Dienstag 8–10 in MZH 1400 & Donnerstag 8–10 in MZH 7260 (4 SWS. 4 ECTS. VAK 03-799.20)
nake@informatik.uni-bremen.de www.agis.uni-bremen.de 0421-218 3525 linzer str. 9a 3015
www.agis.informatik.uni-bremen.de/IMZ/

Zur dritten Kennzeichnung: Die Software als algorithmisches Zeichen

1.12.2003

Gesehen hatten wir nun längst: Computer sind Maschinen zur automatischen Auswertung berechenbarer Funktionen an gegebenen Argumentstellen.

Die zu berechnende Funktion und ihre Argumente werden als „Software" gegeben: als Programme und Daten.

Unsere Behauptung ist, es sei günstig und käme dem wesentlichen Kern von Software nahe, wenn wir Software semiotisch auffassen, ihr also Zeichencharakter zubilligen.

Wie erscheinen uns Dinge und Phänomene, die wir „Zeichen" nennen? Wir finden visuelle Zeichen, wie sie durch die Geschichte und Kulturen hindurch bekannt sein mögen, in Sammlungen wie denen von Rudolf Koch oder Adrian Frutiger (s. Literaturliste). Dort erscheinen Flecken von Druckerschwärze auf weißem Papier. Naiv nennt man sie „Zeichen". Sie sind jedoch nur Anlass für Zeichenprozesse. In einem Buch finden wir kein Zeichen. Was wir im Buch finden, können wir jedoch zum Zeichen machen. In solch einem Zeichenprozess (einer „Semiose") wird die vorliegende und wahrgenommene Druckerschwärze zum *Repräsentamen* eines Zeichens gemacht (unbewusst in aller Regel). Hiermit beginnt die Zeichenbildung, die noch zu einem Objekt („was wird bezeichnet?") und einem Interpretanten führt („wozu wird bezeichnet?").

Zeichen sind in dieser Auffassung von Peirce triadische Relationen. Sie sind gemacht, nicht gegeben; sind Relation, nicht Ding; dreistellig, nicht zweistellig. Dieser Zeichenbegriff ist rekursiv. Denn der Interpretant ist selbst wieder nur als Zeichen zu fassen. Der Vorgang hat kein Ende. Gerade darin liegt der Reiz des Peirceschen Begriffs.

Im Zeichen erscheint uns ein Abwesendes als gegenwärtig. Unser Gedanke verbindet das abwesende Etwas mit einem anwesenden, das als Signal wahrgenommen wird. Die mittelalterliche Formel von Augustinus (354-430) lautet: *aliquid pro aliquo*. Zeichen umgeben uns immer schon. Wir kennen keine semiosefreie Existenz.

„Zeichen ist alles, was zum Zeichen erklärt wird, und nur was zum Zeichen erklärt wird. Jedes beliebige Etwas kann (im Prinzip) zum Zeichen erklärt werden." (Max Bense, Semiotik, 1967).

Die Reduktion des Zeichens auf die einstellige Unterrelation (Repräsentamen) wird in der *Syntaktik* behandelt („Wie wird bezeichnet?" Mittel). Hier geht es um die Präsentationsfunktion des Zeichens. Sie geschieht durch Akte der Selektion.

Die Reduktion des Zeichens auf die zweistellige Unterrelation (Repräsentamen, Objekt) wird in der *Semantik* behandelt („Was wird bezeichnet?" Inhalte). Hier geht es um die Repräsentationsfunktion des Zeichens. Sie geschieht durch Akte der Denotation.

Die Betrachtung des Zeichens als dreistellige Relation (Repräsentamen, Objekt, Interpretant) wird in der *Pragmatik* behandelt ("Wozu wird bezeichnet?" Zwecke). Hier geht es um die Kommunikationsfunktion des Zeichens. Sie geschieht durch Akte der Interpretation.

Der Interpretant des Zeichens ist kein Mensch, sondern ein Zeichen. Es gibt aber kein Zeichen ohne den Interpretationsakt eines Menschen. Der Mensch erscheint als Interpret.

Peirce unterscheidet in feinerer Betrachtung das unmittelbare und das dynamische Objekt sowie den unmittelbaren, dynamischen und finalen Interpretanten. Hierauf gehe ich hier nicht ein.

Wir sind stets gegenwärtig, wenn Zeichenprozesse geschehen. Wir sind gegenwärtig (1) als Individuen, die (2) zu Gruppen gehören, die (3) in einer Gesellschaft existieren. (Beispiel: das Individuum (1) FN ist (2) Hochschullehrer der Informatik in Bremen und (3) Mitglied der deutschen Gesellschaft.)

Da der Zeichenprozess der fortgesetzten Bildung von Interpretanten nicht endet, sondern immer nur abgebrochen wird, gibt es keine endgültigen Erklärungen, stets nur vorläufige. Aus diesem prinzipiuellen Agnostizismus gibt es für Menschen kein Entrinnen. Dem Faust verbrennt das schier das Herz. Uns tröstet es.

Die informatischen Gegenstände nun sind semiotisch geprägt. Damit behaupten wir, dass sich der Charakter der informatischen Gegenstände (also der Software) günstig als Zeichen (und Zeichenprozess) beschreiben lässt. Wir sagen genauer: „Software erscheint uns als algorithmisches Zeichen."

Software ist physikalisch im Speicher vorhanden (auf einem externen Massenspeicher, einer Platte, im Arbeitsspeicher). Sie existiert vielleicht zusätzlich auf Papier oder auf dem Bildschirm. Doch in diesen wahrnehmbaren Formen fungiert sie nicht mehr als Software, sie wird in der Darstellung etwas anderes.

Auf dem Papier skizziert der Programmierer die Software, so meint er. Er schafft dort, genauer, Zeichen *für* Software. Aus seinen Zeichen nämlich wird Software, indem diese Zeichen als Signale in den Speicher gebracht werden. Was wir auf dem Bildschirm sehen, sind Zeichen *von* Software. Wir erfahren durch sie etwas von der Existenz der Software. Wir glauben, indem wir diese Projektion wahrnehmen, dass es die Software gibt. Die Projektion auf der GUI gibt uns Kunde von der Software.

Die Software haust im Speicher. Dort können wir sie in keiner Weise wahrnehmen. Wir haben kein Sinnesorgan für die Art der Materialität von Software. Insofern besitzt sie einen quasi-immateriellen Charakter. Erst durch den Akt der Projektion wird uns die Software wahrnehmbar. Sie ist in ihrem für uns unsichtbaren Zustand für den Computer manipulierbar. In ihrem für uns sichtbaren Zustand aber ist sie für den Computer nicht manipulierbar (selbst das Licht auf dem Bildschirm wird dadurch manipuliert, dass der Display Buffer manipuliert wird).

Da wir die Nicht-Wahrnehmbarkeit von Software nicht hinnehmen, verlangen wir nach Darstellungsflächen (Präsentationen). Sie werden als GUIs organisiert. In ihrem immateriellen Charakter nun zeigt sich ein relationaler und also semiotischer Charakter von Software. Sie setzt nämlich zueinander ins Verhältnis: Code, *Runs* und Kontextualisierung.

Im (unsichtbaren) *Code* als Speicherinhalt ist die Software der statische Programmtext. Er füllt einen Teil des Speichers. Der Code ist, zusammen mit codierten Daten, Anlass für einen Lauf (*Run*) des Programmes. Die tendenziell unendliche Menge von *Runs* stellt die dynamische Seite der Software dar, das Objekt, das bezeichnet wird. Jeder dieser Läufe zeigt sich in einem Verhalten, das in Ergebnissen mündet, bzw. in Eingabeoperationen, die wir tätigen, um einen Lauf beginnen zu lassen oder fortzusetzen. Dabei können wir nicht anders als zu interpretieren, also in unsere *Kontexte* und Situationen einzubetten. Diese interessegebundene Situierung stellt den Interpretanten der Software als Zeichen dar.

Wir begegnen der Welt in den Dingen. Oder wir zerlegen Welt in Dinge, wenn wir uns in ihr bewegen. Wir geben uns mit den Dingen so, wie sie sind, nicht zufrieden. Wir reflektieren sie, abstrahieren von ihren zufälligen Eigenschaften, schaffen uns Modelle der Dinge, also Zeichen von ihnen und für sie. In den Wissenschaften behandeln wir die Zeichenwelten auf je spezifische Weise.

Dienstag 8–10 in MZH 1400 & Donnerstag 8–10 in MZH 7260 (4 SWS. 4 ECTS. VAK 03-799.20)
nake@informatik.uni-bremen.de www.agis.uni-bremen.de 0421-218 3525 linzer str. 9a 3015
www.agis.informatik.uni-bremen.de/IMZ/

Die formalen Zeichenwelten der Mathematik werden nun von der Informatik ergriffen und erneut zu Dingen gemacht, die uns gegenüberstehen. Sie stehen uns jedoch anders gegenüber als die uns bekannten materiellen Dinge: als quasi-immaterielle, die prozessieren, sich in Prozesse verwandeln können durch ein geringes Dazutun. Hierin enthüllt sich deren Charakter als *algorithmische* Zeichen.

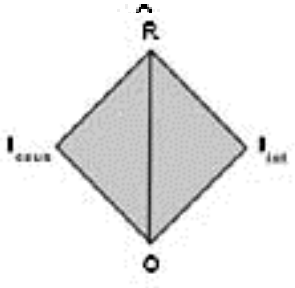
Erscheinen der Mathematik die Dinge als formale Modelle, so erscheinen der Informatik umgekehrt die Modelle als algorithmische Dinge. Aus der Welt, wie sie ist und geworden ist, geht der Gang der Abstraktion in eine Zeichenwelt, wie sie gemacht wird; von dort aber führt ein Gang der Konkretion in die Welt des Werdens zurück. Was hier geschieht, ist die Fleischwerdung, die Inkarnation, der Zeichen als algorithmische, als prozessierende, als operationale Zeichen.

Software ist Zeichen, das in Bewegung gesetzt weerden kann. Wir kommen aus der Welt zur Software durch drei reduzierende Transformationen: als erstes die *semiotische Transformation*, die Weltlichkeit auf Zeichenhaftigkeit zurückführt; als zweites die *syntaktische Transformation*, die Zeichenhaftes auf Syntaktisches reduziert, d.h. Semiosen auf Signalprozesse; als drittes die *algorithmische Transformation*, die Syntaktisches auf Berechenbares reduziert, d.h. auf berechenbare Funktionen. Nur sie gelten, nur sie existieren für den Computer.

Nur so existiert Software: sie ist Zeichen besonderer, reduzierter Art. In dieser reduzierten Art aber kann Software einer Maschine zu einem Akt der Interpretation übergeben werden. Denn Software wird von zwei Instanzen, nennen wir sie „Agenten“, gelesen: vom menschlichen Interpreten und vom computationalen Prozessor. Der Mensch liest die Software auf ihre Korrektheit, Effizienz, Benutzbarkeit hin, auf ihre allgemeinen Eigenschaften, die sich am Text des Programmes (Code) feststellen lassen. Der Prozessor liest sie mit dem Zweck ihrer Ausführung. Er stellt fest, welche Operationen auszuführen sind, um dem Code zu entsprechen.

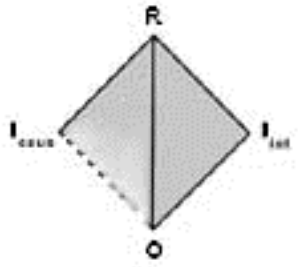
Während die menschliche Interpretation eine Interpretation im wahren Sinne des Wortes ist, ist die maschinelle Interpretation nur der Grenzfall einer Interpretation. Die eigentliche, wahre Interpretation kennt Spielräume, innerhalb deren der Interpretant gesucht wird (der Spielraum ist nicht explizit gegeben). Die verengte Quasi-Interpretation durch den Computer aber ist diejenige, deren Bedeutung eindeutig feststeht, wo keinerlei Spielraum besteht, wie er für eine Interpretation typisch wäre, wo nicht fraglich interpretiert, sondern sicherlich determiniert wird. Die eine und einzige Bedeutung, die der Code für die Maschine hat und haben soll, muss von der Maschine festgestellt werden. Sie ist – als semiotische Mascchine – so gebaut, dass sie tatsächlich interpretieren muss. Weil wir ihr aber misstrauen, ist ein Abweichen vom Weg des rechten Interpretierens der Maschine nicht erlaubt.

Determination als Grenzfall der Interpretation – das ist die Ergänzung für das algorithmische Zeichen. In ihm existieren zu einem Repräsentamen und Objekt zwei Interpretanten: einer wird vom Menschen geliefert, der intentionale Interpretant, der andere wird vom Computer determiniert, der kausale Interpretant (Determinant). Das algorithmische Zeichen besitzt eine (eingeschränkte) Doppelnatur (R = Repräsentamen, O = Objekt, I_{int} = intentionaler Interpretant, I_{caus} = kausaler Interpretant):



Dienstag 8–10 in MZH 1400 & Donnerstag 8–10 in MZH 7260 (4 SWS. 4 ECTS. VAK 03–799.20)
nake@informatik.uni-bremen.de www.agis.uni-bremen.de 0421–218 3525 linzer str. 9a 3015
www.agis.informatik.uni-bremen.de/IMZ/

Wir müssen auf Grund des determinierten Charakters des kausalen Interpretanten die Rolle des Objektes hinterfragen. Es verschwindet, wird eines mit dem Repräsentamen. Das Signal, das als das wahrnehmbare Substrat des Zeichens im Computer erscheint, fällt mit dem Bezeichneten zusammen. Die folgende Skizze zeigt das vielleicht deutlicher:



Der gestrichelte, aufgehellte Bereich im „Computer-Teil“ des algorithmischen Zeichens soll andeuten, dass es hier zum Zusammenbruch oder Abbruch der echt dreistelligen Relation kommt.