Vorwort

Als fachlicher Lektor hatte ich frühzeitig die Gelegenheit, das Buch zu lesen, das Sie jetzt in Händen halten. Schon in der Rohfassung war es großartig. David Thomas und Andrew Hunt haben etwas zu sagen, und sie wissen, wie man es sagen muss. Ich hatte erkannt, was sie taten, und wusste, dass es funktionieren würde. Ich bat sie, dieses Vorwort schreiben zu dürfen, um zu erklären, wieso.

Einfach gesagt, zeigt Ihnen dieses Buch auf nachvollziehbare Weise, wie man programmiert. Sie werden denken, dass das nicht schwierig sei, aber das ist es. Wieso? Einerseits werden nicht alle Programmierbücher von Programmierern geschrieben. Viele werden von den Erfindern neuer Programmiersprachen oder von Journalisten verfasst, mit denen sie zusammenarbeiten, um für ihre Arbeit zu werben. Diese Bücher bringen Ihnen bei, in einer Programmiersprache zu *reden*. Das ist sicherlich wichtig, aber nur ein kleiner Teil dessen, was ein Programmierer tut.

Was tut ein Programmierer, außer in einer Programmiersprache zu reden? Das ist eine tiefgründige Frage. Den meisten Programmierern fällt es schwer, ihre Arbeit zu beschreiben. Programmieren ist eine Aufgabe mit vielen Details und sie alle im Blick zu behalten, erfordert Konzentration. In stundenlanger Arbeit entsteht der Quelltext. Man schaut auf und sieht all diese Anweisungen. Wenn man nicht sorgfältig darüber nachdenkt, könnte man denken, Programmieren bestehe lediglich daraus, Anweisungen in einer Programmiersprache einzutippen. Das ist natürlich falsch, aber darauf kommen Sie nicht, wenn Sie sich in einer Buchhandlung bei den Programmierbüchern umsehen.

In *Der Pragmatische Programmierer* erklären uns David und Andrew, wie man programmiert, und das auf verständliche Weise. Wie sind die so schlau geworden? Sind sie nicht genauso auf Details konzentriert wie andere Programmierer? Die Antwort ist: Sie haben darauf geachtet, was sie tun, und dann versucht, es besser zu machen.

Stellen Sie sich vor, Sie sitzen in einer Besprechung. Vielleicht denken Sie, dass diese Besprechung noch ewig so weiter geht und Sie viel lieber programmieren würden. David und Andrew würden darüber nachdenken, wofür diese Besprechung einberufen wurde und ob sie durch etwas anderes ersetzt werden könnte. Sie würden sich überlegen, ob dieses Etwas automatisiert werden kann, damit sich die Aufgabe dieser Besprechung in Zukunft von selbst erledigt. Anschließend würden sie genau dafür sorgen.

Das ist die Art, wie David und Andrew denken. Die Besprechung war nicht etwas, das sie vom Programmieren abgehalten hat, sondern es *war* Programmieren. Und es war Programmieren, das noch verbessert werden kann. Ich weiß, dass sie so denken, weil ihr zweiter Tipp lautet: *Denken Sie über Ihre Arbeit nach*.

Stellen Sie sich also vor, wie diese beiden ein paar Jahre lang auf diese Weise nachdenken. Ziemlich bald haben sie eine Sammlung von Lösungen, die sie noch ein paar Jahre lang bei ihrer Arbeit anwenden oder verwerfen, wenn sie sich als zu schwierig herausstellen oder nicht immer zum gewünschten Ergebnis führen. Dieses Vorgehen ist ziemlich genau das, was man als *pragmatisch* bezeichnet. Malen Sie sich aus, wie sich die beiden ein oder zwei Jahre Zeit nehmen, um ihre Lösungen aufzuschreiben. Sie werden denken: *Diese Erkenntnisse müssen eine Goldgrube sein.* Und Sie haben Recht.

Die Autoren erklären uns, wie sie programmieren, und sie erklären es auf eine Weise, die wir nachvollziehen können. In der zweiten Aussage liegt mehr, als Sie vielleicht denken. Lassen Sie es mich erklären.

Die Autoren haben es sorgfältig vermieden, eine Theorie der Software-Entwicklung aufzustellen. Zum Glück, denn wenn sie es getan hätten, hätten sie jedes Kapitel verzerren müssen, um ihre Theorie zu verteidigen. Diese Art von Verzerrung ist Tradition in, sagen wir, Naturwissenschaften, wo Theorien irgendwann Gesetze werden oder still und leise verworfen werden. Im Gegensatz dazu hat das Programmieren (wenn überhaupt) nur wenige Gesetze. Programmiertipps, die in Möchtegern-Gesetze verpackt sind, machen sich vielleicht gedruckt gut, aber helfen einem in der Praxis nicht weiter. Darin liegt der Fehler vieler Bücher über Vorgehensmodelle.

Ich habe mich mit diesem Problem schon ein Dutzend Jahre lang beschäftigt und habe die besten Erfolgsaussichten bei einem Mittel namens *Mustersprache* (engl. Pattern Language) gefunden. Kurz gesagt, ist ein *Muster* eine Lösung und eine Mustersprache ist ein System von Lösungen, die sich gegenseitig stützen. Inzwischen beschäftigt sich eine ganze Gemeinde von Fachleuten mit der Suche nach diesen Systemen.

Dieses Buch ist mehr als eine Sammlung von Tipps. Es ist eine Mustersprache im Schafspelz. Ich sage das, weil jeder dieser Tipps aus Erfahrungen heraus entstanden ist. Jeder ist als konkreter Ratschlag formuliert und bezieht sich auf andere, so dass ein System entsteht. Das sind die Merkmale, die es uns ermöglichen, eine Mustersprache zu erlernen und sie nachzuvollziehen. Hier funktionieren sie genauso.

Sie können die Ratschläge aus diesem Buch unmittelbar anwenden, weil sie konkret sind. Diffuse Abstraktionen werden Sie nicht finden. David und Andrew wenden sich direkt an Sie, als ob jeder Tipp eine entscheidende Strategie wäre, die Ihrer Programmierkarriere neuen Schwung gibt. Sie machen es einfach, erzählen Geschichten, verwenden einen lockeren Stil und untermauern alles mit Antworten auf die Fragen, die auftauchen werden, wenn Sie es selbst ausprobieren.

Das ist noch nicht alles. Wenn Sie zehn oder fünfzehn Tipps gelesen haben, werden Sie eine weitere Dimension dieser Arbeit erkennen. Wir nennen das manchmal *die Eigenschaft ohne Namen*. Das Buch hat eine Philosophie, die in Ihr Bewusstsein einsickern und sich mit Ihrer eigenen vermischen wird. Es predigt nicht, sondern erzählt nur, was funktioniert. Beim Erzählen kommt aber mehr zum Vorschein. Das ist die Schönheit dieses Buches: Es verkörpert eine Philosophie und ist dabei ganz und gar nicht protzig.

Hier ist es also: Ein verständliches und anwendbares Buch über die gesamte Praxis des Programmierens. Ich habe immer wieder darüber geredet, wieso es funktioniert. Sie werden sich vermutlich nur dafür interessieren, dass es funktioniert. Das tut es, Sie werden sehen.

Ward Cunningham