„Classic Mistakes“

Klassische Fehler in der Softwareentwicklung

Ausarbeitung von

Jan Hoffmeyer

01-IND

Einführung

Der gesamte Bereich der Softwareentwicklung ist ein komplizierter und verstrickter Vorgang. Innerhalb von wenigen Arbeitsschritten, die häufig in der Planung und Ausarbeitung liegen, gibt es genug Gelegenheiten anfangs einfache Fehler zu produzieren, die im späteren Verlauf der Entwicklung das Projekt gefährden können, wenn nicht sogar es zum Scheitern bringen.

„One bad apple can spoil your whole project”. Dieser Ausdruck trifft auch sehr gut auf die Softwareentwicklung zu. Egal wie klein ein Fehler, oder ein Apfel in diesem Zitat, sein mag, er kann das gesamte Projekt gefährden. Deswegen sollte vor allem in der Planung eines Projektes darauf geachtet werden Fehler zu vermeiden.

Ein häufiger Trugschluss offenbart sich, wenn man den Unterschied zwischen einer langsamen und einer schnellen Entwicklung betrachtet. Es genügt bereits ein Fehler, um in den Bereich einer langsamen Entwicklung zu kommen. Um aber eine schnelle Entwicklung zu machen genügt es nicht einfach diesen Fehler nicht zu begehen, sondern überhaupt keinen Fehler zu begehen.

Diese Ausarbeitung beschreibt einige Fehler die gemacht wurden. Die Quelle die zugrunde liegt ist eine Studie aus dem Jahr 1984. In dieser Studie wurden 44 verschiedene Softwareprojekte nach deren Fertigstellung noch einmal analysiert.

„Klassische“ Fehler

Im Verlauf der Softwareentwicklung gibt es verschiedene Bereiche wo sich kleine oder größere Fehler einschleichen. Unter dem Aspekt „Mensch“ wurden Fehler betrachtet, die an Menschen begangen wurden. Während im „Ablauf“ Fehler zum Tragen, die sich mit der Planung und der zeitlichen Abfolge befassen. Im Bereich „Produkt“ werden Fehler offen gelegt, die, wie der Name schon sagt, im Produkt liegen oder dieses betreffen. Und zum Schluss wird noch die „Technologie“ betrachtet. Hier offenbaren sich Fehler in verwendeten Techniken und Werkzeugen für die Programmierung.

Der Mensch

In diesem Abschnitt werden Fehler gezeigt, die sich größtenteils mit dem Menschen befassen und hier direkt auch auf die Psyche.

Untergrabene Motivation

Zu den wichtigsten Punkten in einem Unternehmen gehört die Motivation. Sie ist der Hauptmotor für die Produktivität und die Qualität, der in der Firma angebotenen Produkte. Eine mangelnde oder fehlerhafte Motivierung der Arbeitnehmer kann die Qualität herabsetzen und die Quantität beeinträchtigen.

Die Ursache für diese Erscheinung liegt nicht allzu selten in der mangelnden Konversation zwischen Arbeitnehmer und Arbeitgeber. Als Beispiel wäre hier ein Manager zu nennen. Er beschafft seinem Betrieb und damit auch seinen Arbeitnehmern Arbeit über den Sommer hinaus. Während dieser Zeit wird jedoch eine Urlaubssperre verhängt, so dass das Arbeitspensum erreicht werden kann und der Auftrag fristgerecht erledigt wird. Während dieser Zeit macht aber gerade die Manager Urlaub und bleibt dem „Team“ mehrere Tage fern. Da sich niemand in dem Betrieb getraut hat den Manager über sein Fehlerverhalten in Kenntnis zu setzen, sinkt die Motivation der Arbeitnehmer und der festgesetzte Termin kann nicht eingehalten werden. Versuche des Managers wieder rum zu reißen, scheitern. Damit kommen wir auch zum nächsten Fehler, der manchmal aus der untergrabenen Motivation entsteht.

Hinzufügen von Personen

In der Studie wird dieser Fehler als der klassischste von allen beschrieben. Betrachten wir den oben genannten Manager. Er kommt auch seinem Urlaub zurück und bemerkt, dass der Endtermin nicht mehr eingehalten werden kann. Er stellt weitere Entwickler ein, die den bereits vorhandenen unterstützen sollen. Aber die erhoffte Steigerung der Produktivität bleibt aus. Wo liegen die Gründe? Diese Ursachen für das Ausbleiben der Produktivitätssteigerung sind offensichtlich, werden aber gerade dann begannen, wenn kein Ausweg mehr in Sicht ist. Was passiert mit neuen Entwicklern in einem Team?

- Sie müssen angelernt werden.
- Sie müssen sich in das Projekt einarbeiten
- Sie müssen die Werkzeuge kennen lernen

Alle diese Punkte kosten Zeit. Und gerade das ist eine Sache, die das Projekt, der Manager und die Entwickler nicht haben. Somit erhebt sich das Wunschdenken: Mehr Leute, Schnellere Entwicklung zu einem Trugschluss. Im Endeffekt wird das genaue Gegenteil bewirkt. Das Projekt ist unter Umständen schneller fertig gestellt, wären die zusätzlichen Arbeitskräfte nicht eingestellt worden.

Spannung zwischen Entwickler und Kunde

Häufig entsteht ein gewisses Feld zwischen dem Kunden und den Entwickler. Dieses Feld ist geprägt von Spannungen, die zwischen beiden Parteien liegen. Sie bauen sich erst allmählich auf und sind häufig das Produkt von mangelnder Kommunikation zwischen den Parteien.

Zeitpläne, gewünschte Anforderung oder User Interface sind nur einige Punkte, wo sich Spannungen entwickeln können. Ergebnis dieser Prozesse ist häufig eine Ablehnung des Kunden für Teile des Produktes. Es gibt aber auch Projekt, wo der Kunde das gesamte Produkt ablehnt am Ende.

Die Spannungen, wie sie auch immer entstehen, sind Zeit verbrauchend und ablenkend von der eigentlichen Sache.

Unrealistische Erwartungen

Dieser Fehler baut auf den vorhergehenden „Spannung zwischen Entwickler und Kunde“ auf. Durch die wenige Kommunikation, die eintreten kann, entsteht eine Blase von Hoffnungen in dem Kopf des Kunden. Diese Hoffnungen wirken sehr schnell unrealistisch. Dieser Fehler verlängert zwar nicht den Prozess der Softwareentwicklung, er trägt aber maßgeblich dazu bei.

Mangelnde Benutereinbeziehung

Für ein gutes Produkt gibt es nichts Wichtigeres als die Meinung und die Kritik von den Personen, die es später tag täglich benutzen, die Benutzer. Ein Benutzer ist zuständig für den Erfolg, den ein Produkt erreichen kann. Wenn ein Benutzer, es muss nicht der Auftraggeber sein, das Produkt ablehnt, dann war die komplette Arbeit umsonst. Deswegen sollte schon früh der Benutzer in die Entwicklung einbezogen werden. Er kennt das Einsatzfeld eines Produktes am besten. Ohne dies kann es schnell zu Missverständnissen und der Verwundbarkeit eines Produktes kommen.

Politik über Inhalt

Innerhalb der Studie kristallisierten sich vier verschiedene Orientierungen der Softwareentwicklung heraus. Die politische, forschende, isolierende und die allgemeine Richtung.

Die *Politiker*, wie kann es auch anders sein, versuchen alles zu „managen“. Sie achten mehr auf die Beziehung zu ihren Managern als zu allem anderen.

Die *Forscher* sind auf der Suche nach Grenzen und versuchen alles Neue auszuprobieren und zu ergründen.

Die *Isolationisten* versuchen sich abzugrenzen und blockieren alle Nicht-Mitglieder.

Und dann gibt es noch die *Generalisten*. Sie machen ein bisschen von allem. Kommunikation mit ihren Managern ist genauso wichtig wie die Erforschung neuer Techniken und die Einbeziehung von anderen Teams in das Projekt, um Meinungen und Ideen zu erhalten.

Die Studie zeigte ferner, das im Vergleich zwischen Politikern und Generalisten anfangs kein großer Unterschied ist. Aber nach einem oder einem halben Jahr fielen die Politiker in ein Tal und gerieten an einem toten Punkt. Die Politik über das Ergebnis zu setzen ist verheerend für eine schnell-orientierte Entwicklung.

Wunschdenken

Das Wunschdenken findet man in vielen Softwareprojekten. Es ist auch eine Nebenerscheinung der Spannung zwischen Entwickler und Kunde. Dieses Denken ist nicht nur einfach ein Optimismus. Es ist ein grober Fehler in Planung und Kommunikation zwischen den Parteien. Ein Großer Knall am Ende des Projektes ist häufig das Resultat dieses Fehlers.

Der Ablauf

Fehler in diesem Prozessabschnitt verschwenden sehr oft die Talent und Ideen der Entwickler und erhöhen somit die Produktionszeit. In diesem Abschnitt sind einige der schlimmsten Fehler dargelegt, die die Studie gezeigt hat.

Zu optimistische Zeitpläne

Nicht selten trifft das Phänomen auf, dass unterschiedliche Zeitpläne für ein und dasselbe Projekt aufgestellt werden. Und nicht selten wählt der Kunde den Zeitplan auf, der am optimistischsten aufgestellt wurde. Diese ausgewählten Zeitpläne führen zu einem Projektrahmen der schon am Anfang nicht erfüllt werden kann. Somit ist für die Entwickler ein effektives Planen bereits jetzt schon nicht möglich und wichtige Arbeitsschritte wie zum Beispiel die Analyse und das Design müssen verkürzt werden.

Der Druck, der auf die Entwickler entsteht, äußert sich in der Moral und die Produktivität.

Lieferantenfehler

Vor allem bei großen und schwierigen Projekten werden meistens Teilprojekte an andere Firmen übertragen. Doch gerade diese Übertragung an andere, eröffnet die Möglichkeit, dass sich weitere Fehler einschleichen. Die Auslagerung hat Einfluss auf die Entwicklungszeit des Projektes. Niemand kann vorhersagen wann die Zulieferfirma mit dem Teilprojekt fertig ist. Auch die Qualität kann eine andere sein, als vorher angenommen. Zu guter letzt gibt es noch die Spezifikation, die, wenn sie nicht ausreichend dokumentiert ist, zu unnötiger Nacharbeit führt. All dies verlangsamt den Prozess der Softwareentwicklung anstatt ihn zu beschleunigen.

Fuzzy Front End

Als Fuzzy Front End wir die Zeit bezeichnet wo es um die Genehmigung und die Finanzierung eines Projektes geht. Es ist nicht selten der Fall, das es Monate, manchmal sogar Jahre dauert bevor ein Projekt genehmigt und finanziert wird. Die Gefahren die dadurch entstehen sind ein aggressiver Zeitplan, aus dem wider rum weitere Fehler entstehen und ein Produkt was „Out of date“ ist.

Streichung von Entwicklungsschritten

Zeitnot ist einer der Hauptfaktoren der im Entwicklungsprozess zu Fehler führt. Einer davon ist die Streichung von Entwicklungsschritten. Es werden dann Schritte herausgestrichen die einfach wichtig für die Entwicklung sind. Als „begehrtes“ Ziel für die Streichung sind Schritte, die kein Code produzieren. Darunter fallen unter Anderen die Anforderungsanalyse, die Architektur und das Design.

Aufgrund dieser Streichung kommt es zu einem so genannten „jumping into coding“. Also dem reinen drauf losprogrammieren ohne vorher zu überlegen wie es später auszusehen hat.

Unzureichendes Design

Das ist ein Folgefehler der Streichung von Entwicklungsschritten. Wenn zu wenig Zeit für das Design eingeplant oder die geplante Zeit in andere Schritte verschoben wird, so kommt es zu einem befriedigenden Design, welches nicht anpassungsfähig ist. Dabei ist das Design meistens wichtiger als die Qualität.

Streichung der Qualitätszusicherung

Ein weiterer Schritt in der Entwicklung, der häufig dem Rotstift zum Opfer fällt, ist die Qualitätszusicherung. Die Kontrolle von Design und Code ist meistens das Ziel dieser Streichung. Es erfolgen keine Testplan und auch Funktionstests bleiben außen vor.

Was man durch diese Streichung einspart hängt man meistens am Ende wieder ran. Man sagt, wenn man 1 Tag gespart hat, so muss man 3 bis 10 Tage dranhängen.

Überhastete Annäherung

Wenn das Projekt langsam zu Ende geht und man feststellt, dass die Zeit nicht reicht, kommt es in den meisten Fällen zu einer überhasteten Annäherung. Kurz vor Ende des Projektes wird eine übereilte Fertigstellung angestrebt. In der Eile werden dann Punkte wie zum Beispiel Performance, Dokumentation und Hilfesystem fertig gestellt. Aber häufig passiert dies auch zu oft. Was wider rum Zeit verschwendet und die Planung verlängert.

Code-like-hell Programmierung

In vielen Firmen sitzen Entwickler, die nur zu gern mal zeigen möchten welche Fähigkeiten sie haben. So kommt es nicht oft zu einer All-as-you-go Programmierung. Alles was möglich ist, wird programmiert. Egal ob es nun benötigt wird oder nicht. Diese Kombination aus Code-and-Fix Programmierung vereint mit einer ehrgeizigen Planung funktioniert nicht.

Das Produkt

In Gold gepresste Bedürfnisse

Der Kunde kommt meistens zum Entwickler mit einem Sack voll von Bedürfnissen. Erwünscht sich zum Beispiel das sein Produkt eine enorme Performance aufzuweisen hat, obwohl dies absolut nebensächlich für den Einsatzbereich ist. Da nun aber der Kunde König ist, werden diese Bedürfnisse gestillt. Leider werden dann andere, wichtigere Merkmale vernachlässigt, wie zum Beispiel das Marketing oder die Entwicklung selbst. Diese Punkte können dann auch in der Planung nicht berücksichtigt werden.

In Gold gepresste Entwickler

Wer kennt sie nicht die Entwickler, die von neuen Technologien begeistert sind. Diese Faszination kann aber Gift sein für die aufgestellte Planung. Viele Entwickler probieren neue Techniken und Merkmale an den Produkten aus, ohne sie vorher richtig zu kennen oder deren Tücken zu wissen. Dies kann die Planung entscheidend verlängern.

Forschungsorientierte Entwicklung

Seymour Cray sagte einmal: “Entwicklung in mehr als 2 Gebieten zur gleichen Zeit birgt ein hohes Risiko von Fehlern“. Vor allem in der Erstellung von neuen Algorithmen verbergen sich enorme Fehlerpotentiale. Und auch gerade diese Erstellung macht sehr schnell aus der Softwareentwicklung eine Softwareforschung, und diese ist im Gegensatz zur Entwicklung nicht vorhersehbar. Gerade die Verwendung von State-of-the-Art Algorithmen macht eine sorgfältige Planung zur einer höchste spekulativen.

Die Technologie

Das Silberkugel Syndrom

Unter vielen Entwicklern gibt es einen Glauben. Ein Tool kann Wunder bewirken. Aber gerade dieses Wunder bleibt viel zu oft aus. Denn gerade das Wissen über das Tool ist begrenzt. Häufig gibt es zu wenig Information über den Nutzen und das Wirken. Es kommt dann einfach zum Einsatz und dies entpuppt sich dann als einer Enttäuschung, die zwangsläufig passieren musste.

Überschätzte Einsparung durch neue Praktiken

Neue Praktiken und Methoden verheißen immer einen enormen Entwicklungsschub. Dort dies ist selten. Gigantische Sprünge der Produktivität kommen so gut wie nie vor. Denn was eines meistens vergessen wird, ist die Zeit um diese neuen Praktiken zu erlernen.

Aber auch die neuen Praktiken sind nicht ganz frei von Tücken und so bringen auch sie nicht selten Risiken mit, die die Planung verlängern können.

Als eine Praktik wäre zu nennen, die Verwendung von Codesegmenten aus anderen Projekten. Dies mag eine einfache und schnelle Möglichkeit sein, ein Projekt vor ran zutreiben, verbirgt aber ein Risiko was häufig unterschätzt wird: die Einarbeitung in fremden Code bei Fehlersuche und Optimierung.

Toolwechsel während des Projektes

Dies ist eine der ältesten Praktiken die angewendet wird. Doch nur selten ist der Einsatz von neuen Tools während eines Projektes sinnvoll. Auch der Versionsupdate bringt nicht immer den erhofften Erfolg. Denn es gibt genug Punkte, die den Vorteil schmälern oder ganz zerstören. Zum einen wäre die Lernkurve zu nennen, die entsteht um die neuen Features und Funktionen kennen zulernen. Dann die Neuarbeit am alten Code um die neue Version optimal einzusetzen und zum Schluss ist auch nicht jede Version fehlerfrei, was somit bedeutet, das es auch noch weitere unvorhersehbare Fehler auftreten können.

Mangel automatischer Source Code Control

Dies ist wohl ein überflüssiges Risiko was sehr schnell und einfach zu lösen ist. Doch dies passiert nur sehr selten und tritt häufig in kleineren Projekten oder Firmen auf. Es erfolgt meistens eine manuelle Koordination des Codes. Dies birgt aber die Gefahr der Verwendung von alten Code-Segmenten. Im Durchschnitt gibt es etwa 10% Code-Änderung in einem Projekt mit SCC pro Monat. Ohne SCC beläuft sich diese Zahl auf mehr als 50%.

Zusammenfassung

Der Mensch

Untergrabene Motivation	Hinzufügen von Personen	Mangelnde Projekt-unterstützung	Wunschdenken
Schwaches Personen	Lautes Büro	Mangelnder Teilhabereinkauf	
Unkontrollierte Problemarbeiter	Spannung zwischen Entwickler und Kunde	Mangelnde Benutzereinbeziehung	
Helden	Unrealistische Erwartungen	Politik über Inhalt	

Legende

vorgestellt

außerdem noch vorhanden

Der Ablauf

Zu optimistische Terminpläne	Fuzzy Front End	Überhastete Annäherung
Unzureichendes Risikomanagement	Streichung von Entwicklungsschritten	Vernachlässigung wichtiger Aufgaben
Lieferantenfehler	Unzureichendes Design	Planen zum späteren Auffangen
Unzureichende Planung	Streichung der Qualitätszusicherung	Code-like-hell Programmierung
Planungsabbruch unter Druck	Unzureichendes Management	

Legende

vorgestellt

außerdem noch vorhanden

Das Produkt

Goldgepresste Bedürfnisse	Drück mich – Zieh mich Negierung
Schleichendes Merkmal	Forschungsorientierte Entwicklung
Goldgepresste Entwickler	

Legende

vorgestellt

außerdem noch vorhanden

Die Technologie

Silberkugel Syndrom	Mangel an automatischer SCC
Überschätzte Einsparung durch neue Praktiken	
Toolwechsel während des Projektes	

Legende

vorgestellt

außerdem noch vorhanden

Gilligan‘s Insel

Der Autor Steven McConnell beschreibt die Verhinderung von Fehlern in einem Softwareprojekt wie die Flucht von Gilligan‘s Insel.

Man erstellt einen guten Plan. Zu Anfang hält sich jeder an den Plan und es scheint zu funktionieren. Wenn dann die Episode sich dem Ende nähert, geht irgendetwas schief. Eine Sache, die man vorher nicht bedacht hatte. Obwohl man vorher alle Fehler, die schon einmal begangen wurden, vor vornherein vermieden hatte. Am Ende einer Episode ist man dann wieder am Anfang und quasi kein Stück weiter gekommen, außer viel Zeit zu verschwenden.

Der Ausweg

Es mag vielleicht viele Auswege geben. Ein möglicher ist der folgende. Man schreibt die schlimmsten Fehler der letzten Projekte, so man sie denn kennt, auf. Man kann für den Start die Liste von Fehler aus dem Buch von Steven McConnell nehmen. Man tauscht sich mit Kollegen in anderen Firmen aus und bringt in Erfahrung welche Fehler sie gemacht haben. Die Liste sollte ebenfalls öffentlich gehalten werden, so dass anderen Programmierer sie ebenfalls als Start nehmen können.

Quellen

McConnell, Steven. 1996 „Rapid Development“ Microsoft Press