▶ chefredakteur

„Wir sind gerade dabei, das zu refaktorieren", gehört zu den falschen Sätzen, die man in Teams derzeit am häufigsten hört. Was ist daran falsch? Zum einen bezieht sich das englische Wort „Refactoring" auf die Umformung mathematischer Gleichungen, zu Deutsch „Herausfaktorisieren". Die korrekte Übersetzung für „Refactoring" lautet daher auch „Refaktorisierung". Zum anderen verwenden die meisten Teams den Begriff nur als modern klingenden Anglizismus für „Umbau", ohne wirklich die Ideen des Refaktorisierens zu beachten – was freilich der sehr viel folgenschwerere Fehler ist.

Urheber des Begriffs „Refactoring" ist **Bill Opdyke**, der Anfang der 90er Jahre bei **Ralph Johnson**, einem der Autoren des „Design Patterns"-Buchs, über das „Refaktorisieren objektorientierter Frameworks" promovierte. Die Idee war einfach: Jeder Schüler kennt aus der Mathematik Gleichungsumformungen, also Änderungen an Gleichungen, von denen bewiesen ist, dass sie die Aussage der Gleichung nicht verändern. Opdykes wollte solche Umformungen auch für Code sammeln.

Einige dieser Umformungen sind naheliegend. Ändert man zum Beispiel den Namen einer Variablen an der Definition und allen ihren Verwendungen, hat das keine Auswirkung auf die Funktionalität. Allerdings ist selbst diese Änderung alles andere als trivial: Vererbungs- und Verschattungsregeln müssen ebenso beachtet werden wie Aufrufe über *Reflection*.

Was genau zu beachten ist, legt die „formale Semantik" der Programmiersprache fest, die üblicherweise aus einem Wust mathematischer Formeln besteht. Um einen einzelnen Refaktorisierungsschritt sauber zu beweisen, benötigt man oft 10 oder 20 Seiten eines schwer verdaulichen Formelwerks – nicht gerade der Stoff, aus dem Hypes gemacht sind.

Es dauerte entsprechend auch, bis eine breitere Öffentlichkeit Notiz von den Überlegungen nahm. Zunächst einmal verlangten die komplizierten Regeln nach Werkzeugunterstützung. **Don Roberts** und **John Brant**, zwei weitere Doktoranden von Ralph Johnson, bauten noch in den 90er Jahren den ersten Refaktorisierungs-Browser für Smalltalk, der unter anderem vom CCC-Team um **Kent Beck** eingesetzt wurde und damit seinen Weg ins Extreme Programming nahm.

Ende der 90er Jahre trugen dann zwei wesentliche Ereignisse zum breiten Einsatz von Refaktorisierung bei: Zum einen wurde Eclipse mit Refaktorisierungsfunktionen ausgestattet, zum anderen veröffentlichte **Martin Fowler** mit seinem Buch „Refactoring" die erste auch für Nicht-Akademiker lesbare Sammlung von Codeumbauten. Als dann auch noch XP mit seinen Ansätzen von testgetriebenem Design, das ja unter anderem ständiges Refaktorisieren als Basis hat, Furore machte, setzte sich die Technik zunächst im agilen Bereich durch. Mittlerweile wird es verfahrensunabhängig auf breiter Basis eingesetzt, aber oft noch ohne Verständnis für die Hintergründe und Grenzen. Deshalb widmen wir diese Ausgabe von OBJEKTspektrum dem Thema „Refaktorisierung".

In dem ersten Schwerpunkt-Beitrag „Refaktorisierungswerkzeuge: Ein Blick hinter die Kulissen" geben **Thomas Corbat, Peter Sommerlad** und **Mirko Stocker** einen Überblick, wie Refaktorisierungswerkzeuge eigentlich funktionieren und mit welchen Problemen sie zu kämpfen haben.

Ein sehr häufiges Problem behandelt **Michael Klenk** in seinem Artikel „Sprachübergreifendes Refaktorisieren": ein interessantes Forschungsgebiet mit hoher Praxisrelevanz, das derzeit mehr Fragen als Antworten aufwirft.

Friedrich Steimann plädiert in seinem Beitrag „Korrekte Refaktorisierungen" dafür, Codetransformationen als eigene Disziplin der Informatik zu etablieren, ähnlich dem Compilerbau.

Der Praxisteil geht dann stärker auf den Einsatz von Refaktorisieren in der praktischen Projektarbeit ein.

Mit der Suche nach lohnenden Refaktorisierungsobjekten in altem Code beschäftigt sich **Thomas Haug** in dem Artikel „Kennzahlenbasierte Beseitigung von ‚Code Smells'".

Dass nicht nur Produktivcode gepflegt werden muss, sondern auch Testcode, unterstreicht **Markus Tiede** in seinem Erfahrungsbericht „Refaktorisieren von Tests".

Über die Entflechtung eines kompletten Altsystems berichtet **Mark Paluch** in seinem Beitrag „Code-Recycling".

Und schließlich behandeln **Jan Ortmann** und **Alexander Weickmann** ein Problem, mit dem jeder konfrontiert ist, der modellgetriebene und agile Ansätze kombinieren möchte: „Refaktorisieren in der modellgetriebenen Entwicklung".

Ich wünsche Ihnen viel Spaß beim Lesen.

Jens Coldewey