Ein grundlegender Begriffsrahmen für das Wissensmanagement im Software-Engineering

Prof. Dr.-Ing. Siegfried Wendt

Geschäftsführender Direktor
des Hasso-Plattner-Instituts für Softwaresystemtechnik GmbH

Prof. Dr. Helmert-Straße 2-3
14482 Potsdam

eMail: siegfried.wendt@hpi.uni-potsdam.de

Ein grundlegender Begriffsrahmen für das Wissensmanagement im Software-Engineering

Abstract:
Many problems in software engineering have been solved in the past, but one major problem is still unsolved, and its relevance is growing fast: Because of the lack of adequate abstractions and models, the knowledge on complex software systems cannot be communicated efficiently from those who have it to those who need it. The paper presents the outline of a theory of description for the field of software engineering. Key concepts of this theory are
(1) the separation of knowledge about system structures from knowledge about mapping system structures into source structures,
(2) the classification of software systems as special cases of dynamic systems which can be modeled and represented on the basis of well defined categories of abstractions,
(3) the explanation of implementation and programming as relations between different system models.

1 Motivation und Schwerpunktsetzung

Software-Engineering als neue Ingenieurdisziplin hat mit den traditionellen Ingenieurdisziplinen alle Probleme gemeinsam, die sich aus der immer weiter wachsenden Komplexität der Systeme ergeben. Es ist sogar angebracht, die primäre Aufgabe aller Ingenieurdisziplinen darin zu sehen, die Komplexität von Systemen mit unüberschaubar vielen heterogenen Komponenten kommunikativ zu beherrschen. Dabei geht es nicht vorrangig um die Kommunikation im Dialog, sondern um die Weitergabe von Wissen im Monolog. Die Komplexität der Systeme bedingt einen so hohen Grad an Arbeitsteilung, dass der überwiegende Teil des Wissensflusses nicht mehr durch einen Dialog erreicht werden kann, sondern durch schriftliche und zeichnerische Monologe realisiert werden muss.

Dabei ergeben sich für das Wissensmanagement zwei Problemfelder, die in Bild 1 veranschaulicht sind. Im Problemfeld "Didaktik" geht es um die Frage, wie die Wissenslieferanten ihr mitzuteilendes Wissen darstellen sollen, damit die Empfänger das Wissen durch Interpretation der Darstellungen unmissverständlich und mit geringstmöglichem Aufwand gewinnen können.

Im Problemfeld "Logistik" geht es um die Frage, wie die Dialoge zur Wissensablieferung und zur Wissensbeschaffung gestaltet werden sollen, damit die Lieferanten und die Empfänger einen optimalen Wissensfluss erleben können, ohne dass sie durch einen zu hohen Dialogaufwand behindert werden.

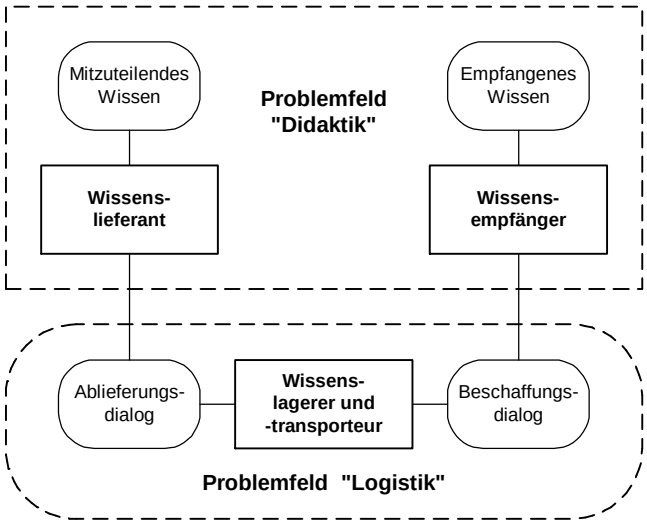

Bild 1 Die zwei Problemfelder des Wissensmanagements

Der Schwerpunkt der vorliegenden Arbeit liegt im Problemfeld "Didaktik", denn im Software-Engineering ist der state-of-the-art didaktisch noch keineswegs befriedigend. Zum Problemfeld "Logistik" werden im Abschnitt 4 einige Hinweise auf Lösungsansätze gegeben.

2 Planungswissen versus Abbildungswissen

Das Wissen, das durch die Arbeit von Ingenieuren geschaffen wird, sollte zweckmäßigerweise in Planungswissen und Abbildungswissen unterteilt werden. Bild 2 veranschaulicht den Unterschied. Oben im Bild geht es um Arbeitsergebnisse von Ingenieuren auf dem Gebiet der Elektronik. Links ist das Schaltbild eines einfachen Transistorverstärkers dargestellt. Symbole und Layout entsprechen dem professionellen Standard, so dass jeder Elektronikingenieur sofort sieht, welche Funktionalität hier planerisch realisiert wurde. Rechts ist die Platine dargestellt, die den konkreten Verstärker enthält. Auf dieser Platine findet man die Bauelemente und die Verbindungen dazwischen. Der Pfeil zwischen dem Schaltplan und der Platine symbolisiert den Sachverhalt, dass die Platine als

Ergebnis eines Abbildungsvorgangs gewonnen wurde, der vom Schaltbild ausging.

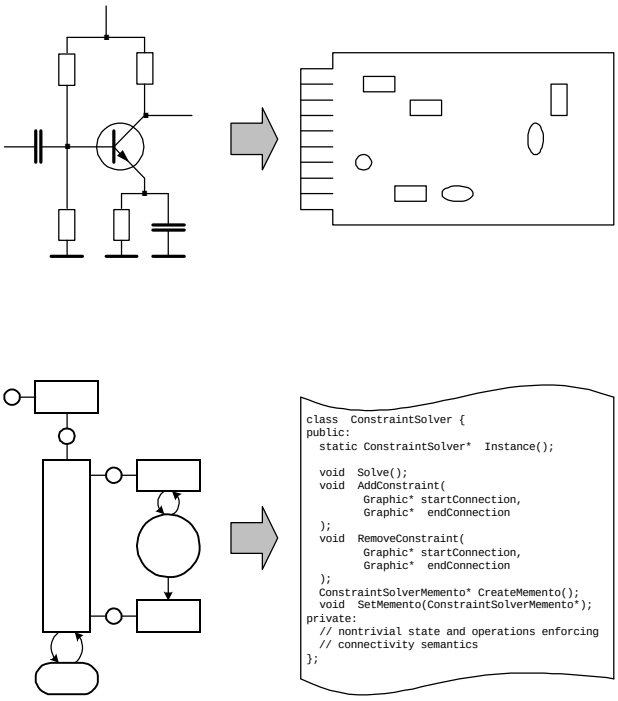

Bild 2 Programm als Abbildungsergebnis aus Planungswissen

In Analogie zur Elektronik ist im unteren Teil des Bildes 2 gezeigt, dass Software in Form programmiersprachlicher Texte als Abbildungsergebnis betrachtet werden sollte und nicht als ursprüngliche planerische Struktur. Der Leser sollte nicht versuchen, den Graphen links vom Pfeil zu interpretieren, denn dieser Graph wurde als reine graphische Struktur ohne Bedeutung gestaltet. Es sollte lediglich zum Ausdruck gebracht werden, dass angemessene "Ingenieurzeichnungen" gefunden werden müssen, wenn der Wissensfluss im Software-Engineering eine ähnliche Qualität bekommen soll, wie man sie aus den traditionellen Ingenieurdisziplinen gewohnt ist. Wie die entsprechenden Zeichnungen von Ingenieuren der Softwaresystemtechnik aussehen könnten, wird in dem folgenden Abschnitt 3 gezeigt.

3 Systemplanungswissen

Über ein konkretes System gibt es unendlich viel zu wissen, d.h. man kann unendlich viele Fragen bezüglich eines konkreten Systems stellen. Im jeweiligen Kontext interessiert aber stets nur ein endlicher Ausschnitt aus der Fülle des möglichen Wissens. Diesen Ausschnitt kann man als ein "Modell des Systems" bezeichnen. Wenn das konkrete System eine unendliche Fülle von Wissen umfasst, ein Systemmodell aber durch einen endlichen Wissensumfang gekennzeichnet ist, muss es konsequenterweise unendlich viele unterschiedliche Systemmodelle geben, d.h. es muss unendliche viele unterschiedliche Alternativen geben, durch Abstraktion vom konkreten System zu einem Modell zu gelangen (siehe Bild 3). Dass mit einem Modell nur endlich viel Wissen verbunden ist, ist gleichbedeutend mit dem Sachverhalt, dass ein Modell durch einen endlichen Beschreibungsaufwand vollständig dargestellt werden kann. Die Menge der Dokumente, die einem Modell in Bild 3 jeweils zugeordnet ist, ist also stets eine endliche Menge.

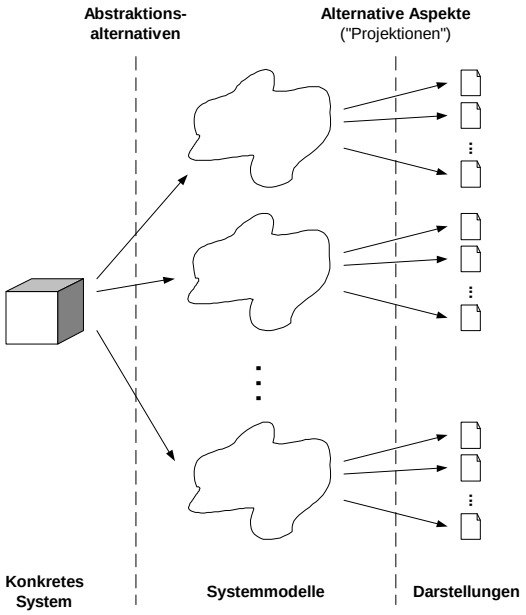

Bild 3 Der Zusammenhang zwischen System, Systemmodell und Darstellung

Die Systeme, für die man sich im Software-Engineering interessiert, sind Computersysteme, deren Verhalten durch Software bestimmt wird. Es handelt sich

also um eine bestimmte Art dynamischer Systeme. Bezüglich der Frage, wie man die interessierenden Systeme zweckmäßigerweise modellieren sollte, ist es hilfreich, die Betrachtung nicht gleich auf die dynamischen Systeme "der bestimmten Art" einzuschränken, sondern mit dem allgemeinen Begriff des dynamischen Systems zu beginnen und danach schrittweise bestimmte Einschränkungen einzuführen, bis man die eigentlich interessierende Art von Systemen erreicht hat. Dieses Vorgehen ist in Bild 4 veranschaulicht.

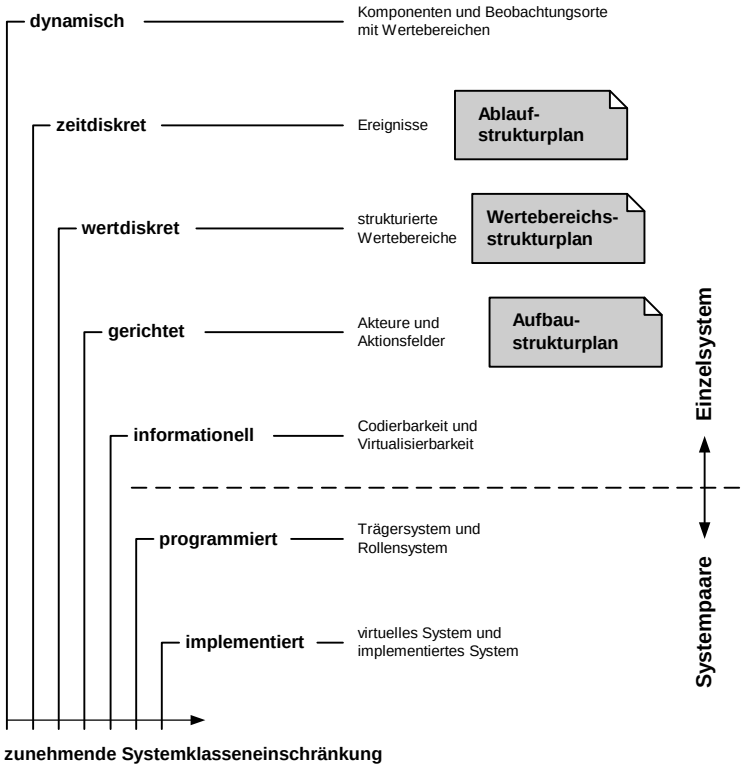

Bild 4 Systemklassen, Strukturpläne und Systembeziehungen

Zur vollständigen Beschreibung eines Modells vom Typ "Dynamisches System" genügen vier Kategorien von Strukturen: Aufbaustruktur, Wertebereichsstruktur, Ablaufstruktur und funktionale Beziehung. Zu dem Modell gehört genau eine Aufbaustruktur, die den Systemaufbau zum Zeitpunkt des Beginns der Systemexistenz beschreibt. Falls es sich um ein strukturinvarian-tes System handelt, gilt die Anfangsstruktur für die gesamte Systemlebensdauer, andernfalls enthält das System Komponenten, die Komponenten eliminieren oder erzeugen können, wodurch sich die Aufbaustruktur mit der Zeit ändern kann. Damit eine Komponente in eine Aufbaustruk-tur eingebunden werden kann, muss sie

Anschlüsse haben, an denen das Komponentenverhalten beobachtbar ist und die als Orte für eine mögliche Wechselwirkung mit anderen Komponenten in Frage kommen. Diese Wechselwirkung entsteht dadurch, dass durch die Verbindung zweier Anschlüsse zwei Beobachtungsorte zusammenfallen und die Gleichheit der Beobachtungsergebnisse erzwungen wird.

Der Begriff "Anschluss" wird hier in einem sehr abstrakten Sinne gebraucht und ist nicht auf elektrische oder mechanische Systeme beschränkt. Jeder Beobachtungsort in einem Systemmodell entspricht einem Anschluss oder einer Verbindung von Anschlüssen. Mit jedem Beobachtungsort ist ein Wertebereich verbunden; es handelt sich um das Repertoire der an diesem Ort möglichen Beobachtungsergebnisse. Ein zeitlicher Verlauf von Beobachtungsergebnissen wird als Ablauf bezeichnet. Innerhalb eines Ablaufs oder zwischen mehreren Abläufen kann es funktionale Beziehungen geben.

Als erste Einschränkung des dynamischen Systems in Richtung auf die eigentlich interessierende spezielle Art von Systemen wird die Zeitdiskretheit hinzugenommen. Es werden nun keine kontinuierlichen Verläufe mehr betrachtet, sondern Folgen von Ereignissen. Dadurch kann man die Gesetzmäßigkeit von Verläufen graphisch erfassen. Als Darstellungsmittel bieten sich Petrinetze an, die als generative Schemata für Partialordnungen mathematisch ausgereift sind und sich im Hinblick auf die Gestaltwahrnehmung graphisch optimal gestalten lassen.

Als nächste Einschränkung wird die Wertdiskretheit hinzugenommen. Nun stammen die Beobachtungsergebnisse aus einem diskreten Repertoire, und ein Beobachtungsergebnis kann eine eigene Struktur haben, d.h. es kann aus verschiedenartigen Elementen bestehen, die zueinander in bestimmten Beziehungen stehen. Als Darstellungsform für Wertebereiche, in denen die Elemente selbst Strukturen sind, eignen sich sehr gut die Entity-Relationship-Diagramme.

Als weitere Einschränkung wird die Gerichtetheit hinzugenommen. Nun können die Anschlüsse der Komponenten in Ein- und Ausgänge klassifiziert werden, so dass man nun zwischen Akteuren und Aktionsfeldern unterscheiden kann. Die Akteure sind zuständig dafür, dass auf den Aktionsfeldern etwas geschieht. Die Aktionsfelder sind die Orte für die Beobachtung des Geschehens. Die Aufbaustruktur wird als bipartiter Graph gezeichnet, dessen zwei Knotenmengen die Akteure und die Aktionsfelder sind. In der Anschauung ist ein Aktionsfeld entweder ein Speicher oder ein Kanal.

Bild 5 zeigt die drei eingeführten Plantypen anhand eines sehr einfachen Beispiels. Alle drei Plantypen sind bipartite Graphen. Die Wahl der Knotenform wurde so entschieden, dass mit den Rechtecken eher das Aktive, die agierende Blackbox, und mit den runden Knoten eher das Passive, der Behälter, assoziiert werden kann. Die einheitliche Bipartitheit aller drei Plantypen vereinfacht die Vorgehensweise im Falle von Vergröberungen oder Verfeinerungen der Strukturen.

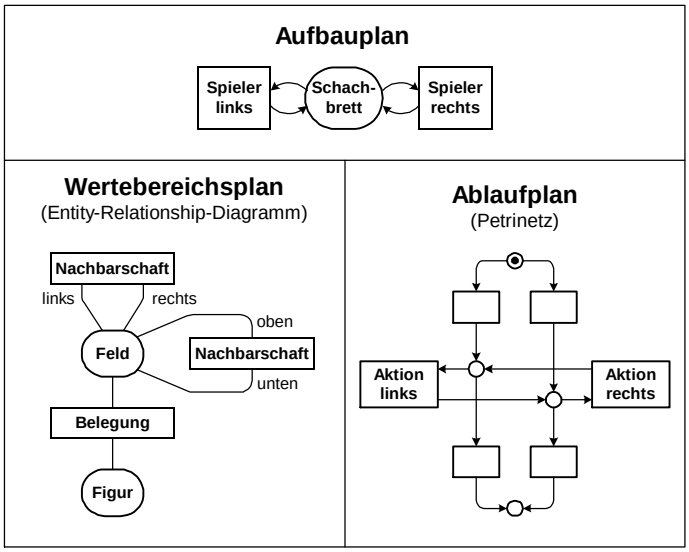

Bild 5 Beispiel für die drei Strukturplantypen eines Systemmodells

Der Leser sollte bemerkt haben, dass die drei Plantypen nicht zur Voraussetzung haben, dass das System informationell ist (siehe Bild 4). Bei informationellen Systemen sind die Beobachtungsergebnisse interpretierbare Formen, wobei nicht die Formen selbst, sondern die ihnen zugeordneten Bedeutungen von primärem Interesse sind. Durch diese Einschränkung kommt man zur Codierbarkeit der Wertebereiche und zur Virtualisierbarkeit der Systeme. Dies hat zur Konse-quenz, dass man nun die Betrachtung auf Paare von Systemmodellen erweitern muss.

Die Systemrealisierung durch Programmierung zwingt zur Unterscheidung zwischen dem Rollensystem und dem Trägersystem. Das Trägersystem ist das programmierbare System, also der Schauspieler, und das Rollensystem ist das System, zu dem das Trägersystem wird, wenn es sich programmgemäß verhält, also die Person, die vom Schauspieler dargestellt wird.

Implementierung und Programmierung werden hier nicht gleichgesetzt. Programmierung liegt immer dann vor, wenn man zwischen einem Rollensystem und einem Trägersystem unter-scheiden kann. Implementierung wird hier als Übergang von einem spezifizierten System zu einem "System größerer Machbarkeit" verstanden. Hätte man ein Trägersystem, welches die Spezifikation als Programm akzeptieren kann, läge Programmierung vor, aber nicht Implementierung. Wenn dagegen das Trägersystem die Spezifikation nicht als Programm akzeptieren kann, sondern verlangt, dass atomare Elemente aus der

Spezifikation als Strukturen aus einfacheren Elementen codiert werden, liegen sowohl Implementierung als auch Program-mierung vor.

Dem Betrachter von Bild 2 wird suggeriert, dass die Abbildung von den Plänen, welche die Systemmodelle darstellen, zum Programmtext ähnlich leicht zu beschreiben sei wie die Abbildung vom Schaltplan zur Platine. Im Normalfall kann diese Abbildung aber nur nachvollzogen werden, wenn das diesbezügliche Wissen durch sogenannte Brückenpläne vermittelt wird. Im Rahmen der vorliegenden Arbeit ist allerdings kein Raum für eine detaillierte Einführung solcher Pläne. Es können lediglich einige Hinweise gegeben werden, die dem Leser möglicherweise verständlich machen, welche Art von Plänen hier gemeint sind. Es handelt sich bei den Brückenplänen i.a. um Graphen, bei denen zusammenhängende Programmtextstücke in Form von Knoten vorkommen. Man denke an graphische Darstellungen der Import/Export-Beziehungen zwischen Modulen oder an Vererbungsbäume bei der objekt-orientierten Programmierung.

4 Wissenslogistik

Die Komplexität der hier interessierenden Softwaresysteme kann leicht durch die Angabe von Umfangs- und Aufwandszahlen veranschaulicht werden: Die Systeme erreichen inzwischen Umfänge von über 50 Millionen Programmzeilen, und die Entwicklung erfordert häufig einen Aufwand von über 10 Tausend Entwicklerjahren. Damit 4.000 Entwickler über drei Jahre hinweg koordiniert an einem Großsystem arbeiten können, muss die Wissenslogistik einen Stand haben, wie er zur Zeit noch nicht allgemein üblich ist.

Bild 6 zeigt, dass man in der dem Wissensmanagement zugrunde liegenden Dokumentensammlung sowohl auf der Planseite als auch auf der Programmtextseite jeweils drei unterschiedliche Arten von Wissenselementen unterscheiden kann. Die Unterscheidung ergibt sich aus der Lage der Wissenselemente im Verständnispfad. Es gibt Programmtextstücke, die der Fachmann ohne Zuhilfenahme zusätzlicher Dokumente verstehen kann - man denke an eine Prozedur zum Sortieren oder an die Implementierung des abstrakten Datentyps *Stack*. Auf der anderen Seite gibt es Systempläne, die auf einem so hohen Abstraktionsniveau liegen, dass sie noch gar keine nachvollziehbare unmittelbare Beziehung zum Programmtext haben. Das Verständnis dieser Pläne ist aber Voraussetzung für das Verständnis tiefer liegender Pläne, die Systemmodelle beschreiben, die durch Programmierungs- und Implementierungsbeziehungen aus höher liegenden Systemmodellen gewonnen wurden. Die Abbildung zwischen Knoten auf Systemmodellplänen und Programmtextstücken ist teilweise unmittelbar verständlich, teilweise sind aber Brückenpläne erforderlich. Manche Programmtextstücke kann man erst verstehen, nachdem man zuvor andere Programmtextstücke studiert und verstanden hat.

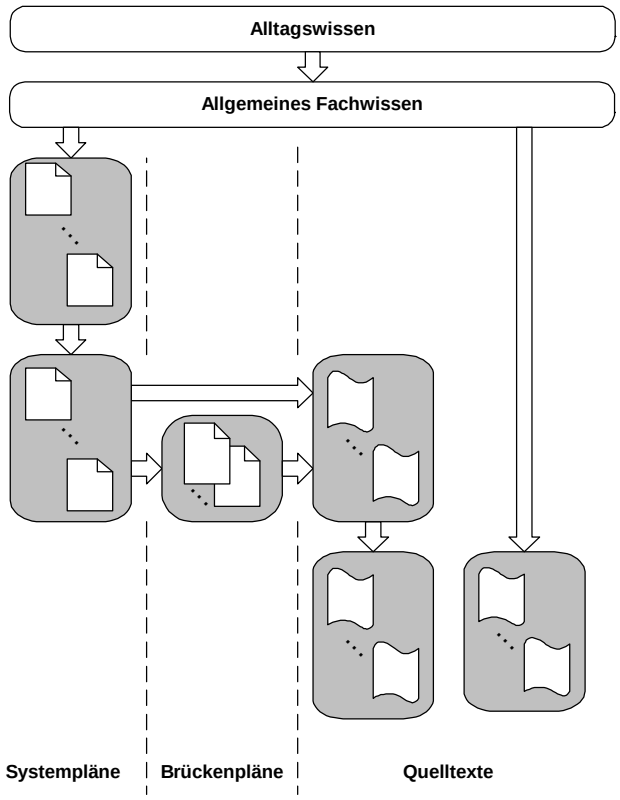

Bild 6 Dokumentennetz zur Wissenserschließung

5 Ausblick

Die hier betrachteten Probleme gehören nicht in den Bereich Programmierung, auch nicht in den Bereich "programming in the large". Es handelt sich um Probleme, die man nur lösen kann, wenn man einen eigenständigen begrifflichen Rahmen zugrunde legt, der speziell für diesen Problembereich geschaffen wurde. Ein angemessener Begriffsrahmen ist aber selbstverständlich nicht hinreichend, sondern nur notwendig für die Beherrschung dieses Problemfelds. Was hinzukommen muss ist eine effiziente Unterstützung durch zugeschnittene Tools. Die oft geäußerte Behauptung, die UML-bezogene Begriffswelt und die diesbezüglichen Tools stellten eine Lösung der vorgestellten Probleme dar, hält einer Überprüfung in der Praxis nicht stand. Dies ist eine Konsequenz des

Sachverhalts, dass dem Begriffsrahmen, auf dem UML ruht, die nötige Präzision und Orthogonalität fehlt.

Der hier vorgestellte Begriffsrahmen wird sich nicht verbreiten können, solange eine daran angepasste Tool-Suite fehlt. In der Umgebung des Autors wird an der Entwicklung einer solchen Tool-Suite gearbeitet.

Literatur

Bungert, Andreas: Beschreibung programmierter Systeme mittels Hierarchien intuitiv verständlicher Modelle. Shaker Verlag, Aachen, 1998.

Jackson, Michael: Defining a Discipline of Description. IEEE Software, September/October 1998, p. 14-17

Kleis, Wolfram: Konzepte zur verständlichen Beschreibung objektorientierter Frameworks. Shaker Verlag, Aachen, 1999.

Tabeling, Peter: Der Modellhierarchieansatz zur Beschreibung nebenläufiger, verteilter und transaktionsverarbeitender Systeme. Shaker Verlag, Aachen, 2000.

Wendt, Siegfried: Der Kommunikationsansatz in der Software-Technik. Siemens data report, 17. Jg. (1982), H. 4, S. 4-7

Wendt, Siegfried: Erfahrungen mit graphischen Strukturplänen in Großprojekten der Softwareentwicklung. Tagungsband: 36. Internationales Wissenschaftliches Kolloquium Illmenau, S. 769-774, 1991.