unterlaufen Fehler bei der Interpretation der Ergebnisse.

Gerade bei komplexen Produkten bietet es sich an, individuelle Lösungen zur automatischen Ausführung der Testfolgen zu entwickeln. In vielen Unternehmen werden solche In-House-Entwicklungen bereits eingesetzt, um die Testphase zu beschleunigen. Die durch die manuelle Testfallbeschreibung bedingten Unschärfen werden jedoch bei einer automatischen Ausführung der Testfolgen keinesfalls geringer. Das Ergebnis ist nur so präzise wie die Eingaben. Ein Beispiel: wenn in einer Tabellenkalkulation ein Ergebnis automatisch berechnet wird, ist es nicht deshalb zuverlässiger, weil es bis auf sechs Nachkommastellen genau berechnet wird. Wenn die Formel oder die Eingabewerte fragwürdig sind, ist es auch das Ergebnis.

Der einzige Weg, die tragfähigen Voraussetzungen für automatische Testverfahren zu schaffen, besteht in der formalen Beschreibung der Testfälle und Testsequenzen. Eine formale Notation, die hersteller-unabhängig standardisiert ist, bringt eine Reihe von Vorteilen:

- ▶ sie ist präzise und eindeutig, sie wird von jedem in der gleichen Weise interpretiert
- ▶ Testbeschreibungen in einer formalen Notation können automatisch erzeugt werden und sind damit a priori korrekt (unter der Annahme, dass der entsprechende Generator korrekt arbeitet)
- ▶ Testbeschreibungen können automatisch ausgeführt werden, so dass Fehler in der Ausführung ausgeschlossen sind

▶ Autorin

Dipl.-Inf. RENATE STÜCKA ist Director of Marketing für den Bereich Automotive und Embedded Systeme bei Telegic Telelogic Deutschland GmbH; Otto-Brenner-Strasse 247, D-33604 Bielefeld
Fon: 0521/4 503-254, Fax: 0521/14503-50
e-Mail: renaete.stuecka@telegic.de

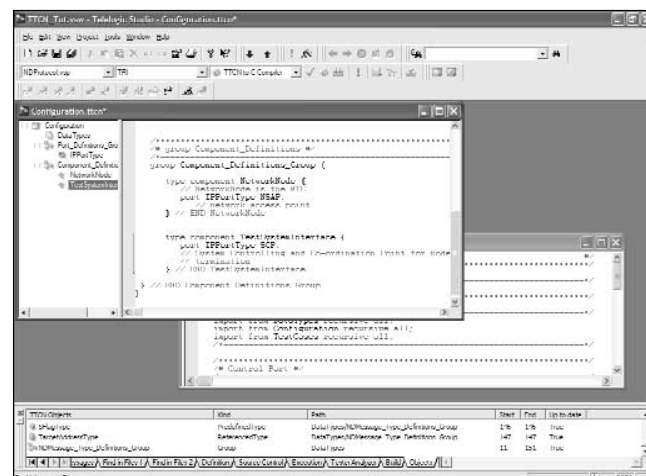


Abb. 1:
Die Syntax der TTCN-3 ist vergleichbar der Syntax einer modernen Programmiersprache

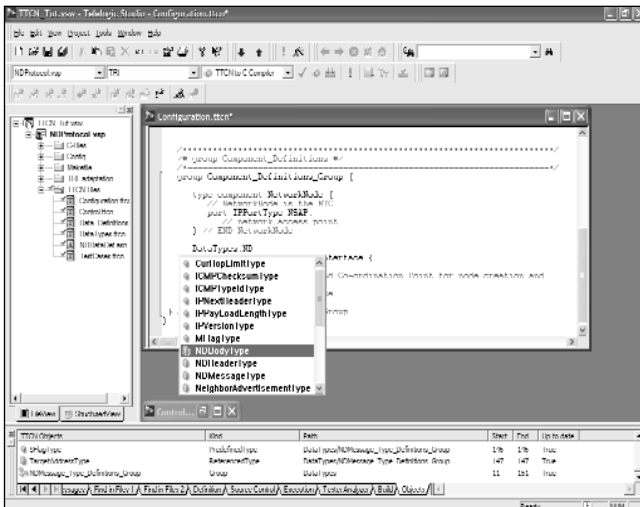


Abb. 2:
Tau/Tester stellt alle notwendigen Informationen und Aktionen auf einen Blick zur Verfügung

- ▶ formale Testbeschreibungen können weiterverwendet werden

Eine formale Testbeschreibungssprache, die in der Telekommunikationsindustrie bereits seit langem bewährt ist, ist TTCN. Diese Sprache wurde ursprünglich entwickelt, um Testsequenzen zur Überprüfung der Implementierung von Protokollen im Rahmen des OSI-Referenzmodells eindeutig und ohne Bezug auf die interne Realisierung zu beschreiben. TTCN ist standardisiert (ISO 9646) und damit hersteller-unabhängig.

Erst seit kurzem gibt es die TTCN-3 (Testing and Test Control Notation). TTCN-3 (Abb. 1) ist nicht einfach eine neue Version des Standards, es handelt sich in weiten Teilen um eine neue Sprache, die von der ETSI (ES 201873-1) entwickelt wurde und inzwischen den Status einer ITU-Empfehlung hat (Z.140). Die Motivation zur Entwicklung von TTCN-3 und die ihr zugrundeliegende Philosophie ist, die in Telekommunikationsanwendungen bewährten Stärken der TTCN-2 zu erhalten und mit Hilfe einer neuen erweiterten Notation auch für die vielfältigen Anwendungen anderer Industrien nutzbar zu machen. Mit TTCN-3 steht heute eine moderne und flexible Sprache zur Verfügung für die Spezifikation vieler Arten von Systemtests über ein breites Anwendungsspektrum. Die Syntax der TTCN-3 ist vergleichbar der einer konventionellen Programmiersprache und beinhaltet unter anderem auch synchrone und asynchrone Kommunikationsmechanismen sowie die Möglichkeit, auch dynamische parallele Tests zu definieren. Damit kann TTCN-3 nicht nur für Protokolltests angewendet werden, sondern auch für den Test von Modulen, CORBA-basierten Plattformen, Anwendungsschnittstellen (APIs) und vieles mehr.

Ein weiteres Highlight der TTCN-3 ist das TTCN-3-Test-Management-Interface (TRI). Dieses Interface erlaubt die Anpassung vorhandener

ausführbarer Testsequenzen an ein konkret zu testendes Zielsystem. Wenn ein TRI-kompatibles TTCN-3-Werkzeug eingesetzt wird, um die abstrakte Beschreibung der Testsequenzen in ausführbaren Code zu konvertieren, bleibt die Anpassung auch bei einem Wechsel des Werkzeugs weiterhin nutzbar: die Investition ist geschützt.

Bei der Object Management Group (OMG) wird daran gearbeitet, ein Testprofil für die UML 2.0 zu entwickeln. Mit Hilfe dieses Profils wird es möglich sein, Testspezifikationen aus einem formalen UML-Modell herzuleiten und zu validieren. So wird der Zeitaufwand zur Erzeugung der Testsequenzen minimiert und die Fehleranfälligkeit manueller Testbeschreibungen umgangen. Dabei wird TTCN-3 eine der empfohlenen Notationen des UML-Testprofils sein.

Obwohl TTCN-3 eine Neuentwicklung ist, beinhaltet sie nach wie vor einige Konzepte der TTCN-2, so dass erfahrene TTCN-2 Anwender keine Schwierigkeiten haben, sich zurechtzufinden und weiterhin die vertrauten Darstellungsformen wählen können. Darüber hinaus ist es möglich, mit Hilfe automatischer

Übersetzungswerkzeuge aus TTCN-2 Beschreibungen neue TTCN-3 Testspezifikationen zu erzeugen.

Die Nutzung der vielfältigen Vorteile der TTCN-3 wird möglich durch den Einsatz eines geeigneten Werkzeugs. Eines der ersten Werkzeuge auf dem Markt, das TTCN-3 unterstützt, ist Telelogic Tau/Tester. Dabei handelt es sich um eine integrierte Umgebung, die den gesamten Testzyklus mit Hilfe eines einzigen Desktops mit dem allgemein verbreiteten vertrauten Studio Interface unterstützt. Tau/Tester (Abb. 2) bietet Testentwurf, -entwicklung, -analyse, -ausführung sowie vielfältige Debug-Funktionen. Damit ist es erstmals möglich, eine einzige Testplattform unternehmensweit für vielfältige Anwendungen einzusetzen und kostenintensive dedizierte Inhouse-Lösungen zu ersetzen.

Ein besonderes Merkmal ist der hohe Grad an Automatisierung, für den der TTCN-3-Standard die Basis bildet und die durch Tau/Tester umfassend umgesetzt wird. Das Projektteam kann sich auf die inhaltliche Arbeit der Testentwicklung konzentrieren, die Generierung, Ausführung und Ergebnisdokumentation der Testsequenzen erfolgt automatisch. Dies erlaubt einen sehr viel effizienteren Arbeitsstil für das Projektteam. Darüber hinaus sind alle Tests exakt gleich und beliebig oft wiederholbar. Eigene Testsequenzen können überdies ergänzt werden durch bereits vorhandene Testfälle aus internen oder auch externen Quellen, wie z.B. Standardisierungsgruppen.

Eine der wichtigsten Neuerungen der TTCN-3 ist die Fähigkeit, auch Testsequenzen für komplexere verteilte Systeme zu definieren, die parallel ausgeführt werden. Dem trägt Tau/Tester Rechnung durch besondere Unterstützung für große, auch über verschiedene Standorte verteilte Projektteams und die nahtlose Integration mit führenden Konfigurations- und Versionsmanagement-Werkzeugen (Abb. 3).

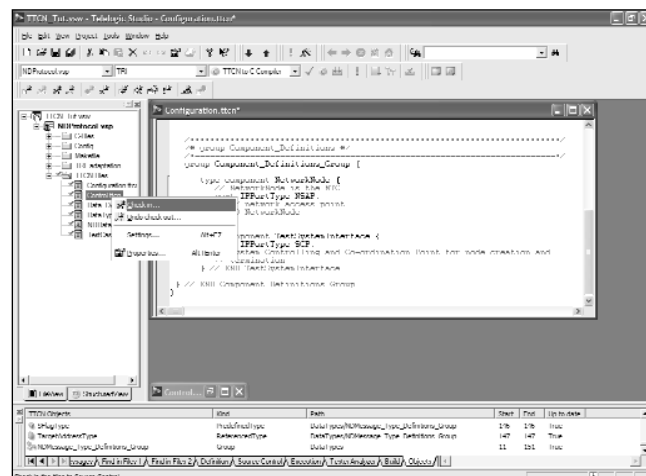


Abb. 3:
Die Integration mit Konfigurations- und Versionsmanagement-Werkzeugen ist in großen Projekten unverzichtbar

Ergänzende White-Box-Tests

Neben dem Nachweis, dass ein System sich nach außen hin so verhält, wie es erwartet wird, interessiert natürlich auch die Frage, ob der Quell-Code tatsächlich im Rahmen der durchgeführten Tests vollständig durchlaufen wurde. Falls dies nicht der Fall ist, sind die häufigsten Ursachen unvollständige Testsequenzen oder fehlerhafte Codierung; es gibt Pfade, die niemals ausgeführt werden („dead code“). Dieser Code ist für sich genommen vielleicht unschädlich, die Tatsache, dass es dead code gibt, deutet jedoch auf schwerwiegende konzeptionelle oder Implementierungsfehler hin.

Die Frage nach der Abdeckung des Quell-Codes durch die definierten Testsequenzen lässt sich beantworten: Der Quell-Code wird vor der Generierung des ausführbaren Codes instrumentiert und anschließend unter Anwendung der Testsequenzen ausgeführt. Als Ergebnis erhält man aussagefähige Reports und graphische Darstellungen, welche die Testabdeckung für jedes einzelne Modul dokumentieren. Dieses Ergebnis ist eindeutig, objektiv und jederzeit reproduzierbar. Basierend auf diesen Messwerten werden nun die Testsequenzen so lange optimiert, bis eine 100-prozentige Abdeckung des Quell-Codes durch die Tests erreicht wird.

Das Zeitverhalten des instrumentierten Codes ist offensichtlich nicht vergleichbar mit

dem Zeitverhalten des Original-Codes. Um dennoch ein fundierte Ergebnis zu erhalten, wird die Instrumentierung nach Ermittlung der Testsequenzen, die für den „100%-Test“ erforderlich sind, entfernt, und dieselben Testsequenzen werden noch einmal, diesmal mit dem realen Zeitverhalten ausgeführt. So hat man Gewissheit, dass der endgültige Code kein einziges Statement enthält, das nicht mindestens einmal ausgeführt wurde.

Auch hierfür ist der Einsatz eines geeigneten Werkzeugs erforderlich. Telelogic Tau Logiscope TestChecker bietet volle Unterstützung für dieses Verfahren zur Messung der Testabdeckung (Abb. 4).

Fazit

Die Entwicklung eines Produkts wird heute nicht mehr als isolierter, individuell zu betrachtender Vorgang angesehen, sondern als ein wesentlicher Bestandteil eines umfassenden Produktlebenszyklus verstanden. Dazu zählen Anforderungsanalyse, Modellierung, Spezifikation, Test und Dokumentation. Moderne Prozesskonzepte zur Unterstützung des Produktlebenszyklus messen dem Qualitätsmanagement besondere Bedeutung bei: Maßnahmen zur Sicherstellung oder Verbesserung der Qualität durchziehen den gesamten Prozess. Dabei gewinnt der Test zunehmend strategische Bedeutung. Würde die

Testphase früher als eher untergeordnete Teilfunktion wahrgenommen, so wird heute mit der unaufhörlich wachsenden Komplexität der Produkte die Testphase als essentielle Voraussetzung für eine akzeptable Qualität immer wichtiger. Gleichzeitig wird auch immer deutlicher, dass eine effiziente Testphase nicht einzeln für sich steht, sondern bereits in früheren Projektphasen berücksichtigt werden muss, und weitere Iterationen im Produktlebenszyklus beeinflusst.

Daher ist es zwingend notwendig, bei der Anwendung von Methoden und Werkzeugen während der Testphase dasselbe hohe Niveau anzustreben, das bei der Modellierung und Spezifikation für viele Unternehmen bereits selbstverständlich ist. Formale Sprachen, angemessene methodische Ansätze und entsprechende Werkzeuge helfen dabei, Testmethoden, -definition und -ausführung zu vereinfachen, zu automatisieren und zuverlässiger zu gestalten. Mit TTCN-3 sind die formalen Voraussetzungen dafür geschaffen worden, und es gibt mit Telelogic Tau/Tester bereits ein leistungsfähiges Werkzeug, welches das Potential von TTCN-3 für den praktischen Projekteinsatz erschließt. Die ergänzende Messung der Testabdeckung, z.B. mit Hilfe von Telelogic Tau Logiscope ist eine sinnvolle Maßnahme für Anwendungsbereiche, in denen es auf ein besonders niedriges Fehlerisiko ankommt. **TEST**

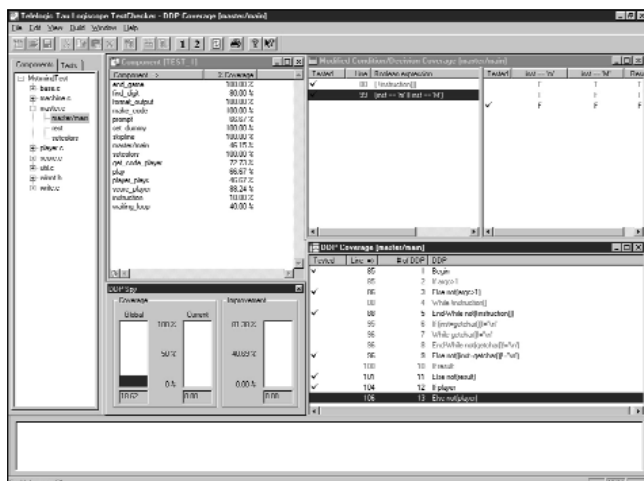


Abb. 4:
Die Abdeckung des Quell-Codes durch die Tests wird eindeutig und reproduzierbar gemessen

Literatur

- [1] The Evolution of TTCN, ITU 2001, <http://www.itu.int/ITU-T/studygroups/com17/ttcn.html>
- [2] ITU Recommendation Z.140, Tree and Tabular Combined Notation Version 3, 2001
- [3] Johan Nordin, Telelogic: Meeting the Demands of Software Testing, 2002, <http://www.telelogic.com/products/tau/languages/ttcn.cfm>
- [4] Telelogic White Paper: TTCN-3 The Ultimate Test, <http://www.telelogic.com/products/tau/languages/ttcn.cfm>
- [5] Telelogic Tau/Tester, <http://www.taustester.com/>

www.publish-industry.net
more @ click TK3A0103

LESERTIPP Sie suchen nach Beiträgen zu der Thematik EMV-Messtechnik?

Unter der Griffmarke B.03 finden Sie aktuelle und praxisnahe Beiträge!

publish industry
TECHNIK KOMMUNIZIEREN

Gollierstraße 23 · D-80339 München · Fon. +49/89/500383-0 · Fax. +49/89/500383-10 · info@publish-industry.net · www.publish-industry.net