Vorwort

Hast du schon einmal ein Buch über Software gelesen und das Gefühl bekommen, dass der Autor oder die Autorin an dir vorbeiredet? Wurde das Wissen in unbekannte Fachwörter und übermäßig komplexe Konzepte verpackt? Hast du dich beim Lesen gefühlt, als sei das Buch für eine kleine Elite von Besserwissern geschrieben, der du nicht angehörst?

Das ist nicht dieses Buch. Dieses Buch ist bodenständig, fokussiert und auf den Punkt gebracht.

Dieses Buch ist aber auch keine Einführung. Es fängt nicht bei Adam und Eva an und erklärt die Grundlagen der Programmierung. Es versucht nicht, dich zu verhätscheln und von Gefahren fernzuhalten. Ich garantiere, dass dich dieses Buch herausfordern wird, aber dabei wird es dich weder einschüchtern noch deine Intelligenz beleidigen.

Refactoring ist das Fachgebiet mit der Aufgabe, schlechten Code in guten Code zu verwandeln, ohne dabei Fehler einzubauen. Wenn man bedenkt, dass unsere gesamte Zivilisation von Software zusammengehalten wird, dürfte es kein wichtigeres Thema geben.

Klingt »dass unsere gesamte Zivilisation von Software zusammengehalten wird« übertrieben? Ist es nicht. Schau dich um. Wie viele Prozessoren, die Software ausführen, trägst du gerade mir dir herum? Deine Uhr, dein Telefon, deinen Autoschlüssel, deinen Kopfhörer … Wie viele findest du in dreißig Metern Umkreis? Die Mikrowelle, den Herd, die Spülmaschine, den Thermostat, die Waschmaschine … Und wie viele sind es in deinem Auto?

Nichts geht mehr ohne Software. Du kannst ohne Software nichts kaufen oder verkaufen, nicht fahren, nicht fliegen, kein Würstchen kochen, nicht fernsehen, niemanden anrufen.

Und wie viel dieser ganzen Software besteht aus gutem Code? Denk über die Systeme nach, an denen du selbst arbeitest – sind sie guter Code? Oder sind sie, wie so viele andere, ein Chaos, das geradezu nach Refactoring schreit?

In diesem Buch geht es nicht um das einfache Refactoring an sterilen Beispielen, von dem du wahrscheinlich schon gelesen oder gehört hast. Hier geht es um *echtes* Refactoring: Refactoring an *realen* Projekten, Refactoring an Legacy-Systemen, Refactoring in der Art Umgebung, mit der wir es täglich zu tun haben.

Dieses Buch wird dir keine Schuldgefühle einreden, weil du nicht für alles automatisierte Tests hast (puh, Schwein gehabt! *Anmerkung des Übersetzers*). Dem Autor ist klar, dass die meisten *echten* Systeme über eine lange Zeit gewachsen sind und sich entwickelt haben. Wir haben selten das Glück, dass wir eine komplette Testsuite zusammen mit der Software geerbt haben.

Dieses Buch gibt dir einfache Regeln an die Hand, die du anwenden kannst, um an komplexen, unsauberen, verworrenen und ungetesteten Systemen zuverlässig Refactorings vorzunehmen. Indem du diese Regeln lernst und befolgst, kannst du in den Systemen, die du wartest, einen echten Qualitätsunterschied bewirken.

Versteh mich nicht falsch – Refactoring ist keine Wunderwaffe. Alten, schmutzigen, ungetesteten Code mit Refactorings zu verschönern ist nie einfach. Aber mit den Regeln und Beispielen aus diesem Buch hast du die Waffen in der Hand, mit denen du den schrottigen Systemen, die deine Arbeit zur Hölle machen, zu Leibe rücken kannst.

Ich rate deshalb, dieses Buch aufmerksam zu lesen. Schau die Beispiele genau an. Denk scharf über die Absichten und Abstraktionen nach, die der Autor vorstellt. Hol dir die Codebasis, die er anbietet, und führe das Refactoring mit ihm gemeinsam durch. Begleite ihn auf dieser Refactoring-Reise vom ersten bis zum letzten Schritt.

Es wird dauern. Es wird dich entmutigen. Es wird dich fordern. Aber am Ende wirst du Fähigkeiten erlernt haben, die dir für den Rest deiner Karriere helfen werden. Und du wirst ein intuitives Verständnis erlangt haben, was guten Code von schlechtem unterscheidet.

Robert C. Martin (»Uncle Bob«)