Ein moderner „Turmbau zu Babel“

Siegfried Wendt

Die Sprachverwirrung, Bibelillustration von Gustave Doré (1865); Quelle: Wikipedia, gemeinfrei

1. Einordnung des Titels

Im Unterschied zu meiner Jugendzeit ist es heute nicht mehr selbstverständlich, dass jeder Leser gleich weiß, was es mit einem Titel auf sich hat, der sich auf eine biblische Geschichte bezieht. Deshalb sei hier kurz skizziert, um was es beim sog. Turmbau zu Babel geht. Der erste Teil der Bibel, das sog. Alte Testament, beginnt mit dem ersten Buch Mose. In dessen elftem Kapitel wird berichtet, dass alle damaligen Bewohner der Welt eine einheitliche Sprache hatten. Eines Tages beschlossen sie, einen Turm zu bauen, der bis an den Himmel reichen sollte. Da Gott aber nicht wollte, dass ihnen dieses Vorhaben gelänge, verwirrte er ihre Sprache, so dass keiner mehr seine Mitmenschen verstehen konnte.

Im Laufe meines Aufsatzes werden die Leser erkennen, weshalb ich im Titel auf diese biblische Geschichte Bezug nehme.

2. Hinführung zum Thema

Zu Beginn möchte ich über zwei weit zurückliegende, aber inhaltlich zusammenhängende Erlebnisse berichten und damit die Aufmerksamkeit der Leser auf ein Problemfeld lenken, das in den letzten Jahrzehnten unbemerkt von der breiten Öffentlichkeit riesige Dimensionen angenommen hat und von dem ich befürchte, dass es unsere technische Zivilisation sehr bald vor nahezu unlösbare Schwierigkeiten stellen wird.

(1) In den Herbstsemesterferien 1961 arbeitete ich als Werkstudent in der Konstruktionsabteilung der Autofirma Porsche in Zuffenhausen, wo ich ein paar grundlegende Programme zur Ingenieursarithmetik entwickelte. Ein halbes Jahr danach, als ich bereits wieder im Studium war, rief mich ein ehemaliger Kollege von Porsche an und bat um einige Erklärungen zu Teilen der von mir entwickelten Programme, deren Funktionalität er erweitern sollte.

(2) Im Sommer 1973 holte mich ein ehemaliger Studienfreund in ein Projekt, bei dem es um die sog. „Nachdokumentation“ eines Softwaresystems ging, welches von der Firma Siemens mit einem Aufwand von ca. 400 Mitarbeiterjahren zur Nutzung bei der Olympiade 1972 in München entwickelt worden war. Dieser Studienfreund erzählte mir damals den folgenden Witz: *„Frage: Was sind die wichtigsten Informationen für einen Softwareprojektmanager? Antwort: Die Telefonnummern seiner ehemaligen Mitarbeiter.“*

3. Ein revolutionärer technologischer Wandel

Die beiden im Abschnitt 2 beschriebenen Erlebnisse deuten darauf hin, dass es im vorliegenden Aufsatz um das Problem der Verteilung des Knowhows über komplexe technische Systeme geht, deren Funktionalität zum Teil durch Software realisiert wird. Bis ungefähr zum Jahre 1965 spielte Software in komplexen technischen Systemen – man denke an Pkws, Kraftwerke, Flugzeuge oder Fernsehsender – noch keine wesentliche Rolle. Dann erst gab es den revolutionären technologischen Wandel, der zu einer geradezu explosionsartigen Zunahme der wirtschaftlichen Bedeutung von Software führte. Abbildung 1 veranschaulicht den Sachverhalt, dass dieser Wandel nicht alleine durch die Erfindung des Computers ausgelöst wurde, sondern dass noch die Entwicklung der Mikroelektronik hinzukommen musste. Diese hat nämlich eine Reduktion des Raumbedarfs für Computer und Speicher um Faktoren in der Größenordnung von Millionen gebracht, was zur Folge hatte, dass nun Computer als Komponenten in Systemen beliebiger Funktionalität benutzt werden konnten. So sind beispielsweise in modernen Pkws oder Flugzeugen jeweils etliche Computer zur Realisierung von Steuerungsfunktionen enthalten, die im Vergleich zur Vorcomputerzeit so komplex sind, dass sie dem Nichtfachmann als „Zaubereien“ erscheinen. Diese Zaubereien werden ausschließlich durch entsprechende Software erreicht, wobei der Umfang der Software über die Jahre hinweg immer größer werden konnte.

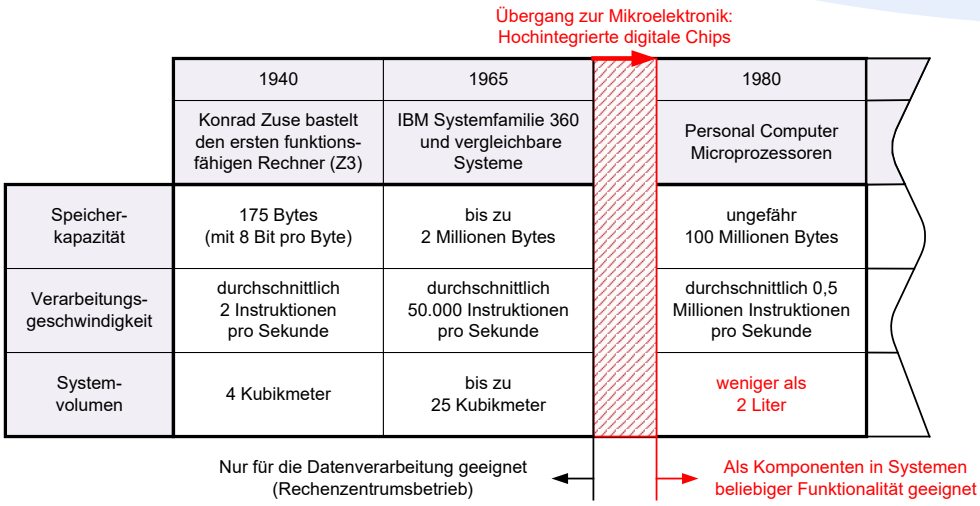


Abb. 1 Die Rolle von Computern seit ihrer Erfindung

Das Fassungsvermögen der Speicher für die Software ist nämlich aufgrund der Mikroelektronik in schier unvorstellbarem Maße gewachsen. Während in der Anfangszeit der Computertechnik in einen Programmspeicher nur ungefähr so viel Software hineinpasste, wie ein einzelner Programmierer in einem halben Jahr programmieren konnte, ist das Fassungsvermögen der Speicher inzwischen so groß, dass man darin heute problemlos das Millionenfache unterbringen kann.

4. Das Problem der Knowhow-Verteilung

Planung, Entwurf und Realisierung komplexer technischer Systeme erfordern eine hochgradige Arbeitsteilung. Das bedeutet, dass Hunderte oder gar Tausende von Fachleuten, die sich größtenteils nie treffen, wohlkoordiniert zusammenwirken müssen. Dazu müssen ihnen die Informationen über ihre jeweiligen Teilaufgaben in genormter Form über ein wohldefiniertes Netzwerk von Kanälen zufließen. Diesen Informationsfluss zu organisieren ist die Aufgabe von Ingenieuren. Konkret äußert sich diese Aufgabe in den folgenden drei Fragen:

1. Anhand welcher Dokumente können die Teilaufgaben abgegrenzt werden?
2. Anhand welcher Lehrunterlagen können diejenigen Personen geschult werden, die im Laufe des Projekts neu in die Entwicklermannschaft aufgenommen werden sollen?
3. Wie können sich Personen, die nicht an der Planung und Realisierung des ursprünglichen Systems beteiligt waren, später das Wissen verschaffen, welches sie benötigen, um Fehler im System zu korrigieren oder um das System um neue Funktionen zu erweitern?

Die Notwendigkeit hochgradiger Arbeitsteilung im technischen Bereich ist eine Folge der großen Variationsbreite des Knowhows, auf dem die Machbarkeit komplexer technischer Systeme beruht. Abbildung 2 gibt eine grobe Übersicht über die zeitliche Entwicklung des Knowhow-Bedarfs im Bereich der Technik.

Die vor 1800 benutzten technischen Systeme waren von so geringer Komplexität, dass fast jedermann durch bloßes Betrachten der Systeme die Kausalzusammenhänge erkennen konnte, auf denen das Funktionieren dieser Systeme beruhte, und die man kennen musste, damit man die Systeme nutzen konnte. Die nächsthöhere Komplexitätsstufe erforderte dann bereits eine Aufteilung des Knowhows auf kleine Gruppen von Knowhow-Trägern, von denen jeder nur noch für einen speziellen Teil des Gesamtwissens über das System zuständig war. Ungefähr um 1900 war das verfügbare technische und naturwissenschaftliche Wissen bereits so umfangreich, dass man nun Systeme bauen konnte, deren Komplexität nur noch durch hochgradige Arbeitsteilung beherrscht werden konnte. Das war auch die Zeit des Entstehens der modernen Ingenieurdisziplinen Maschinenbau und Elektrotechnik. Es ist kennzeichnend für diese Disziplinen, dass im ersten Teil des jeweiligen Studiums die Vermittlung der genormten Darstellungsformen – insbesondere Konstruktionszeichnungen und Schaltpläne – eine zentrale Rolle spielt, denn auf diesen beruht die Effizienz der hochgradigen Arbeitsteilung.

In Abbildung 2 behaupte ich, dass die kommunikative Beherrschung der arbeitsteilig entstehenden Systeme verloren ging, als die Funktion der Systeme in zunehmendem Maße durch Software bestimmt wurde. Das war ungefähr im Jahre 1965. Im folgenden Abschnitt wird aufgezeigt, was zu den bereits gelösten Problemen der hochgradigen Arbeitsteilung hinzugekommen ist, wodurch ein neues und noch nicht gelöstes Problem entstand.

5. Die Besonderheit informationstechnischer Systeme

Informationstechnische Systeme zeichnen sich dadurch aus, dass man durch bloße Analyse ihres Aufbaus und Verhaltens nicht zwingend auf ihre Funktion schließen kann, sondern dass man dazu auch noch das Wissen benötigt, wie die beobachtbaren Sachverhalte zu interpretieren sind. So kann man beispielsweise eine Verkehrsampel nur verstehen, wenn man weiß, dass das rote Signal „Stop“ und das grüne Signal „Freie Fahrt“ bedeuten. Die Verkehrsampel ist also ein informations-

	vor 1800	ab 1800	ab 1900	ab 1965
Art des Knowhows	"Natürliche" Mechanik (z. B. Kutsche und Pferd, Windmühle)	Technische Mechanik (z. B. Dampfmaschine, Nähmaschine)	Maschinenbau, Elektrotechnik Physik und Chemie (z. B. Pkw, Flugzeug, Energieversorgung, Rundfunk, Telefon)	zusätzlich Informatik
Organisationsgrad der Knowhow-Träger	kein Organisationsbedarf	kleine Teams	hochgradige Arbeitsteilung	**ungelöstes Problem**

Abb. 2 Zeitliche Entwicklung des Knowhow-Bedarfs im Bereich der Technik

technisches System. Dagegen ist eine Schreibmaschine kein informationstechnisches System, denn man braucht nicht unbedingt lesen zu können, um die Funktion einer Schreibmaschine zu verstehen. Deren Funktion besteht nämlich nur darin, die optischen Muster, die auf den Tasten stehen, aufs Papier zu drucken. Die auf diese Weise erzeugten Musterfolgen müssen keine in irgendeiner Sprache interpretierbaren Texte sein. Wenn es zum Verständnis der Funktion erforderlich wäre, die gedruckten Musterfolgen auch interpretieren zu können, wäre es unmöglich, dass jemand, der kein Englisch kann, Schreibmaschinen zur Benutzung in England baut.

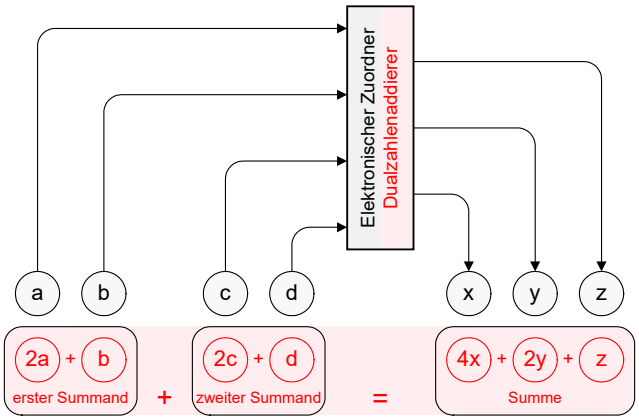
Die Interpretationsnotwendigkeit ist der Grund für die Schwierigkeiten, die sich der kommunikativen Beherrschung komplexer technischer Systeme entgegenstellen, bei denen ein Großteil der Funktionalität durch informationstechnische Komponenten realisiert wird.

Ob ein System kommunikativ beherrscht wird, entscheidet sich an der Frage, ob das Wissen des Einzelnen, welches verteilt werden sollte, standardisiert so beschrieben werden kann, dass es die Empfänger mit zumutbarem Aufwand aufnehmen und verstehen können. Dabei handelt es sich immer um Wissen, welches das Verständnis der Kausalzusammenhänge zwischen den beobachtbaren Erscheinungen ermöglicht. Falls diese Kausalzusammenhänge ausschließlich mit naturwissenschaftlichen Begriffen erklärt werden können und keine Interpretationsnotwendigkeit besteht, könnte man sich das Wissen grundsätzlich auch ohne Kommunikation mit Wissensträgern verschaffen, indem man das System analysiert und im Betrieb beobachtet. Dies ist aber nur eine grundsätzliche und keine realistische Möglichkeit, denn ein System der hier betrachteten Art besteht immer aus einer sehr großen Zahl von Komponenten, deren Zusammenwirken man nur erkennen könnte, indem man das System vollständig in seine Teile zerlegt und anschließend wieder zusammenbaut. Deshalb gibt es Pläne, die den Systemaufbau in genormter Form zeigen. Diese Pläne entstehen als Ergebnis der Entwurfsaktivitäten und müssen vorliegen, bevor mit der Systemrealisierung begonnen werden kann. Das ist der Grund, weshalb Systeme ohne Interpretationsnotwendigkeit kommunikativ so effizient beherrscht werden.

Ganz anders liegt der Fall, wenn es um Systeme geht, die man nur verstehen kann, indem man bestimmte Strukturen des Aufbaus und die im Betrieb auftretenden Erscheinungen interpretiert. Dabei betrifft die Interpretationsnotwendigkeit nicht nur die Software, sondern auch die Hardware. Denn alle Muster, die im Betrieb der Hardware auftreten und mit geeigneten Messinstrumenten beobachtet werden können, sind ausschließlich Folgen von Nullen und Einsen. Abb. 3 veranschaulicht das Problem anhand eines einfachen Beispiels. Es wird ein einfacher elektronischer Zuordner betrachtet, zu dem vier Eingangskanäle hin- und von dem drei Ausgangskanäle wegführen. Auf jedem dieser Kanäle findet man pro Beobachtungszeitpunkt entweder eine Null oder eine Eins. Die Funktion dieses Zuordners besteht also darin, jeweils dem aus vier Bits bestehenden aktuellen Eingangsmuster (a, b, c, d) ein aus drei Bits bestehendes Ausgangsmuster (x, y, z) zuzuordnen. Welche Ausgangsmuster welchen Eingangsmustern zugeordnet werden, steht in den grauen Zeilen der Tabelle in Abb. 3.

Es wäre geradezu ein Wunder, wenn jemand durch ausschließliches Betrachten der Einträge in den grauen Zeilen den Zweck dieser Zuordnung erraten würde. Deshalb gibt es in Abb. 3 zusätzlich zu den grau unterlegten Informationen auch noch die rot unterlegten Informationen, denen man entnehmen kann, dass es sich bei dem Zuordner um einen Dualzahlenaddierer handelt, der jeweils den beiden aus zwei Bits bestehenden Summanden die dreistellige Summe zuordnet.

Wenn man schon Muster aus sieben Bits nicht ohne zusätzliche Information interpretieren kann, sieht man leicht die Probleme ein, vor denen man steht, wenn es um die Interpretation von einer Million oder noch mehr Bits geht. In jedem hochintegrierten digitaltechnischen Baustein – Chip genannt – kommen derart riesige Zahlen von Bits vor.



0	0		0	0			0	0	0
0	+	0	=	0					
0	0		0	1			0	0	1
0	+	1	=	1					
0	0		1	0			0	1	0
0	+	2	=	2					
0	0		1	1			0	1	1
0	+	3	=	3					
0	1		0	0			0	0	1
1	+	0	=	1					
0	1		0	1			0	1	0
1	+	1	=	2					
0	1		1	0			0	1	1
1	+	2	=	3					
0	1		1	1			1	0	0
1	+	3	=	4					
1	0		0	0			0	1	0
2	+	0	=	2					
1	0		0	1			0	1	1
2	+	1	=	3					
1	0		1	0			1	0	0
2	+	2	=	4					
1	0		1	1			1	0	1
2	+	3	=	5					
1	1		0	0			0	1	1
3	+	0	=	3					
1	1		0	1			1	0	0
3	+	1	=	4					
1	1		1	0			1	0	1
3	+	2	=	5					
1	1		1	1			1	1	0
3	+	3	=	6					

Abb. 3 Zum Problem der Interpretation von Null-Eins-Mustern

6. Die aktuelle Situation und ihre nicht abschätzbaren Gefahren

Im Folgenden wird anhand von Abb. 4 sowohl die Fertigung als auch der Betrieb eines informationstechnischen Systems betrachtet. Die vorkommenden Symbole sind wie folgt zu interpretieren:

Jedes Rechteck symbolisiert einen Akteur, der durch seine Aktionen zur Fertigung oder zum Betrieb des informationstechnischen Systems beiträgt. Manche dieser Akteure sind Menschen, die übrigen sind technische Systeme. Es wird angenommen, dass man nicht ins Innere der Akteure hineinschauen kann. Um etwas über die Art ihrer Aktionen zu erfahren, muss man deshalb auf die sog. „Aktionsfelder“ schauen, wo sich die Aktionen äußern. Alle flächigen Netzknoten in Abb. 4, die keine Rechtecke sind, stellen Aktionsfelder dar.

Das zu fertigende und anschließend nutzbare System entsteht in dem großen gelben Aktionsfeld oben in der Abbildung. Auf diesem Aktionsfeld agiert nur ein einziger Akteur, was man an dem einzigen Pfeil erkennt, der auf dem Rand dieses Aktionsfeldes endet. Dieser Pfeil geht von der Vereinigung der technischen Hardwarefertigungshelfer aus, welche für die Schaffung des in dem großen Aktionsfeld dargestellten Systems zuständig ist. Erst nachdem das System geschaffen ist, kann die Vereinigung der technischen Softwareentwicklungshelfer Kontakt zu ihm aufnehmen und ihm die Programme zur Einspeicherung übergeben. Dies erfolgt über den Kanal, der als kleines kreisförmiges Aktionsfeld zwischen der Vereinigung der technischen Softwareentwicklungshelfer und dem neu geschaffenen System liegt. Die Kanäle zwischen den Menschen, die für die Hardware- und Softwareentwicklung zuständig sind, und ihren jeweiligen technischen Helfern zeigen, dass die Helfer nicht autonom agieren, sondern von den Menschen gelenkt werden.

Das Interesse im vorliegenden Aufsatz gilt den Dokumenten, die im Zuge der Hardware- und Softwareentwicklung entstehen oder zumindest entstehen sollten. Dokumente, die zwangsläufig entstehen, sind die Programme und die Hardwarebeschreibung, die von den jeweiligen Entwicklern in formalen Sprachen formuliert werden müssen, damit sie an die jeweiligen technischen Helfer übergeben werden können. Diese Dokumente sind für die Weitergabe des grundsätzlichen Knowhows der Entwickler völlig ungeeignet, denn die formalen Sprachen wurden nicht für die zwischenmenschliche Kommunikation geschaffen, sondern für die Kommunikation zwischen Menschen und technischen Systemen.

Da die Systembenutzer selbstverständlich erwarten, dass zu dem System auch ein Benutzerhandbuch mitgeliefert wird, wird ein solches auch in fast allen Fällen erstellt. Allerdings bleibt der dafür aufgebrachte Aufwand meist recht gering, so dass die didaktische Qualität der Dokumente selten befriedigend und oft sogar äußerst mangelhaft ist. Viele Benutzer haben sich an diese Situation gewöhnt und versuchen erst gar nicht, das Benutzerhandbuch sorgfältig zu studieren. Sie erleben es als effizienter, sich durch „intuitives Rumprobieren“ mit der für sie relevanten Funktionalität des Systems vertraut zu machen.

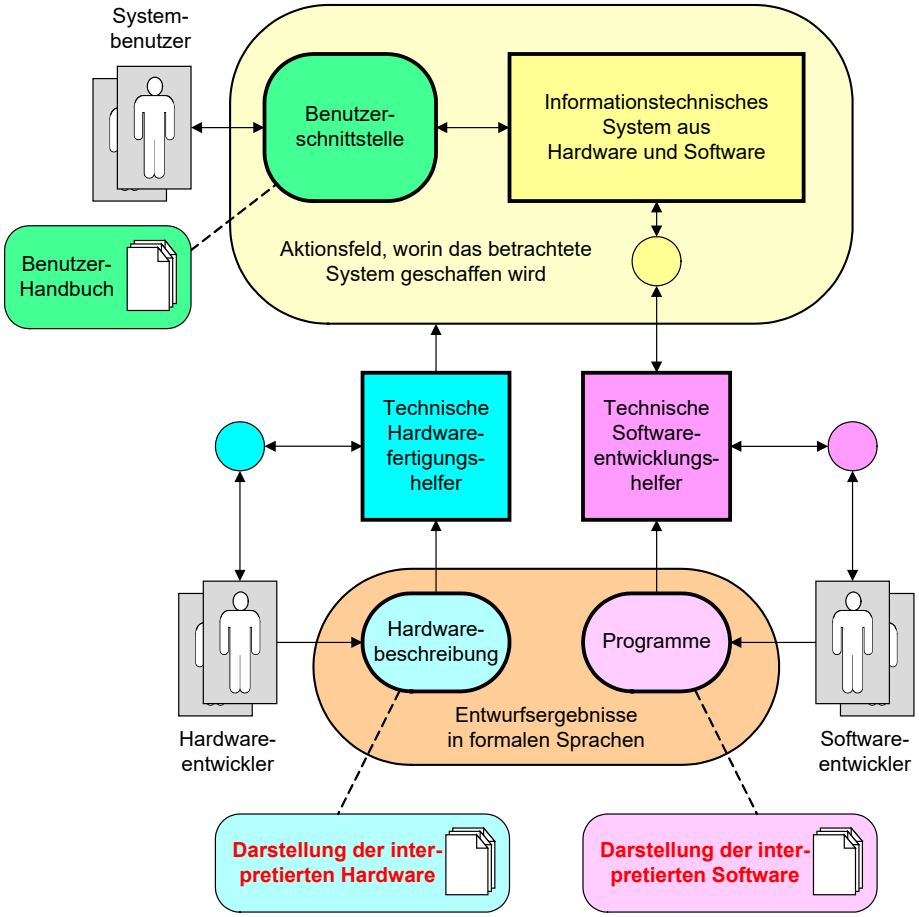


Abb. 4 Akteure und Dokumente im Bereich der Informationstechnik

In Abb. 4 kommen drei Arten von Dokumenten vor, die für die zwischenmenschliche Kommunikation erstellt werden oder zumindest erstellt werden sollten. Zum einen ist dies das Benutzerhandbuch und zum anderen sind dies die Darstellungen der interpretierten Strukturen der Hardware und der Software. In der Abbildung wird nicht gezeigt, welche Akteure diese Dokumente erstellen bzw. erstellen sollten. Durch die gestrichelten Brücken wird lediglich gezeigt, auf welche Inhalte sich diese Dokumente beziehen.

Die Darstellungen der interpretierten Hardware und Software sollen der Weitergabe des Knowhows der jeweiligen Entwickler bezüglich der von ihnen entworfenen Strukturen dienen. Deshalb kommen als Autoren dieser Dokumente eigentlich nur diese Entwickler selbst in Frage. Diese Bedingung wird aber so gut wie nie erfüllt. Vielmehr werden diese Dokumente – falls sie überhaupt erstellt werden – als sog. „Nachdokumentati-

on“ von Kollegen der Entwickler erstellt, die mehr oder weniger erfolgreich versuchen, sich mit der Gedankenwelt der Entwickler vertraut zu machen. Doch unabhängig davon, ob die Entwickler selbst oder ihre Kollegen die Nachdokumentation erstellen, ist verständlicherweise ihre Motivation für diese Tätigkeit recht begrenzt. Denn das System ist ja schon fertig und kann benutzt werden, so dass man sich lieber mit Fragen des Entwurfs neuer Systeme befassen möchte als seine Zeit mit einer – von vielen als nutzlos betrachteten – „Vergangenheitsbewältigung“ zu vergeuden. Deshalb ist es nicht verwunderlich, dass die Qualität der „Nachdokumente“ dem Zweck der Knowhow-Weitergabe fast nie gerecht wird.

Ich bin überzeugt, dass man eine Knowhow-Weitergabe durch Nachdokumentation gar nicht versuchen sollte, sondern dass man die damit verbundenen Probleme vermeiden muss, indem man sich den Bereich der Systeme ohne Interpretationsnotwendigkeit zum Vorbild nimmt. Dort hat es sich seit Jahrzehnten bewährt, zuerst zu planen und zu entwerfen und erst mit der Systemrealisierung zu beginnen, wenn die Entwurfsdokumente vorliegen. Diese Dokumente sind nämlich auch die Voraussetzung für die Prüfung des Entwurfs durch Fachkollegen, so dass mögliche Mängel frühzeitig entdeckt und beseitigt werden können.

Möglicherweise fragt sich der eine oder andere Leser, was denn daran schlimm sein soll, wenn die Knowhow-Verbreitung nicht gewährleistet ist. Dieser Leser sollte aber bedenken, dass es ja nicht nur die Software gibt, die in den individuell benutzten Systemen sitzt, sondern dass heute praktisch alle Infrastruktursysteme – Energieversorgung, Verkehrswesen, industrielle Fertigungsanlagen, globale Kommunikation, Bankwesen, Gesundheitswesen, öffentliche Verwaltung usw. – vernetzt und durch Software geprägt sind. Die Software in diesen Systemen ist so umfangreich, dass die formalsprachlich formulierten Programme, würde man sie ausdrucken, einen Papierstapel ergäben, der den Eiffelturm um ein Vielfaches überragen würde. Glaubt man wirklich, dass diese Systeme so durchdacht und zuverlässig sind, dass es nichts ausmacht, wenn im Grunde „niemand mehr durchblickt“?

7. Die Frage nach den Ursachen der „Misere“

Es gibt meines Erachtens zwei völlig unterschiedliche Bereiche, in denen grundsätzliche Änderungen geschehen müssten, damit die oben geschilderte kritische Entwicklung gestoppt und entschärft werden könnte. Der eine Bereich ist die akademische Informatik, und der andere Bereich ist unser auf Wachstum fixiertes Wirtschaftssystem.

Zur Zeit der Gründung der akademischen Informatik gab es noch keine Softwaresysteme, deren Umfang eine hochgradige Arbeitsteilung erforderlich gemacht hätte. So war es ganz natürlich, dass nicht Ingenieure, sondern Mathematiker als erste Informatikprofessoren berufen wurden, die dann die Lehrinhalte des neuen Faches festlegten. Und als Folge davon ist es auch verständlich, dass hochgradige Arbeitsteilung in den Informatiklehrplänen nicht vorkam, denn so etwas kennen die Mathematiker in ihrer Disziplin nicht. Erstaunlicherweise fühlt sich die akademische Informatik aber immer noch nicht zuständig für die oben geschilderten Probleme, so dass diese auch heute noch nicht in den Lehrplänen zu finden sind. Es war für mich ein sehr bezeichnendes Erlebnis, als vor ein paar Jahren ein renommierter deutscher Informatikprofessor die Veröffentlichung eines von mir verfassten Aufsatzes mit dem Titel „Defizite im Software Engineering“ mit der Begründung ablehnte, in diesem Aufsatz gehe es nicht um Themen der wissenschaftlichen Informatik, sondern um Managementprobleme. So ist es kein Wunder, dass Absolventen eines Informatikstudiums, worin sie nicht das Geringste über die oben dargestellte Problemwelt erfahren haben, überfordert sind, wenn sie in der Berufspraxis plötzlich mit dieser Problemwelt konfrontiert werden.

Als zweiten Ursachenbereich betrachten wir nun noch unser auf Wachstum fixiertes Wirtschaftssystem. Dieses System hat in den letzten Jahrzehnten die Schere zwischen Arm und Reich immer weiter aufgehen lassen. Deshalb gibt es nun sehr viele Reiche, die viel mehr Geld haben, als sie durch eine egal wie luxuriöse Lebenshaltung verbrauchen könnten. Es ist allgemein akzeptiert, dass sie ihr überschüssiges Geld „anlegen“, damit es wachsen kann. Wachsen kann es aber nur, wenn es immer wieder sog. Innovationen gibt. In der Menschheitsgeschichte gab es tatsächlich immer wieder Innovationen, die einen geradezu revolutionären Fortschritt brachten – man denke beispielsweise an den Verbrennungsmotor, der die Zugtiere Pferd und Ochse ablöste, oder an die Entdeckung des Zusammenhangs zwischen Elektrizität und Magnetismus, die zur Entstehung der Elektrotechnik mit all ihren vielfältigen Systemen führte. Diese Innovationen sind aber „vom Himmel gefallen“ und wurden nicht zielstrebig geschaffen. Da nun aber heute Anlagemöglichkeiten in großem Umfang gebraucht werden, kann man nicht warten, bis neue Innovationen vom Himmel fallen, sondern man fordert, dass Innovationen geschaffen werden. Und dies ist mit Hilfe von Software tatsächlich möglich geworden. Egal wie die Schlagwörter heißen – Industrie 4.0, Smart Factory, Smart City, Künstliche Intelligenz, autonom fahrende Pkws –, all dies beruht auf Software. Und diese Software muss natürlich schnell und mit möglichst geringen Kosten entwickelt werden. Da kommt es doch sehr gelegen, dass Software durch bloßes Hinschreiben „gefertigt“ werden kann und man dazu keine Gießereien, Walzwerke, Drehbänke oder Schweißroboter benötigt, sondern nur Programmierer, die begeistert ihre Programme basteln und „zum Laufen bringen“. Da bleibt natürlich keine Zeit, über die hier aufgezeigten Probleme und erst recht nicht über die Frage nach Lösungsmöglichkeiten nachzudenken.

Zum Autor
Prof. Dr.-Ing. Siegfried Wendt

geb. 1940, Studium der Elektrotechnik und Promotion an der Technischen Hochschule Karlsruhe, Hochschullehrer für Digitale Systeme, State University of New York in Buffalo, USA (drei Jahre), Universität Hamburg (drei Jahre), Universität Kaiserslautern (24 Jahre), Gründungsdirektor des Hasso-Plattner-Instituts in Potsdam (sechs Jahre). Im Ruhestand seit 2005.

s. a.: http://de.wikipedia.org/wiki/Siegfried_Wendt