

1 Prolog

*Wer sich nicht verändern will, wird auch verlieren,
was er bewahren möchte.
(Gustav Heinemann)*

Folgt man aktuellen Statistiken über Softwarebau, so belegen diese, dass zwischen 60% und 80% aller Projekte im strengen Sinne gesehen nicht erfolgreich sind, d. h., dass die drei Kriterien in-time, in-quality und in-budget nicht planungsgemäß erfüllt wurden. Nicht berücksichtigt sind hierbei die unnötigen Folgekosten, die Firmen dadurch entstehen, dass sie durch Unachtsamkeit und Mängel entstandene Fehler während der Planungs- und Produktionsphasen nachbessern müssen bzw. durch konstruktive Gründe dauerhafte Mehraufwände bei der Pflege oder Weiterentwicklung ihrer Software haben.

1.1 Sichten müssen sich ändern

Sollten Sie sich die Frage stellen, ob, was und wie Dinge zum Besseren verändert werden können, so werden Sie in diesem Buch Antworten finden. Dabei wird nicht in erster Linie die Rede von Bits & Bytes, Objekten & Klassen oder irgendwelchen spezifischen Entwicklungsmethoden der Software sein. Hierzu gibt es vielfältige andere Fachliteratur. Zudem werden vermutlich Sie und Ihre Mitarbeiter ohnehin über viel Sachkenntnisse in diesem Bereich verfügen. Im Rahmen dieses Werkes möchte ich mich mit dem „nichtbinären“ Umfeld der Softwareentwicklung befassen. Der Tatsache, dass den größten Projektrisiken, nämlich Komplexität und Dynamik, nicht ausreichend oder rechtzeitig begegnet wird, soll besonderes Augenmerk gewidmet werden. Die Rede wird sein von all denjenigen Aspekten, die sich nicht mit Programmen und Prozessen abbilden lassen, sondern diesen zugrunde liegen müssen. Es geht um Menschen, die hierin aktiv handeln, persönliche Erfahrungen einbringen oder aufgrund persönlicher Unkenntnis Dinge verbergen oder verhindern. Dieses Buch versteht sich als ein hinterfragendes Werk und nicht als ein Text mit fertigen Rezepten. Es wird Ihnen viele Hintergrundinformationen anbieten und möchte Sie vor allem zum Nachdenken und Hinterfragen Ihrer aktuellen Vorgehensweisen einladen.

1.1.1 Lohnt sich dieses Buch für Sie?

Soweit zum Einstieg – jetzt sind Sie gefragt. Da ich Ihre Zeit nicht unnötig strapazieren möchte, schlage ich Ihnen drei Kategorien vor, die Ihnen dabei helfen sollen, für sich zu klären, ob dieses Buch es wert ist, dass Sie es zum jetzigen Zeitpunkt lesen:

- Wenn Sie mit Ihren Resultaten als IT-Professional bzw. denen Ihrer IT-Mannschaft völlig einverstanden sind und Sie auf durchweg positive Resultate blicken können, so sollten Sie dieses Buch beiseite legen und sich wichtigeren Themen zuwenden.
- Für den Fall, dass es bei Ihnen gewisse Anfragen oder Zweifel bzgl. der Effizienz Ihrer Softwareentwicklung geben sollte, so dürfte dieses Buch Sie an der einen oder anderen Stelle durchaus zum Nachdenken anregen.
- Stellen Sie sich den entsprechenden Fragestellungen, bevor Sie dieses Buch wieder weglegen!

Einen weiteren Punkt möchte ich Ihnen an dieser Stelle nicht vorenthalten. Dieses Buch hat die erklärte Absicht, nicht nur als eine Quelle für Sachinformationen zu fungieren. Vielmehr möchte ich Sie diskret dazu herausfordern einige Ihrer eigenen Handlungsparadigmen in diesem Kontext zu hinterfragen. Wenn Sie grundsätzlich bereit sind, selbst erste Schritte zu gehen, dann investieren Sie sinnvoll in diese Lektüre, andernfalls sind Sie gerade im Begriff, eine Fehlinvestition an Zeit und Geld zu leisten.

1.1.2 Nicht Bits und Bytes, sondern Menschen

Nach diesen Anmerkungen sind noch einige Punkte klarzustellen und deutlich herauszuheben. Sollten Sie an dieser Stelle ein Buch erwarten, welches Ihnen sagt, wie man z. B. UML¹, Java-Klassen oder Softwarearchitekturen erstellt, so wird Sie dieses Buch enttäuschen. – Natürlich wird in der einen oder anderen Form die Technik zu Wort kommen, erklärtes Ziel dieses Werkes ist es jedoch, die Menschen hinter diesen Technologien zu betrachten: die Entwickler, IT-Architekten, Fachspezialisten sowie die entsprechenden Manager. Der Schwerpunkt wird dabei auf dem Umgang mit der innewohnenden Komplexität bei der Erstellung von Software liegen und den Möglichkeiten, diese zu beherrschen. Es geht sowohl um Vorgehensweisen bei der Produktion wie auch deren Risiken, die durch diese Prozessbeteiligten hervorgerufen werden, und die Beleuchtung möglicher Fehlerquellen. Wir mögen inzwischen auf faszinierende Softwarewerkzeuge und

¹ UML – „Unified Modelling Language“ – ist eine derzeit häufig verwendete Softwareentwicklungs-Standardsprache zur Spezifizierung, Visualisierung und Konstruktion von Softwarekomponenten.

Entwicklungsmethoden blicken. Diese bleiben jedoch ineffizient, wenn die Menschen, die sie nutzen, zum Flaschenhals werden und nicht ausreichend in diese Vorgehensweisen integriert wurden. Da diese Techniken in enger Interaktion mit ihrem Umfeld, den eigenen Entwicklungsteams, dem Management und vor allem auch den Kunden, stehen, möchte ich mit Ihnen integrative Ansätze aufspüren und diskutieren.

Ja, Sie lesen richtig: Dieses Buch stellt sich immer wieder der Frage, ob nicht gerade Sie und ich – oder aber Ihre und meine IT-Kollegen – zum zentralen Faktor des Erfolgs oder Misserfolgs eines Softwareprojektes werden! Haben wir eine genügend breite Sichtweise auf die wirklichen Probleme, die sich geschickt hinter dem Tagesgeschäft verstecken, oder haben wir diese eigentlichen Verursacher noch nicht in ausreichendem Maße als die eigentlichen Feinde erkannt und ihnen den Krieg erklärt? Diese Probleme werden nur dann gelöst werden können, wenn Sie und ich fundierte, aber pragmatische Lösungen in die Wege leiten.

1.2 Die drei INs

Was erwartet Sie also in dieser Lektüre? Überall strebt man in der IT-Branche nach Software, die **IN³** gerecht wird, nämlich „in-time“, „in-quality“ und „in-budget“. Die Software soll robust, am besten fehlerfrei und vor allem für die Nutzungsdauer leicht wartbar und erweiterbar sein. Aufgrund des hohen Kostendrucks soll die zu bauende Software dazu beitragen, die Unternehmensgewinne stabil zu halten. Viele andere Kostenfaktoren werden absehbar steigen, hier wurde jedoch mittels Unterstützung durch ein Softwareprodukt eine Nische erspäht, von der man sich bei sinkenden Stückpreisen immer noch steigende oder wenigstens gleichbleibende Erlöse erhofft. Ein solches Unterfangen ist grundsätzlich richtig, es verlangt jedoch einen hohen Reifegrad der jeweiligen Entwicklungsabteilungen. Diese müssen geprägt sein durch ganzheitliches, vorausschauendes und unternehmerisches Denken. Ist Ihre Organisation bereits so mündig?

Der erfolgreiche Bau solcher Software ist nicht nur geprägt von den technischen Randbedingungen und dem Einsatz geeigneter Technologien, sondern die Resultate werden stark durch die darin mitwirkenden Menschen beeinflusst. So betrachtet gehen diese für die Zeit der Softwareentwicklung eine Art Symbiose mit der Technik ein, um auf diese Art und Weise ihr „Baby“ hervorzubringen.

Auf diesen Seiten geht es somit um diejenigen Randbedingungen und Einflussfaktoren, die **IN³** stören. In diesem Werk wird die Rede davon sein, welches die anfänglichen Einflussfaktoren sind, die Ihnen die Chance

geben, auch noch gegen Ende eines Software-Lebenszyklus Kosten und Qualität zusammenzuhalten, bzw. für welche frühen Fehler Sie den Rest des Software-Lebenszyklus büßen müssen. Hierbei möchte ich Sie mit wirkungsvollen Ansätzen bekannt machen, die man im wahrsten Sinne als **ganzheitlich Psycho-Logische** Aspekte beim Bau von Software bezeichnen könnte.

Jeder dieser drei Teilbegriffe steht hierbei für eine entscheidende Grundaussage:

Unter **Ganzheitlichkeit** wird es in diesem Kontext um das Gesamtsystem Softwareentwicklung gehen. Die Rede wird von denjenigen Aspekten sein, die häufig unterschlagen werden, weil sie nicht unbedingt in ein ingenieurmäßiges Weltbild passen. Daneben wird immer wieder die Frage beleuchtet werden, was Software komplex macht und wie man dieses entschärfen kann.

Der zweite Begriff, nämlich „**Psycho**“, steht für die Skills, Teamstrukturen und Eigenarten der Entwicklungsmannschaft.

Als Letztes bezieht sich die „**Logik**“ auf all das, was wir ohnehin schon in der einen oder anderen Weise innerhalb der IT betreiben: auf Prozesse, Verfahren, Methoden und ingenieurmäßiges Vorgehen.

Da der Ganzheitlichkeit in diesem Buch große Bedeutung zugeordnet wird, mag es Themen geben, die sie ansprechen, andere wiederum könnten Ihre eigenen Handlungsparadigmen betreffen und zuletzt wird es solche Ansätze geben, die Sie völlig anders sehen bzw. ablehnen. Jede dieser Positionen ist legitim! Im Fall der letzten Sichtweise hilft mir persönlich eine Einstellung des protestantischen Reformators Martin Luther weiter: Bei Aussagen, die er nicht verstand oder akzeptierte, so berichtet man über ihn, zog er innerlich seinen Hut, grüßte freundlich, ließ das Thema stehen und zog weiter. Diesen Ansatz halte ich persönlich für differenzierter und adäquater als solche Vorgehensweisen, bei denen man wegen divergenter Sichtweisen zu bestimmten Teilthemen den gesamten Ansatz verwirft.

1.3 Bilanz ist gefragt

Sollte ich Sie jetzt neugierig gemacht haben, so möchte ich Sie zu einer Bestandsaufnahme in dem für Sie zutreffenden Kontext einladen. Im klassisch medizinisch-therapeutischen Sinne möchte ich hierbei im Wesentlichen folgende Schwerpunkte setzen:

Anamnese. Zunächst heißt es Inventur machen! Wie sieht es tatsächlich in Ihrer Organisation aus? Kennen Sie die kleinen und großen „Sünden“, die sich regelmäßig wiederholen? Das Augenmerk wird hier auf dem gesamten Lebenszyklus einer Software liegen. Ebenso geht es darum, einen Blick auf die Prozessbeteiligten zu werfen und die guten wie schlechten Einflüsse zu ergründen, welche diese auf Ihre Produkte haben.

Analyse. Zunächst wird es darum gehen, die eigentlichen „Feinde“ im Softwarebau zu ermitteln und herauszufinden, durch welche Mechanismen diese wirksam werden. Hieran anschließend werde ich mit Ihnen verschiedene „Best Practices“, also bewährte Verfahren – auch aus anderen Industriezweigen –, betrachten und deren Verwertbarkeit für effizienten Softwarebau beleuchten. Es wird die Rede von Prozessmodellen sowie von gut untersuchten Be- und Entschleunigungsfaktoren aus dem Praxisalltag sein.

Therapie. Mit diesem so ermittelten gemeinsamen Begriffshintergrund ist der letzte Teil dieses Werkes Praxisaspekten gewidmet. Anhand von Schwerpunktthemen aus dem Bereich Projekt- und Prozessmanagement werde ich solche Aspekte zusammenfügen, die m. E. zusammengehören. Es werden geeignete Methoden und „Werkzeuge“ vorgestellt und es wird darüber gesprochen werden, wie sich die beteiligten Menschen so integrieren lassen, dass hierdurch die Qualität eines Softwareproduktes und die Effizienz der Umsetzung deutlich gesteigert werden können. Zum Abschluss werden diese Elemente, Puzzleteilen ähnlich, zu einem praxistauglichen Gesamtmodell verbunden.

Die hier zusammengestellten Vorgehensweisen verstehen sich als holistischer, also ganzheitlicher Ansatz, wobei diese Ganzheitlichkeit sowohl ingenieurmäßige wie auch personenorientierte Aspekte umfasst. Was dies bedeutet, lässt sich mittels eines kleinen Gedankenexperiments anhand von Abb. 1.1 beantworten: In dieser Abbildung sehen Sie unterschiedlich weite Röhren, die in verschiedener Weise kombiniert und übereinander gestapelt wurden. Nach Abschluss des Aufbaus wird nun z. B. Sand durch die jeweiligen Konstrukte hindurchgeleitet. (Ich unterstelle dabei, dass kein Sand zwischen den Rohrversatzstücken verloren geht!) Um eine solche „Sandmaschine“ zu bauen, müssen Sie Geld ausgeben: Dünne Rohre kosten wenig, mittlere Rohre mehr, dicke Rohre viel Geld!

Die Gretchenfrage lautet nun: Mit welchem System machen Sie die größten Gewinne? – Nun, simpel wie das Beispiel aufgebaut ist, ergibt sich nachfolgende Lösung (vgl. Abb. 1.2, Experiment, Auswertung & Regeln). Sie lautet: Fall (C) hat das Rennen klar gewonnen, obwohl es NICHT die billigste Variante in der Erstellung war. Punktuelle Verbesserungen von abhängigen Prozessen mögen werbewirksam sein, sie verringern jedoch nicht per se die Stückkosten. – Um möglichst rasch eine

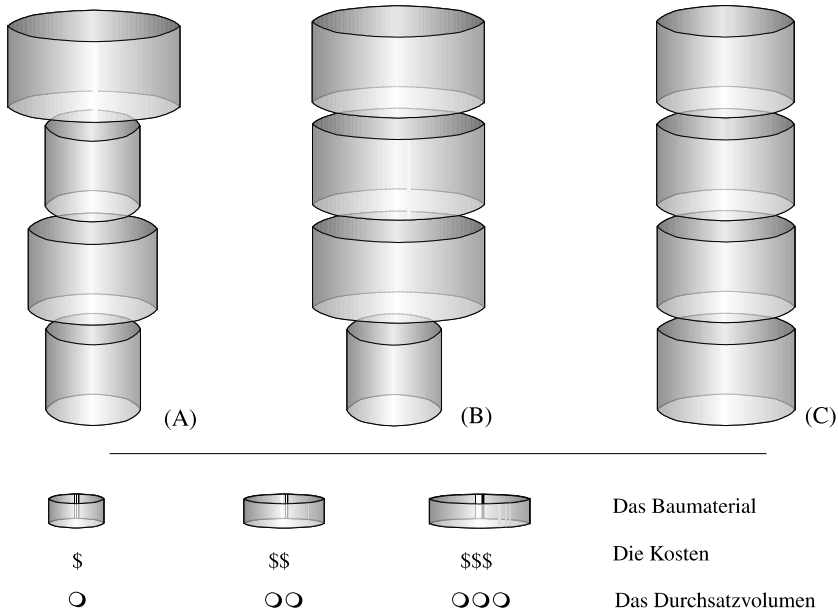


Abb. 1.1. Ein Gedankenexperiment

messbare Steigerung des Durchsatzes bei möglichst geringen Gesamtkosten zu erreichen, müssen alle betroffenen Segmente zeitgleich, dafür aber in kleinen Schritten auf ein gemeinsames Maß gebracht werden.

Um auf das Kernthema dieses Buches zu kommen, werde ich nun dieses kleine Röhrenexperiment auf IT-Entwicklungsabteilungen, wie sie vielfach zu finden sind, übertragen (vgl. Abb. 1.2, Übertragung): Die IT-Mitarbeiter kennen die neuesten Softwaretechnologien und brennen meistens darauf, diese einsetzen zu können. Unternehmen investieren stetig mehr in Testen und Qualität, um so zu qualitativ besseren Produkten zu gelangen. Unternehmensführungen investieren z.T. erhebliche Summen in Softwarewerkzeuge und bilden die Mitarbeiter hierin auch entsprechend aus, um hierdurch die Effizienz im Softwarebau zu steigern. Die Menschen mit ihren Grenzen und den hierdurch verursachten Störungen, die massiv auf die Produktion einwirken können, bleiben vielfach unberücksichtigt. Ihnen wird häufig unterstellt, dass sie als Professionals hierfür bereits alle notwendigen Skills besäßen. Vergessen werden dabei jedoch zwei Aspekte:

- Der erste Aspekt besteht darin, dass die allerwenigsten IT-Werker diese Faktoren in ihren Ausbildungen vermittelt bekommen haben.
- Zum Zweiten haben sehr viele IT-Entwicklungsmannschaften ein eher jugendliches Durchschnittsalter, weshalb sie neueste Technologien gut kennen und auch sehr schnell in deren Ausimplementierung sind, jedoch

fehlt es oftmals an der notwendigen Erfahrung in den anderen Bereichen. Hierbei ist nicht die Erfahrung zu kodieren gemeint: Dafür sind viele Softwaremitarbeiter schon seit ihrer Jugend mit diesem Medium vertraut. Die Erfahrung, von der ich hier spreche, hat mit ganzheitlichen Lösungen zu tun, denen erschwerte Randbedingungen in einer komplexen Gesamtsituation beigegeben sind.

Das vorliegende Buch kann und will die aufgeworfenen Einzelthemen nicht in aller Tiefe ergründen. Es schließt sich nicht der Fastfood-Mentalität an, die oftmals einem „Kochrezept“ gleich, lautet: „How to get all your complexity related IT problems solved within 10 minutes.“ Untersuchungen bei Unternehmen, die solche oder ähnliche Veränderungsprozesse erfolgreich hinter sich gebracht haben und nun die Früchte ihrer Anstrengungen einsammeln, belegen, dass mit 3–5 Jahren harter Arbeit zu rechnen ist. Primäres Ziel dieses Buches ist es daher, Ihnen zu vermitteln, wie weit hier die wechselseitigen Abhängigkeiten reichen. Ich möchte Sie für diese Thematik sensibilisieren, um Ihnen danach

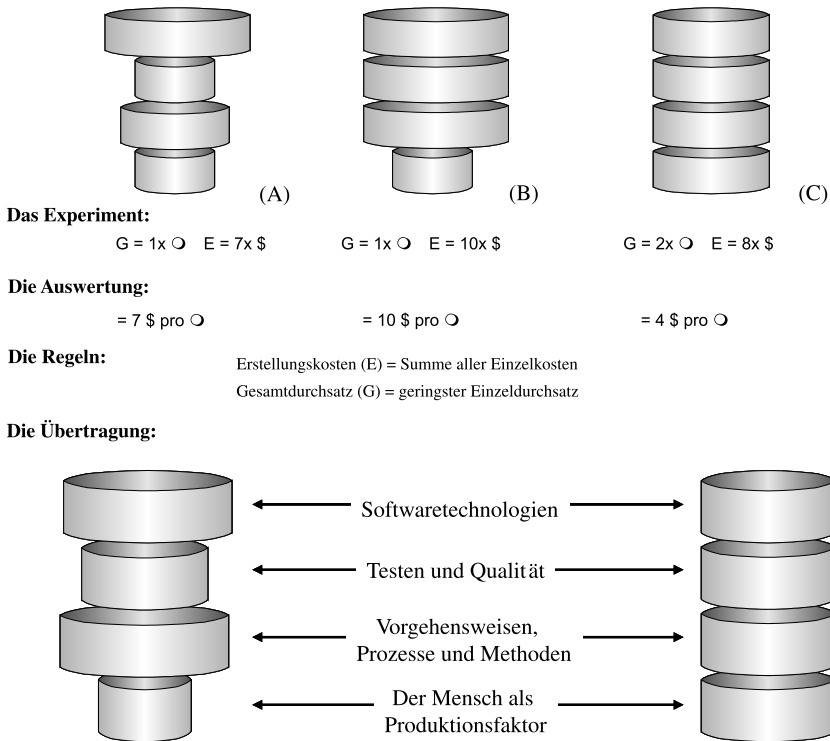


Abb. 1.2. Auflösung des Gedankenexperiments und Übertragung auf die Softwareproduktion

schrittweise in ausgewählten Bereichen Verbesserungen vorzuschlagen. – Wie schon am Eingangsbeispiel ersichtlich, wird dies mit zusätzlichen Aufwänden verbunden sein, die sich nicht nur im kaufmännischen Sinne definieren lassen, sondern darin bestehen, dass persönliche Investitionen im Sinne von Engagement und Durchsetzungswille eingebracht werden müssen. Nur dann, wenn Sie bereit sind, großzügig alte Paradigmen und Klischeevorstellungen hinter sich zu lassen und neue Wege zu beschreiten, wird eine Umstellung gelingen.

Genug der Vorrede. Ich lade Sie nun in die Wirklichkeit der Softwareproduktion ein, wie sie immer noch in vielen IT-Häusern anzutreffen ist.

1.4 Tote Pferde reiten

Bei den Dakota-Indianern in Amerika gab es früher eine Stammesweisheit, die besagte: „Wenn Du feststellst, dass Du auf einem toten Pferd reitest, so besteht deine beste Strategie darin abzusteigen.“ Gerade in der IT-Branche gehen wir leider recht häufig anders mit dieser Thematik um, weil Projektdruck und Anforderungen an Management und Mitarbeiter ein Überleben auf Tagesbasis notwendig werden lassen. Daher betreiben wir oftmals mit Akribie Symptombehandlung, vergessen dabei jedoch sehr häufig, die eigentlichen Probleme wahrzunehmen und zu lösen. Nicht selten geschieht es dann, dass die Weisheit der Dakotas in eine oder mehrere eigene Weisheiten umgemünzt wird:

- Wir besorgen uns eine stärkere Peitsche, um das tote Pferd anzutreiben. (Etwa indem Service-Level-Verträge mit Zulieferern abgeschlossen und diese vertraglich zur Besserung der Gesamtsituation gezwungen werden.)
- Wir wechseln den Reiter, um mit frischem Elan voranzukommen. (Erfahrene Mitarbeiter werden durch weniger erfahrene Kräfte ausgetauscht. Die neuen Kräfte lassen sich – so hofft man wohl – besser durch das Linienmanagement steuern und leisten weniger Widerspruch?!)
- Wir sagen: „So haben wir das Pferd immer geritten“ und bleiben bei Altbewährtem. (Man verschließt sich gegen echte Innovation. Es wird damit argumentiert, dass man ja nur „ein wenig“ neue Software haben wolle und keine Weltrevolution.)
- Wir gründen einen Arbeitskreis, um das Pferd gründlich zu analysieren und all seine Leistungen der Vergangenheit akribisch zu dokumentieren. (Leitungskreise werden eingesetzt, die wiederum Mitarbeiterstäbe einsetzen, die wiederum akribisch jedes einzelne alte Bauteil bibliographisch erfassen.)

- Wir besuchen andere Orte, um zu sehen, wie man dort tote Pferde reitet, um zu neuen Erkenntnissen zu gelangen. (Man reist zu Schwester-, Nachbar- oder Referenzfirmen, um zufrieden festzustellen, dass man dort genauso arbeitet wie man selbst. Vergessen wurde bei der Auswahl der Firmen leider nur, dass alle besuchten Firmen aus dem Verzeichnis „Club der Halter toter Pferde“ stammen.)
- Wir erhöhen die Qualitätsstandards für den Beritt toter Pferde und führen hierfür sehr strenge Audits ein. (Qualität kommt ins Spiel und bewertet die Güte ineffizienter Vorgehensweisen – was in letzter Konsequenz oftmals nichts anderes heißt: Beim Audit ist herausgekommen, dass die (ineffizienten) Prozesse hervorragend gelebt werden. – Die Frage nach der Werthaltigkeit der Prozesse wird häufig nicht gestellt.)
- Wir bilden eine Taskforce, um das tote Pferd wiederzubeleben. (Es wird viel Kraft und Kapital mobilisiert, um halbtote Software zu revitalisieren, ihr etwas Kosmetik zu verpassen oder dem Kunden gegenüber ein neues Produktimage aufzubauen. – Technisch gesehen bleibt die Software jedoch unverändert. Die Tour zum Kosmetiker bringt vielfach nur teilweise erfüllte Versprechungen für die Kunden und garantierte Zusatzaufwände für das eigene Haus mit sich!)
- Wir schieben eine Trainingseinheit ein, um effizienter tote Pferde zu reiten. (Mitarbeiter werden in Prozessen oder auch Softwareproduktionen geschult – dumm dabei ist nur, dass das Ganze vielfach in den Köpfen der Mitarbeiteten endet. Bis ans Licht der Wirklichkeit dringt es nicht vor, weil z. B. die hierfür notwendige Infrastruktur noch fehlt – und somit alles Theorie und unnützer Aufwand bleibt.)
- Wir stellen Vergleiche zwischen unterschiedlich toten Pferden an, um die Wettbewerbschancen zu erhöhen. (Es werden Gremien gebildet, die sich Firmen mit ähnlicher Arbeitsweise anschauen, um herauszufinden, mit Hilfe welchen Mysteriums bestimmte marginale Verbesserungen erreicht werden konnten.)

Langer Rede kurzer Sinn: Es werden viele Nebenaspekte betrachtet und diskutiert, die oftmals jedoch eher den Gesetzen blinden Aktionismus folgen, statt Probleme von den Wurzeln her grundsätzlich zu lösen. In diesem Sinne möchte ich Sie an dieser Stelle zu einer kurzen Selbstreflexion einladen. Nachfolgend habe ich Ihnen noch einige Aspekte zu toten Pferden offen gelassen. Die Frage an Sie lautet nun: Wie sehen Ihre Pferdekadaver aus?

- Wir ändern die Kriterien, die besagen, ob ein Pferd tot ist.
- Wir kaufen Leute von außerhalb ein, um das tote Pferd zu reiten.
- Wir schirren mehrere tote Pferde zusammen an, damit diese schneller werden.

- Wir machen zusätzliche Mittel locker, um die Leistung des toten Pferdes zu erhöhen.
- Wir geben Studien in Auftrag, um zu sehen, ob es billigere Berater für tote Pferde gibt.
- Wir kaufen etwas hinzu, das tote Pferde schneller laufen lässt.
- Wir erklären, dass das Pferd „besser, schneller und billiger“ tot ist.
- Wir bilden einen Qualitätszirkel, um eine Verwendung für tote Pferde zu finden.
- Wir überarbeiten die Leistungsbedingungen für Pferde.
- Wir richten eine unabhängige Kostenstelle für tote Pferde ein.



<http://www.springer.com/978-3-540-31743-2>

Dynaxity

Management von Dynamik und Komplexität im
Softwarebau

Hamilton, P.

2007, XI, 230 S., Hardcover

ISBN: 978-3-540-31743-2