

Mit Isis und Usus die Softwarequalität messen und Projekte steuern

Hotspots im Spaghetticode

Softwaremetriken sind keine akademische Spielerei, sondern ein nützliches Werkzeug für den Projektalltag. Mit ihrem Tool Isis überzeugt die andrena objects ag beispielsweise ihre Kunden davon, dass Refaktorisierungen nötig sind. Und mit dem Werkzeug Usus finden die Entwickler diejenigen Hotspots im Code, die besonders dringend überarbeitet werden müssen.

Auf einen Blick



Matthias Lohrer ist freiberuflicher IT-Fachjournalist. Sie erreichen ihn über www.mlohrer.de.

Inhalt

- ▶ Mit Isis die Entwicklung der Softwarequalität über die gesamte Projektlaufzeit hinweg messen und protokollieren.
- ▶ Mit Usus diejenigen Stellen im Code finden, die besonders dringend überarbeitet werden müssen.

Grundlagen

- ▶ Metriken für Softwarequalität, www.dotnetpro.de/A0511Metriken

dnpCode

A1009Isis

Der zweiwöchige Scrum-Sprint bei der andrena objects ag geht zu Ende. Das Team trifft sich zum Sprint Review und präsentiert dem Product Owner, was es in den zurückliegenden beiden Wochen geschafft hat. Es hat mehrere Komponenten erstellt und zum Laufen bekommen. Eine wesentliche Rolle bei diesem Review spielt ein Softwarewerkzeug namens Isis. Mit ihm dokumentiert das Team nicht nur, was es gemacht hat, sondern auch, wie gut die Qualität der entstandenen Software ist. Ein Entwickler weist auf eine Darstellung ähnlich der in Abbildung 1 hin und resümiert: „Die Qualität der Software steigt seit Projektbeginn kontinuierlich an. Mittlerweile haben wir einen Qualitätsindex von 50 Prozent erreicht. Unser Ziel ist es, ungefähr 70 Prozent zu erreichen.“

Das Entwicklerteam führt eine umfangreiche Migration durch. Zunächst hat es eine alte Delphi-Software mithilfe von Softwarewerkzeugen automatisch auf die .NET-Plattform migriert. Zu diesem Zeitpunkt hatte diese generierte .NET-Software noch eine sehr geringe Qualität. Seitdem führen die Entwickler zahlreiche Refaktorisierungen durch, um eine moderne, gut wartbare Architektur zu erstellen. Anhand der Isis-Grafiken erkennt der Kunde, dass diese Arbeit sich rentiert – er will schließlich am Ende eine flexible Software haben, die sich leicht an neue Anforderungen anpassen lässt. Das gelingt jedoch nur, wenn die Codequalität stimmt.

Die Softwarequalität messen

Isis, so könnte man sagen, ist die pragmatische Anwendung von Softwaremetriken für die Steuerung von Softwareprojekten [1]. Dabei erfindet Isis das Rad nicht neu, sondern bindet im Wesentlichen vorhandene Tools zu einem nützlichen Werkzeug zusammen. So arbeitet Isis:

- Am Ende jedes Sprints führen die Entwickler eine statische Codeanalyse durch und messen eine Handvoll Metriken. Isis speichert die gemessenen Werte in einem Projekt-Logbuch.
- Einige andere Werte – wie zum Beispiel die Testabdeckung und die Anzahl der Compilerwarnungen – tragen die Entwickler manuell in Isis ein.

Tabelle 1

So stellt Isis den Softwarequalitätsindex zusammen.

Gemessener Wert	Gewichtung im Softwarequalitätsindex
Compilerwarnungen	2 %
Methoden mit mehr als 15 Lines of Code	6 %
Klassen mit mehr als 20 Methoden	6 %
Average Component Dependency	9 %
Zyklomatische Komplexität	10 %
Packages in Zyklen	11 %
Mittlere Testabdeckung	13 %
Schätzabweichung	11 %
Anzahl Programmierfehler	15 %
Kundenzufriedenheit	17 %
Summe	100 %

- Über eine grafische Schnittstelle können die Isis-Anwender die Entwicklung der Softwarequalität über die gesamte Projektlaufzeit nachverfolgen.

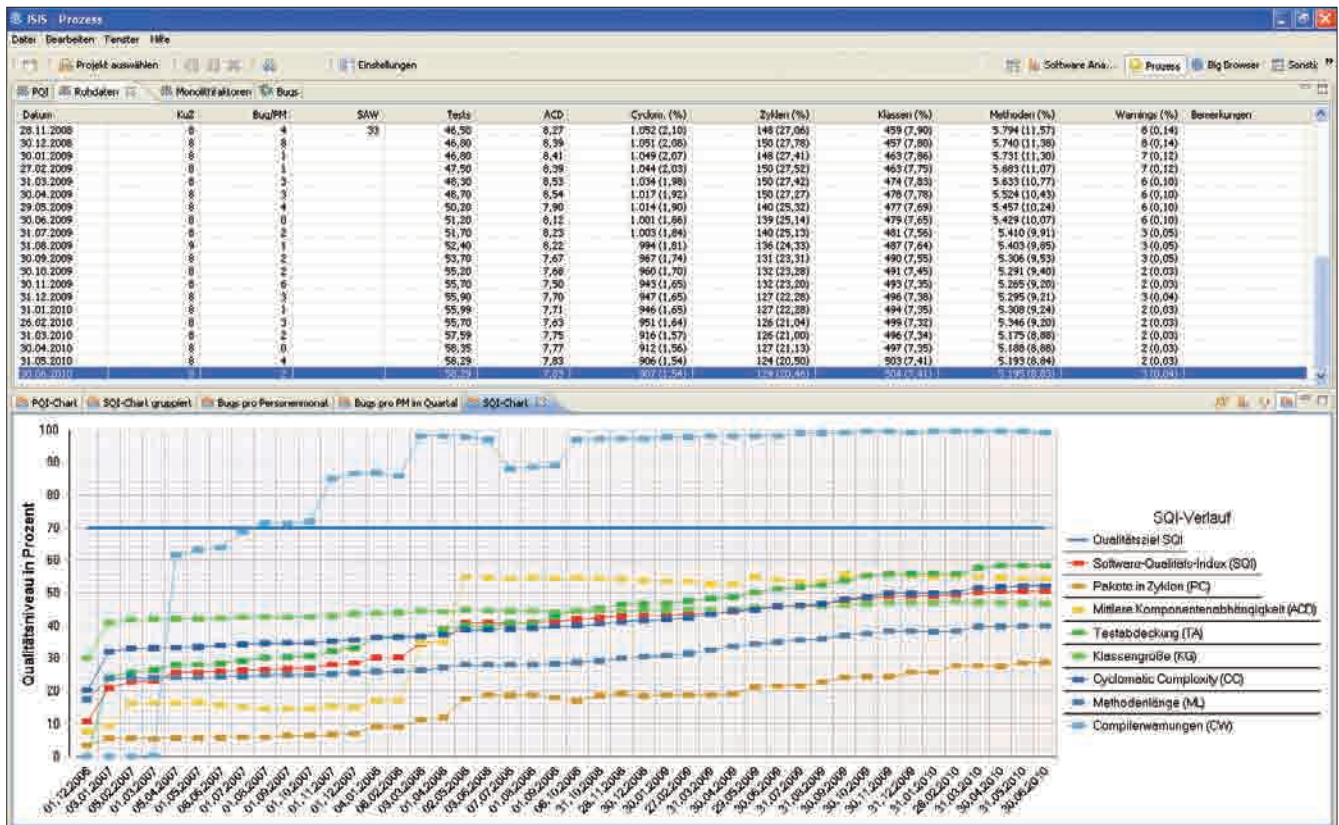
Isis misst einen Softwarequalitätsindex (SQI) und einen Prozessqualitätsindex (PQI). Der SQI bestimmt die Qualität des Codes. Der PQI bestimmt die Qualität des Entwicklungsprozesses.

Der Softwarequalitätsindex basiert auf folgenden Messwerten:

- zyklomatische Komplexität,
- Methodenlänge,
- Klassengröße,
- Testabdeckung,
- Zahl der Compilerwarnungen,
- Average Component Dependency,
- Package-Zyklen.

Diese Werte zeigt Isis in einer Balkengrafik an, siehe Abbildung 2.

- Der Prozessqualitätsindex basiert auf
- den Abweichungen zwischen dem geschätzten Entwicklungsaufwand und dem tatsächlichen Entwicklungsaufwand,
 - der Anzahl der Bugs, die es bis in das Release „geschafft“ haben,
 - der Kundenzufriedenheit.



[Abb. 1] Isis misst die Softwarequalität über die gesamte Projektlaufzeit.

Diese Werte müssen manuell eingetragen werden. Für die Messung der Kundenzufriedenheit führt andrena eine Befragung beim Kunden durch.

Ein Analysewerkzeug kann die genannten Werte des SQI automatisiert messen. Etwas kann es zunächst noch nicht: Es kann die gemessenen Werte nicht automatisch als mehr oder weniger gut beurteilen. Eine solche Beurteilung muss im Prinzip künstlich hinzugefügt werden, und hier ist ein gewisser Spielraum vorhanden. andrena hat die entsprechenden Werte auf heuristischem Wege ermittelt, das heißt, die Entwickler haben aufgrund ihrer Erfahrung solche Werte bestimmt, die sie selbst für optimal halten.

Im Einzelnen ergeben sich hier folgende Werte:

- **Zyklomatische Komplexität:** Die zyklomatische Komplexität [2] misst die Anzahl der möglichen Verzweigungen in einem Programmabschnitt. Je komplexer eine Methode ist, desto höher wird der gemessene Wert, und desto schwieriger ist dieses Konstrukt für einen Menschen zu verstehen. Für Isis liegt der Grenzwert bei einer Komplexität von 5 pro Methode.
- **Methodenlänge:** andrena favorisiert sehr kurze Methoden bis zu einer Länge von 5 bis 10 Zeilen. Methoden ab einer Länge

von 15 Zeilen erhalten eine schlechtere Bewertung. Folgende Überlegungen spielen hierbei eine Rolle: Eine Methode sollte komplett auf eine Bildschirmseite passen. Denn wenn ein Entwickler etwas auf einen Blick erfassen kann, muss er keine zusätzliche Gedächtnisleistung für diejenigen Teile erbringen, die gerade nicht sichtbar sind. Das vermeidet Fehler, und genau darum geht es.

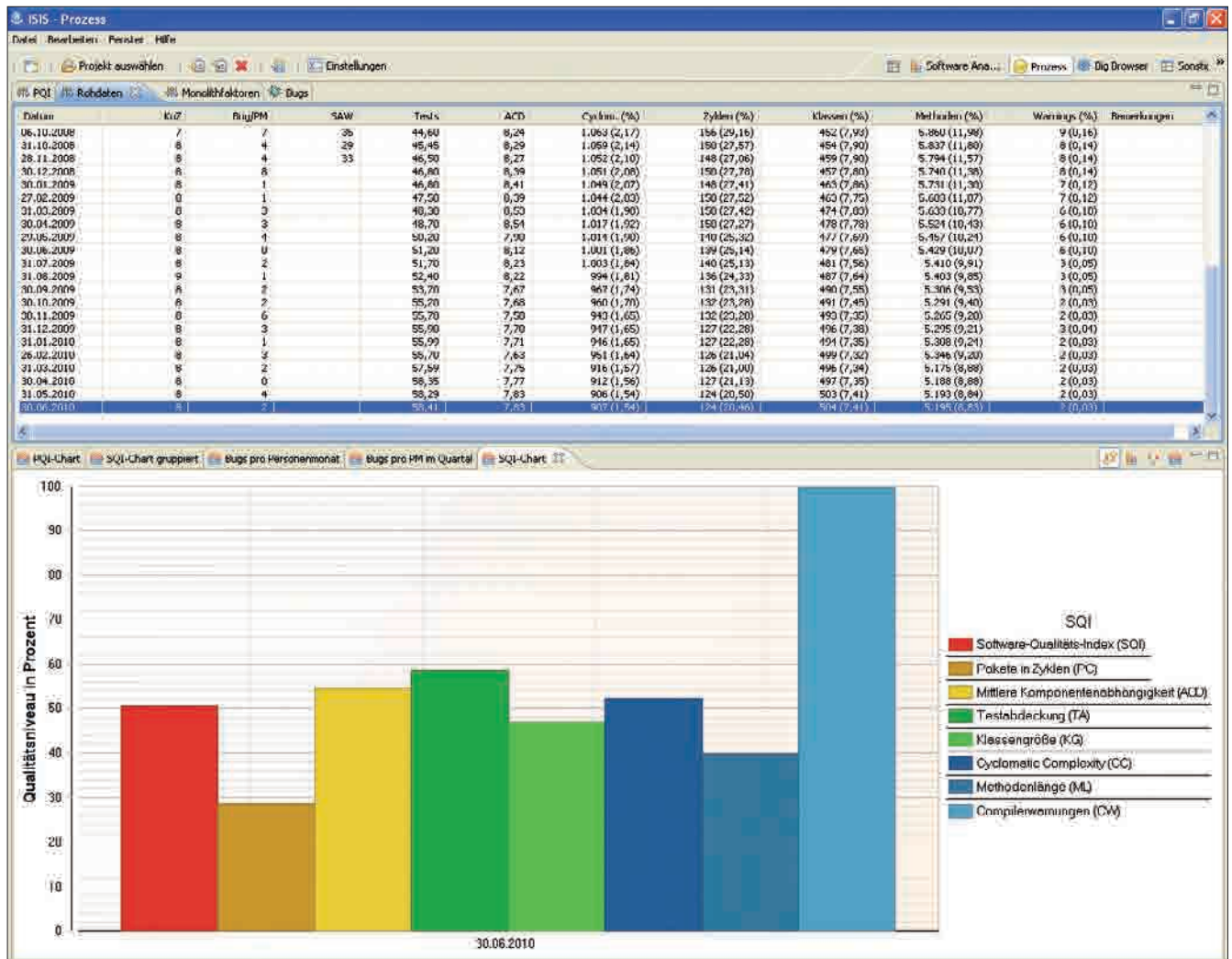
- **Klassengröße:** Für Klassen wurde festgelegt, dass sie maximal 20 Methoden enthalten sollen, um als gut zu gelten.
- **Testabdeckung:** Die Entwickler übertragen den prozentualen Wert für die Testabdeckung manuell.
- **Zahl der Compilerwarnungen:** Auch diese Zahl wird manuell übertragen.
- **Average Component Dependency:** Die Average Component Dependency misst, von wie vielen anderen Klassen eine Klasse in einem Verbund von Klassen durchschnittlich abhängt. Die Überlegung dahinter ist: Eine Klasse ist schwierig zu verstehen und schwierig zu ändern, wenn sie von vielen anderen Klassen abhängt. Ein niedriger Wert erleichtert die Wartbarkeit. Abbildung 3 zeigt ein Beispiel, wie für einen Verbund von drei Klassen die Average Component Dependency ermittelt wird.

Die Klasse a kennt die Klassen a, b und c. Die Klasse b kennt außer sich selbst nur noch c. Die Klasse c kennt nur sich selbst. Für jede Klasse nimmt man die Zahl der bekannten Klassen und teilt sie durch die Summe aller Klassen. Damit ergibt sich die Rechnung $3/3 + 2/3 + 1/3 = 6/3 = 2$, womit man die Average Component Dependency ermittelt hat. Jede Klasse des Klassenverbundes kennt im Durchschnitt 2 Klassen.

- **Package-Zyklen:** Der Ausdruck Package kommt aus der Java-Welt. Dort entspricht ein Package einem Namespace bei .NET. Wenn Packages/Namespace gegenseitig auf sich verweisen, dann sind die Packages wahrscheinlich falsch geschnitten worden. Ein Verweis von einem Package auf ein anderes Package ist ok. Ein Rückverweis auf das erste Package führt jedoch zur Abwertung, auch dann, wenn der Rückverweis über mehrere Packages führt. Dann gibt es für jedes beteiligte Package einen Minuspunkt.

Tabelle 1 zeigt, welcher Messwert mit welcher Gewichtung in den Softwarequalitätsindex einght.

Vom Aufbau her besteht Isis aus einem GUI plus einem Datenspeicher für die gemessenen Werte. Die eigentliche Code-



[Abb. 2] Die einzelnen Werte des Softwarequalitätsindex.

analyse führen darunterliegende Werkzeuge durch.

Je nach Plattform und Einsatzzweck kommen verschiedene Werkzeuge zum Einsatz:

- Die statische Codeanalyse von Java-Code führt das kommerzielle Tool Sotograph durch.
- Für .NET-Code kommt NDepend zum Einsatz.
- Für Forschungszwecke können noch andere Tools verwendet werden. Für ein Forschungsprojekt vom Forschungszentrum Informatik in Karlsruhe wurde beispielsweise das Tool SISSY eingesetzt.
- Die Freeware EclEmma misst die mittlere Testabdeckung und identifiziert lokale Defizite auf Klassen- und Methodenebene.
- Eigenentwicklungen von andrena ermöglichen die Erfassung und Historisierung von Programmierfehlern sowie die Integration, Verdichtung, Visualisierung und Historisierung der erfassten Messwerte.

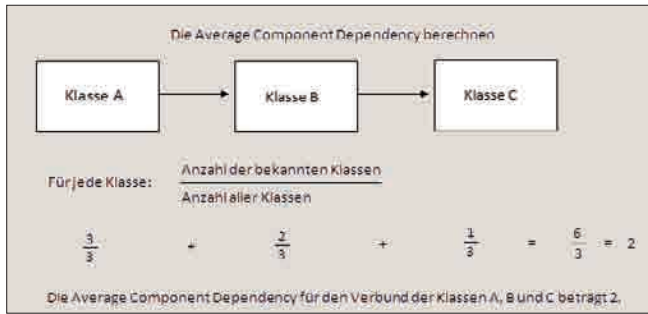
Projektleiter Frank Adler betont allerdings, dass man den Messwerten nicht blind vertrauen darf, sondern immer auch selbst einen Blick in den Code werfen muss, etwa wenn man ein Code Review für ein externes Unternehmen durchführt. Auf diese Weise ist etwa der Monolith-Faktor entstanden. Adler berichtet: „Bei einem Code Review haben wir die bestehende waren recht gut. Beim genaueren Blick in den Code stellte sich aber Folgendes heraus: Es gab Hunderte von ziemlich kleinen Klassen und eine monströse Klasse, in der die gesamte Programmlogik steckte. Wir haben daraufhin in Isis einen Monolith-Faktor ergänzt, der solche Konstellationen erkennt und entsprechend abwertet.“

Usus

Isis dient vorwiegend der Kommunikation mit den Kunden. Die Messwerte sind aber auch für Entwickler interessant. Denn diese müssen wissen, an welchen Stellen sie

etwa bei Refaktorisierungen am besten ansetzen. Für diesen Zweck nutzen die andrena-Entwickler das Werkzeug Usus. Usus misst den Softwarequalitätsindex anhand der gleichen Metriken, die auch Isis verwendet. Dabei nutzt Usus allerdings nicht die Tools von Drittanbietern, sondern verfügt über seine eigenen Routinen. Usus ist ein Eclipse-Plug-in, das andrena für Java-Projekte nutzt [3]. Der Code ist über Google Project Hosting frei verfügbar [4]. Außerdem befindet sich ein Add-in für Visual Studio in Vorbereitung.

Leif Frenzel, Senior Softwareentwickler bei andrena, erläutert die Benutzung von Usus: „Zunächst selektiert der Anwender in der View *Covered Projects* diejenigen Workspace-Projekte, die er mit Usus überwachen will. Wenn er dann das Usus-Cockpit aufruft, sieht er die Angaben zur Softwarequalität. Dazu gehören etwa die Anzahl der Methoden, die bestimmte Schwellwerte bezüglich der Zahl der Statements oder der zyklomatischen Komplexität übersteigen.“



[Abb. 3] Die Average Component Dependency für einen Klassenverbund berechnen.

Bis hierhin entspricht die Funktionalität von Usus derjenigen von Isis. Im nächsten Schritt geht Usus aber weiter und bietet dem Entwickler zusätzlichen Nutzen. Dazu Frenzel: „Wenn man eines der Elemente, die im Cockpit angezeigt werden, doppelt anklickt, gelangt man zu den sogenannten Hotspots. Das ist eine Liste der schwerwiegendsten Überschreitungen. Ein weiterer Mausklick führt den Entwickler direkt zum Code im Editor.“

Das Anzeigen von Hotspots funktioniert nicht nur mit den einfachen Metriken wie der Länge der Methoden, sondern etwa auch mit den komplexeren Abhängigkeiten zwischen Namensräumen und Klassen untereinander. Dazu ein Beispiel: Wenn eine Klasse über Bezüge zu mehreren eingehenden und mehreren ausgehenden Klassen verfügt, ist es ab einer bestimmten Grenze sinnvoll, an dieser Stelle eine Interface-Schicht einzuziehen. Usus multipliziert die Anzahl der eingehenden Klassen mit der Anzahl der ausgehenden Klassen. Wenn dieser Wert einen vorher definierten Wert überschreitet, wird das als schlechte Qualität erkannt.

Auf diese Weise kann der Entwickler mithilfe von Usus die sogenannten Bottleneck-Klassen finden und durch spezielle Veränderungen die Softwarequalität gezielt verbessern.

Fazit

„Spaghetticode“ sagte man früher abfällig zu Basic-Programmen, deren Programmlogik wegen vieler Goto-Sprünge schwer durchschaubar war. Java- und .NET-Programmierer wenden sich hier mit Grausen ab und glauben, sie seien etwas Besseres – sie entwickeln schließlich objektorientiert. Leider ist es aber kein Privileg der Basic-Programmierer, kaum entwirrbaren Code zu schreiben.

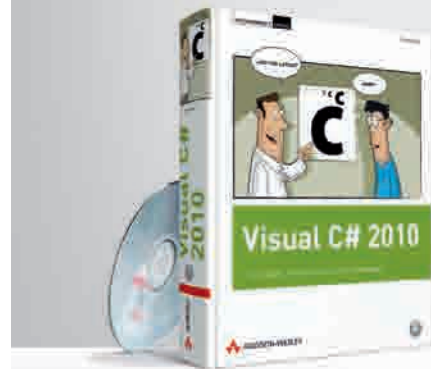
Im Dickicht ausufernder Klassenhierarchien und verschachtelter Packages kann man sich genauso verirren, wenn man keine klaren Entwurfsrichtlinien einhält. Wer solchen Code refaktorisieren muss, der ist für jede maschinelle Unterstützung dankbar. Genau hier setzen Isis und Usus an. Usus lenkt den Entwickler zu solchen Stellen, an denen Refaktorisierungen besonders dringend nötig sind. Isis misst den Fortschritt der Softwarequalität über die gesamte Projektlaufzeit hinweg. So gelingt es mit beiden Werkzeugen, die Qualität der Software zu quantifizieren und die Projektarbeit aufgrund der gemessenen Werte zu steuern. [ml]

[1] www.andrena.de/know-how/isis und www.andrena.de/know-how/usus

[2] Holger Gubbels, Die Quantität der Qualität, Metriken für Softwarequalität, dotnetpro 11/2005, Seite 110ff., www.dotnetpro.de/A0511Metriken

[3] Leif Frenzel, Nicole Rauch und Stefan Schürle, Das ist hier Usus..., Unmittelbares Feedback mit den Plug-ins von projectusus.org, eclipse magazin 2/2010, Seite 89

[4] projectusus.org



Frank Eller bietet Ihnen umfassende Informationen über die Anwendungsentwicklung mit .NET 4.0, C# und WPF. Neben grundlegenden Themen wie Sprachgrundlagen, Fehlerbehandlung, Dateihandling und Multithreading erhalten sie außerdem umfangreiches Wissen darüber, wie Sie Ihre Anwendungen besser strukturieren und die Oberfläche nahezu vollständig vom Code trennen können. Neueste Technologien wie das Entity Framework, LINQ oder LINQ to SQL werden ebenso detailliert erläutert wie der Umgang mit der Windows Presentation Foundation.

ISBN 978-3-8273-2916-5
1200 Seiten, 1 DVD
€ 49,80 [D]
<http://www.awl.de/2916>

Dirk Louis liefert Ihnen eine umfassende Einführung in die Programmierung mit Visual C++ 2010. Im ersten Teil werden die wichtigsten Sprachgrundlagen behandelt. Weiter geht es im zweiten Teil zu den notwendigen Erweiterungen der Sprache für die Zusammenarbeit mit dem .NET Framework. Anschließend sind Sie bereit, in die Programmierung von grafischen Oberflächen einzusteigen, erstellen Steuerelemente, Menüleisten, Grafiken und vieles mehr. Der letzte Teil ist speziellen Programmieretechniken wie z.B. der Anbindung an Datenbanken, der Programmierung mit Threads oder der Verarbeitung von XML-Daten gewidmet.



ISBN 978-3-8273-2901-1
1024 Seiten, 1 DVD
€ 49,80 [D]
<http://www.awl.de/2901>



Jürgen Bayer bietet Ihnen in seinem Codebook fertige Lösungen zu praxisbezogenen, täglich auftretenden Programmierproblemen. Alle diese »Rezepte« sind in Kategorien sortiert und somit sehr leicht auffindbar. Die CD zum Buch enthält nochmals alle vorgestellten Codes, so dass diese sehr schnell und effizient in eigene Projekte eingefügt werden können.

ISBN 978-3-8273-2903-5
1280 Seiten, 1 CD
€ 89,80 [D]
<http://www.awl.de/2903>

TIPP

Unser Addison-Wesley-Blog mit vielen Informationen, Interviews und News aus der IT-Szene auf <http://blog.addison-wesley.de>



[The Sign of Excellence]
ADDISON-WESLEY