HANSER

Ship it!

von Jared R. Richardson
und William A. Gwaltney

ISBN 3-446-40425-2

Leseprobe Kapitel 1

Wir sind, was wir immer wieder tun.
Eine herausragende Leistung ist daher kein einmaliger Akt,
sondern eine Gewohnheit.

Aristoteles

Kapitel 1

Einführung

Viele Softwareentwickler sind heutzutage frustriert. Sie arbeiten lange und hart, aber ihr Team scheint nicht in der Lage zu sein, das aktuelle Projekt fertig zu stellen. Das geschieht nicht aus Mangel an Wille oder Anstrengung – jeder im Team will das Projekt zu einem sauberen Abschluss bringen, aber niemand weiß, wie man alles zu einem Ganzen zusammenschnürt. Selten finden Entwickler die Zeit, um durch Lesen und Experimentieren herauszufinden, was funktioniert und wie sie es in ihrer Firma einsetzen können. Sie sind zu beschäftigt, um diese Art von Forschung zu betreiben.

Hier kommt *Ship It!* ins Spiel. Dieses Buch ist eine Sammlung von grundlegenden, praktischen Ratschlägen, die sich bei der täglichen Arbeit bewährt haben, in verschiedenartigen Projekten, in Firmen jeder Größe. Es ist das, was unserer Erfahrung nach funktioniert. Wir sind keine Consultants, die nach ein paar Wochen wieder weg sind; wir haben über einen längeren Zeitraum täglich in diesen Firmen gearbeitet. Wir haben nicht nur schlaue Vorschläge unterbreitet und sind dann ins nächste Projekt verschwunden. Wenn etwas nicht funktionierte, waren wir immer noch da, um es scheitern zu sehen. Aber wir haben auch mitbekommen, wenn etwas richtig gut lief.

Einige dieser Konzepte sind aus bekannten Vorgehensmodellen übernommen, und wir haben uns bemüht, entsprechend darauf hinzuweisen. Andere Konzepte wurden unter Blut, Schweiß und Tränen geboren. Wir haben mit vielen Werkzeugen, Techniken und Verfahren experimentiert, und wenn sich etwas bewährt hat, haben wir es übernommen. Wenn nicht, haben wir es über Bord geworfen.

Wenig von dem, was Sie hier lesen werden, ist außergewöhnlich originell (und das ist gut so!). Wir standen vielmehr „auf den Schultern von Riesen“, haben Konzepte der besten Köpfe der Branche aufgegriffen und daraus das geformt, was Sie hier sehen.

50 bis 70 Prozent der Softwareteams nutzen heute grundlegende, wohlbekannte Softwareverfahren nicht ([Cus03]). Auch wenn sie diese Verfahren kennen, ist ihnen oft nicht klar, wie man sie in ihrem Umfeld am besten einführt. Wir zeigen Ihnen, wie Sie Ihr Management für jedes dieser Konzepte gewinnen, wir legen praktische Schritte für den Start dar und weisen auf Warnzeichen hin, so dass Sie nicht vom Weg abkommen.

Ship It! wurde von Entwicklern geschrieben, die an vorderster Front im Einsatz waren. Dieses Buch enthält unsere Erfahrungen, keine Theorie, gewonnen sowohl im kleinen Startup als auch in der größten privaten Softwarefirma der Welt. Es ist ein realitätsnaher Ratgeber, der unabhängig von etablierten Vorgehensmodellen zeigt, wie Softwareprojekte funktionieren.

Wir haben versucht, uns mit diesem Buch an das Vorbild der Reihe *Pragmatisch Programmieren* zu halten: wir wollen eine praktische, einfache und leichtverdauliche Lektüre bieten. Wir hoffen, auf dem Fundament aufbauen zu können, das die anderen Bücher gelegt haben.

1.1 Herausragendes aus Gewohnheit

Wie also passt hier das Aristoteles-Zitat ins Bild? „*Wir sind, was wir immer wieder tun. Eine herausragende Leistung ist daher kein einmaliger Akt, sondern eine Gewohnheit.*“ Herausragende Leistungen sind nicht dadurch gekennzeichnet, dass wir ein großartiges Produkt herstellen (oder eine Anzahl großartiger Produkte). Sie sind durch das gekennzeichnet, was wir jeden Tag tun: unsere Gewohnheiten. *Außergewöhnliche Produkte sind lediglich eine Nebenwirkung unserer Gewohnheiten.*

Dieses Zitat auf uns anzuwenden (sowohl beruflich als auch persönlich) bedeutet, unser Leben als eine Nebenwirkung unserer Gewohnheiten zu erkennen – also sollten wir unsere Gewohnheiten lieber sorgfältig wählen. Die meisten Leute geraten eher zufällig an ihre Arbeitsabläufe, aus einer Vielzahl von Gründen: so haben sie es gelernt, so hat es ihr Chef immer gemacht usw. Wir können das aber auch besser.

Suchen Sie gezielt nach guten Gewohnheiten, und bereichern Sie damit Ihren täglichen Arbeitsablauf.

Versuchen Sie Folgendes. Finden Sie ein Vorgehensmodell, das Sie untersuchen wollen, und entnehmen Sie daraus eine Gewohnheit, die Ihnen brauchbar erscheint (und die sich eigenständig nutzen lässt). Verwenden Sie diese eine Woche lang. Wenn sie Ihnen gefällt und sie sich als nützlich erweist, dann wenden Sie sie einen weiteren Monat lang an. Praktizieren Sie die neue Gewohnheit, bis sie ein natürlicher Teil Ihres Arbeitsablaufs wird, und dann fangen Sie mit diesem Prozess wieder von vorne an. Wiederholen Sie diesen Prozess, so wie man ein Fundament Stein für Stein legt, und legen Sie sich so einen Grundstock von Fähigkeiten an, eine neue Gewohnheit nach der anderen. Scheuen Sie sich nicht, etwas wieder zu entfernen, wenn es in Ihrem Umfeld nicht funktioniert, und behalten Sie ein Verfahren nicht nur deshalb bei, weil es bekannt oder populär ist. Arbeiten Sie Ihren eigenen Weg heraus, je nachdem, was für Sie funktioniert und was nicht.

„Wie wir unsere Tage verbringen, so verbringen wir auch unser Leben.“[1] Wenn das so ist, dann müssen wir darauf achten, wie wir unsere Tage verbringen.

Tipp 1: ***Wählen Sie Ihre Gewohnheiten***

Nehmen Sie Gewohnheiten nicht zufällig an. Wählen Sie Ihre Gewohnheiten bewusst aus.

[1] Annie Dillard, US-Autorin, Dichterin, Pulitzerpreisträgerin Sachbuch 1975

1.2 Eine pragmatische Betrachtungsweise

Dieses Buch ist keine akademische Untersuchung, warum etwas funktionieren oder nicht funktionieren sollte. Es ist auch kein Verzeichnis verfügbarer Verfahren und Methoden, aus denen Sie auswählen können.

Dieses Buch zeigt stattdessen, was für uns in realen Softwareprojekten funktionierte. Wir führten üblicherweise ein neues Werkzeug oder Verfahren ein und verwendeten es, bis klar war, ob es funktionierte oder nicht. Diejenigen, die funktionierten, fügten wir unserer Methodensammlung hinzu. Schließlich hatten wir tatsächlich den Punkt erreicht, an dem wir den Eindruck hatten, dass wir wussten, was wir taten! Wir hoffen, dass diese Werkzeuge und Verfahren für Sie ebenfalls funktionieren.

Wir haben Zeit in Startups verbracht, die nicht über den Luxus verfügten, eine Methode nur deshalb zu verwenden, weil es „die richtige“ war. Unsere Verhältnisse haben uns gezwungen, Konzepte zu finden, die sofort in die Tat umgesetzt werden konnten. Wir haben auch in größeren Unternehmen gearbeitet, denen erheblich mehr Mittel und Möglichkeiten zur Verfügung standen. Wir fanden heraus, dass selbst große Unternehmen ein Werkzeug nicht nur deshalb einsetzen, weil es elegant ist oder von irgendeinem Guru propagiert wird. Sie wollen Lösungen für heutige Probleme, schnell und preiswert. Wir haben also hier eine Gewohnheit aufgeschnappt, haben dort eine Gewohnheit abgelegt, bis unsere Methodensammlung ausreichend allgemeingültig und übertragbar war und Probleme trotzdem effektiv lösen konnte. Dieses Buch ist eine Sammlung guter Gewohnheiten, die auch in Ihrem Unternehmen etwas bewegen werden – die Ergebnisse können erstaunlich sein.

Wir möchten Ihnen dazu die Geschichte zweier Firmen erzählen (Entschuldigung an Charles Dickens).

Die erste Firma war ein einziges Durcheinander. Sie hatten dort ein ziemlich teures Versionsverwaltungssystem gekauft, es aber nie installiert. Im Ergebnis verloren sie die Quellen der Demoversion, die sie potenziellen Kunden zeigten. Niemand war sich sicher, welche Features in ihrem Produkt enthalten sein sollten, aber das

gesamte Entwicklungsteam arbeitete trotzdem daran. Der Quelltext war instabil und stürzte ungefähr alle fünf Minuten ab (üblicherweise zum ungünstigsten Zeitpunkt – während einer Live-Vorführung). Dieses Durcheinander förderte nicht gerade die Moral; Firmenversammlungen endeten üblicherweise im lautstarken Streit. Manche Entwickler vermieden die Situation, indem sie sich den ganzen Tag in ihrem Büro versteckten. Alles in allem war dies *kein* guter Ort zum Arbeiten. Jeder wusste, dass es größere Probleme gab, aber niemand wusste, wie sie zu lösen waren.

Die zweite Firma war in deutlich besserem Zustand. Mit ungefähr der gleichen Anzahl von Entwicklern arbeiteten sie an drei größeren Projekten gleichzeitig. Diese Projekte hatten ihre Quellen in einem Versionsverwaltungssystem abgelegt; der Quelltext wurde automatisch kompiliert und getestet, sobald er geändert wurde. Das gesamte Team hatte tägliche Besprechungen, die kurz, professionell und effektiv waren. Jeder Entwickler wusste, an welchen Features er zu arbeiten hatte, weil jedes Projekt über einen Rahmenplan verfügte. Sie folgten dem Credo eines Steinmetzes: *Wir, die bloße Steine behauen, müssen uns immer Kathedralen vorstellen* ([HT03]). Das bedeutet, dass jeder sein Fachwissen und sein Können im Rahmen eines größeren, aufeinander abgestimmten Systems anwenden konnte. Ihre Produkte wurden pünktlich und ohne größere Schwierigkeiten ausgeliefert; sie waren stabil, weil sie handwerklich gut gemacht waren.

Das Erstaunlichste an diesen zwei Firmen ist, dass es sich um *dieselbe* Firma handelt, getrennt durch weniger als 6 Monate und die Anwendung der Prinzipen in diesem Buch (aber das hatten Sie sicherlich schon erraten, oder?). Nach dem Umschwung äußerte der Vorstand, dass wir „eine Atmosphäre der Perfektion“ eingeführt hätten und er „den Laden kaum wiedererkennt“. Die Arbeit in dieser Firma liegt noch nicht lange zurück, und wir haben dort die Prinzipien dieses Buches in nahezu der gleichen Form zum Tragen gebracht, in der wir sie Ihnen hier präsentieren. Die Verwandlung, die wir dort erlebten, ist einer der Gründe, weshalb wir uns entschieden, dieses Buch zu schreiben.

Wir haben diese Konzepte in Firmen jeder Größe entdeckt und angewendet, vom Startup mit vier Personen bis zu SAS, der größten privaten Softwarefirma der Welt. Ehrlich gesagt, wir waren erstaunt, wie gut diese Prinzipien in Firmen jeder Größe funktionierten.

Stellen Sie sich diese Prinzipien als Grundlage eines großartigen Produktes vor. Wenn Sie sich zu Beginn die Zeit nehmen, Ihre Infrastruktur ordentlich aufzubauen, werden Sie für den gesamten Lebenszyklus des Produktes davon profitieren. Natürlich lässt sich ein Projekt einfacher beginnen, wenn all diese Praktiken bereits im Einsatz sind. So wie bei einem rissigen Fundament eines Hauses, lassen sich einige Probleme leicht beheben, während andere tief in die Struktur gehen und nur mit viel Aufwand repariert werden können.

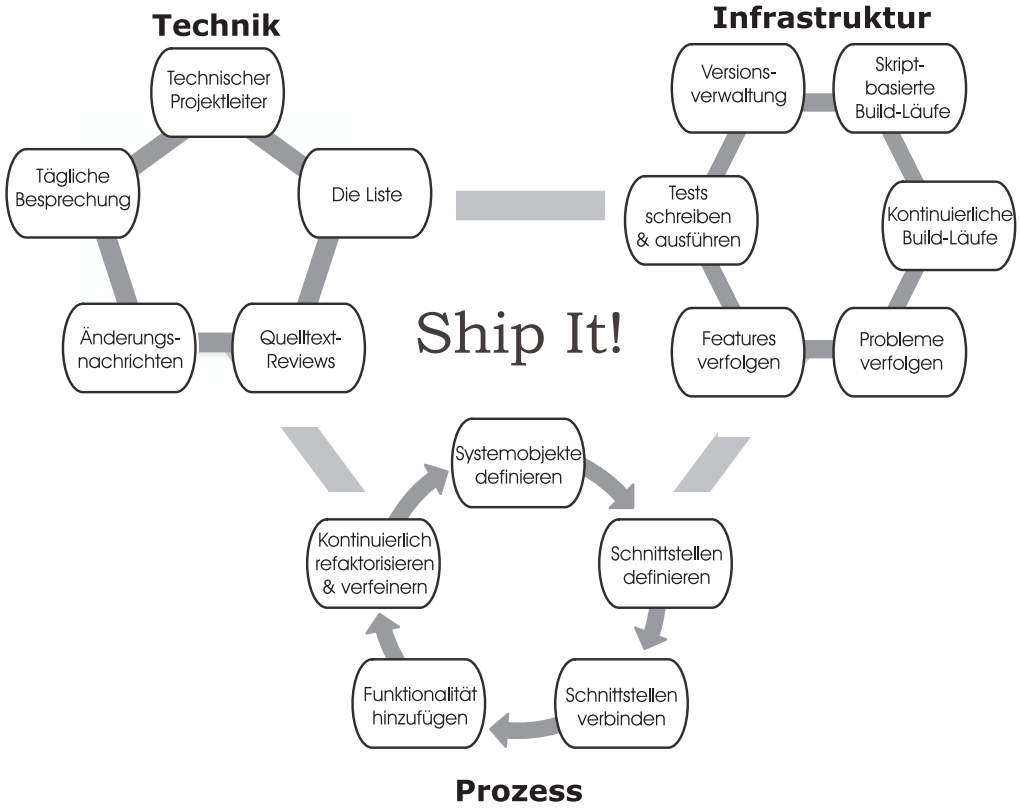


Abbildung 1.1: Wie man ein großartiges Produkt herstellt

Auch wenn Sie derzeit mitten in einem Projekt stecken sollten, so ist es doch niemals zu spät, gute Gewohnheiten anzunehmen. Viele der Konzepte können Sie in ein existierendes Projekt einführen, und wir werden im letzten Kapitel Wege dazu vorstellen.

1.3 Aufbau des Buches

Wir haben unsere Konzepte in drei Hauptbereiche gegliedert: Infrastruktur, Technik und Prozess (siehe Abb. 1.1, Seite 6). Diese Bereiche haben direkten Einfluss auf die Fähigkeit Ihres Teams, beständig ein den Wünschen Ihrer Kunden entsprechendes Produkt zu liefern.

Infrastruktur

In *Werkzeuge und Infrastruktur* behandeln wir Softwarewerkzeuge, die Ihr Leben und das Leben Ihres Teams einfacher machen. Ein gutes Versionsverwaltungssystem zum Beispiel hält die „Kronjuwelen“ Ihres Projekts – Ihren Quelltext – sicher und konsistent. Ein automatisiertes Build-System garantiert Ihnen wiederholbare Build-Läufe, überall und jederzeit. Weiterhin erörtern wir, wie man den Überblick über bekannte Fehler, Anforderungen und andere Themen behält, die aufkommen, sobald der Rest der Welt Ihre Entwicklungen zu Gesicht bekommt. Schließlich zeigen wir Ihnen, wie eine gute Testumgebung Ihnen die Sicherheit gibt, dass Ihr Quelltext auch tatsächlich das tut, was er tun soll.

Technik

In *Techniken für Pragmatische Projekte* behandeln wir bestimmte Verfahren, die Sie und Ihr Team nutzen können, um „klüger, nicht härter“ (engl. *smarter, not harder*) zu arbeiten. Wir berichten Ihnen, wie ein *Technischer Projektleiter* Ihr Team von der Außenwelt abschirmt, so dass Sie nur die Informationen bekommen, die Sie wirklich brauchen. Nutzen Sie *Die Liste*, um Ihre eigene Arbeit zu planen und Ihr Team auf dem richtigen Weg zu halten. Wird in Ihrem Team zu wenig kommuniziert? Wissen Sie nicht, wer was tut? Dann helfen Ihnen *Tägliche Besprechungen* dabei, die Arbeit der Team-

mitglieder zu koordinieren und die Kommunikation im Team zu fördern. Kurze *Code-Reviews* helfen Ihnen, von den Fachkenntnissen Ihrer Kollegen zu profitieren bzw. Ihr eigenes Fachwissen mit ihnen zu teilen. Und vergessen Sie nicht, die übrigen Teammitglieder über die Ergebnisse des Reviews zu informieren, indem Sie *Änderungsnachrichten* zu den Quelltextänderungen versenden.

Prozess

Kein Buch über Softwareentwicklung wäre komplett ohne die Vorstellung des persönlichen Vorgehensmodells des Autors – und dieses Buch macht da keine Ausnahme. Wir haben also einen Abschnitt über etwas eingefügt, das wir *Leuchtspurentwicklung* nennen. Bei dieser Vorgehensweise erstellen Sie ein durchgehend arbeitsfähiges System, weitgehend ohne Funktionalität, und fügen dann die fehlenden funktionalen Teile hinzu. Damit können Sie große Projekte gut aufteilen und Teams an den einzelnen Teilen parallel arbeiten lassen. Es eignet sich auch gut zum automatischen Testen.

Häufige Probleme und wie man sie behebt

Abschließend zeigen wir häufig auftretende Probleme – und Warnzeichen, an denen man sie erkennt – und geben Ratschläge, wie sie mit Hilfe der in diesem Buch besprochenen Werkzeuge, Techniken und Verfahren behoben werden können. Auf zahlreiche dieser Probleme sind wir selbst im Laufe der Jahre gestoßen. Einige haben wir gelöst, bei anderen haben wir erst im Nachhinein eine Lösung gefunden (hinterher ist man immer schlauer). Unsere Erfahrungen werden es Ihnen hoffentlich ersparen, dieselben Fehler zu machen.

Was fehlt?

Personalmanagement und Anforderungsanalyse sind zwei Gebiete, die wir nicht behandeln. Manche sehen Werkzeuge, Techniken und Verfahren als die wichtigsten Teile eines Projekts an. Wie man ein großartiges Team zusammenstellt und hält, wäre dagegen ein Thema für ein eigenes Buch (oder eine Reihe von Büchern!). Wir konzentrieren uns stattdessen darauf, die Fähigkeiten zu nutzen und zu erweitern, über die Ihr Team bereits verfügt.

Das Herausarbeiten von Produktanforderungen ist ein weiteres großes Feld. Es gibt viele Möglichkeiten, Anforderungen zu sammeln, von Notizzetteln bis zu komplizierten formalen Systemen. Diesem großen Thema könnten wir in einem einzigen Kapitel niemals gerecht werden. Wir haben uns stattdessen für die Darstellung von Konzepten entschieden, die flexibel mit veränderlichen Anforderungen umgehen können, unabhängig von der Herkunft der Anforderungen. Die Konzepte in diesem Buch lassen sich sowohl an Projekte anpassen, deren Anforderungen sich nie ändern, als auch an solche, deren Anforderungen sich ständig wandeln. Sie können diese Konzepte also unabhängig davon nutzen, ob Sie Ihre Anforderungen aus einem Stapel Karteikarten oder einem 10.000-Seiten-Vertrag entnehmen.

Wir haben versucht, die Darstellungen so allgemein wie möglich zu halten, so dass Sie sie in jeder Firma und mit jeder Technologie nutzen können. Deshalb haben wir auch keine Abschnitte zu Erstellung von Installationsprogrammen und zu Werkzeugen zur Quelltextoptimierung aufgenommen.

1.4 Wie geht's weiter?

Wir hoffen, dass Sie die hier vorgestellten Konzepte und Angewohnheiten so nutzen, wie sie erarbeitet wurden (d.h. pragmatisch). Lesen Sie sie und probieren Sie sie aus. Behalten Sie das, was in Ihrer Umgebung funktioniert, und verwerfen Sie den Rest.

Ermitteln Sie nach jedem Abschnitt, ob Sie das Konzept bereits verwenden. Wenn nicht, dann lesen Sie *Wie fange ich an?*. Wenn Sie das Konzept verwenden, lesen Sie *Mache ich es richtig?* oder *Warnzeichen*, um sicherzustellen, dass Sie sich auf dem richtigen Weg befinden.

1.5 Wie sollte ich dieses Buch lesen?

Wie Sie an dieses Buch herangehen, hängt von Ihrer Rolle im Projekt ab. Wenn Sie als Entwickler oder Tester arbeiten, werden Sie an das Buch natürlich anders herangehen als Ihr Teamleiter; Sie werden jedoch in beiden Fällen aus diesem Buch eine Menge Nutzen ziehen.

Sie sind Entwickler oder Tester

Wenn Sie an vorderster IT-Front tätig sind, lesen Sie dieses Buch von vorne bis hinten. Jeder Abschnitt enthält praktische Konzepte, die Sie täglich nutzen können, sei es als einzelner Mitarbeiter oder als Teamleiter. Entwickler, die selbst ohne Leitungsfunktion sind, überspringen oft Abschnitte, deren Schwerpunkt auf dem Team liegt. Das ist keine gute Idee. Das Entwicklungsumfeld eines Teams besteht meist aus einer Sammlung von Werkzeugen, die Teammitglieder direkt angefordert oder mit denen sie Erfahrung haben. Bringen Sie in Erfahrung, welche Werkzeuge, Techniken und Verfahren Ihr Unternehmen positiv beeinflussen und unterbreiten Sie stichhaltige Gründe für jede Anforderung, die Sie stellen. Schon oft haben wir von Entwicklern als Begründung für ein bestimmtes Werkzeug gehört, dass es „einfach das richtige“ ist. Mit diesem Argument überzeugen Sie das Management nie, es ist sogar kontraproduktiv. Seien Sie sich sicher, dass ein Vorschlag dem Team nutzen kann, bevor Sie ihn unterbreiten.

Welche Anfrage würde Sie überzeugen? „Wir brauchen das Versionsverwaltungssystem von Acme Code Systems, weil es einfach eine gute Sache ist und es jeder einsetzt. Es gilt als *best practice*!“ oder „Wir sollten ein Versionsverwaltungssystem haben, weil wir damit auf frühere Auslieferungen zugreifen, bestimmte Quelltextänderungen rückgängig machen und an mehreren parallelen Entwicklungszweigen gleichzeitig und gefahrlos arbeiten können. Es ist die einfachste Möglichkeit, unsere Entwicklungsinvestitionen zu schützen. Acme Code Systems stellt ein großartiges Produkt her,

das wir in Betracht ziehen sollten. Joe und ich haben es in den letzten Monaten eingesetzt und es hat unsere Produktivität entscheidend beeinflusst. Hier ist eine Aufstellung, wie es uns geholfen hat.“

Sie sind Teamleiter

Nutzen Sie dieses Buch, um ein Audit der Arbeitsumgebung und Abläufe Ihres Teams durchzuführen. (Sie tun das bereits von Zeit zu Zeit, nicht wahr?) Nutzen Sie die Gelegenheit, um zu hinterfragen, wie Ihr Team arbeitet. Haben Sie eine grundlegende Sammlung von Werkzeugen, die Ihre wesentlichen Erfordernisse abdecken? Führt die Arbeitsweise Ihres Teams zu einem stabilen Produkt und erfahrenen Entwicklern? Haben Sie saubere, wohldefinierte Arbeitsabläufe?

Berücksichtigen Sie bei der Beurteilung auch die Relevanz der einzelnen Elemente. Nutzen Sie Werkzeuge oder Verfahren, die einst gepasst haben, heute aber nicht mehr effektiv sind?

Kennen Sie die Geschichte von der Frau, die beim Braten eines Schnitzels immer zuerst ein Drittel abschnitt und wegwarf? Als sie gefragt wurde, warum sie das tat, meinte sie, dass ihre Mutter das immer so gemacht hat. Als man ihre Mutter fragte, sagte sie, dass ihre Mutter das immer so gemacht hat. Als man Großmama schließlich danach fragte, gab sie zu, dass sie in ihrer Jugend keine Pfanne besaß, die groß genug für ein ganzes Schnitzel war, so dass sie immer einfach ein Ende abschnitt; das wurde ihr dann zur Gewohnheit.

Versichern Sie sich, dass Ihre Gewohnheiten auf Ihren derzeitigen Bedürfnissen basieren, nicht auf denen des letzten Projekts oder den Schrullen von Großmama.

Stellen Sie sicher, dass Ihr Team Zugang zu den Werkzeugen, Techniken und Verfahren hat, die es *heute* benötigt. Zu wissen, was funktioniert und weshalb, ist die einzige Möglichkeit, Ihr Team effektiv zu führen. Jeder Abschnitt enthält Tipps, die Ihnen beim Start helfen, und Warnzeichen, die Sie auf Probleme aufmerksam machen, bevor diese außer Kontrolle geraten.

Sie sind Manager (oder Kunde)

Das höhere Management kann die Arbeit eines Teams erheblich beeinflussen, einfach indem es die richtigen Informationen anfordert. Dieses Buch zeigt Ihnen wesentliche Elemente, die Ihr Team nutzen sollte, und welche gezielten Fragen Sie stellen sollten. Wenn Sie z.B. nach einer Liste der Fehlerbehebungen im letzten Release fragen, teilen Sie damit mit, dass dieser Information nachgegangen werden soll. Halten Sie beim Lesen jedes Abschnitts nach Arbeitsergebnissen (engl. *deliverables*) Ausschau, die Sie von Ihren Teamleitern verlangen; Sie können so die Entwicklungsrichtung vorgeben. Seien Sie mit diesen Anforderungen aber vorsichtig – Sie wollen keine bürokratische Fleißarbeit generieren; Sie wollen mit sorgfältig platzierten Anfragen leiten.

Arbeitsergebnisse

Da Sie nicht tagtäglich programmieren, werden Sie das jeweilige *Wie fange ich an?* vielleicht nur überfliegen; Sie werden jedoch das *Was* und *Warum* jedes Themas verstehen wollen.

Einzelpersonen bilden ein Team

Nahezu jedes Konzept in diesem Buch wurde von Teammitgliedern, ganzen Teams *und* Managern angewandt. Meist ist es ein Teammitglied, das ein Verfahren zuerst nutzt, es für gut befindet und es dann im Team verbreitet. Wir haben dies wiederholt selbst getan und es oft bei anderen beobachtet – und Sie können das auch. Hier ist eine Geschichte über jemanden, der genau das getan hat.

Agilität durch Technik bei CafePress.com

von Dominique Plante und Justin McCarthy

Als wir Anfang vergangenen Jahres begannen, für CafePress.com zu arbeiten, war das Management davon begeistert, auf Agilität zu setzen. Der Arbeitsumgebung der Entwickler fehlten jedoch die grundlegenden Systeme, um Änderungen gesichert vorzunehmen.

Betrachten wir das Projekt genauer: In Erweiterung zum Kerngeschäft von CafePress sollten die Anwender Werbeartikel (T-Shirts, Kaffeetassen, etc.) selbst entwerfen und kaufen können. Es war der erste Anlauf, Geschäftslogik und Datenschicht sauber zu trennen, so wie es bereits für die Web-Präsentationsschicht geschehen war.

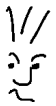
Die beiden neuen Schichten wurden mit Test-First entwickelt, für die Entwicklertests wurde NUnit eingesetzt. Gleichzeitig führten wir NAnt ein, was auch für die Klassen der Präsentationsschicht einen wiederholbaren Build-Prozess sicherstellt. Außerdem installierten wir CruiseControl.NET, um eine kontinuierliche Integration (insbesondere Kompilieren und Ausführen der Unit-Tests) bei jedem Einchecken in unser Subversion-Repository zu bekommen. Als Feinschliff haben wir eine Workstation namens *Chicken Little* eingerichtet und auf ihr CCTray konfiguriert, um so den Status des letzten Build-Vorgangs zu beobachten. Dieser Rechner ist für alle Teammitglieder gut sichtbar (und macht sich auch akustisch bemerkbar).

Durch das problemlose Zusammenspiel von Subversion, CruiseControl.NET, NAnt und NUnit erhielten wir eine gute technische Umgebung, ohne dafür eine komplizierte Werkzeugauswahl oder schwierige Kaufentscheidungen zu durchlaufen. Besser noch, all diese Änderungen gingen auf Vorschläge der Entwickler zurück, keine war eine Forderung des Managements.

Seit der Einführung dieser Automatisierungen ist unser Team gewachsen und viele neue Mitglieder haben die Qualität unserer Tests und der Automatisierungswerkzeuge weiter verbessert. Zu den jüngsten Erweiterungen gehören das vollständig skriptgesteuerte Einrichten einer Entwicklungsumgebung und das automatische Installieren der Anwendung in der Test-Umgebung, wobei `nant test` noch immer der am häufigsten ausgeführte Befehl ist.

Als wir diese technischen Hilfen noch nicht hatten, drehte sich unsere Kommunikation meist um die Suche nach Fehlern und die Bekanntmachung von API-Änderungen. Seitdem wir CruisControl.NET einsetzen, werden Integrationsprobleme sofort aufgedeckt. Entwickler verstehen die Wichtigkeit eines fehlerfreien Build und setzen sich auch dafür ein. Wer will schon am Arbeiten gehindert werden, weil es irgendwo irgendwelche Integrationsprobleme gibt? Die installierte automatische Unterstützung führt dazu, dass sich die Gespräche der Entwickler jetzt sehr viel mehr um Softwaredesign und Implementierung drehen; der Frust über Unzulänglichkeiten der Arbeitsumgebung hat deutlich abgenommen.

All unsere Anstrengungen trugen dazu bei, sofort getesteten Quelltext zu liefern, und unsere Anfangsinvestition zahlt sich jedes mal aus, wenn Chicken Little anschlägt.



Joe fragt:

„Was ist Agilität?“

Agilität bezieht sich auf die Fähigkeit eines Softwareteams, sich verändernden Situationen schnell anzupassen. Manchmal ist dazu ein Redesign notwendig; zu anderen Zeiten ist damit das schnelle Reagieren auf Fehler oder das Aneignen einer neuen Technologie gemeint. Im Allgemeinen interessieren sich agile Teams mehr für Ergebnisse als für Bürokratie.

Sie können unter `http://www.agilemanifesto.org/` mehr über agile Software erfahren.

Das folgende Zitat von dieser Webseite fasst die agile Betrachtungsweise ziemlich gut zusammen:

„Wir entdecken bessere Wege zur Entwicklung von Software, indem wir selbst Software entwickeln und anderen dabei helfen. Durch diese Tätigkeiten haben wir gelernt, dass uns

- Personen und Interaktionen wichtiger sind als Prozesse und Werkzeuge,
- funktionierende Software wichtiger ist als umfangreiche Dokumentation,
- Zusammenarbeit mit Kunden wichtiger ist als Vertragsverhandlungen,
- Reagieren auf Veränderungen wichtiger ist als Festhalten an einem starren Plan.

Natürlich sind auch die Dinge rechts wichtig, aber im Zweifelsfall schätzen wir die linken höher ein.“