Buchbesprechungen

M. Pol, T. Koomen, A. Spillner: **Management und Optimierung des Testprozesses - Ein praktischer Leitfaden für erfolgreiches Testen von Software mit TPI und TMap**; dPunkt-Verlag, Heidelberg, 2000

Immer wieder melden die Medien Nachrichten über (mehr oder weniger katastrophale) Software-Fehler, wobei die Auswirkungen mit dem zunehmenden Grad der „Computerisierung“ unserer Alltagswelt für eine rasch wachsende Anzahl von Menschen spürbar werden. Sei es die Regelung des Herzschrittmachers, die Einkommenssteuererklärung mit dem Heimcomputer oder der Bordcomputer des Airbus A-320 - überall zieht unsichtbar im Hintergrund Software an den Fäden der Geschehnisse. Um Software-Fehlern entgegenzuwirken, wurden seit den siebziger Jahren eine Vielzahl von mittlerweile auch in deutschsprachigen Büchern erschienenen Testmethoden entwickelt. Pol, Koomen und Spillner sagen hierzu: „Die bloße Kenntnis der [Test-]Methoden reicht für deren Einsatz allerdings nicht aus. Das Umfeld der Entwicklung muss so gestaltet werden, dass eine Qualitätssteigerung des Produkts erreicht wird.“ Gemäß diesem Zitat aus dem Buchrückentext gehen Pol, Koomen und Spillner die komplexe Problematik der Software-Qualitätssicherung gleich in doppelter Weise an. So stecken denn in den zwei Teilen ihres Werkes - wie es auch schon im Titel „Management und Optimierung des Testprozesses“ anklingt - eigentlich zwei Bücher: Eines, das die praxiserprobte Methode TPI (Test Process Improvement) zur Optimierung des Testprozesses vorstellt, und eines, welches das ausgereifte Testkonzept TMap (Test Management Approach) für IT-Systeme zum Thema hat.

TPI identifiziert im strukturierten Testprozess 20 Kernbereiche wie z.B. Teststrategie, Testplanung, Metriken oder Testspezifikation. Diese Kernbereiche werden weiter in durchschnittlich drei aufeinander aufbauende (Optimierungs-)Ebenen unterteilt. Um einen Kernbereich eines konkreten Testprozesses einer bestimmten Ebene zuzuteilen, müssen bestimmte quantifizierbare Kontrollpunkte erfüllt werden. Jedem Kernbereich ist auf jeder Ebene ein Bündel von Optimierungsvorschlägen zugeordnet, mit dem der Testmanager gezielt seinen Testprozess verbessern kann. Der Einsatz des TPI-Modells wird für alle Kernbereiche ausführlich erläutert und mit konkreten Fallbeispielen unterlegt. Im Gegensatz zu anderen aus der Literatur bekannten Modellen wie z.B. dem CMM (Capability Maturity Model) oder der ISO 9000 Normenreihe erleichtert der hohe Grad an Konkretheit des TPI den praktischen Einsatz.

TMap basiert auf den vier Pfeilern eines strukturierten Testprozesses: ein mit dem Vorgehensmodell der Entwicklung im Zusammenhang stehendes „Phasenmodell“ der Testaktivitäten, eine gute „organisatorische Einbettung“, die richtige „Infrastruktur“ sowie brauchbare „Techniken“ für die Durchführung der Testaktivitäten. Diese vier Pfeiler werden ausführlich im zweiten Teil des Buches behandelt. Eingebettet in die fünf Phasen „Planung&Verwaltung“, „Vorbereitung“, „Spezifikation“, „Durchführung“ und „Abschluss“ des TMap Phasenmodells werden die einzelnen Aktivitäten des Testprozesses in einem einheitlichen, übersichtlichen Beschreibungsrahmen vertieft. Die Kapitel über Test-Spezifikationstechniken und Testwerkzeuge können als umfassende Einstiege in diese Bereiche gewertet werden. Zahlreiche Checklisten unterstützen die Umsetzung von TMap im eigenen Projekt. Ein Anhang, in dem die in TMap verwendeten Qualitätsmerkmale auf die der ISO-9126 abgebildet werden, erleichtert die normenkonforme Anwendung der Methode.

Trotz der etwas knapp gehaltenen Hinweise auf ergänzende Literatur kann ich das Buch als Handbuch für den Qualitätsmanager als auch für den im Projekt stehenden Tester uneingeschränkt empfehlen.

M. Winter, Praktische Informatik III, Fernuniversität Hagen; E-mail: Mario.Winter@FernUni-Hagen.de

Graeme P. Smith: **The Object-Z Specification Language**; Kluwer Academic Publishers, Boston, Dordrecht, London, 2000 (ISBN 0-7923-8684-1; xi+146 Seiten US $ 146.00)

Die von M. Spivey und seiner Oxforder Gruppe eingeführte und untersuchte Spezifikationssprache Z basiert auf einer Mischung aus Prädikatenlogik erster Stufe, Mengenlehre und kryptischer Notation. Z strukturiert Spezifikationen durch eine Folge von lose miteinander verbundener Schemata (ähnlich wie dies C mit Funktionen tut). Aus einer Z-Spezifikation kann durch eine Folge von Verfeinerungsschritten eine Implementierung abgeleitet werden (auch wenn dieser Kalkül nicht soweit entwickelt zu sein scheint wie R. Paiges *Reduktion der Stärke*). Z ist als Spezifikationssprache attraktiv, es fehlen aber doch einige Konzepte, die inzwischen als unabdingbar angesehen werden. Hierzu gehört sicher die Möglichkeit, Abstrakte Datentypen zu definieren (also Daten und auf ihnen operierende Funktionen in einer gemeinsamen Kapsel zu organisieren und wohldefinierte Export-Schnittstellen anzubieten) und das Konzept der Vererbung (eigentlich kein einheitliches Konzept, eher eine Kollektion von Mechanismen zur Modellierung von Spezialisierungsbeziehungen).

Objekt-Z nimmt sich dieser Probleme an, das zu besprechende Buch stellt diese Spezifikationssprache vor. Der Verfasser, Graeme Paul Smith, legte in seiner Dissertation von 1992 an der neuseeländischen University of Queensland einen Entwurf einer objektorientierten Erweiterung von Z vor. Hieraus ist schließlich das in Rede stehende Buch hervorgegangen. Es setzt für die Lektüre Z im Umfang etwa des Buchs von Spivey voraus und lehnt sich im Vorgehen im übrigen auch an dieses Buch an. Die Einführung diskutiert die Konstruktionen von Objekt-Z, führt also Klassen sowie Objekte ein und diskutiert Vererbung sowie Polymorphie. Eine sehr schöne Fallstudie zeigt auch gleich, wie sich das bekannte Spiel *Tetris* in Objekt-Z modellieren läßt. Die Modellierung ist vollständig und zeigt ziemlich nachdrücklich die beiden prinzipiellen Herangehensweisen an die Vererbung, die unterstützt werden. Sie zwingt zudem zu gründlichen Mitdenken, da einige Konstruktionen erst nach ausführlichem Nachdenken erfaßt werden (versuchen Sie es selbst: Welche Operationen kann der **L**-förmige Stein noch in der linken unteren Ecke des Spielfelds ausführen?). Derart vorbereitet kann sich der Leser mit der semantischen Basis in Kapitel 2 befassen. Hier werden insbesondere Fragen zur Objekt-Identität und zur Modularität von Klassen diskutiert, das formale Niveau hält sich in Grenzen. Kapitel 3 stellt die syntaktischen Konstruktionen zusammen. Es wird also etwa gezeigt, woraus sich Klassen zusammensetzen, welche Operationen (einschließlich Rekursion) zur Verfügung stehen, und welche Operationen Klassen zu neuen Klassen zu kombinieren gestatten. Sprachdefinition und konkrete Syntax sind in den Kapiteln 4 und 6 zu finden, das Kapitel 5 ist nebenläufigen Systemen gewidmet, wobei Kompositionsoperatoren und der nichtdeterministischen Auswahl naturgemäß breiter Raum gegeben wird. Auch hier erläutert eine Fallstudie — ein einfaches Kartenspiel — die Konzepte recht anschaulich.

Das Buch eignet sich als erste Einführung in Objekt-Z, wenn man bereits über Grundkenntnisse der Sprache Z verfügt, und wenn man den abschreckend hohen Preis zu zahlen gewillt ist (der Verlag kalkuliert offenbar seine Preise für Kunden wie Bill Gates). In der Arbeitsgruppe von Smith wurde ein Makro-Paket für LaTeX entwickelt, mit dessen Hilfe die leicht involvierte Notation einfacher beherrschbar wird. Die so entwickelten Spezifikationen können auf ihre syntaktische Konsistenz und die korrekte Typverwendung durch ein Paket namens `wizard` überpüft werden, das frei verfügbar ist. Der `wizard` ist recht hilfreich bei der syntaktischen Prüfung, ich hatte jedoch Probleme bei rekursiven Spezifikationen, die nicht akzeptiert werden.

Insgesamt hat man mit Objekt-Z einen interessanten und vielversprechenden Zugang zu formalen Spezifikationen. Dieses Buch hilft dabei, ihn darzustellen.

Ernst–Erich Doberkat, Lehrstuhl für Software-Technologie, Universität Dortmund; E-mail: doberkat@acm.org