

Patrick Hamilton

Wege aus der Softwarekrise

 Springer

Patrick Hamilton

Wege aus der Softwarekrise



Springer

Wege aus der Softwarekrise

Patrick Hamilton

Wege aus der Softwarekrise

Verbesserungen bei der
Softwareentwicklung

Dr. Patrick Hamilton
Am Grundweg 22
64342 Seeheim-Jugenheim

ISBN 978-3-540-72869-6

e-ISBN 978-3-540-72871-9

DOI 10.1007/978-3-540-72871-9

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

© 2008 Springer-Verlag Berlin Heidelberg

Dieses Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere die der Übersetzung, des Nachdrucks, des Vortrags, der Entnahme von Abbildungen und Tabellen, der Funk-sendung, der Mikroverfilmung oder der Vervielfältigung auf anderen Wegen und der Speicherung in Datenverarbeitungsanlagen, bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten. Eine Vervielfältigung dieses Werkes oder von Teilen dieses Werkes ist auch im Einzelfall nur in den Grenzen der gesetzlichen Bestimmungen des Urheberrechtsgesetzes der Bundesrepublik Deutschland vom 9. September 1965 in der jeweils geltenden Fassung zulässig. Sie ist grundsätzlich vergütungspflichtig. Zuwider-handlungen unterliegen den Strafbestimmungen des Urheberrechtsgesetzes.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk be-rechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Einbandgestaltung: KinkelLopka GmbH, Heidelberg

Gedruckt auf säurefreiem Papier

9 8 7 6 5 4 3 2 1

springer.de

*Es wäre gut Bücher kaufen,
wenn man die Zeit,
sie zu lesen,
mitkaufen könnte,
aber man verwechselt meistens
den Ankauf der Bücher
mit dem Aneignen ihres Inhalts.
Arthur Schopenhauer (1788–1860)*

Inhaltsverzeichnis

1	Prolog.....	1
2	Irrwege und Auswege	9
2.1	Softwarebau in der Dauerkrise.....	9
2.1.1	In den Sümpfen des mystischen Mann-Monats.....	11
2.1.2	Fehler ohne Verjährungsfristen	13
2.1.3	Handeln mit Kurzsichtigkeit.....	15
2.2	Haben und Schein	18
2.2.1	Der Traum von Fließbandproduktion	20
2.2.2	Handeln von heute, Trends von morgen.....	23
2.2.3	Auf der Suche nach Schwachstellen.....	28
2.3	Ausbruch aus dem schwarz-weißen Zeitalter.....	30
2.3.1	Abschied von einer diskreten Welt	32
2.3.2	Wissensgetriebener Softwarebau	36
2.4	Wege zur Professionalisierung.....	37
2.4.1	Organisationsaspekte in der IT-Produktion	41
2.4.2	Rollen mit Zukunft	45
3	Aufbruch in die Zukunft.....	49
3.1	Modellwelten stellen sich vor	49
3.1.1	Messlatten der IT-Organisation	50
3.1.2	Schemagetrieben – und doch nicht schematisch.....	53
3.1.3	ITIL – eine Library für Servicemanagement	55
3.1.4	Six Sigma oder „Besser ist billiger“	58
3.2	Qualifizierte Hersteller brauchen gute Produkte.....	61
3.2.1	Was gute Software auszeichnet	61
3.2.2	Das Eisberg-Prinzip	66
3.2.3	Erfolgreich und doch kein Zufallsprodukt?	68
3.2.4	Erweitertes Kano-Prinzip.....	72
3.2.5	Was macht ein Softwareprodukt gut?	74
3.3	Alte Ziele, neue Wege.....	81
3.3.1	Wenn Softwarebau vom Gehirn lernt	81
3.3.2	Konstruierte Wirklichkeiten	82
3.3.3	Wahrnehmung und Denken	84
3.3.4	Konstrukte reiner Logik.....	85

3.4	Systeme im Quadrat?	89
3.4.1	Die Kunst der Steuerleute	92
3.4.2	Mehr als nur systemisches Arbeiten	94
3.4.3	Was heißt systemisches Handeln?	99
3.5	Systemisch arbeiten.....	102
3.5.1	Umzug ins nicht-triviale Kontinuum	105
3.5.2	Konstruierte Wirklichkeiten	111
4	Hilfsmittel aus der Krise	115
4.1	Nieder mit den binären Zöpfen	115
4.1.1	Weiche Worte schaffen harte Fakten	119
4.1.2	Kommunizieren – eine Kunst!	122
4.1.3	Mäeutik – oder die Kunst zu fragen.....	129
4.1.4	Fragen für Fortgeschrittene.....	133
4.2	Kreativität, Wissen und Lernen.....	139
4.2.1	Kreative Softwareproduzenten	139
4.2.2	Kreativität und Softwarebau	140
4.3	Denken, wissen und kommunizieren	145
4.3.1	Universum des Wissens	146
4.3.2	Piaget lässt grüßen	150
5	Wege zum Softwarebau von morgen	155
5.1	Systemischer Softwarebau	155
5.1.1	Vom Gedanken zum Produkt.....	155
5.1.2	Schwachstellenbeseitigung	158
5.2	Architekten in der Softwareproduktion.....	160
5.2.1	IT-Architekten in Vergangenheit und Gegenwart	162
5.2.2	IT-bewanderte Systemiker und Coaches gesucht	169
5.2.3	Erweitertes Rollenverständnis	172
5.3	Systemisches Arbeiten in der IT	177
5.3.1	Ein Fallbeispiel	178
5.3.2	Der Weg ist das Ziel	181
6	Epilog.....	183
7	Anhang	185
7.1	Abbildungsverzeichnis	185
7.2	Weiterführende Literatur.....	185
7.3	Quellenübersicht	191
	Stichwortverzeichnis	195

1 Prolog

*Und wenn das, was du tust, dich nicht weiterbringt, dann tu etwas
völlig anderes – statt mehr vom gleichen Falschen!*

Paul Watzlawick (1921–2007)

Einmal angenommen, Softwarebau wäre – anders als heute zugrunde gelegt – keine Engineering-Disziplin, sondern wäre stattdessen den psychologischen Fächern zugeschlagen worden. Einmal angenommen, dieses Fach hätte sich als Folge der seit den frühen 70er Jahren schwelenden Softwarekrise zu einer dort angesiedelten Disziplin gemausert, und einmal angenommen, wir redeten über die Art und Weise, wie die Ergebnisse in Bezug zu ihrer Aufgabenstellung stünden, und nicht so sehr darüber, mit welchen technischen Methoden oder Maschinen man Code implementiert. Würden wir dann in einer anderen Welt leben (soweit es die industrielle IT betrifft)? Könnten möglicherweise Dinge innerhalb der Computerwissenschaften als gelöst betrachtet werden, die sich bis heute als ungelöst, mühevoll und ineffizient darstellen?

Keine Angst, dieses Buch hat es sich nicht zur Aufgabe gesetzt, bewährtes im Softwarebau neu zu definieren. Ausgangspunkt ist hier allerdings eine Betrachtungsweise, deren primärer Fokus nicht techniklastig ist, sondern die, vergleichbar den Kognitionswissenschaften, anstehende Fragen aus einem erweiterten Blickwinkel ergründet und hieraus ihre Schlüsse zieht. Dabei steht im Fokus der Betrachtungen und Überlegungen immer wieder die im industriellen Softwarebau so entscheidende Fragentrias:

Wie wird Softwarebau effizienter?

Wie wird Softwarebau kostengünstiger?

Wie wird die entwickelte Software qualitativ besser?

Viel ist hierzu in den vergangenen Jahrzehnten unternommen und ausprobiert worden. Manches hat sich als gut erwiesen, anderes hat nur für kurze Zeit das Licht der Welt gesehen. Egal welche Methoden und Techniken gerade en vogue sind – diejenige Größe, die sich am wenigsten im Sinne der eigenen Beteiligung verändert hat, ist der Mensch selbst. Auch wenn ihm neuartige Automaten zur Verfügung stehen, wenn Konstruktionshilfen jeglicher Art ihm das Leben beim Bau von Software erleichtern

sollen und eine immer höhere Automatisierung das Ihre dazu beiträgt: Am Ende sind es trotz all der beeindruckenden Technik gerade die Software-Entwickler in ihren verschiedenen Rollen, die durch ihr Wissen, ihr Zusammenspiel und ihre individuellen Vorgehensweisen maßgeblich darüber entscheiden, ob ein Produkt die geforderten Ziele erreicht oder nicht.

Dabei unterliegt industrielle Softwareproduktion ein paar sehr einfachen Grundparadigmen. Die Betriebswirtschaft lehrt, dass ein Produkt in Richtung auf den Markt ein materielles oder immaterielles Objekt sei, das diesem angeboten werde und in der Lage sei, Kundennutzen zu stiften. Mit anderen Worten: Mit einem solchen Produkt möchte die eigene Organisation die Wettbewerbsposition verbessern, bestimmte Marketingziele erreichen, Marktsegmente ausbauen bzw. diese halten – kurz Geld verdienen und damit erfolgreich sein.

Diese Grundsatzziele liegen normalerweise außerhalb des Zeitfensters, in welchem die eigentliche Projektarbeit zur Softwareerstellung stattfindet. Hierdurch werden gerade diese Anliegen oft für die Entwicklungsmannschaften intransparent, weshalb sie leicht in Vergessenheit geraten. Da Softwarebau alles andere als trivial ist, sondern in Hinblick auf die Prozesse, die eingesetzten Werkzeuge und Techniken oder auch die erwarteten Resultate den beteiligten Projektmitarbeitern sehr viel abverlangt, geraten diese Grundparadigmen gerne außer Sichtweite. Dabei ist es längst kaum noch möglich, dass eine geniale Entwicklungsplattform, gepaart mit ein paar guten Ideen sowie dem obligaten Internetanschluss über Nacht zu Wohlstand und Ansehen verhelfen und neue IT-Garagenfirmen erfolgreich das Licht der Welt erblicken.

Dieser Wunschtraum mag bei einigen funktioniert haben oder sich in Ausnahmefällen auch noch so erfüllen, für den Rest von uns heißt es jedoch strategisch ganzheitlich zu planen, ungenutzte Kapazitäten in der eigenen Organisation aufzuspüren, diese zu kanalisieren bzw. fehlerträchtige oder ineffiziente Bereiche zu verbessern. Mit anderen Worten: Statt liebevoller Bastelstuben muss die eigentliche Softwareproduktion immer mehr industrialisiert werden, will sie auch weiterhin den Wünschen der Kunden nach einem entsprechenden Preis-Leistungs-Verhältnis gerecht werden.

Dieser Einsicht folgend werden hier vor allem jene Bereiche analysiert, die Verbesserungspotential in der Herstellung versprechen, oftmals jedoch übersehen oder unterschätzt werden, zugleich aber einen ungenutzten Pool an Ressourcen bereithalten. – Das Hauptaugenmerk dieser Abhandlung wird daher auf einer genaueren Betrachtung der zentralen Rollen im Softwarebau liegen. Dabei werden Skill- und Rollenentwicklung wesentliche Parts zukommen, wobei Wissenserfassung im Sinne des Anforderungsmanagements sowie Wissenstransformation im Sinne von IT-Architektur und Implementierung ein hohes Maß an Aufmerksamkeit erhalten werden.

Bei all dem versteht sich dieses Buch nicht als eine rein akademische Abhandlung, sondern stammt aus vielen Jahren Praxis im Softwarebau sowie dem IT-Beratungsbereich. Es geht in erster Linie um die beteiligten Mitarbeiter und deren qualifizierte Weiterentwicklung. Dies geschieht vor dem Hintergrund, dass viele Publikationen über Produktionsprozesse oder Einzelverfahren den Eindruck vermitteln, dass die eigentliche Workforce in ihren jeweiligen Rollen bereits Teil einer nahezu heilen Welt sei: Mit ihren ohnehin schon vorhandenen Qualifikationen – so der leicht aufkommende Eindruck – brauche diese nur noch auf das jeweils anzuwendende neuartige Vorgehensmodell oder die zu nutzenden Werkzeuge umgeschult zu werden und schon wurde der qualifizierte Umstieg umgesetzt. – Die Wirklichkeit weicht hier ganz offensichtlich von solchen Wunschvorstellungen deutlich ab – und dies insbesondere deshalb, weil die beteiligten Personen mehr als nur ein neues Handbuch oder eine Tagesschulung über einen neuen Softwareprozess bzw. eine entsprechende Methode brauchen, sondern selbst oftmals durch einen mentalen und intellektuellen Veränderungsprozess hindurchgeführt werden müssen.

Wie oft wird Verbesserung der eigenen IT-Organisation in eben der Weise vollzogen, dass die betroffenen Mitarbeiter einen Kurzlehrgang für ein Tool oder ein Verfahren erhalten und man managementseitig im Anschluss daran bessere Ergebnisse erwartet. Dieser Ansatz, der ein wenig an Kochrezepte erinnert, hat mit Sicherheit immer wieder einmal seinen Platz – auch in der Informatik! Aber reicht dies tatsächlich aus, um anspruchsvolle Software zu bauen – und dies mit entsprechend qualifizierten und erfahrenen Mitarbeitern? – Ich fürchte nein! Ein renommierter Architekt wie etwa Norman Forster, der bekannt wurde durch seine genialen Entwürfe, etwa den des Berliner Reichstages oder der Londoner „Gurke“, hat seine Popularität sicherlich nicht deshalb erlangt, weil er mit Hilfe teurer CAD-Systeme, guter Templates oder ausgefallener Werkstoffe geniale Wasserleitungsrohre oder Fassaden perfekt in Bauwerke integrierte. Nein, ganz offensichtlich bedarf es deutlich mehr als nur ingenieurmäßig das Richtige zu tun. Dass dies nicht nur für Bauarchitektur gilt, sondern gleichermaßen auch für den IT-Bereich notwendig ist und z. T. auch schon stattfindet, ist daran zu erkennen, dass große Softwarehäuser, etwa der Gigant Microsoft, dies in ihrer eigenen Binnenorganisation entsprechend berücksichtigen¹.

¹ Hier findet sich – bezogen auf das Jahr 2007 – eine dreiköpfige Führungsmannschaft, die sich zusammensetzt aus der politischen Führung (= klassisches Management), einer kaufmännischen Führung sowie einer technischen Führung, die zum einen Visionär und Impulsgeber ist, zum anderen aber sicherstellt, dass alle Produktentwicklungen in die technische Gesamtvision passen.

Wenn dem so ist, dann bedarf es zweifellos eines Paradigmenwechsels in der Branche. Dieser kann wohl am besten mit demjenigen Wandel verglichen werden, der sich in der vergangenen Dekade im Bereich des Web-Designs vollzogen hat. Waren dort ursprünglich codenahe Kenntnisse gefragt, um z. B. HTML-Tags richtig zu platzieren, die wiederum die Basis der oft stereotypen Klötzchenwelt früherer Web-Seiten waren, so ist heute deutlich anderes gefragt! Anstatt primär die im Hintergrund laufende Technik zu sehen, ist dieser Bereich inzwischen zu einer Domain von Designern geworden. Dabei sind Design und Nutzerinteraktionsplanung zu tragenden Säulen bei der Erstellung solcher Seiten geworden, während die vorhandenen IT-Techniken den für die Realisierung notwendigen technischen Rahmen liefern. Die Erkenntnis hat sich durchgesetzt, dass der wichtigste Kunde solcher Webseiten das menschliche Auge mit seiner nachgeschalteten Wahrnehmung ist. Genau dieses gilt es mit den entsprechend aufbereiteten Informationen sinnesgerecht zu „beliefern“!

Nun kann komplexer Softwarebau nicht ohne weiteres mit Webdesign auf eine Stufe gestellt werden, dennoch gibt es einige Parallelen, die es zu bedenken gilt.

Um es auf den Punkt zu bringen: Wann wird der Tag kommen, an dem standardmäßig im industriellen Softwarebau zentrale IT-Mitarbeiter mit einer Zusatzqualifikation in Gesprächsführung, eher Coaches oder Change-managern gleich, so mit den Kunden zusammenarbeiten, dass nicht länger IT-technische Dornröschenschlösser mit vielen nachträglichen Erkern und Anbauten entstehen, während der Anwender weiter von märchenhafter Software träumt?

Professionalisierung des Softwarebaus

Erfolg hat nur, wer etwas tut, während er auf den Erfolg wartet.

Thomas Alva Edison (1847–1931), amerikanischer Erfinder

Der Softwarebau hat sich in der Tat erheblich weiterentwickelt, wird aber in den kommenden Jahren noch deutlich auf dem Weg hin zu fabrikmäßiger Produktion zulegen müssen, da es mit Sicherheit kein akzeptabler Zustand ist, wenn um die 50% aller Softwareprojekte² nicht im Erwartungshorizont der Auftraggeber enden. Folgt man dem „Chaos-Report“ der Standish Group für das Jahr 2006, so wurde in 46% aller registrierten Fälle das Kosten- oder Zeitbudget gesprengt. In der Fachpresse wurde dies als Erfolg gefeiert, da die Quote zwei Jahre zuvor noch stolze 53% betragen

² Mit Blick auf das Jahr 2006 wurde die Erreichung der 50%-Marke als großer Erfolg in der Fachpresse gefeiert.

hatte. Ebenso wurde als neuer Rekord gefeiert, dass für denselben Vergleichszeitraum jetzt 35% aller berücksichtigten Softwareprojekte planmäßig in Betrieb gehen konnten, im Gegensatz zu 29% im Berichtszeitraum zwei Jahre zuvor. Wiederum zehn Jahre zuvor, wir schreiben das Jahr 1994, als die Standish Group zum ersten Mal diese Zahlen erhob, belief sich diese Erfolgsquote auf ganze 16%. Und noch eine Auswertung, die zu denken gibt: Im Jahr 1994 musste jedes dritte Softwareprojekt als Totalverlust abgeschrieben werden, inzwischen wird relativ gleichbleibend „nur“ noch jedes fünfte bis sechste Projekt komplett versenkt.

Die hier wiedergegebenen Zahlen drücken eine deutliche Verbesserung aus, gleichwohl kann man sich leicht von diesen Quoten blenden lassen! Aber Hand aufs Herz: Würden Sie es als Produzent – egal ob Sie Autos, Fahrräder oder Nähmaschinen herstellen – sonderlich amüsant finden, wenn Sie nur bei jedem zweiten Produkt auf die geplanten Einnahmen kämen und jedes fünfte als Ausschuss aus der Produktion nehmen müssten? – Ich denke, dass kein Industriemanager oder Geschäftsführer im Non-IT-Bereich eine solche Ausschussquote lange akzeptieren würde, wenn ihm sein Job lieb ist!

Auch wenn die Professionalisierung im Softwarebau Fortschritte macht, so kann nicht behauptet werden, dass die von Edgar Dijkstra 1968 erstmals ausgerufene „Softwarekrise“ tatsächlich überstanden sei. Aus diesem Grunde ist es notwendig von anderen Branchen zu lernen, die dort gewonnenen Erkenntnisse zu analysieren, wo immer geeignet, sich diese für den IT-Bereich zu eignen zu machen und die Entwicklung mit aller Kraft voranzutreiben. Dabei entsteht der Eindruck, dass sich der Softwarebau derzeit in einer vergleichbaren Situation befindet wie der Automobilbau der 70er Jahre: Während man in Europa und den USA nach den besten Verfahren und Methoden suchte, zog die japanische Automobilindustrie – allen voran der damals fast vom Untergang bedrohte Automobilproduzent Toyota – zum Schrecken aller an der Konkurrenz vorbei und setzte neue Standards in Sachen Effizienz und Qualität (Womack, Jones et al. 1991). Ganzheitliche Produktionsmethoden sowie die optimierte Einbeziehung der wichtigsten Ressourcen, nämlich der Mitarbeiter, bildeten hier die Ausgangsbasis für den Erfolg.

Was aber bedeutet in diesem Kontext Professionalisierung? Folgt man einer kurzen, prägnanten Definition von Scott-Morgan (Scott-Morgan, Hoving et al. 2001), so steht dieser Begriff für Folgendes:

Professionalität = stetiger Fortschritt + keine Überraschungen

Professionalität = Fachwissen + neue Herausforderungen

Dies bezieht sich sowohl auf die Produktseite, wie auch auf die Seite der Akteure. Professionelles Arbeiten des Einzelnen alleine reicht dabei nicht aus. Erst dann, wenn eine Organisation die notwendige Innovationsfähig-

keit aufweist, wird dies dauerhaft zu besseren Produkten und damit verbunden größeren Marktchancen führen.

Erlauben Sie mir daher anhand dieser, zugegebenermaßen sehr einfachen Formeln folgende ketzerische Frage: Leben Sie in einer IT-Organisation, bei der die laufende Produktion bzw. die ausgelieferten Produkte keine regelmäßigen Überraschungen mehr zu bieten haben? – Oder ergeht es Ihnen (wenn Sie ganz ehrlich sind) so wie übrigens immer noch recht vielen IT-Häusern – dass IT Management in der eigenen Organisation gleichzusetzen ist mit dem Bewältigen all der ständig neuen Überraschungen und Hiobsbotschaften.

Auch der zweite Teil der Scott-Morgan'schen Aussage lässt sich durch eine weitere einfache Formel definieren:

Innovationsfähigkeit = Kreativität + Ideenrealisierung

Somit werden aus Business-Sicht einerseits Fachwissen in Verbindung mit Kreativität und andererseits Innovation gepaart mit der tatsächlichen Umsetzung dieser Ideen zu den zentralen Business-Drivern. Dem gegenüber steht ein in der IT-Community recht verbreiteter Ansatz: Er besteht darin, dass primär nach immer neuen Strukturierungsmethoden und technischen Werkzeugen gefahndet wird, um hiermit Software hoch systematisch zu entwickeln. Was aktive Ideenfindung anbelangt, so wird diese vielfach nur diffus erwartet, so als würden die richtigen Ideen an der korrekten Stelle eines Prozesses auf Knopfdruck ins Dasein purzeln. Dementsprechend werden die hierfür notwendigen intellektuellen Handwerkszeuge kaum kommuniziert oder geplant zum Einsatz gebracht. Dabei werden gerade mit deren Hilfe die wesentlichen Voraussetzungen dafür geschaffen, dass ein Unternehmen mit innovativen Produkten und Lösungen schnell und flexibel am Markt agieren kann, um so erfolgreich zu sein zu können.

Während wir durch ingenieurmäßig strukturiertes Vorgehen Software als Konstrukte binärer Logik mit Hilfe solcher Methoden und Modelle entwickeln, werden mehr und mehr Grenzen sichtbar. Diese haben Ähnlichkeiten mit dem Übergang von der klassischen Mechanik zur Quantenmechanik, wo irgendwann einmal schmerzlich klar wurde, dass die schöne und klar umrissene Welt eines Isaac Newton und seiner Fallgesetze eben doch nur in erster Näherung passt und viele Fragen offen lässt. Sich dieser Erkenntnis zu stellen mag für manchen Betroffenen unbequem sein, ist doch jeder Computerchip und jede darauf ablaufende Software eine definierte Abfolge von logischen Schaltelementen bzw. Befehlsketten. Dies ist zweifellos der Fall, berücksichtigt aber nicht, dass die inzwischen ungeheure Menge dieser Schaltungen bzw. Schaltanweisungen superponierte Effekte hervorbringt, die eben nicht mehr ausschließlich klassisch logisch zu verstehen und zu lösen sind, obschon ihnen genau diese Logik zugrunde liegt.

Und so wundert es nicht, wenn Bill Gates davon spricht, wie viel Neuland noch zu betreten sei. In einem Artikel über das, was Datenverarbeitung in den vergangenen 25 Jahren geleistet und hervorgebracht hat, berichtet er (Gates 2007) von den Grundsatzentwicklungen auf dem langen Weg zur Konsolidierung und flächendeckenden Nutzung von Software und weist Perspektiven auf neue Arbeitsfelder aus, wie z. B. die Robotics. Aber ist das Software-Engineering an sich, also diejenige Kunst, die auch bei noch komplexeren IT-technischen Fragestellungen reproduzierbare und qualitativ vorhersagbare Applikationen hervorbringen soll, auch schon bei solch einer Konsolidierung angelangt? Ich meine nein und betrachte die eingangs zitierten Zahlen als einen Indikator dafür.

Zwar versuchen schon längst viele technisch Verantwortliche unter dem Motto „Simplify your IT“ dem Rad in die Speichen zu greifen und eine solche Konsolidierung bzw. Standardisierung in die Wege zu leiten, aber gelingt es ihnen tatsächlich, der rasch wachsenden Komplexität sowie den kontinuierlich neu hinzukommenden Anforderungen entgegenzusteuern? Viel wird investiert, um reproduzierbare Abläufe in möglichst weiten Bereichen der Software-Produktion zu etablieren und somit die Softwareentwickler in einem gewissen Maß zu Fließbandarbeitern zu machen. Ebenso sucht man einem der Hauptfeinde innerhalb der IT-Produktion, nämlich der Komplexität (Hamilton 2006) entgegenzuwirken, indem Standorte geschlossen werden oder einheitliche Plattformprodukte initiiert werden. Was bei all diesen Bemühungen jedoch offen bleibt ist die Frage, ob eine solchermaßen vorgenommene Konsolidierung tatsächlich zum gewünschten Erfolg führt, muss doch die Software von gestern im Sinne von Veränderungsmanagement und die Software von morgen im Sinne von innovativem Planen und Handeln gleichermaßen berücksichtigt werden.

In diesem Sinne möchte ich Sie auf den folgenden Seiten dazu einladen, mich auf der Suche nach Verbesserungspotentialen im Softwarebau von heute und morgen zu begleiten, und dies vorrangig mit Blick auf die eigentlichen Akteure und deren hierfür benötigte Handwerkszeuge.