Thesen zu Gegenstand, Zeichen, Objekt - und deren Bedeutung für die objektorientierte Analyse

Wolfgang Hesse[1] , Hubert v. Braun[2]

[1] FB Mathematik und Informatik, Univ. Marburg, hesse@informatik.uni-marburg.de
[2]ASG München, hubert.vbraun@uumail.de

Zusammenfassung: In diesem Beitrag beleuchten wir einige Begriffe, die für die in der Informatik seit ca. 1990 propagierte und praktizierte "objektorientierte Analyse und Modellierung" (OOAM) wichtig sind. Dazu gehören u.a. *Gegenstand, Zeichen, Bezeichner* und *Bezeichnetes, Konzept, Bezug* (referent), und *Objekt*. Während "Gegenstand" als unbestimmter "Universal-Designator" erkannt und eingestuft wird, kommt "Objekt" (wenn es nicht synonym zu Gegenstand verwendet wird) - je nach Kontext - eine spezifischere Bedeutung zu. Das gilt vor allem in der Informatik für den Kontext der "Objektorientierung". Aber auch hier findet man sehr unterschiedliche Begriffsverständnisse nebeneinander. Dem Anspruch der OOAM, (nicht-symbolische) "Realwelt" und (symbolische) "Software-Welt" miteinander zu verbinden , kann man gerecht werden, wenn man drei Aspekte von "Objekt" gemeinsam betrachtet: den universell-ontologischen (U-Aspekt), den konzeptionellen (C-Aspekt) und den Repräsentations- (R-) Aspekt.

Einleitung: Warum wir uns mit dem Objektbegriff beschäftigen

"Objektorientierung" ist seit mehr als 20 Jahren eines der meistgebrauchten Schlagworte in der Softwaretechnik. M. Broy und J. Siedersleben haben kürzlich den damit verbundenen Denkansatz einer kritischen Prüfung unterzogen und sind zu einem eher ernüchternden Urteil gekommen [B-S 02]. Dort wird allerdings fast ausschließlich der Bereich der *objektorientierten Programmierung (OOP)* angesprochen, während sich die im Titel angekündigte Ausdehnung auf die "Software-Entwicklung" auf einige eher allgemein gehaltene Aussagen in einem abschließenden Abschnitt beschränkt.

Gerade auf dem Gebiet der *objektorientierten Analyse (OOA)* - die der Programmierung vorausgeht und die u.a. den Bezug zwischen den Aufgabenstellungen aus der "Realwelt" und ihren Software-Lösungen herstellen soll, halten wir eine solche kritische Prüfung für mindestens ebenso angebracht - wenn nicht noch dringlicher:

- Was heißt es, wenn "Software-Objekte Realwelt-Objekten nachgebildet" (*"are modeled after ..."* [C-W 98]) werden sollen, welcher Art sind diese Objekte, wie findet man sie und wie hängen sie zusammen?

- Was bedeutet z.B. eine Anweisung "**`delete Brown`**"? Soll hier schlicht ein Datensatz (etwa für den Kunden *Brown* in einer Kundendatei) gelöscht werden oder geht es um eine Aufforderung zur Liquidation des körperlichen "Objekts" *Brown* oder gar zur gänzlichen (körperlichen und geistigen) Tilgung im Sinne einer Orwell'schen "Vaporisierung"?

- Wie verträgt sich der (anscheinend?) auf Dinge zentrierte, im Grunde statische "objekt-orientierte" Ansatz mit den Anforderungen an eine Software-Entwicklung, die konkrete Probleme der Lebenswelt mit aller ihrer Dynamik und Handlungsvielfalt möglichst "ganzheitlich" lösen soll?

Diese und ähnliche Fragen haben wir in zwei Arbeitskreisen (zur OOA und zur Begriffsbildung in der Softwaretechnik) in den letzten Jahren ausgiebig diskutiert und Zwischenergebnisse daraus regelmäßig veröffentlicht (siehe z.B. [H-M 98], [BHA+ 00], [H-B 01]). Der vorliegende Aufsatz soll diese Diskussion fortsetzen.

"Gegenstand" - ein Universal-Designator

(1) Ein *Gegenstand* ist "etwas uns Entgegenstehendes", d.h. im Prinzip jeder mögliche Fokus unserer Wahrnehmung, Betrachtung oder sonstiger Handlungen einschließlich der Fiktion. Die Frage, ob es "Objekte" jenseits unserer Wahrnehmung[1] gibt, ist müßig - gerade durch unsere erstmalige Wahrnehmung (und sei diese auch nur fiktiv) und Bezeichnung werden sie (automatisch) zum Gegenstand (unserer Betrachtung). In diesem sehr allgemeinen Sinne ist Gegenstand (oder "Objekt im weiten Sinne") zunächst nichts weiter als der Fokus einer "deiktischen" (Sprach- oder sonstigen Zeige-) Handlung, "das auf, das hingewiesen wird" oder kürzer: "das Bezeichnete". B. Russell sagt dazu im Vorwort der engl. Ausgabe des *Tractatus* [Wit 61]:

> "... This amounts to saying that 'object' is a pseudo-concept. To say 'x is an object' is to say nothing. It follows from this that we cannot make such statements as 'there are more than three objects in the world', or 'there are an infinite number of objects in the world'. Objects can only be mentioned in connexion with some definite property. We can say 'there are more than three objects which are human', or 'there are more than three objects which are red', for in these statements the word 'object' can be replaced by a variable in the language of logic, the variable being one which satisfies in the first case the function 'x is human'; in the second the function 'x is red'. But when we attempt to say 'there are more than three objects', this substitution of the variable for the word 'object' becomes impossible, and the proposition is therefore seen to be meaningless. ..."

D.h. hier ist "Gegenstand" weder ein "Konzept" noch ein "Prädikator" - nicht einmal ein "Leer-Prädikator" (Lorenzen), sondern eine Art *Universal-Designator* oder sprachlicher Zeige-Operator, der aus sprachergonomischen Gründen verwendet wird: Statt zu sagen: *"Dies ist rot"* können wir (mit etwas mehr Nachdruck) sagen: "Dieser Gegenstand ist rot".

Objekt = aufgefasster und gedeuteter Gegenstand

(2) 'Gegenstand' ist Teil der deiktischen Sprachgeste, verweist allerdings auf etwas, was jenseits der Sprache liegen kann. Damit dient es (wie andere hinweisende Partikel und Bezeichner) dazu, Nicht-sprachliches sprachlich verfügbar zu machen. So wie das Wort "Gegenstand" ein "General-Bezeichner" für jedwedes, nicht näher spezifiziertes Etwas ist, können wir durch die Wahl spezieller Bezeichner die Menge der bezeichneten Dinge einschränken - bis hin zur Nullmenge. F. de Saussure hat diese Beziehung wie folgt dargestellt:

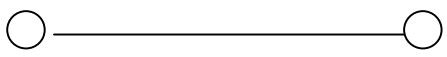

Abb. 1: “Bezeichnungs”-Beziehung nach F. de Saussure

(3) Um zu einem "Konzept" zu werden, muss etwas Wahrgenommenes bezeichnet und aufgefasst *("konzipiert")* werden. Dazu gehört u.a. seine Abgrenzung und "Erschließung", d.h. eine Beschreibung seiner Eigenschaften und Merkmale. Eine solche *Auf*-Fassung (*"conception,"* vgl. FRISCO [FHL+ 98]) führt Unbekanntes auf Bekanntes zurück und ist damit notwendigerweise reduktionistisch: Sie "erschließt" Neues mit dem (begrenzten) Vorrat von Wissen über Bekanntes und setzt es dazu in Beziehung [Gha 03]. Sie ist immer endlich und kann die potentiell unendlich

[1] *Wahrnehmung* (engl.: *perception*) wird hier als bloße unreflektierte Unterscheidung eines vom anderen und als Vorstufe zur *conception* ("als Etwas erkennen") verstanden (vgl. [FHL+ 98]).

vielen Eigenschaften des Gegenstands nur eingeschränkt erfassen. Sie ist auch nicht eindeutig, d.h. zum gleichen Gegenstand können beliebig viele unterschiedliche Konzeptionen existieren.

Wenn wir den Unterschied auch sprachlich deutlich machen wollen, könnten wir die Koexistenz der beiden Worte "Gegenstand" und ""Objekt" in der deutschen Sprache nutzen: *Objekt* ist dann ein *aufgefasster Gegenstand*, d.h. einer, den man nicht nur bezeichnen, sondern auch durch Eigenschaften und Merkmale beschreiben kann.

Triadische Zeichenbegriffe in der Semiotik und Informatik

(4) Hier kann man den von Pierce und anderen Autoren vorgeschlagenen triadischen Zeichenbegriff nutzen (vgl. [FHL+ 98]) : Peirce unterscheidet *sign, object* und *interpretant*: Ein Zeichen (*sign*) kann ein Symbol, eine Zeigehandlung oder ein als solches gedeutetes Phänomen (wie z.B. ein Rauch"zeichen") sein. Es weist auf etwas "Bezeichnetes" (*"object"* im Sinne vom o.g. "Gegenstand") hin. Dazu gehört immer ein *interpretant*, d.h. eine Deutung des Bezeichneten im Kontext des Zeichen-Senders bzw. Zeichen-Empfängers. Der Peirce'sche Interpretant sollte aber nicht mit dem Interpreter (Zeichen-Sender bzw. -Empfänger) verwechselt werden.

Eine Variante der Peirce'schen Triade liefert der FRISCO-Bericht (vgl. Abb. 2). Dort wird anstelle des vielfältig deutbaren *"Object"* das Wort *"referent"* (= das, auf das verwiesen wird) verwendet - mit der an Peirce erinnernden Zusatz-Erklärung *sign object*. Dabei handelt es sich um einen (nicht weiter bestimmten) Bereich (engl.: *domain*). Das von Peirce gebrauchte Wort *sign* könnte insofern missverstanden werden, als man damit womöglich die gesamte Zeichen-Triade ansprechen möchte. Im FRISCO- Bericht hat man deshalb das präzisere *sign token* oder *representation* vorgezogen. Schließlich hat man das Peirce'sche Kunstwort *Interpretant* als *conception* (oder *sign concept* - vgl. (4)) gedeutet.

Konzeptionen und Repräsentationen sind nicht zu trennen von den Individuen bzw. Gemeinschaften, die diese nutzen, sei es, um mit Hilfe von Zeichenprozessen etwas mitzuteilen oder empfangene Mitteilungen zu interpretieren. Im FRISCO-Report hat man dem durch die Erweiterung zum "Semiotischen Tetraeder" Rechnung getragen, bei dem der (menschliche oder - in bestimmten Fällen automatisierte -) *"Aktor"* in der Mitte steht (vgl. Abb. 4).

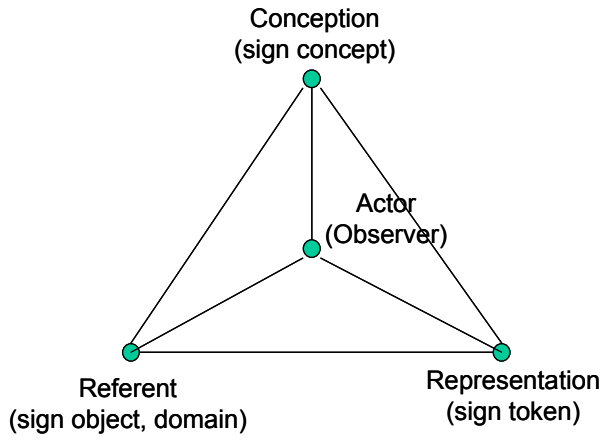

Abb. 2: Semiotischer Tetraeder nach FRISCO

(5) In der Informatik haben wir es mit verschiedenen Arten und Formen von Objekten zu tun. "Objekt" wird hier (fast) immer spezifischer aufgefasst als ein bloßes Synonym zu "Gegenstand" im Sinne von (1) und (2). Elementare Objekte der (herkömmlichen) Programmierung sind die sog. *Konstanten* und *Variablen*: Sie tragen einen *Bezeichner* (*identifier*) bzw. eine *Adresse* und haben einen Bezug, den *Variablen-* oder *Konstantenwert.* Werte sind selbst wieder Bezeichner - entweder für andere Variablen oder Konstanten (das kann man z.B. nutzen, um damit Referenz-Ketten aufzubauen) oder aber sie sind für Programmbenutzer (menschliche *Aktoren*) bestimmte Zeichenreihen (z.B. Texte oder Ziffernfolgen), die diese (bei gelingender Interpretation) womöglich mit Gegenständen ihrer Lebens- oder Gedankenwelt verknüpfen können.

Die Rolle des Interpretanten nimmt in diesem Falle die Interpretationsvorschrift ein, die es z.B. einem Compiler erlaubt, den Variablen- oder Konstantenwert bestimmungsgemäß - etwa nach seiner Zugehörigkeit zu einem bestimmten *Typ* - abzuspeichern bzw. zu interpretieren

Herkömmliche Programmiersprachen haben einen Satz festgelegter Typen - damit ist der Handlungsraum für die "Objekte" (d.h. Variablen und Konstanten) durch einen Satz vorbestimmter, standardisierter Operationen (wie Zuweisungen, Vergleiche, Arithmetik etc.) festgelegt. Bei der *objektorientierten Programmierung (OOP)* wird dieser Handlungsraum (innerhalb bestimmter Grenzen) *programmierbar*: An die Stelle des vordefinierten Typs tritt die vom Programmierer individuell konzipierte *Klasse*, die alle zugehörigen ("gleichartigen") Objekte gemeinsam charakterisiert. Diese enthält nicht nur Aussagen darüber, welche Werte ein Objekt hat oder annehmen kann sondern auch, welche Operationen man darauf anwenden kann, d.h. welche Handlungen man damit vornehmen kann.

Technisch gesehen ist jedes solches "Objekt" ein Speicherbereich - mit einer (Speicher-) Adresse und einer Reihe von Speicherzellen, die die Ausprägungen der o.g. Charakteristika in geeigneter Form als "Wert" repräsentieren. Damit wird die gesamte Triade zum Teil der symbolischen, Computer-verarbeitbaren Welt:

> "An object represents a component of the Smalltalk-80 software system. .. An object consists of some private memory and a set of operations." [G-R 83]

Zeichentriade bei der Objektorientierten Analyse

(6) Eine anscheinend neue Situation schafft in dieser Hinsicht die seit etwa 1990 propagierte *objektorientierte Analyse (OOA).* Sie erhebt den Anspruch, über den symbolischen Bereich hinauszugehen:

> "You can look around you now and see many examples of real-world objects: your dog, your desk, your television set, your bicycle. These real-world objects ... all have state, and they all have behaviour. ... Software objects are modeled after real-world objects in that they, too, have state and behaviour. ..." ([C-W 98], p. 40)

Damit wird eine Repräsentations-Beziehung (*"are modeled after ..."*) zwischen sog. *real-world objects* und *software objects* unterstellt. Natürlich wurde eine solche Beziehung auch schon vor dem Aufkommen der OOA oft unterstellt, doch sind die damit auftretenden terminologischen Probleme mit der OOA erst richtig deutlich geworden. Können wir mit dem oben ausgeführten triadischen Ansatz auch Realwelt-Objekte im Sinne der OOA angemessen erfassen?

(7) Ein "Bezeichnetes" (sign object) muss nicht notwendigerweise der symbolischen Welt angehören - es kann auch "draußen" - jenseits der Grenzen von Computer und Sprache angesiedelt sein (vgl. (3)). Nur: was ist genau gemeint, wenn wir eine sprachliche Repräsentation (ein *sign token*) benutzen, um damit einen *Gegenstand der Realwelt* zu bezeichnen? Prinzipiell ist der Gebrauch eines *sign token* mit den gleichen Unsicherheiten behaftet wie das Zeigen mit dem Zeigefinger, mit einem Augenaufschlag oder dem Gebrauch von Wendungen wie *"Dies ist .."* oder *"Dieser Gegenstand ist .."*. Allesamt sind dies Zeigehandlungen, die der Fokussierung auf ein "Gemeintes", aber nicht notwendigerweise dessen eindeutiger Bestimmung und schon gar nicht dessen vollständiger Erfassung dienen können.

Die "vollständige Erfassung" eines Realwelt-Gegenstands (*"real world object"*) kommt allein schon deshalb nicht in Frage, weil diese immer ein endloses Unterfangen wäre: Über jeden Gegenstand - und sei er noch so banal - lassen sich prinzipiell unendlich viele zutreffende Aussagen machen. Hier kommt wieder die 3. (obere) Ecke unserer Triade ins Spiel: Wir können einen solchen Gegenstand zwar nie vollständig erfassen, aber ihn beliebig genau charakterisieren - und zwar durch endlich viele Zuschreibungen (oder "Prädikatoren") - und das ist genau die Rolle einer *Auffassung (conception).* Also: zu einem unendlich charakterisierbaren Gegenstand können beliebig viele endliche Charakterisierungen (*conceptions*) existieren, die jeweils eine *Sicht* auf diesen Gegenstand darstellen. Jede dieser Charakterisierungen kann in einer bestimmten Sprache *repräsentiert* werden. Diese Beziehung muss nicht ein-eindeutig sein, d.h. für die gleiche Charakterisierung können mehrere, verschiedene Repräsentationen benutzt werden und umgekehrt kann eine Repräsentation unterschiedlich gedeutet werden, d.h. für verschiedene Auffassungen stehen.

Die drei Aspekte von "Objekt"

(8) Wenn wir also einen erweiterten, triadischen Objektbegriff im Sinne von (6) bzw. (7) zugrunde legen und dabei jede Art von Bezügen (*referents*) einschließlich solcher auf Realwelt-Objekte zulassen wollen, so können wir dies durch die Unterscheidung dreier *Aspekte* tun, die jedem Objekt zukommen (vgl. Abb. 6): Der ***U-Aspekt*** ist universal, bezieht sich auf einen Gegenstand als Teil eines *Universe of Discourse* und umfasst alles, was man prinzipiell über diesen Gegenstand wissen kann. Damit ist dieser Aspekt potentiell unendlich und durch sprachliche Mittel (einschließlich Computer) womöglich nicht vollständig erfassbar. Mit dem U-Aspekt von Realwelt-Objekten verhält es sich ähnlich wie mit den irrationalen Zahlen (z.B. der Zahl *pi* oder dem nicht-rationalen Grenzwert einer mathematischen Folge oder Funktion): Man kann sie beliebig genau repräsentieren, aber niemals vollständig erfassen..

Will man einen Gegenstand sprachlich "erfassen", so muss man ihn (durch sprachliche Aussagen) charakterisieren ("auffassen" im Sinne von FRISCO, vgl. (4)). Solche Charakterisierungen sind immer endlich, aber oft nicht eindeutig, sondern "kreisen" in gewissem Sinne um den Gegenstand herum, ohne ihn jemals vollständig erfassen zu können. Diese sprachliche Charakterisierung wollen wir als ***C-Aspekt*** eines Objekts bezeichnen.

Für die gefundenen Charakterisierungen lassen sich nun sprachliche Repräsentationen finden - z.B. in Form von Symbolen, Texten, Zahlen, Graphiken, Tabellen - kurzum in jeder Form von Daten. Wir wollen diesen (rein syntaktischen) Aspekt unserer Objekte ***R-Aspekt*** nennen.

Als Beispiel aus einem Informationssystem nehmen wir den Kunden *Brown* (vgl. Abb. 3): Sein U-Aspekt umfasst alles, was wir aus der unendlichen Mannigfaltigkeit seiner physischen und geistigen Existenz über ihn zusammentragen könnten (aber niemals vollständig oder erschöpfend tun können!).

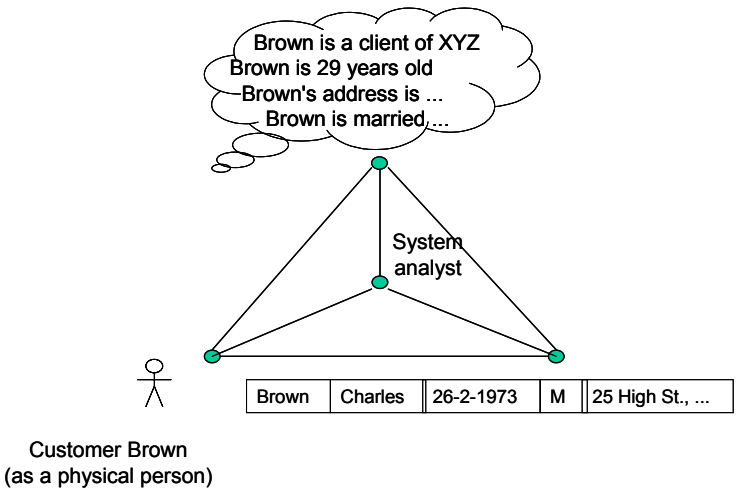

Abb. 3: Beispiel: Die 3 Aspekte eines Objekts in einem Informationssystem

Ein möglicher C-Aspekt kann aus der Menge von Aussagen bestehen, die in Abb. 3 an der oberen Ecke des Dreiecks stehen. Ein möglicher R-Aspekt dazu könnte eine Zeile in einer Datenbank-Tabelle (aber ebenso eine textuelle Personenbeschreibung, ein Bild oder eine sonst geartete Repräsentation) sein.

Auf Repräsentationen kann man - z.B. als Werte in einem OO-Programmsystem - selbst wieder Bezug nehmen. Damit lassen sich Ketten von Triaden bilden wie in Abb. 4 angedeutet. Hier beziehen wir uns mit dem UML-Konstrukt Brown: Customer auf den Datenbanksatz aus Abb. 3, die Interpretationsvorschrift liefert das entsprechende DB-Schema.

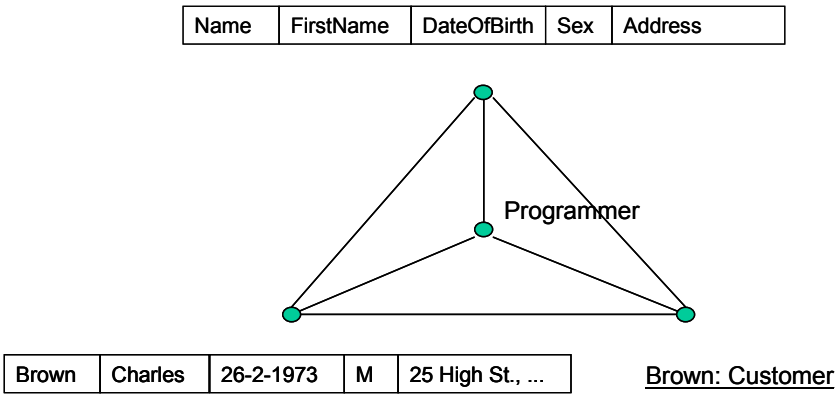

Abb. 4: Beispiel eines Software-Objekts

(9) Zu einer weiteren möglichen Deutung des U-Aspekts gelangen wir, wenn wir die Idee des "Einkreisens" eines Gegenstands durch fortlaufende Charakterisierungen aufnehmen und uns einen "Grenzübergang" analog zur Infinitesimalrechnung in der Mathematik vorstellen. Als

"Fixpunkt" einer solchen Folge von Charakterisierungen könnte man dann vom "Ding an sich" sprechen. Eine solche Redeweise würde zum einen den (aus guten Gründen) in der Informatik sehr verbreiteten "naiven Realisten" eine Brücke bauen und zum anderen nicht mit der guten konstruktiven Tradition brechen: Denn dieses "Ding an sich" ist eben nicht "da" und muss nur "gepflückt" werden ([Mey 88]), sondern entsteht im Zuge von fortlaufenden (im Grunde nie endenden) konstruktiven Charakterisierungs-Handlungen.

Will man dieses "Ding" (aller prinzipiellen Unmöglichkeit zum Trotz) darstellen, so bedient man sich eines "Surrogats" (wie z.B. des Strichmännchens in Abb. 3 oder der internen Objekt-Id bei objektorientierten Datenbanken). Damit bildet man (gedanklich) eine weitere Zeichentriade, die nun das Surrogat (als Repräsentation) mit dem eigentlichen Ding über die Konzeption des Zeige- und Ersetzungsvorgangs verbindet (vgl. dazu auch [Sow 02]).

Folgerungen für die Software-Entwicklung und ihre "frühen Phasen"

(10) Welche Folgerungen können wir daraus für die Informatik und im besonderen für die sog. "frühen Phasen" der Software-Entwicklung ziehen? Wir fassen zusammen:

- Der Objektbegriff der Informatik (und speziell deren sog. "OO-Techniken") geht über den eines Universal-Designators oder Russell'schen Pseudo-Konzepts hinaus. Objekte sind auch keine natur- oder metaphysisch gegebenen Entitäten, sondern soziale Konstrukte der Beteiligten, die deren Vorstellungen und Intentionen widerspiegeln. Sie sind nicht vorgefertigt, einfach "da" oder "zum Pflücken", sondern müssen von System-Eignern, -Benutzern und -Entwicklern gemeinsam im Diskurs entwickelt - eben *"konstruiert"* werden (vgl. [BHA 00])..

- *OO-Objekte* (wir nutzen diese im Grunde pleonastische Bezeichnung zur Unterscheidung von anderen Objekt-Auffassungen) lassen sich durch die 3 Aspekte *Repräsentation* (Bezeichner, Adresse) , *Konzeption* (Interpretation, intensionale Beschreibung) und *Bezug* (Wert, Extension) fassen. Dies gilt in gleicher Weise für Software-Objekte der OOP und für die sog. "Realwelt-Objekte" der OOA:

- Aus dieser Auffassung von Objekten ergeben sich klare Konsequenzen für den Software-Entwicklungsprozess: Am Anfang der Software-Entwicklung stehen keine fertigen Objekte, sondern Wahrnehmungen, Konzeptionen und Beschreibungen von Sachverhalten und Handlungen, die es zu verstehen, zu bearbeiten und im gemeinsamen Diskurs zur Deckung zu bringen gilt. Objekte können daraus als Brennpunkte des gemeinsamen Verständnisses herausgearbeitet werden - im ständigen Wechselspiel zwischen den Aspekten Bezugsbildung - Konzeption und Interpretation - Repräsentation. Moderne OO-Techniken unterstützen - zumindest teilweise - diesen Prozess: So geht es bei der für die OOA empfohlenen Anwendungsfall-Analyse (*use case analysis*, vgl. [Jac 93]) zunächst darum, Sachverhalte, Vorgänge, Handlungen im Bezugsbereich zu untersuchen, diesen dabei näher abzugrenzen und dann erst zu Konzeptionen, Repräsentationen und schließlich zu Objekten zu kommen. Ebenso beruht der Erfolg der Datenabstraktionstechnik (einer wesentlichen Grundlage für die OOP) nicht zuletzt auf der Einsicht, dass zentrale Konzepte wie Typen und Klassen primär nicht aufgrund struktureller Eigenschaften, sondern als Handlungspotentiale (= Mengen anwendbarer Operationen) zu definieren sind.

- Die vorherrschende Ansicht, man könne die (OO-) Objekte aus den Anwendungsfällen einfach durch Anstreichen der Substantive und Sammeln in einer "Objektliste" (vgl. [Jac 93])

herausholen, greift dagegen zu kurz. Hier können z.B. Begriffs-basierte Verfahren wesentlich mehr Hilfestellung leisten (vgl. [D-H 00]).

- Objektorientierte Analyse und Modellierung ist somit keine "Abbildung" unverrückbarer, von vornherein feststehender "natürlicher" Sachverhalte, sondern interessen-geleitete Konstruktionsarbeit von Akteuren - eben den Protagonisten der Software-Entwicklung. Das bedeutet für den einzelnen Akteur, dass sein Modellierungs-Spielraum möglicherweise größer als von ihm selbst angenommen ist. Was diesen begrenzt, sind oft weniger die (vorgeschobenen) "Sachzwänge", sondern die Interessen anderer (sichtbarer oder verborgener) Akteure.

Dank: Allen Mitgliedern und Mit-Diskutanten des Arbeitskreises "Terminologie der Softwaretechnik" sowie des Münchner Arbeitskreises zur Objektorientierten Analyse und Modellierung danken wir an dieser Stelle für die langjährige Zusammenarbeit und die zahllosen daraus erwachsenen Anregungen und Ideen.

Literaturhinweise:

[BHA 00] H. v. Braun, W. Hesse, U. Andelfinger, H.B. Kittlaus, G. Scheschonk.: Conceptions are social constructs - Towards a solid foundation of the FRISCO approach. In: [FLV 00]

[B-S 02] M. Broy, J. Siedersleben: Objektorientierte Programmierung und Software-Entwicklung - Eine kritische Einschätzung. Informatik-Spektrum ... 2002

[C-W 98] M Campione, K. Walrath: The Java Tutorial – Object-Oriented Programming for the Internet. 2nd ed., The Java Series, Addison Wesley 1998[D-H 00]S. Düwel, W. Hesse: Bridging the gap between Use Case Analysis and Class Structure Design by Formal Concept Analysis. In: *J. Ebert, U. Frank (Hrsg.): Modelle und Modellierungssprachen in Informatik und Wirtschaftsinformatik. Proc. "Modellierung 2000"*, pp. 27-40, Fölbach-Verlag, Koblenz 2000

[FHL+98] E. Falkenberg, W. Hesse, P. Lindgreen, B.E. Nilsson, J.L.H. Oei, C. Rolland, R.K. Stamper, F.J.M. Van Assche, A.A. Verrijn-Stuart, K. Voss: FRISCO - A Framework of Information System Concepts - The FRISCO Report. IFIP WG 8.1 Task Group FRISCO. Web version: http://www.wi.leidenuniv.nl/~verrynst/frisco.html (1998)

[FLV 00] E.D. Falkenberg, K. Lyttinen, A.A. Verrijn-Stuart (Eds.): Information System Concepts - An Integrated Discipline Emerging. Proc. IFIP TC8/WG8.1 Conference ISCO-4. Kluwer Academic Publishers 2000

[Gha 03] M. Ghasempour: Weisheit als Kritik der totalitären Wissensauffassung. Persönliche Kommunikation, Jan. 2003

[G-R 83] A. Goldberg, D. Robson: Smalltalk 80: The language and its implementation; Addison-Wesley 1983

[H-B 01] W. Hesse, H. v. Braun: Wo kommen die Objekte her? Ontologisch-erkenntnistheoretische Zugänge zum Objektbegriff. In: K. Bauknecht et al. (eds.): Informatik 2001 - Tagungsband der GI/OCG-Jahrestagung, Bd. II, S. 776-781. books_372ocg.at; Bd. 157, Österr. Computer-Gesellschaft 2001

[H-M 98] W. Hesse, H.C. Mayr: Highlights of the SAMMOA framework for object oriented application modelling. *Proc. DEXA 98 – 9th Int. Conf. and Workshop on Database and Expert Systems Applications*

[Jac 93] I. Jacobson et al.: Object-Oriented Software Engineering - A Use Case Driven Approach; Revised Printing, Addison-Wesley 1993

[Mey 88] B. Meyer: Object-Oriented Software Construction; Prentice Hall 1988

[Sow 00] J.F. Sowa: Knowledge Representation: Logical, philosophical and computational foundations. Brooks Cole Publ. Co. 2000

[Wit 61] Wittgenstein, Ludwig: Tractatus logico-philosphicus. Translated by D.F.Pears & B.F.McGuinness, with the Introduction by Bertrand Russell, Humanities Press International, Inc., Atlantic Highlands,NJ, 1961