Freie Universität Berlin
Institut für Informatik

Seminar zu Ursachen und Vermeidung von Fehlern in der Softwareentwicklung
Dozenten: Lutz Prechelt, Sebastian Jekutsch

Seminararbeit

Thema:

„Semantische Validierung der objektorientierten Analyse“

Julia Völkel
Mehringdamm 111
10965 Berlin
030 / 69.81.59.26
hektisch@web.de

SS 2006

Semantische Validierung der objektorientierten Analyse

Inhalt:

Einleitung

Im Rahmen der Fragestellung nach Ursachen und der Vermeidung von Fehlern allgemein bei der Entwicklung von Software zeigt sich, dass zahlreiche erst spät erkannte Probleme schon auf derjenigen Ebene wurzeln, die das Aufnehmen und Umformen von Anforderungen beinhaltet, sei es unbemerkt oder wider besseres Wissen.
Da es sich nicht grundsätzlich um Probleme handelt, die in der Natur der Softwareentwicklung liegen, sondern auch um solche, die mit geeigneten Methoden im Vorfeld erkannt werden könnten, lohnt eine genauere Betrachtung dieses Vorgangs. Somit werde ich mich vor allem mit semantischen Aspekten, d.h. der semantischen Validierung der objektorientierten Analyse (OOA) befassen. „Analyse“ wird hier in Abgrenzung zu „Entwurf“ (OOE) verwendet und heißt daher nichts anderes als die Fokussierung auf den Problembereich, also nicht auf den Entwurfs– bzw. Lösungsbereich.
Die objektorientierte Softwareentwicklung zeichnet sich vor allem durch ihre verhältnismäßig intuitive Herangehensweise aus. Die zu modellierenden Objekte lassen sich, so die idealistische Auffassung, direkt aus der so genannten „realen Welt“ auflesen und in entsprechende Modelle übertragen. Doch warum gelingt dies oft nicht so, wie der Benutzer bzw. der Anwendungsbereich es erfordert?
Dass es sich hier nicht um syntaktische Fehlerquellen handelt, liegt auf der Hand. Es ist die Ambivalenz der Sprache, die es zu erfassen gilt.
Ziel dieser Arbeit ist es, zunächst die Unterschiede zwischen der „natürlichen“, d.h. alltagspsychologischen Sichtweise einerseits und der objektorientierten Sichtweise andererseits zu erhellen, um im folgenden die daraus resultierenden Problembereiche anhand konkreter Beispiele zu analysieren. Im ersten Kapitel möchte ich auf die Konzeptstrukturen in der Psychologie eingehen, um sie dann mit jenen der Objektorientierung vergleichen zu können. Anschließend werde ich die daraus resultierenden Fehlerquellen aufzeigen. Im zweiten Kapitel werden die Heuristiken und

Probleme der objektorientierten Softwareentwicklung erläutert und anhand von Beispielen konkretisiert. Das dritte Kapitel schließlich resümiert die behandelten Aspekte und bietet Verbesserungsansätze an.

I) Pychologische Konzeptstrukturen und objektorientierte Konzepte

Das Ziel dieses Kapitels liegt darin, den Leser für die Schwierigkeiten von intuitiven Modellbildungen zu sensibilisieren. Zu diesem Zwecke werden im Folgenden die dazu verwendeten strukturellen Grundlagen natürlichsprachlicher und objektorientierter Konzepte angerissen. Besonderes Augenmerk wird dabei auf die sprachimmanenten, also psycholinguistischen Aspekte sowie die sozialen Interdependenzen gelegt. Eine eingehende Analyse der zwischenmenschlichen Kommunikation selbst wäre zwar ebenfalls sinnvoll, würde aber den Rahmen dieser Arbeit sprengen.

a) Vergleich

Ein Konzept ist das Ergebnis von Abstraktionen. Unter psychologischen Konzeptstrukturen sind Strukturen zu verstehen, die die menschliche Wahrnehmung der „Wirklichkeit“ als mentales Modell abbilden. Ebenso sollen objektorientierte Konzepte dazu dienen, ein handhabbares Modell der „realen Welt“ zu liefern.
Ein Vergleich beider Strukturen zielt darauf ab, aus den konzeptuellen Unterschieden mögliche Fehlerquellen für die objektorientierte Analyse ablesen zu können. Zentrale Fragen im Hinblick auf mentale Modelle sind dabei solche nach der kognitiven Verarbeitung von Information sowie nach der Funktion der Sprache bei diesem Vorgang. Daraufhin gilt es zu untersuchen, auf welche Weise die sprachlichen Elemente im objektorientierten Konzept anzusiedeln sind. Es ist in diesem Zusammenhang sinnvoll, zunächst zu erhellen, wodurch sich menschliche Kognition allgemein auszeichnet.

Kognitive Vorgänge

Ein kognitiver Vorgang bezieht sich auf komplexe Wahrnehmungs- und Erkenntnisleistungen. Das meiste, was unsere Interaktion mit der Umwelt steuert, vollzieht sich aber außerhalb unseres Bewusstseins.[1] So gestaltet sich auch die menschliche Wahrnehmung keineswegs nach objektiven Gesichtspunkten, sondern es handelt sich um eine Hypothesenbildung nach Gedächtnisinhalten, die sich bewährt haben.

„Wir machen uns nicht klar, wie stark unsere Ausgangsannahme die Art beeinflusst, wie wir Daten sammeln und auswerten."[2]

Aus Effizienzgründen automatisiert das Gehirn möglichst schnell sämtliche Prozesse, um so wenig Ressourcen wie möglich für kontrollierte Prozesse, d.h. bewusste Leistung aufzuwenden. Da automatisierte Prozesse wenig Aufmerksamkeit erfordern, werden sie für gewöhnlich als selbstverständlich empfunden. Im Rahmen unserer Fragestellung bedeutet dies zum Einen, dass Anforderungen oft nur unvollständig übermittelt werden, da die vermeintlichen Selbstverständlichkeiten auch bei sorgfältigem Nachdenken meist nicht in den Fokus der Aufmerksamkeit geraten und daher nicht erwähnt werden. Zum Anderen führt die – ebenfalls weitgehend unbewusste – Tendenz, bei anderen Menschen von einer weitgehend identischen Wahrnehmung auszugehen, zu ähnlichen Folgen. Dies gilt für den Anwender wie für den Entwickler gleichermaßen: der Anwender liefert vage oder lückenhafte Beschreibungen, der Entwickler ergänzt Unausgesprochenes durch sein eigenes, intuitives Verständnis.

Wissensorganisation

Um Informationen bzw. Erinnerungen zu verarbeiten und zu speichern, verwendet das menschliche Gehirn semantische Netzwerke: „Semantically related propositions are represented as nodes linked together in a hierarchical fashion."[3]

Damit die Informationen mit minimaler Redundanz verwaltet werden können, werden Hierarchien gebildet, die Gemeinsamkeiten zusammenfassen und Unterordnungen

1 Vgl. Roth (1997), S.30 ff.
2 Calvin (2004), S. 131
3 Ormerod: Human Cognition and Programming. In: Hoc, M-J. (Hrsg): Psychology of Programming. S. 71

herstellen. Hier lassen sich Parallelen zu dem Vererbungskonzept in der Objektorientierung ziehen (siehe dazu auch Kap.II c, *Erarbeitung einer Abstraktionshierarchie*). In diesem Aspekt bestehen also wichtige zu nutzende Gemeinsamkeiten zwischen den Psychologie-Konzepten und jenen der Objektorientierung.

Was aber versteht man allgemein unter „Objekten“? Im alltagspsychologischen Rahmen tendiert man dazu, statische Elemente zu assoziieren, im späteren OO-Modell treten Objekte allerdings als dynamische „Akteure“ auf.[4] Auf diese Unterscheidung wird im zweiten Teil noch näher eingegangen.
Um das Verständnis der im praktischen Teil beschriebenen Methoden und Schwierigkeiten zu sichern, ist eine Erläuterung der folgenden Begrifflichkeiten, *innerbegriffliche* und *zwischenbegriffliche Relationen* notwendig.

Innerbegriffliche und zwischenbegriffliche Relationen

Dynamische Abläufe werden in der menschlichen Wissensorganisation auf zweifache Weise repäsentiert: als einem Objekt zugeordnete Handlungen (deklarativ) oder als eigenständige operative Heuristiken (prozedural).[5]
Unter innerbegrifflichen Relationen (deklarativ) sind Vergleiche zwischen Konzepten hinsichtlich ihrer Merkmale zu verstehen; dies können sowohl Objekt- als auch Ereigniskonzepte sein. Mithilfe von innerbegrifflichen Relationen lassen sich Verfeinerungen bzw. Vergröberungen erfassen. So stehen z.B. die Aktivitäten „telefonieren“ und „schreiben“ in innerbegrifflicher Relation zu dem Oberbegriff „kommunizieren“.
Unter zwischenbegriffliche Relationen (prozedural) hingegen versteht man diejenigen Relationen, die Fragen nach Urheber, Ziel und Zweck, Instrument und Ort des Ereignisses beantworten. Sie sind weniger greifbar und verlangen nach umfangreicheren Betrachtungen.

4 Es sei darauf hingewiesen, dass die Unterscheidung von Objekten und Klassen im informatischen Sinne hier keine zentrale Rolle spielt.
5 Vgl. Philipsen: Konzeptuelle Validierung des objektorientierten Entwurfs, S.117

Abstraktionsebenen

Ebenfalls von Belang sind Fragen, die sich auf sprachliche Abstraktionen beziehen. In semantischen Konzeptstrukturen unterscheidet man drei verschiedene Abstraktionsebenen: die spezifische Ebene, welche sich auf einen speziellen Kontext bezieht und von dem etwas allgemeineren, so genannten „basic level“ abzugrenzen ist, und schließlich die abstrakte Ebene. Ist die spezifische Ebene beispielsweise „Forelle“, wird die Basisebene „Fisch“ sein und die übergeordnete Ebene „Tier“. Auch bei der Bildung von OO-Modellen spielt der Einsatz der entsprechenden Abstraktionslevel eine wichtige Rolle und sollte nach Möglichkeit sorgfältig erarbeitet werden.

In diesem Abriss hat sich gezeigt, dass das OO-Vorgehen den semantischen Konzeptstrukturen der Psychologie zwar durchaus nahe steht und es damit zu einem sinnvollen Unterfangen macht. Andererseits ließ sich erahnen, dass gerade die Stärke der Objektorientierung, ihr intuitiver Charakter, einige Gefahren birgt. Diese sollen im folgenden näher betrachtet werden.

b) Fehlerquellen

Der Objektbegriff

Die Auswahl von Objekten und Zuordnungen von Operationen ist nicht so intuitiv durchzuführen, wie es scheinen mag. Die Schwierigkeit liegt darin, dass der Objektbegriff der Psychologie, der zunächst die Beschaffenheit und erst zur weiteren Spezifizierung der Kategorien die verhaltensrelevanten Charakteristika betrachtet, nicht unbedingt mit dem Objektbegriff der Informatik, welcher die Funktionsweise von Objekten betont, übereinstimmt. Problematisch wird es allerdings hauptsächlich bei der Beschreibung von prozeduralen Abläufen. Im alltagspsychologischen Rahmen bezeichnet ein Objekt eine statische Einheit, im dynamischen Software-Modell jedoch etwas Aktives, dem als Operationen auszuführende Funktionen zugrunde liegen. Der Objektbegriff der Informatik ist auf alltagssprachlicher Ebene eher mit dem Begriff

„Ereignis“ vergleichbar. Diese unterschiedliche Perspektive begründet unter anderem die Schwierigkeit, den Objekten Handlungen, d.h. Operationen zuzuweisen.

Abstraktionsebenen

Eine weitere Fehlerquelle liegt in der Handhabung der verschiedenen Abstraktionsebenen. Es ist im praktischen Gebrauch stets darauf zu achten, dass die Ebenen sich nicht unbemerkt vermischen und so zu Fehlern führen. Bei der OOA sollte man bedenken, dass Anwender oft auf der mittleren, d.h. der alltagsrelevanten Abstraktionsebene antworten. Zu überlegen wäre, ob eine Beschränkung der Mehrfachvererbung auf drei Ebenen auch in der Modellbildung bzw. späteren Implementierung sinnvoll, da intuitiv leichter erfassbar wäre.[6]

Missverständnisse und Fehlinterpretationen

Weit grundlegendere Schwierigkeiten entstehen jedoch durch Missverständnisse während der Planungsphase. Sie beruhen auf kommunikativen Unwägbarkeiten. So wird immer wieder vernachlässigt, dass Unterschiede in der Wahrnehmung, z.B. durch unterschiedliches Vorwissen, andere Schwerpunktsetzung usw. auch unterschiedliche Interpretationen desselben Sachverhalts bedingen. Oft geht man selbstverständlich von einer gleichen Wahrnehmung aus. Im Zuge dieser Überlegungen gilt es weiterhin zu berücksichtigen, dass Äußerungen keineswegs einen eindeutigen Informationsgehalt transportieren. Somit kann die Kontextabhängigkeit von Begriffen, die im natürlichen Sprachgebrauch grundlegend für ihr Verständnis ist, zu einer beträchtlichen Fehlerquelle werden. Wird der Kontext vernachlässigt oder falsch verstanden, so ergeben sich daraus oft falsche Schlussfolgerungen, die sich auf die Brauchbarkeit der Software auswirken. Denn der Informationsgehalt bestimmt sich allein durch den Zusammenhang. „Verstehen [..] kann es deshalb nur dann geben, wenn es einen festen oder fest verabredeten semantischen Kontext gibt.“[7] Lässt der Entwickler aber sein intuitives Verständnis einfließen, statt sich eingehend mit dem Kontext

6 Vgl. Philipsen (1995), S.115
7 Roth (1997): 107 ff.

auseinanderzusetzen, so sind Fehlinterpretationen eine häufige Folgeerscheinung. Schärft man hingegen den Blick für die Ursachen von Missverständnissen, so könnten letztere wesentlich reduziert werden.

II) Heurismen der OOA

Ein Ansatz zur semantischen Validierung besteht in der Untersuchung, ob und inwieweit die in der OOA und OOE genutzten Methoden geeignet sind, um die semantischen Konzepte der alltäglichen Sprache herauszuarbeiten und zu berücksichtigen. Basierend auf der von Booch[8] vorgeschlagenen, alternierenden Vorgehensweise des „analyze a little, design a little“ werden in diesem Kapitel exemplarisch einige zentralen Methoden herausgegriffen und näher betrachtet. Dabei werde ich zunächst die jeweilige Heuristik beschreiben, um davon ausgehend anhand geeigneter Beispiele die jeweiligen Schwierigkeiten zu entwickeln. Die vier hier behandelten Heurismen

a) Auswahl von Objekten
b) Bestimmung von Operationen
c) Erarbeitung einer Abstraktionshierarchie
d) und Erarbeitung einer Teil/Ganzes-Relation

werden im Folgenden näher erläutert. Innerhalb des von Booch[9] entwickelten Ansatzes betreffen die Heurismen vor allem den Schritt der Herleitung von Klassen und Objekten und der Identifikation der dahinter stehenden Semantiken.

a) Auswahl von Objekten

Als einer der ersten Schritte der Softwareentwicklung werden relevante Objekte und Attribute aus dem Arbeitsbereich herausgearbeitet und in einer unstrukturierten Materialsammlung angelegt, in der zunächst die Einheitlichkeit der Abstraktionsebene eine untergeordnete Rolle spielt. An dieser Tätigkeit setzt die Heuristik an und sieht vor,

8 Booch, G.: Object-Oriented Design, S. 201
9 Booch, G.: Object-Oriented Design, S. 190

aus der Dokumentation der Aufgabenstellung die Nomen auszusondern und in die Sammlung der Objektbeschreibungen aufzunehmen. Ferner können bei direkter Befragung der Anwender konkrete Gegenstände der Arbeitsumwelt in die Objektliste aufgenommen werden und später durch abstraktere Begriffe ergänzt bzw. unter abstrakteren Begriffen zusammengefasst werden

Die beschriebene Vorgehensweise beinhaltet jedoch die Gefahr, Objektbeschreibungen und Attribute zu verwechseln. So kann das Nomen „Farbe“ im Kontext des Objektes „Brief“ ein Attribut sein, in einer physikalischen Anwendung jedoch durchaus selbst als Objekt auftreten, dem Attribute wie beispielsweise „Wellenspektrum“ zugeordnet werden. Auch die Psychologie kennt ähnliche Probleme und kann ebenfalls keine kontextunabhängige Grenze zwischen Konzepten und Merkmalen ziehen[10]. Aus Untersuchungen zur Differenzierung von Merkmalen und Objekten lassen sich jedoch zwei grundsätzliche Aussagen herleiten[11]:
Zunächst kann nachgewiesen werden, dass die Aufgabenstellung Einfluss auf die Benennung von Objekten und deren betrachteten Merkmalen hat. Weiterhin konnte festgestellt werden, dass sich die Vorgabe irrelevanter Merkmale ungünstig auf Problemlösungsprozesse auswirkt und daher vermieden werden sollte.

Zusammenfassend kann daher empfohlen werden, dass bei der Dokumentation der Aufgabenstellung und bei der Formulierung der Objektbeschreibung der Aufgabenkontext berücksichtigt und lediglich aufgabenrelevante Informationen aufgenommen werden sollten.

b) Bestimmung von Operationen

Nach der Identifikation der Objekte muss herausgearbeitet werden, welche Operationen an beziehungsweise von den Objekten ausgeführt werden sollen. Operationen können

10 vgl. Herrmann, Th.: Allgemeine Sprachpsychologie, S.80
11 vlg. Labov, W.:The boundaries of words and their meanings, 340ff; vgl. Duncker, K.: On Problem Solving, S. 270

dabei sowohl Anfragen bezüglich der Attribute des Objektes darstellen als auch Funktionsweisen des Objektes beschreiben. Dabei werden gleichzeitig dynamische Aspekte wie zeitliche oder örtliche Voraussetzungen für die Ausführung der Operation betrachtet. Zur Bestimmung relevanter Operationen sieht die Heuristik vor, die Verben aus der Problembeschreibung auszusondern und den zuvor bestimmten Objektbeschreibungen zuzuordnen. Alternativ können Anwender zu ihren gewohnheitsmäßigen Umgangsformen mit den Objekten befragt werden, wobei darauf zu achten ist, dass Anwender selbstverständliche Operationen oft nicht thematisieren[12]. Unabhängig von der Vorgehensweise bei der Identifizierung können und sollen Operationen mehreren Objekten zugeordnet werden, was Booch[13] damit begründet, dass die Eigenständigkeit von Objekten weiterhin aufrechtzuerhalten ist, um die Interaktionsstruktur zwischen Objekten nicht vorwegzunehmen.

Die Problematik der Heuristik zeigt sich bei der Betrachtung von Verben, die in ihrer Bedeutung stark variieren können und folglich näher durch ein Objekt bestimmt werden müssen. So kann das Verb „bewegen“ abhängig vom Akkusativobjekt sehr unterschiedlich interpretiert werden und kann erst durch die Zusammenführung mit Objekten wie „Cursor“ oder „Mausrad“ in seiner Bedeutung nachvollzogen werden. Sobald jedoch ein Objekt beigestellt wird, verliert sich die Eigenständigkeit der Objekte und Objektinteraktion wird hergestellt. Dies führt jedoch unmittelbar zur Frage nach der Abstraktionsstufe, die dem Objekt zugeordnet werden sollte (vgl. auch Abschnitt c), und erhöht damit die Komplexität der Analyse. Clauser[14] weist in diesem Zusammenhang darauf hin, dass Anwender u.U. Objekte einer höheren Abstraktionsebene benennen, auch wenn der Kontext eine genauere Bezeichnung zuließe. Zusammenfassend sollten daher Operationen, die unabdingbar bereits ein Objekt voraussetzen, besonders auf die mögliche Konkretisierung des Objektes geprüft werden.

12 Clauser, Constanze: Überlegungen zur semantischen Validierung der objektorientierten Analyse, S. 96
13 Booch, G.: Object-Oriented Development, S. 213
14 Clauser, C.: Überlegungen zur semantischen Validierung der objektorientierten Analyse, S. 97

c) Erarbeitung einer Abstraktionshierarchie

Mit der hierarchischen Ordnung von Objekten gewinnt der Aspekt der Beschreibungsebenen (Objektensemble, abstrakte und konkrete Klassen, etc.) an Bedeutung, der zur Bestimmung von Objekten und Operationen bewusst ausgeblendet wurde[15]. Der Zweck der Hierarchisierung liegt dabei in der Vererbung von Attributen und Funktionsweisen und kann so beispielsweise zur Einsparung von Programmcode eingesetzt werden. Aus Ergonomiegründen sollten Hierarchien jedoch primär entsprechend der Konzeptstrukturen des Anwenders gestaltet werden und die Reduzierung von Programmcode nachgestellt werden.

Die Heuristik schlägt zur Bildung einen zweistufigen Ansatz vor: Zunächst werden Klassen aus Objekten mit gemeinsamen Operationen gebildet: „Focus on behavior and group by behavior“ (Gibson, 1990)[16]. Zur Bildung von Hierarchien zwischen Klasse_A und Klasse_B wird dann in einem zweiten Schritt die Frage „Ist Klasse_A eine Klasse_B?“ eingesetzt. Lässt sich dies aus Sicht des Anwenders bestätigen, wird Klasse A hierarchisch Klasse B untergeordnet. Als Klasse_A sollte dabei diejenige gewählt werden, die über mehr bzw. spezifischere Operationen verfügt. „Klasse“ ist hier (wie auch in Abschnitt *d*) zunächst als allgemeine Klassifizierung zu verstehen. Ob es sinnvoll ist, diese Strukturen später äquivalent in die OO-Klassenstruktur zu übertragen, muss je nach Fall entschieden werden.

Das Problem dieser heuristischen Methode liegt darin, dass nur ein Teil der Operationen zur Hierarchisierung verwendet werden. Die Klassen- und Hierarchiebildung verfolgt das Ziel, Objekte mit gleichen Operationen zusammenzufassen. Operationen können dabei sowohl Attribute (innerbegriffliche Aspekte) als auch Funktionsweisen (zwischenbegriffliche Aspekte) des Objekts betreffen. Die oben genannte Frage zielt jedoch ausschließlich auf den Vergleich der

15 Clauser, C.: Überlegungen zur semantischen Validierung der objektorientierten Analyse, S. 97

16 vgl. Gibson: Objects – Born and Bred, S. 246

Attribute von Klasse_A und Klasse_B ab, zwischenbegriffliche Aspekte werden nicht berücksichtigt. So wird ein Anwender die Frage, ob Kugelschreiber, Bleistifte, Füller und Tintenkiller in die Klasse „Stift“ fallen, in aller Regel bejahen, da diese insbesondere in der äußeren Form vergleichbar sind. Wird ein Stift jedoch mit üblichen Funktionsweisen wie „schreiben“, „ankreuzen“ oder „unterstreichen“ charakterisiert, so lässt sich ein Tintenkiller nicht mehr dieser Klasse zuordnen. Gibson wies bereits auf dieses Problem hin: „People generally identify objects initially in terms of state and then use the behavioral characteristics to distinguish and refine the groupings or categories.”[17] Hussy[18] relativiert dieses Ergebnis jedoch, indem er darauf hinweist, dass Attribute, die zur Hierarchiebildung herangezogen werden, mit ansteigendem Abstraktionsniveau zunehmend handlungsrelevant werden. So ließen sich die zuvor genannten Beispielbegriffe strukturieren, indem eine weitere Hierarchieebene hinzugenommen wird: Kugelschreiber, Bleistifte und Füller lassen sich einer Klasse „Schreibstift“ zuordnen, die die oben genannten Funktionsweisen „schreiben“, „ankreuzen“ und „unterstreichen“ berücksichtigt. Sind daneben auch Funktionsweisen wie „leicht in der Hand zu halten“ oder „kontrolliert über Papier zu bewegen“ für die Problemlösung relevant, so können diese einer zweiten Klasse „Stift“ zugeordnet werden. Über Attribute wie Form und Größe können dann sowohl die in der Klasse „Schreibstift“ zusammengefassten Objekte als auch der Tintenkiller in die Klasse „Stift“ aufgenommen werden.

Zusammenfassend kann empfohlen werden, die Heuristik einzusetzen, wenn Objektattribute bei den Vererbungsrelationen vorrangige Bedeutung haben oder durch abstrakte Attribute funktionale Merkmale zur Hierarchisierung herangezogen werden. Steht dagegen die Vererbung gemeinsamer Funktionsweisen im Vordergrund, sollte auf Methoden zur Untersuchung zwischenbegrifflicher Relationen ausgewichen werden. Clauser[19] weist außerdem darauf hin, dass eine beschränkt einsetzbare Heuristik wie die hier beschriebene vor allem dann zu Problemen führen kann, wenn der

17 vgl. Gibson: Objects – Born and Bred, S. 250
18 Hussy, W.: Denkpsychologie, S. 91
19 Clauser, C.: Überlegungen zur semantischen Validierung der objektorientierten Analyse, S. 99

Softwareentwickler keine Erfahrung im Anwendungsbereich hat und das Ergebnis nicht auf Plausibilität prüfen kann.

d) Erarbeitung von Teil/Ganzes-Relationen

Neben der Hierarchisierung von Objekten in Klassen spielt die Objektstrukturierung in Teil/Ganzes-Relationen (T/G-Relationen) eine wichtige Rolle in objektorientierten Softwareentwürfen. Nach Clauser[20] besteht eine T/G-Relation dann, wenn „ein Objekt als Teilstruktur (d.h. in einem Feld) in einem anderen Objekt enthalten ist“. Aus der Perspektive des Softwaredesigns bergen T/G-Relationen sowohl technische Vorteile, wie beispielsweise die Optimierung des Speicherplatzbedarfs, als auch ein Risiko bei Änderungen, im Rahmen derer die T/G-Relation aufgebrochen werden. Alternativ zur T/G-Relation ließen sich Objekte auch in einer Assoziation, also einer flexibleren Aufteilung in getrennten Klassen, realisieren. Ausschlaggebend für die Organisation von Objekten in T/G-Relationen ist daher die Bewertung der Änderungswahrscheinlichkeit und des damit verbundenen Aufwands. Dabei ist der Softwareentwickler auf die Erfahrungen von Experten aus dem Problembereich angewiesen. Soll der Anwender bei der Erarbeitung von T/G-Relationen eingebunden werden, wird häufig die Frage-Heuristik „Hat Klasse_A eine Klasse_B“ empfohlen[21]. Wird die Frage bejaht, so wird Klasse_B als Teil von Klasse_A betrachtet und eine T/G-Relation (oder alternativ eine Assoziation) eingefügt.

Die Vorgehensweise ist in mehrerer Hinsicht mit Problemen belastet: Zunächst kann das Verb „haben“ unterschiedliche Bedeutungen wie beispielsweise „besitzen“, „über etwas verfügen“, „innewohnen“, „aufweisen“ oder auch „Bestandteil sein von“ annehmen. So sollte die Bejahung der Frage „Hat Martin Masern?“ nicht als T/G-Relation zwischen „Masern“ und „Martin“ interpretiert werden. Durch eine genauere Formulierung der Frage „Ist Klasse_A Bestandteil von Klasse_B?“ lassen sich falsche

20 Clauser, C.: Überlegungen zur semantischen Validierung der objektorientierten Analyse, S. 100
21 Clauser, C.: Überlegungen zur semantischen Validierung der objektorientierten Analyse, S. 101

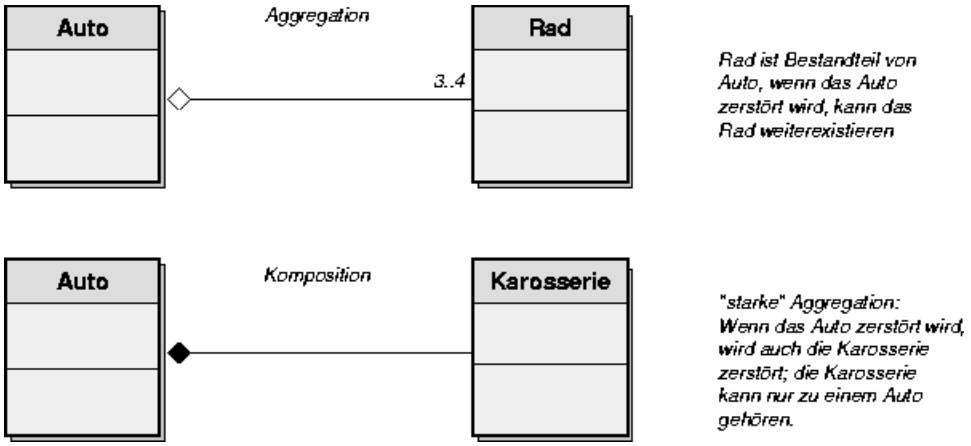
Interpretationen jedoch weitgehend vermeiden.

Wesentlich problematischer ist dagegen die Verwendungsweise der T/G-Relation selbst, vor allem wenn T/G-Relationen bewusst oder unbewusst in T/G-Hierarchien kombiniert werden. De Champeaux und Faure[22] weisen auf die Risiken unklarer Transitivitätsbedingungen hin. So könnte aus den Sätzen „Herr B. ist Teil der Belegschaft." und „Das Herz von Herrn B. ist Teil von Herrn B." geschlossen werden, dass das Herz von Herrn B. ein Teil der Belegschaft ist. Chaffin und Herrmann[23] vertiefen die Frage der Transitivität in T/G-Strukturen und kommen zu dem Schluss, dass verschiedene Arten von T/G-Relationen (z.B. Component/Integral Object, Member/Collection) existieren. T/G-Relationen sind dann transitiv, wenn in allen T/G-Relationen der Hierarchie die selbe Klasse von T/G-Relationen vorliegt.

Die kontextabhängige Beurteilung von T/G-Relationen stellt ein weiteres Problem der Erarbeitung von Teil/Ganzes-Beziehungen dar[24]. So kann ein Anwender den Monitor seines Computers als Bestandteil seiner Computeranlage verstehen und damit den Monitor in einer T/G-Relation zum Computer sehen. Ein Elektronikfachmann betrachtet dagegen Monitor und Computer als separate Klassen, die in keinem Falle in einer Teil-Ganzes-Beziehung stehen.
Schließlich ist es in manchen Fällen schwierig, zwischen T/G- und hierarchischen Vererbungs-Strukturen zu unterscheiden. Als Beispiel beschreibt Meyer[25] die Relation zwischen den Klassen „Fenster" und „Editierfenster". So könnte die Klasse „Fenster" Funktionsweisen wie „Öffnen", „Schließen", „Minimieren" oder „Maximieren" an die Klasse „Editierfenster" vererben. Allerdings könnte auch „Editierfenster" als ein Konzept betrachtet werden, das unter anderem aus einem Fenster, Editormenu und anderen Elementen besteht. Diese Interpretation spräche für die Abbildung einer T/G-Relation. Meyer empfiehlt hier anhand von technischen Argumenten wie beispielsweise dem

22 De Champeaux D., Faure, P.: A Comparative Study of Object-Oriented Analysis Methods
23 Chaffin, M.E., Herrmann, D.: A Taxonomy of Part-Whole Relations
24 Clauser, C.: Überlegungen zur semantischen Validierung der objektorientierten Analyse, S. 102
25 Meyer, B.: Object Oriented Software Construction

Aufwand im Falle von Implementierungsänderungen zu entscheiden. Es sollte jedoch auch das Verständnis des Benutzers in die Entscheidung einfließen. Aus dem Gesichtspunkt der Wiederverwendbarkeit ist außerdem eine einheitliche Gestaltung wünschenswert.
Als zusammenfassende Bewertung wird die Heuristik „Ist Klasse_A Bestandteil von Klasse_B?“ lediglich zum Einstieg in die Erarbeitung von T/G-Relationen oder Assoziationen empfohlen. Im Falle von T/G-Hierarchien muss zusätzlich geklärt werden, ob Transitivität erwünscht ist. Hierzu kann die Studie von Chaffin und Herrmann[26] herangezogen werden. Auch die Entscheidung, ob eine T/G-Relation oder eine Vererbungs-Relation für den vorliegenden Problembereich geeignet ist, muss im Einzelfall geprüft werden. Zur Entscheidung, ob eine Teil-Ganzes-Beziehung in Form einer T/G-Relation oder einer Assoziation umgesetzt wird, werden die von De Champeaux und Faure[27] formulierten Fragen empfohlen:
Kann ein Objekt Teil von verschiedenen Ganzen sein?
Kann ein Teil bestehen bleiben, wenn das übergeordnete Ganze, dessen Teil es ist, nicht mehr existiert?
Siehe folgendes UML-Diagramm zur Veranschaulichung:



26 Chaffin, M.E., Herrmann, D.: A Taxonomy of Part-Whole Relations
27 De Champeaux D., Faure, P.: A Comparative Study of Object-Oriented Analysis Methods

III) Zusammenfassung

In dieser Arbeit wurden einige Ansätze einer semantischen Validierungsmethode aufgezeigt, die einen am Benutzer orientierten Entwurf unterstützt. Besonderes Augenmerk wurde dabei auf die Vermeidung von Irrtümern und Fehlern gelegt.
Es wurde deutlich, dass die Relevanz von kommunikativen Prozessen inklusive ihrer Kontextbetrachtung kaum unterschätzt werden kann.
Zusammenfassend lässt sich festhalten, dass vor allem in der Planungsphase zahlreiche Fehlerquellen lauern, die oft zu Irrtümern führen. Da auf diese Weise entstandene Defekte besonders aufwändig zu finden und zu beheben sind, sollte gerade der Anforderungsermittlung besondere Aufmerksamkeit zukommen. Wie sich im Vorangehenden herausstellte, beruhen viele Fehler auf kommunikativen Missverständnissen. Diese entstehen wiederum oft aus der falschen Annahme heraus, alles Relevante gesagt zu haben bzw. intuitiv erfasst zu haben, was gemeint ist. Es ist daher sinnvoll, verstärkt auch nach vermeintlichen Selbstverständlichkeiten zu fragen, um Fehlinterpretationen zu vermeiden.
Wie sich gezeigt hat, sind die hier vorgestellten Methoden der OOA zwar hilfreich, können aber nicht die eingehende Beschäftigung mit dem Aufgabenkontext ersetzen, dessen Verständnis konstitutiv für das Gesamtverständnis ist. Ein solches ist vor allem bei komplexen Aufgaben, die mehrere Möglichkeiten der Umsetzung zulassen, unabdingbar. Ein Softwareentwickler sollte also über ein fundiertes Domänenwissen verfügen oder zumindest sehr gut verstehen, aus welchen Gründen eine Anwendung wie funktionieren soll. Da er zu diesem Zwecke eine gültige Beschreibung der Objekte und der Interaktion zwischen Objekten benötigt, müssen diese sorgfältig herausgearbeitet werden.
Allerdings werden Objekte in der OOA als aktiv beschrieben, in der Psychologie aber als Objekte im Sinne von Instrumenten. Daher ist es schwierig, den Objekten Handlungen bzw. Operationen zuzuweisen. Es bietet sich hierbei an, - im Idealfall mithilfe des Anwenders - zu überlegen, was am natürlichsten erscheint.
Die größten Probleme wurden dabei in bezug auf zwischenbegriffliche Relationen

ausgemacht. Gründe dafür sind, dass zwischenbegriffliche Relationen als langfristig gespeichert, aber unterschiedlich leicht zugänglich gelten, während innerbegriffliche Relationen bei der Beurteilung jeweils neu erschlossen werden. Die Schwierigkeit führt also wiederum auf die menschlichen Denkschemata zurück.
Es gilt also, diese Besonderheiten der Gedächtnisleistungen bei der OOA stets im Auge zu behalten und nach Möglichkeit noch stärker zu berücksichtigen.
Ein weiterer Schritt der semantischen Validierung könnte darin bestehen, neben der kritischen Betrachtung der Heuristiken der OOA auch Überlegungen zur Prüfung von OOA– und OOE-Ergebnissen anzustellen.

IV) Literaturverzeichnis

Booch, G.: *Object-Oriented Development*. In: IEEE Transactions on Software Engineering, 12(2), 211-221, 1986

Booch, G.: *Object-Oriented Design*. Redwood City: Benjamin/Cummings, 1991

Calvin, William H.: *Wie das Gehirn denkt. Die Evolution der Intelligenz.* München: Spektrum Akademischer Verlag, 2004

Chaffin, M.E., Herrmann, D.: *A Taxonomy of Part-Whole Relations*. In: Cognitive Science, 15, S. 273-281

de Champeaux, D. and Faure, P.: *A Comparative Study of Object-Oriented Analysis Methods*. In: Journal of Object-Oriented Programming, 5 (1), S. 21 – 33, 1992

Clauser, Constanze: *Überlegungen zur semantischen Validierung der objektorientierten Analyse.* In: Dzida, Konradt (Hrsg.): *Psychologie des Software-Entwurfs*. Stuttgart: Verlag für Angewandte Psychologie, 1995

Détienne, Françoise: *Software Design – Cognitive Aspects.* London: Springer, 2002

Duncker, K.: *On Problem Solving*. In: Psychological Monographs, 58, S. 270, 1945

Gibson, E.: *Objects – Born and Bred*. In: Byte, 15 (10), S. 245 – 254, 1990

Herrmann, Th.: Allgemeine Sprachpsychologie. München: Urban & Schwarzenberg, 1985

Hoc, J.-M. (Hrsg): *Psychology of Programming.* London: Academic Press, 1990

Hussy, W.: *Denkpsychologie*. Stuttgart: Kohlhammer, 1984

Labov, W.: *The boundaries of words and their meanings*. In: C.-J.N. Bailey und Shuy, R.W. (Hrsg.): *New ways of analysing variation in English*. Washington: Georgetown University Press, S. 340-373, 1973

Meyer, B.: Object *Oriented Software Construction*. New Jersey: Prentice Hall, 1988

Philipsen, G.: *Konzeptuelle Validierung des objektorientierten Entwurfs.* In: Dzida, Konradt (Hrsg.): *Psychologie des Software-Entwurfs*. Stuttgart: Verlag für Angewandte Psychologie, 1995

Roth, Gerhard: *Das Gehirn und seine Wirklichkeit.* Frankfurt am Main: Suhrkamp, 1997

Weinberg, Gerald M.: *The Psychology of Computer Programming.* New York: Litton Educational Publishing, 1971