

Sich an die Führungsrolle anpassen

Bei jeder neuen Arbeitsstelle haben wir alle große Hoffnungen und – in gewissem Umfang – auch ein bisschen Sorge, dass wir versagen könnten. Als erfolgreicher Programmierer haben Sie ohne Zweifel Ihre Erfahrungen mit dem Neubeginn in Projekten und an Arbeitsplätzen. Nachdem Ihnen die Leitung einer Gruppe von Programmierern übertragen wurde, steht Ihnen nun eine sehr neue und vielleicht auch »beeindruckende« Aufgabe bevor. Sie müssen sich so schnell wie möglich von einem Programmierer zu einem Leiter entwickeln, um mit Ihrer neuen Rolle in der Software-Entwicklung erfolgreich zu sein. Dazu gehört das Anpassen an einen neuen sozialen Kontext und das Übernehmen anderer Kommunikationsformen in Ihrer Arbeitswelt und den Personen, die dazu gehören.



Die Anpassung, eine treibende Kraft der biologischen Evolution, war sehr erfolgreich bei der Unterstützung unserer Spezies auf dem Weg aus dem Urschlamm hin zur dem, was wir jetzt sind. Es dauerte Millionen von Jahren, aber jetzt sind wir da, verwenden Sprache und beschäftigen uns mit abstrakten Konzepten wie der Computerprogrammierung. Wie sind wir dahin gelangt? Sie müssen diese Frage dem Biologen Ihres Vertrauens stellen, aber in diesem Buch werden Sie intensiv auf Ihre Fähigkeit zur Anpassung zurückgreifen müssen, um sich den Herausforderungen beim Leiten von Programmierern stellen zu können.

In diesem ersten Kapitel werden Sie so etwas wie ein Anthropologe. Sie werden auf die menschlichen Aspekte des Code-Schreibens schauen – genauer gesagt, untersuchen Sie die verschiedenen Arten von Individuen, die sich dieser wundervollen Aufgabe widmen. Indem Sie mehr über die Personen lernen, die Sie leiten, erhalten Sie Einblicke in das erfolgreiche Führen dieser Menschen. Es gibt heutzutage viele unterschiedliche Ideen, wie man Programmierer leiten sollte. Jede Generation von Leitern fängt mit seiner eigenen Perspektive an und baut darauf auf, was sie weiß und was ihrer Meinung nach als Vorgehen im Management und in der Leitung funktioniert. Ich bin aus der Generation, die mit Rechenschiebern und Lochkarten aufwuchs, und dies wird ohne Zweifel meine Präsentation beeinflus-

sen. Jedoch habe ich in all den Jahren, die ich mit Programmierern zusammengearbeitet habe, die deutlich jünger sind als ich, gelernt, dass meine Generation nicht die einzigen Methoden besitzt, die funktionieren. Ich musste mich viele Male an die geänderten Geschäftsanforderungen, technologischen Revolutionen sowie die Wandlungen und Sturheiten meines eigenen Charakters anpassen, wenn ich mich den Herausforderungen einer Leitung stellte. Ich werde diese Erfahrungen mit Ihnen teilen, und ich glaube, wir werden eine schöne Zeit miteinander verbringen.

1.1 Tragen echte Führungskräfte Schwarz?

Manche tun es. Andere wiederum zeigen stolz ihren Pferdeschwanz, um – abhängig von ihrer Haarfülle – besser zu manchen der jüngeren Geeks oder (falls Ihnen das mehr zusagt) Nerds zu passen. Sie mögen vielleicht keinen dieser Begriffe, sondern sehen sich eher als moderner Leiter, der Männer und Frauen führt, die wie Sie das Programmieren als eine intellektuelle Herausforderung sehen. Daher sollten die von mir angedeuteten äußeren »Erscheinungsformen«, einschließlich der im Titel dieses Abschnitts erwähnten, nicht allzu ernst genommen werden. Was Sie allerdings unbedingt ernst nehmen sollten, ist Ihre Fähigkeit, selbst eine Verbindung zu Ihren Programmierern aufzubauen und sich mit ihnen zu identifizieren. Nachdem Sie nun selber der Leiter sind, können Sie sich nicht einfach wie bisher als Teil der Gruppe einordnen. Zudem sollten Sie das auch gar nicht zu sehr versuchen, denn Sie sind der Chef, und ab und zu benötigen Sie diesen Vorteil, wenn es darum geht, die Entwicklung im Kurs zu halten. Irgendjemand hat einmal gesagt: »Gebt mir einen Hebel, der groß genug ist, und ich kann die Erde verschieben.« Chef zu sein kann ein solcher Hebel sein.

Sie sind vielleicht nicht davon überzeugt, dass das Aussehen unwichtig ist. Es hat lange Zeit gedauert, bis ich merkte, dass mein äußeres Erscheinungsbild nicht unbedingt ein Abbild dessen ist, was meinen Charakter ausmacht. Ich liebe immer noch die Insignien des »Nerd-Seins«, aber ich weiß auch, dass das Führen eines Teams mehr erfordert als nur Stil. Sicher, ein bestimmtes Bild kann sehr dabei helfen, Ihren Mitarbeitern klar zu machen, dass Sie einer von ihnen sind. Aber ist es eine zwingende Führungseigenschaft? Sie erinnern sich vielleicht, wie im Film *Das Netz* Angela (Sandra Bullock) von ihren Online-Bekanntem akzeptiert wurde. Alle sagten, sie sei eine von ihnen, und akzeptierten sie daher in ihrem kleinen verrückten Kreis. Natürlich zeigte sich am Ende, dass dies gar nichts so Besonderes war. Lernen Sie daraus, dass ein Bild nur oberflächlich ist. Was zählt, ist der Charakter. Das ist auch der Grund, warum all die Management-Techniken, die Sie vielleicht gelernt haben und nun versuchen, umzusetzen, so häufig fehlschlagen: Es sind Techniken, die Sie nur in Ihr Gehirn eingepflanzt haben, anstatt sie zu hegen und in Ihrem Herzen wachsen zu lassen.

Wie wichtig ist es, cool zu sein?

Um also diesen ernsthaften Tonfall beizubehalten: Sollten Sie Schwarz tragen und die Verhaltensweisen vorspielen, von denen Sie glauben, dass sie als Leiter von Programmierern Ihre Coolheit zeigen? Machen Sie den Check »Testen Sie Ihre Coolheit« im Kasten und schauen Sie, wie Sie sich schlagen. Beachten Sie, dass manche den Begriff »Ninjitsu«¹ anstelle von »cool« bevorzugen, aber ich bin halt noch einer der alten Schule. Denken Sie daran, dass dies eine Buch zur Selbsthilfe ist, daher müssen Sie schon ein wenig Arbeit investieren. Überraschende Tests gibt es nicht nur bei böswilligen alten Professoren – sie tauchen auch immer wieder in Ihrer täglichen Arbeit auf.

Testen Sie Ihre Coolheit

Wählen Sie eine oder mehrere Antworten für die folgenden Fragen.

1. Ein »Hacker« ist eine Person, die
 - a. Möbel mit einer Axt herstellt
 - b. mit Begeisterung programmiert, anstatt nur darüber zu fachsimpeln
 - c. die geistigen Herausforderungen liebt, die sich ihr beim kreativen Überwinden oder Umgehen von Begrenzungen stellen
 - d. böswillig versucht, geheime Informationen zu erhalten
 - e. ein Charakter war, der von Angelina Jolie gespielt wurde, bevor sie ein Tomb Raider wurde
2. Ein »Cracker« ist jemand, der
 - a. die Sicherheitsmaßnahmen eines Systems ausschaltet
 - b. ein weißer Südstaatler ist wie Ihr Autor
 - c. dünner als ein Cookie ist
 - d. ein Hacker im Larvenstadium ist
3. »Phreaking« ist
 - a. die Kunst und Wissenschaft, das Telefonnetz zu »cracken«
 - b. ein alter Nerd, der versucht, cool zu sein
4. »Ping« ist
 - a. ein Internet-Paket-Taster
 - b. der Klang eines Sonarpulses
 - c. die andere Hälfte von Pong
 - d. ein Quantum Fröhlichkeit
5. »Worm« bezieht sich auf
 - a. ein optisches Plattenlaufwerk, dessen Medien einmal beschrieben und beliebig häufig gelesen (write-once, read-many) werden können

¹ Sie wissen, was ein Ninja ist – dieses Wort bezieht sich auf die Fähigkeit, einer zu sein, wie bei der Programmierung auf dem Niveau eines schwarzen Gürtels.

- b. ein Virus-Programm, das die Daten im Speicher oder auf der Festplatte zerstören soll
 - c. ein zweiseitig symmetrisches, wirbelloses Tier
6. Ein »Cookie« ist
- a. ein Teil einer Vereinbarung zwischen kooperierenden Programmen
 - b. etwas, was Amos berühmt machte
 - c. ein Element, das zum Speichern und manchmal auch zum Lernen der Surf-Gewohnheiten von Benutzern verwendet wird

Nun, was denken Sie, wie Sie den Test gemeistert haben? Ich gab diese Aufgaben einmal während eines Vortrags über Computersicherheit einer Gruppe Leuten, die mit Programmieren nichts am Hut haben. Damit sollten die Arten von Personen vorgestellt werden, die hacken oder versuchen, Computer vor Hackern zu schützen. Die Teilnehmer kamen bei diesem Test nicht sehr weit, aber ich vermute, dass Sie deutlich über dem Durchschnitt lagen. Alle Antworten sind für jede Frage korrekt.² Na gut, vielleicht ist Antwort b in Frage 3 etwas fiktiv, aber dieser Test zeigt, wie Programmierer traditionell charakterisiert wurden, um zu einer bestimmten Subkultur zu gehören. Manchmal wird sie – nicht einmal abwertend – als »Hacker-Kultur« bezeichnet (siehe auch die netten Antworten zu Frage 1). Heutzutage verschwinden diese Hacker-Stereotypen. Ein Programmierer beginnt mittlerweile mit einem Informatikdiplom oder einem MBA. Trotz alledem hat jede Gemeinschaft eine Kultur und auch die Ihres Teams ist so einmalig wie die Personen, aus denen diese Gruppe besteht. Wie auch immer die Kultur definiert ist, es geht auch immer darum, dass Sie Ihre Mitarbeiter leiten und managen. Wenn Sie den Faden und den Stoff (will sagen, die Arten der Zusammenarbeit und des Denkens) der Kultur Ihrer Programmierer verstehen, können Sie sie besser verstehen und Ihre Führungstätigkeit vereinfachen. Tragen Sie also ein cooles schwarzes T-Shirt mit einem esoterischen Text, der vorne drauf prangt, wenn Sie wollen. Es gibt aber deutlich effektivere Wege, auf Ihre Mitarbeiter einzugehen, als sich nur einem stereotypen Bild anzupassen. Das ist der entscheidende Punkt in diesem Kapitel.



Hacker-Stereotypen verschwinden. Ein Programmierer beginnt heutzutage mit einem Informatikstudium oder einem MBA.

Seien Sie mehr als cool: Seien Sie auf der Hut

Klar, wenn Sie selbst ein Hardcore-Hacker sind, sind die Beziehungen zu Programmierern nicht unbedingt ein Problem. Trotzdem sollten Sie auf Folgendes achten: Gute Programmierer werden häufig ins Management befördert, sind aber nicht

² Sie können die meisten dieser Begriffe nachschlagen in Eric S. Raymond, *The New Hacker's Dictionary*, 3rd Edition, The MIT Press, 1998.

sehr häufig die besten Manager oder Leiter von Programmierern. Sie haben den starken Wunsch, selbst an den coolsten Projekten zu arbeiten, wenn Sie dies eigentlich delegieren sollten. Sie verbringen oft viele Stunden mit dem Code-Review, wenn es eine einzelne auch getan hätte, nur um jeden Kommentar und jede Einrückung perfekt zu machen. Es gibt Zeiten, in denen Sie es aufgeben, anderen zu erklären, was Sie wollen, und es einfach selber machen. Verstehen Sie mich hier nicht falsch, Sie müssen sich schon Sorge um die Details im Code machen, für den Sie verantwortlich sind. Aber der Programmierer-wird-Manager hat häufig Probleme damit, den Wald vor lauter Bäumen zu sehen.



Sie müssen sich schon Sorge um die Details im Code machen, für den Sie verantwortlich sind. Aber der Programmierer-wird-Manager hat häufig Probleme damit, den Wald vor lauter Bäumen zu sehen.

Im anderen Extrem leiten Sie vielleicht bei Tag und schreiben den Code bei Nacht, was davon abhängt, ob Sie ein Leben haben oder nicht. Vielleicht ist das Coden Ihr Leben und das Verwalten Ihre tägliche Arbeit – das kann funktionieren, bis Sie die Leidenschaft für Ihre Arbeit verlieren. Das Managen Ihrer Leidenschaft ist in meinen Augen grundlegend wichtig für das, was es bedeutet, Programmierer zu leiten. Sie werden sich in diesem Kapitel und in den nächsten mit einer Reihe von Management-Fähigkeiten befassen, die Ihnen helfen werden, Ihr Arbeitsleben ausgeglichen zu gestalten und Ihre Leidenschaft für Ihre Arbeit groß zu halten. Eine wichtige Management-Fähigkeit, die es Ihnen erlaubt, auch die Zeit zur Leitung zu haben, ist das Delegieren. Es ist ein Grundstein der Führung und ich werde mich damit schwerpunktmäßig in Kapitel 8 befassen. An dieser Stelle sollten Sie sich klar machen, dass zum Delegieren auch gehört, Ihren Mitarbeitern zu vertrauen. Vertrauen muss erst aufgebaut werden, ist aber grundlegend für eine erfolgreiche Leitung. Vertrauen ist auch eine menschliche Aktivität, die auf Gegenseitigkeit beruht. In diesem Kapitel werden Sie lernen, Ihren Instinkten zu Personen zu vertrauen, wenn Sie diese Instinkte mit ein bisschen anthropologischer Einsicht in die Hirne und Herzen von Programmierern verbessern.

1.2 Verrückte, exzentrische, seltsame und normale Leute erfolgreich führen

Nein, ich möchte nicht den ganzen Spaß aus dem Leiten von Programmierern nehmen, selbst wenn es oft als eine Übung im »Katzen hüten« beschrieben wurde – ohne Zweifel eine Referenz auf die unabhängige Natur dieser kreativen Individuen, die Code schreiben. Der lustige Teil ist, dass diese manchmal Ärger bereitenden, immer fragten und im Allgemeinen faszinierenden Mitarbeiter in ihrer Zusammenarbeit sehr anstrengend sein können. Indem Sie sie besser kennen lernen, verbessern Sie auch Ihren Führungsstil.

Wenn Sie wirklich gerne programmieren, verstehen Sie auch, was es bedeutet, Ihrem Code nahe zu stehen – es kann für Sie wie eine zweite Natur sein. Wie Ellen Ullman in *Close to the Machine* schreibt:

Ein mir bekannte Projektleiter sagte einmal, dass das Leiten von Programmierern wie das Hüten von Katzen ist ... Ich meine, Sie wollen keine treuen, gefolgssamen Hunde. Sie wollen all diese seltsame Verrücktheit, die einen guten Programmierer ausmacht. Andererseits müssen Sie irgendwie dafür sorgen, dass sie sich in die gleiche Richtung bewegen.³

Diese »gleiche Richtung« ist das Ziel beim Leiten von Programmierern, aber da jeder Programmierer verschieden ist, müssen Sie auch jeden unterschiedlich leiten. Sie können keine Programmierer führen, wenn Sie sie nicht verstehen. Im folgenden Abschnitt beschreibe ich verschiedene Typen von Programmierern und die Eigenschaften, die sie definieren. Sie mögen vielleicht ein paar Ihrer Mitarbeiter in dieser Liste mit Typen wiedererkennen, die ich »Rassen« nennen möchte, da es in diesem Buch schließlich um Katzen geht.

Erkennen von Programmierer-Rassen

Wie ist ein typischer Programmierer? Kann man Programmierer zu Stereotypen zusammenfassen, nur um sie zu verstehen? Wie bei so vielen Selbsteinschätzungs-tests aus dem Bereich der Psychologie ist es hilfreich, die Verhaltensweisen von Programmierern isoliert zu betrachten und zu berücksichtigen, dass viele dieser Charakterzüge gleichzeitig in derselben Person existieren können, selbst wenn sie gegensätzlich zu sein scheinen. Ich habe die Rassen in drei Kategorien eingeteilt: Hauptrassen, Nebenrassen und Mischlinge. Die *Haupttrassen* beziehen sich auf die häufigsten Typen, die Sie im Arbeitsleben finden werden. Die *Nebenrassen* finden sich manchmal, allerdings nicht so häufig wie die Hauptrassen. *Mischlinge* sind, wie Sie vielleicht vermuten, nicht sehr begehrt, aber auch sie gibt es im Arbeitsleben und daher müssen Sie auch diese erkennen können. Mischlinge können wunderbar arbeiten, solange Sie ihnen helfen, ihre Fähigkeiten auszubauen, um die in den Programmierablauf eingebrachten angeborenen Schwächen zu überwinden.

Wie schon erwähnt, können alle Individuen eine Verschmelzung der Charakteristiken darstellen, die mit einer Rasse in Verbindung gebracht werden – es macht die Arbeit mit dieser Person zu einer Herausforderung, aber es lohnt den Aufwand. Programmierer sind wundervoll komplexe Personen. Genießen Sie die Unterschiede und einzigartigen Verhaltensweisen jeder Rasse. Sie werden vielleicht viele dieser Erscheinungen wiedererkennen, wenn Sie das nächste Mal in den Spiegel schauen.

³ Ellen Ullmann, *Close to the Machine*, City Lights Books, San Francisco, 1997, Seite 20

Die Hauptrassen

Die folgende Aufzählung behandelt die Hauptrassen und ihre Charakteristika.

- **Der Architekt.**⁴ Diese Rasse wird von den meisten Managern hoch angesehen und kann Ihrem Team eine wertvolle Hilfe sein. Architekten kümmern sich meistens um die übergeordnete Struktur des Codes. Sie träumen in Objekten und ausdrückbare Flipcharts sind ihre besten Freunde. Sie leben, um Geschäftsprobleme durch Abstraktion und Systemanalyse zu lösen und dann konkrete Lösungen als Code zu erstellen. Dies ist eine notwendige Komponente beim Programmieren, aber es reicht alleine nicht aus. Ein Architekt kann oft großartige Ideen haben, aber sein oder ihr Code kann so eingeschränkt und reduziert sein, dass ihn niemand nehmen und erweitern kann. Der seltene Architekt kann ein gutes System in seinem Kopf erstellen – oder vorzugsweise in Visio –, um dann den Code zu erzeugen und der einsame Held zu sein. Der Nachteil dieses Vorgehens ist, dass der Code des Architekten manchmal ein Haustier werden kann, das nur einem Einzelnen gehorcht: Es kann keine Tricks für irgendjemand anderen machen.⁵ Manche Architekten sind nur daran interessiert, den Code zum Laufen zu bekommen, um ihn dann jemandem auf »niedrigerer« Ebene zum Vollenden zu überlassen. Manchmal finden Sie auch seltsame Konstrukte im Code eines Architekten, wie Nachrichtenfenster in einer Fehlerschleife, wenn der Code eigentlich als DLL auf einem Server laufen soll.
- **Der Erbauer.** Dieser Programmierer liebt ganz einfach den Prozess und das Ergebnis des Code-Schreibens. Erbauer haben nicht immer einen Master-Plan, aber sie sind oft schnell und ihr Code ist im Allgemeinen selbst im Alpha-Stadium ziemlich fehlerfrei. Der Code eines Erbauers entsteht aus Intuition heraus und daher sieht es so aus, als ob sie ihn nur aus dem Ärmel schütteln würden. Ein Erbauer kann auch eine erstaunliche Intuition haben und der Master-Plan mag sich in seinem Kopf befinden, so dass der Code ganz natürlich aus dieser Quelle entsteht. Fragen Sie einen Erbauer nach der Dokumentation, wird er sagen, dass der Code selbsterklärend ist. Fordern Sie einen Erbauer auf, eine Dokumentation zu erstellen, werden Sie vermutlich ein ziemlich gutes Ergebnis erhalten. Natürlich sollte der Code selbsterläuternd sein, aber in seinem Herzen liebt der Erbauer den Akt des Erstellens, so dass dies als Erstes auf seiner To-do-Liste stehen wird. Der Erbauer wird so viele Kompilierungsläufe pro Tag anstoßen, dass selbst Microsoft beeindruckt wäre. Dies kann zu solidem

4 Ich benutze den Begriff »Architekt« hier im Sinne eines Programmierers, nicht eines ausgereifen Software-Architekten. Schauen Sie auch in Kapitel 6, wenn Sie Interesse an einer Diskussion über die Wichtigkeit der Architektur in dem großen Weltbild der Entwicklung haben.

5 Dieses Konzept ist wichtig, da Schätzungen davon ausgehen, dass mindestens 70 Prozent der Software-Kosten mit der Wartung zusammenhängen. Siehe dazu William H. Brown et al, *Anti-Patterns: Refactoring Software, Architectures, and Projects in Crisis*, John Wiley & Sons, 1998, Seite 121.

Code führen, aber manchmal, wenn sich das Ziel ändert (was es immer tut), kann die Abgeschlossenheit des Codes zerbrechen und der Erbauer wird sich dabei wiederfinden, die (neue) Lösung mit neuen Hacks zu erreichen, um selbst zufrieden zu sein und das Gefühl zu haben, die Aufgabe gut erledigt zu haben. Bringen Sie einen Erbauer mit einem Architekten zusammen, haben Sie ein solides Team. Finden Sie einen Erbauer im selben Körper wie einen Architekten, lösen Sie damit den größten Teil Ihrer Personalprobleme.

- **Der Künstler.** Das Schreiben von Code ist im selben Maße eine Kunst wie eine Wissenschaft – das ist der Grund, warum Universitäten häufig beide Bereiche zusammenfassen und es »Hochschule für Kunst und Wissenschaften« nennen. Nehmen Sie die künstlerische Seite beim Programmieren weg, und Sie werden viele Leute verlieren, denen es unglaublichen Spaß macht, Code zu erschaffen. Der Künstler liebt den Akt der Erschaffung: Geschäftsanforderungen zu nehmen, sie auf Programmkonstrukte abzubilden und auf elegante Art und Weise Benutzerschnittstellenobjekte zu bauen, die sich gut darstellen. Manche Künstler erstellen wunderschöne Symmetrien in den Logiken, wenn sie an Komponenten arbeiten, die keine sichtbare Schnittstelle haben. Der Nachteil eines Künstlers ist, dass sich die Entwicklungszeit häufig stark verlängert, wenn der Programmierer versucht, herauszufinden, wie viele Gleichheitszeichen er in einer Zeile Code unterbringen kann, um immer noch das richtige Boolesche Ergebnis zu erhalten. Der Vorteil ist, dass Code, der keine Kunst darstellt, häufig echtes Design und Handwerkskunst fehlt. Nehmen Sie diese künstlerischen Qualitäten weg, erhalten Sie eine Zeitbombe, die irgendwann bei Ihren Benutzern hochgehen wird. Der Künstler teilt seine Fähigkeiten mit dem Erbauer und dem Architekten, hat aber zudem ein Gefühl für Stil.
- **Der Ingenieur.** Sie müssen diese Jungs und Mädels einfach lieben. Sie kaufen jedes verfügbare Tool von Drittherstellern, schreiben Dutzende von COM-Objekten und schrauben sie alle so zusammen, dass sie schon in Version 1 funktionieren. Nur wenn es daran geht, Version 1.1 zu erstellen, zeigt sich die dunkle Seite ihrer Liebe zur Komplexität. Das Programmieren wird häufig als Software-Engineering beschrieben, und viele Aspekte unseres Berufs können durch diesen Ansatz abgebildet werden. Passen Sie nur auf, dass der Ingenieur nicht alles alleine macht. Software von einem Ingenieur ist nicht das schlechteste, weil »Engineering« im besten Sinne die Anwendung wissenschaftlicher Prinzipien auf Software-Probleme ist. Sie brauchen Entwickler, die keine Angst vor Komplexität haben, aber Sie brauchen auch keine, die Komplexität nur aus Spaß an der Freude einführen. Ich möchte Ingenieure keineswegs verteufeln – ich war selber für viele Jahre ein solcher auf der Hardware-Seite. Allerdings ist es gerade dieser Hardware-Aspekt, der manchmal dem zuwiderläuft, was Software ausmachen sollte (zum Beispiel flexibel und leicht wiederverwendbar zu sein). Hardware dient im Allgemeinen einem einzelnen ingenieurmäßigen Zweck und Sie wollen normalerweise nicht, dass Ihre Software ebenso ist.

- **Der Wissenschaftler.** Das sind Männer und Frauen ganz nach dem Herzen von Babbage und Turing. Sie würden nie im Leben ein GoTo nutzen. Alles wäre nach den Grundlagen der Informatik ausgerichtet, wenn irgendwann einmal der Tag kommt, an dem Programmieren mehr Wissenschaft als Kunst ist. Und das ist häufig das Problem: Denen geht es vor allem um die akademische Reinheit, während es Ihnen eher um den morgigen Entwicklungsschluss und die funktionierende Software geht. Wissenschaftler sind nützlich und ihre Ideen sind oft grundlegend, wenn es um schwierige Coding-Probleme geht. Achten Sie nur darauf, dass die Vollkommenheit nicht die Praktikabilität überwiegt. Ingenieure und Wissenschaftler haben einige Gemeinsamkeiten, vor allem wenn es um die Wertschätzung des Komplexen geht – manchmal könnten Sie denken, dass sie am Altar irgendeines Gottes beten, der die Komplexität darstellt. (Oft bringen sie ihre Opfergaben zu diesem Tempel!) Honorieren Sie die tiefen Einblicke der Wissenschaftler und nutzen Sie ihre Produkte, wenn sie nützlich sind. Vermeiden Sie aber das Vermächtnis der Komplexität, das sich auf ewig einstellen wird, wenn sie freies Spiel mit dem Code betreiben können.
- **Der Geschwindigkeitsteufel.** Wie der Name schon sagt, sind diese Männer und Frauen schnell. Keine Kommentare, keine Einrückungen, seltsame Konventionen für Variablennamen, aber sie erzeugen Ergebnisse und es funktioniert meist recht gut, bis der erste nicht abgefangene Fehler auftaucht. Manchmal sind diese Coder einfach noch nicht lange im Geschäft und wollen Sie beeindrucken, weil sie denken, dass Geschwindigkeit die vorrangige Eigenschaft ist, die Sie als Manager erwarten. Haben wir nicht oft diesen Eindruck hinterlassen? Vielleicht sind wir Manager selber schuld an diesen Geschwindigkeitsteufeln. Unsere Chefs geben uns die Deadlines, die sie aus irgendwelchen Meetings mit denen ganz oben erhalten haben, und unser Job ist, es möglich zu machen. Haben wir nicht oft gehört, wie dumm es ist, einen Entwicklungsschluss festzulegen, bevor die Anforderungen feststehen? *Gewöhnen Sie sich daran.* Die reale Welt ist nun mal so und Benutzer und Marktsituation verlangen oft, zu versprechen, bevor wir planen. Das ist der Grund, warum Sie dieses Buch lesen – Sie wollen etwas Unterstützung in der schnellen, grausamen und häufig nachtragenden Welt der Software-Entwicklung.

Die Nebenrassen

Im Folgenden sind die Nebenrassen und ihre Charakteristika aufgeführt.

- **Der Magier.**⁶ Sie wissen nicht, wie es dieser Programmierer macht, aber er oder sie scheint immer die vermeintlich hartnäckigsten Probleme mit einer Lösung zu knacken, an die keiner vorher gedacht hat. Der Magier schafft es auch ohne Zeitverzug und manchmal erhält man sogar verständliche und wartbare Software. Ein bisschen Magie kann einen in unserem Handwerk sehr weit bringen;

⁶ Manche bevorzugen den Begriff »Guru« oder »Zauberer«. Ich mag »Magier«.

zu viel sorgt allerdings dafür, dass Sie sich eher als Zauberlehrling denn als bodenständiger Manager von hart arbeitenden Menschen wiederfinden. Mit anderen Worten: Wenn Sie sich zu viel auf den Magier verlassen, wird er Sie eventuell enttäuschen: Niemand kann andauernd zaubern.

- **Der Minimalist.** Dieser Programmierer erzeugt wenig Code, trotzdem ist dieser oft sehr mächtig. Jede Prozedur passt im Code-Editor ohne Scrollen auf den Bildschirm. Objekte sind klein und schnuckelig und haben eine eindeutige Aufgabe. Hört sich gut an, oder nicht? Das kann es auch sein, solange der Minimalist nicht nur versucht, den Job hinter sich zu bringen, um dann zum nächsten, interessanteren Projekt zu wechseln. Manchmal – und diese Eigenschaft wird ab und zu mit den Architekten geteilt – sind Minimalisten schnell gelangweilt, nachdem das Problem einmal gelöst ist und sie keine Lust mehr haben, sich mit den Untiefen des Codings auseinander zu setzen, wenn im Alpha-Teststadium Fehler auftreten. Manche Minimalisten sind ziemlich wählerisch, was die Applikationen angeht, an denen sie arbeiten. Sie sind oft auch nicht sehr erfolgreich bei der Code-Wartung.
- **Der Analogist.** Okay, ich habe dieses Wort selbst erfunden, und nein, es ist nicht die Krankenschwester, die Sie vor einer OP in Narkose versetzt – dies ist der Programmierer, der nicht wirklich gut abstrahieren kann, aber sehr gut Analogien zieht. Analogisten treiben Sie bei Design-Meetings in den Wahnsinn, weil Sie die dauernden Vergleiche ermüden, aber oft verstehen sie das Problem schnell und erzeugen praktikablen, wartbaren Code. Manchmal haben sie bevorzugte Analogien, die sie dann auf jedes Software-Problem anzuwenden versuchen. Sie mögen es, Komponenten als bewegliche Teile anzusehen, und wenn alles läuft, sagen sie, dass ihr Code »auf allen Zylindern läuft«. Ihre Analogien sind immer mit einem greifbaren Objekt verbunden anstatt mit einem abstrakten. Sie begreifen es. Kombinieren Sie sie mit einem Architekten und wenn sich die beiden nicht gleich gegenseitig umbringen, erhalten Sie anständige Software. Die einzige Gefahr bei einem Analogisten ist, dass er nicht abstrakt genug arbeitet, um Objekte zu erstellen, die klare Schnittstellen haben, um mit den anderen Schichten in der Software zusammenzuarbeiten. Die Möglichkeit, ausreichend abstrakte Objektschnittstellen erstellen zu können, ist eine der Stärken objektorientierter (OO) Programmierung. Da ist es manchmal für Personen, die immer in konkreten Beispielen denken, nicht möglich, ihre Aufgabe ordentlich zu erledigen.
- **Der Spielzeugbauer.** Dieser Programmierer betont den Spaß an der Technik zu sehr. Sie haben eine Person, die neue Spielzeuge mag, aber Sie bekommen die gleichen alten Probleme. Wenn wir ehrlich sind, mögen wir alle in diesem Bereich den Spielaspekt der Technologie. Ich erinnere mich an meinen ersten Computer: Er war analog, man drehte an Rädchen, die Schalter in einem vorbestimmten Hardware-Algorithmus umlegten. Es war wie ein Rechenschieber, der auf einem Steroide-Trip war, aber ich liebte das Teil. Ich mag immer noch

den Spaß, der aus der Arbeit mit tollen technologischen Tools entsteht. Dämpfen Sie die Lust der Spielzeugbauer auf Spielzeuge durch den Grund ihrer Anstellung: dem Erstellen von Geschäftslösungen. Nur weil sie es schaffen, 30 Eingabefelder auf einem Bildschirm unterzubringen, der für eine Auflösung von 800 x 600 Pixel gedacht ist, bedeutet das nicht, dass sie auch die Anforderungen der Benutzer erfüllt haben.⁷ Spielzeugbauer kommen zwar sehr gut mit verschiedenen Technologien klar, vergessen aber oft den eigentlichen Zweck der Software. Häufig denken sie, dass ihre Aufgabe darin besteht, Spaß mit den Tools zu haben, anstatt die Aspekte der Programmierung im Auge zu behalten, die Wartung ohne massive Aufwände ermöglichen.

Die Mischlinge

Die folgende Aufzählung behandelt die Mischlingen und ihre Eigenschaften.

- **Der Schlamper.** Es gibt nicht viel Gutes dazu zu sagen. Manche Zeitgenossen sind einfach schlampig und das sieht man in ihrem Code. Sie ignorieren kleine Dinge wie anständig benannte Variablen in korrekter ungarischer Notation. Vielleicht halten sie persönliche Probleme davon ab, ihre Aufgaben anständig zu erledigen. Vielleicht brauchen sie eine Anleitung zum Schreiben effektiven Codes. Sie beginnen mit einem Stil und nach einer oder zwei Prozeduren haben sie einen anderen übernommen. Das Nachvollziehen ihres Codes ist sehr hart und manchmal müssen Sie ihn noch spät in der Nacht neu schreiben, nur um sicherzustellen, dass Sie die Deadline erreichen. Sie, ihr Manager, haben versagt – nicht die Programmierer, die nur Schlamper sind, die vielleicht zum Beta-Testen versetzt werden sollten. Nein, streichen Sie das, das würde die Probleme nur an eine andere Stelle verschieben und doch wieder auf sie zurückfallen. Manche Schlamper können rehabilitiert werden, wenn sie wirklich gerne Code schreiben. Dann sollte man ihnen mehr Aufmerksamkeit widmen oder sie besser betreuen. Die das nicht können, brauchen vielleicht nur einen sprichwörtlichen Tritt in den Hintern oder eine Vorstellung bei einem Berufsberater.
- **Der Eingeschüchterte.** Dieser Programmierer weiß nicht, wo er anfangen soll. Er oder sie schaut dauernd auf die Spezifikation (oder wartet auf sie) und versucht, einen Punkt zu finden, mit dem er starten kann. Verstehen Sie mich nicht falsch: Zurückhaltung kann eine gute Eigenschaft sein, wenn sie zu sorgsam erstelltem Code führt, selbst wenn es das Ergebnis eines schlechten Programmierers ist, der keine Laufzeitfehler erzeugen will. Ihre Aufgabe ist, dem Eingeschüchterten Beispielcode zu geben, der zeigt, wo man beginnt und wie man programmieren sollte. Oft zeigen Leute, die erst ein paar Jahre im Geschäft sind, diese Zurückhaltung und durch entsprechende Erziehung können Sie dies ändern. Manchmal finden Sie Zurückhaltung auch bei einem erfahrenen Programmierer, der nicht so einen tollen Lebenslauf hat. Vielleicht war sein letz-

⁷ Hassen Sie nicht auch die Benutzer? Was für Spaß könnten wir haben, wenn wir nur Software für Programmierer schreiben würden.

tes Jahresgespräch schlecht und er möchte es nun besser machen, hat aber Angst, es zu vermasseln. Fehlende Sicherheit zeigt sich oft als Zurückhaltung, also kümmern Sie sich um diese Leute und sorgen Sie dafür, dass sie von Zeit zu Zeit ein wenig Erfolg haben, wenn Sie ihr Händchen halten. Betreuung, die beste Erziehung für einen zurückhaltenden Programmierer, werde ich in Kapitel 8 behandeln. Sie werden entdecken, dass das Mentorieren eine Ihrer Hauptaufgaben als Leiter ist. Die Aufwendungen dafür machen sich aber sehr gut bezahlt.

- **Der Amateur.** Amateure sind Möchtegern-Programmierer. Sie gelangen in den Rang eines Hackers als Power-User irgendeines Tools, das Makros schreibt. Sie verlassen ihre gemütliche Rolle im Support oder beim Testen, weil sie denken, dass Programmierer richtig cool sind. Natürlich *sind* wir cool, aber das ist nur ein Nebenprodukt dessen, was wir tun. Diese Jungs brauchen Ausbildung und Sie müssen sorgfältig ihr Vorankommen auf der Lernkurve abschätzen, bevor Sie sie bei der Erstellung von geschäftskritischen Applikationen einsetzen. Diese Möchtegerns werden bei ihren Aufgaben oft desillusioniert, wenn sie einmal festgestellt haben, wie schwierig das Programmieren sein kann und wie viel Aufmerksamkeit jedes Detail erfordert. Sie begreifen häufig nicht, dass objektorientierte Methoden dem prozeduralen Paradigma überlegen sind, da sie keine echte göttliche Erscheinung hatten. Zur Verteidigung der Amateure sollten Sie sich den folgenden Spruch merken: »Amateure haben die Arche Noah gebaut, Profis die Titanic.« Manchmal kann die unverbrauchte Ansicht eines Amateurs auch uns alten, angesäuerten Techno-Nörglern hilfreich sein.
- **Der Ignorant.** Dieser Programmierer ist auch als Herr oder Frau Töricht bekannt oder schlimmer noch, er oder sie ist dumm und weiß es nicht einmal. Geben Sie auf diese Personen Acht. Vielleicht haben Sie sie vererbt bekommen – bitte stellen Sie sie nicht selber ein. Nun, ich habe keine Vorurteile gegenüber geistig eingeschränkten Personen, aber sie haben keinen sinnvollen Platz in einem Beruf, der konstantes Lernen und Selbstdisziplin erfordert. Ignoranz kann toleriert werden, solange sie nicht absichtlich ist. Vielleicht sollte diese Person in die Testabteilung versetzt werden, da manchmal besonders dumme Benutzer die Fehler finden.⁸ Ein anderes Wort zur Dummheit: Wir alle leiden unter dem andauernden Problem, das zwischen Tastatur und Stuhl existiert. Wenn das Schreiben von Code nicht ein bisschen Gehirnschmalz benötigte, würde es jeder tun, oder? Geben Sie nur darauf Acht, dass Sie nicht Ignoranz mit Dummheit verwechseln. Ignoranz kann geheilt, Dummheit sollte vermieden werden. Wenn Sie in eine Abteilung geraten, die nicht von einem professionellen Programmierer aufgebaut wurde, kann es sein, dass sie ein paar dieser Rassen in Ihrem Dunstkreis finden. Vielleicht haben ein paar technisch nicht versierte Manager Leute zusammengesetzt, die sich als Programmierer verkauft haben, aber nicht über die entsprechenden Fähigkeiten verfügen.

⁸ Ich bevorzuge den Begriff »Programmanomalie« oder »undokumentiertes Feature« anstelle von »Fehler«.

- **Der Salat-Chefkoch.** Hier existiert eine Liebe für das Zusammenkochen von Software. Diese Rasse setzt sich ein bisschen aus dem Ingenieur, dem Schlamper und einem nicht sehr begabten Künstler zusammen, leider stehen aber die Zutaten nicht im richtigen Verhältnis zueinander. Das Ergebnis ist eine Vorspeise aus Code-Stilen und Zusatzkomponenten sowie einer allgemeinen Unordnung im Code. Es mag ansprechend aussehen, aber ein Biss und Sie wissen, dass Sie sterben werden. Schicken Sie diesen Programmierer in einen Kochkurs und gehen Sie sicher, dass es sich bei dem, was oberflächlich nach etwas Talent aussieht, nicht um einen ausgewachsenen Schlamper handelt. Diese Mischlingsrasse taucht vermutlich selten in reiner Form auf, aber ich habe sie aufgeführt, weil die Eigenschaften in einer Reihe von Code-Stilen bei Programmierern erscheinen. Wenn sie nicht mit Ihren Firmenstandards warm werden können, werden Sie sich nur damit beschäftigen können, herauszufinden, was sie taten und wie ihr Code gewartet werden kann. Ihre Rolle als Code-Reviewer (siehe Kapitel 6) wird entscheidend sein, wenn es um die Rehabilitation des Salat-Chefkochs geht.

Mit den Rassen arbeiten

Programmierer sind zunächst einmal Menschen, so dass alle die im vorigen Abschnitt erwähnten Eigenschaften zu einem größeren oder kleineren Teil in derselben Person existieren können. Manche dieser Eigenschaften stehen im Gegensatz zueinander, aber machen Sie sich darum keine Sorgen. Jeder Mensch, der heute Nacht auf diesem Planeten herumläuft, ist in Teilen ein Widerspruch in sich, und Sie und Ihre Programmierer machen da keine Ausnahme. Wichtig für Sie als Manager dieser Wunder der Natur ist, mit Verständnis und entsprechender Motivation und vor allem mit Weisheit zu reagieren, die Sie am besten durch Erfahrungen und das Kennenlernen Ihrer Mitarbeiter aufbauen. Versuchen Sie, sich mit Ihren Programmierern zu identifizieren und die Facetten ihrer Persönlichkeiten wahrzunehmen, die am hellsten unter der Sonne der Bemühungen und den Blitzen der Projekte, die kurz vor ihrem Abschlusstermin stehen, hervorstechen.



Versuchen Sie, sich mit Ihren Programmierern zu identifizieren und die Facetten ihrer Persönlichkeiten wahrzunehmen, die am hellsten unter der Sonne der Bemühungen und den Blitzen der Projekte, die kurz vor ihrem Abschlusstermin stehen, hervorstechen.

Aus welchen Katzenrassen ließe sich nun ein gutes Programmiererteam zusammenstellen, wenn man einmal annimmt, dass Sie Ihre Abteilung von Grund auf selber aufbauen können? Meine erste Wahl wäre, eine gute Balance zwischen Architekten und Erbauern zu haben. Diese zwei Rassen bringen die besten benötigten Fähigkeiten zum Erstellen von Software mit: Eine hat einen guten Überblick auf hoher Ebene, während die andere im Detail glänzt. Ein Künstler könnte Ihre

nächste Wahl sein, um ihn von Zeit zu Zeit einzubringen. Aber leider, leider, werden Sie vermutlich nicht in der Lage sein, Ihre Gruppe aus den idealen Kandidaten zusammenzustellen. Sie müssen mit dem arbeiten, was Sie haben, und daher sollte Ihre Fähigkeit des Umgangs mit der Mischung der Eigenschaften, die ich klischeehaft dargestellt habe, durch Ihr Verständnis, Ihre Geduld und die Betreuung geschärft werden – das Führen benötigt alle drei Komponenten in einer guten Kombination.

Ein weiterer Entwicklertyp, vor dem Sie aufpassen sollten, ist der Cowboy-Programmierer. Dieser Typ passt nicht sehr gut in die Liste der anderen Rassen, da er am besten als eine umfassende Einstellung beschrieben wird. Diese Einstellung beschreibt den Programmierer, der sehr gut in dem ist, was er oder sie macht, aber leider praktisch überhaupt nicht zu managen ist. Cowboys sind der Meinung, dass sie sich die Programmieraufgabe herausuchen können, die sie haben möchten und zwar zu ihren Bedingungen, nach ihrem Zeitplan und so, wie sie meinen, dass es funktionieren soll. Sie können diesen Typ als einsamen Wolf bezeichnen (oder eher als streunende Katze, um im Kontext dieses Buchs zu bleiben). Diese Cowboys können in Abhängigkeit von ihren Bedürfnissen und Fähigkeiten bei der Toleranz exzentrischer Personen Wunder bewirken oder Verwüstungen anrichten. Seien Sie vorsichtig mit Cowboys, sie werden nie ein Teil Ihres Teams sein. Greifen Sie nur im Notfall auf sie zurück oder falls das Projekt wirklich ein Ein-Personen-Job ist, dessen Ergebnisse auch nicht vom Team gewartet werden müssen.

Warum finden wir alle diese interessanten Charakterzüge in Programmierern? Ich glaube, es liegt daran, dass die Natur der Software-Entwicklung eine bestimmte Art von Menschen anzieht. In dem Klassiker *The Mythical Man-Month* beschreibt Frederick Brooks⁹ unseren Beruf als einen, der fünf verschiedene Arten von Freude bereitet:

1. Die Freude daran, Dinge zu erstellen.
2. Die Freude daran, Dinge zu machen, die für andere Personen nützlich sind.
3. Die Faszination, puzzleartige Objekte zu entwerfen, die sich ineinander fügen und dabei auch noch ihren Zustand ändern.
4. Die Freude daran, dauerhaft Neues zu erlernen.
5. Die Freude daran, mit einem so flexiblen Medium zu arbeiten – rein gedankliche Elemente, die aber trotzdem existieren, sich bewegen und in einer Weise funktionieren, in der es andere Gedankenspiele nie könnten.

Diese Freude zieht die Art Personen an, die Sie managen. Wenn Sie verstehen, was sie (und Sie selber) motiviert, kann dies eine enorme Hilfe beim Umsetzen Ihrer Führungsrolle sein.

⁹ Frederick P. Brooks, *The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition*, Addison-Wesley, 1995, Seite 230. Dies ist ein zeitloser Klassiker – nur sehr wenige Bücher in unserem Bereich werden nach 25 Jahren neu aufgelegt und dieses Buch ist es wirklich wert.

Katzenkämpfe! Ein Wettbewerb im Fauchen und Kratzen

John und Kevin lagen während der Design-Meetings andauernd im Clinch. Wir hatten gerade mit der Überlegung begonnen, wie sich der Benutzer am System anmelden würde, als sie über Details der Entwurfstechniken auf sehr niedriger Ebene diskutierten. Das Meeting kam nicht voran und es waren sehr viele weitere Features zu entwerfen. Zumindest dachte das jeder. Da es aber keine klare Agenda für die Besprechung gab, kämpften John und Kevin miteinander, da John, ein Berater, und Kevin, ein langjähriger Angestellter und sehr kreativer Programmierer, beide sehr verschiedene Motive und Pläne für das Meeting hatten. Sie waren mehr daran interessiert, festzustellen, wer von beiden der Cleverere war, statt ein System zu entwerfen. Denn John war als Entwicklungsleiter benannt worden, einem Job, den Kevin unbedingt haben wollte. Es war auch keine Hilfe, dass der Chef nicht mit in der Besprechung war; so gab es niemanden, der als Schlichter dienen konnte.

Ein Tag verging und die anderen Programmierer wurden immer stiller, während John und Kevin über jedes kleine Detail beim Design stritten. Am Ende des zweiten Tages waren nur sehr wenige Ergebnisse auf dem Flipchart zu finden und das, was gemacht worden war, war das Resultat langer und ermüdender Kämpfe zweier balgender Katzen: John und Kevin. Der Rest des Teams wurde so sehr abgeschreckt, dass sie daran zu zweifeln begannen, ob das System je erstellt werden würde. Zudem war die Gruppe beauftragt worden, dieses neue System so schnell wie möglich zu bauen, da das bestehende System die Firma im Markt nicht sehr gut dastehen ließ.

Diese kurze Geschichte zeigt einige der Schwierigkeiten auf, wenn Berater und eigene Mitarbeiter im selben Team arbeiten, besonders, wenn der Chef nicht mit im Raum ist. Sie mögen vielleicht auch die Entscheidung in Frage stellen, einem Berater die Leitung der Gruppe zu übertragen. Zudem sieht man, dass ein Design-Meeting ohne einen sorgfältig aufgestellten Plan, eine Agenda und eine Methode zum Lösen von Differenzen Zeitverschwendung sein kann und die gesamte Idee des »Design vor dem Coden« in Gefahr bringt. Die Dynamik zwischenmenschlicher Beziehungen im Entwicklungsteam zeigt auch die Notwendigkeit, Ihre Leute zu kennen, bevor Sie die Führungsrolle im Team vergeben.

Das Ergebnis dieser speziellen Geschichte war, dass das Projekt abgebrochen wurde und ein konkurrierendes Team einer anderen Abteilung die Software entworfen und erstellt hat. Der abwesende Manager hatte auch ein paar schwierige und düstere Tage, während er seinem Chef erklären musste, warum sie nicht in die Gänge gekommen sind und auf den Punkt ein Ergebnis abgeliefert haben, wo es doch beim Starten des Projekts so viele Versprechungen und Versicherungen gab.

1.3 Ruhm, Ehre und kleine Scheinchen

Jeder mit einem Job möchte geschätzt werden und das Gefühl haben, dass sein Beitrag wertvoll ist. Auch inkompetente Mitarbeiter wollen gewürdigt werden, selbst angesichts der Tatsache, dass sie eher einen negativen Beitrag für die Firma leisten. Sicher, wir alle meinen, dass das Coden alleine schon eine Belohnung ist, aber wenn man uns unseren Gehaltsscheck wegnimmt, sieht man, wie lange wir noch fleißig sind. Selbst wenn man uns gut bezahlt, wünschen wir uns Anerkennung von unseren Kollegen und ein »Gut gemacht!« von unserem Chef. Echte Katzen putzen sich vielleicht in einer Ecke, wo sie glauben, dass niemand sie beobachtet, aber Programmier-Katzen mögen es, sich in aller Öffentlichkeit zu putzen. Das Erstellen von Easter Eggs, zurzeit etwas aus der Mode geraten, war schon immer ein sicheres Zeichen, dass Programmierer ein Publikum wollen und brauchen. Sie als deren Leiter sitzen in der ersten Reihe und die Entwickler hoffen auf Ihren Applaus. Klar, manche verdienen keinen Beifall, sie brauchen eher eine ordentliche Standpauke.

Es gibt eine Zeit des Lobens und eine Zeit, innezuhalten und zu prüfen, ob Ihre Leute das Geld wert sind, das Sie in sie stecken. Das ist Ihre Herausforderung als Leiter und Manager. Wenn Sie sie loben, stellen Sie sicher, dass es öffentlich ist. Wenn Sie kritisieren, machen Sie es ohne Publikum. Diese Richtlinien wurden nicht nur erstellt, weil sie mit als höflich gelten – sie sind auch notwendig, weil Ihre Aktivitäten gegenüber einer Person sich auch auf alle anderen Teammitglieder auswirken. Öffentliche Demütigung sorgt nie dafür, dass eine Gruppe zu einer produktiven Einheit wird. Öffentliche Belobigung, sofern aufrichtig und auch verdientermaßen, kann Wunder bewirken. Machen Sie es aber nicht beiläufig, indem Sie nur rufen »Ihr habt tolle Arbeit geleistet!«, wenn Ihre Mitarbeiter die Besprechung verlassen. Nehmen Sie sich die Zeit, damit es auch ankommt. Teilen Sie die Gründe für das Lob mit und lassen Sie das Team an Ihren Gedanken teilhaben.



Es gibt eine Zeit des Lobens und eine Zeit, innezuhalten und zu prüfen, ob Ihre Leute das Geld wert sind, das Sie in sie stecken.

Noch etwas anderes zum Thema Loben. Vielleicht fühlen Sie sich je nach dem Verhalten Ihres Chefs übergangen, wenn es Zeit für Komplimente ist. Als Leiter bemühen Sie sich die ganze Zeit, Ihre Gruppe zum Erfolg zu führen. Wenn die Gruppe Erfolg hat, loben Sie sie. Wer lobt Sie? Manchmal ist die Antwort: »Keiner«. Dann wünschen Sie sich von Zeit zu Zeit selbst einen Klaps auf die Schulter. Diese Position, in der Ihre Anstrengungen und Erfolge der Grund für den guten Ruf des Teams ist, kann unglücklich sein, wenn Sie selber den Ruhm suchen. Leiter müssen lernen, ihre eigenen Qualitäten daran zu messen, wie gut ihre Mitarbeiter arbeiten.

Wenn Sie selber aus der Gruppe zu deren Leiter aufgestiegen sind, kann Ihre Arbeit doppelt schwierig sein, da Sie dann in der Stellung sind, über das Fortkommen Ihrer Freunde zu entscheiden. Lassen Sie Freundschaften nicht in die geschäftlichen Entscheidungen einfließen, sondern nutzen Sie sie vielmehr als Motivation für den Gewinn, der am Ende herauskommt. Niemand wird das Gefühl haben, dass Sie Freundschaften manipulieren, wenn am Ende jeder den Erfolg genießt.

Motivation durch Geld

Also, ich habe Geld schon erwähnt, oder? Nein? Oh je, ich vergesse das immer wieder. Schon die Bibel sagt »... Geld ist die Antwort auf alles.«¹⁰ Eine aktuelle Gehaltsumfrage bei amerikanischen Programmierern erbrachte Stundensätze in einem Bereich zwischen \$ 30 und \$ 150, wobei die meisten von uns im unteren Bereich dieses weiten Spektrums liegen dürften. Was bestimmt, ob Ihre Mitarbeiter ihr Geld wert sind? Leistung, Erfahrung, Effektivität, Berücksichtigung von Deadlines, der aktuelle Entwicklermarkt und ökonomische Gesichtspunkte, das alles sind Faktoren. Dazu kommen auch die bisherigen Gehälter für High-Tech-Mitarbeiter in Ihrer Firma. Ihre Aufgabe beim Einstellen neuer Personen oder beim Gewähren von Gehaltserhöhungen ist, fair und bedacht zur selben Zeit zu sein.

Fair und bedacht. Hmmm, das ist eine schwierige Aufgabe, weil Sie vielleicht das Geld mit vollen Händen ausgeben wollen, als ob es auf Bäumen wachsen würde, und hoffen, dass damit die Arbeitsleistung gesteigert wird. Denken Sie noch einmal nach – heutzutage etwas Luxus ist morgen eine Notwendigkeit. Geld ist wie Macht: Es kann korrupt machen. Lassen Sie mich nicht noch eine andere Zeile aus dem Neuen Testament zitieren.¹¹ Zurück zum Thema: wie können Sie bei den monetären Entschädigungen eine Balance erreichen? Wenn Sie die Aufgabe als ausgleichend betrachten, sollten Sie sich die Waagschalen Justitias vorstellen: Auf der einen Seite haben Sie eine Schale für Fairness, auf der anderen Seite eine für Bedächtigkeit. Fairness akzeptiert Gewichte, die der Erfahrung und Leistung der Programmierer entsprechen. Bedächtigkeit dagegen nimmt typische Geschäftskennntnisse auf, wie zum Beispiel die Beobachtung der Untergrenze und des durchschnittlichen Gehalts von Programmierern. Behalten Sie dies im Hinterkopf, wenn Sie Entscheidungen zum Thema Geld fällen – es ist eine gute Theorie.

Theorie? Was ist mit der Anwendung? Das ist der Grund, warum Geld so ein schwieriger Bereich sein kann, den Sie in Ihrem Job verwalten müssen. Einerseits haben Sie Prinzipien im Kopf, nach denen Sie Ihre Mitarbeiter entlohnen wollen, andererseits müssen Sie aber auch gleichzeitig die momentane Wirtschaftssituation und die Firmenrichtlinien berücksichtigen. Das kann bei Ihrer Planung sehr

¹⁰ Tatsächlich klagt sie diejenigen an, die das sagen. Siehe auch Prediger 10:14–19 in einer moderneren Version. Versuchen Sie, nicht allzu deprimiert zu werden, wenn Sie das lesen.

¹¹ Siehe im Neuen Testament, 1. Timotheus 6:10, bei dem Liebe, Geld und der Teufel in einem netten, logischen Syllogismus miteinander verbunden sind.

frustrierend sein. Das Gehalt kann durch Boni ergänzt werden, die auf eigenen Verdiensten und/oder dem Erfolg der Firma basieren. Diese Zugaben sind sinnvoll, solange der Beitrag der Mitarbeiter in der Formel zum Bestimmen des Bonus ausreichend berücksichtigt wird. Sie können die Boni regelmäßig zahlen, aber ich finde das problematisch, weil sie erwartet werden, sobald man einmal damit angefangen hat. Sie sollten mit Ihrem Chef einen Plan abstimmen, der zu Ihrem Unternehmen passt. Wenn Sie die Entscheidung selber zu treffen haben, sollten Sie das tun, was Sie für richtig halten, und dabei berücksichtigen, fair und bedächtig zu sein.

1.4 Der Thinking-Layer¹²

Fühlen Sie sich nun alle warm und kuschelig? Vermutlich nicht. Am meisten warten Sie sicher auf eine Liste mit Stichpunkten, welche Dinge man vermeiden sollte und welche forciert werden müssen. Es wird in den weiteren Kapiteln noch haufenweise solche Listen geben, im Moment aber möchte ich das Selber-Denken in Ihrer neuen Rolle hervorheben, anstatt Ihnen eine lange Liste mit Dos und Don'ts mitzugeben. Denken ist vermutlich das schwerste Ding, mit dem wir als Manager und Leiter zu tun haben, aber es ist wichtig – absolut wichtig – für unseren Erfolg. Wie Jim McCarthy in *Dynamics of Software Development* schreibt:

*Die wirkliche Aufgabe beim Management von Software-Entwicklung ist, so viel Intelligenz wie möglich zusammenzubringen und sie in die Aktivitäten zu investieren, welche die Entwicklung des Produkts unterstützen.*¹³

Denken Sie, während Sie unter der Dusche stehen. Denken Sie, während Sie auf dem Fahrrad sitzen, zu Fuß gehen oder inlineskateten. Denken Sie, während Sie den Beschreibungen der Dilemmas zuhören, die sich aus Design-Entscheidungen ergeben haben. Denken Sie, anstatt Fernsehen zu schauen oder im Web zu surfen – beides hat vielleicht 500 Kanäle, aber keiner hat etwas Passendes, auch wenn es Sie vom Denken befreit. Denken Sie sich voll, arbeiten Sie, bis Sie wieder leer sind, und fangen Sie dann wieder von vorne an. Das Ergebnis wird Sie überraschen.

Okay, lassen Sie uns ein wenig darüber nachdenken, wie man ein paar typische Situationen handhabt.

Nehmen wir einmal an, Sie haben eine Katze, die vor allem ein Minimalist ist, wenn auch ein sehr cleverer. Sie brauchen sie, um ein Produkt zu verbessern, das sie nicht geschrieben hat, das aber entscheidend für die aktuellen Ziele des Unternehmens ist. Wie motivieren Sie sie, wenn sie einen Blick auf den Code der anderen Entwickler geworfen hat und sagt: »Das ist zu kompliziert und muss neu

¹² Sie wissen, was »thinking« ist, wenn Sie sich mit Compilern auseinander gesetzt haben. Sie werden schnell mein Wortspiel verstehen.

¹³ Jim McCarthy, *Dynamics of Software Development*, Microsoft Press, 1995, Seite 5.

geschrieben werden.« Sie sagt dies natürlich, während sie Code betrachtet, der zwei Jahre brauchte, um geschrieben zu werden, und der Firma Geld einbringt. Und der ursprüngliche Entwickler ist schon lange nicht mehr da, um Erläuterungen geben zu können. Sie haben nun zwei Optionen: Stimmen Sie zu, machen Sie sie glücklich und zerstören Sie jede Hoffnung darauf, die Deadline halten zu können. Oder helfen Sie ihr dabei, zu sehen, wie sie aus der bestehenden Architektur lernen und einen signifikanten Beitrag gestalten kann. Appellieren Sie an ihren Sinn für eine ordentliche Architektur, indem Sie sie bitten, den bestehenden Code zu dokumentieren, um in Zukunft, wenn etwas mehr Zeit vorhanden ist, ein paar der Objekte so umzuschreiben, dass man sie einfacher nutzen kann. Wenn sie eine intelligente Programmiererin ist, sollte sie in der Lage sein, sich vorzustellen, was jemand anderes verwirklicht hat. Zögern Sie nicht, einen Vergleich zu nutzen. Für den Minimalisten ist alles, was er nicht geschrieben hat, Müll, aber in Wirklichkeit hat er vielleicht nur Angst, dass er den Code nicht versteht und ihn daher nicht administrieren will. Schauen Sie nach der versteckten Motivation, die wir Menschen alle haben, und denken Sie daran, dass sich Programmierer häufig hinter Ausreden verstecken, anstatt zuzugeben, dass sie sich noch nicht genug mit der Materie auseinandergesetzt haben.

Wie gehen Sie mit einem Architekten um, der denkt, dass seine Objektentwürfe um Klassen besser sind als alles, was bisher erfunden wurde – nur dass Sie durchaus der Meinung sind, dass es da ein paar Schwachstellen in seinem Design gibt? Erzählen Sie ihm nicht, dass sein Design mangelhaft ist, denn ansonsten werden Sie vermutlich direkt sein persönlicher Feind. Bitten Sie ihn stattdessen darum, zu erläutern, wie all die Elemente seines Entwurfs zusammenarbeiten, und einen Prototyp oder Testprogramme zu bauen, um die Funktionen anschaulich zu machen. Wenn diese Prototypen keinerlei Probleme aufzeigen, haben Sie vielleicht Schwachstellen gesehen, die gar keine sind. Bitten Sie den Architekten, seine Architektur zu unterteilen, wenn Sie das Gefühl haben, dass sie wie ein massiver monolithischer Block erscheint. Wenn die Komponenten zusammenarbeiten können, hat er vielleicht ein paar gute Ideen. Wenn die Objekte zu sehr miteinander verflochten sind, liebt der Architekt auch Komplexität, was in Zukunft zu ausgesprochen hohen Wartungskosten führen kann. Ein wirklich guter Architekt kann ein Framework erstellen, das jeder, der es nutzt, verstehen und leicht erweitern kann. Zudem bricht es nicht gleich zusammen, wenn der nächste Satz Erweiterungen ansteht. Der Schlüssel in der Zusammenarbeit mit einem Architekten liegt darin, den Code aus seinem Blickwinkel zu betrachten, auch wenn er auf dem einen Auge blind ist und das andere schlecht sieht.



Hören Sie erst zu und versuchen Sie, zu verstehen, bevor Sie Ihre Autorität als Manager einsetzen, um eine Lösung auf die richtigen Gleise zu bringen.

Was ist die gemeinsame Idee, die ich beim Umgang mit diesen Situationen vorschlage? Es geht darum, zunächst zuzuhören und zu versuchen, zu verstehen, bevor Sie Ihre Autorität als Manager einsetzen, um eine Lösung auf die richtigen Gleise zu bringen. Programmierer unterscheiden sich nicht vom Rest der Menschheit, wenn es zu Konfrontationen kommt: Sie wollen eine faire Möglichkeit, ihren Standpunkt darzustellen. Wie Stephen Covey in *Die sieben Wege zur Effektivität* schreibt: »Versuchen Sie erst, zu verstehen ... dann versuchen Sie, verstanden zu werden.« Das Erstellen eines Konsenses bei technischen Entscheidungen ist eine Kunst, dessen Leinwand die Offenheit für Ideen anderer ist. Es benötigt Geduld, solch eine Leinwand aufzubauen, selbst wenn Sie oft das Gefühl haben, dass die Zeit beim Erstellen der Software fehlt, um die Zustimmung von jedem zu erhalten. Dies mag in vielen Fällen Ihr *Gefühl* sein, aber ein Konsens sollte immer Ihr Ziel sein.¹⁴ Ich werde mehr über das Aufbauen eines Konsenses in Kapitel 5 bringen, das sich mit dem Führen und Verwalten eines Design-Meetings beschäftigt. Sie werden überrascht sein, wenn Sie lernen, dass ein Konsens nicht unbedingt durch Kompromisse erreicht werden muss.

Ein anderes Beispiel soll das Konzept des Verständnisses vor der Beurteilung erläutern. Manche Sprachen, wie zum Beispiel Visual Basic (VB) erlauben keine echten Objektkonstruktoren. Ich habe einen Künstler gesehen, der das Ereignis »Initialize« einer VB-Klasse verwendete, um einen Großteil der Dinge zu erledigen, mit denen das Objekt in einen Zustand gebracht wurde, in dem es von der nutzenden (Eltern-)Klasse benötigt wurde.¹⁵ Wenn ein Objekt in VB nicht instanziiert werden kann, ist es schwierig, den Fehler zu finden – Sie erhalten schlicht einen Fehler beim Erstellen des Objekts. Als ich ihn fragte, warum er dieses Ereignis verwendet hat, um fehleranfällige Aktionen durchzuführen, erwiderte er, dass dies elegant und sauber wäre und keinerlei Aktionen von Seiten des aufrufenden Objekts benötigte, um die Schnittstelle nutzen zu können. Meine Meinung ist natürlich, dass eine sichere Fehlerbehandlung Vorrang vor allen Versuchen haben sollte, Code »schön« zu machen. Ich habe ihm das zunächst nicht erzählt. Ich hörte mir seine Begründung an, beschrieb, was mit diesem Objekt schief gehen konnte, und führte dann ein Beispiel im Code vor. Er hat aus dieser Coding-Beispiel-Stunde mehr gelernt, als wenn ich ihm nur gesagt hätte: »Ändere es, das ist nicht richtig.« Auch hier gilt: Wenn Sie jemandem die Chance geben, seinen Standpunkt darzustellen, wird sich diese Person auch Ihrer Perspektive nicht verschließen.

14 Viele würden sagen, dass bei einer Zustimmung aller jedem die Schuld gegeben werden kann, wenn die Sache schief geht. Das kann stimmen, aber als Manager sollten wir uns mehr Sorgen um das Lösen von Problemen machen, anstatt Schuldige zu suchen. Erfolg hat viele Väter – seien Sie einer davon.

15 In VB übernimmt das Ereignis »Initialize« keinerlei Parameter und liefert auch keine Werte zurück.

1.5 Wie passen Sie sich an?

Sie haben in diesem Kapitel viele Ideen vorgestellt bekommen. Vielleicht fühlen Sie sich von dem Umfang an Anpassungen überwältigt, die notwendig sind, um ein erfolgreicher Leiter zu werden. Machen Sie sich keine Sorgen. Wir Menschen sind immer noch zu 99 Prozent genetisch identisch mit den Affen, und das eine Prozent, das uns unterscheidet, entstand nicht über Nacht. Die folgenden Kapitel wollen Ihnen helfen, sich zu ändern, sich zu überwinden und Erfolg zu haben, wenn Sie die anderen Elemente von Führung und Management betrachten.

Lassen Sie uns zusammenfassen, was wir bisher behandelt haben, um die Schlüsselprinzipien in Ihrem Herzen und Ihrem Hirn festzumachen.

- **Sie müssen sich anpassen.** Das Erlernen neuer Management-Fähigkeiten zum Führen erfordert, dass Sie sich selber an einen neuen sozialen Kontext anpassen. Sie sind der Chef und das ändert Ihre Beziehungen zu den Leuten in Ihrem Team.
- **Die Entwicklung des Charakters ist wichtiger als Verfeinerungen am äußeren Bild.** Führung kommt von innen und wird nicht durch ein äußeres Bild bestimmt. Sie sind immer noch ein Programmierer, aber Ihre Rolle als Leiter von Programmierern erfordert, dass Sie sich tief in die Persönlichkeiten der anderen, aber auch von sich selber einarbeiten.
- **Kennen Sie Ihre Mitarbeiter.** Werden Sie ein Anthropologe der Programmierkultur und der Leute. Lernen Sie, warum Ihre Mitarbeiter Code in der Art schreiben, wie sie es tun, und wie Sie deren beste Eigenschaft nutzen, während die schlechteren nur berücksichtigt werden. Das Hüten von Katzen bedeutet, sie dazu zu bringen, in die gleiche Richtung zu laufen: Das ist, was ein Leiter versuchen sollte, zu erreichen.
- **Belohnen Sie Ihre Mitarbeiter adäquat.** Loben Sie mit Worten und unterstützen Sie mit Geld. Bedenken Sie alle Faktoren, die Sie finanziell betreffen, aber seien Sie fair und bedächtig. Loben Sie überschwänglich in der Öffentlichkeit, wenn es angebracht ist, aber rügen Sie unter vier Augen.
- **Denken Sie.** Versuchen Sie anzuwenden, was Sie über die Personen wissen, um einen Konsens zu erzielen. Hören Sie zu und verstehen Sie andere Ansichten, bevor Sie ein Urteil fällen. Kümmern Sie sich mehr um das Denken: machen Sie das Lernen zu Ihrer zweiten Natur als Leitung. Vorgefertigte Pläne anderer zum Lösen von Problemen mit Ihren Leuten sind kein Ersatz für Ihre selbst entworfenen Pläne und Methoden, die auf die Probleme und Möglichkeiten in Ihrer Gruppe abgestimmt sind.

1.6 Was kommt noch?

Im nächsten Kapitel werden Sie sich selber als Manager betrachten, so wie Sie Ihre Mitarbeiter in diesem Kapitel analysiert haben. Ich werde ein paar schwierige Fragen darüber stellen, wie Sie sich Ihrer Aufgabe als Leiter der ausgesprochen wichtigen Bemühungen zum Erstellen von Software unter dem heutigen ökonomischen Klima annähern. Sie müssen Ihre Mitarbeiter managen, um ein guter Leiter zu sein, aber als Erstes müssen Sie sich selber managen.