PEARSON Education

ADDISON-WESLEY

Kent Beck
Martin Fowler

Extreme Programming planen

ADDISON-WESLEY

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England
Don Mills, Ontario • Sydney • Mexico City
Madrid • Amsterdam

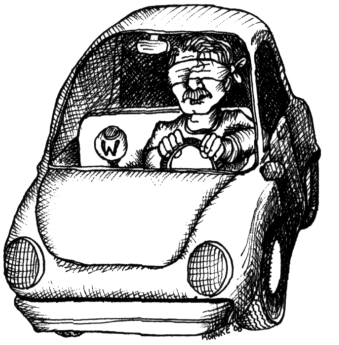
Kapitel 3

Software fahren

»Natürlich bin ich ein hervorragender Fahrer.«
Raymond in »Rain Man«

Wir benutzen das »Fahren« als eine Metapher für die Softwareentwicklung. Fahren bedeutet nicht, das Auto in eine Richtung zu lenken und darauf zuzusteuern, sondern es geht darum, viele kleine Richtungsänderungen vorzunehmen.

Die Metapher des Fahrens wurde schon in *Extreme Programming Explained* eingeführt, wegen ihrer großen Bedeutung für XP wollen wir aber noch einmal auf sie zurückkommen. Fall Sie *XPE* gelesen haben, werden Sie dieses Kapitel nur lesen wollen, um zu sehen, ob wir es irgendwie hinbekommen haben, die Geschichte ein bisschen dramatischer zu gestalten.



Es war ein wundervoller, sonniger Tag. Kent und seine Mutter fuhren entlang eines geraden Teilstücks des Interstate 5 nahe Chico. Er war ungefähr zwölf Jahre alt.

»Es wird Zeit, dass du lernst, wie man Auto fährt«, sagte seine Mutter.

»Wirklich?« In Kents Brust brodelte die Aufregung.
»Ja. Ich möchte jetzt, dass du den Wagen gerade zwischen den Streifen hältst«, meinte seine Mutter.

»Das kann ich.«

Kent richtet äußerst vorsichtig den Stern des beigefarbenen Mercedes 240D schnurgerade auf den Horizont ein. Seine Augenbrauen heben sich ein bisschen, da dieses Autofahren doch in Wirklichkeit einfach zu sein scheint. Nach kurzer Zeit bleibt sein Blick an einem Straßenschild haften.

ggggrrrrrrccccchhhh (na, versuchen Sie mal ein Geräusch aufzuschreiben, das Reifen auf Schotter und einen jugendlichen Aufschrei kombiniert). Kents Mund wird ganz trocken, sein Herz beginnt zu pochen.

»Also gut«, sagt seine Mutter, während sie ein Grinsen unterdrückt, »so fährt man kein Auto. Autofahren bedeutet nicht, den Wagen in eine Richtung zu lenken. Es geht darum, ständig kleine Korrekturen durchzuführen. Man lenkt ein bisschen dahin, ein bisschen dorthin. Dahin, dorthin, solange man fährt.«

Sie fahren keine Softwareentwicklung, indem Sie Ihr Projekt in die richtige Richtung lenken (den Plan erstellen). Sie tun es, indem Sie bemerken, dass Sie in eine Richtung abdriften, und in die andere Richtung lenken. Dahin, dorthin, solange Sie die Software entwickeln.

Einmal prägte ein sehr lautstarker XP-Gegner den Ausspruch: »Achtung ... Schießen ... Zielen!« Diese Aussage war klar abwertend. Wie kann man ein Ziel treffen, wenn man nicht zuerst zielt? Der Punkt jedoch ist, dass wir gar nicht vorhaben, ein Ziel zu treffen. Stattdessen versuchen wir, den Nutzen eines Prozesses zu maximieren.

Die Metapher »Fahren« hilft uns ein weiteres Mal. Das Erste, was Sie machen, wenn Sie in den Wagen steigen, ist nicht, am Lenkrad zu drehen, so dass es auf Ihr Ziel zeigt. Ihre erste Handlung ist gewöhnlich, die Zündung zu betätigen. Tatsächlich hat die anfängliche Bewegungsrichtung auch nicht viel mit Ihrem Ziel zu tun, sondern vielmehr mit den örtlichen Gegebenheiten. Sie werden vielleicht aus Ihrer Garage fahren wollen, bevor Sie sich auf den Weg nach Peoria machen. Obwohl Sie möglicherweise ein Ziel im Sinn und eine Route geplant haben, sind diese doch veränderbar. Das Radio könnte Sie vor Staus warnen, was Sie veranlassen würde, Ihre Route zu ändern. Ihr Ehepartner könnte Sie anrufen und bitten, dass Sie etwas Milch kaufen. Dies würde Sie dazu bringen, Ihr Ziel dementsprechend zu modifizieren.

Softwareentwicklung ist ein Prozess. Sie kann gut laufen oder schief gehen. Um sie am Laufen zu halten, müssen wir sie stets lenken. Um sie zu lenken, müssen wir ihre Richtung abschätzen, diese mit unseren Vorstellungen vergleichen und dann vorsichtige Änderungen vornehmen. So kann gute Projektleitung durch diesen Satz charakterisiert werden: »Achtung ... Schießen ... Zielen ... Zielen ... Zielen ... Zielen ... Zielen«

Kapitel 4

Das Gleichgewicht wahren

»Ich möchte Babys haben.«
»Aber du kannst keine Babys haben. Du bist ein Mann.«
»Unterdrücke mich bitte nicht.«
Die Judäische Volksfront in »Das Leben des Brian«

Unser Planungsprozess beruht auf der Trennung der Rollen von Geschäftsleuten und Softwareverantwortlichen. Dies gewährleistet, dass die Geschäftsleute all die Geschäftsentscheidungen und die Softwareverantwortlichen all die Softwareentscheidungen fällen.

Der Schlüssel des Projektmanagements besteht darin, das Gleichgewicht zwischen den Geschäftsleuten und den Programmierern zu wahren. Macht man dies richtig, verfügt das Management des Softwareprojekts über:

- Geschäftsleute, die Geschäftsentscheidungen fällen,
- technisches Personal, das technische Entscheidungen fällt.

Klingt dies nicht wie eine Binsenweisheit? Natürlich fällen Geschäftsleute Geschäftsentscheidungen.

Wie wäre es hiermit?

»Wir denken, dass es sechs Monate dauern wird, dieses System zu entwickeln.«

»Sie haben drei Monate.«

»Was können wir weglassen?«

»Nichts. Es muss alles beinhalten.«

Zu schätzen, wie lange das Programmieren wahrscheinlich dauern wird, ist gänzlich eine technische Entscheidung. Die Programmierer legen ihre Erfahrung in die Waagschale, mischen ihr Verständnis unter, welche Unterschiede das neue System aufweist, und würfeln dann eine Zahl aus. Nein, um ehrlich zu sein, erfolgt dies ein bisschen systematischer (siehe Kapitel 12). Wie schwierig das

Schätzen auch sein mag, die Programmierer können besser raten als irgendjemand sonst. Dementsprechend ist die Schätzung eine technische Entscheidung.

In dem vorherigen Dialog, der aus einem tatsächlichen abgebrochenen Projekt stammt, wurde die Schätzung von einem Geschäftsmann aus geschäftlichen Gründen durchgeführt. Die daraus folgende Annahme, dass die Arbeit drei Monate dauern würde, kann unmöglich besser gewesen sein als die der Programmierer, dass sie sechs Monate bräuchten. Jedoch ging, ohne dass eingegriffen wurde, jeder auf Basis von ungeheuer ungenauen Informationen zum nächsten Schritt der Planung über. Der resultierende Plan, wie preiswert, flexibel und verständlich er auch gewesen sein mag, war, um es einfach auszudrücken, Quatsch.

Wenn auch Geschäftsleute gelegentlich technische Entscheidungen fällen mögen, mischt sich das technische Personal ja wenigstens nicht in Geschäftsentscheidungen ein.

Aha, was ist hiermit?

> »Ich habe zehn Dinge zu erledigen. Ich weiß, dass ich nur fünf davon schaffen werde. Ich werde mich zuerst dieser DCOM/COBRA-Infrastruktur widmen. Das scheint toll zu sein.«

Halt! Es ist eine geschäftliche Entscheidung, die relativen Prioritäten von Funktionen zu bestimmen. Ob ein Merkmal der Benutzerschnittstelle wichtiger ist als eine weitere Berichtsspalte, ist eine Geschäftsentscheidung. Die Geschäftsleute nehmen her, was sie vom Markt wissen, vermischen dies mit ihrer Erfahrung aus ähnlichen Systemen und würfeln dann eine Zahl aus. Nein, um ehrlich zu sein, kann auch dies ein bisschen systematischer verlaufen (manchmal ist es so, manchmal nicht). Wie schwierig die Entscheidung auch sein mag, welche Funktion als Nächstes dran ist, die Geschäftsleute können diese Entscheidung viel eher fällen als die Programmierer.

Geschäftliche Entscheidungen sind:

- Termine
- Umfang
- Prioritäten

Technische Entscheidungen sind:

- Aufwandsschätzungen

Wenn die richtigen Personen die Entscheidungen fällen, wird die Planung so gut wie möglich verlaufen. Wir werden in der Lage sein, mit unseren Unfällen umzugehen. Wir werden dies tun, indem wir die Anzahl dieser Unfälle so weit wie möglich minimieren, mehr über sie herausfinden und möglichst viele Optionen möglichst lange offen lassen.

Das politische Gleichgewicht zu wahren, mag für einen einfachen Projektmanager wie eine große Aufgabe erscheinen. Wenn die Welt dies nach einigen Jahrtausenden gemeinsamer Anstrengung zwischen den Nationen nicht schafft, welche Chance haben dann Sie?

Das Gleichgewicht zu wahren, ist nicht so schlimm, wie es sich anhört. Schaffen Sie eine einfache Sammlung von Regeln, die dafür sorgen, dass das technische Personal die technischen und die Geschäftsleute die geschäftlichen Entscheidungen treffen.

Der Kunde

Wir sprechen in XP viel über den Kunden. Mit *Kunde* meinen wir die Person, die die geschäftlichen Entscheidungen trifft. Oft haben Sie nicht wirklich nur einen einzigen Kunden, sondern Benutzer, Geschäftsleiter, Unternehmen – alle Arten von Personen, die Kunden sind. Wenn Sie Standardsoftware erstellen, können Sie Tausende von Kunden haben. Wenn XP funktionieren soll, muss der Kunde jedoch mit einer einzigen Stimme sprechen. Manche Leute nennen solch ein Wesen Produktmanager oder Herrscher über die Anforderungen. Wir benutzen den Ausdruck *Kunde*, weil es dieser ist, der durch jene Person vertreten wird.

Viele Planungsprozesse betrachten den Kunden als eine Art körperloses Wesen außerhalb der Softwareentwicklung, das Anforderungen zur Verfügung stellt. Sie interpretieren, quälen sich, veranstalten JAD-Sitzungen – aber der Kunde befindet sich außerhalb des Teams.

XP sieht das nicht so. In XP geht die Planung davon aus, dass der Kunde sehr wohl Teil des Teams ist – sogar wenn er für eine andere Firma und der Rest des Teams für einen Auftragnehmer arbeitet. Der Kunde muss Teil des Teams sein, da seine Rolle viel zu wichtig ist, als dass man sie einem Außenseiter überlassen könnte. Jedes (XP-) Projekt wird schief gehen, wenn der Kunde nicht in der Lage ist zu steuern.

Somit stellt die Aufgabe des Kunden eine große Verpflichtung dar. Die besten Talente, Technologien und Prozesse der Welt werden nichts nutzen, wenn der Kunde nicht die Erwartungen erfüllt. Leider ist dies auch eine Rolle, für die wir

keine genauen weisen Ratschläge geben können. Immerhin sind wir nur Computertrottel und keine Geschäftsleute. Aber hier ist, was wir sicher wissen.

Einen Kunden finden

Da der Kunde eine solch entscheidende Rolle einnimmt, ist es wichtig, jemanden zu finden, der sie gut spielen wird. Ein guter Kunde

- kennt den Problembereich, da er in ihm arbeitet und auch versteht, wie er funktioniert (das ist nicht immer dasselbe),
- kann mit Hilfe der Entwickler verstehen, wie die Software im Problembereich einen Geschäftswert zur Verfügung stellen kann,
- ist bestimmt, regelmäßig einen Gegenwert abzuliefern, und scheut sich nicht davor, lieber zu wenig als gar nichts abzuliefern,
- kann Entscheidungen fällen, was gerade benötigt wird und was später,
- ist bereit, unmittelbare Verantwortung für den Erfolg oder Misserfolg eines Projekts zu übernehmen.

Verantwortung für den Erfolg oder Misserfolg des Projektes zu übernehmen, scheint der schwierigste Teil der Aufgabe zu sein. Es mag für den Kunden in gewisser Weise angenehm sein, Abstand zum Team zu halten – und zwar am besten hinter einen Berg von Anforderungsdokumenten. In XP wird dies nicht funktionieren. Wenn Sie sich verfahren, ist dies nicht die Schuld des Autos, sondern des Fahrers.

Für einen Kunden ist das kniffligste an XP, sich an den Rhythmus der regelmäßigen Auslieferung zu gewöhnen. Viele Prozesse wollen vom Kunden gleich alles wissen, was er gerne hätte. Statt dessen verlangt der Kunde bei XP nur so viel, dass gerade ein Gegenwert vorhanden ist, und das Team erfüllt diese Vorgabe. Wenn Sie keinen Kunden finden, der auf diese Weise mit Ihnen zusammenarbeiten möchte, sollten Sie sich gar nicht erst auf XP einlassen.

Richtlinien für Kunden

Wenn Sie ein Kunde sind und dies lesen, kommen nun einige wichtige Dinge, die Sie sich merken sollten.

Fragen Sie sich jederzeit: »Was ist die wertvollste Funktion, die wir als Nächstes haben wollen?« Langfristige Planungen können Spaß machen, aber es sind die regelmäßigen, kleinen Lieferungen, die Ihnen das Geld einbringen.

Vertrauen Sie den Schätzungen der Entwickler, denken Sie jedoch daran, dass es nur Schätzungen sind, die sich nicht bewahrheiten werden. Das Abschätzen der Softwareentwicklung ist schwierig; die Entwickler geben ihr Bestes und werden immer besser.

Verschieben Sie niemals einen Termin. Dies ist eine der schlimmsten Gewohnheiten in der Softwareentwicklung. Sie verschieben nur einen oder zwei und kommen davon nach einer Weile nicht mehr los. Es ist nicht gänzlich gegen die Regeln, einen Termin zu verschieben, es ist nur so, dass die XP-Methode erfordert, dass Sie sich jedes Mal, wenn Sie es tun, einen Finger abhacken.

Stellen Sie mit jedem Versionsschritt kleine, wertvolle Funktionalität bereit und geben Sie diese so oft wie möglich an den wirklichen Kunden weiter. Benutzen Sie Ihre Kreativität, um Mittel und Wege zu finden, wie Sie eine große, neue Funktionalität nehmen und in kleine Teile zerlegen können, um fortlaufend weitere Anforderungen bereitzustellen. Wenn Sie regelmäßig genug Versionsschritte ausliefern, werden Sie nicht lange warten müssen, bis Sie mehr von dem bekommen, was Sie haben möchten.