Das Kommunikationsproblem der Informatiker und ihre Unfähigkeit, es wahrzunehmen

Siegfried Wendt
Universität Kaiserslautern

Abstract: The Communication Problem of Software Engineers and their Inability to notice it

One important characteristic of mature academic disciplines is the high quality of the communication between professionals. Over more than two decades, the author has analysed the communication in the field of software engineering, and he came to the conclusion that this is still a very immature discipline compared to the traditional engineering fields. Though the deficits are obvious, the professionals don't seem to notice them. Many examples are given which indicate that conceptual confusion is widely accepted and that optimal representations, once found, are not preserved because their relevance is not realized. The dominant role of documentation instead of making designs by drawing plans is another indication that software engineering has not yet reached the necessary maturity. Up to now, a high academic reputation in software engineering can only be achieved by developing algorithms and creating formal representations. Therefore, computer scientists and academic software engineers are not motivated to search for optimal informal representations of complex semantic structures. The author considers software engineering as a branch of systems engineering, and this view guided his search for representations which provide optimal intuitive insight into complex informational systems. His results have been applied successfully in industrial projects.

1. Die Kritiker in der Minderheit

Wer aus der frischen Luft des Waldes kommend einen miefigen Saal betritt, wird selbstverständlich die Luft im Saal als schlecht bezeichnen, wogegen diejenigen, die sich schon lange in diesem Mief aufgehalten haben, gar nicht wahrnehmen können, daß es hier stinkt. Sie werden allerdings auch nicht kämpferisch behaupten, die Luft im Saal sei besonders gut und besser als die im Wald.

So behaupten auch die Informatiker nicht, daß sich ihr Kommunikationsverhalten durch besonders hohe Qualität auszeichne, aber sie sind durchaus der Meinung, daß die Art und Weise, wie sie kommunizieren, ganz in Ordnung sei. Somit kann es für sie auch keinen Anlaß geben, über eine Verbesserung der Qualität ihres Kommunikationsverhaltens nachzudenken.

Derjenige, der von draußen reingekommen ist und dabei gemerkt hat, wieviel schlechter die Luft im Saal als die im Wald ist, steht vor der Frage, ob er die schlechte Luft im Saal als gegeben hinnehmen muß, oder ob er nach Möglichkeiten suchen soll, die Luft im Saal zu verbessern. Dabei kann er verständlicherweise keine Unterstützung von den Leuten im Saal erwarten, denn diese sind ja mit der derzeitigen Qualität der Luft ganz zufrieden. Es ist dagegen sogar möglich, daß sie seinem Bemühen, den Saal zu lüften, Widerstand entgegensetzen, weil dadurch ihr gemütliches Beisammensein unnötigerweise gestört wird. Derjenige, der weiß, wie schlecht die Luft im Saal ist und wieviel besser sie sein könnte, wird möglicherweise versuchen, die Leute im Saal zu überreden, für eine kurze Zeit den Saal zu verlassen und mit ihm an die frische Luft zu gehen. Er ist nämlich überzeugt, daß die Leute, wenn sie aus der frischen Luft wieder in den Mief des Saales

zurückkehren, von ganz alleine dafür sorgen werden, daß der Saal gelüftet wird. Aber warum sollte jemand, der gemütlich im Saal sitzt und sich wohlfühlt, auf jemanden hören, der ihn bittet, mit ihm einmal kurz nach draußen zu gehen?

Eine klare schlüssige Begriffswelt und die dieser Begriffswelt angemessenen Darstellungsmittel stellen in dieser Analogie die Entsprechung zur frischen Luft dar. Da die Informatiker aber gar keinen Grund sehen, nach einer klareren Begriffswelt und besseren Ausdrucksmitteln zu suchen, nehmen sie entsprechende Angebote gar nicht zur Kenntnis oder, falls sie sie doch zur Kenntnis nehmen, machen sie sich nicht die Mühe, sie genau zu studieren. Sie verhalten sich wie jemand, dem man die Konstruktionspläne für ein Perpetuum Mobile vorlegt, der doch weiß, daß es ein Perpetuum Mobile gar nicht geben kann und daß es deshalb eine verlorene Mühe wäre, die Pläne genau zu studieren.

Derjenige, der den Unterschied zwischen frischer Luft und Mief kennengelernt hat, kann vorläufig nur in den Sälen, für deren Luft er selbst verantwortlich ist, den Mief vertreiben. Hinsichtlich der anderen Säle aber, worin die anderen Leute zur Zeit noch so selbstzufrieden hocken, braucht er nur zu warten, bis dort die Mangelerscheinungen so offenkundig werden, daß man dann auch dort gerne die Existenz frischer Luft zur Kenntnis nehmen wird.

2. Die Mangelerscheinungen

Nachdem bisher das Problem nur anhand einer Analogie charakterisiert wurde, soll es nun konkretisiert werden, d.h. es sollen nun die Mangelerscheinungen aufgeführt werden, die man mühelos feststellen kann, wenn man die Kommunikationsformen der Informatik an Maßstäben mißt, die man aus anderen Disziplinen gewohnt ist.

> Sprachliche Begriffsverwirrungen werden kritiklos hingenommen.

Man betrachte die folgenden Beispiele aus Informatiktexten:

- Rechenvorgänge sind aus Rechenschritten, den Operationen zusammengesetzt. Eine Operation ist eine Anweisung oder ein Klartext.
- Das Prädikat für die Eingabe ist `read(x)`. Dieses Ziel liest einen Term vom aktuellen Eingabestrom.
- Durch die Auftrennung eines Objekts in einen Objekttyp und einen Objektnamen lassen sich bequem Systeme beschreiben.
- Das Ereignis beschreibt das Eingetretensein eines Zustands, der eine Folge bewirkt.
- Not every object required to carry out an operation flows through its trigger.
- The control condition does not continuously check if the condition is true.
- Each activity in the chart may contain a control activity whose role is to supervise the behaviour of its siblings. The internal descriptions of control activities are given by statecharts. A statechart starts activities through actions on transitions. When an activity with a control activity is started, the control activity and its associated statechart are started.

Die hier gestiftete Verwirrung entspricht derjenigen, die entstünde, wenn in einem Text der Unterschied zwischen einem Sänger, seinem Gesang, seinen Noten und dem Konzertsaal verwischt würde.

Man muß selbstverständlich akzeptieren, daß es disziplinspezifische Fachsprachen gibt, die nicht jeder Laie verstehen kann. Wer keine Noten lesen kann, gehört nicht zum vorgesehenen Leserkreis eines Aufsatzes über den Kontrapunkt. Man muß also fragen, ob die oben angeführten Beispielstexte in einer Informatikfachsprache abgefaßt sind, so daß sie nur dem Laien konfus erscheinen, aber für die Informatiker klar verständlich sind. Diese Frage läßt sich leicht beantworten, indem man die Texte überdurchschnittlich begabten und sehr gut ausgebildeten Informatikern vorlegt und sie um eine Interpretation bittet. Dann stellt sich heraus, daß diese Texte auch von den Informatikern als konfus erlebt werden und sie auf Vermutungen bezüglich der Interpretation angewiesen sind.

Nun könnte man natürlich einwenden, daß es in jeder Disziplin möglich sei, Beispielstexte von unfähigen Autoren zu zitieren. Deshalb komme den angeführten Beispielen keinerlei Beweiskraft zu. Gegen einen solchen Vorwurf kann hier tatsächlich kein wissenschaftliches Argument vorgebracht werden, denn dieses könnte ja nur darin bestehen, daß man auf sehr viele unabhängige andere Wissenschaftler verweist, die in ähnlicher Weise wie der Autor über zwei Jahrzehnte hinweg das Kommunikationsverhalten der Informatiker studiert haben und dabei erkennen mußten, daß in dieser Disziplin ein unbeholfenes "Schauderwelsch" nicht die Ausnahme, sondern die Regel ist.

Wer die Texte Hegels mit den Texten Schopenhauers vergleicht, ist geneigt, Schopenhauer Recht zu geben, der behauptet, Hegel könne nicht klar schreiben, und daraus müsse man zwangsläufig schließen, daß er auch nicht klar denken könne.

Darf man also aus den konfusen Texten der Informatiker schließen, daß sie nicht klar denken können? Dies wäre sicher kein logisch zwingender Schluß. Es ist durchaus denkbar, daß sich in einer Disziplin die Gepflogenheit herausgebildet hat, sehr klare Sachverhalte sehr umständlich auszudrücken gemäß der Redewendung "Warum etwas einfach machen, wenn es auch umständlich geht." Man kann nie sicher sein zu wissen, was die Leute zu dem assoziieren, was sie reden oder schreiben. Dennoch glaubt der Autor aufgrund seiner jahrelangen intensiven Beobachtung des Kommunikationsverhaltens der Informatiker zu der Vermutung berechtigt zu sein, daß die Schwierigkeiten der Informatiker daher kommen, daß sie sich bei ihrer Kommunikation grundsätzlich auf einer niederen Abstraktionsebene bewegen, auch wenn es um Sachverhalte geht, die erst auf einer höheren Abstraktionsebene überhaupt ausdrückbar werden. Man könnte sagen, daß sie den Wald vor lauter Bäumen nicht sehen, genauer gesagt, daß sie nur den Begriff Baum und das dazugehörige Wort kennen, aber nicht den Begriff Wald und dementsprechend auch kein zugehöriges Wort. Eine Fortführung dieser Überlegungen wird weiter unten angeschlossen; an dieser Stelle geht es ja nur um die Charakterisierung von Mangelerscheinungen.

Die Relevanz optimaler Darstellungsformen wird nicht erkannt,
und deshalb werden gefundene Optima nicht bewahrt.

Jeder kennt aus seinem elementaren Mathematikunterricht die Bedeutung des Gleichheitszeichens. Wenn also jemand die Form

$$x = x + 2$$

sieht, wird er denken, es handle sich um eine Gleichung mit einer Unbekannten, deren Auflösung zu einer Zahl führen müßte, die ihren Wert nicht ändert, wenn man 2 hinzuaddiert. Da es eine solche Zahl nicht gibt - außer in der Welt der Kardinalzahlen für unendliche Mengen -, wird der Leser diese Gleichung als unlösbar bezeichnen.

Wenn er dagegen den Ausdruck

$$x := x + 2$$

sieht, wird er entweder nicht wissen, wie er diesen Ausdruck interpretieren soll, oder aber er wird - falls er programmieren kann - damit die Vorstellung einer Anweisung verbinden, die verlangt, daß auf den aktuellen Inhalt der Speicherzelle x die Zahl 2 hinzuaddiert werden soll - was nur möglich ist, wenn die Speicherzelle vorher schon eine Zahl enthält. Er wird zumindest aber nicht auf die Idee kommen, es könne sich um eine Gleichung handeln, denn die Verbindung des Doppelpunkts mit einem Gleichheitszeichen erzeugt eine Vorstellung von Unsymmetrie, die sich mit der intuitiven Vorstellung der Symmetrie der beiden Waagschalen auf den beiden Seiten einer Gleichung nicht verträgt.

Während in der Programmiersprache FORTRAN noch das Gleichheitszeichen als Zuweisungsoperator verwendet wurde, hat man bereits in der Programmiersprache ALGOL60 die unsymmetrische Symbolkombination := für den Zuweisungsoperator festgelegt. Zweifellos bedeutete es einen Rückschritt, als man bei der Schaffung der Programmiersprache C wieder zu der in FORTRAN üblichen Symbolisierung zurückkehrte.

Ein zweites Beispiel für die fehlende Wertschätzung von optimalen Darstellungsformen ist die Prädikatsschreibweise in Bedingungen: In ALGOL60 wurden Bedingungsprädikate durch Schlüsselwörter und nicht durch mathematische Symbole ausgedrückt:

```
IF j EQUAL 5 THEN ...
```

Man hätte hier zwar auch das Gleichheitszeichen verwenden können, aber man hat dieses bewußt vermieden, um eine größere symbolische Distanz zum Zuweisungsoperator := zu bekommen. In der Programmiersprache C wird dagegen geschrieben

```
if (j == 5) ...
```

Diese Schreibweise ist sehr weit vom Optimum entfernt. Häufig kommt es vor, daß ein C-Programmierer versehentlich

```
if (j = 5) ...
```

schreibt, was noch nicht einmal vom Compiler als Fehler festgestellt werden kann, weil in C bei der Berechnung von Prädikaten auch Zustandsänderungen als Seiteneffekte akzeptiert werden.

Man kann der weltweiten Informatikergemeinde den Vorwurf nicht ersparen, daß sie das Qualitätsbewußtsein bezüglich menschenfreundlicher Kommunikationsformen entweder nie gehabt oder inzwischen längst verloren hat.

Die Nichtexistenz von Plänen und die bedeutende Rolle der Dokumentation in der Informatik weisen auf eine mangelnde Reife im Vergleich mit den traditionellen Ingenieurwissenschaften hin.

Wenn ein Architekt ein Gebäude entwirft, ein Maschinenbauer eine Getränke-Abfüllanlage konstruiert oder ein Elektroingenieur eine Automatisierungselektronik konzipiert, zeichnen sie Pläne - Baupläne, Konstruktionspläne oder Schaltpläne. Ihre Tätigkeit bezeichnen sie als entwerfen, konstruieren oder planen, aber selbstverständlich nicht als dokumentieren, obwohl die Pläne ja auch Dokumente sind.

In der Informatik unterscheidet man zwischen Mitdokumentation und Nachdokumentation, um auszudrücken, ob während oder erst nach der Softwareentwicklung dokumentiert wird. Durch diese Vorsilben wird dem Wort Dokumentation Gewalt angetan, denn umgangssprachlich bezeichnet Dokumentation immer eine nachträgliche Zusammenstellung von Dokumenten. Man denke an eine Dokumentation der französischen Revolution oder der Judenverfolgung im Dritten Reich. Die Erstellung von Dokumentationen ist normalerweise eine Aufgabe für Historiker. Das Wort Dokumentation gibt keinerlei Hinweis auf die Art der verwendeten Dokumente - es können Texte sein, Fotographien, Zeichnungen und sogar Gegenstände wie beispielsweise die Brillen umgebrachter Juden.

Wenn man demgegenüber von Plänen spricht, verbindet man damit unwillkürlich die Vorstellung bestimmter Plantypen, d.h. also von Darstellungen unter Verwendung bestimmter genormter Formen. In der Informatik hat sich über viele Jahre praktisch niemand die Frage gestellt, wie denn die Pläne für ein Betriebssystem oder ein Textverarbeitungssystem aussehen könnten. Es sind zwar etliche Systeme zur rechnerunterstützten Dokumentation mit Hilfe graphischer Darstellungen auf den Markt gekommen - die Kürzel SA, SADT, SART oder UML stehen für derartige Darstellungen -, aber daß selbst die Anbieter solcher Systeme ihren Plantypen keine große Bedeutung beigemessen haben, erkennt man daran, daß die Softwaresysteme zur Edition, Verwaltung und Transformation solcher Pläne selbst nicht mit dieser Art von Plänen entworfen wurden.

Die Nichtexistenz allgemein akzeptierter und brauchbarer Pläne mußte zwangsläufig zur Entstehung sogenannter Gurus führen. Ein Guru ist ein Softwareentwickler, der die inneren Strukturen eines großen Softwaresystems im Kopf hat und zu dem jeder kommen muß, der etwas Spezielles über das Innere des Systems wissen will. In den traditionellen Ingenieurwissenschaften gibt es kein derartiges Guruwesen, denn dort werden die Fragen anhand von Plänen beantwortet, die jeder Ingenieur lesen kann. Da alle Gurus eine Machtstellung haben, die sie nicht gerne verlieren möchten, bringen die Gurus immer sehr viele Gründe vor, weshalb die Einführung von Plänen in der Informatik nicht möglich sei.

Wissenschaftliche Wertschätzung kann man in der Informatik nur mit formalen Darstellungen erwerben, weshalb praktisch kein Aufwand in die Optimierung nichtformaler Darstellungen gesteckt wird.

Die Informatik hat ihre wissenschaftlichen Qualitätskriterien aus der Mathematik bezogen, wo die Konzentration auf das Formale bzw. Formalisierbare zweifellos gut begründet ist. Nun sind aber

die Gegenstände der Informatik keine mathematischen Gegenstände, sondern technische Systeme, obwohl der renommierte Informatiker Hoare [Hoare-86] sagt: Programs are mathematical expressions. Eine solche Aussage entspricht in ihrem Nutzen der Aussage: Ein Auto ist nur ein Haufen von Elementarteilchen. Die Aussage ist zwar richtig, aber es gibt keinerlei praktische Konsequenzen, die man aus ihr ziehen könnte. Es gibt allerdings sehr viele unsinnige Konsequenzen, die man aus einer solchen Aussage ziehen kann und die leider in der Informatik auch gezogen werden.

Es ist durchaus verständlich, weshalb jemand, der sich einmal in der Welt der Mathematik eingenistet hat, diese Welt nicht mehr gerne verläßt. Die Welt der Mathematik ist eine viel klarere reinere Welt als die Welt der Ingenieure, denn in der Mathematik geht es immer um die Entscheidung zwischen wahr und falsch, während es bei den Ingenieuren um Entscheidungen zwischen angemessen und unangemessen oder zweckmäßig und unzweckmäßig geht, und da sind die Entscheidungskriterien selbstverständlich nicht mehr eindeutig.

Die Unterschiede zwischen der mathematischen und der nichtmathematischen Welt fallen schon jedem Schüler auf, der Schwierigkeiten hat, Textaufgaben zu lösen, obwohl er doch seine Mathematik gut beherrscht. Er kann rechnen, aber er kann nicht ein Problem in Formeln überführen. Das Nichtformale bzw. die Brücke zwischen dem Nichtformalen und dem Formalen ist aber eigentlich das interessantere, denn das Formale alleine kann man der Maschine übergeben, die dann die sturen Rechenschritte durchführt. Wer sich auf das Formale konzentriert, sieht somit seinen primären Kommunikationspartner nicht in anderen Menschen, sondern in der Maschine. Deshalb ist er auch schnell bereit, sich mit einer hieroglyphenähnlichen Symbolik zufriedenzugeben, solange sie nur eindeutig ist. Deshalb ist es kein Wunder, daß der Quellcode häufig das einzige ist, was von einem großen Softwaresystem als Dokumentation zur Verfügung steht.

Welche Bedeutung das Nichtformale für das menschliche Verständnis hat, erkennt man schon an sehr einfachen Beispielen: In Bild 1 sind die fünf Peano'schen Axiome sowohl in der üblichen prädikatenlogischen Symbolik als auch in einer graphischen Veranschaulichung dargestellt. Es wäre völlig absurd anzunehmen, es könne jemand, der die natürlichen Zahlen als Begriff noch nicht kennt, zu diesem Begriff geführt werden, indem man ihm einfach die prädikatenlogischen Formeln vorsetzt. Es ist selbstverständlich, daß auch Peano selbst den Begriff der natürlichen Zahl schon kannte, bevor er anfing, über ihre axiomatische Festlegung nachzudenken. Aufgrund der graphischen Darstellungen in Bild 1 erkennt jeder leicht, daß die ersten vier Axiome nicht ausreichen, die natürlichen Zahlen eindeutig festzulegen; hätte man aber diese Graphen nicht, sondern nur die prädikatenlogischen Formeln der ersten vier Axiome, hätte man vermutlich große Schwierigkeiten zu begründen, weshalb man noch ein fünftes Axiom braucht.

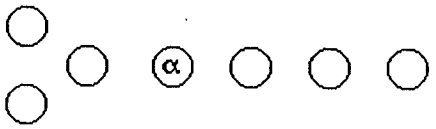
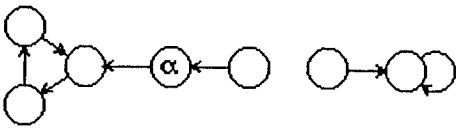
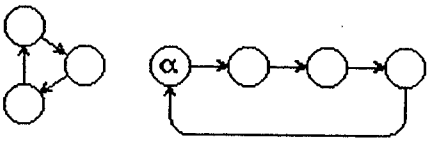
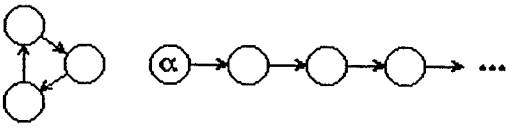
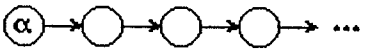
Axiome - als prädikatenlogische Formel - als übersetzte Formel in deutscher Sprache - in einer Formulierung, die auf die gewählte Veranschaulichung zugeschnitten ist	Beispiele von Strukturen, die mit den jeweiligen bisher eingeführten Axiomen verträglich sind
(1) $N(\alpha)$ Es wird die Menge aller Individuen betrachtet, auf die das Prädikat N zutrifft. Eines dieser Individuen heißt α. Es wird eine Menge kreisförmiger Knoten betrachtet. Einer dieser Knoten ist mit "α" beschriftet.	
(2) $\forall x: N(x) \rightarrow N(r(x))$ Auf der Menge der betrachteten Individuen ist eine Funktion r definiert, die in diese Individuenmenge hinein abbildet. Von jedem der Knoten geht ein Pfeil aus, der zu einem Knoten führt.	
(3) $\forall x, y: N(x) \cdot N(y) \cdot (r(x) = r(y)) \rightarrow (x = y)$ Wenn zwei Ergebnisse von r gleich sind, müssen die beiden Argumente gleich sein, d.h. r ist umkehrbar eindeutig. Es gibt keinen Knoten, auf dem mehr als ein Pfeil endet.	
(4) $\forall x: N(x) \rightarrow (r(x) \neq \alpha)$ α kommt nie als Ergebnis von r vor. Auf α endet kein Pfeil.	
(5) $\forall P: P(\alpha) \cdot (\forall x: P(x) \rightarrow P(r(x))) \rightarrow (\forall x: N(x) \rightarrow P(x))$ Jedes beliebige Prädikat P, das für α gilt und dessen Gültigkeit stets vom Argument auf das Ergebnis von r übertragen wird, gilt für alle Individuen der betrachteten Menge. Es gibt keine Knoten, die nicht in der mit α beginnenden Kette hängen.	

Bild 1 Axiome von Peano zur Definition der Struktur
"Einseitig begrenzte Kette aus unendlich vielen diskreten Gliedern"

Da die Informatiker über mathematische Sachverhalte untereinander und mit anderen sehr gut kommunizieren können, ist ihre Kommunikationsschwäche eine Konsequenz der Tatsache, daß sie überwiegend über nichtmathematische Sachverhalte kommunizieren müssen, sie dies aber bisher nicht als Problem erkannt haben oder ihm zu wenig Aufmerksamkeit widmeten.

3. Die Besonderheiten des Problems und seiner Lösung

Die Produkte, über die in den traditionellen Ingenieurwissenschaften kommuniziert werden muß, sind immer gegenständliche Produkte, die man sehen und anfassen kann. Deshalb ähneln die Darstellungen auf den Plänen auch immer den Ansichten, die man sehen kann, wenn man das Produkt von einer bestimmten Seite oder in einer bestimmten aufgeschnittenen Form betrachtet.

Bei Softwareprodukten dagegen gibt es solche Ansichten nicht, denn dort ist das einzig konkret Wahrnehmbare der Quellcode und die Ein- Ausgabeerscheinungen auf dem Bildschirm oder auf dem Druckerpapier. Man muß also in der Informatik eine größere Abstraktionsleistung erbringen als in den traditionellen Ingenieurwissenschaften, wenn man die Individuen finden will, die man auf den Plänen symbolisiert. Daß dieses Problem in der Anfangszeit der Informatik fast von niemandem und bis heute nur von recht wenigen erkannt wurde, ist zwar dem Autor, der dieses Problem schon vor über 20 Jahren in Angriff nahm, nicht wirklich verständlich, aber er versucht wenigstens, Argumente zu sammeln, die das Versäumnis entschuldbar erscheinen lassen. In der Anfangszeit der Informatik waren die Softwaresysteme so klein, daß es möglich war, alles Wesentliche in Form des Quellcodes zu vermitteln. Solange man nur eine Hundehütte zusammennagelt, braucht man sich keine Gedanken über die Gestaltung von Bauplänen zu machen. Das Problem ist erst im Laufe der Jahre dringlicher geworden, als die Programmsysteme immer komplexer wurden. Daß dennoch fast niemand das Problem erkannte, läßt sich möglicherweise durch eine Analogie erklären: Es wurde experimentell nachgewiesen, daß sich Leute, die in einem anfänglich wohltemperierten Raum sitzen, dessen Temperatur mit der Zeit langsam erhöht wird, erst bei sehr hoher Temperatur eines Problems bewußt werden und anfangen zu überlegen, wie man die Situation verbessern könne.

Die "Temperatur im Informatikraum" ist nun inzwischen doch so hoch geworden, daß immer mehr Informatiker anfangen, über eine Abhilfe nachzudenken. Inzwischen gibt es nämlich Softwaresysteme, an denen über tausend Entwickler etliche Jahre gearbeitet haben. Die Manager, welche die Verantwortung für derart komplexe Systeme tragen, sehen immer deutlicher die Gefahren, die darin bestehen, daß die vielen Beteiligten über diese Systeme nicht mehr angemessen kommunizieren können.

Der Autor, der das Kommunikationsproblem vor über 20 Jahren zum Schwerpunkt seiner Forschungsarbeit gemacht hat, hat in der Zwischenzeit selbstverständlich nicht nur das Problem analysiert, sondern er hat unter Mithilfe mehrerer Generationen junger Wissenschaftler eine Begriffswelt und die zugehörigen Darstellungsmittel entwickelt, die das Problem auf befriedigende Weise lösen. Die Angemessenheit der Begriffswelt und der Darstellungsmittel konnte inzwischen in mehreren großen Projekten mit der Industrie nachgewiesen werden.

Vermutlich war der Autor gegenüber den meisten Informatikkollegen bezüglich der Lösungssuche dadurch im Vorteil, daß er über die Systemtheorie und nicht über die Programmierung zur Informatik kam. Somit konnte er Software als Information über eine Klasse dynamischer informationeller Systeme ansehen, woraus er den Schluß zog, daß nicht die Software der primäre Kommunikationsgegenstand sei, sondern das dynamische informationelle System. Er brauchte also nur nach geeigneten Begriffen zur Modellierung dynamischer informationeller Systeme zu suchen. Während er noch vor Jahren wegen seines Ansatzes von Wissenschaftskollegen ausgelacht wurde, konnte er zu seiner Genugtuung vor kurzem von einem renommierten Informatiker [Jackson-98] die Aussage lesen: "A program is a description of a machine: a general-purpose computer accepts the description and, by executing it, becomes the machine described."

Da die weltweite Informatik erst jetzt zu dieser Einsicht gelangt ist, hatte sie selbstverständlich noch keine Zeit, daraus die angemessenen Konsequenzen zu ziehen. Und da in der Informatik die Neigung vorherrscht, jedes Rad, welches man in den traditionellen Ingenieurwissenschaften schon lange kennt, noch einmal zu erfinden, ist zu vermuten, daß man trotz der nun vorliegenden Einsicht nicht auf die Idee kommen wird, in den alten Schriften der Systemtheoretiker nachzuschauen, was man von dort übernehmen könne.

4. Betrachtung des Problems unter Bezug auf das Sprachwertedreieck

Das in [Frank-96] dargestellte Sprachwertedreieck geht von einer Klassifikation der Sprachen in die drei Klassen *Sachsprachen*, *Ritualsprachen* und *Marktsprachen* aus und stellt deren Merkmale - Genauigkeit, Verständlichkeit und Glaubwirksamkeit - zueinander in Beziehung.

Das hier betrachtete Problem kann nur durch eine angemessene Sachsprache gelöst werden. Bei der Entwicklung einer solchen Sachsprache mußte der Autor Semantik und Syntax gleichzeitig schaffen, d.h. es ging nicht nur darum, einen bekannten Semantikbereich mit einer angemessenen Syntax zu verbinden.

Abgesehen von der mathematischen Begriffswelt, wo die Informatiker eine bewährte Sachsprache verwenden, haben sich die Informatiker ein Sprachverhalten angewöhnt, bei dem sowohl die Genauigkeit als auch die Verständlichkeit fehlen. Ihre Sprache erfüllt sicher das Kriterium einer Ritualsprache, denn sie vermittelt zweifellos den typischen Informatikerstallgeruch, und außerdem hat sie viele Kennzeichen einer Marktsprache, denn sie wird intensiv dazu genutzt, Produkt- und Produzentenmängel zu verschleiern.

5. Literatur

[Frank-96] Helmar Frank: Bildungskybernetische Sachsprache im Sprachwertedreieck. Grundlagenstudien aus Kybernetik und Geisteswissenschaft - grkg, Band 37, 1996, Heft 4, S. 184-195.

[Hoare-86] C.A.R. Hoare: The Mathematics of Programming. Clarendon, Oxford, 1986.

[Jackson-98] Michael Jackson: Will there ever be Software Engineering? IEEE Software, Februar 1998, S. 36-39.

[Wendt-91] Siegfried Wendt: Nichtphysikalische Grundlagen der Informationstechnik. 2. Auflage. Springer-Verlag, Berlin; 1991.