KARTEN STATT KURVEN

Digitales Entwerfen, Actor Network Theory und ein Multiagentensystem in Grasshopper

maps rather than curves – digital design, actor network theory and a multi agent system in grasshopper

KARTEN STATT KURVEN

Digitales Entwerfen, Actor Network Theory und ein Multiagentensystem in Grasshopper

Diplomarbeit zur Erlangung des akademischen Grades
Diplomingenieur
der Studienrichtung Architektur

eingereicht von
MATTHIAS STANDFEST

am Institut für Architekturtheorie, Kunst- und Kulturwissenschaften
der Technischen Universität Graz
Erzherzog Johann Universität

Betreuer: MAG.ART. DR.PHIL. DANIEL GETHMANN

Graz, im April 2011

ποταμοῖσι τοῖσιν αὐτοῖσιν ἐμβαίνουσιν ἕτερα καὶ ἕτερα ὕδατα ἐπιρρεῖ·
Man kann nicht zweimal in denselben Fluss steigen, denn andere Wasser strömen nach.

Heraklit von Ephesos, Fragmente, B 12

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

……

Matthias Standfest, Graz am 27. April 2011

KURZFASSUNG

Die Arbeit untersucht das Digitale Entwerfen aus dem Blickwinkel der Actor Network Theory. Neben der gegenseitigen Beeinflussung der Aktanten werden die Referenzketten und Übersetzungen eingehend betrachtet. Der Bogen spannt sich von Alberti bis zur CNC-Maschine, und mündet in der Gegenüberstellung von Formengrammatiken, parametrisierten Geometrien und objektorientierten Ansätzen. Als Alternative zu formalistischen und materialistischen Methoden wird das Experiment eines kollaborativen Entwurfsnetzwerkes mit digitalen Aktanten präsentiert. Es soll darlegen, wie der Entwurfsfokus von den beiden Enden der Referenzkette weg, hin zu ihrem Zentrum rücken kann und sich dadurch neue Möglichkeiten für das Digitale Entwerfen eröffnen.

ABSTRACT

The work discusses the field of digital design from the Actor Network Theory's perspective. In addition to mutual interaction of Actants, the chain of circulating reference is described in detail. Thus, the range of issues extends from Alberti, through CNC machines, to the comparison of shape grammar, parametric design and object-orientated design. As an alternative to formalistic and materialistic methods, the experiment of a collaborative design network including digital Actants is introduced. It demonstrates how to move the focus of design intention from both endings of the chain of reference to its center. Thereby the work is offering an alternative approach for digital design.

INHALTSVERZEICHNIS

VORWORT

Danksagung und Widmung

Die intensive Auseinandersetzung mit Theorie und Praxis, die zum Erstellen dieser Arbeit notwendig war, hat sehr viel Zeit in Anspruch genommen. Zeit, die zuvor keineswegs ungenützt verstrich, oder gar als zusätzlich Reserve zur Verfügung stand, sondern nach und nach von anderen überaus wichtigen Lebensbereichen abgezogen werden musste. Ich möchte deshalb gerade den Menschen in meiner Nähe für ihre Geduld danken, allen voran meinen Eltern, die mich während meiner gesamten Ausbildung moralisch und materiell unterstützt haben und Yvonne, die nicht müde wurde meine Arbeit Korrektur zu lesen. Besonderer Dank gilt daneben Daniel Gethmann für seine Betreuung bei der Erstellung dieser Diplomarbeit.

1. EINLEITUNG

Ein System interagierender Bausteine

Meine Generation ist eng mit dem Aufstieg des Personal Computer verbunden. Die Sensation, die in früher Kindheit noch der Commodore verkörperte, wurde wenig später durch das Internet über 56-kbit/s-Telefonmodems abgelöst. Heute, nur kurze Zeit danach stellen Smart Phones sowohl in Rechenleistung als auch in Verbindungsgeschwindigkeit diese damit bereits wieder veralteten Techniken in den Schatten womit sich Moore's Law zur Schaltkreisentwicklung nach wie vor bewahrheitet:

> The complexity for minimum component costs has increased at a rate of roughly a factor of two per year (...) Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000. (Moore, 1965)

Vor diesem Hintergrund wurde der Kinofilm »Tron« aus dem Jahr 1982 zum Kultfilm, und das nicht nur in der Hackerszene. Die Hauptfigur, ein Programmierer und Hacker, wird von einer Maschine eingescannt und erlangt so Zutritt in die Digitale Welt. Dort kämpft der User Seite an Seite mit den Programmen (die von Menschen dargestellt werden) darum die Herrschaft des »Master Control Program« zu beenden. In der im Film dargestellten Digitalen Welt gibt es keine Unterscheidung zwischen Programm und Mensch, so wie es in der Actor Network Theory keine Unterscheidung zwischen Mensch und Artefakt gibt. Im Dezember 2010 brachte Walt Disney eine Fortsetzung des Filmes in die Kinos, in dem diese Durchmischung noch weiter geht. Am Ende des Films rettet die menschliche, männliche Hauptfigur die weibliche Hauptfigur, ein Programm, durch die Verwandlung zum Menschen, in die reale Welt. Während im ersten Film noch der Sturz der Hierarchie in der digitalen Welt gezeigt wird, demonstriert der zweite den Untergang der Idee der »perfekten« Systeme. Tron ist eine Darstellung der Digitalen Interaktion, die

aktuellen wissenschaftlichen Betrachtungen tatsächlich näher liegt als der Verklärung ein modernes Märchen zu sein.

Viele Forschungen in der Architekturtheorie befassen sich inzwischen mit den Teilnehmern am Entwurfsprozess (De Landa, Oosterhuis, Rocker, Trummer), und der Frage welche Rolle Architekten, Menschen, Werkzeuge und Bauteile dabei einnehmen. Bruno Latour hat eine Theorie hervorgebracht, die zur Beantwortung dieser Frage neue Einsichten bringt. Mit ihrer Hilfe soll in dieser Arbeit gezeigt werden, wie Entwerfen gedacht werden kann und welche Auswirkung dieses Umdenken auf die Zusammenarbeit mit digitalen Aktanten hat. Um den Fokus der Arbeit auf die kollektive Leistung zu unterstreichen, wurde als Stilmittel großteils auf wörtliche Zitate als gleichwertige Bestandteile des Fließtextes zurückgegriffen, um all jene sichtbar zu Wort kommen zu lassen, auf deren Werk aufgebaut wird. Frei nach Deleuze verbindet dieser Text ein breites Pool an Ideen und Theorien, die alle um die Digitalisierung im Entwerfen kreisen, um daraus eine neue Entwurfsmethode zu entwickeln.

2. NETZWERKE UND AKTANTEN

Darüber wie alles zusammenhängt

DAS RHIZOM: VOM BAUM ZUM NETZWERK

Es gibt eine Parallele zwischen dem Entwerfen von Architektur und dem Schreiben von Büchern. Darum scheint es als Einstieg in diesen Text passend, zu beschreiben auf welche Art und Weise Deleuze und Guattari die drei Möglichkeiten vergleichen, in denen Bücher gedacht werden können. Man kann davon auf die Denkweise hinter kreativen Prozessen schließen, und erhält als Mehrwert eine Leseanleitung, die sich nicht nur auf dieses Projekt anwenden lässt. Kern der Überlegungen von Deleuze und Guattari, die sie am Beispiel Buch präsentieren, ist der Aufbau und die Anordnung von Konstruktionen. Sie spannen dabei den Bogen von hierarchischen bis zu rhizomatischen Ansätzen, mit der ausdrücklichen Intention ihre Ideen auch in Beziehung zu anderen setzen zu können. Folgen wir ihnen auf dem Weg zum Rhizom, treffen wir zunächst auf die erste Variante, das Wurzelbuch:

> Der erste Buchtyp ist das Wurzelbuch. Der Baum ist bereits Bild der Welt, oder besser: die Wurzel ist Bild der Baumwelt. Das ist das klassische Buch, schöne organische Innerlichkeit, jede Schicht signifikant und subjektiv. Das Buch imitiert die Welt wie die Kunst die Natur: mit seinen eigenen Verfahrensweisen, die das zum guten Ende führen, was die Natur nicht oder nicht mehr vermag. Das Gesetz des Buches ist dasjenige der Reflexion: das Eine, das zwei wird. Wie sollte das Gesetz des Buches in der Natur zu finden sein, da es doch gerade der Teilung zwischen Welt und Buch, Natur und Kunst vorausgeht? Aus Eins wird zwei: jedes mal, wenn wir dieser Formel begegnen, ob sie nun Mao als Strategie ausgegeben oder man sie so »dialektisch« wie möglich begriffen hat, haben wir es mit dem reflektiertesten, ältesten klassischen Denken zu tun, das völlig abgenutzt ist. Die Natur geht so nicht vor: dort sind Wurzeln Pfahlwurzeln mit zahlreichen Verzweigungen, seitlichen und sternförmigen, jedenfalls keine dichotomischen. Der Geist bleibt hinter der Natur zurück. Als natürliche Realität gleicht das Buch der Pfahlwurzel, mit seiner Achse und den Blättern drumherum. Seine geistige Realität hingegen, sofern Baum oder Wurzel dafür Modell stehen, bringt unaufhörlich das Gesetz hervor: aus Eins wird zwei, aus zwei wird vier … Die binäre Logik ist die geistige Realität des Wurzelbaum. (…) Insofern kann man sagen, dass dieses Denken die Viel-

> heit nie begriffen hat: es muss von einer starken, vorgängigen Einheit ausgehen, um zu zwei zu kommen, und es folgt dabei einer geistigen Methode. (Deleuze, 1977, S. 8)

Deleuze und Guattari stellen ihren Ansatz hier einer gängigen Herangehensweise gegenüber. Der lineare Aufbau und die binäre Logik der Dichotomie werden mit dem Bild des Wurzelbaums verglichen. Dabei streichen sie heraus, dass sich in der Natur im Unterschied zur Theorie die Wurzel in mehr als nur zwei Teile aufspalten kann. Die Denkweise, in der Verbindungen nur von einem zu einem anderen Punkt führen, bezeichnen sie als veraltet, während die Vielheit, die sie ins Spiel bringen, diese dichotomische Strategie ablösen soll. Das Wurzelbuch ist derweilen immer noch die gängigste Art von Buch, die wir antreffen. So, wie der Prozess hinter der Entstehung des Buches beschrieben wird, kann man auch die Gestaltung von Architektur denken. Die eine Wurzel, von der sich alle anderen in immer kleiner werdenden Teilen ableiten, kommt dem Prozess des Entwerfens sehr nahe. Aber welcher Architekt lässt sich schon auf einen einzigen Entwurfsgedanken reduzieren? Eine andere Möglichkeit wird im Beispiel der »büscheligen Wurzel« vorgestellt, in der die Hauptwurzel eine untergeordnete Rolle spielt:

> Die büschelige Wurzel oder das System der kleinen Wurzeln ist der zweite Buchtyp, den die Modernde gern für sich in Anspruch nimmt. Die Hauptwurzel ist verkümmert, ihr Ende abgestorben; und schon beginnt eine Vielheit von Nebenwurzeln wild zu wuchern. (…) Ein Text wird mit einer Vielzahl von Wurzeln, sogar Luftwurzeln zusammengeschnitten, wobei eine Vielzahl von Wurzeln, sogar Luftwurzeln entstehen (man könnte auch Stecklinge sagen), was eine supplementäre Dimension impliziert, die zu den jeweiligen Texten hinzutritt. In ihr setzt die Einheit ihre geistige Arbeit fort. Auf diese Weise kann man ein noch so zerstückeltes Werk als Gesamtwerk oder Magnum Opus hinstellen. Zwar eignen sich die meisten modernen Methoden durchaus dazu, Serien sich vermehren oder Vielheiten linienförmig wachsen zu lassen; in einer anderen Dimension aber, der des Kreises oder Zyklus, setzt sich eine totalisierende Einheit dann um so mehr durch. (…) Nietzsches Aphorismen brechen die lineare Einheit des Wissens nur auf, um im gleichen Zuge auf die zyklische Einheit der ewigen Wiederkehr zu verweisen, die im Denken als Nichtgewußtes anwesend ist. Man sieht also, daß auch ein System gebündelter Wurzeln nicht wirklich mit dem Dualismus, mit der Komplementarität von Subjekt und Objekt, Natur und Geist bricht. (…) Welch seltsame Mystifikation: das Buch wird umso totaler, je zerstückelter es ist. Das Buch als Bild der Welt – langweilig in jeder Hinsicht! Es genügt eben nicht zu rufen: Hoch lebe das Viele (multiple)! (Deleuze, 1977, S. 9)

In dieser zweiten Variante kritisieren Deleuze und Guattari, dass zwar zum einen von einem übergeordneten Strang abgelassen, aber zum anderen doch wieder eine abschlie-

ßende Kuppel über das ganze Projekt gestülpt wird. Der Versuch die Einheit hinter sich zu lassen, führt in diesem Fall zu einem Erstarken dieser. Denn genau im Fokus auf das Viele wird die Rolle der Vielheit ignoriert, und als Hilfskonstruktion wird versucht das Zerstückelte als Eigenschaft des Ganzen zu betrachten - wobei die einzelnen Fragmente nicht verbunden werden, sondern ihre Verbindungen undefiniert bleiben. Das Nebeneinanderstellen von Werken zu einem Gesamtwerk ist dafür nur ein Beispiel. Man kann den Gedankenprozess, der hier mit Büchern dargestellt wird, gleichermaßen auf die Architektur übertragen. Aus diesem Blickwinkel sieht man, wie auf mehreren Ebenen Vieles zu Einem zusammengefasst wird; ob es nun verschiedene Ideen in einem Entwurf oder verschiedene Entwürfe in einem Gesamtwerk sind. Dieses Gruppieren gaukelt eine, sich vom Rest abgrenzende Einheit vor, deren Ambivalenz durch das Anhäufen von Vielem anstelle einer strukturellen Entwicklung erreicht werden will. Das neuartige System, das den beiden vorschwebt bezeichnen sie als »Rhizom«, und mit ihm wollen sie das Viele herstellen, anstatt es zusammenzutragen.

> Als unterirdischer Sproß unterscheidet sich ein Rhizom grundsätzlich von großen und kleinen Wurzeln. Knollen und Knötchen sind Rhizome. Pflanzen mit großen oder kleinen Wurzeln können in vielerlei Hinsicht rhizomorph sein: man muß sich wirklich fragen, ob nicht das Rhizomorphe das Spezifische an der Botanik ausmacht. Auch Tiere sind es, wenn sie Meuten bilden, z.B. die Ratten. Ein Bau ist in allen seinen Funktionen rhizomorph: als Wohnung, Vorratslager, Rangiergelände, Versteck und Ruine. Das Rhizom selbst kann verschiedenste Formen annehmen, von der Verästelung und Ausbreitung nach allen Richtungen an der Oberfläche bis zur Verdichtung in Knollen und Knötchen. (Deleuze, 1977, S. 11)

Obwohl der Text in vielen Passagen Beiträge zum linguistischen Diskurs rund um Chomsky beinhaltet, kann er – und auch das ist seine gedachte Natur – ebenso in andere Bezüge gesetzt werden. Das Rhizom kann als System auch für die Architektur, und in diesem Fall im Speziellen für den Architekturentwurf, herangezogen werden. Neben dem im Zitat angeführten Ansatz Gebautes rhizomatisch zu erfassen, besteht die Chance das Entwerfen auch dementsprechend zu denken. Viele einzelne Knötchen, die alle miteinander verbunden sind, bilden unzählige Verästelungen und damit ein dichtes Netzwerk. Nicht ein Gedanke, der alles bestimmt, und nicht viele unzusammenhängende Ideen, die einfach zusammengefasst werden, sondern ein Netzwerk an Abhängigkeiten und Interaktionen bestimmen plötzlich das Schicksal des Entwerfens.

> Fassen wird die wichtigsten Merkmale eine Rhizoms zusammen: im Unterschied zu den Bäumen und ihren Wurzeln verbindet das Rhizom einen beliebigen Punkt mit einem

> anderen; jede seiner Linien verweist nicht zwangsläufig auf gleichartige Linien, sondern bringt sehr verschiedene Zeichensysteme ins Spiel und sogar nicht signifikante Zustände (etats de non-signes). Das Rhizom lässt sich weder auf das Eine noch auf das Viele zurückführen. Es ist nicht das Eine, das zwei wird, auch nicht auf das Eine, das direkt drei, vier, fünf, etc. wird. Es ist weder das Viele, das vom Einen abgeleitet wird, noch jenes Viele, zu dem das Eine hinzugefügt wird (n+1). Es besteht nicht aus Einheiten, sondern aus Dimensionen. Ohne Subjekt und Objekt bildet es lineare Vielheiten mit n Dimensionen, die auf einem Konsistenzplan ausgebreitet werden können, und von denen das Eine immer abgezogen wird. Eine Vielheit variiert ihre Dimensionen nicht, ohne sich selbst zu ändern und zu verwandeln. (Deleuze, 1977, S. 34)

Das von Deleuze und Guattari definierte Rhizom besteht im Gegensatz zu den beiden zuvor beschriebenen Systemen aus den Vernetzungen jedes Punktes mit allen anderen. Nicht nur, dass dabei die klassische Hierarchie der Wurzelbaumstruktur keine einschränkende Rolle mehr spielt, können diese Verknüpfungen auch verschiedene Dimensionen übergreifen. Dabei wird auf eine abschließende Zusammenfassung, welche weitere von außen kommende Verbindungen behindert, verzichtet, wodurch zusätzliche Überlappungen ermöglicht werden. Es rücken damit die Verbindungen neben den Vielheiten in den Mittelpunkt der Beobachtung. Elemente wie Vielheiten sind rhizomatisch untereinander verbunden. Das heißt, dass sie nicht binär hierarchisch verschachtelt sind. Es gibt damit keine Linearität mehr im Aufbau. Hinzu kommt das System der Vielheit, mit dem Deleuze und Guattari zeigen, wie die Vernetzungen mehrdimensional aussehen können:

> Eine Vielheit hat weder Subjekt noch Objekt; sie wird ausschließlich durch Determinierung, Größen und Dimensionen definiert, die nicht wachsen, ohne dass sie sich dabei gleichzeitig verändert (die Kombinationsgesetze wachsen also mit der Vielheit). Als Rhizom oder Vielheit verweisen die Fäden der Marionette nicht auf den angeblich einheitlichen Willen eines Künstlers oder Marionettenspielers, sondern auf die Vielheit seiner Nervenfasern. Diese bilden nämlich in anderen Dimensionen, die mit den Fäden der Marionetten verknüpft sind, selbst eine Marionette. (Deleuze, 1977, S. 13)

Das angeführte Modell des Marionettenspielers zeigt, wie das System der Puppenfäden auf ein System von Nervenbahnen verweist, und wie sich dieses auf die gleiche Weise wiederum von dahinter liegenden Systemen ableitet. Die Beobachtung von verschachtelten Rhizomen zeigt, wie komplex Strukturen ausgebildet sein können, egal ob bei Puppenspielern, Autoren oder Architekten. Sehen wir die tanzende Puppe als Entwurfswerkzeug, drängt sich die Frage auf wer ihre Fäden zieht. Bei manchen Werkzeugen ist man verleitet vorschnell zu urteilen, und unterschätzt was womöglich mit einer Blackbox »Bleistift« alles verbunden sein kann. Bei anderen wiederum, wie zum Beispiel jeglicher

Art von Kennzahl, lässt sich bereits erahnen, dass sich ihr Entwicklernetzwerk sehr weit erstrecken kann. Aus diesem Problem der Weitläufigkeit heraus entsteht das Konstrukt des Plateaus:

> »Jede Vielheit, die mit anderen durch an der Oberfläche verlaufende unterirdische Stängel verbunden werden kann, sodass sich ein Rhizom bildet und ausbreitet, nennen wie »Plateau«. Wir schreiben dieses Buch als Rhizom. Wir haben es aus Plateaus zusammengesetzt. Zum Spaß haben wir ihm eine zirkuläre Form gegeben. Morgens nach dem Aufstehen hat sich jeder überlegt, welchen Plateaus er folgen soll, und dann fünf Zeilen hier und zehn Zeilen dort geschrieben. Wir haben halluzinatorische Erfahrungen gemacht, haben Linien gesehen, die wie Kolonnen winziger Ameisen von einem Plateau zu einem anderen liefen. Wir haben Konvergenzkreise gezogen. Jedes Plateau kann an beliebiger Stelle gelesen und zu beliebigen anderen in Beziehung gesetzt werden« (Deleuze, 1977, S. 35)

Auf die Buchmetapher übertragen, zeigt sich der Unterschied zur weiter oben beschriebenen Möglichkeit der »büscheligen Wurzel«. Anstatt Passagen oder Werke nebeneinanderzustellen, und dadurch den Eindruck einer Gesamtheit zu provozieren, sind die Plateaus per Definition vernetzbar, und bilden aus der Summe ihrer Verbindungen das Werk als Rhizom. Der Prozess, der dabei beschrieben wird, ist besonders interessant, denn Deleuze und Guattari stellen das Werk aus seinen Bestandteilen zusammen, an welchen sie parallel arbeiten. Dabei ist ihnen bewusst, dass sie nicht alle Verbindungen der Plateaus zeichnen können, und sie lassen es frei mit welchen Rhizomen sie zukünftig vernetzt werden sollen. Es wird davon abgesehen, zentrale Plateaus zu inszenieren:

> »In zentrierten (oder auch polyzentrischen) Systemen herrschen hierarchische Kommunikation und von vornherein festgelegte Verbindungen; dagegen ist das Rhizom ein nicht zentriertes, nicht hierarchisches und nicht signifikantes System ohne General, organisierendes Gedächtnis und Zentralautomat; es ist einzig und allein durch die Zirkulation der Zustände definiert.« (Deleuze, 1977, S. 35)

Es gibt in einer derartigen Form keinen dominanten Part, es handelt sich um ein System »ohne General«. Wenn alle Vielheiten gleichwertig sind, gewinnt die »Zirkulation der Zustände«, und damit die Kommunikation an Bedeutung. Im Kontext der später noch genauer betrachteten Multi-Agenten Simulation wird hier bereits vorweggenommen, wie sich der rhizomatische Ansatz zu einem System spinnen lässt. Die präsentierte Organisationsform entsteht als Pendant zur Hierarchie, um beliebige Elemente untereinander in Verbindung setzen zu können. Dabei handelt sich nicht etwa um die bloße Umkehr von

Hierarchieverhältnissen, sondern um die völlige Negation dieser. Das Rhizom wirft die Frage auf, wie sich die Teilnehmer am architektonischen Entwerfen organisieren. Welche Netzwerke mit den Entwurfsrhizomen in Austausch stehen, und was das für das Entwerfen bedeuten kann. Die Actor Network Theory kann helfen, diese Zusammenhänge nachzuvollziehen und zu verstehen.

DAS ZIRKULIEREN: REFERENZEN UND ÜBERSETZUNGEN

Im Aufsatz zur »zirkulierenden Referenz« beschreibt Latour die Übersetzungskette von Form zu Materie und umgekehrt. Dabei geht es um die schrittweise Übertragung der Natur in ihre Abbildungen, die unter anderem in Form von Schriftstücken, Diagrammen, Modellen und Kennzahlen auftreten können. Damit können die Stellung des Labors und die Konstruktion von Fakten hinterleuchtet werden. Sein Beispiel ist dafür die Wissensproduktion einer Amazonasexpedition. Um die wissenschaftliche Praxis verstehen zu können, hat er eine Untersuchung beobachtet, die klären sollte, woher der natürliche Wechsel im Bewuchs am besuchten Ort stammt und in welche Richtung er sich entwickeln wird. Während sich zur einen Seite hin die Savanne ausbreitet, liegt auf der anderen Dickicht und Wald. Ob der Urwald nun vordringen oder zurückweichen wird, ist dabei für die Untersuchung zur zirkulierenden Referenz nicht weiter von Belang. Im Gegensatz zur Entwicklung von der Vermessung hin zur grafischen Darstellung. Als ersten Schritt beobachtet Latour das Abstecken der Flächen und das Nummerieren der dort existenten Fauna:

> Sie hat also in regelmäßigen Abständen eines dieser kleinen Schilder angebracht und so die paar Hektar ihres Untersuchungsfeldes mit einem Netz von kartesischen Koordinaten überzogen. Diese Nummern werden ihr später helfen, die Unterschiede im Wachstum und im Vorkommen der Arten festzuhalten. Jede Pflanze besitzt, was man eine Referenz nennt – sowohl eine geometrische (durch Zuschreibung von Koordinaten) als auch eine regelmäßige (durch Zuschreibung einer Nummer). (Latour B. , 2002, S. 44f)

Der erste Übersetzungsschritt ist damit das Umwandeln von Natur zu Labor, indem sie vorbereitet wird, vernetzt und referenziert zu werden. Er zeigt dabei, wie das Zuschreiben einer Nummer die Pflanze zum einen Ende einer Verbindung macht. Am anderen Ende steht dabei die Ziffer als Identifikation, die wiederum für eine Verbindung offen steht.

Über verschiedene Methoden lassen sich die Übersetzungen verschieden ausformen, als Beispiele führt er dafür eine Koordinatenliste und eine durchlaufende Nummer an. Generell zeigt sich die Koordinate in den Beobachtungen Latours als wichtiges Mittel, als Referenz um die Welt übersichtlicher zu machen. Und gerade sie ist es auch, die für die Arbeit der Architekten ein unverzichtbares Instrument darstellt. Für manche Untersuchungen ist es aber notwendig, dass ein Vertreter einer Art auf seine Genossen referenziert, und so werden einige der nummerierten Pflanzen im weiteren Verlauf gesammelt und genauer untersucht:

> An diesem Strauß, den Edileusa gepflückt hat, können wir zwei Eigenarten der Referenz erkennen: einerseits eine gewisse Ökonomie, Induktion, Verkürzung und Engführung, die es ihr erlauben, einen Grashalm als den einzigen Repräsentanten von Tausenden von Gräsern anzusehen; und andererseits die Konservierung eines Spezimens, auf das sie später garantiert zurückgreifen kann, wenn ihr Zweifel kommen oder wenn andere Kollegen ihre Ansichten in Zweifel ziehen. So wie Fußnoten in wissenschaftlichen Arbeiten als Referenzen (…) so wird auch die Handvoll Spezimen eine Garantie für den Text ihres Arbeitsberichtes sein. (Latour B. , 2002, S. 46f)

Betrachten wir nun stellvertretend für den Prozess der Übersetzung den hier beschriebenen Anfang der Kette. Es wird nicht mehr davon ausgegangen, dass eine Grafik oder Ähnliches direkt auf die Welt verweist, sondern viele Zwischenschritte dafür notwendig sind. Die auf der Pflanze angebrachte Nummer ist so ein kleiner Schritt, und genau so die gesammelten Spezimen. Schlussendlich wird ein Element auf ein Element verweisen, das wiederum über einen Umweg auf die Pflanze im Urwald verweist. Das Besondere dabei ist, dass mit jedem Zwischenschritt weiter weg von der Natur hin zu ihrer Darstellung gegangen wird. Jede Abstraktionsstufe hat dabei eine eigene Qualität, die es erlaubt die jeweilige Referenz in einen andern Kontext zu setzen, um so neue Erkenntnisse gewinnen zu können. Unabhängig von Ort und Zeit der gesammelten Proben, können diese am Arbeitstisch nebeneinandergelegt und wiederholt arrangiert werden, bis sie neue Muster erkennen lassen, die sonst nicht ersichtlich gewesen wären.

> Indem man den Urwald verliert, gewinnt man das Wissen über ihn. In einem wunderbaren Widerspruch erfasst das Wort »Übersicht« genau die beiden Bedeutungen dieser Beherrschung durch den Blick, da es gleichzeitig bedeutet, etwas zu überblicken und etwas zu übersehen, d. h. zu ignorieren. (Latour B. , 2002, S. 51)

Das ist der Sprung, den es zu verstehen gilt. Ist etwas referenziert, lässt sich an seiner Stelle mit der Referenz selbst arbeiten, was Vieles vereinfacht oder überhaupt erst ermöglicht.

Die Referenz ist aber nicht das Original, so wie der Grashalm im Labor nicht die Steppe am Waldrand ist. Ob es sich nun womöglich um statistische Ausreißer handelt, Transportschäden, oder irgendeine andere Form von Manipulation des Referenzierten, es gibt einen Unterschied zwischen Beiden. Die Differenz, die sich am klarsten darstellt, ist sicher der andere Ort, an dem sich die Beiden befinden. Der Grashalm ist in der Steppe Teil eines anderen Systems als im Labor. Während er im einen System neben vielen anderen ähnlichen Vertreten seiner Gattung sprießt, wird er im anderen an einer Stelle angeordnet um ein schlüssiges Bild zu schaffen. Er ist keiner Witterung mehr ausgesetzt, und muss auch keinen Schädlingsbefall mehr fürchten. Es sind zwei Systeme, die unterschiedliche Beobachtungen ermöglichen, wodurch die einen bewusst für die anderen eingetauscht werden. Manche Methoden, die zur Analyse oder zum Arrangieren der Referenzen im Labor eingesetzt werden, haben Einfluss auf die Auswahl des Referenzierten in der Natur:

> Die mit Bodenproben gefüllten Schachteln haben Zeichencharakter angenommen. Aber man weiß, dass die leeren Gevierte einer solchen Tafel (…) immer die wichtigsten sind. Die leeren Schachteln zeigen uns, was noch fehlt. Sie geben uns einen Vorgeschmack der Arbeit von morgen. Dank ihr sehen wir die Leerstellen unseres Protokolls. In den Worten Renés: »Der Pedokomparator *sagt* uns, ob man mit dem Querschnitt fertig ist.« (…) Der zweite Vorteil des Pedokomparator ist: ein *pattern* entsteht, und wie bei den Funden Edileusas wäre es auch hier wieder befremdlich, wenn es nicht so wäre. Auf den neuen Zugriff, den eine neue Übersetzung und ein neuer Transport nach sich ziehen, folgt beinahe immer eine Erfindung. (Latour B. , 2002, S. 65ff)

Ähnlich wie zuvor bei den Pflanzen hilft die Übersetzung auch beim Erkennen von Mustern. Die Box mit den Schachteln der Bodenproben, die analog zu ihrem Fundort in einer tabellarischen Form angeordnet werden, ist ein Werkzeug zum leichteren Finden dieser. Das aus der Anordnung hervorgehende Muster kann dabei in eine Grafik weiter übersetzt werden, oder aber aufzeigen wo der Datenbestand lückenhaft ist, und weitere Probebohrungen mitverursachen. Je nach Bedeutungsebene werden so auch andere Werkzeuge zur Analyse herangezogen. In der Ebene des Diagrammes werden wieder neue Verknüpfungen schlagend, und auf einen Teil der alten wird verzichtet. Durch eine neue Anordnung wird neue Information erzeugt, und das ohne auf den Urwald direkt zu referenzieren, sondern statt dessen auf die Proben im Pedokomparator. Bis zum Schluss ein Bericht entsteht, wird eine Kette von Vermittlungen und Übersetzungen gesponnen, in der keines der Glieder verzichtbar ist.

> Wichtig ist, dass diese Kette *reversibel* bleibt. Die Nachvollziehbarkeit der Schritte muss es im Prinzip erlauben, sie in beide Richtungen auszuführen. Unterbricht man sie an irgendeinem Punkt, so ist auch der Transport, die Produktion, die Konstruktion, gewissermaßen die Leitfähigkeit des Wahren unterbrochen. *Die Referenz ist eine Eigenschaft der Kette in ihrer Gesamtheit* und nicht der *adaequatio rei et intellectus*. Die Wahrheit *zirkuliert* in ihr wie die Elektrizität entlang eines Drahtes, und zwar so lange, wie er nicht zerschnitten wird.
> Noch eine Eigenschaft kommt zum Vorschein, wenn man beide Schemata überlagert: Die Kette hat weder auf der einen noch auf der anderen Seite ein Ende. Während im alten Modell (…) sowohl die Welt als auch die Sprache geschlossene Ensembles blieben, die beide in sich selbst zurückliefen, kann man im neuen Modell im Gegenteil die Kette endlos verlängern, indem man ihr an beiden Enden neue Glieder hinzufügt. Aber man darf die Kette nicht zerschneiden und auch keine Sequenz überspringen – obwohl man sie in einer einzigen *black box* zusammenfassen kann. (Latour B. , 2002, S. 85f)

Die Übersetzung der Materie in die darauf verweisende Form beinhaltet immer einen Bruch, Latour spricht dabei vom »gap«. Diese Brüche besitzen das Potenzial Neues zu finden, und haben genau aus diesem Grund eine große Auswirkung auf die Architektur. Verlassen wir nämlich den wissenschaftstheoretischen Standpunkt, und ersetzen die Wahrheit durch die Idee, können wir aus einer Analyse des Forschens eine des Entwerfens ableiten. Der Weg von der Skizze zum Gebauten ist nur das Abschreiten des gleichen Pfades in anderer Richtung. Aus dem konstruierten Endpunkt Urwald wird in diesem Bild das Gebäude (oder bei manchen Architekten die fotografische Abbildung davon), und auf der gegenüberliegenden Seite der Kette steht nicht mehr ein Aufsatz, der die »Wahrheit« beschreibt, sondern der Entwurfsgedanke, der in Gebautes übersetzt werden soll.

> Man braucht nur kurz nachzudenken, bevor man eingestehen muss, dass der euklidische Raum der Raum ist, in dem Gebäude auf Papier *gezeichnet* werden, nicht jedoch das Umfeld ist, in dem Gebäude *entstehen* – und noch weniger die Welt, in der sie *bewohnt* werden. (Latour B. u., 2008, S. 81)

Der Ablauf scheint geläufig: Im ersten Schritt wird der Entwurf skizziert. Es entsteht in einem kompakten Maßstab ohne hohen Detaillierungsgrad eine grobe Skizze der Idee. Der Entwurf wird hierbei bereits angepasst, denn dem geschulten Auge fallen beim Zeichnen die einen oder anderen Unstimmigkeiten negativ auf. Dem folgt ein erstes Entwurfsmodell, welches womöglich bereits in ein Umgebungsmodell gestellt werden kann. Hat man erst die Form in Materie gegossen, wird man einen Unterschied zur Vorstellung entdecken können und versuchen das Modell umzubauen. Nach dem Vorbild

des Pedokomparators hat das wiederum eine Auswirkung auf die Ebene der grafischen Darstellung. Es entsteht ein Hin und Her zwischen der Skizze und dem referenzierten Modell. Der Bruch, der dabei jeweils entsteht, nötigt den Entwerfer zum Reagieren. Es gibt dabei nicht nur den Sprung von Modell zu Zeichnung und zurück. Albena Yaneva hat im Büro von Rem Koolhaas, dem *Office for Metropolitan Architecture* in Rotterdam die Sprünge zwischen den verschiedenen Maßstäben, in denen entworfen wird, untersucht. Ihre Beobachtungen spiegeln wieder, was hier bereits umrissen wurde:

> Moreover, as compared with many other architectural firms in which models are built only at the final stage of a project, OMA fabricates models at every step of the design process, along with two-dimensional representations. These models are important tools for shared cognition: architects think of building by modelling, by cutting foam and paper and using various scoping techniques. It is not a free intuitive creation of a building shape generated 'out of the blue'. The first small models of the Whitney building are produced according to few important 'constraints'. These constraints are mainly negative – 'not to exceed the zoning envelope', 'not to demolish the brownstones', 'not to damage the adjacent buildings' – in the way they place limits on the process of experimentation that we can witness in the office of Rem Koolhaas. (Yaneva, December 2005, S. 872)

Natürlich hat jedes Büro seine eigene Herangehensweise an das Entwerfen, der Einsatz von Modellen hat nicht überall den gleichen Stellenwert. Viele arbeiten mit einer großen Menge an grafischen Darstellungen wie Skizzen, Plänen und Diagrammen. Zwischen diesen wechselt im Prozess der Fokus ihrer Aufmerksamkeit, und sie werden parallel bearbeitet. Die Referenzen reihen sich nicht mehr nur nach Abstrahierungsgrad, die Ketten spalten sich auch in verschiedene Maßstäbe. Immer wieder kann man auch das Komprimieren des ganzen Entwurfs in eine Kennzahl feststellen, um ihn leichter vergleichbar zu machen. Niemand würde mehr davon ausgehen, dass eine Zahl ein Gebäude darstellt, aber sie vertritt es sehr wohl in einem definierten Rahmen. Die Darstellung durch Ziffern statt Graphen produziert unabhängig vom Maßstab dieselben Resultate, vor allem dadurch, dass beim Zeichnen am Computer Ableseungenauigkeiten nur mehr eine verschwindend geringe Bedeutung haben. Jede Darstellung und jede Abstrahierungsebene ermöglicht einen anderen Zugang und eine andere Vernetzung und ist dabei selbst einzigartig in der Übersetzung verflochten.

> Alle diese widersprüchlichen Eigenschaften – widersprüchlich zumindest in den Augen der Philosophie – laden das Diagramm mit der Wirklichkeit auf. Es ist nicht realistisch, und es ist nicht ähnlich. Es vollbringt etwas anderes, *Besseres*. Es *vertritt die Ausgangssituation*, mit der es durch eine Serie von Transformationen verbunden bleibt und deren

> Spur wir zurückverfolgen können dank dem Protokollbuch, den Schildern, dem Pedokomparator, den Mappen, den Absteckungen und dem feinen Netz, das der Geländefaden gesponnen hat. Dennoch können wir das Diagramm nicht aus der Gesamtheit dieser Transformation extrahieren. Isoliert sagt es überhaupt nichts mehr. Es ersetzt, ohne etwas zu ersetzen. Es fasst zusammen, ohne das ersetzen zu können, was es zusammenfasst. Ein seltsames transversales Objekt, ein Ausrichtungsoperator, der nur insoweit wahrheitsgetreu ist, als er den Übergang zwischen dem erlaubt, was vorangeht, und dem, was folgt. (Latour B. , 2002, S. 82)

Obwohl der Architekt nach wie vor seine Vorstellungen nur in den seltensten Fällen numerisch formuliert, hat die Benützung von Rechnern (das Wort »rechnen« lässt bereits auf die Art der Datenverarbeitung schließen) die Zahl als Darstellungsform maßgeblich gefördert. Auf Knopfdruck können digitale Zeichnungen in eine Menge verschiedener Kennwerte übersetzt werden, und lassen so Optimierungspotenziale erkennen, die in anderen Instanzen umgesetzt werden können. Die Zahl erhält ihre Bedeutung nur aus ihrem Kontext: Ist diese Zahl günstig um den Wert einer anderen, durch diverse Netze verbundene Zahl positiv zu beeinflussen? Kann der Wert näher an einen Idealfall getrimmt werden?
Was in der Analyse funktioniert, kann in der Zirkulierenden Referenz auch umgekehrt gelingen. So wie sich der Entwurf durch Zahlen darstellen lässt, wird er auch von ihnen beeinflusst. Dabei kann vermutlich keine den Einfluss der durch Ziffern ausgedrückten Kosten erreichen. Denn wie geografische Bedingungen an einem Ende der Kette, existieren am anderen Auflagen in Form von Bruttogeschossfläche, Baukosten oder Besonnungszeiten, um nur einige zu nennen. Daneben gibt es aber auch andere Parameter, wie lichte Durchgangsbreiten, die einen direkteren Bezug zur Plandarstellung aufweisen. Der Parameter stellt sich in dieser Form nicht als Eigenschaft des ganzen Gebäudes, sondern nur als ein Teil davon dar. Das Maß als Referenz bildet eine weitere abstraktere, aber gerade von Normen und anderen Richtlinien oft gewählte Form des Entwerfens. Das Springen zwischen all diesen Formen, die das ungebaute Gebäude annehmen kann, ist essenziell für seinen Entwicklungsprozess.

> The ‚jump' accelerates the visualization process. It allows one to gain better visibility of internal displays and spatial arrangements, to inspect them with precision and foresight. In the 'jump', materials and dispositions can 'talk back' to architects. When they step back, they reinterpret what they see or reframe the problem to be solved. By doing so, architects constantly go down to refine the small model instead of simply progressing slowly towards a bigger and bigger version of the building, increasing precision, and attaining more and more knowledge about it. (Yaneva, December 2005, S. 882f)

Wenn man diverse Schritte zu einer *Blackbox*[1] zusammenfasst, sind die in ihr stattfindenden Brüche vom Architekten nicht mehr erkennbar, und eine Auswirkung auf den Entwurfsprozess wird wahrscheinlich. Umso mehr Schritte in einem digitalen Netzwerk gebündelt werden, auf das der Architekt nur minimalen Einfluss hat, desto mehr büßt er ihn auch ein. Der Architekt verliert sukzessive Kontakt zum Entwurfsobjekt in all seinen Referenzen, und damit die Kenntnis über ihn. Gerade die Übersetzung vom Parameter in die Form hat als eine derartige Vereinfachung der Referenzkette den Architekten stark zurückgedrängt. Im gleichen Atemzug eignet sich der Architekt immer mehr Kenntnisse an selbst Parameter zu produzieren. In der Wissenschaft gibt es keine direkte Übertragung von Materie zu Form, sondern es existiert in der Übersetzung immer ein Bruch. Es wird Zeit, dass dem Verhältnis von Parameter zu Form im Entwerfen die gleiche Spannung zugestanden wird.

DIE AKTANTEN: EINE ALTERNATIVE ZUR DICHOTOMIE

Hauptbestandteil der ANT ist der Aktant und seine semiotische Definition, die weder eine menschliche Intention noch andere menschliche Eigenschaften impliziert. Unter der Voraussetzung, dass es zu einer Handlung beiträgt kann alles, egal ob Mensch oder Artefakt, ein Aktant sein (Vgl. Latour B. , 1996). Alles ist in diesem Modell ein Aktant, egal ob lebendig oder leblos. Die Objekt-Subjekt Dichotomie wird mit diesem Ansatz obsolet. Hier findet sich die Neuerung zu anthropozentrischen Modellen, die davon ausgehen, dass Handeln einen menschlichen Ursprung hat. Werkzeuge spielen dort eine untergeordnete Rolle, der Mensch beherrscht das Artefakt. Und genau so zu materialistischen Modellen[2], die beschreiben wie jedem Werkzeug eine Bedienungsanleitung immanent ist, die das menschliche Handeln diktiert. In diesem Spannungsfeld zwischen beherrschen und beherrscht werden liefert die ANT ein neue, dritte, Option: Menschlicher und nichtmenschlicher Aktant sind Teil eines zusammengesetzten Aktanten, dessen Ziel sich aus den Zielen seiner Bestandteile summiert. Weder werden damit Menschen verdinglicht noch Artefakte vermenschlicht, vielmehr sind beide nun Varianten einer neuen Art.

1 Als Black Box wird eine Zusammenstellung bezeichnet, deren Aufbau und Einzelteile nicht bekannt sind.

2 z. B. die Herrschaft der Technik bei Heidegger (Vgl. Heidegger, [1962] 2007, S. 26)

> »Feuerwaffen töten Menschen« ist eine Losung jener Leute, die sich in den USA für eine Einschränkung des freien Waffenverkaufs einsetzen. Darauf kontert die *National Rifle Association* (NRA) mit dem Slogan: »Es sind die Menschen, die töten, nicht die Waffen.« Die erste Losung ist materialistisch: Die Waffe tut selbst etwas aufgrund ihrer *materiellen* Bestandteile, die sich nicht auf soziale Eigenschaften des Schützen reduzieren lassen. Die Waffe macht auch einen braven Mann und gesetzestreuen Bürger gefährlich. Dagegen bietet die NRA einen *soziologischen* und politisch eher für die Linke typischen Zugang an (…) Für sich genommen oder aufgrund ihrer materiellen Bestandteile tut die Waffe nichts. Sie ist nur ein Werkzeug, ein Medium, ein ganz neutraler Träger für einen dahinterstehenden menschlichen Willen. Ist der Waffenbesitzer ein guter Bürger, so wird er die Waffe auch nur wohlüberlegt einsetzen und nur im äußersten Notfall jemanden töten. Wenn er aber ein Gangster oder Irrer ist, dann ist die Waffe, ohne dass sich an ihr etwas ändert, einfach nur ein effizienteres Tötungsmittel für eine Tat, die ohnehin begangen worden wäre. (Latour B. , Die Hoffnung der Pandora, 2002, S. 214)

Das Beispiel der Schusswaffe dient stellvertretend für alle anderen Techniken und Werkzeuge zum Aufzeigen eines komplizierten Verhältnisses zwischen zwei Aktanten. Übertragen auf den Kontext der Kunst ist die Beziehung des Schützen zur Waffe vergleichbar mit der von Künstler und perspektivischer Darstellung, oder in der Architektur dementsprechend zum Beispiel die Verbindung von Architekt und Zeichenprogramm. Wer bestimmt aber das Handeln, die Waffe oder der Mensch? Ist der Architekt Sklave seiner Werkzeuge, oder ist es egal womit er arbeitet? Beide Varianten vereinfachen das Verfahren der Zielfindung des zusammengesetzten Aktanten und verschleiern auf diese Weise eine dritte Möglichkeit, die sich zwischen sie drängt: Weder Waffe noch Mensch setzen ihr Ziel durch. Die Waffe, der verschiedene Verhaltensskripte zugeschrieben sind, ermöglicht dem Menschen seine Zielsetzung zu modifizieren. Beide zusammen erreichen als Schütze ein neues Ziel. Der Schütze, zusammengesetzt aus Waffe und Mensch, ist dabei eine neue dritte Rolle. Für den Architekten bedeutet das, dass er durch die von ihm genutzten Werkzeuge in seinem Handeln beeinflusst wird. Sie ermöglichen ihm auf einen Katalog von Handlungsweisen zurückzugreifen, und schränken ihn gleichzeitig auf diese ein. Wie wird aber das gemeinsame Ziel von Mensch und Werkzeug ermittelt?

> Die Übersetzung vollzieht sich ganz symmetrisch. Mit der Waffe in der Hand bist du jemand anderes, und auch die Waffe in deiner Hand nicht mehr dieselbe. Du bist ein anderes Subjekt, weil du die Waffe hältst; die Waffe ist ein anderes Objekt, weil sie eine Beziehung zu dir unterhält. Nicht länger handelt es sich um die Waffe-im-Arsenal oder die Waffe-in-der-Schublade oder die Waffe-in-der-Tasche, nein, jetzt ist es die Waffe-in-deiner-Hand, gerichtet auf jemanden, der um sein Leben schreit. Was für das Subjekt gilt, gilt auch für das Objekt, was für den Schützen, auch für die zielende Waffe. Der gute Bürger wird zum Schurken, der Gangster zum Killer, der stumme Revolver zu einer

> abgefeuerten Waffe, der neue Revolver zum gebrauchten, das Sportgerät zum Tötungsinstrument. (…) Wenn wir die Waffe und den Bürger dagegen als Proposition begreifen, bemerken wir, dass weder Subjekt noch Objekt (noch ihre Ziele) festgelegt sind. Wenn Propositionen artikuliert werden, verbinden sie sich zu einer neuen Proposition. Sie werden »jemand« oder »etwas« anderes. (…) Wir müssen lernen, Handlungen sehr viel mehr Agenten zuzuschreiben – auf sie zu verteilen -, als es in materialistischen oder soziologischen Erklärungen annehmbar ist. Außer menschlichen gibt es nichtmenschliche Agenten (wie hier die Waffe), und beide können Ziele haben (Ingenieure sprechen eher von Funktionen). (Latour B. , 2002, S. 218f)

Latour bedient sich eines Geniestreiches: Er lässt die Objekt-Subjekt Dichotomie außen vor, und definiert Mensch wie Artefakt als Aktant. Die Handlung wird dabei von beiden zusammen gestaltet, und unterscheidet sich von den jeweiligen Individualzielen. Nehmen wir wieder an, der Mensch hat das Ziel sich zu verteidigen, und nutzt um diese Verteidigung effizienter zu gestalten das Werkzeug Pistole. Dass er sich mit einer Pistole anders zur Wehr setzen kann als zum Beispiel mit einem Spaten, ändert nichts an seiner Intention sich zu verteidigen, wohl aber am Verteidigen an sich. Der Ausgang eines Pistolenschusses ist mit den Folgen eines Handgemenges nicht zu vergleichen, das Ziel des Menschen driftet ab und zu dem des Schützen. Effektives Verteidigen wird im Ernstfall zum »das Ziel treffen«. Dieses der Pistole inskribierte Verhalten lässt sich teilweise auf Ziele zurückführen, die ihre Konstrukteure und Erschaffer hatten, und die aus einer langen Entwicklung letaler Waffen hervorgehen. Das Optimieren der Austrittsgeschwindigkeit, verbesserte Ergonomie und die geringe Anfälligkeit für Fehlfunktionen wie Ladehemmungen, vereinfachen das Zufügen von todbringenden Verletzungen. Addiert man nun das Ziel des menschlichen Aktanten »zur Wehr setzen« mit dem Ziel des Aktanten Waffe »Schuss abfeuern«, ergibt sich für den aus beiden zusammengesetzten Aktanten Schütze das Ziel seinen Angreifer mit einem Schuss außer Gefecht zu setzen. Dabei entscheidet nicht nur das Geschick über den Ausgang der Situation, sondern auch ob der menschliche Aktant die anderen möglichen Funktionen des Waffenaktanten zu nutzen weiß. Denn neben dem Konstruktionsziel, hat die Waffe inzwischen auch eine Abschreckungsfunktion zugewiesen bekommen, oder sie könnte als Wurfgegenstand missbraucht werden. Es bleibt im zusammengesetzten Verhalten der beiden ein breiter Spielraum an möglichen resultierenden Zielen, aber keines davon wird exakt einem der Individualziele entsprechen.

Handeln zwei oder mehrere Aktanten kooperativ, so beeinflussen sich ihre Ziele gegenseitig. Die Abweichung von ihrem ursprünglichen Ziel bezeichnet man als »Drift«. Weil es äußerst selten ist, dass menschliche Aktanten alleine und ohne Hilfsmittel handeln, ist die

Drift nahezu Regelfall. Will man diese Differenz vom ursprünglichen zum resultierenden Ziel besser verstehen, lohnt es sich zuerst die Teilziele zu analysieren. Der menschliche Aktant hat eine unscharf formulierte Intention, er weiß was er erreichen will und was nicht, und nimmt dafür gewisse Kompromisse in Kauf. Übertragen wir dieses Verhalten auf den Architekten und seine »Hilfsmittel«, können wir eine unglaubliche Menge dieser Interaktionen beobachten: Die zu verwendenden Bauelemente beeinflussen das Entwerfen gleich wie hinzugezogene Fachplaner und die genutzten Entwurfswerkzeuge. Bei alledem ist es egal, in welcher Abstraktionsebene man gerade arbeitet. Man kann rechnerische oder grafische Verfahren zum Optimieren diverser Kennzahlen heranziehen, oder beim Zeichnen zwischen Bleistift und CAD wählen. Die gewählten Mittel beeinflussen das Handeln und verursachen ein Abdriften der Intention des Architekten. Ein gutes Beispiel im gegenwärtigen Architekturschaffen findet man im Architekt-Computer-Software Entwurfsaktanten. Der Einfluss den die CAD-Software auf den Entwurf hat, wurde durch Arbeit Greg Lynns deutlich.

> Wenn der Computer für die Architektur überhaupt von Nutzen sein soll (abgesehen von Zeichnung, Buchhaltung, Management), dann muss die durch den Computer ermöglichte grafische Virtuosität auf besonders überzeugende Weise nutzbar gemacht werden, um die unvermeidliche (alte) Frage zu unterbinden oder zu vereiteln: wie real das Projekt denn wirklich sei. Aber hier bekommen wir es auf quälende Weise mit den Folgen unserer Weigerung zu tun, diese Projekte im Hinblick auf ihre Besonderheit zu betrachten. Denn am erstaunlichsten und letzten Endes am überzeugendsten ist an diesen Projekten, dass nichts jemals zuvor so ausgesehen hat. Die Projekte weisen ein Computer-Aussehen auf, das sich völlig von der *AutoCad*-Architektur unterscheidet. Die metallischen sperrigen Schalen (Lynn) und die schwere Masse geodätischer Struktur (Reiser/Umemoto) scheinen, was Aussehen angeht, ihren eigenen Anspruch zu etablieren. (Hays, Ingraham, & Kennedy, 2003, S. 150)

Dabei geht es nicht nur um die Schnittstelle Mensch-Computer, die nur eine stark eingeschränkte Interaktion zulässt. Vielmehr geht es darum, wie unterschiedlich mit verschiedener Software entworfen wird. Frank O. Gehry hat zusammen mit der, aus dem Flugzeugbau stammenden, CAD-Software CATIA das Guggenheim-Museum in Bilbao entworfen, und Greg Lynn schuf zusammen mit der Animationssoftware Maya sein Embryological House. Während das eine Programm zum Konstruieren aerodynamischer Geometrien entwickelt wurde, bietet das andere die Werkzeugpalette um zwischen verschiedenen Formen Übergangszenarien zu interpolieren. Die Liste von Gebäuden, die massiv durch die verwendeten Zeichenprogramme beeinflusst wurden, kann mit einer

ständig wachsende Anzahl von Beispielen erweitert werden. Und die vielen Vorteile, die CAD Software mit sich bringt, lassen kaum vermuten, dass dieser Boom nachlassen wird. Um die Auswirkungen seines Handelns bewusster gestalten zu können, ist es hilfreich über die Ziele der im Entwerfen zur Auswahl stehenden Aktanten möglichst umfassend Bescheid zu wissen. Man kann die Entscheidungen welche Aktanten man hinzuzieht kritisch hinterfragen, und damit Richtungen für den Entwurf vorgeben, oder man stellt betriebswirtschaftliche Aspekte über gestalterische. Oft kann beobachtet werden, dass das Werkzeug der Wahl jenes ist, in dessen Umgang der Zeichner am geübtesten ist. Durch eine festgelegte Werkzeugwahl verfällt man immer einer ähnlichen Drift, und einstudierte Abläufe verstärken diese Tendenz zusätzlich. Als Know-how oder Stil umschrieben, liegt darin ein Grund für gewisse Lösungs- und Formtendenzen von Büros. Ähnliche Teilnehmer führen zu ähnlichen Lösungswegen und die führen zu ähnlichen Ergebnissen. Die Ziele, die den Softwareaktanten dabei innewohnen hängen von dessen Verbindungen ab. Massiv werden diese Eigenschaften durch seine Entwickler sowie deren Umfeldern beeinflusst.

> Bestimme ich dich nach dem, was du hast (die Waffe) und nach der Reihe von Verbindungen, in die du dich begibst, wenn du gebrauchst, was du hast (wenn du die Waffe abfeuerst), dann wirst du von der Waffe verändert – mehr oder weniger, das hängt vom Gewicht deiner anderen Verbindungen ab. (Latour B. , 2002, S. 218)

Es gibt in diesem Modell keinen Unterschied zwischen Objekt und Subjekt, was für den Menschen gilt, gilt somit auch für den nichtmenschlichen Aktanten. Wie stark dieser von seinem »Ziel« abgebracht werden kann, hängt von seinen und den Vernetzungen der anderen Mitglieder des Kollektives[3] ab. Nimmt man das Pistolen Beispiel wieder zur Hand, so ist die Waffe auch mit Volkswirtschaft und in Politik verstrickt. Sie ist Spielball und Projektionsfläche von Konzernpolitik und Lobbying, ihre Funktion hat Auswirkung auf zukünftige technische Entwicklungen und die verwendeten Materialien besitzen als Rohstoffe gewissen Wert. Die verschiedensten Gebiete greifen ineinander. Die Methode, eine Aufgabe oder Funktion auf einen Aktanten zu übertragen, wie es bei jedem Werkzeug der Fall ist, wird auch als »Delegieren« bezeichnet. Nicht genug damit, dass durch das Delegieren Arbeit abgegeben werden kann, hebt es neben der fachlichen und räumlichen auch noch die zeitliche Schranke aus:

3 Kollektiv, die Gemeinschaft nichtmenschlicher und menschlicher Aktanten (Vgl.Latour B. , 2008, S. 11)

> Von diesem Umweg der Delegation ist die gesamte Technikphilosophie absorbiert gewesen. Man denke nur an die Vorstellung von Technik als geronnener Arbeit. (…) Durch solche Umwege wird die normale Ordnung von Raum und Zeit auf den Kopf gestellt – in einer Minute kann ich Kräfte mobilisieren, die schon vor Jahrmillionen an weit entfernten Orten in Bewegung gesetzt worden sind. (…) Die relative Ordnung von Anwesenheit und Abwesenheit wird neu verteilt – unentwegt begegnen wir Hunderten, ja Tausenden von abwesenden Herstellern, die weit entfernt in Raum und Zeit und dennoch gleichzeitig aktiv und anwesend sind. (Latour B. , 2002, S. 230f)

Die Waffe wurde hergestellt. Sie wurde entworfen, konstruiert, gebaut und getestet, und ist in dieser Zeit mit einem ganzen Kollektiv an Herstellern Bindungen eingegangen. Mit dem Griff zu ihr werden diese auch zu Verbindungen des Schützen. Auf die gleiche Weise entwerfen die Programmierer von CAD Software, obwohl sie nicht direkt anwesend sind, an Bauwerken mit. Sie haben Einfluss auf das entstehende Gebäude, und auf alle die in den Prozess der Entstehung eingebunden sind. Verschiedene Programme werden für verschiedene Einsatzgebiete entwickelt, sei es nun Auto-, Flugzeug- oder Maschinenkonstruktion, Modellierung für Animationen oder andere Spezialanwendungen. In jedem dieser Felder haben sich eigene Arbeitsabläufe entwickelt, und darauf abgestimmt haben die jeweiligen Programme andere Funktionsumfänge und Benutzeroberflächen. In ihrer Anwendung ermöglichen sie den Zeichnern zum einen verschiedene Wege zum Erreichen ihrer Ziele, man kann sie effizient oder umständlich gebrauchen, und zum anderen verschiedene Funktionen auf ihre Auswirkungen zu testen. Die Funktion mit der getätigte Arbeitsschritte rückgängig gemacht werden können hat dieses Testen ungemein erleichtert, und »Trial and Error«[4] hat als Methode die Software noch stärker in den Entwurfsprozess miteinbezogen.

Die Befehle, die sich hinter den Schaltflächen der Software verstecken, verweisen auf kleine Programmteile die spezielle Anweisungen unter genau definierten Bedingungen ausführen. Auf ein paar Zeilen Code definieren Entwickler und Programmierer welche Parameter berücksichtigt werden, und welche Ergebnisse produziert werden können. Können zwei-, drei- oder vierdimensionale[5] Koordinaten eingegeben werden, um eine Gerade, einen Strahl oder eine Strecke zu produzieren? Können nur plane Flächen oder auch nicht-uniforme rationale Bézier-Splines (kurz: NURBS) erzeugt werden? In wie

4 Trial and Error, dt. Versuch und Irrtum, eine heuristische Methode bei der verschiedene Möglichkeiten probiert werden. Ein gewünschtes Resultat wird um den Preis wahrscheinlicher Fehler erreicht.

5 Wie etwa eine Zeitleiste in einer Animationssoftware

vielen Untermenüs ist die Schaltfläche versteckt? Techniken wie NURBS, Animationen oder dynamische Blöcke eröffnen neue Wege der Geometrieerstellung, und schränken durch die Bedingungen, die sie an die Eingabe stellen im gleichen Zug wieder ein. Auf diese Weise definieren die Entwickler Handlungen, die mit der Software ausgeführt oder nicht ausgeführt werden können. Selbst wenn die Elektronik mit ihren technischen Begrenzungen als Schnittstelle außen vor gelassen wird, zeigt sich bereits, wie die codierten Handlungsanweisungen auf den Entwurfsprozess Einfluss nehmen. Software, Hardware und Architekt bilden zusammen einen Entwurfsaktanten, dessen Handlungen kollektiv herbeigeführt werden. Im erweiterten Kreis haben auch die Entwickler, die Techniker und die Lehrenden Einfluss auf den Prozess. Ziele die vom Architekten verfolgt werden, sich von der Technik aber nur in abgewandelter Form umsetzen lassen, führen zu einem Übersetzen des Entwurfs, und damit zu neuen Sichtweisen und Ergebnissen. Der Maßstabssprung im Modellbau wird durch die Übersetzung in eine vom Programm verarbeitbare Anweisung ersetzt.

> Wir werden von der Handlung leicht *überrascht*, wie jeder Baumeister weiß. (…) Werden wir darum schon von dem, was wir tun, hinters Licht geführt? (…) Nein, nicht immer, nicht ganz. Was uns leicht überrascht, wird durch das *clinamen* unserer Handlung selbst *ebenfalls* leicht überrascht und modifiziert. (Latour B. , 2002, S. 345)

Diesen Übersetzungsprozessen ist immer ein Bruch – oder Überraschungspotenzial – immanent, der von Wissenschaftlern und Architekten gleichermaßen genutzt werden kann. Wie bereits beschrieben sind die Brüche von Vorteil, um Kenntnis über den Entwurf zu erhalten und ihn weiterzuentwickeln. In diesem Kontext scheint es wenig verwunderlich, dass manche die »Trial and Error« Methode einsetzen, um ein für sie akzeptables Resultat herbeizuführen. Das Ergebnis und die Erwartungshaltung daran werden maßgeblich von der Interaktion mit dem Softwareaktanten bestimmt. Latour schreibt in diesem Zusammenhang vom »überrascht werden«, wenn er auf den unbekannten Ausgang der Interaktion zweier zusammengefasster Aktanten verweist. Vom besagten Überrascht-Werden und den damit verbundenen Erkenntnisgewinnen getrieben, gewinnen die Verbindungen des Werkzeugs in gleichem Maße an Gewicht, in dem es die anderen einbüßen. Im unkritischen Umgang begünstigt die Drift die Technik und führt dazu, dass der Architekt von seinem Entwurfsziel weiter abrückt. Als Beispiel kann hier das Projekt »Seek« aus dem Jahr 1970 gesehen werden. Nicholas Negroponte und die Architecture Machine Group hatten anlässlich einer Kunstausstellung Rennmäuse, Aluminiumwürfel und einen Roboter samt Steuerung in einem Terrarium zusammen

entwerfen lassen. Die Nagetiere konnten mit den Würfeln ihre Höhlen bauen, und wenn ihre Veränderungen nicht im Bereich der gültigen Werte für die Steuerung des Robotergreifers waren, schlichtete dieser die Gebilde immer wieder um (Vgl. Burnham, 1971, S. 19-27).

Wie die eingesetzten Techniken überraschen können liegt nicht zuletzt an ihrer eigenen Ausgestaltung. Die Techniker die Programme entwickeln, greifen ebenso auf andere Aktanten zurück, um mit ihnen gemeinsam Techniken zu entwickeln. Sind es nun neue Datenbankstrukturen, verschiedene Programmiersprachen oder mathematische Verfahren – sie versetzen den Programmierer in die Lage sein Ziel in abgewandelter Form zu erreichen. Gemeinsam sind sie dafür verantwortlich, wie das Zeichenprogramm des Architekten diesen von seinem ursprünglichen Ziel abbringt. Welche Ziele ein einzelner Aktant verfolgt, oder welche ihm eingeschrieben sind, hängt damit zusammen, in welche Netzwerke er eingebunden ist. Die Netzwerke oder Rhizome können verschiedenste Inhalte haben, und dementsprechend kann das Miteinbeziehen eines Aktanten sich auch auf alle mit ihm verbundenen auswirken.

> With critique, you may debunk, reveal, unveil, but only as long as you establish, through this process of creative destruction, a privileged access to the world of reality behind the veils of appearances. Critique, in other words, has all the limits of utopia: it relies on the certainty of the world *beyond* this world. By contrast, for compositionism, there is no world of beyond. It is all about *immanence*. (Latour B. , 2010, S. 475)

Durch das Auflösen der Objekt-Subjekt Dichotomie und dem Ansatz der Delegation von Arbeit lassen sich die Netze besser verfolgen. Uns wurde gezeigt, dass es kein Außerhalb in jedweder Form gibt, und sich die Netze unendlich weit spinnen lassen. Die Softwareentwickler, die Architektur mit entwerfen, sind dafür nur ein stellvertretendes Beispiel. Es stellt sich die Frage, wie viel Einfluss Aktanten über sieben Ecken noch haben können, aber nicht, ob sie es tun. Überträgt man den beobachteten Einfluss der Software auf Häuser, zeigt sich, wie massiv Entwurfsaktanten bei allem anderen ihre Finger im Spiel haben. Der Zeichner-Hardware-Software Aktant verursacht mit Häusern mehr als nur Räume, denn es kommen auch all jene sozialen, ökologischen und ökonomischen Verbindungen zum Tragen, die in Architektur und Bauen bereits eingegangen sind.

> Denn es häufen sich die Hybridartikel, die eine Kreuzung sind aus Wissenschaft, Politik, Ökonomie, Recht, Religion, Technik und Fiktion. (…) Und dennoch scheint niemand sich daran zu stoßen. Die Seiten Wirtschaft, Politik, Wissenschaft, Literatur, Kultur, Religion, Vermischtes bilden nach wie vor die Rubriken, so als wäre nichts gewesen. Der

> winzigste Aidsvirus bringt uns vom Geschlecht zum Unbewussten, von dort nach Afrika, zu Zellkulturen, zur DNS, nach San Francisco. Aber Analytiker, Denker, Journalisten und Entscheidungsträger zerschneiden das feine Netz, das der Virus zeichnet. Übrig bleiben nur säuberlich getrennte Schubladen. (Latour B. , 2008, S. 8f)

Als Folge gibt es keine »reine« Architektur mehr. Natürlich hat es diese nie gegeben, und gerade Architekten sind sich darüber bewusst, dass sie Künstler und Techniker sind, Politiker und Ökonomen, und seit kurzem auch Programmierer und Umweltsystemwissenschaftler. Entweder ist alles über kurz oder lang Architektur, oder gar nichts. Die Gebäude sind von ihrem Sockel der isolierten Betrachtung zu stoßen. Sie existieren nie in einem euklidischen Raum, und schon gar nicht statisch. Latour schlägt vor, sie im Gegensatz als »Fluss von Transformationen« zu verstehen. Das Gebäude ist ein Aktant, den ein Kollektiv geschaffen hat und an den es dadurch eine Menge an Handlungen delegieren konnte. Die Architektur kann sich dieser Verantwortung nicht entziehen. Man kann sich ihrer zusätzlichen Verbindungen nicht entledigen, nur um sie auf eine ihrer Bedeutungen zu reduzieren. Erst durch das Ablegen der statischen Sichtweisen, kann Architektur Gebäude als Aktanten statt als Objekte begreifen.

> Nur durch das Betrachten der Bewegungen eines Gebäudes und durch das sorgfältige Aufarbeiten seiner »Sorgen« wäre man in der Lage, seine Existenzen darzulegen: Damit würde dargestellt, wie das Gebäude agiert, wie es sich Versuchen der Umwandlung widersetzt, wie es Aktivitäten von Besuchern herausfordert oder andere Gruppen von Akteuren mobilisiert. (…) Wir sollten heute in der Lage sein, ein Gebäude als *Navigation* durch eine kontroverse Datenlandschaft zu denken: mit einer lebendigen Folge von erfolgreichen und gescheiterten Konzepten und Entwürfen, mit einer kreuz und quer verlaufenden Zeitschiene unbeständiger Definitionen und wechselnder Kompetenzen, mit widerspenstigen Materialien und Technologien; wechselnden Nutzeransprüchen und Beurteilungen. (Latour B. u., 2008, S. 86)

Der Architekt entwirft mit Gebäuden bereits Aktanten. Er delegiert Handlungsweisen und beeinflusst so zukünftige Bewohner, Investoren, Passanten oder Baufirmen. Aber er entwirft es nicht allein, er ist Teil eines viel größeren Kollektivs: Über scheinbar leblose Werkzeuge haben auch deren Erschaffer Einfluss, wahrscheinlich in den meisten Fällen ohne tief gehende Absicht dahinter. Was aber, wenn schon der Architekt an seinen Werkzeugen mitentwickelt? Wenn er Einfluss darauf nehmen kann, wie er anschließend von ihnen beeinflusst wird? Die vielen Werkzeuge und Fakten, von anderen konstruiert und beeinflusst, haben seine Rolle im Entwerfen geschwächt. Kennzahlen und Techniken lassen seine Entwürfe abdriften, und verdrängen ihn zusehends. Eine kritische Auswahl der Mitglieder seines Entwurfskollektives wäre ein erster Schritt, und geht mit dem Be-

wusstsein einher, dass seine Entwürfe nicht exakt übersetzt werden können. Will er aber Einfluss auf sein eigenes Handeln zurückgewinnen, wird ihm nicht erspart bleiben selbst Hand anzulegen und eigene Werkzeuge zu entwerfen.
Anstatt die Auswirkungen des Surrealismus auf das Denken und die Entwurfsphilosophie von Rem Koolhaas zu analysieren, sollten wir vielmehr versuchen, das Fehlverhalten des Schaumstoffs beim Modellbau in seinem Büro zu begreifen (Latour B. u., 2008, S. 87). Auch wenn ihm die Türen zu den Entwicklungslabors der Schaumstoffproduzenten verschlossen bleiben mögen, gibt es andere Optionen Werkzeuge mit zuentwickeln. Der massive Einsatz von Software im architektonischen Entwerfen, und die Möglichkeit sie mit relativ wenig materiellen Ressourcen zu manipulieren, macht diese zu einem erstklassigen Ziel unserer Aufmerksamkeit.

> The evolution of computational design technique from mere substitution of hand drawing to customized design algorithms exhibiting certain degree of intelligence, has brought up not only great design varieties, but also the demand for a critical study on the relationship between human designer and their customized design algorithms. (…) This relationship could be mapped as a one-way command and execution relationship, which is heavily relying on the automation of design algorithm, while limiting the input of a human designer to mere programming. This relationship can however be set up as an interactive process, in which a human designer not only program the design algorithm, but also actively participates in an interactive computation process to bring in a set of non-parameterizable design inputs, such as intuition, aesthetics and associational reasoning that are essential to any design activity. (Feng, 2009, S. 183)

Dabei geht es nicht nur darum das Verhältnis des Entwerfers zu individualisierter Software zu hinterleuchten, wie bereits vielerorts gefordert wird, sondern darum, dem Entwerfer mehr Einfluss zu ermöglichen. Interaktive Prozesse sind dafür sicher ein Fortschritt, genauso wie eigene Algorithmen zu entwerfen. Das Hauptaugenmerk muss aber auf die Auswahl der Aktanten gerichtet werden, die als Fakten Grundlage für die Algorithmenmodelle liefern. Ein interaktives Entwurfsspiel kann unkritisch programmiert, selbst wenn es aus der Feder des Architekten stammt, den Entwerfer einer stärkeren Drift aussetzen, als hätte dieser sich nicht eingemischt. Auf der anderen Seite kann er Entwurfshandlungen an Aktanten delegieren, und im Entwurf mit Vertretern seiner selbst zusammenarbeiten. Konstruiert man einen Aktanten, hat das meist den Hintergrund, dass man diverse Aufgaben an ihn delegieren will, und die eigenen Ziele am Erschaffen in zukünftigen Interaktionen weiter bestehen. Und weil, wie weiter oben beschrieben, die angestrebten Ziele nie exakt umgesetzt werden können, überrascht jeder Aktant schlussendlich mit einer eigenen Zielsetzung.

Jeder Versuch einer isolierten Betrachtung von Aktanten erleidet das gleiche Schicksal wie die Objekt-Subjekt Dichotomie, er wird überflüssig. Eine digitale Entwurfsumgebung soll neben Interaktivität und Programmierbarkeit auch das Zusammenstellen des Entwurfskollektives beeinflussen. Die gewonnene Transparenz verschafft dem Entwerfer eine Übersicht, die als solche zu neuen Entwurfsimpulsen führen und für eine leichtere Umsetzung seiner Ziele behilflich sein kann.

3. INTERAKTION

Wie Kontrollverlust und Einflussgewinn Hand in Hand gehen

DIE HERRSCHAFT: OBSOLETE MACHTGEFÜGE

Eine Frage, die sich bei der netzwerkartigen Betrachtungsweise aufdrängt, ist die nach einer möglichen Beherrschung: Wer hat welche Macht über wen? Und existieren Top-Down Prozesse überhaupt? Einen klassischen Zugang zu diesem Thema schildern Gerd De Bruyn und Wolf Reuter, die die Interaktion zwischen den Aktanten auf einen Machtfluss zurückführen. In ihrem Modell stützt sich das Durchsetzungsvermögen eines Architekten auf eine Machtposition, durch die der Architekt sich in Fragen der Gewichtung verschiedener Entwurfsaspekte durchzusetzen vermag. Sie erkennen auch, dass in einem Entwurf verschiedene Zielsetzungen, bei ihnen als »Wertvorstellungen« bezeichnet, mit einander im Wettkampf stehen. Aus einer Architekten-Bauherren Allianz in der Renaissance entwickeln sie ein System von Teilnehmern, die um die Entscheidungsmacht im Entwurf wetteifern.

> Macht ist für einen Architekten immer dann wichtig, wenn er sie für die Entwicklung oder Ausführung seiner Pläne, Ideen und Konzepte braucht. Hatten in der Renaissance noch große Mäzene ihre Macht an Architekten weitergegeben und geliehen – wobei man davon ausgehen konnte, dass die bestimmenden Wertsysteme bei beiden ziemlich kongruent waren – , so ist für moderne Gesellschaften die Diversität der Werte charakteristisch und sogar prinzipiell vorherrschend. Oft ist es nur die triviale aber entscheidende Differenz in der Gewichtung wirtschaftlicher und funktionaler Aspekte gegenüber der kulturellen Verankerung eines Entwurf. Hinzu kommt, dass nicht allein die Wertvorstellung verschiedener Subsysteme miteinander konkurrieren, sondern zudem jedes dieser Systeme in sich selber werteheterogen ist. (DeBruyn & Reuter, 2011, S. 131)

Diese These beschreibt nicht nur eine tief gehende Verbindung zwischen Architektur und Macht, sie geht soweit Macht als ein Grundelement des architektonischen Handelns zu definieren. Die Interaktion der Teilnehmer erfolgt in ihrem Modell nicht rein über Diskurs oder Macht, sondern über ein Zusammenspiel von Diskurs und Macht,

komplementäres Wissen entsteht. Diese These stößt dort an ihre Grenzen, wo die Objekt-Subjekt Dichotomie aufgehoben wird, und der Machtfluss zwischen Mensch und Nicht-Mensch nicht mehr zustande kommt (aber übt der Statiker der einen Träger bemisst Macht auf den Architekten aus?). Das von De Bruyn und Reuter gezeichnete Herrschaftsbild in der Architektur spiegelt die Haltung der Moderne wieder:

> Wenn wir daher den Anspruch unserer These aufrechterhalten, dass Architektur Weltwissen ist und verkörpert, dann wird außerdem zu zeigen sein, dass Verbindung von Architektur und Macht Teil der von uns angesprochenen universalen Netzstruktur ist. Wir stellen daher die These auf, dass es nicht nur einen tief sitzenden, strukturellen Zusammenhang zwischen Architektur und Macht gibt, sondern dass dieser Zusammenhang die soziale Vernetzung der Architektur repräsentiert und dass Macht ein wesentliches Konstituens des Handelns der Architektinnen und Architekten der Welt ist. (DeBruyn & Reuter, 2011, S. 119f)

Im Gegensatz dazu versucht die Actor Network Theory diese Frage, besonders in der Interaktion mit nicht-menschlichen Aktanten, anders zu beleuchten. Bruno Latour hat mit der Überraschung eine Alternative zur Beherrschung entworfen, und untergräbt damit die Bestrebungen anderer in Netzwerken eine »Herrschaft« zu konstruieren. Der Aktant wird in Latours Theorie von seinem Handeln überrascht, da dieses durch die Kollaboration mit anderen Aktanten abdriftet. Der Ursprung dafür liegt aber nicht in der Dominanz des Artefaktes oder des Menschen, sondern in der Übersetzung. Der Handelnde hat keine absolute Kontrolle über sein Handeln und dieses nicht über ihn. Und genau hier setzen gerade für den Architekten die Probleme ein: Der Architekt mag sich vielleicht nicht als »Herrscher« über den Entwurf sehen, aber er will sich auch nicht auf einer Stufe mit allen anderen wiederfinden. Und das, obwohl inzwischen klar ist, wie viele verschiedene Aktanten an einem Entwurf mitarbeiten. Niemand käme auf die Idee zu sagen, Herzog & de Meuron hätten das Nationalstadion in Peking (das »Vogelnest«) zu zweit und ohne Hilfe entworfen; und sicher auch nicht, Dassault Systèmes wären es mit ihrer Software CATIA gewesen, oder die Ingenieure von ARUP.

> Wie wir in Kapitel 4 gesehen haben, ändert sich alles von Grund auf, wenn wir auf das hören, was praktizierende Wissenschaftler sagen, ohne etwas hinzuzufügen oder abzuziehen. Der Wissenschaftler macht den Fakt, doch wann immer wir etwas machen, haben nicht *wir* das Kommando: Wir werden von der Handlung leicht *überrascht*, wie jeder Baumeister weiß. Das Paradox des Konstruktivismus liegt also darin, dass er ein Vokabular der *Beherrschung* verwendet, das kein Architekt, Maurer, Stadtplaner oder Zimmermann je gebrauchen würde. Werden wir darum schon von dem, was wir tun, hinters Licht geführt? Werden wir kontrolliert, vereinnahmt, entfremdet? Nein, nicht

> immer, nicht ganz. Was uns leicht überrascht, wird durch das *clinamen* unserer Handlung selbst *ebenfalls* leicht überrascht und modifiziert. (Latour B. , 2002, S. 345)

Als Architekt und Urheber gelten aber erstere, und ihr Büro. Nicht zuletzt durch den Umstand, dass sie tatsächlich großen Einfluss auf den Entwurf hatten. Diese lapidare Unterscheidung von Einfluss und Herrschaft lässt die vorgestellte Interaktion von menschlichen und nicht-menschlichen Aktanten plötzlich noch brisanter werden. Drift und Übersetzung haben, genauso wie die Vernetzungen der Aktanten, Einfluss auf das Handeln. Das Fehlen eines Kommandos, eines »Herrschers«, entspricht der rhizomatischen Natur des Netzwerks. Da immer wieder Hilfskonstrukte entwickelt wurden um das »Machtausüben« diskutieren zu können wurde die Frage nach dem Einfluss verdrängt (Vgl. ebenda). Deleuze und Guattari haben bereits im Jahr 1977 die Machtbestrebungen in Netzwerken kritisiert und erfolgreiche Gegenstrategien dazu recherchiert.

> Vom Standpunkt der Guerillalogik aus findet man die Lösung ohne General für eine nicht zentrierte Vielheit, die eine endliche Anzahl von Zuständen und Signalen entsprechender Geschwindigkeit enthält. Es wird sogar gezeigt, dass eine solche Vielheit, maschinelle Verkettung oder maschinelle Gesellschaft, jeden zentralisierenden und vereinheitlichenden Automaten als »asozialen Eindringling« abweist. (Deleuze, 1977, S. 28)

Die Guerillataktik, in der zerstreute Einheiten ohne militärische Hierarchie ihr Handeln koordinieren, ist eines der Beispiele, die in diesem Zusammenhang fallen. Das Rhizom ist per Definition nicht hierarchisch aufgebaut, es kann dadurch keine Befehlskette im klassischen Sinn etabliert werden. Die Aktanten wirken aufeinander ein, aber sie haben keine Macht oder Herrschaft über die anderen, in dem Sinn wie sie einst Max Weber definiert hat. Guerillas haben keine Generäle und Aktanten haben keine Herren, die Interaktion unterliegt anderen Regeln. Wie bereits erwähnt, liegt das jeweilige Abdriften ihrer Ziele an ihren übrigen Vernetzungen, nicht an direkter Kontrolle. Das Einwirken aufeinander kann nicht als »Macht ausüben« deklariert und analysiert werden, denn Macht würde einen Widerstand im Bruch der Übersetzung voraussetzen, der nicht gegeben ist.

> *Macht* bedeutet jede Chance, innerhalb einer sozialen Beziehung den eigenen Willen auch gegen Widerstreben durchzusetzen, gleichviel worauf diese Chance beruht. *Herrschaft* soll heißen die Chance, für einen Befehl bestimmten Inhalts bei angebbaren Personen Gehorsam zu finden; *Disziplin* soll heißen die Chance, kraft eingeübter Einstellung für einen Befehl prompten, automatischen und schematischen Gehorsam bei einer angebbaren Vielheit von Menschen zu finden. (Weber, 1984, S. 89)

Etzioni führt in seinem Text als nicht unterdrückendes Pendant zur Macht den Einfluss an. Er weist daraufhin, dass die Aktanten ihre Ziele zu einem gemeinsamen addieren. Über Einfluss kann die Abweichung vom resultierenden Ziel zum eigenen ursprünglichen Ziel definiert werden. Wenn diese Abweichung allerdings bewusst in die Wege geleitet wird, ist der daraus resultierende Einfluss in seiner Folge dem der Macht ähnlich. Einfluss setzt kein Machtgefüge voraus, weil kein Zwang herzustellen ist. Nicht-menschliche Aktanten wären auch selten in der Lage, diesen auszuüben – außer in materialistischen Sichtweisen natürlich. Sie können in dieser Form aber Einfluss auf menschliche Aktanten ausüben, ohne diese unterdrücken zu müssen. Es gibt keine Herrschaft und es gibt keine Macht mehr um sich über andere zu stellen – was bleibt ist der eigene Einfluss auf Gleichgestellte. Ohne die Objekt-Subjekt Dichotomie fällt das primäre Herrschaftsgefüge der Moderne und damit die Idee der Dominanz. Manche Architekten haben bereits festgestellt, dass sie nicht mehr alleine an der Spitze des Entwurfs stehen, und dass zwischen ihren ersten Vorstellungen und der gebauten Realität Differenzen aufklaffen, für die sie nicht direkt verantwortlich sind. Sie haben begonnen andere Teilnehmer am Entwurfsprozess anzuerkennen, und haben angefangen Macht abzugeben. Aber auch die ambitioniertesten Ansätze gehen nicht soweit dem Weltgefüge Bruno Latours in die Augen zu blicken. Anstatt Einfluss anzuerkennen, der unterschiedlich schwer ausfallen kann, werden neue Machtpositionen entworfen.

> Einfluss und Macht werden oft als Synonyme gebraucht. Wir halten es jedoch für sinnvoll, die beiden Begriffe getrennt zu halten, weil dadurch eine signifikante konzeptuelle Unterscheidung ausgedrückt werden kann. Die Anwendung von *Macht* verändert die Situation eines Aktors und/oder seine Konzeption von der Situation – nicht aber seine Präferenzen. Widerstand wird nicht deshalb überwunden, weil derjenige, auf den Macht angewendet wird, seinen »Willen« ändert, sondern weil Widerstand teurer oder unmöglich gemacht wurde. Die Ausübung von *Einfluss* hingegen hat eine authentische Veränderung der Präferenzen des Aktors zur Folge; in derselben Situation würde er nicht wieder diejenige Handlungsrichtung wählen, die er bevorzugt hatte, bevor auf ihn Einfluss ausgeübt wurde. Während vom Standpunkt der Machthaber der Unterschied zwischen Einfluss und Macht relativ gering erscheinen mag (…) ist er für die Objekte der Macht von größerer Bedeutung, weil Einfluss nicht eine Ablösung oder Unterdrückung ihrer Präferenzen, sondern eine Neuspezifizierung ihrer Bindungen bedeutet. (Etzioni, 2009, S. 379)

Durch das marktwirtschaftliche System begünstigt, hat sich eine Trennung zwischen Auftraggeber und Auftragnehmer etabliert, die versucht Ersteren auf eine Machtposition zu heben. Während Auftragnehmer in der Agenturtheorie als kollaborierende Agenten mo-

delliert werden, gibt es eine über Verträge definierte Beziehung zu einem Prinzipal, oder mehreren Prinzipalen. Dieses Szenario stützt sich auf die Annahme, eine direkte Übersetzung der Ziele des Prinzipals in Handlungen der Agenten sei möglich. Wie uns die Actor Network Theory aber zeigt, entspricht diese Beziehung, obwohl formell definiert, der zwischen zwei sich zusammensetzenden Aktanten. Der Prinzipal wird von den Agenten beeinflusst, so wie diese von ihm. Die beschriebene Trennung ist der Versuch einer Ersatzkonstruktion von Herrschaft, um eine Kontrollinstanz zu systematisieren.

> Die Agenturtheorie stellt ein Phänomen in den Mittelpunkt ihrer Betrachtung, das von anderen Organisationstheorien wie auch von der neoklassischen Mikroökonomik vernachlässigt wurde, nämlich die Abhängigkeit der Leistungserstellung von einer effizienten vertraglichen Gestaltung der Beziehung zwischen Prinzipal und Agent. Sie fasst Organisationen als Netzwerke von vertraglich geregelten Auftragsbeziehungen auf. Mit dieser Konzeption ist sie einerseits einzigartig, andererseits kann sie aber anknüpfen an mehrere organisationstheoretische Ansätze. (Saam, 2002, S. 35)[6]

Während die Agenturtheorie vor allem zum Erforschen vom Risikoverhalten kooperierender Akteure in Unternehmen herangezogen wird, wird ein vergleichbarer Ansatz zur Trennung von Architekt und »Entwurfsschwarm« genutzt. Die Frage nach dem Risiko wird durch die Frage nach der Prozesskontrolle ersetzt. Ein Vertreter dieser Denkweise ist Kas Oosterhuis mit seiner Theorie der Swarm Architecture (zukünftig SAT genannt). Sie greift die Idee des Entwurfskollektivs auf und gestaltet »bottom-up« Prozesse transparenter. Das Kollektiv setzt sich in der SAT vor allem aus menschlichen Aktanten zusammen, die im Prozess des Entwerfens untereinander Informationen austauschen. Neben diversen Fachplanern und Spezialisten umfasst das Kollektiv auch jede Art von Kommunikation und Grafik wie auch alle Arten von wirtschaftlichen Beteiligungen. Insofern ist der Ansatz dem großen Bild, dass uns die ANT liefert, nicht unähnlich. Er begreift Entwerfen bereits als kollaborativen Prozess und führt ihn nicht auf einen Urheber zurück.

> In the design swarm, the designers exchange information with their clients, and with the other stakeholders in the process of building their vision. They exchange information with other disciplines in the collaborative design process, they may operate together with visual artists, composers, graphic designers, planners, publishers, broadcasters of information, with other architects. They exchange information with construction engineers,

6 Prinzipal, Auftraggeber; Agent, Auftragnehmer (Vgl. ebenda)

> installation engineers, project managers and process managers. (Oosterhuis, 2003, S. 58f)

Neben der SAT befassen sich inzwischen viele andere aus dem Umfeld der Architektur mit dem verteilten Entwerfen. Ingeborg Rocker hat dazu den Entwurf des BMW Ausstellungsgebäudes für die Internationale Automobilausstellung 2001 untersucht. Abgesehen davon, dass sich über 75 Experten am Entwurf beteiligt haben, stellte sie fest, dass der Architekt anstelle des Entwurfs den Prozess kontrollierte. Der Autor als einzelnes Subjekt wird hier durch einen kollaborativen Prozess ersetzt. In der Analyse fällt neben der Objekt-Subjekt Dichotomie die Konstruktion des kontrollierenden Architekten auf. Es wird ganz klar eine Hierarchie zitiert, in der Subjekte über Objekte herrschen, und der Architekt über allem steht. In Folge kann in der SAT Kollaboration nur unter gleichrangigen Aktanten erfolgen, die wie in der Agenturtheorie vom Architekten hinzugezogen werden.

> With the increasing evolution of large and complex information systems, procedures – such as versioning – become necessary as they are supporting and organizing the often synchronously performed data and knowledge transfers. Versioning has hence been receiving growing attention in nearly all engineering domains as it facilitates and controls via SCM the collaborative authoring by expert groups throughout the world, replacing the single authorial subject through a complex collaborative co-authorial process. DynaForm's evolution had to rely, for example, on versioning, allowing the more than 75 architects, structural and mechanical engineers, communications experts, light designers and AV-media specialists to cooperate often simultaneously with one another. (…) The architect find himself – as a designer – controlling a process rather than a design. (Rocker I. , 2002, S. 12-17)[7]

Obwohl die SAT anfangs einen offeneren Zugang vermuten lässt, konstruiert auch diese Theorie einen Ausweg aus der Führungslosigkeit. Oosterhuis hingegen will zwischen den »top-down« und »bottom-up« Prozessen ein nicht näher definiertes Gleichgewicht herstellen. Steht der Architekt nicht an der Spitze der Ordnung, muss diese von einem anderen eingenommen werden. Die Zusammenarbeit der Mitglieder des Kollektivs wird anerkannt, aber als würde es Unbehagen auslösen werden fast zwanghaft neue Formen der Herrschaft über eben dieses Kollektiv gestellt. Der Entwurf darf, wie es den Anschein hat, nicht den Kopf verlieren. In dieser Konstruktion nutzt der Architekt als Prinzipal die Dienstleister als Agenturen, was eine Machtposition impliziert, wie die eines Subjekts

7 SCM, Software Configuration Management

über ein Objekt. Würde diese Einteilung fallen, beeinflussten sich alle gegenseitig, und die Herrschaftsstrukturen gerieten ins Wanken.

> The SAT aesthetics come from the bottom-up processing within the system, but the styling of the complex surfaces come into play as top-down interventions from the exterior of the skin of the building. I am very much keen upon establishing a balance between the bottom-up and top-down aspects of design. (Oosterhuis, Swarm Architecture II, 2006)

Die Actor Network Theory beschreibt einen alternativen Weg möglicher Zusammenarbeit. Ist man erst bereit die verstaubten Methoden Macht und Herrschaft außen vor zu lassen, eröffnen sich neue Wege des Entwerfens. Der Architekt hat keine Kontrolle über Objekte, über andere Subjekte, und schon gar nicht über die Auswirkungen seines Handelns. Er hat aber in gleicher Weise auch keinen Herren über sich, und braucht sich auch keinen zu schaffen. Der Handelnde wird von seinem Handeln überrascht, und seine Ziele werden beeinflusst. Dieses Überraschungspotenzial ist maßgeblich am Erkenntnisgewinn über den Entwurf beteiligt. Entwerfen ist nicht deterministisch, und nicht hierarchisch. Alle beteiligten Aktanten, und über unzählige Umwege sind das alle die existieren und existiert haben, haben ihren Anteil am Entwerfen. In diesem Kollektiv kann sich der Architekt stärker einbringen, und mehr Einfluss ausüben um den Prozess in eine Richtung hin zu seinen Vorstellungen zu lenken. Er hat aber kein Kommando.

> Warum immer einen Befehlshaber durch einen anderen ersetzen? Warum nicht anerkennen, was wir wieder und wieder im Verlauf dieses Buches gelernt haben? Dass nämlich Handeln ein wenig von dem überrascht wird, worauf es einwirkt; dass es durch Übersetzungen driftet; dass ein Experiment ein Ereignis ist und ein wenig mehr aufweist als seinen Input; dass Ketten von Vermittlungen nicht das gleiche sind wie der reibungslose Übergang von Ursache zu Wirkung; dass Übertragung von Information nie stattfindet außer durch subtile und mannigfache Transformationen; dass es nichts dergleichen gibt, wie einer gestaltlosen Materie Kategorien aufzuzwingen; und dass im Bereich der Technik niemand kommandiert – nicht weil die Technik selbst das Kommando hätte, sondern weil wirklich niemand und gar nichts es hat, nicht einmal ein anonymes Kräftefeld. Etwas kommandieren oder beherrschen ist weder Eigenschaft von Menschen noch von nichtmenschlichen Wesen, ja nicht einmal von Gott. (Latour B. , 2002, S. 366)

Neben der Interaktion menschlicher Aktanten lohnt es jene der nicht-menschlichen genauer unter die Lupe zu nehmen. Bleiben wir beim Beispiel des Digitalen Entwerfens, stellt sich die Frage nach den Strukturen der Interaktion verschiedener Softwareelemente. Manuel de Landa erkennt eine folgenschwere Wende im Wechsel von starr hierarchisch

organisiertem Code zu objektorientiertem Programmieren. Er vergleicht die Übergabe der Kontrolle von einem zentralen »masterprogram« zu mehreren Modulen mit dem Verlust der Kontrolle des Menschen. Nach de Landa hat der Mensch die Kontrolle erst an die Hardware verloren, damit die sie wiederum an die Software verlieren konnte. Er prognostiziert eine weitere Abwanderung der Kontrolle, weg vom Code hin zu der, durch Sensoren und Parameter vertretenen, Außenwelt. So hüpft die Herrschaft in dieser Betrachtungsweise wie eine Heuschrecke von einem Punkt zum nächsten. Lässt man den konstruierten Machtbezug in den Erklärungen außen vor, liefern die vorgestellten Techniken jedoch einen guten Einblick in die Interaktionsmöglichkeiten von und mit digitalen Aktanten.

> From the point of view of the conceptual history of software, the creation of worlds populated by semi-autonomous virtual creatures, as well as the more familiar world of mice, windows and pull-down menus, have been made possible by certain advances in programming language design. Specifically, programming languages needed to be transformed from the rigid hierarchies which they were for many years, to the more flexible and decentralized structures which they gradually adopted as they became more »object-oriented«. One useful way to picture this transformation is as migration of control from a master program (which contains the general task to be performed) to the software modules which perform all the individual tasks. Indeed, to grasp just what is at stake in this dispersal of control, I find it useful to view this change as a part of a larger migration of control from the human body, to the hardware of the machine, then to the software, then to the data and finally to the world outside the machine. (DeLanda, 1998, S. 274)

Führt man den gesamten Entscheidungsprozess auf ein einziges Programm zurück, das Masterprogramm, müssen alle Szenarien, auf die es womöglich reagieren soll, von seinen Entwicklern mitüberlegt werden. Ähnliche, sich wiederholende, Anweisungsketten, sogenannter »cloned code«, hat bei solchen Taktiken in der Frühphase des Computers nicht nur wertvollen Speicher verbraucht, sondern verursacht bei unsauber ausgeführten Quelltexten noch heute Probleme. Nicht nur, dass der Wartungsaufwand der Software dadurch erhöht wird, es steigt damit auch das Fehlerrisiko und auf diese Weise wird ein inkorrektes Programmverhalten wahrscheinlicher (Vgl. Juergens, Deissenboeck, Hummel, & Wagner, 2009, S. 485f). Als Ausweg dazu wurde in der Softwareprogrammierung das Unterprogramm, oder auch Subroutine, geschaffen. Dieses wird vom Masterprogramm aufgerufen, und mehrfach zu verwendende Befehlsfolgen müssen nur noch einmal formu-

liert werden[8]. Üblicherweise handelt es sich dabei um in sich abgeschlossene, präzise definierte Teilaufgaben. Die Struktur spiegelt die Auftraggeber-Auftragnehmer Hierarchie wider. Während die Subroutinen also dazu gedacht sind, alleine für eine gewisse Aufgabe zuständig zu sein, um Kompetenzüberschneidungen zu verhindern, ist ihrer Struktur eine zweite Besonderheit immanent. Das Masterprogramm arbeitet die Arbeitsabfolge seriell ab: Wird eine Subroutine aufgerufen, wird auf deren Ende gewartet um mit der nächsten Handlungsanweisung folgen zu können. Aufgrund des seriellen Ablaufs, könnte der modular gestaltete Quelltext, mit einigen Nachteilen, in einen einzigen langen zusammengefasst werden. Der »Entscheidungsprozess« bleibt so trotz der Aufteilung zentralisiert. Hinzu kommt, dass die Auskopplung von Handlungsanweisungen ermöglicht, dass mehrere verschiedene Programme auf die Unterprogramme zugreifen können. Analog zur Prinzipal-Agenten Teilung, kann das Unterprogramm also von mehreren Hauptprogrammen beauftragt werden.

> Trying to build a roboter using a hierarchy of subroutines meant that researchers had to completely foresee all the tasks that a robot would need to do and to centralize all decision making into a master program. But this, of course, would strongly limit the responsiveness of the robot to events occurring in its surroundings, particularly if those events diverged from the predictions made by the programmers. One solution to this was to decentralize control. The basic tasks that a robot had to perform were still coded into programs, but unlike subroutines these programs were not commanded into action by a master program. Instead, these programs were given some autonomy and the ability to scan the data base on their own. Whenever they found a specific pattern in the data, they would perform whatever task they were supposed to do. In a very real sense, it was now the data itself that controlled the process. (DeLanda, 1998, S. 278)

Das Problem bei dieser Art von Steuerung ist, dass alle Eventualitäten bereits im Vorhinein bedacht werden müssen. Für jede mögliche Aufgabe besteht eine Handlungsanweisung, nichts geschieht zufällig (vorausgesetzt man verzichtet auf das Generieren von Pseudozufallszahlen, wobei selbst diese bei rekursiven arithmetischen Verfahren prognostizierbar wären). Weicht die Praxis aber von der Theorie ab, stößt der Code an seine Grenzen. Ein erster Schritt, um auf diesen Umstand einzugehen, ist die Abkehr von linearen Programmabläufen. Gerade an der Schnittstelle zur Umwelt ist da-

8 **Beispiel: Anstelle zur Berechnung eines Kreisumfangs jedes mal die ganze Formel (u=r*2* π) aufzurufen, wird diese durch das Unterprogramm „umfang" ersetzt (umfang(x)= x*2*π), welches beliebig oft aufgerufen werden kann (u=umfang(r)). Bei einem Ändern der Formel kommt diese nur noch an einer Stelle vor.**

mit die Möglichkeit gegeben nicht auf die Prognostizierbarkeit von Ereignissen angewiesen zu sein. Im Speziellen wird das ereignisorientierte Programmieren im Gestalten von grafischen Benutzeroberflächen verwendet, da Ereignisse meist Aktionen des Benutzers darstellen. Tastendruck, Mausklick oder jede Art von Sensorinteraktion rufen anstelle eines Masterprogramms ein Unterprogramm auf, und zwar nur exakt dann wenn es gebraucht wird (Vgl. Meyer, 2004, S. 239f). Die Kontrolle über das Ausführen wird mit dieser Technik von einem zentralen Masterprogramm auf einen dezentralen Ereignisauslöser übertragen, der Handlungsablauf wird damit nicht mehr vorherbestimmt.
Lange Zeit hat die Prozessorarchitektur bei solchen Ereignissen den Hauptprozess unterbrechen müssen, um das damit verbundene Unterprogramm abarbeiten zu können (Vgl. Ziesche, 2005, S. 16). Ein paralleles Prozessieren konnte durch das abwechselnde Ausführen mehrerer Programme simuliert werden. Für eine tatsächlich zeitgleiche Ausführung sind jedoch mehrere Prozessoren notwendig. Eine Voraussetzung, die durch die Fortschritte in der Prozessorentwicklung inzwischen bereits von Smartphones erfüllt werden kann. Vergleichbar mit den Abläufen in einem Architekturbüro, werden die anfallenden Arbeitsaufgaben nicht mehr nur von einem Aktanten erledigt, sondern auf mehrere aufgeteilt. Sie werden während des Prozesses nicht nacheinander mit viel Wartezeit, sondern parallel abgearbeitet.
Vor diesem Hintergrund ist die Entwicklung von Software-Agenten[9] zu beobachten: Bei diesen autonomen Programmen laufen je nach Zustand bestimmte Vorgänge ab. Weder ist ein Startsignal dazu notwendig, noch irgendeine Art von Steuerungseingriff. Das besondere an diesen Agenten ist, dass sie Aktionen auf eigene Initiative hin ausführen. Sie können auf Umwelteinflüsse, Störungen, Erfahrungen und andere Agenten reagieren und sind damit in ihrer Ausführung vom Benutzer unabhängig. Gleichzeitig ausgeführt können sie komplexeste Aufgaben erfüllen, die mit klassischen Programmiermethoden nur schwer – falls überhaupt – umzusetzen wären. In ihrem Verhalten sind sie dem zellulären Automaten ähnlich, der in Bezugnahme auf seinen Zustand, und den seiner Nachbarn, diesen nach definierten Regeln ändert (Vgl. Hütt, 2001, S. 103f). Durch die Interaktion der Automaten können diverse dynamische Systeme modelliert werden.
Der wichtigste Aspekt im Modellieren von Agenten ist, dass sie sich in einer realen oder simulierten Umwelt befinden. Andere Agenten und Elemente, die sich in ihrer Umwelt

9 langlebige autonome Software Programme, die eine Umwelt beobachten, darauf reagieren, mit anderen Agenten und Benutzern kommunizieren und zusammenarbeiten. (Vgl.Detlor, 2004, S. 147f)

befinden, können auf sie einwirken, oder von ihnen beeinflusst werden. Die einfachste Form von Agent ist der »reaktive« oder »subkognitive«, der sein Handeln anhand seiner gegenwärtigen Situation wählt. Die höchste Form von Agentenarchitektur ist die »deliberative«, zu der ausschließlich BDI-Agenten[10] zählen. Er reagiert nicht direkt auf seine Wahrnehmung, sondern formt daraus erst ein internes Weltmodell, mit dem er sein Handeln planen kann (Vgl. Klügl, 2001, S. 19-27). Auf diese Weise entscheidet der Agent nicht mehr nur über den Zeitpunkt der Ausführung, sondern auch über die Handlung, die er setzt. Die Flexibilität der Software hat sich damit gegenüber früheren Architekturen verbessert, und ihre Vorhersagbarkeit ist im gleichen Maß gesunken. Die Handlungsverantwortung ist mit dieser Entwicklung vollständig dezentralisiert, unterliegt aber einer durch die Umwelt vorgegebenen Drift.

> First of all, from the previous comments it should be clear that the degree of hierarchical and homogenizing components in a given interface is a question which affects more than just events taking place in the computer's screen. In particular, the very structure of the workplace, and the relative status of humans and machines is what is at stake here. Western societies have undergone at least two centuries of homogenization, of which the most visible element is the assembly-line and related mass-production techniques, in which the overall thrust was to let machines discipline and control humans. In this circumstances, the arrival of the personal computer was a welcome antidote to the development of increasingly more centralized computer machinery, such as systems of Numerical Control in factories. (DeLanda, 1998, S. 284)

Wie die Software im digitalen Raum agiert und interagiert hat eine Auswirkung auf das Bild der Zusammenarbeit im Entwerfen. De Landa beschreibt in seinem Aufsatz ein Hin-und-her-Wechseln der Macht, vom Menschen zur Maschine und zurück, ohne die Möglichkeit einer nicht-hierarchischen Kollaboration in Betracht zu ziehen. Fällt die Trennung von Mensch und Maschine zugunsten des Aktantenmodells, ist die Frage nach der Dominanz mehr eine Frage nach Vernetzungen als nach Macht. Die ins Spiel gebrachte Homogenisierung ist ebenfalls eine Auswirkung der Drift, die eine Kooperation mit dem Aktanten »Fließbandfertigung« mit sich bringt. Nimmt man an, dass die Maschinen als Aktanten, genauso wie der Mensch oder die Programme, auf das gegenseitige Handeln zwar einwirken aber keinen Zwang ausüben können, entsteht ein kollaborierendes Kollektiv. Das Verhalten der digitalen Aktanten kann dem der menschlichen ähnlich gestaltet werden, und so auch ihre Hierarchie. Mit der Entwicklung der Software-Agenten hat sich

10 BDI, Belief (Umweltwissen) Desire (Grobziel) Intention (konkrete Pläne)

der Aufbau des Systems den Strukturen von reellen Büros angenähert. Manuel de Landa stellt damit eine Auswirkung auf den menschlichen Entwerfer außer Frage.

> We must also recognize that by its very nature, systems governed by nonlinear dynamics resist absolute control and that sometimes the machinic phylum can only be tracked, or followed. For this task even our modicum of free will may suffice. The »searching device« constituted by genetic variation and natural selection, does in fact track the machinic phylum. That is, biological evolution has no foresight, and it must grope in the dark, climbing from one attractor to another, from one engineering stable strategy to another. Perhaps one day virtual environments will become the tools we need to map attractors and bifurcations, so that we too can track the machinic phylum in search for a better destiny for humanity. (DeLanda, 1994, S. 286)

Dynamische nicht-lineare Systeme wie jene interagierender Agenten, können auch nach de Landa nicht kontrolliert werden. Für ihn hat der Entwerfer in diesen Fällen nur die Möglichkeit ihre Entwicklung aufzuzeichnen und zu verfolgen. Und obwohl er diese Feststellung getroffen hat, oder vielleicht gerade durch sie, formuliert auch er die Forderung, eine richtungsweisende mächtigere Institution in das System zu integrieren. Zwar misst er den Systemen die Fähigkeit zu, passiv durch ihre Analyse zu einem besseren Verständnis der Gesellschaft zu verhelfen, wagt es aber nicht sie von der fiktiven Leine zu lassen. Die Dezentralisierung der Entscheidungsgewalt führe zu Netzwerken, die unkontrollierbar in ihrer Entwicklung driften. Was aber wenn wie in der ANT keine Beherrschung möglich ist? Wie viele andere vertritt de Landa die Position, dass im Entwerfen eine Richtung vorgegeben werden können muss. Eine Möglichkeit dies zu tun bleibt Einfluss auszuüben, was dem Konzept der Dezentralisierung in keiner Form widerspricht.

> After all, meshworks grow by drift and they may drift to places where we do not want to go. The goaldirectedness of hierarchies is the kind of property we may desire to keep at least for certain institutions. Hence, demonizing centralization and glorifying decentralization as the solution to all our problems would be wrong. (DeLanda, 1998, S. 285)

Einen ähnlichen Ansatz verfolgt auch Kas Oosterhuis mit seiner Theorie der »Swarm Architecture«. Neben der beschriebenen Erweiterung der Gruppe der menschlichen Entwurfsteilnehmer stützt diese Theorie auch auf dem Konzept der Software-Agenten. Entwerfen ist in diesem Kontext das Zusammenspielen vieler verschiedener Systeme, in dem die einzelnen Aktanten gewissen Handlungsregeln folgen. Die Beschreibung des Prozesses geht größtenteils mit der Actor Network Theory konform, wenn auch die bereits erwähnte Forderung nach »top-down« Prozessen eine Ausnahme darstellt. Gerade das

Anerkennen von Aktanten verschiedenster Natur hat sie jedoch anderen Ansätzen voraus. Neben der großen Menge an menschlichen Teilnehmern umfasst die SAT auch die Menge an geometrischen Bedingungen, hier unter dem Begriff »Point Cloud« zusammengefasst. Im Vergleich zu einfachen formalen Bedingungen werden durch eine Ausführung der Elemente als zelluläre Automaten komplexere Eingriffe erreicht.
Als letzte Gruppe entwerfen die involvierten Artefakte selbst mit. Das geschieht indem Materialien parametrische Konstruktionsdetails beeinflussen oder elektromechanisch korrespondierende Bauteile digital vernetzt werden. Jedes Mitglied führt wie beschrieben eine einfache Arbeitsanweisung aus (auch »spielen« genannt). In der SAT wird der Entwurfsprozess als das Erstellen von Spielregeln und das anschließende Durchspielen dieser dargestellt. Die Herausforderung liegt im Entwickeln von Regeln die Komplexität fördern anstatt Prozesse zum Erliegen zu bringen. Ein logischer Schnittpunkt ergibt sich in dem Moment, da vom Entwerfen der Spielregeln zum Spielen des Entwerfens gewechselt wird. Den Startschuss dafür gibt der Architekt, der auch maßgeblich die Spielregeln definiert.
In einem weiteren Projekt versucht Oosterhuis, die Interaktion von Artefakten durch eine Mechanisierung von Bauteilen herzustellen. Für eine Ausstellung in Paris agieren in einem Projekt namens MUSCLE 72 Pneumatikeinheiten als programmierbare Struktur. Parameteränderungen öffnen oder schließen Ventile und steuern so Druckluftzylinder, angelehnt an diverse Inputdiagramme. Oosterhuis beschreibt in diesem Sinn die Mechanisierung von Gebäuden als Zukunftsszenario des Entwurfsprozesses und logische Erweiterung des architektonischen Einflussbereiches (Vgl. Oosterhuis, 2004, S. 25).

> In other words, a Boid is not moving in an empty world, a Cellular Automaton cannot live as an isolated machine, Smart Dust particles do have contact with other systems. All machines feed on information, and all machines produce information of some sort. All machines are a small player in a complex structure of many interacting machines. But the necessity remains that in order to see the world from the next level, designers must start from simple rules placed in a complex environment rather than starting from a superficially complex structure without a clear concept of how to generate the data needed for customized production.
> In the end we must think of building and evolving networks relating all the different players in the dynamic process of the evolution of the 3d model. Each player in the process can be seen as having its own specific view on the data. (…) Each of them sends signals to the model which receives the signal, processes it and acts accordingly. From other disciplines the model would receive another class of signals leading to adjustment of the model for completely different reasons. In essence this awareness leads to a process of Collaborative Design and Engineering. All players in this process - people, materials, forces, algorithms, money and energy alike - are in their own way connected to the evolutionary 3d model. Each of them performing some simple set of rules, without

> complete awareness of what the other parties are doing or capable of. They all contribute from their own systems to the complex set of related systems as a whole. In this sense, even a traditional building process behaves like a swarm. But now we can learn from the new kind of science that we must build design processes on swarming intelligent particles in the Point Cloud communicating with each other. As humans we must learn to relate to the dynamics of super-fast real time computational processes. We must build the computational tools for Collaborative Design and Engineering in order to meet the rich expectations created by looking at the world from one or two levels up. (Oosterhuis, 2004, S. 23)

Im Zentrum der Überlegungen zur SAT steht das Computermodell des Entwurfs mit der ihm zugrunde liegenden Datenbank. Sie ermöglicht es allen Mitentwerfern auf die Daten zuzugreifen, um diese nach ihren Bedürfnissen darzustellen, auszuwerten und zu ändern. Auch bei diesem Modell wird ein einzelnes langes und komplexes Programm durch eine Menge einfacherer interagierender Programme abgelöst, und der hierarchische Zugang zur Codegestaltung durch einen distributiven ersetzt. Das Zusammenspielen aller Aktanten wird durch die digitalisierte Schnittstelle ermöglicht, und der Speicher stellt in diesem Szenario direkt die »belives« der menschlichen Aktanten dar. Auf diese Weise unterscheiden sich die einzelnen Mitglieder nicht in der Art, in der sie auf den digitalen Entwurf einwirken können. Zwar unterliegt die Übersetzung der Entwurfshandlungen in den digitalen Raum einer Drift, sie ermöglicht aber die Interaktionen der Aktanten transparent zu gestalten. Weil sämtliche Ereignisse aufgezeichnet werden können, sind auch ihre diversen Auswirkungen besser zu beobachten. Nicht nur das Werkzeuge für das digitale Entwerfen einfacher herzustellen sind, sie können auch dazu beitragen den Einfluss ihrer Entwickler zu protokollieren. Die Forderung an den Architekten selbst Werkzeuge zu programmieren, wie sie Oosterhuis hier aufstellt, um den Erwartungen an ein solches Entwerfen gerecht werden zu können, ist vernünftig. Mit den erwähnten Techniken der BDI-Agenten wird bei vielen Aufgaben der Unterschied zwischen möglichen Handlungen von Software im Vergleich zu denen von Menschen verschwindend gering. Was bleibt ist ein Kollektiv in dem plötzlich irrelevant ist, wer dem Prozessor die Arbeit zuweist. Sowohl menschliche Aktanten sowie nicht-menschliche arbeiten in ihren Welten bereits nicht-hierarchisch kollaborativ, die einzige Erweiterung besteht nun darin diese Welten miteinander zu kombinieren.

> Before we can generate buildings themselves, we must model the decision-making processes, that give rise to them. And in order to do this, we must be able to devise intelligent decision-making agents that can influence others and reflect upon their own decisions. (Leach & DeLanda, 2009, S. 55)

In einem Interview bremst Manuel De Landa allerdings die Euphorie durch hoch entwickelte Simulationen Gebäude generieren zu können. Nicht dass die SAT oder die Actor Network Theory dies beabsichtigen würden, bei beiden Ansätzen geht es nur um die Art und Weise der Zusammenarbeit, liegt in seiner Forderung aber ein interessanter Ansatz versteckt. De Landas schlägt vor, die Entscheidungsfindungen hinter den Entwurfsprozessen mit BDI-Agenten zu simulieren. Sämtliche Personen und Institutionen, die am Entwerfen aktiv, politisch oder finanziell beteiligt sind, sollten durch Programme dargestellt werden. Diese Verhaltenssimulation weist starke Parallelen zu denen auf, die Biologen und Zoologen verwenden um Bienen- und Ameisenvölker zu untersuchen. Abgesehen davon, dass das Fehlen von Übersetzungsvorgängen und die Sprünge im Bewusstsein durch gewonnene Übersichtlichkeiten auch mit modernen Verfahren zur Mustererkennung nur schwer realisierbar sind, liegt eines der Probleme auch in der Begrenztheit der Teilnehmer.

> So we need something like the type of agent known as Belief-Desire-Intention agents, who can not only make decisions based on their beliefs and desires, but also attribute to others such intentional states and use those attributions to modify their own decisions. With these agents, and some way of representing authority structures so that we can have binding centralized decisions, we could test the collective effect of many of them on the form of neighborhoods or even entire cities. (Leach & DeLanda, 2009, S. 54)

Andererseits verbindet De Landa das digitale Entwerfen mit der Suche nach den analogen Interaktionen der Teilnehmer um so von dem einen auf das andere schließen zu können. Der Rückschluss funktioniert nur unter der Annahme, dass mit der erwähnten Technik eine ähnliche Struktur im Digitalen realisiert werden kann wie im Realen. Bis auf das Vermischen der beiden Welten sind sich Oosterhuis und de Landa somit unabhängig voneinander einig, dass Agenten wie Menschen handeln können. Um die Frage nach den Interaktionen zu beantworten, ist der Ansatz des MACOSPOL[11] Projekts gut geeignet. Bei diesem Projekt werden die einzelnen Teilnehmer auch grafisch zueinander in Beziehung gesetzt, wodurch die entstehenden Netzwerkknoten Rückschlüsse auf die verschiedenen Einflüsse zulassen. Im Mittelpunkt der Aufmerksamkeit stehen bei dieser Methode–die menschlichen, institutionellen, sowie nicht-menschlichen Aktanten, und deren Einfluss auf den Diskurs. Nicht inbegriffen sind in der Analyse die restlichen am Entwurf beteiligten Aktanten, die sich nicht in den Diskurs einbringen.

11 MACOSPOL, Mapping Controversies on Science for Politics

> On the one hand, architectural knowledge advances very rapidly, with new types of materials and technological innovations entering the field and multiplying architectural invention. On the other hand, urban experts, architects, and engineers often publicly debate uncertain urban knowledge and technologies, polarizing opinion as witnessed on numerous architectural blogs, citizen fora and newspaper websites. The disputes continue even when an architectural competition is won, a building constructed, or a city-wide development plan implemented. This radical transformation in building technologies, in the reliance upon experts, and in the expansion of architectural networks could have remained practically invisible were it not for the presence of another phenomenon - the digitalization and availability of enormous databases on architecture (…). The digital technology at our command concerning a variety of buildings, both iconic and ordinary, planned or existing, built many years ago or still under construction, provide us with abundant resources to follow controversies surrounding design and architecture. (…) Controversy points to the series of uncertainties that a design project, a building, an urban plan or a construction process undergoes; it is rather a synonym of 'architecture in the making'.
> Mapping controversies means 'analyzing controversies' and covers the research that enables us to describe the successive stages in the production of architectural knowledge and artifacts, buildings and urban plans. By mapping controversies we refer to a variety of new representational techniques and tools that permit us to describe the successive stages of architectural controversies. For example, the rhythm, intensity and scope of the disputes; the dispersion of the actors' positions; the trajectory of their arguments; the timing and spacing devices; and the different ways of slowing down the pace of the controversy and closing it. (Yaneva, 2009)

Mit der Mapping-Methode kann der Fluss der Argumente im Entstehungsprozess eines Gebäudes dargestellt werden. Das dynamische System der Entscheidungsfindung der teilnehmenden Mitglieder wird dabei ebenso transparent, wie auch vormals unbeachtete Aktanten. Genauso, wie die Vorphase des Entwerfens betrachtet wird, ist es auch sinnvoll auf den Entwurfsprozess selbst zu blicken. Von den Skizzen über den wahrscheinlichen Einsatz digitaler Methoden bis hin zu den Interaktionen auf der Baustelle eignet sich das Zeichnen von Karten um die Einflüsse im Kollektiv zu protokollieren. Softwarekonzepte, die im digitalen Raum diese Aufgabe übernehmen sind bereits im Einsatz, und könnten in das Mapping miteinbezogen werden.

In der menschlichen und nicht-menschlichen Kollaboration hat weder ein Herrscher noch eine andere Art von Machtgefüge Bestand. In der realen und in der digitalen Welt wurde der Zwang durch den Einfluss verdrängt, welcher zumindest in der realen bereits wissenschaftlich dokumentiert wird. Dadurch, dass viele konstruierte Aktanten einen Teil des Entwurfskollektivs bilden, haben ihre Entwickler und Hersteller indirekten Einfluss auf das Endergebnis des Entwurfs. Zeichnete man nun eine Karte des Entwerfens, hätten

Architekten, die beim Schaffen der Software involviert gewesen sind unter Umständen mehr Einfluss auf den Entwurf, als solche die dies verabsäumt haben.

DIE MOBLISIERUNG: REFERENZEN ZIRKULIEREN LASSEN

Auf den vorhergehenden Seiten wurde bereits beschrieben, wie Entwerfen inzwischen wieder als kollektive Leistung verstanden wird. Anstatt, dass ein Architekt allein für die Planung zuständig ist, interagieren viele Aktanten im Entwurfsprozess. Diese Form der Zusammenarbeit ist auf ähnliche Weise bereits im Mittelalter als »Bauhütte« in Erscheinung getreten. In ihrer Frühgeschichte bestand sie aus von Mönchen ausgebildeten Arbeitern, die teilweise bewaffnet von Bauaufgabe zu Bauaufgabe zogen. Mit der Zeit schlossen sich Gewerke und Zünfte an, und selbst die Steinmetzbruderschaft integrierte sich. Rufen wir uns in Erinnerung wie Bauen damals organisiert wurde: Es war üblich, dass der Architekt den Bau vor Ort betreute und die Arbeiten einteilte und überwachte. Die langen Bauzeiten erforderten oft Wohnortswechsel aller Angehörigen der Bauhütte. Das betraf vom Werkmeister über die Steinmetze bis zu den Bäckern und Kirchendienern alle notwendigen Professionen und Zünfte. Eine räumliche Trennung zwischen Architekt und Baumeister war zu dieser Zeit nicht vorhanden, er konnte jeden Fortschritt vor Ort überwachen und direkt auf diverse Entwicklungen Einfluss nehmen.

> Zur Hütte gehörten der Werkmeister und die verschiedenen Handwerker, u. a. Steinmetz, Zimmermann, Maurer, Schmied und Glaser, soweit sie längerfristig auf der Baustelle erforderlich waren, sowie der Hüttenknecht und weiteres Personal zur Aufrechterhaltung des Kirchenbetriebes und für die Versorgung der Hütte; es konnte gelegentlich eine recht große Personenzahl sein, wie es für Straßburg und Wien im 15. Jh. überliefert ist: neben einem Kaplan für die seelsorgerische Betreuung der Hüttenmitglieder waren es nicht nur Küster, Meßner, Organist und Kirchendiener für den Kirchenbetrieb, sondern auch Bäcker, Koch, und Gesinde für den Haushalt der Hütte. (Binding, 1993, S. 102)

In diesem Umfeld suchte Leon Battista Alberti Möglichkeiten um seinen Entwurf unmissverständlich und unveränderbar kommunizieren zu können. Überzeugt davon, dass die Umsetzung des Entwurfes in Gebautes eine rein mechanische aber in keiner Weise eine anspruchsvolle Tätigkeit darstellt, wollte er sich davon befreien. Dadurch prägte er die Trennung zwischen dem Bauen und dem Entwerfen und in weiterer Folge das Berufsbild des Architekten, wie es später noch lange Zeit vorherrschen sollte. Verbunden

mit dieser Trennung war auch die Notwendigkeit der Übersetzung vom Entwurf in die bauliche Umsetzung; wobei ersterer so genau wie möglich dargestellt werden musste, um nicht allzu sehr vom gebauten Resultat überrascht zu werden. Für Alberti war mit dem Entwurf und seiner Darstellung dieser abgeschlossen, und nachträgliche Änderungen nicht im Sinne des Architekten.

> Diese Definition des Architekten wurde zum ersten Mal von Leon Battista Alberti in der Mitte des 15. Jahrhunderts aufgebracht, als dieser bekanntlich in seiner Abhandlung *Über die Baukunst* sagte, dass Architekten keine Dinge machen, sondern dass sie Dinge entwerfen sollten. Der Entwurf, ausgedrückt in Zeichnungen und Modellen, wurde so zum unausweichlichen Flaschenhals, durch den die meisten Gebäude in der westlichen (Alberti'schen) Tradition erreicht werden sollten: Zu diesem Zweck mussten die Bilder des Entwerfers in Gebäude übersetzt werden, doch aus Albertis Sicht war das nicht mehr Sache des Architekten. Im Alberti'schen Paradigma ist Bauen nämlich eine rein mechanische Operation, bar jeglichen intellektuellen Mehrwerts, deren einziger Zweck darin besteht, die Idee des Architekten zu materialisieren (…) Als unverzichtbarer Nachsatz zu dieser Definition bestand Alberti darauf, dass der Entwurf des Autors, (…) ohne jede Änderung physisch ausgeführt werden sollte. Die miteinander gekoppelte Vorstellung von der idealen Trennung zwischen Entwerfer und Konstrukteuren und der gleichfalls idealen Schnittlinie zwischen dem Ende des Entwurfs und dem Anfang von dessen physischer Reproduktion im Bauen, war revolutionär angesichts des kollaborativen Ansatzes im Bauen, der am Ende des Mittelalters weit verbreitet war. (Carpo, 2010, S. 26)

Bevor Alberti den Architekten aus dem Verbund der Bauhütten lösen konnte, musste er eine Alternative zur damaligen Form der Informationsübertragung finden. Wie Mario Carpo beschreibt, waren die zur Verfügung stehenden Mittel nur unter Vorbehalt zum Übermitteln von Entwürfen in Form von Text und Grafiken geeignet. Schreiber und Kopisten nahmen sich im Abschreiben von Text große Freiheiten, und noch größere im Abzeichnen von Bildern. Dieser Spielraum machte die Bemühungen um eine exakte Informationsübertragung zu Nichte, und damit auch die Chance auf eine Trennung von Bauhütte und Architekt. Da das Zirkulieren der Referenz ohne adäquaten Übertragungsmethoden nicht hergestellt werden konnte, wäre der Entwurf in Folge dessen nicht direkt mit dem Gebäude in Verbindung gestanden. Um das Entwerfen abstrakter (also auf einer höheren Referenzebene angesiedelt, und damit weiter von der realen Umwelt entfernt) behandeln zu können, war eine möglichst genaue, und fremden Eingriffen gegenüber resistente Übersetzung notwendig.

> Alberti war ein Humanist und als solcher mit den verfälschten Überresten antiker Texte vertraut; er wusste also zu genau, dass die Überlieferung von Texten und Bildern in Raum und Zeit mittels Manuskripten ein riskantes Unterfangen war. Kopisten machen

> Fehler, manchmal interpretieren sie auch den Text, schieben etwas ein oder erfinden ganze Textteile. (…) In ihrer Bewegung durch Raum und Zeit verwandeln und ändern sich Texte durch Zufall oder auch absichtlich von einer Manuskriptabschrift zu nächsten. (Carpo, 2009, S. 53)

Natürlich stellt das Problem der Mouvance, wie die Veränderung des mittelalterlichen Textes über mehrere Generationen hinweg genannt wird, eine extreme Form des Kopierverlustes beziehungsweise Kopierfehlers dar. Aber sie zeigt deutlich die Probleme die im Umgang mit analogen Quellen entstehen. Egal ob ein menschlicher oder ein nichtmenschlicher Aktant für das Kopieren zuständig ist, beim Übertragen von einer Vorlage auf ihre Kopie kommt es zu Verlusten und Fehlern[12]. Während die Eigeninterpretationen der Schreiber in den ihnen vorgelegten Texten noch halbwegs einfach zu sind, sind die Fehler, die durch das Abtasten analoger Vorlagen (egal ob Film oder Bauteilvorlage) entstehen weniger leicht zu eruieren. Man kann diesen Effekt bei der Elektrofotografie (umgangssprachlich »Fotokopie«) gut beobachten, und auch wie sich der Fehler durch Kopieren der Kopie verstärkt. Ein zweites Problem der analogen Kopie tritt vor allem beim oftmaligen Wiederholen des Vorgangs in Erscheinung. Die mechanische Beanspruchung der Vorlage führt unweigerlich zu ihrer Abnützung, und damit zu einer Veränderung der Informationsquelle. Es gibt Informationsquellen, die durch scheinbar geringe Veränderungen unbrauchbar gemacht werden, und verloren gehen. Information persistent zur Verfügung zu stellen ist mit einem analogen System wie der Fotokopie nicht möglich.

> Every time you copy an analog source, such as a video tape or a reel of film, it degrades quality. In fact, every time you even view certain analog sources (including both film and video), it suffers a loss of quality. This is because most analog-viewing devices are mechanical and have moving parts that can damage the source. (…) Each copy of an analog source introduces additional, lasting damage to the copy. A copy of a copy retains all the errors. (…) To maximize the quality, it's preferable to work from the original material whenever possible. (James, 2006, S. 91)

Alberti hat sich deswegen mit dem Digitalisieren der Information beholfen. Er nutze das Polarkoordinatensystem um grafischen Inhalt in Koordinatenlisten übersetzt zu übertragen. Dabei beinhalteten der Text mit den Listen auch die Anleitung zum Rückübersetzen

12 Fehler die beim „Fotokopieren" auftreten können sind zum Beispiel Rauschen, falsche Wiedergabe von Grauwerten, Verzerrungen oder mitkopierter Schmutz.

der Koordinaten in eine Zeichnung. Auch wenn Mario Carpo schreibt, die Liste ermögliche neue Originale herzustellen, ist das nur die halbe Wahrheit. Alberti hat eine Darstellungsform entworfen, die die Zeichnung nur referenziert. Das eigentliche »Original« bildet damit die Liste der Koordinaten, die selbst wiederum Übertragungsverlusten ausgesetzt ist. Und davon gibt es bei diesem Verfahren einige wahrscheinliche, wie Messfehler, Zahlendreher, Fehler bei der Herstellung der Mess- und Übersetzungsvorrichtung oder die Ungenauigkeit beim Ablesen des Skala (Parallaxenfehler[13]). Die aufwändige Übersetzung war der Preis den Alberti für eine verschlüsselte Übertragung zu zahlen bereit war.

> Da er seine Zeichnungen nicht in Worte übertragen konnte, erfand er (Alberti, Anm.) ein Verfahren, um sie in Zahlen zu übersetzen. In der »Descriptio Urbis Romae«, einem kurzen lateinischen Werk, erklärt Alberti, wie er zunächst den Plan aufgemessen und anschließend das Bild anhand eines Polarkoordinatensystems »digitalisiert« hat. Albertis Buch besteht im Weiteren aus einer Liste von Zahlen; und er erwartet von seinen Lesern (…) die Bilder identisch oder zumindest proportional identisch zum Original wiederherzustellen, indem ein spezifisches Instrument diese Zahlen verarbeitet, das Alberti ebenfalls beschreibt, und das wir heutzutage einen Plotter nennen würden. (Carpo, 2009, S. 55)

Mit der beschriebenen Methode konnten nun Anweisungen von Alberti an die, die den Entwurf in Gebautes umsetzen sollten, mit relativ wenig Fehlern übertragen werden. Dabei muss allerdings festgehalten werden, dass dies auf die Umsetzung des Entwurfs in Gebautes selbst keinen Einfluss hatte. Es handelt sich hierbei um zwei verschiedene Referenzierungssprünge, zum einen von der Skizze in das Koordinatensystem und zurück, und zum anderen von der Skizze zum Produkt. In einer Variante der angewandten Technik konnten schließlich auch dreidimensionale Objekte vermessen und referenziert werden. Auf diese Weise gelang es ihm, Listen als Referenzen zu allen erdenklichen Körpern herstellen zu können. Alberti erreichte damit das Ziel, Bauanweisungen so exakt formulieren zu können, dass sogar die einzelnen Teile einer Statue in verschiedenen Werkstätten gefertigt werden konnten. Eine Abwandlung seiner Technik wird noch heute in der industriellen Messtechnik wie im Maschinenbau verwendet. Dabei werden ebenfalls durch ein mechanisches Abtasten einer Vorlage die erhaltenen Koordinaten in ein numerisches Modell übertragen (Vgl. Benbow, Elshennawy, & Walker, 2003, S. 100).

13 Perspektivischer Beobachtungsfehler, tritt auf wenn kein rechten Winkel zur Skala des Messgerätes eingenommen wird.

> Wie in »Descriptio Urbis Romae« ist die zentrale Apparatur in Albertis »De Statua« ein drehbares Instrument, genauer, ein skaliertes Rad, in diesem Fall etwas unkonventionell auf dem Kopf des zu digitalisierenden Körpers befestigt. Die daraus resultierende Zahlenreihe schafft durch ihre Notation die Möglichkeit, den originalen Körper unendlich oft zu kopieren und irgendwo und irgendwann maßstabsgetreu oder in anderen Proportionen zu reproduzieren. Alberti schlug auch vor, mittels der gleichen Technik verschiedene Teile der gleichen Statue in unterschiedlichen Werkstätten gleichzeitig anfertigen zu lassen. (Carpo, 2009, S. 56f)

Den Einfluss anderer auf den Entwurf und seine Umsetzung zu reduzieren, war besonders bei einem Bau Albertis notwendig: dem Tempio Malatestiano in San Francesco. Dieses Bauvorhaben leitete er von Rom aus, großteils mit schriftlichen Anweisungen. Vor Ort war Baumeister Matteo de'Pasti für das Projekt verantwortlich, und Empfänger wie Befehlsausführer. Alberti wollte mit der entwickelten Übertragungstechnik fremden Einfluss auf den Entwurf unterbinden, und hat damit eine Hierarchie etabliert. Der Architekt war für den Entwurf zuständig geworden, was zwar auch schon vorher der Fall war, aber nun stand er alleine mit dieser Aufgabe da. Mit der Referenzierbarkeit der Zeichnung hatte er ein Werkzeug zur Hand, das ihn in die Lage versetzte, den Einfluss der Schreiber und Kopisten auf ein minimales Maß zu reduzieren. Seine genauen Anweisungen versetzten ihn in die Lage, auf eine Überwachung der Arbeiten vor Ort verzichten zu können, die Referenz machte also eine geografische Nähe überflüssig.

> Albertis Aufgabe war, die bestehende, dem hl. Franziskus geweihte, gotische Kirche zu einem Mausoleum umzugestalten, in dem Sigismondo Malatesta und seine Geliebte Isotta, umgeben von humanistischen Gelehrten ihres Hofstaates, einst bestattet werden sollten. Die Kirche, die deshalb auch als Tempio Malatestiano bezeichnet wird, ist nie vollendet worden. Aus zeitgenössischen Quellen ist aber der Umfang von Albertis Gesamtplanung bekannt. Da Alberti seit 1443 in Rom im Dienst des Papstes stand, übernahm der Bildhauer Matteo de' Pastis für ihn die Bauleitung in Rimini. Aus brieflichen Anweisungen Albertis und aus einer von de' Pastis 1450 angefertigten Medaille geht hervor, dass die alte Kirche mit einer neuen Schale vollständig ummantelt werden sollte. (Nerdinger, 2002, S. 77)

Wie uns die Geschichte lehrt, war der Bau der Kirche in San Francesco aus Albertis Sicht nicht wirklich gelungen. Die von ihm entworfene Marmorfassade blieb unvollendet, während sich d'Pasti massiv in den Entwurf einbrachte. Alberti scheiterte, obwohl er mit seinem Verfahren seiner Zeit weit voraus war. Schließlich führte ein ähnlicher Ansatz in

der Mitte des zwanzigsten Jahrhunderts zur Erfindung der CNC[14] Maschine. Albertis Dilemma war, sich entweder nicht vom Bauprozess befreien zu können oder wegen der fehlenden technischen Möglichkeiten auf eine exakte Umsetzung seiner Vorgaben zu verzichten. Seine Bemühungen genaue Kopien herstellen zu können entstanden aus dem Problem, dass Schreiber den Inhalt nicht genau reproduzierten. Mit dem Übersetzen in eine andere Darstellungsform, aus welcher der Inhalt der Nachricht abgeleitet werden konnte, hatte er einen Lösungsweg eingeschlagen, der sich in der Geschichte der Architektur seitdem einige Male wiederholt hat.

Einige Seiten zuvor wurde bereits eine Verbindung von Architektur und Flugzeugbau erwähnt. Die CAD Software CATIA wurde ursprünglich zur Konstruktion von Luftfahrzeugen entwickelt, und Jahre später von Architekten zum Entwerfen von Gebäuden wie dem Guggenheim-Museum in Bilbao verwendet. Wie sich zeigt, hat Flugzeugindustrie mit der CNC-Maschine ein weiteres Werkzeug hervorgebracht, das gerade in der jüngeren Vergangenheit das Entwerfen immer stärker beeinflusst. Die Geschichte der CNC-Maschine beginnt, als die junge United States Airforce[15] ein neues Flugzeug bauen lassen will, und sich mit diesem Vorhaben an die renommierte Firma Lockheed wendet. Diese nutzte, um die für die Tragstruktur der Flügel notwendigen Stringer (der Bauteil liegt zwischen Holm und Haut) herzustellen, eine neu angeschaffte fünfachsige Schablonenkopierfräsmaschine[16]. Die bereits erwähnten Probleme dieser Art von Technik führten schlussendlich dazu, dass die Maschine für das Erreichen der aerodynamischen Anforderungen an die fertigen Tragflächen zu ungenau arbeitete. Es gab bereits Probleme eine in die Toleranzen passende Schablone zu fertigen, in Kombination mit den Kopierverlusten kann man erahnen, dass die Endergebnisse nicht zur Zufriedenheit ausfallen konnten (Vgl. Olexa, 2001).

Etwa zeitgleich arbeiteten John T. Parsons und Frank Stulen an der neuen Fertigungsmethode die »by-the-numbers method« später auch die »Cardamatic Milling Machine« genannt (Vgl. Jain & Chitale, 2010, S. 460f). Sie produzierten Holme für Helikopterrotoren, deren Geometrie vom Auftraggeber durch siebzehn Messpunkte definiert wurde. Diese Art der Informationsübertragung erinnert stark an die Koordinatenlisten Albertis. Die Menge der Punkte, und folglich auch die Auflösung ihrer Darstellung, war zu gering

14 CNC, Computer Numerical Control

15 Gegründet am 18. September 1947, zuvor Teil der US Army

16 Bei dieser Form des analogen Kopierens wird eine Vorlage mit einem Fühler abgetastet, und mechanisch in die Bewegung des Werkzeugs übertragen. Diese Art der Maschine ist mit einem zwischengeschalteten Computer noch heute in Verwendung (Vgl.Menning, 2008, S. 59).

um sie direkt für die Fertigung nutzen zu können. Parsons musste deshalb die Punkte mit einem Kurvenlineal zu einer Kante verbinden, um sie anschließend auf das Werkstück übertragen zu können. Dieses Verfahren war zum einen extrem aufwändig, und zum anderen eine potenzielle Fehlerquelle.

> To define an airfoil template, they gave us 17 points between the radii on the upper and lower surfaces. The coordinate points were different for each of the two surfaces. Then you had to take a French curve and connect those points. It wasn't accurate, and you didn't know the accuracy of the French curve between the coordinates. So I asked Stulen if he could use his college education, something I don't have, and give me 200 points along the radius of each surface, which he had no problem doing. He made up a chart describing X axis and Y-axis coordinates for a milling machine. Then, using a Bridgeport mill, we put one man on the left-to-right axis, and one man on the in-and-out axis. (Olexa, 2001)

In Parsons Firma war zu dem damaligen Zeitpunkt Frank Stulen beschäftigt. Diese Zusammenarbeit brachte Parsons auf eine neue Idee: Stulen war einer der Wenigen weltweit, die im Umgang mit Lochkartenrechnern geübt waren. Das, und der Zugang zu einem derartigen Gerät, ermöglichte ihnen, einen alternativen Lösungsweg zum Verbinden der Punkte zu entwickeln. Sie übersetzten die siebzehn Punkte in mathematische Formeln und fütterten die Rechenmaschine mit den Daten um eine zweihundert Punkte lange Fräsanweisung zu generieren. Damit ausgerüstet setzten sie an der Maschine drei Männer zum Ausführen der Fräsanweisung ein: einen zum Einstellen der X-Koordinate, einen zum Einstellen der Y-Koordinate und den letzten zum Vorlesen der Punkte. Dadurch war es ihnen gelungen, engere Toleranzen in der Übersetzung von Entwurf zu Werkstück einzuhalten (Vgl. Olexa, 2001). Wie Alberti nutzten sie die Möglichkeit einer höheren Referenzebene, um die Anweisung genauer transportieren zu können. Sie übersetzten die Punkteliste in eine mathematische Funktion, aus der sich wiederum Punkte ableiten ließen. Sie verlängerten damit Albertis Referenzkette am abstrakten Ende, aber damit nicht genug: Der Einsatz der Fräsmaschine, die durch drei Achsen (ihre technische Ausführung würde inzwischen auch mehr erlauben) definierte Eingriffe ausführen konnte, schränkte die Freiheiten im Interpretieren der Punkteliste enorm ein. Es gab zwar noch die eingesetzten Maschinenführer als »Schwachstellen«, aber ihr Ende war bereits eingeläutet. Schlussendlich waren es die Achsen der Maschine, die die Referenzkette auch zur »Realisierung« hin verlängern sollten.
Obwohl Parsons Methode von Lockheed abgelehnt wurde, bekam er von der USAF 1949 das Kapital seine Maschine entscheidend zu verbessern. Sein Ziel wurde es, die Arbeiter

zwischen Rechenmaschine und Fräsmaschine überflüssig zu machen. Mit engeren Koordinatenabständen und viel längeren Anweisungslisten wollte er mit seinem Verfahren noch genauer werden. Seine Fortschritte gerieten aber ins Stocken, und so wendete er sich an die Experten vom MIT. Mit ihrer Hilfe gelang eine entscheidende Verbesserung: anstatt aus den Funktionen Punkte zu generieren, nutzten sie Servomotoren um den Fräskopf zu bewegen. Der Qualitätssprung war gewaltig, und die zeitliche Einsparung in der Handhabung der Maschine gelungen (Vgl. Noble, 1995).
Die Referenzkette war mit ihrer mittelalterlichen Verwandtschaft nicht mehr zu vergleichen. Ermöglichte sie einst nur die Informationsübertragung des Entwurfs, konnte sie nun bis zum Werkzeug selbst vordringen. Die Übersetzung der Formeln in Bewegung anstelle von Koordinaten die einzeln angesteuert werden mussten, machte die seit Alberti bekannten Listen obsolet. Der politische Aspekt dieser Entwicklung ist dabei nicht nur nicht zu unterschätzen, sondern war eine der Grundintentionen der dahinter stehenden Geldgeber. Denn ganz in ihrem Sinne wurde die Arbeitsleistung von den Maschinenführern aus den gewerkschaftlich organisierten Betriebshallen in die nicht organisierten Büros wegübersetzt. Mit der Computersteuerung wurde die Arbeitswelt nachhaltig verändert, wie zuvor in der Industrialisierung mit Webstühlen[17] und vielen anderen Anlagen, aber im Gegensatz zur Delegation von Arbeit handelte es sich jetzt sogar um das Umverteilen von Kompetenz. Dank dieses Umstands wurden Maschinenführer zu Maschinenbeschickern, und verloren den Großteil ihres Einflusses auf den Gestaltungsprozess.

> The former technology retained the skill of the machinist in the labor process; the latter largely eliminated the need for that skill. (Boreham, Parker, Thompson, & Hall, 2008, S. 23)

Mit der automatisierten Steuerung wurde die NC-Maschine einige Jahre später zwar einsatzbereit, der Sprung in die Produktionsstätten blieb ihr zunächst aber dennoch verwehrt. Ihr großes wirtschaftliches Manko war, dass sämtliche zeitlichen Einsparungen im Fräsen durch den Mehraufwand im Programmieren zu Nichte gemacht wurden. Das genauere Fertigen wurde zwar erreicht, und die Arbeitszeit aus dem Einflussbereich der Arbeiter wegrationalisiert, aber stattdessen hin zu einer teureren Kostenstelle. Der Aufwand im Programmieren, der nun in den (zwar gewerkschaftlich nicht organisierten)

17 Auf die Automatisierung der Zeit der Industrialisierung ist auch ein Großteil sozial- und arbeitspolitischer Fragen und Konzepte zurückzuführen. (Vgl.Pfahlmann, 1974, S. 49ff)

Büros hohe Kosten verursachte, fraß sämtliche finanziellen Vorteile der neuen Technik auf. Das NC-Konzept war damals für die Hersteller so befremdlich und unwirtschaftlich, dass es aus eigener Kraft maximal den Sprung in die Schubladen schaffen konnte. Erst die USAF hat den Stein wieder ins Rollen gebracht, indem 120 der Maschinen in Auftrag gegeben wurden, nur um sie günstig vermieten und weiterverbreiten zu können. (Vgl. Holland, 1989, S. 34f)

> From the start of the late 1940s down to the present day, the air force has been and remains the major sponsor of industrial automation. With regard to numerical control, the air force underwrote the first several decades of research and development of both hardware and software, determined what the technology would ultimately look like by setting design specification and criteria to meet military objectives, created an artificial market for the automated equipment by making itself the main customer and thereby generating demand, subsidized both machine-tool builders and industrial (primarily aerospace) users in the construction, purchase, and installation of the new equipment, and even paid them to learn to run it. (Noble, 1995, S. 83)

Ohne dieser Intervention wäre Parsons Konzept ähnlich wie Albertis vermutlich nicht weiter verfolgt worden. Als Insellösung für Spezialaufgaben konnte sie angewandt werden, aber für den großflächigen Durchbruch brauchte es mehr. Deswegen schuf die USAF einen künstlichen Markt, und mit erheblichem finanziellem Aufwand gelang es ein Netzwerk von Betrieben, die die NC-Technologie nutzten, aufzubauen. Durch die gestiegene Nachfrage begünstigt, wurden wirtschaftlichere Möglichkeiten der Eingabe gesucht. Fünf Jahre[18] nach der ersten kommerziellen NC-Maschine, kam 1959 als Lösung für das Eingabe Problem eine dafür entwickelte Programmiersprache auf den Markt. Von nun an ging es mit der Verbesserung der Technologie rasant voran, neben der standardisierten Programmiersprache ermöglichte auch die Anbindung an CAD Software immer einfacheres und schnelleres Erstellen von Anweisungen für die Maschinen. Die CNC-Maschine und die Entwicklungen in ihrem Feld dehnten die Referenzkette immer weiter in die abstrakten Ebenen aus, und vernetzten diese immer mehr. Heute wissen wir, dass hinter der beschriebenen Entwicklung starkes politisches Interesse gesteckt hat. Das allgegenwärtige Schreckgespenst des Kalten Krieges hat damals dazu geführt, dass der Kommunismus und mit ihm die ganze Arbeiterbewegung diskreditiert wurden. Das Militär wollte mit der

18 1954 erste kommerzielle NC-Maschine der Bendix Corporation, 1959 Veröffentlichung der APT Programmiersprache des MIT, 1980er erste grafische Programmieroberflächen, 1997 OMAC Systeme ersetzen Firmware Controller und ermöglichen auf dem PC (Windows/NT) zu programmieren. (Vgl.Jain & Chitale, 2010, S. 461)

CNC-Maschine Folgen ähnlich denen des automatisierten Webstuhls provozieren, und die Fertigung ausfallsicherer aufbauen. Es wollte keine Unruhen schüren, sondern das Risiko des Arbeitskampfes minimieren. Also entmachtete es mit der Verlegung des Aufwands aus den gewerkschaftlich organisierten Fertigungen in die nicht organisierten Büros die Arbeiterklasse.

> The effects of this military involvement reflect the peculiar characteristics of the military world. First and most obvious is the military emphasis upon command, the quintessence of the authoritarian approach to organization. This means, essential, that subordinates must do as they are told, with no ifs, ands, or buts; the intent is to eliminate wherever possible any human interventions between the command and the execution. (…) In the military outlook, an army of men behaving like machines is readily replaced by an army of machines. This command orientation neatly complements and powerfully reinforces the managerial obsession with control. If the business suit and the uniform are interchangeable in our day, so too are the minds that go with them. (Noble, 1995, S. 83f)

Dabei kommt die Gesinnung des Militärs noch klarer zum Ausdruck als bereits jene Albertis: Es sollte eine Befehlskette in der Fertigung etabliert werden, um die Struktur des Militärs zu übertragen. Alberti wollte Herr seines Entwurfes sein, und Befehle an sein Heer aus Arbeitern geben, die diese unverändert umzusetzen gehabt hätten. Die USAF ging noch einen Schritt weiter: Sie wollten sämtliche Eingriffsmöglichkeiten zwischen dem Befehl und seiner Ausführung für den Menschen unterbinden. Aus Angst die Gewerkschaft könnte die Rüstungsproduktion zu Kriegszeiten lahmlegen, wurde die CNC-Maschine zur politischen Machtfrage. Eine Armee williger Automaten, die nicht Gefahr laufen konnten zu rebellieren schien eine verlockende Option. Man glaubte ernsthaft daran, mit dem Digitalisieren von Information Einflüsse auf die Kette zu reduzieren. Diese Direktheit, Disziplin ohne Wenn und Aber, die Herrschaft nicht nur über den Entwurf, sondern über die ganze Produktion zu stellen, das alles wurde zu einem militärischen Interesse. Und daher erklären sich die Unsummen, die ein Umstellen der amerikanischen Fertigungsstätten in die Wege leiteten. Der wirtschaftliche Hintergrund der Automatisierung war einem militärischen gewichen. Die Referenzkette konnte von dieser Unterstützung nur profitieren. Das CAM[19] im Auge, bemühten sich und bemühen sich noch heute viele Entwickler, die nahtlose Integration bis tief in die Designprozesse herzustellen. Die technischen Möglichkeiten holten aber bald die Vordenker der ameri-

19 CAM, Computer Aided Manufactoring

kanischen Luftwaffe ein, denn mit der weltweiten Vernetzung der Computer konnten sich die Referenzen auch horizontal verbinden.

> Das CAD-CAM der 1990er basierte vorwiegend auf kontrollierten, zugangsbeschränkten Netzwerken und der Schwerpunkt lag auf der vertikalen Integration aller unterschiedlichen Design- und Produktionsphasen und des Potenzials, das die nahtlose Kontinuität des Entwurfs von singulären, komplexen Objekten für die Produktion von seriellen Variationen bot. Aber in den letzten Jahren hat sich die Netzwerkumgebung von den früheren, überwiegend monodirektional ausgerichteten Informationstechnologien zu voll symmetrischen, bidirektional ausgerichteten Informationssystemen entwickelt. Diese technische Entwicklung wird für eine Vielzahl von Zwecken genützt, manche davon sind rein technologischer Natur (wie P2P und verteilte Verarbeitungsnetzwerke), andere besitzen ungeheure soziale Implikationen – deren Software wird oft zutreffend »kollaborative« oder sogar »soziale« Software genannt. So wie jeder Knoten im Netz zugleich ein Empfänger und ein Sender von Information sein kann, haben viele Benutzer begonnen, das Netz auch ebenso zu verwenden. (Carpo, 2010, S. 23)

Die Verbindungen die die Digitalisierung geschaffen hat, funktionieren im Gegensatz zu militärischen Befehlsketten nicht nur in eine Richtung. Informationssysteme haben sich von der Sender (Transmitter) – Empfänger (Receiver) Dichotomie weg, hin zum Transceiver als Hauptelement entwickelt, und damit einen Umbruch in der Systematik eingeläutet. Peer-to-Peer Netzwerke, in denen alle Teilnehmer gleich gewichtet, nicht hierarchisch miteinander verbunden sind, unterlaufen so die ursprüngliche Intention des Militärs. Nimmt man auch noch die Trennung zwischen Subjekt und Objekt aus dem Spiel, wird das Entwerfen über die Digitalisierung zu einem kollektiven Unterfangen wie einst die Bauhütte vor Alberti. Unterstützt durch die lange Kette an Referenzen, und die große Anzahl an möglichen Verbindungen hat sich dieses Kollektiv inzwischen etablieren können. Wenn es das Ziel des US-Militärs war, die Herrschaft über die Produktion zu etablieren, und diese gegen äußere Einflüsse abzuriegeln, hat sich die dafür entwickelte Technik gegen ihre Herren gewandt. Denn durch die digitalisierten Schnittstellen ist es inzwischen mehr Aktanten möglich am Prozess teilzunehmen denn je. Die Luftwaffe hat einen beachtlichen Aufwand betrieben, um einen neuen Mitspieler zu etablieren, nur um bemerken zu müssen, dass dieser nach anderen Regeln spielt als gewünscht[20]. Natürlich

20 Die Parallele zur US Taktik bezüglich Osama Bin Laden und den Mudschahiddin drängt sich förmlich auf, wurden sie doch auch erst durch die ausgiebige Militär- und Finanzhilfe über die CIA zu einer ernst zu nehmenden Bedrohung für die sowjetischen Besatzer (Vgl.Randal, 2005, S. 70-74) um sich später gegen ihre früheren Fördere aufzulehnen.

konnten sie damals die Langzeitfolgen ihrer Strategie nicht abschätzen, sind diese den ursprünglichen doch diametral entgegengesetzt.

> Die partizipatorische Natur des Entwurfsprozesses, der von der digital gestützten horizontalen Integration nun unterstützt und gefördert wird, evoziert die gleiche kollektive und häufig anonyme Art des Bauens wie in den mittelalterlichen Bauten vor der humanistischen Revolution. (Carpo, 2010, S. 28)

Wenn sich die Struktur des Bauprozesses durch die hoch entwickelte Digitalisierung wieder zu einer »Hütte« formt, so wie vor Alberti, dieser aber mit dem gleichen Ansatz lange Zeit davor das genaue Gegenteil durchsetzen konnte, muss es doch Unterschiede zwischen den beiden so ähnlichen Entwicklungen geben. Beide Ansätze fußen auf dem Referenzieren von Daten in eine andere Form der Darstellung. Und beiden ist die Koordinatenliste gemein, obwohl sie bei dem einen Referenz und bei dem anderen Referenziertes ist. Die Überlegungen die Bruno Latour zu dem Thema angestellt hat, sind womöglich auch bei diesem Aspekt der Entwurfsbetrachtung hilfreich. Sein Ausgangspunkt ist natürlich wieder der eines wissenschaftlichen Diskurses und er definiert, wie sich eine Aussage durchsetzen kann:

> Wer gewinnt in einer agnostischen Begegnung zweier Autoren sowie zwischen ihnen und all jenen, die sie dazu brauchen, um eine Aussage A aufzubauen? Antwort: Derjenige, der in der Lage ist, *am schnellsten die größte Anzahl gruppierter und treuer Alliierter aufzubieten*. (…) Ich behaupte, dass Schreiben und bildliche Darstellung nicht selbst die Veränderung in unserer wissenschaftlichen Gesellschaft erklären können, sondern dazu *verhelfen, diese agonistische Situation günstiger zu gestalten*. (…) Wir sollten uns lieber auf jene Aspekte konzentrieren, die beim Anbieten, der Präsentation, der Zunahme, der effektiven Gruppierung oder der Rückversicherung der Treue neuer Verbündeter helfen. (Latour B. , 2006, S. 264)

Nach Latour hat jener Erfolg, der am schnellsten die meisten Verbündeten findet. Im Kollektiv der Hütte ist das leicht nachzuvollziehen, und auch im digitalen Kollektiv lassen sich Fürsprecher (oder abstrakter: Ermöglicher) ausfindig machen. Aber wie schneidet in diesem Vergleich Albertis Trennung ab? Ihr war immanent, dass der Entwurf nur von Alberti selbst verfasst wurde. In der Art und Weise wie man den Prozess nach der Actor Network Theory betrachten kann, findet man auch noch die anderen, die ihn dabei unterstützen, und deren Name nicht auf den autorisierten Papieren steht. Zum einen hat die Gerätschaft zum Übersetzen in Polarkoordinaten ihren Teil beigetragen, und zum anderen der Schreiber. In weiterer Folge war Alberti durch seine Abwesenheit dem Trei-

ben an der Baustelle machtlos ausgesetzt, konnte er auf Problemstellungen doch nicht mit angemessener Geschwindigkeit reagieren. Auch seine Annahme, mit dem Entwurf alle Eventualitäten prognostizieren zu können ist alleine schon durch einen einzigen Übersetzungssprung gefallen. Die Anzahl der Mitglieder des Kollektivs war auch in Albertis Umsetzung größer als eins. Durch die von ihm konstruierte Trennung hat er aber nicht wie geplant an Macht gewonnen, sondern durch seine Isolation an Einfluss auf die Entstehung des Gebäudes verloren[21].

Hat diese Isolation, da sie ja systematischer Natur zu sein scheint, auch bei der Entwicklung der CNC-Maschine existiert? Ja. Zwar waren die Einflussmöglichkeiten auf das Werkstück nur gering variabel, aber der Programmierer war von der Fertigung abgeschnitten. Anhand zahlreicher Normierungen konnte man das Verhalten von Werkstoff, Werkzeug und Maschine in großen Bereichen vorhersagen, aber eben nicht exakt. Welche Auswirkungen falsche Einstellungen, Anzeigen oder Manipulation der Steuerung haben können, zeigt ein Beispiel aus der jüngeren Geschichte. Unbekannte Urheber haben mit einem Computerwurm die Anzeigen zur Verfahrenskontrolle im iranischen Urananreicherungsprozess, im speziellen die der Drehzahlen der Zentrifugen, so manipuliert, dass die Maschinenführer diese Drehzahlen unwissend in einen kritischen Bereich steuerten (Vgl. Matrosov, Rodionov, Harley, & Malcho, 2011, S. 5). Wäre der Maschinenführer oder Programmierer direkt zum Ausführen und nicht nur zum Betreuen der ausführenden Maschine eingesetzt, könnte eine solche Manipulation nicht funktionieren. Die Digitalisierung hat in Kombination mit der Vernetzung zu einer schnelleren Verfügbarkeit von Empfängern geführt. Albertis Verbindung zum Bauprozess war hingegen langsam und umständlich, das war also das erste Problem.

> Wenn man von *seinem* gewohnten Weg abweichen und schwer beladen zurückkehren möchte, um andere dazu zu zwingen, *ihre* gewohnten Wege zu verlassen, besteht das hauptsächlich zu lösende Problem in der *Mobilisierung*. Man muss fortgehen und mit den »Dingen« zurückkehren, wenn die Bewegungen nicht vergeblich sein sollen; die »Dinge« müssen aber in der Lage sein, die Rückreise zu überstehen, ohne Schaden zu nehmen. (…) Kurz: Man muss Objekte erfinden, die mobil, aber auch *unveränderlich, präsentierbar, lesbar* und miteinander *kombinierbar* sind. (Latour B. , 2006, S. 266)

21 Man könnte jedoch Einwerfen, dass das Gebaute dem Entworfenen nicht entsprechen müsste, da es sich hier um zwei verschiedene Aufgabengebiete handelt. Meinte man, die Kontrolle über das tatsächlich Gebaute ist erst seit seiner Darstellung in den Medien von Interesse, frage ich, warum Alberti dann so versessen auf die Herstellung exakter Kopien gewesen ist.

Ein zweiter Punkt, der schlagend wird, ist die Informationssicherheit. Hier kommt ohne Frage der Aspekt der Mouvance zum Tragen mit dem Alberti Zeit seines Lebens zu kämpfen hatte. Seine Nachrichten waren durch das Abschreiben nicht vor Einfluss geschützt, und deshalb weit davon entfernt unveränderlich zu sein. Er konnte nicht davon ausgehen, dass seine Entwürfe in ihrer abgesendeten Form auch den Empfänger erreichen würden. Nicht nur, dass er einen Weg gefunden hatte, die Nachrichten zu verschlüsseln, seine Technik hat ihn auch in die Lage versetzt, die Information genauer darzustellen. Alberti hat zwar ein Verfahren zur Übersetzung entwickelt, aber keine geeignete Übertragungsmethode dafür gehabt. Er war nach wie vor auf Schreiber angewiesen, die zwar sicher wenig Kreativität im Verändern von Zahlen gesehen haben, dies aber immer noch tun konnten.
Ein Schwachpunkt der auch bei der »by-the-numbers«-Methode von Parsons zu finden war. Während der Druckvorgang der Liste bereits automatisiert war, blieb ihr Vorleser eine potenzielle »Fehlerquelle«. Erst durch den Einsatz von Servomotoren konnte die Übertragung direkt von der Quelle zum Empfänger, ohne die Notwendigkeit eines Mittelsmannes erfolgen. In weiterer Folge wurde auch die Übersetzung von Zeichnung zu Programm automatisiert, und so ein weiterer menschlicher Aktant aus der Gleichung genommen. Dies wurde durch die elektronische Übermittlungstechnik erreicht, mit der die digitalen Objekte mobil wurden. Weitere technische Fortschritte ermöglichten schlussendlich auch Daten je nach Bedürfnis des Empfängers aufzubereiten und darzustellen, ohne die Daten selbst zu modifizieren.

> Unveränderbarkeit wird durch den Prozess des Druckens vieler identischer Kopien, sichergestellt, Mobilität durch die Anzahl der Kopien, das Papier, die beweglichen Lettern. Die Verbindung zwischen verschiedenen Orten in Zeit und Raum werden von dieser fantastischen Beschleunigung unveränderlich mobiler Elemente, die irgendwo in allen Richtungen Europas zirkulieren, vollständig modifiziert. Wie Ivins gezeigt hat, ist Perspektive plus Druckerpresse plus *Aqua Forte* die wirklich wichtige Kombination, da Bücher nun die realistischen Bilder dessen, worüber sie sprechen, bei sich tragen. Zum ersten Mal kann eine Örtlichkeit andere, in Raum und Zeit weit entfernte Orte akkumulieren und sie dem Auge synoptisch präsentieren; diese synoptische Präsentation kann (…) an anderen Plätzen verbreitet und zu anderen Zeiten verfügbar gemacht werden. (Latour B. , 2006, S. 272)

Eine Technik die in ihren Grundzügen Alberti zur Verfügung hätte stehen können, das gedruckte Bild, verfügte damals bereits über einen Großteil der notwendigen Eigenschaften, die für das Mobilisieren von Entwürfen notwendig sind. Die Verbindung von Druckerpresse und Perspektive ist vergleichbar mit der von Digitalisierung und elektroni-

scher Übertragung, anstelle von Büchern werden dabei Datenpakete durch Leitungen geschickt. Selbst wenn Alberti bereits Bücher zum Übermitteln seiner Koordinatenlisten genutzt hätte, wäre er mit seiner Referenzkette nur bis zum Arbeiter und nicht bis zum Werkzeug vorgedrungen. Möglicherweise gründet der Erfolg von Parsons Konzept auf diesem Unterschied. Alberti konnte zwar aufgrund seines Ansatzes ein Verfahren zum Abmessen von Statuen entwickeln, die Umkehrung dieses Prozesses wäre mit den damaligen Technologien aber nicht möglich gewesen. Erst die Entwicklung der Fräsmaschine[22] hat es geschafft, ein dem Abtasten verwandtes Verfahren der Bearbeitung umzusetzen. Damit war einer symmetrischen Datenübertragung Tür und Tor geöffnet.

> Wenn wir die agonistische Situation betrachten, die ich als Referenzpunkt verwende, wird die Notwendigkeit, die so etwas wie die Druckerpresse begünstigt, klarer. *Alles*, was die Mobilität der Spuren, die eine Örtlichkeit über einen anderen Ort erhält, beschleunigt, oder *alles* was diesen Sprung gestattet, sich ohne Transformation von einem Ort zu einem anderen zu bewegen, wird favorisiert. (…) Das Privileg der Druckerpresse kommt von ihrer Fähigkeit, vielen Innovationen dazu zu verhelfen, auf einmal zu agieren, aber sie selbst ist nur eine Innovation unter vielen, die helfen, die einfachste Frage zu beantworten: Wie ist die Dominanz in großem Maßstab möglich? Diese Umformulierung ist nützlich, da sie uns zu sehen hilft, dass derselbe Mechanismus (…) auch heute noch funktioniert. (…) Ein paar Tage in einem Labor enthüllen, dass dieselben Kräfte, die die Druckerpresse notwendig machten, noch immer agieren, um neue Datenbanken, Raumteleskope, Chromatografien, Gleichungen, Scanner, Fragebögen usw. zu produzieren. (Latour B. , 2006, S. 275f)

Durch die Fräsmaschine konnte die Referenzkette direkt mit dem Bearbeitungseingriff am Material beginnen und sich bis in den Entwurf ausbreiten. Die mathematischen Formeln, die Parsons und Stuhlen aus ihren 17 Punkten extrahierten, waren im Kern der Nachbau des Entwurfes selbst (heute würde man diesen Ansatz als Reverse Engineering[23] bezeichnen). Sie benötigten dieses Konstrukt um eine genauere Anleitung, mit weit mehr Koordinaten herstellen zu können. Durch den Schritt die Formel Parsons mit der originalen, auf aerodynamischen Berechnungen basierenden Formel der Entwerfer zu vernetzen, wurde eine weitere Übersetzung bewerkstelligt. Durch das Zusammenspiel vieler Erfin-

22 Die erste Fräsmaschine wird auf 1818 datiert (Vgl.Woodbury, 1964, S. 17). Sie wurde von Eli Whitney erfunden, um unter anderem Schusswaffen und später andere Werkzeuge herzustellen.

23 Forward Engineering bewegt sich vom abstrakten Ende der Referenzkette weg, während sich Reverse Engineering darauf zubewegt (Vgl.Raja & Fernandes, 2008, S. 2f).

dungen wurde eine Mobilität des Entwurfs entlang der Referenzkette erreicht, die sowohl in Dimension als auch in Geschwindigkeit alles Bisherige in den Schatten stellte.

> Ein Mensch ist niemals mächtiger als ein anderer – sogar von einem Thron aus; von einem Mann jedoch, dessen Auge Aufzeichnungen dominiert, durch die gewisse Verbindungen mit Millionen anderer hergestellt werden können, kann man sagen, dass er *dominiert*. Diese Herrschaft ist jedoch kein gegebenes Faktum, sondern eine langsame Konstruktion, und sie kann korrodiert, unterbrochen oder zerstört werden, wenn die Aufzeichnungen, Akten und Zahlen immobilisiert, veränderbarer und weniger les- und kombinierbar oder bei ihrer Ausstellung undeutlich gemacht werden. Der *Maßstab* eines Akteurs ist mit anderen Worten kein absoluter, sondern ein relativer Begriff, der mit der Fähigkeit variiert, Information über andere Orte oder Zeiten zu produzieren, zu erfassen, zusammenzufassen und zu Interpretieren. Sogar die bloße Idee eines Maßstabs ist unmöglich zu verstehen, ohne eine Inskription oder Karte im Kopf zu haben. Der »große Mann« ist ein kleiner Mann, der auf eine gute Karte schaut. (Latour B. , 2006, S. 297)

Alberti und Parsons sind mit zwei ähnlichen Konzepten angetreten, um die Übersetzung in der Produktion zu revolutionieren. Der Buchdruck und später die technischen Zeichnungen haben Alberti und seine Methode auf die Plätze verwiesen, während das CAM aus den gegenwärtigen Produktionen nicht mehr wegzudenken ist. Die ähnlichen politischen Interessen, die hinter den Ansätzen standen, konnten aber beide nicht erfüllen. Hierarchie und Herrschaft sind von den weicheren Wirkungen Einfluss und Dominanz abgelöst worden. Die Referenzen sind ein Kern dieser Modelle und ihre Mobilität ein wichtiger Faktor. Will sich der Architekt mit seinen Vorstellungen und Zielen im Entwurfskollektiv durchsetzen, kann er nicht darauf bauen, dass ihm sein Status den gewünschten Einfluss verschafft. Die Digitalisierung war so erfolgreich, dass diverse abstrakte Darstellungen[24] des Entwurfs diesen über den Kopf des Architekten hinweg vernetzt haben. Das Darstellen in Zahlen ist, wie uns Alberti lehrt, nur die halbe Miete, und der Architekt kann Einfluss zurückgewinnen, wenn er wie Parsons und Stulen einen Weg findet, die Ebene der Zahlen zu erschließen.

24 Ein Beispiel unter vielen wäre der Energieausweis

4. ZWEI ENDEN EINER KETTE

Grammatik, Parametermodell und Objektorientierung

DER PARAMETER: VON DER MOUVANCE ZUR SPRACHE

Das große Problem Albertis war die zu seiner Zeit vorherrschende Berufsauffassung der Schreiber und Kopisten, die ihnen ein Modifizieren seiner Übertragungen gestattete. Die Mouvance, die der sicheren Informationsübertragung im Weg stand, war aber keinesfalls eine beliebige Einflussnahme. Wie Cramer beschreibt, war sie eher eine andere Art der Literaturästhetik. Die Schreiber verstanden es als ihre Aufgabe den Inhalt zu verändern, denn das Potenzial für Veränderung wurde als Qualität des Werkes angesehen. Der ursprüngliche Text musste dabei aber nicht fertiggestellt werden, seine Variationen waren vielmehr ein Teil seiner Konzeption. Er wurde nicht fahrlässig falsch abgeschrieben, sondern von einer höheren Abstraktionsebene abgeleitet. Was wie in diesem Fall bei literarischen Werken funktionieren kann, hat jedoch bei technischen Anweisungen wie jenen Albertis ungewünschte Auswirkungen.

> Wesentlich der romanistischen Literaturwissenschaft und dem von Paul Zumthor eingeführten Begriff der Mouvance ist es zu danken, dass wir gelernt haben, die Umstellung von Gedichtstrophen nicht grundsätzlich als Korruptelen, sondern als je authentische Zeugnisse eines historischen Zustandes, mehr noch: als Ergebnis eines planvollen, bewussten und beabsichtigten Arbeitsprozesses am Gedicht anzusehen, wer immer ihn zu verantworten hat. Dies unterstellt eine Literarästhetik, die das Werk des Autors prinzipiell offenhält für Veränderungen, die das Spielen mit dem vom Autor bereitgestellten Wort- und Formenmaterial nicht nur für legitim, sondern möglicherweise sogar für geboten hält. Es handelt sich bei der Lyrik um besonders prägnante Fälle von »unfesten Texten«, wie sie für die volkssprachliche mittelalterliche Literatur charakteristisch sind. Dies bedeutet keinesfalls, dass der Autor ein unfertiges Werk, gleichsam nur das Rohmaterial zu gefälliger Fertigstellung durch den Benutzer anböte. Zumthors stark an der Mündlichkeit orientierter, prinzipiell wichtiger Begriff vom mittelalterlichen Text (…) suggeriert unabsichtlich die Vorstellung eines Vervollkommnungsprozesses. Veränderung hat jedoch weder etwas mit Verschlechterung, noch mit Vervollkommnung zu tun. Im Gegenteil: die Offenheit für variierende Veränderungen, das kombinatorische Poten-

> zial ist eine ästhetische Dimension und muss als eine Qualität des Werks angesehen werden. (Cramer, 1998, S. 50f)

Der von Cramer beschriebene »unfeste Text« stellt, durch die nicht genau definierten Methoden der Schreiber, ein Extrem des parametrischen Entwurfes dar. Andrea Palladio hat diese unfeste Konzeption im sechzehnten Jahrhundert für die Architektur erschlossen. Er entwickelte eine neue Art von Villa (vereinfacht kann man sich ein Wohnhaus mit Tempelfassade vorstellen), die er in einer von ihm selbst illustrierten Entwurfsanleitung »I Quattro libri dell' Architettura« publizierte. Darin enthalten waren genaue Beschreibungen zu Konstruktion und Gestaltung, die sich wiederum in mehrere Schritte unterteilten. Mit diesen Regelsätzen formulierte er die Formengrammatik des Palladianismus, die das Erzeugen von Villen im Stile Palladios ermöglichte. Sein Ansatz ist damit mit dem Parsons vergleichbar: Palladio schuf mit den Regeln ebenso eine höhere Ebene des Entwurfs; denn wie aus den mathematischen Formeln Parsons, konnte nun aus den Vorlagen und Regeln Palladios ein Gebäude abgeleitet werden. In den vier Büchern, speziell im zweiten Band, formulierte er Anordnungen und Größenverhältnisse, die sich von heutigen parametrischen Modellen kaum unterscheiden. Spezifische Maße werden als das Vielfache anderer Abmessungen (z. B.: Gang zu Atrium) in Relation gesetzt, womit das Bestimmen einer Abmessung die damit verbundenen mitdefiniert.

> The following design illustrates the atrium with four columns, the breadth of which is three-fifths of the length. The side passages are a quarter of the length of the atrium. The columns are Corinthian and their diameter is half the breadth of the side passages; the unroofed part is one-third of the breadth of the atrium; the tablinum is half as broad as the atrium and the same length. One goes through the atrium and the tablinum to the peristyle, which is a square and a half in length; the columns of the ground floor are Doric and the porticoes are as broad as these columns are long; those above on the second story are Ionic and thinner by a quarter than those of the first; below them is a poggio or pedestal two and three quarter feet high. (Palladio, [1570] 2002, S. 103)

Solange man sich an die Anleitung hält, ist es möglich, in beliebiger raumzeitlicher Distanz zu Palladio, nach seinen Regeln zu entwerfen und zu bauen. Genau wie bei Alberti der Entwurf vom Bauen getrennt wurde, konnte er mit dieser Schnittstelle den Entwurf in Gestalt der Formengrammatik vom Entwerfen mit der Formengrammatik als Werkzeug trennen. Es entstand dadurch eine Trennung zwischen dem »abstrakten« Entwurf, aus dem das Werkzeug hervorgeht und dem »konkreten«, der mit einem bestehenden Werkzeug arbeitet. Palladio hat auf diese Weise die Referenzkette nach oben hin erweitert, und mit der Schnittstelle zugleich neue Möglichkeiten der Einflussnahme über diverse Para-

meter eröffnet[25]. Wobei man zwischen den einzelnen Anweisungen differenzieren muss: während im Grundrissaufbau eine einfache Grammatik erstellt wurde, sind andere Kapitel (wie das oben angeführte) und alle erwähnten Proportionsregeln Parametermodelle. Der Unterschied ist, dass während in parametrisierten Modellen durch die Definition weniger Variablen alle anderen (unmittelbar) berechnet werden können, in einer Grammatik ein Regelsatz[26] besteht, nach dem verschiedene Resultate (in mehreren Generationen) abgeleitet und produziert werden können. Parameter sind also Einflussfaktoren und die formale Grammatik ist eine Schritt-für-Schritt Anleitung. Als einfach anzuwendende Technik für das architektonische Entwerfen war Palladio damit ein Meilenstein gelungen.

> Palladio's *I Quattro Libri dell' Architettura* sets out rules of classical architectural usage in much the same way as a traditional grammar sets out rules of Latin usage. The conventions of composition and construction governing correct building practice are established by prescription and example. An architectural Language is thus defined that owes its power and richness to the coherence and clarity of these »precepts of art … which reason dictates«. (Stiny & Mitchell, 1978, S. 5)

Neben der Anwendbarkeit war ein weiterer Erfolgsfaktor, der als »Palladianismus« bekannten Technik, der Buchdruck und die später darauf fußende massive Vernetzung im britischen Raum (Vgl. Evers & Thoenes, 2003, S. 414). Bereits um 1716 veröffentlichte der aus Italien stammende Architekt Giacomo Leoni die erste vollständige englische Übersetzung der vier Bücher. Dass das Werk in gedruckter Form auflag und damit verfügbar wurde, war das eine, aber die zahlreichen gedruckten Abbildungen, die typisch für dieses Medium waren, ermöglichten erst, was Alberti verwehrt blieb: den Entwurf beinahe unverfälscht zu transportieren zu können. Auf diese Art ist die Kombination Palladio und Buchdruck mit der Kombination von Digitalisierung und elektronischer Informationsübertragung vergleichbar, auf der Parsons Erfolg beruht. Den eigentlichen Durchbruch schaffte der Palladianismus dann mit der Publikation von Campbell, die annähernd zur gleichen Zeit erschien. In einem reichlich bebilderten Werk präsentierte er die britische Architekturgeschichte als homogenisierten nationalen Trend und stellte ihr Palladios vier Bücher als Idealbild und logische Weiterentwicklung gegenüber. Die von Palladio auf

25 Komplexe Geometrien konnten auf weniger numerische Eingaben reduziert werden, was diese einflussreicher werden ließ.

26 Eine formale Grammatik besteht dabei aus einem Startsymbol, Terminalsymbolen, nichtterminalen Symbolen und syntaktischen Regeln. Die formale Sprache ist menge aller Sätze die damit erzeugt werden können. (Vgl.Wirth, 2008, S. 6ff)

diese Weise geschaffene Vorbildwirkung verfehlte ihr Ziel nicht, und so erfuhr der Palladianismus im englischsprachigen Raum einen enormen Boom. Bis in das 19. Jahrhundert wurden Villen im Stil Palladios gebaut. Selbst Thomas Jefferson, dritter Präsident der Vereinigten Staaten von Amerika und Architekt, entwarf gemäß den beschriebenen Regeln die Universität von Virginia und sein Herrenhaus Monticello[27].

> The three-volume publication *Vitruvius Britannicus* by the English architect Colen Campbell is considered to be the manifesto that established English Palladianism. However, the work in question is not a literary reflexion but an extravagant compendium of illustrated plates containing a total of 295 architecture engravings. The text section consists of no more than a brief foreword to the first volume, and explanations of the illustrations. (Evers & Thoenes, 2003, S. 412)

Palladio nahm durch die von ihm entwickelte Technik durch Delegation massiv Einfluss auf die damit hergestellten Entwürfen und Formen. Sein Einfluss reichte aus, um ihn zum Namenspatron eines ganzen Stils zu machen. Obwohl er einen Großteil des Entwurfs in eine vorgegebene Richtung gelenkt hat, wird er nicht als Architekt der umgesetzten Bauten geführt. An seiner Stelle erscheinen die Namen derer, die mit dem von ihm zur Verfügung gestellten Werkzeug entworfen haben. Viele Architekten konnten von den Büchern viele Häuser ableiten, hingegen blieb das Werk Palladios mit dem Entwerfen der Technik abgeschlossen. Während in Bezug auf die Mouvance der Autor der Grundlage nach wie vor der Autor blieb, und nur das Werk als solches unfest war, hat die viel spezifischere Vorgabe Palladios ihn selbst von diesem Podest gestoßen. Er wurde vom Architekten des Gebäudes zum Autor einer Technik.

Um diese Diskrepanz noch klarer darzulegen helfen Stiny und Mitchell: Viele Jahre nach Palladio machten sie sich daran, Palladios Formengrammatik in ein, für Computer berechenbares, parametrisiertes Modell zu übersetzen. Ihr Beitrag zum Diskurs über ästhetische Systeme wuchs dabei über die reine Beschreibung eines Interpretationswerkzeugs hinaus. Sie ermöglichten Teile des Entwurfs zu automatisieren, indem sie dem Computer auch das Ausarbeiten der Formen nach den erstellten Regeln übertrugen. Stiny hat in einem späteren Aufsatz beschrieben, wie anhand einer derartigen »shape grammar«, von einer Ausgangsgeometrie stufenweise ein Ergebnis ermittelt werden kann (Vgl. Stiny G. , 1980, S. 348). Während anfangs Palladios Sprache noch mit Latein verglichen wird, stellen Stiny und Mitchell später fest, dass die Menge der erzeugbaren Sätze der Gramma-

27 Monticello, gebaut 1772, in Charlottesville, Virgina, USA

tik, also die Menge der Gebäude, im Gegensatz dazu aber eingeschränkt und bei Weitem nicht unendlich ist.

> The definition of the Palladian style by use of the parametric shape grammar specified here allows other issues and questions of aesthetic and historical interest to be investigated. For example, the grammar provides the basis for classifying villas in terms of the properties of the sequences of rules applied to generate their plans and even count the number of possible villas of a certain type. Indeed a simple combinatorial analysis of possible sequences of rule applications would allow for the computation of the number of all possible villas that have an underlying grid size of say 5 x 3. The grammar could also be used to distinguish those stylistic features, in neo-Palladian movements, that are canonical in the Palladian sense from those that diverge from Palladio's standard architectural usage. Using the grammar explicated here, architectural historians could ask and find answers to questions such as: how deep (or superficial) was Lord Burlington's appreciation of his mentor's system of design? (Stiny & Mitchell, 1978, S. 17f)

Parametermodelle lassen als Übersetzungstechnik gerade in digitalisierter Form einfach von der Referenz zum Referenzierten schließen, und umgekehrt. Diese Rückverfolgbarkeit kann bei Zuhilfenahme einer formalen Grammatik um das Vielfache erschwert werden, sie gelingt Stiny und Mitchell am Beispiel Palladios aber dennoch. Durch den Umstand, dass nur eine geringe Anzahl an verschiedenen Varianten herstellbar war begünstigt, konnten sie mit einer Brute Force Taktik[28] die Entscheidungen des jeweiligen Planers rückverfolgen. Auf diese Weise gelang es ihnen, eine Aussage darüber treffen zu können, inwieweit die eigenen Ideen der Architekten über die ursprünglichen Palladios gestellt wurden. Das Zirkulieren der Referenz war somit in beiden Richtungen entlang der Kette belegt, und die zugrunde liegenden Annahmen im Arbeiten mit der Formengrammatik konnten mit dieser Methode als zusätzliche, abstraktere, Referenzebene analysiert werden. Im Kern ist die Grammatik dem Parametermodell also ähnlich, denn bei beiden können die Einflussfaktoren und Annahmen determiniert werden. Bei komplexeren Regelsätzen kann hingegen die Sprache und damit die Menge der Sätze stark anwachsen.

> A shape grammar interpreter is a computer program that can perform the operations necessary for determining if a rule can be applied (shape recognition) and can perform operations needed to apply a rule (shape addition and shape subtraction). (…) The gen-

28 Brute Force Search, eine Suchmethode bei der alle Möglichkeiten unabhängig von der ihres Wahrscheinlichkeit Vorkommens untersucht werden; nur sinnvoll bei geringer Anzahl an Varianten (Vgl.Patten, 1988, S. 9)

> eral shape grammar interpreter introduced in this paper differs from other interpreters. Previous approaches used a combination of Euclidean transformations (that is translation, rotation, and isotropic scaling) and required that the transformations be applied to the entire shape when performing shape searches. Our approach supports general parametric relations by decomposing a shape into a hierarchy of subshapes thereby allowing for any parts of the shape to be transformed with any possible transformation. The ordering of the shapes into a hierarchy provides an efficient shape-search order. In this paper a default hierarchy based on our own observation and interpretation of engineering and architectural grammars is introduced that is applicable in many cases and efficient in general; other decompositions could be applicable as well and still use the basic approach. (McCormack & Cagan, 2002, S. 914f)

Für die Menge an Sätzen ist gerade die Anzahl der Symbole (in dem Fall die definierten Geometrien) ein entscheidender Faktor, denn es können verschiedene Anordnungen auch mit einfachen Regeln nahezu unendlich viele Ergebnisse produzieren. Auf der anderen Seite lässt die Gestaltung der Regeln selbst die Sprache komplexer werden, ohne dazu aber im gleichen Maße zu zusätzlichen Resultaten zu führen. Klassische Formengrammatiken konzentrieren sich auf das Ersetzen von Symbolen durch andere, oder beinhalten Regeln zu deren Manipulation. McCormack soll hier nur stellvertretend erwähnt werden, aber an seinem Projekt der »subshape recognition« kann man einen Aspekt dieser Technik klar erkennen: Eine Grundlage von Formengrammatiken ist das Erkennen, ob bestimmte Regeln gültig angewendet werden können, also definierte Ausgangsszenarien existieren. Manche Methoden vergleichen dazu die gesamte Geometrie mit einer zuvor definierten, während in diesem Ansatz die Geometrie zuerst in ihre Bestandteile zerlegt wird. Die Bestandteile werden dann einzeln analysiert, und hierarchisch in immer umfangreichere Gruppen zusammengefasst. Je nach Ebene können unterschiedliche Regeln angewandt werden, aber welche zum Zug kommen, wird noch von einem Anwender bestimmt. Abgesehen von einer grafischen Hierarchie, die nicht die der referenzierten Bauteile widerspiegeln muss, sondern rein formalen Gesichtspunkten entspricht, klafft hier noch ein zweites Problem. Um die für die Gestaltung oft notwendigen, aber nur für spezielle Bauteile geltenden Anweisungen zu übertragen, ist die Methode der Grammatiken nur bedingt geeignet. Ihre Beschränktheit auf divergierende Aufgaben angepasst zu werden, macht sie eher für spezifische und eintönige Problemstellungen[29] interessant. Darüber hinaus fällt das sinnvolle Übersetzen von Formengrammatiken in die digitalisierte Ferti-

29 Eine der Aufgaben für die sie sich als geeignet herausgestellt hat, ist das automatische Vektorisieren von Pixelgrafiken.

gungskette wegen ihrer formalen Natur sehr schwer, was sie in Anbetracht verfügbarer Alternativverfahren unattraktiv erscheinen lässt.

Ein Vorteil, den die Grammatik im Werk Palladios hingegen mit sich brachte war, dass die Ergebnisse durch ihre Anwendung nicht offen determiniert, sondern nur angeleitet wirkten (dass sie tatsächlich prädeterminiert waren zeigten Mitchell und Stiny). Schließlich ließ das die jeweiligen Architekten ihre Gebäude auch als ihre eigenen Entwürfe ansehen. Arbeitsanweisungen, denen durch die Definition einiger weniger Parameter bereits das Ergebnis immanent war, lassen hingegen diesen Freiraum nicht mehr zu. Der Unterschied von formbildenden zu formalen Anleitungen geht, wie es den Anschein macht, mit einer Unterscheidung der Autorenschaft des Entwurfs einher, die wiederum nicht in einer Unterscheidung des Einflusses der Technik begründet ist. Denn in beiden Ansätzen werden mögliche Formen vorweggenommen und determiniert, um dann auf unterschiedliche Weise vom Architekten hergestellt zu werden. Begünstigt wird der Anschein der Trennung Palladios als Werkzeugentwickler vom Entwurf in diesem Fall auch durch einen weiteren Umstand: Bereits in der Analyse von Palladios Texten und Grafiken stießen Stiny und Mitchell auf kleine Fehler und Unstimmigkeiten, und zwischen den gezeichneten und den konstruierten Gebäuden fanden sie viele weitere Abweichungen. Ohne Frage bestand also auch bei Palladio (und im Palladianismus) das Problem der gesicherten Informationsübertragung, da gerade die händisch hergestellten Zeichnungen und Kupferstiche ein hohes Fehlerpotenzial hatten.

> The drawings in the *Quattro Libri* contain numerous small errors and inconsistencies, and the constructed buildings often deviate in minor ways from the drawings. Thus there is often a certain arbitrariness in what we take to be the intended design. Furthermore the definition of what constitutes a trivial or accidental variation in form, which need not be accounted for by the grammar, is also arbitrary to some extent. (Stiny & Mitchell, 1978, S. 17)

Vor dem Digitalisieren bot die analoge Schnittstelle mit ihren Ungenauigkeiten noch einen zusätzlichen Interpretationsspielraum, der die eine oder andere Entscheidung beeinflussen konnte. Daneben ermöglichte die Übersetzung auch erstmals die raumzeitliche Trennung von der Herstellung der Entwurfstechnik und denen die damit arbeiteten. Zusammengefasst sind also neben der offensichtlichen Trennung Palladios von den diversen Entwürfen auch die Fehleranfälligkeit, der von ihm genutzten Informationsübertragung und die Freiheiten, die durch die in Teilbereichen genutzte Grammatik bestanden, mit ein Grund für das Wahrnehmen der Rolle Palladios im Entwurfsgefüge mit den von ihm

beeinflussten Architekten als untergeordnet. Die Frage, die sich hier stellt, ist wie viel Einfluss er dennoch auf die Entwürfe gehabt hat, und ob sich die vielen Entwürfe wirklich von einem einzigen parametrisierten und grammatikalisch dargestellten Archetypus ableiten lassen. Die passende Antwort darauf haben Stiny und Mitchell mit ihrer Analyse gefunden, als sie eine Rückverfolgbarkeit herstellen konnten. Sie zeigten, dass Palladio alle Entwürfe bereits vorweggenommen hatte, und die konstruierten seine bis auf geringfügige bewusste oder unbewusste Abweichungen darstellen.
Obgleich die Übersetzung eine andere Technik als das Parametermodell bemüht, gestattet die angewandte formale Grammatik die Entwurfskette auf eine abstraktere Ebene zu erweitern. Wie Kennzahlen stehen dort alle Entscheidungen aufgelistet, die als Input für die Grammatik nur einen Output ermöglichen. Der Einfluss den beide Techniken (und deren jeweilige Autoren) auf den Entwurf haben ist somit ähnlich, nur der Input, der zur Herstellung notwendig ist, differiert. Ein Vorteil den parametrische Techniken auf sich verbuchen können, liegt genau in der abstrakten Abbildung des Entwurfs durch Zahlen. Wo formale Grammatiken mit einer Darstellung der mit ihr gemachten Entscheidungen bislang eine Sackgasse darstellen, kann mit anderen Methoden angesetzt und weiterverknüpft werden. Viele der ursprünglichen Begleitumstände sowohl von Parametermodell und als auch von der formalen Grammatik bei Palladio, wie die verfügbaren Rechentechniken und Medien, sind heute im computergestützten Entwerfen durch die Digitalisierung längst überholt. Kombiniert mit vernetzbaren Referenzen, hat das zum Erfolg des Parameters gegenüber der Grammatik im digitalen Entwerfen beigetragen.

DIE SERIE: KURVEN STATT ZEIT

Anstelle von Grammatiken ist, von der einfachen Umsetzung mit Software gefördert, die durch Parameter manipulierbare Geometrie getreten. Bei dieser Technik wird die Form durch numerische Angaben wie Bemaßungen definiert. Im Gegensatz zu früheren Methoden, bei denen der Wert nur beim Erstellen eingegeben werden konnte, bleibt die Form dabei »unfest«, und kann beliebig oft durch Einsetzen eines neuen Wertes verändert werden. Welche Zustände dabei die Parameter beschreiben ist nicht zwingend definiert, es kann sich um einzelne Teilmaße, Gesamtlängen, Winkel, Flächen, Geometrien u.v.m. handeln. Im Unterschied dazu werden statische Elemente durch das Aufrufen einer Funktion einmalig konstruiert, um danach bearbeitet zu werden. In den letzten Jahren wurde

bei unzähligen Entwürfen auf die parametrische Methode zurückgegriffen, den hierbei gängigen Vorgang beschreiben Vanderfeesten und de Vries folgendermaßen:

> In the parametric procedure certain parameters are given before the script is executed. Essential elements from the client brief are subtracted into parameters with unified quantity of comparable mathematical values. The essential elements are mostly derived from either the building blocks or the urban context. Quantity, dimension and orientation are the basic parameters used in an urban design, which requires a large number of elements. Together with other non-derived parameters, these form the basic starting parameters of the parametric design method. At the beginning of the script the option is given to set the parameters to suitable values. (…) The repetitiveness of the script allows rerunning the sequence many times with different input parameters in order to generate different results. (Vanderfeesten & DeVries, 2006, S. 308)

Viele der verwendeten Parameter sind bereits vor dem Entwurf mit dem digitalen Modell fixiert. Sie leiten sich aus topografischen, soziologischen oder wirtschaftlichen Umgebungsvariablen und Bedingungen ab. Von ihrer abstrakten Kennzahlenebene können sie mit Hilfe eines digitalen Modells in eine konkrete Form übersetzt werden. Im Gegensatz zur Grammatik entsteht eine Schnittstelle, die zwischen Geometrie und Interessen vermittelt. Hinzu kommen Parameter, die beliebig veränderbar sind, und auf diese Weise einen Ausweg aus dem Formendeterminismus ermöglichen. Egal ob nun die Parameter der Modelle die Kennzahlen referenzieren (im Entwerfen) oder umgekehrt (in der Analyse), lässt sich die Kette nur bis zum geometrischen Körper spannen. Auf der anderen Seite ist der Vorteil »unfester« Geometrien im Planungsprozess unbestritten, und hat die Interaktion von Mensch und Maschine stark beeinflusst. So können Werte und die damit verknüpften Modelle in Echtzeit variiert und in zahllosen Durchgängen von Ändern-und-Analysieren an die »eigenen Vorstellungen« angepasst werden. Man kann zur Darstellung dieses Vorgangs auf ein Modell aus den Arbeitsberichten zur Planungsmethodik zurückgreifen, dort wurde der zyklische Planungsvorgang mit drei Ausgängen definiert: Gefiel das Ergebnis, war der Entwurf fertig, gefiel es nur mittelmäßig, mussten die Eingaben verbessert werden, und bei Nichtgefallen des Resultates hieß es, ein neues Modell zu konstruieren (Vgl. Joedicke, Rietkötter, Schmöller, Schulte, & Seibert, 1975, S. 23f). In den beschriebenen Szenarien wird der Entwurf zyklisch durch das langsame Annähern der Zahleneingabe an einen die gewünschte Form produzierenden Wert erstellt. Dabei geht mit jedem neuen Zustand der alte verloren.

> The shapes that are formed in computer-aided design are the result of the decisions made using parameters. Numerical data which describe characteristics of the virtual design en-

> vironment – such as temperature, gravity, and other forces – have an impact on the forms which result. For example, dynamic modelling systems are based on the interaction of multiple parameter statements calculated sequentially rather than in an instant. Numerical parameters can be keyframed and dynamically linked through expressions to alter the shape of objects. (…) The linkages between these characteristics of time, topology, and parameters combine to establish the virtual possibilities for designing in an animate rather than static space. (Lynn, 1999, S. 25)

Anstatt ein für das technische Zeichnen entwickeltes CAD Programm zu nutzen, hat Greg Lynn die für die Filmbranche konzipierte Software Maya[30] zum Entwerfen verwendet. Das für ein Animationsprogramm typische Darstellen der Arbeitsfläche als Frame eines Filmstreifens hat ihm ermöglicht, die Zustände nicht mehr verwerfen zu müssen, sondern in eine zeitliche Abfolge zu bringen. Mit der Zeit (oder Frameposition) als zusätzlicher Achse konnten er Szenarien (Keyframes) definieren, und deren Zwischenpositionen oder Entwicklungsstadien interpolieren. Sein Lösungsansatz ist dabei mit dem von Parsons und Stulen ident, die auch eine mathematische Methode nutzten, um die zwischen den definierten Punkten liegenden restlichen Punkte zu determinieren. Was er dabei aber erhielt, war eine animierte Form, theoretisch in unendlich viele Frames unterteilbar, die »unfeste« Referenz neu visualisiert.

> Das von Lynn hauptsächlich verwendete 3D-Modellierungs- und Animationsprogramm *Maya* ermöglicht darüber hinaus die modellhafte Darstellung dynamischer Effekte. Mit dem Programmelement *Dynamics* lassen sich Bewegungen und Deformationen von geometrischen Objekten oder Partikeln nachbilden, die durch Kräfte wie Gravitation oder Wind beeinflusst werden. So können Objekte mit Anziehungskräften ausgestattet werden, innerhalb derer es zu Verschmelzungen der Formen kommt. Die physikalischen Kräfte, die ein Objekt in Bewegung versetzen oder verformen, werden mit aufwendigen mathematischen Verfahren berechnet, wobei je nach gegebenen Variablen ein jeweils anderes Ergebnis entsteht. Vor dem Hintergrund solcher Animationstechniken ist der geometrische Körper nicht mehr nur diskret, sondern als verformbares Kontinuum vorhanden. (Höfler, 2005, S. 67)

Um die für die Formgebung ausschlaggebenden Parameter zu konstruieren, hat Lynn nicht nur auf »reale« Vorgaben zurückgegriffen, sondern sich auch bewusst an den im virtuellen Raum simulierbaren »Kräften« orientiert. Die Software versetzte ihn in die Lage physikalische Eigenschaften wie Gravitation als Gestaltungselement einzusetzen, und

30 1998 von Alias Systems herausgegeben, produzierte Walt Disney Feature Animation damit den im Jahr 2000 veröffentlichten Animationsfilm „Dinosaur"

Gebäudehüllen auf diese Weise zu verformen. Mit einer enormen Bandbreite an mathematischen Modellen ausgestattet, konnte so eine Übersetzung fiktiver Kennzahlen realisiert und die geringe Datenmenge der Vorgaben aus dem »realen« Raum erweitert werden. Ist jede Formel, die eingebaut wird, von ihrem Input abhängig, sind umso komplexer die Animation werden soll, entsprechend mehr Werte notwendig. Durch die Menge an Information, die sich aus der zur Verfügung gestellten und aus der in weiterer Folge durch Interpolation daraus neu erzeugten zusammensetzte, ergab sich eine Konzentration von Einfluss auf den Entwurf, die schlussendlich zu seiner Wahrnehmung als Kontinuum mit unendlich vielen Zuständen geführt hat.

> Der Ablauf derartiger Prozeduren zielt darauf ab, endgültige Formen oder Strukturen zu gewinnen. Die architektonische Gestalt erhält Lynn, indem er die fortlaufende Koordinatentransformation an einem bestimmten Zeitpunkt anhält und die in dem Moment dargestellte Form *einfriert* – was dem dynamischen Charakter der Animation zunächst widerspricht. Die Faktoren, die den fortwährenden geometrischen Transformationsprozess steuern, sind im Animationsprogramm nicht enthalten, sondern werden vom Anwender vorgegeben. Lynn bildet Vorgaben aus empirischen Daten, aus Gegebenheiten des räumlichen Kontextes oder der Aufgabenstellung des Entwurfs. (Höfler, 2005, S. 68)

Das augenscheinliche Problem seines Zugangs liegt im Versuch aus einer animierten Form ein starres Gebäude abzuleiten. Lynn gelingt dies, indem er einen der Frames auswählt. Der interpolierte Zustand der Geometrie lässt ihn auf die aktuellen Werte der zugrunde liegenden Parameter rückschließen, und ein den Vorgaben und Vorstellungen entsprechendes Ergebnis aussuchen. Die Form, die an einem Ende der Referenzkette weiter auf die Parameter verweist, dient als Basis für die Umsetzung. Die dort angewandten Parameter verweisen nicht auf einen weniger abstrakten Zustand des Entwurfs als die grafische Darstellung, wie etwa das Gebäude selbst, sondern auf die Darstellung als solches. Darin ist dieser Ansatz der generativen Formengrammatik ähnlich. Es handelt sich hier ebenfalls um eine Verbindung zweier, voneinander unabhängiger Referenzketten über die Zeichnung; die Umsetzung, die sich bis zur Geometrie erstreckt, und die Geometrie, die bis zu den Vorgaben reicht. Wobei sich die abstrakte Entwurfskette im Vergleich zur Grammatikvariante in der Anzahl ihrer Übersetzungen (und damit möglicher Teilnehmer) ausgedehnt hat.

> Greg Lynn thinks thus today of his design as a series, as versions. Consequently the most interesting to him, is less the literal production of folded architecture, but Leibniz's differential calculus and its ability to fuse the hierarchy of parts and whole to produce a

> deeply modulated whole as well as infinitesimal variation among parts. The Embryological House (1999) is a highly diagrammatic version of an architectural series, but it is also one of the few examples in architecture up to today which explicitly conceptualizes the idea of differential calculus. Lynn's Embryological House is a series of one-of-a-kind houses that are customized for individual clients. (Rocker I. M., 2008, S. 164)

In dieses Bild passt auch das Verständnis Lynns vom Entwurf als Serie. Aus der »unfesten« Form, die Freiheiten bietet, konstruiert er eine unendliche Menge an Variationen. Im Gegensatz zu Palladio galt aber nun er als der Autor all dieser Elemente, und nicht die, die mit seinem Modell arbeiteten. Man hätte maximal die Chance sich für einen Frame entscheiden zu können, aber entworfen wurde er von Lynn. Hat sich die Kette durch das Verbinden von Kennzahl und Parameter in ihre abstrakte Richtung erweitert, ist ihr durch das Vorwegnehmen der Werkzeuganwendung auch die Ausbreitung in die andere, konkrete und materielle, Richtung gelungen. Und da sich der Entwurf nicht auf die Umsetzung bezieht, sondern auf die Geometrie, kann die Serie als virtuelles Produkt bestehen bleiben. Auch wenn Plan und Entwurfsvisualisierung in einer einzelnen Darstellung als dasselbe scheinen, handelt es sich aus der Perspektive der zirkulierenden Referenz um eine Barriere.

> From the particular discursive formation of multiple, diagonally intersecting statements, some form of expression emerges. Through the interaction of a multiplicity of abstract statements, signifiers emerge in a more dynamic manner than mere representational effects might. The shift from linguistic models to the proliferation of asignifying statements marks what Deleuze terms a move from the »archive« to the »diagram«. The move from linguistic construction to statements, or more properly from meaning to machine, is a necessary shift in sensibility if one is to tap the potential of abstract machines such as computational motion geometry and time-based, dynamic force simulation. (Lynn, 1999, S. 40)

Lynn positioniert seinen Ansatz ausdrücklich als Alternative zu linguistischen Modellen wie der Formengrammatik. Während dort geometrische Elemente nur als Symbol behandelt werden, versucht er sie dynamisch zu gestalten. Mit den abstrakten Maschinen[31], die von der Software zur Verfügung gestellt wurden, sollte durch eine Vielzahl an Feststellungen ein neuer Ausdruck produziert, anstatt eine Behauptung konstruiert werden. Ein

31 Eine abstrakte Maschine, oder Automat, ist ein theoretisches Modell eines Computersystems. Es besteht üblicher Weise aus Input, Output und den auszuführenden Prozessen. Zu ihnen gehören u. a. die Turingmaschine und Zelluläre Automaten.

weiterer Unterschied, der sich darlegen lässt, liegt in der Menge der transportierbaren Information. Wo bei Grammatiken durch die Schwierigkeit komplexer Regelsätze meist auf die Symmetrie zurückgegriffen wurde, sieht Lynn durch Erkenntnisse aus der Biologie bestärkt, einen Mangel an Information. Der Forderung durch die Übersetzung mehr Information übertragen zu können, wird beim parametrischen Entwerfen durch die Herstellung von zusätzlichen Kennzahlen und weiteren notwendigen Eingaben genüge getan. Die Information die zur Differenzierung notwendig wird herzuleiten, hat sich in weiterer Folge als Forschungsgebiet für den Digitalen Entwurf aufgetan. Während von vielen zum Zweck der Informationsgenerierung Pseudo-Zufallszahlen verwendet werden, gibt es auch einige, die versuchen ohne ein solches Hilfskonstrukt auszukommen. Die Suche nach einer Abhängigkeit der Eingabe in Form einer Referenz, um die Sackgasse des Zufalls zu vermeiden, hat zu einer Weiterentwicklung von Lynns Konzept durch Ingeborg Rocker geführt.

> So, symmetry wasn't the sign of order and organization - which is what I was always understanding, and as is an architect - symmetry was the absence of information. So, whenever you lost information, you'd move to symmetry; whenever you added information to a system, you would break symmetry. (Lynn, 2005)

Zusätzlich zum Seriengedanken von Lynn hat Rocker die entworfenen Varianten nicht mehr nur zeitlich aneinandergereiht, sondern räumlich als Teilelemente eines Ganzen zusammengefügt. Die Parameter, die zur Bestimmung einer Instanz notwendig sind, werden hierbei von einer übergeordneten Geometrie definiert. In ihren Entwürfen sind das unter anderem einzelne Elemente als Querschnitte eines Ganzen oder Segmente einer parametrisierten Oberfläche. In dem von Rocker beschriebenen »Versioning« dient eine zusätzliche Referenzebene zum Herleiten der notwendigen Information. Als »Kurven von Kurven« bezeichnet, entsteht so ein verschachteltes parametrisches Modell, indem eine Geometrie in mehrere davon abhängige aufgeteilt wird. Während einfachere Anwendungen der Parametertechnik primär dazu dienen Entwurfsiterationen zu simplifizieren, ist die Weiterentwicklung ein komplexes Konstrukt um Potenziale auszuloten.

> The core idea of versioning, however, goes far deeper than simple variation between different parameterized design iterations. Both the power and the limitations of versioning-as-series is demonstrated by Lynn's work – which both suggests the possibility for limitless parameterization across a series. Versioning, however, also operates at the micro-scale, within the structure of digital design itself. (...) The sinuous curve and biomimetic form itself isn't difficult to accomplish without computers. What is rather difficult is to make it mathematically malleable and variable. This is where a higher-order thinking

> comes in - it is not just curves themselves, but curves of curves, recursive functions as they literally inform computer-generated form. (Rocker I. M., 2008, S. 167)

Rocker hat die Methode Lynns, mit der er unendlich viele Varianten herstellen kann, übernommen, und anstelle der Zeitleiste eine mathematische Kurve zum Konstruieren des notwendigen Inputs der Versionen verwendet. Der Unterschied besteht darin, dass die Kurve wieder auf Parameter angewiesen ist statt auf eine aufsteigende Reihe natürlicher Zahlen. Im Kern hat sie also die Menge an notwendiger Information komprimiert, und die Referenzkette verlängert. Es lässt sich auch hier der Lösungsansatz von Parsons und Stulen wiederfinden, in dem Punkte zu einer mathematischen Funktion verbunden wurden. Die Frage, die sich aufdrängt, ist, wie die für die Kurve benötigte Information konstruiert wird. Bildet eine zusätzliche Kurve für eine Kurve für eine Kurve logische Weiterentwicklung? Oder läuft die numerische Schnittstelle des Entwurfs auch in dieser abstrakten Ebene wieder Gefahr, von deterministischen Optimierungstechniken vereinnahmt zu werden.

> Die Entwurfsmethoden sollten anhand bestimmter Merkmale aus gegebenen Zielen, Maximen und Randbedingungen in nachvollziehbarer Weise ein Optimum, etwa nach dem Kriterium der minimalen Weglänge in einem Krankenhaus erreichen. Dies alles sollte unter Berücksichtigung möglichst vieler Randbedingungen geschehen. Hier wurde der Weg zur Konstruktion der Architekturmaschine wenn auch nicht im Sinn Negropontes als »evolutionärer« Apparat beschritten. Allerdings entwickelte sich dieses Thema schon Anfang der siebziger Jahre nicht nur wegen des ungelösten Problems der Übersetzung qualitativer Kriterien in quantifizierbare und sprachlich darstellbare Entwurfseinheiten und der auch aus Cambridge, Mass. Geläufigen »Schnittstellenproblematik« des Mensch-Maschine-Systems zu einem teuren von der DFG finanzierten Spielzeug für Doktoranden, das allenfalls in der »Variantenerzeugung« mit verschiedenen Parametern eines dann nicht so leicht automatisierbaren Entwurfsprozesses (…) endete. (Weckherlin, 2009, S. 220)

Mit dem Definieren von Parametern durch Kennzahlen fallen diese aus dem Einflussbereich des Architekten und werden durch andere Verbindungen stärker beeinflusst. Der »selbstständige« Entwurfsautomat, als kreatives Worst Case Szenario und Extremfall des Optimierungsdogmas, ist an der totalen Automatisierung, wie Weckherlin zeigt, bereits Mitte der Siebziger gescheitert. Ökonomischer (und seit kurzem auch ökologischer) Determinismus geht mit einer sogenannten »Verwissenschaftlichung« des Entwerfens einher, die auf eine isolierte Betrachtungsweise einzelner Faktoren setzt. Dass die Übersetzung aller Kriterien in »objektive«, also konstruierbare Referenzen nicht funktioniert, sieht Weckherlin als Grund für das Scheitern des Entwurfsautomaten. Inzwischen hat sich aber

eine Alternative zum Versuch die Referenzkette von der abstrakten Seite aufzurollen herauskristallisiert, die sich statt dessen am abstrakten Ende der existenten (bis zur Materie reichenden) Referenzkette anzubinden versucht.

DIE OBJEKTORIENTIERUNG: DER ANDERE ANSATZ

Eine andere Form zur Visualisierung des Entwurfs mit digitalen geometrischen Körpern ist die Darstellung der baulichen Elemente in der Plangrafik. Dabei referenzieren Zeichnungen die zu fertigenden Bauteile und nicht die entworfene Form. Stellvertretend für physikalische Formschlüssigkeit wird auf die assoziative Geometrie zurückgegriffen, womit die Bauteile zu Baugruppen organisiert und angeordnet werden können. Alle angeordneten Baugruppen bilden zusammen das Haus, das Trummer als »Population organisierter baulicher Elemente« bezeichnet. Noch deutlicher als beim Versioning Rockers wird hier das Ganze als Summe von Teilen gedacht, die zusätzlich untereinander in Beziehung stehen. Während andere Ansätze versuchen die Information für die Geometrie aus Parametern wie Zeitleisten oder Kurven abzuleiten, beziehen die Elemente in Trummers Methode einen Großteil ihrer Information voneinander. Die Folgen von Parameteränderungen werden damit für den Architekten schwerer einschätzbar. Es entsteht die Möglichkeit, Elemente über einen Umweg zu ändern, ohne auf sie direkt zuzugreifen.

> Wir könnten das chinesische Hofhaus als eine Population organisierter baulicher Elemente bezeichnen, das durch die Leistung seines konstruktiven Systems (seine metrischen Eigenschaften) und durch die ökologischen Kräfte (die nicht metrischen Eigenschaften) bestimmt wird. Es ist der erste architektonische Entwurfsansatz, der auf der assoziativen Geometrie fußt und auf unterschiedliche Maßstäbe anwendbar ist. (Trummer, 2008, S. 50)

Der Schwerpunkt der beschriebenen Technik liegt im Definieren von assoziativen Geometrien. Dabei verhalten sich einzelne Elemente untereinander nach gewissen Regeln[32] und

32 Beispiele für explizite geometrische Regeln sind, dass zwei Linien an ihren Endpunkten zusammenhängen oder die Endpunkte kongruent oder deckungsgleich sind (...), zwei Geraden parallel zueinander sind (...), Kreis und Gerade tangieren (...) oder ein Punkt äquidistant oder mit gleichem Abstand zu zwei parallelen Geraden liegt. (Rembold, 2007, S. 44)

beeinflussen sich wechselseitig. Den Bemühungen eine höchst informationsreiche Geometrie auf eine minimale Anzahl von Parametern herunterbrechen zu können, stellt Trummer die Betrachtungsweise der »Population« entgegen. Dabei sieht er die Geometrie nicht mehr nur als ein Gebilde, das sich entwickelt und wächst, sondern als Summe ihrer Teile, die sich gegenseitig beeinflussen und als Kollektiv eben jene hervorbringen. Dieses Beziehungsmodell überträgt er auch in andere Maßstabsebenen und versucht sie für städtebauliche Aufgaben anzuwenden. In einer verschachtelten Struktur kann so die referenzierte Stadt bis in die referenzierten Einzelteile der Gebäude aufgelöst werden.

> Assoziatives Entwerfen entspricht einem parametrischen Entwurfsansatz, der metrische Parameter verwendet, um eine endlose Anzahl an Variationen zu erzeugen. Es basiert auf einer Technik der assoziativen Geometrie. Diese Geometrie beschreibt die Beziehung zwischen verschiedenen Komponenten und schafft ein Entwurfsobjekt, dessen Konstruktion auf der wechselseitigen Beziehung seiner geometrischen Parameter beruht. (Trummer, 2008, S. 50)

Mit einer weiteren objektorientierten Entwurfsdarstellung hat Bernard Cache (mit seinem Büro Objectile) einen Pavillon umgesetzt. Dabei kam wie bei Trummer die Technik der assoziativen Geometrie zum Einsatz, mit Hilfe derer sie die unzähligen Einzelteile in Abhängigkeit weniger Parameter bringen konnten. Dieses Beispiel veranschaulicht, wie der objektorientierte Zugang bis hin zur materiellen Umsetzung eine Referenzkette spannt; vom Parameter bis zu den Steuerungsprogrammen der Fertigungsmaschinen. Er lässt aber offen, an welchen Zahlen oder Vorgaben sich die Parameter orientieren. Ob sie wie bei Trummer einen Bezug zu metrischen oder nicht-metrischen Eigenschaften haben oder wie bei Rocker in Abhängigkeit einer weiteren Kurve stehen, bleibt ungeklärt. Eine grundlegende Eigenschaft der Modelle wurde demgegenüber bereits durch die Verwendung einer dreiachsigen CNC-Maschine getroffen: Durch den Arbeitsraum des Fertigungsautomaten definiert, konnten nur plane Elemente (wie Platten) bearbeitet werden.

> This Chapter presents a critique of a small pavilion that we (Objectile) designed as an experiment in our investigation of what we call »fully" associative architecture. Although the pavilion is a small piece of architecture, it was designed as »fully associative", which means that by controlling a few points we could modify the geometry of the pavilion and regenerate approximately 800 machining programs needed to manufacture its parts. (Cache, 2003, S. 203)

Essenziell im Zugang von Cache ist, auf welche Parameter die so erstellte Kette hinaus läuft. Er kritisiert zusammen mit Beaucé die Beliebigkeit, mit der in der »Non-Standard Architecture« Werte für Parameter festgelegt werden als Auswuchs einer romantisch verklärten Künstlerattitüde. Für die beiden steht die Referenzkette, wie bei Trummer, mit sozialen und historischen Umständen in Beziehung. Sie können dazu auf eine ganze Stilrichtung verweisen, die sich das Ausreizen digitaler Produktionsmöglichkeiten auf die Fahnen geschrieben hat. Die Methode hat aber auch noch einen anderen Zugang als den formalistischen hervorgebracht, einen sehr viel pragmatischeren: neue objektorientierte Planungssoftware. Sie bildet zum »Non-Standard« das passende Gegenstück:

> Under what conditions can a term like »non-standard architecture« have meaning? Perhaps it's easier to begin by answering in a negative way. If, indeed, a non-standard architecture consists of generating more or less soft surfaces which will then be called a building by transferring them onto a battery of production software in order to create very expensive kinds of sculpture which no longer have any relationship with the historical and social sedimentation that makes up a city, then we are only perpetuating the Romantic myth of the artist-architect. (Beaucé & Cache, 2004, S. 390)

Während bei Beaucé und Cache das Material nur eine Vorgabe darstellt, und seine anderen Einflüsse weitestgehend beschnitten werden, fußt das referenzierte Gebäude, mit all seinen Teilen, am Ende auf vielen mehr. Diese Menge an Eigenschaften versuchen die als BIM[33] bekannten Techniken in den Planungsprozess mit einzubauen. Die Digitalisierung ermöglicht es, die einzelnen Objekte mit zusätzlichen Angaben in einer Datenbank zu vernetzen, und den Informationsmangel, den die grafische Schnittstelle bildet, zu kompensieren. Das BIM nutzt diese Datenbank, um daraus verschiedene Ansichten des Entwurfs zu generieren. Wie bei einem dreidimensionalen Modell die zweidimensionalen Ansichten abgeleitet werden, sind hier auch tabellarische Auflistungen oder Diagramme diverser Merkmale möglich. Die Vernetzung der Daten vernetzt auch die Arten ihrer Einflussnahme auf das gesamte Modell und die Bauteile miteinander. Wechselt man in der Zelle einer Tabelle den Wert für die Höhe einer Trittschalldämmung, ändern sich automatisch alle damit verbundenen Pläne und Listen. Diese Methode führt das Datenmodell als zusätzliche Referenzebene des umzusetzenden Entwurfs ein, auf welches für Architekten lesbare Grafiken verweisen. Wobei auch hier von Seiten der Programme in

33 BIM, Building Information Modelling

der Ebene der Referenz assoziative Geometrien Beziehungen unter den Elementen definieren können, und diese nicht nur aus den Daten abgeleitet werden.

> BIM represents a major departure from the traditional computer-aided drafting method of drawing with vector file-based lines that combine to represent objects. It models the actual parts and pieces being used to build a building. While line drawings can be interpreted by people, they cannot be readily interpreted by computers. BIM represents object parametrically in 3-D; these objects reconfigure themselves automatically when software instructions are invoked to show different views in accordance with defined rules. As 3-D objects are machine-readable, errors can be avoided at all phases of design and construction in real-time, as opposed to the time honored method of post facto drawing review. (Forbes & Ahmed, 2011, S. 216)

Dass das BIM im Vergleich zu genereller Designsoftware auch Probleme mit sich bringt, erläutert Eastman am Beispiel des Entwurfs einer Wand. Er erkennt in ihr das Grundelement jedes Gebäudes, das als Hauptbestandteil von jeder derartigen Lösung unterstützt wird. Wenn grafische Eingaben der Benutzer nicht in das darunter liegende Datenmodell übersetzt werden können, stößt BIM schnell an die Grenzen seiner. Man kann sich nur in den Bereichen bewegen, die von den programmierten Funktionen und eingegebenen Informationen abgedeckt werden, was Eastman als »Eingebettetes Entwurfswissen« beschreibt. Aus der Sicht der Actor Network Theory kann man dieses Entwurfswissen als eine Delegation von Entwurfsarbeit verstehen, über welche die Hersteller und Programmierer der Datenstruktur Einfluss nehmen. So gesehen verursachen die erwähnten Einschränkungen, genau wie die neu geschaffenen Zugänge, eine Drift der Ziele des Architekten.

> Parametric building objects in BIM tools encapsulate design knowledge and expertise. The embedded knowledge facilitates definition and automatic editing. It distinguishes a BIM design tool from a more general parametric modeling tool, such as Solidworks, CATIA or Autodesk Inventor. The most basic parametric capability of an architectural BIM tool is the layout of walls, doors and windows. All BIM tools allow easy placement and editing of wall segments and insertions of doors, windows and other openings. Some systems maintain the topological relations between walls and respond to changes to the walls they built into. Most of the wall objects used in BIM design tools also incorporate layers of construction as a set of ordered parameters of the wall defined in a vertical section, with a structural core, optional insulation, and layers of finishing on both sides, to obtain a built-up wall and implicit material quantities, based on the wall area. ArchiCad and Revit support varying section properties along the vertical section, allowing horizontal variation of construction and finishes. Changes in the horizontal direction require a change in the wall element. We can say that the wall model incorporates this level of design knowledge. BIM tool users will encounter the limitations of this level of definition of

> a wall if they try to design outside this limited conceptualization. (Eastman, 2006, S. 164)

Mit dem Wechsel der Bedeutung der Grafik, vom Symbol zum Objekt, geht eine Veränderung des Digitalen Entwerfens einher. Dadurch, dass die Referenzkette vom Ende der Materie weg geschmiedet wird, und nicht wie bisher von der Form, ergibt sich mitunter auch das Risiko während des Prozesses in einen gewissen »Materialismus« zu verfallen. Während der Formalismus seine Freiheiten mit dem Problem der Realisierbarkeit bezahlt, ist es hier genau umgekehrt. Das Entwerfen wird zu einer Aufgabe des Auswählens, Anpassens und Kombinierens von zuvor definierten Objekten. Es fehlt eine adäquate Vermittlung des Bauteilverhaltens, und die Möglichkeit dieses zu manipulieren und zu entwerfen. Die beiden Ansätze Formalismus und Materialismus, in welche die digitalen Entwurfsmethoden anhand ihrer Mobilisierung bislang eingeteilt wurden, beschreiben die beiden Enden einer Kette, die bei Latour als »Naturpol« und »Subjekt/Gesellschaftspol« bezeichnet werden. Er entwickelt anhand der Actor Network Theory das »Verallgemeinerte Symmetrieprinzip« [34] (Latour B. , 2008, S. 127) um eine Alternative aufzuzeigen. Die Referenzkette teilt sich dabei in zwei Pole: Auf der Seite der Form steht bei ihm die Repräsentation von Natur als Ergebnis einer wissenschaftlichen Übersetzungskette und auf der anderen die Materie, das Objekt oder die Natur:

> Für die erste Form der Kritik zählen Objekte nicht; sie bilden nur die Leinwand, auf die die Gesellschaft ihren Film projiziert. Für die zweite sind die Objekte jedoch so mächtig, dass sie die menschliche Gesellschaft gestalten; ausgeblendet wird dabei die gesellschaftliche Konstruktion der Wissenschaften, die diese Objekte hervorgebracht haben. Objekte, Dinge, Konsumgegenstände, Kunstwerke sind entweder zu stark oder zu schwach. Aber noch seltsamer sind die sukzessiven Rollen, die man der Gesellschaft gibt. In der ersten Form von Kritik ist die Gesellschaft so mächtig, dass sie aus sich selbst heraus existiert. Wie das transzendentale Subjekt, an dessen Stelle sie tritt, hat sie keine Ursache. Sie ist so ursprünglich, dass sie in der Lage ist, ihren Gegenpart zu formen und zu gestalten; dieser stellt kaum mehr dar als beliebige und formlose Materie. In der zweiten Form von Kritik ist die Gesellschaft jedoch machtlos geworden und wird nun ihrerseits von mächtigen objektiven Kräften bestimmt, die ihr Wirken vollständig determinieren. Die Gesellschaft ist entweder zu mächtig oder zu schwach gegenüber Objekten, die abwechselnd zu mächtig oder zu gleichgültig sind. (Latour B. , 2008, S. 71f)

34 Auf keinen Fall zu verwechseln mit der geometrischen Symmetrie. Während es bei der einen um ein rein formales Spiegeln von Elementen anhand einer Achse geht, verwendet Latour diesen Begriff um die Bewegung entlang der Referenzkette zu beschreiben. Ihre Richtung soll nicht von einem Pol zum anderen weisen, sondern sich symmetrisch von der Mitte weg ausbreiten.

Latour zeigt das Spannungsfeld, das von den beiden Polen ausgeht: Die »machtlose Gesellschaft«, die von »objektiven« Kräften bestimmt wird, ist vergleichbar mit den materialistischen BIM-Ansätzen im Digitalen Entwerfen. Der Architekt hat sämtlichen Einfluss an der Formgebung eingebüßt und die Bauteile bestimmen weitgehend selbsttätig den Entwurf. Im Gegensatz dazu stehen formale Methoden, wie der parametrische Entwurf eines formalen Kontinuums, in keinem Bezug zur Materie. Latour beschreibt das als Entwicklung zum transzendentalen Objekt. Entweder verliert in diesen Ansätzen der Architekt seinen Einfluss, oder er verliert jeden Realitätsbezug (ist die Kette über das Verwenden von Parametern bis zu anderen Entscheidungsträgern vorgedrungen, treten in diesem Szenario natürlich sie an die Stelle des Architekten). Die Actor Network Theory hat einen dritten Zugang für das dargelegte Dilemma parat, der sich auf eine neue Betrachtungsweise der Übersetzungen stützt. Dabei werden die sogenannten »Quasi-Objekte« und »Quasi-Subjekte« ins Rampenlicht gerückt, die in bisherigen Theorien als reine Übermittlungsstufen galten:

> Die Modernen wussten genau, wie sie dieses Reich zu denken hatten. Sie ließen die Quasi-Objekte nicht durch Leugnung und Verneinung verschwinden, als wollten sie sie verdrängen. Im Gegenteil, sie erkannten ihre Existenz an, aber nahmen ihr jede Relevanz, in dem sie aus vollwertigen Mittlern bloße Zwischenglieder machten. Ein Zwischenglied wird zwar als notwendig angesehen, aber seine Aufgabe besteht nur darin, die von einem Pol der Verfassung stammende Energie zu transportieren, zu übertragen, zu »übermitteln«. Für sich genommen ist es leer; es kann nur mehr oder weniger getreu oder mehr oder weniger abwegig übermitteln. Ein Mittler jedoch ist originäres Ereignis. Er schafft, was er übersetzt, mit gleichem Recht wie Entitäten, zwischen denen er seine Mittlerrolle spielt. Sobald wir alle Akteure diese Mittlerrolle wieder zuerkennen, hat sich die Welt als solches nicht verändert. Sie wird nach wie vor von denselben Entitäten bevölkert, aber sie ist nicht mehr modern, und wird, was sie nie aufgehört hat zu sein, nämlich nichtmodern. (Latour B. , 2008, S. 104f)

Die auf den letzten Seiten beschriebene Rolle der Übersetzung auf das Entwerfen, und die Rolle der verwendeten Techniken zeigen, wie sich die hier beschriebenen Mittler einbringen. Am Beginn des Ausweges steht, den Mittlern ihre Rolle als Aktanten zuzukennen. Auf diese Weise werden sie zum aktiven Teil des Entwurfkollektivs und werden in ihrer Bedeutung nicht mehr übergangen. Auf die gleiche Weise ändert die Erkenntnis, dass allen Akteuren eine solche Mittlerrolle immanent ist, die Vorstellung die man bisher von Architekten hatte. Die bislang vorgestellten Referenzketten im digitalen Entwerfen haben mit diesen »modernen« Einteilungen zu kämpfen, lässt sich doch bislang keine finden, die in passender Weise die notwendige formale Freiheit mit einem rationalen Realitätsbezug

in Einklang bringt. Latour schlägt hierzu vor, nicht mehr von den Polen ausgehend zu denken, sondern eine neue Mitte anzuvisieren:

> Wenn wir das Reich der Mitte für sich genommen entfalten wollen, müssen wir also die allgemeine Form der Erklärung umdrehen. Der Punkt der Spaltung – und der Verbindung – wird zum Ausgangspunkt. Die Erklärungen verlaufen nicht mehr nur von der reinen Form zu den Erscheinungen, sondern vom Zentrum zu den Extremen. Letztere sind nicht mehr der Verankerungspunkt der Realität, sondern provisorische und partielle Resultate. (…) Wir brauchen unsere Erklärungen nicht mehr an den beiden reinen Formen Objekt und Subjekt/Gesellschaft festzumachen: Diese sind vielmehr die partiellen bereinigten Resultate der zentralen Praxis, welche allein uns interessiert. Auch in der von uns gesuchten Erklärung werden Natur und Gesellschaft enthalten sein, aber als Endresultat, nicht als Ausgangspunkt. Die Natur dreht sich, aber nicht um das Subjekt/die Gesellschaft. Sie dreht sich um das Ding und Menschen produzierende Kollektiv. Das Subjekt dreht sich, aber nicht um die Natur. Es dreht sich um das Kollektiv, aus dem heraus Menschen und Dinge erzeugt werden. Endlich ist das Reich der Mitte repräsentiert. Naturen und Gesellschaften sind seine Satelliten. (Latour B. , 2008, S. 105f)

Mit dem Fokus auf das Zentrum und die Zwischenschritte kommt der Fokus auf das Kollektiv, um das sich alles dreht. Die Gemeinschaft menschlicher und nichtmenschlicher Aktanten, die selbst wiederum Aktanten hervorzubringen vermag, ist ein neuer, dritter Orientierungspunkt. Form und Materie werden von konkurrierenden Ausgangspunkten zu Randpunkten, die sich als Folge der Übersetzungen ergeben. Nach diesem Symmetrieprinzip werden der Entwurfsprozess und das Entwurfskollektiv zum Mittelpunkt. Kern der Überlegungen ist das Konstrukt des Aktanten, das es vollbringt, Objekt und Subjekt zu vereinen. Er ermöglicht es, für das Entwerfen eine neue Technik zu erschließen, und damit formale wie materielle Annäherungen zu ersetzen. Im direkten Vergleich zu bestehenden Methoden werden den Objekten zu den Handlungen, die ihnen als Materie immanent sind, zusätzliche Entwurfsaufgaben übertragen. Konnten bisher Eigenschaften von Objekten beim Einsatz von BIM Lösungen und vergleichbaren Entwicklungen den Entwurf durch die Grenzen der Software entscheidend beeinflussen, wurden gleichzeitig dort fertigungstechnische Grenzen aufgezeigt, wo Programme zum Erstellen von möglichst ausgefallenen Formen eingesetzt wurden. Mit dem neuen Entwerfen von Digitalen Aktanten, die sowohl formalistische wie materialistische Ziele zugeschrieben haben, kann sich nun der Entwurf aus der Mitte der Referenzkette heraus entwickeln. Sowohl die Form als auch das materialisierte Gebäude gehen dabei aus dem Zusammenspiel alle jener Aktanten hervor die sich zwischen ihnen befinden, und können so weder auf das eine noch auf das andere reduziert werden. Mit dem Delegieren von

formaler Entwurfsarbeit spiegelt die im nächsten Kapitel beschriebene Technik Latours Konzept von Symmetrie wider. Genau dieses Delegieren bildet die Weiterentwicklung zu den bislang behandelten Methoden, und eine neue Entwurfsperspektive.

5. DIGITALE AKTANTEN

Entwerfende »Objekte« im Zentrum der Referenzkette

EINE ANDERE REFERENZ: DIE MENGE

Bislang wurden die Mobilisierungen von vier digitalen Entwurfsmethoden untersucht: die der Formalen Grammatik, des Parametrischen Entwerfens, der Assoziativen Geometrien und das Building Information Management (BIM). Dabei ging es nicht nur um ihre variierenden Dimensionen, sondern auch darum eine sich wiederholende Übersetzungstaktik aufzuzeigen. Die mathematisch definierte Kurve hat sich dabei als essenziell für die Beschreibung von Punkten und Geometrien herauskristallisiert, und nicht nur in formalen Verfahren Anwendung gefunden. Parsons und Stulen, Lynn und Rocker haben alle mit der Kurve auf eine Technik zurückgegriffen, die einzelne Werte zuverlässig referenzieren kann. Sie ist damit das, all diesen Bemühungen zu Grunde liegende, Element, auf dem die verschiedenen Referenzketten aufbauen. Mit der Anwendung der ANT auf das Digitale Entwerfen spielen die Quasi-Objekte eine neue Rolle und setzen eine neue Form der Referenz voraus. Anstelle der beschriebenen Differenzial- und Integralrechnungen lassen sich die einzelnen, errechneten Punkte in einem ersten Schritt in eine andere mathematische Form und Referenz bringen: die Menge.

> Traditionally, a line was seen as holistic – traced by continuous motion and not made up of discrete elements. Points were seen as *locations* on the line. The line as a whole was taken to be ontologically independent of any locations one might happen to pick on it, just as a bird resting on a telephone line is ontologically independent of any location on the phone line. According to the new metaphor for a line, A Line Is A Set of Points, a line is constituted by its points and does not exist independent of its points. It is a *composite* entity, rather than a holistic one – like a flock of birds. Just as there is no flock without birds, there is no line without the points. (Núnez & Lakoff, 1998, S. 93f)

Während gerade bei Lynn und Rocker die Kurve noch als Kontinuum, und damit in weiterer Abhängigkeit der Entwurf als Serie, dargestellt wird, führt eine Definition als Menge von Punkten zu einem neuen Verständnis. Das Ganze wird zu einer Zusammen-

stellung. Die 17 Punkte, die durch Stulens Rechenmaschine zu 200 wurden, zeigen das genauso wie Albertis Polarkoordinatenverfahren. Die zwischen den betroffenen Aktanten zu übertragende Information war eine Menge von einzelnen Punkten. Geändert hat sich das erst, als die Menge anders dargestellt werden konnte. Das Formulieren einer Formel brachte die Menge wortwörtlich in eine neue, abstraktere Form. Dieser Übersetzungsprozess ermöglichte es, unendlich viele Elemente einer Menge durch ihre Position (vergleichbar mit einer geografischen Adresse in einem eindimensionalen Raum) abrufbar zu machen. Die so geschaffene neue Referenz verliert zwar durch die Übersetzung Information (wie etwa die genaue Position der Punkte), schafft aber auch eine neue Übersichtlichkeit (die Funktion). Lynn und Rocker nutzen dieses Verfahren wiederum, um ihren Parametermodellen eine Komplexität aufzusetzen, die ihre Versionen der Modelle zu einem Kontinuum zusammenfasst. Mit der Methode der Menge verliert eine Kurve jedoch ihre Gesamtheit. Während Punkt und Gerade in der modernen Mathematik nicht definiert werden (Bronstein, Semendjajew, Musiol, & Muehlig, 2008, S. 131), sondern durch die Inzidenzaxiome[35] beschrieben sind, gibt es für die darüber hinaus bestehenden Elemente sehr wohl Definitionen:

> Eine Strecke *AB* enthält genau die Menge aller Punkte einer Geraden, die zwischen zwei Punkten *A* und *B* dieser Geraden liegen, die Punkte *A* und *B* inbegriffen. (Bronstein, Semendjajew, Musiol, & Muehlig, 2008, S. 131)

Das neue Verständnis der Kurve als Menge aller Punkte wirft die Frage nach einer genauen Definition einer Menge auf. Gleichzeitig lässt sich damit der erfolgreiche Übersetzungsprozess besser nachvollziehen: Die mathematische Funktion ist demnach ein Hilfsmittel, einen Punkt in einer unendlich großen Menge von Punkten (die eine Kurve darstellt) eindeutig zu referenzieren. Das »natürliche Auftreten« der Punkte ist damit nicht mehr die Funktion, sondern die Menge in der sich alle versammeln. Die Funktion hilft nur beim Aufspüren und ausfindig machen der Nadel im Nadelhaufen. Georg Ferdinand Ludwig Philipp Cantor, ein deutscher Mathematiker, publizierte 1895 erstmals seine Theorien zur »Mengenlehre«, mit der er einzelne Elemente zu einem größeren Ganzen zusammenfasste. Dabei ist herauszustreichen, dass sie ausschließlich zusammenfasst und durch das Zusammenfassen nicht mehr oder weniger damit anstellt.

35 Diese sagen z. B. aus, dass durch je zwei verschiedene Punkte je eine Gerade geht oder auf jeder Gerade mindestens zwei Punkte liegen (Vgl.Kowol, 2009, S. 59).

> Unter einer »Menge« verstehen wir jede Zusammenfassung *M* von bestimmten wohlunterschiedenen Elementen *m* unsrer Anschauung oder unsres Denkens (welche die »Elemente« von *M* genannt werden) zu einem Ganzen. (Cantor, Dedekind, & Fraenkel, 1932, S. 282)

Die Menge, die beliebig viele Elemente zusammenfasst, trifft keine Aussage über das Verhältnis in dem diese zueinanderstehen. Ob sie sich wechselseitig beeinflussen, oder zueinander in keinem Kontakt stehen ist für die Menge nicht nur von keiner Bedeutung, es wird von ihr auch nicht dargestellt, und geht deshalb in dieser Übersetzung verloren. Will man hingegen Elemente gruppieren und zugleich ihre Beziehungen darstellen, eignet sich das theoretische Pendant zur Menge: das System. Ein System ist die Menge aller Elemente die, wie Daenzer schreibt, zueinander in einer wechselseitigen Beziehung stehen[36]. Genauer setzt sich 1976 Volker Oberkampf mit dem Verhältnis von System und Menge auseinander. Er zeigt, dass alles zu einer Menge zusammengefasst werden kann, aber im Unterschied dazu, eine Auswahl von Entitäten nicht zwingend ein System bildet. Auf diese Weise ist das System als Erweiterung der Menge denkbar:

> Charakterisiert wird eine Menge dadurch, dass man angibt, welche *Objekte* zu ihr gehören. Kann man eindeutig entscheiden, welche der möglichen Objekte (Elemente) zu einer Menge gehören, dann ist die Menge selbst vollständig definiert. Jedes System lässt sich auf eine solch vollständig definierte Menge (einfaches Ganzes) zurückführen. So ist z. B. das gesamte Universum ein System, physikalische Prozesse und deren Objekte bilden Systeme, eine Unternehmung ist ebenfalls ein System usw. Jedoch bilden z. B. ein oder mehrere Stücke Eisen zwar eine Menge, aber noch kein System. Ähnlich stellen eine Reihe von Markovprozessen, deren Ergebnis zwar jeweils unterscheidbare Zustände ergeben und deshalb insgesamt eine Menge sind, noch lange kein System dar. Um ein System zu erhalten bedarf es einer weiteren Eigenschaft. Wenn zwischen den Elementen einer vollständig definierten Menge *Beziehungen*, *Relationen* oder *Zusammenhänge* bestehen, dann spricht man von einem System. (Oberkampf, 1976, S. 35)

Dieses Wissen lässt sich leicht auf Rockers Versioning-Ansatz der Cross Scalar Seriality übertragen: dort werden die einzelnen Parameter der nebeneinander gereihten Instanzen eines definierten Modells in Abhängigkeit einer Kurve gestellt. Damit sind die einzelnen Parameter voneinander isoliert, sind aber in einer Menge von Zahlen über die Kurvenfunktion referenzierbar. Definiert man den Parameter im Gegensatz dazu, wie in der Actor Network Theory, nicht nur als Mittler, sondern als eigenständigen Aktanten, der zu

36 Unter einem System soll die Gesamtheit von Elementen verstanden werden, die miteinander durch Beziehungen verbunden sind. (Daenzer, 1976, S. 11)

anderen in Beziehung steht, verliert man mit der Zusammenfassung zu einer Menge grundlegende Informationen. Aktanten bilden keine Mengen, sie bilden Systeme oder Netzwerke. Mit dem Fall der Menge ist unweigerlich auch die Darstellung als mathematische Funktion nutzlos. Um die Aktanten auf einer anderen Referenzebene darzustellen, kann man, und muss man wegen des Mangels an Alternativen sogar auf die eingangs beschriebene Kartografie zurückgreifen.

> The primary image is no longer the image of the object but the image of the set of constraints at the intersection of which the object is created. This object no longer reproduces a model of imitation, but actualizes a model of simulation. The anatomical gestures of the user, the surface of the set of constraints of the material, the curves of optimization and of management, all constitute the geography that governs the object. (Cache, [1983] 1995, S. 97)

Der Entwurf findet nicht länger in einer Ebene der Darstellung als Objekt statt, sondern in der von Beziehungen und Verflechtungen. Cache schreibt davon, wie die verschiedenen Einflüsse, die auf den Entwurf wirken zusammen die »Geografie« zur Bestimmung des Objektes bilden. Er definiert auch, dass sich das Entwurfsobjekt vom Umsetzen seiner grafischen Vorstellung zum Aktualisieren seiner Simulation entwickelt. Auf das Projekt MACOSPOL zurückgegriffen, und in Anbetracht der so treffend als »Geografie« beschriebenen Hintergründe, wäre die logische abstrakte Referenz von parametrischen Aktanten eine Karte, in der auch ihre Beziehungen untereinander dargestellt werden. Diese Karte kann die Abhängigkeit von Parametern im Digitalen Entwerfen beenden, und eine neue Referenzebene, die auf digitale Aktanten verweist, einführen.
Hierin unterscheidet sich die präsentierte Herangehensweise massiv von den bislang existierenden, weil sie zugunsten eines höheren Informationsgehalt der Referenz auf eine maximal zusammenfassende Abstrahierung, die eine mathematische Funktion darstellen würde, verzichtet. Zwar verliert man dadurch die Möglichkeit einer einfachen Manipulation zum Herbeiführen diverser formaler Ziele, aber dafür gewinnt man eine weniger abstrakte Darstellungsform, die das Potenzial hat neue Übersichtlichkeiten und Erkenntnisse hervorzubringen. Dadurch, dass mit dieser Methode mehr Information übersetzt wird, eröffnet sich auch eine neue Möglichkeit der Gestaltung von komplexen Digitalen Aktanten auf die damit verwiesen werden kann. Ohne der Zuspitzung des Entwerfens auf die Skalierung und Auswahl weniger Parameter werden die Möglichkeiten des Architekten auf den Entwurf Einfluss zu nehmen wieder zahlreicher und vielschichtiger.

DAS ENTWERFEN VON AKTANTEN: DIE WERKZEUGE

Im Gegensatz zu den bislang dargestellten Theorien, wird im hier beschriebenen Experiment »CDN«[37] eine Variante des Digitalen Entwurfs getestet, die die Charakteristika der bestehender Methoden vereint. Zu diesem Zweck werden assoziative Geometrien mit komplexen Regelsätzen kombiniert, um die »Objekte« um formale Aufgaben zu erweitern. An die so entstehenden digitalen Aktanten werden nicht nur die materiellen Eigenschaften ihrer Umsetzungen delegiert, sondern auch die Entwurfsintentionen des Architekten (oder genauer: des Planerkollektivs). Die digitalen Entwurfsaktanten erhalten in dieser Methode einen neuen Stellenwert, wird doch davon ausgegangen, nun vom Zentrum der Referenzkette aus zu entwerfen. Das Experiment basiert weiters darauf, dass sämtlichen, als Werkzeug verwendeten Aktanten von Natur aus ein gewisses Skript immanent ist. Diesem Verhalten werden über den Umweg einer anderen Darstellung neue Verhaltensmuster hinzugefügt. Anstelle von geometrischen Bedingungen treten neue Anweisungsarten, die sich auf die Umwelt beziehen. Im Unterschied zur Cross Scalar Seriality wird den einzelnen Elementen ihr Input nicht mehr von einer zentralen Funktion diktiert, sondern sie stehen untereinander in Verbindung. Als Weiterentwicklung zur assoziativen Geometrie besitzen die Elemente nicht mehr nur rein reaktives, sondern ein viel komplexeres, aktives Verhalten. Aktive digitale Aktanten stellen eine Erweiterung sowohl von Objekten als auch von formalen Kontinua dar. Ein Entwerfen aus der Mitte der Referenzkette in beide Richtungen wird auf diese Weise möglich.
Wie die Geschichte zeigt, ist eine wichtige Frage im Entwerfen die danach, welche Aktanten (und damit welche Entwickler) man mit ins Boot holt. Für diesen Ansatz des CDN bot sich die Software Grasshopper der Firma McNeel[38] besonders gut an, die als grafischer Algorithmuseditor für das Programm Rhinoceros 3D (der gleichen Firma) fungiert. Zum Einen ist die ungewöhnliche Art der Codedarstellung von Grasshopper ein Entscheidungsgrund gewesen, zum Anderen aber auch die OpenNURBS[39] Initiative von McNeel. Mit ihr steht die gesamte ».3dm« Dateiformatspezifikation mitsamt einer ausführlichen Dokumentation und Codebibliotheken als Open Source Toolkit zur Verfügung. Da zur Zeit noch keine Software existiert, die den Anforderungen einer Erweiterung von Objekten zu Agenten entspricht, mussten hier Anpassungen vorgenommen werden. Dass sich

37 CDN, Collaborative Design Network

38 Grasshopper Build 0.80001, Rhinoceros NURBS modelling for Windows Version 4.0 SR8

39 Mehr zu OpenNURBS auf http://www.opennurbs.org/

das System der offenen Schnittstelle für die Hersteller der Software bereits als erfolgreich herausstellt, zeigt auch das Abschneiden im Wettkampf mit den konkurrierenden Produkten:

> Grasshopper seems to be winning out in the competitive struggle for domination as the preferred tool for scripting – at least in the avant-garde segment of the discipline – both on the diagrammatic level as well as on the level of concrete modelling. The great advantage of Grasshopper is that it transposes most of the scripting syntax into a graphic network language. The system of parametric dependencies that organizes the internal variability and differentiation of the models can now be configured and manipulated via a second order diagram that controls the first order diagram or model. The fact that this new, crucial domain of design decisions – the choice and elaboration of systematic correlations between the variable elements of the design – is now brought back into the ambit of architecture's specialized medium of communications is perhaps the precondition for the full-blown proliferation of parametric scripting techniques within architecture and the design disciplines. (Schumacher, 2011, S. 354)

Schumacher hebt hier besonders die Programmierschnittstelle von Grasshopper hervor, mit der es möglich ist die Beziehungen der einzelnen Codeelemente zu manipulieren. Die Darstellung von Code, die er als Diagramm bezeichnet, entspricht dabei eher einem Netzwerk und bei umfangreicheren Projekten gar einer Landkarte. Das Beispiel Grasshopper zeigt, welchen Einfluss eine andere Darstellungsform und eine andere Referenzebene haben können. Mit diesem Programm fällt zum einen die Hemmschwelle zum Programmieren bei vielen interessierten Architekten, andererseits läuft man Gefahr in eine, wie bereits von Bernard Cache kritisierte, Beliebigkeit abzudriften. Auf jeden Fall trägt die netzwerkartige Darstellung zu einem neuen Verständnis von Code bei, das sich vom Bild einer Menge an Programmzeilen zu einem eines Systems von Komponenten entwickelt hat.

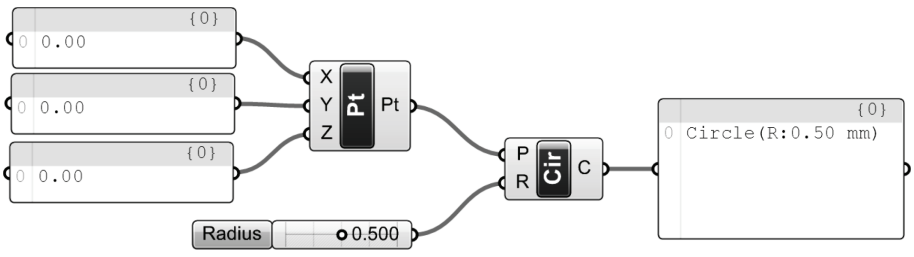

Abb. 1: Grafische Algorithmuskomponenten in Grasshopper

Die Abbildung zeigt, wie Code im beschriebenen Plug-in statt als Text ikonografisch dargestellt wird. Die grafischen Komponenten verweisen auf Anweisungselemente, die als jeweils eigenständige Scriptbausteine entwickelt wurden und auch so funktionieren. Es kann aus einem großen Angebot an vorhandenen Codestücken ausgewählt werden, und diese können, soweit es die Programmlogik und der Datenfluss erlauben, frei angeordnet und verbunden werden. Die Einschränkungen die es gibt, bestehen vor allem durch die definierten Formen der Eingangssignale, wie im gezeigten Beispiel die Zahl als Abstand auf einer Koordinatenachse, oder der Punkt als Dreiertupel[40] für den Mittelpunkt des Kreises. Daneben existiert eine massive Beschränkung dadurch, dass kein zyklischer Datenstrom hergestellt werden kann, worauf aber später noch genauer eingegangen wird. Als Ausweg aus den Einschränkungen in der Auswahl von Codeelementen bietet Grasshopper die Möglichkeit Code für zur Verfügung stehende Skriptkomponenten selbst zu verfassen. Ähnlich der Methode von BIM Software, können so Bauteile selbst angelegt, Anweisungen erstellt und Handlungen delegiert werden. Anhand der Definition eines Kreises kann eine derart individualisierte, also selbst entworfene, Komponente auch andere Funktionen und Einschränkungen umfassen, und so den weiteren Entwurf beeinflussen.

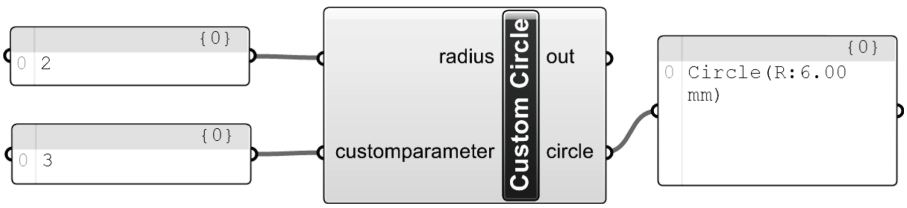

Abb. 2: Angepasste Skriptkomponente zum Erstellen eines Kreises

Als Beispiel wurde ein neuer Baustein zum Kreisherstellen entworfen, der sich vom Standardbauteil in Bezug auf die Eingabemöglichkeiten unterscheidet. Zusätzlich zu einem Radius kann nun auch ein Größenfaktor angegeben werden, über den sich der herzustellende Kreisdurchmesser berechnen lässt. Neben der gewonnenen Einstellmöglichkeit auf der einen Seite, ist auf der anderen aber eine Freiheit des Bauteils verloren gegangen: Der Mittelpunkt des Kreises wird nun unveränderbar mit dem Ursprung des Koordinatensystems angenommen. Eine weitere Möglichkeit über den Bauteil den Entwerfer zu

40 Ein Tupel ist eine endliche Liste mathematischer Objekte, ein Dreiertupel ist eine Liste mit drei Einträgen (auch Tripel genannt)

beeinflussen, liegt in der Definition des Datentyps des Eingangssignals. Bieten die beiden Werte für Radius und Größenfaktor hier noch als »double« kaum Einschränkungen, würde eine Definition als »integer« dazu führen, dass nur ganze Zahlen eingegeben werden könnten[41]. Wurde im vorherigen Beispiel noch dargestellt, wie der Mittelpunkt des Kreises aus dem Verbinden von vier Elementen hervorging, ist der Mittelpunkt in dieser Eingriffs- und Referenzebene nicht mehr direkt ersichtlich. Will man ihn trotzdem ändern, muss man in den Quelltext der neuen Komponente vordringen:

```
01  Rhino.Geometry.Point3d c_center = new Rhino.Geometry.Point3d();
02  double c_radius = new double();
03  c_center.X = 0.0;
04  c_center.Y = 0.0;
05  c_center.Z = 0.0;
06  c_radius = Math.Abs(radius * customparameter);
07  circle = new Rhino.Geometry.Circle(c_center, c_radius);
```

Grasshopper übersetzt die editierbaren Codezeilen in den Code einer privaten Prozedur innerhalb der öffentlichen Klasse, auf die das Icon der Skriptkomponente verweist. Dieser Aufbau stellt die Zugriffsrechte auf die Codezeilen dar, der Benutzer kann nur innerhalb einer Klasse Programmtext erstellen und bearbeiten. Ohne genauer auf die Informatik dahinter einzugehen kann festgehalten werden, dass damit auch die Skriptkomponente von weiteren Einschränkungen betroffen ist. Durch die Befehlszeilen lässt sich der Ablauf erkennen, nachdem über die Eingabe von Werten die Ausgabe des Kreises erfolgt. Zuerst werden die beiden Variablen `c_center` und `c_radius` definiert. Der Mittelpunkt wurde als Punktklasse gesetzt, und hat damit eine gewisse Datenstruktur zugewiesen bekommen. Danach werden dem Punkt die Koordinaten übergeben, und der neue Radius wird über eine Multiplikation mit dem Größenparameter berechnet. Dabei wird nur der absolute Wert übernommen, um nicht über einen negativen Radius einen Fehler zu produzieren. Abschließend wird eine Instanz der Kreisklasse erstellt und ausgegeben.
Im Vergleich zur Netzgrafik kann man klar die geordnete Reihenfolge der Anweisungen erkennen. Wobei nicht übersehen werden darf, dass die Abfolge in der bildlichen Darstellung ebenfalls nur von links nach rechts, also zeitlich geordnet, von Statten geht. Grasshopper übersetzt die Grafik in ausführbaren Code, der bei jeder Parameteränderung

41 Der Datentyp „double" steht für „doppelte Genauigkeit", das heißt, dass acht statt vier Byte (also 64 Bit) im Speicher dafür verwendet werden. Dem gegenüber werden als „integer" nur ganze Zahlen gespeichert, wobei der maximale Wertebereich in verschiedenen Varianten des Datentyps von 3 bis 39 Dezimalstellen schwankt.

durch den Benutzer neu ausgeführt wird. Gleichzeitig erstellt Grasshopper im Arbeitsraum von Rhinoceros eine Vorschaugrafik, mit der zu jedem Zeitpunkt das Resultat des gerade bearbeiteten Programms betrachtet werden kann. Der Entwerfer hat also drei Ebenen zur Verfügung in denen er erkennen und eingreifen kann: die Darstellung der geometrischen Elemente, die Darstellung des Komponentennetzwerks und den Quelltext, der von ihm angepassten Komponenten. Sie sind untrennbar miteinander verbunden, und können in Echtzeit aktualisiert werden.

In der Abbildung des ersten Beispiels stehen zwei Möglichkeiten der Eingabe von Werten in Grasshopper nebeneinander. Die für die Punktkoordinaten notwendigen Werte werden als diskrete Variablen übergeben, der Radius hingegen wird über den Schieberegler nur von einem Minimal- und einem Maximalwert begrenzt und kann in diesem Bereich kontinuierlich verschoben werden. Mit dem Schieberegler steht eine Interaktionsmöglichkeit zur Verfügung, welche von einem Zugang über die Ebene des Quelltextes nicht geboten wird. Die diskrete und die kontinuierliche Methode verhalten sich wie das Modell von Palladio zu dem von Greg Lynn. Bei dem einen werden anfangs die Variablen definiert und daraus eine Geometrie erstellt. Im Optimalfall musste dieser Vorgang damals nicht oft wiederholt werden, da er mit einem großen Zeitaufwand verbunden war. Bei dem anderen durchlaufen die Werte einen gewissen Bereich (gekoppelt an die Zeitleiste), und stellen so anstelle einer Form ein Kontinuum dar. Genau dieses Kontinuum, oder die Serie, findet sich auch im Schieberegler von Grasshopper. Ist das Skript erst fertiggestellt, schafft es eine unscharf definierte Form – einen Bereich in dem die Form liegt. Damit entwerfen alle Programme, die mit diesem Element erstellt werden, schlussendlich Serien oder Kontinua.

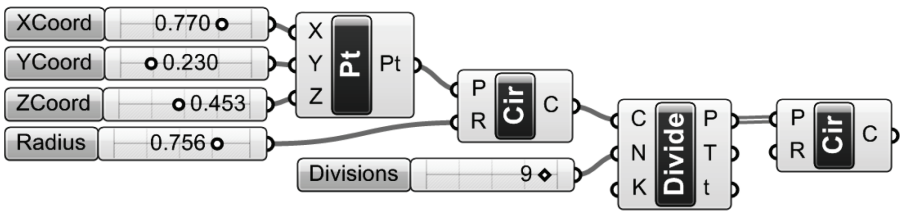

Abb. 3: Mittelpunkte von Kreisen in Abhängigkeit eines Kreises

Die Weiterentwicklung der Serie zur Cross Scalar Seriality ist, mit ihrer Abhängigkeit der Kurve von einer Kurve, ein wichtiger Bestandteil des Daten- und Ablaufnetzes in Grasshopper. Wie im obrigen Beispiel gezeigt wird, kann eine mathematisch definierte Geometrie leicht Daten für weitere zur Verfügung stellen. Hier hängt eine Kreiskompo-

nente von den Unterteilungen eines weiteren Kreises ab. Je nachdem in wie viele Abschnitte sein Umfang geteilt wurde, werden neue Kreise mit dem Zentrum auf den Trennpunkten gezeichnet. Die stark vereinfachte Abstrahierung der Abhängigkeiten soll darstellen, dass es auch in dieser komplexeren Geometrieproduktion immer auf die Ausgangswerte ankommt. Der Kreis, auf den sich die anderen beziehen, muss genauso am Beginn mit Input versorgt werden wie bei einer einfachen Serie. Der Unterschied ist, dass die ermittelten Werte nicht mehr einfach aufsteigende Zahlenreihen bilden.
Bei diesem Beispiel kommt eine weitere Besonderheit zum Tragen, die für ein Cross Scalar Seriality geeignetes Werkzeug notwendig ist: Der Datenstrom von Grasshopper kann auch Listen verarbeiten. Die den Kreis teilende Komponente liefert als Ausgabewert die Koordinatensätze der neuen Trennpunkte. Man kann in der Grafik erkennen, dass die Verbindung zwischen ihr und der Eingabe der davon abhängigen Kreise als Doppellinie ausgeführt ist. Die Darstellung weist darauf hin, dass nicht nur ein einzelner Wert, sondern eine ganze Liste von Werten übermittelt wird. Für jedes Element der Liste werden die nachfolgenden Anweisungen wiederholt, die Kreiskomponente wir also genauso oft ausgeführt wie Eingabewerte zur Verfügung stehen. Damit entsprechen die neun hier erzeugten Kreise den nebeneinander gereihten Modellen in der Cross Scalar Seriality. Grasshopper bietet sogar die Funktion Listen von Listen und Baumstrukturen zu verarbeiten. Hat eine Komponente mehrere Listen als Eingangssignal, kann der Benutzer wählen, wie die Werte miteinander zu kombinieren sind.
Der sichere und durchdachte Datenstrom den die Entwickler der Software geschaffen haben, lässt komplizierte Konstruktionen von Formen produzieren. Dem gegenüber stellt der Datenstrom aber auch ein Problem dar, sobald man versucht daran anzuschließen. Augenscheinlich wird dies, wenn man mehrere Grasshopper Skripte vernetzen, oder die Einschränkung von zyklischen Datenflüssen umgehen möchte. Hinter ersterem steht die Motivation, Anwender von Grasshopper und die von ihnen entwickelten Entwurfsaktanten genauso zu vernetzen wie es bereits mit den Programmelementen geschehen ist. Durch eine solche Vernetzung kann das Programm zu einem kollaborativen Entwurfswerkzeug werden:

> If there were just one telephone in the world, its utility would be small, with two or three not much more useful. But when millions, or hundreds of millions of people own telephones, their utility is immense. Like the telephone, one collaborative authoring tool by itself has limited collaborative value, but when millions have compatible collaboration technology, the utility of each tool is great. When a common network protocol ensures interoperability between collaborative authoring tools and authoring servers, as each tool

> adopts the protocol, the value of servers that support the collaboration protocol increases, since without any further investment in the server, they can now support an additional tool. The reverse is true as well. As more servers become available, the value of the authoring tools increases without any further investment in the tools, since there are now more individuals and organizations capable of collaboration. By creating a network protocol with a focus on interoperability, it is possible to generate network effects among compatible collaborative authoring technologies. (...) We go further and assert that collaborative authoring capability must be added to existing tools. Our hypothesis is the best way to generate network effects and to add collaborative authoring capability to existing tools is to focus on the network protocol, the mechanism by which collaborative tools communicate. (Whitehead & Goland, 2002, S. 291f)

Wie Whitehead und Goland ausführen, stehen und fallen kollaborative Konzepte mit der Technik für die Verbindung deren Teilnehmer. Daneben sucht auch Oosterhuis nach Möglichkeiten, die einzelnen Entwerfer besser miteinander zu vernetzen. Es wurde hierzu, um im CDN eine Verbindung mehrerer Desktops herzustellen, nicht direkt ein neues Protokoll eingesetzt, sondern auf die, aus dem BIM bekannte, Technik von Datenbanken gesetzt. Während eine direkte Verbindung über ein passendes Protokoll den Vorteil hat, dass sich die Teilnehmer des Entwurfskollektivs ohne zentrale Knotenpunkte vernetzen können, hat sie den Nachteil, dass die Zugriffsgeschwindigkeit drastisch fällt. Jener Nachteil der Datenbank, bringt aber auch den Vorteil, dass Zwischenergebnisse gespeichert werden können. Der im CDN eingesetzte Datenbankserver steht über Internet oder LAN[42] mit den Entwurfsaktanten in Verbindung, und muss zum Datenaustausch unter den Aktanten in Betrieb sein. Nachdem Grasshopper die Datenbankfunktion nicht abdeckt, musste diese für das Programm eigens entwickelt werden. Mit dem Datenbank Plugin »GHEloqueraDB« stehen neue Komponenten zur Verfügung, die über weit mehr Funktionen verfügen, als es bei angepassten Skriptkomponenten der Fall ist.
Mit diesen Bausteinen können Werte und Objekte in eine externe Datenbank geschrieben und von dort wieder ausgelesen werden. Darüber hinaus verfügt der Werkzeugsatz über die notwendigsten Steuerungsfunktionen für die Datenbankwartung am Server[43]. Da Grasshopper in .NET entwickelt wird[44], und von einem Objektverständnis ausgegangen wurde, fiel die Entscheidung welche Datenbank gewählt wird zugunsten der jungen und

42 LAN, Local Area Network – Ist eine lokales Netzwerk von Rechnern, z. B. in Firmen, Universitäten oder bei LAN-Parties

43 Eine genaue Auflistung der gebotene Funktionen sowie der Quelltext finden sich im Anhang A

44 RhinoCommon, die .NET Plugin SDK, ist seit Dezember 2010 als Open Source lizenziert und auf Github verfügbar (https://github.com/mcneel)

freien Software Eloquera[45] aus. Als eine objektorientierte Datenbank unterscheidet sie sich in Aufbau und Struktur von den gegenwärtig mehrheitlich eingesetzten, relationalen Datenbanken. Anstatt die Information in Tabellen aufzuteilen, und die Einträge über Schlüsselfelder identifizierbar zu machen, werden in Objektdatenbanken die zu speichernden Objekte in ihrem Datenaufbau (Vererbungshierarchie und zusammengesetzte Objekte) unverändert hinterlegt (Vgl. Heuer, Saake, & Sattler, 2008, S. 530).

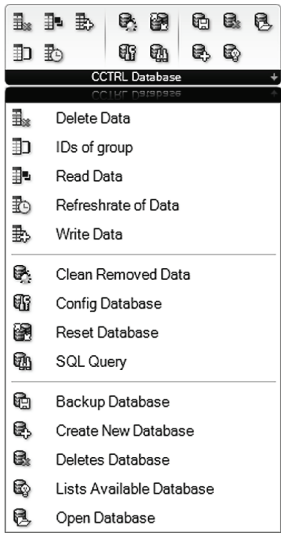

Abb. 4: Für das Experiment entwickelte Datenbankkomponenten

Neben der einfachen Einbindung die Eloquera und Rhinoceros gestatten, hat der Datenstrom in Grasshopper allerdings das Problem, dass manche Klassen nicht über das `Serializeable()` Attribut verfügen. Da viele Geometrie-Klassen in RhinoCommon auf unverwaltete C++ Klassen aufsetzen, kann Eloquera ihre Inhalte nicht korrekt übernehmen[46]. Um die Funktion des Datenaustauschens und -speicherns dennoch gewährleisten zu können, wurde für die einfachen Klassen Boolean, Integer, Number und String eine Sonderlösung entwickelt. Durch das Übertragen in eine eigene Datenklasse, mit allen notwendigen Attributen, stehen ihnen alle Funktionen der Datenbank offen. Es zeigt sich, dass wie bei Whitehead und Goland oder auch wie bei Alberti die Schnittstelle zur

45 Eloquera Database Version 3.0.0.27

46 Die Korrespondenz mit beiden Entwicklerteams zur Erörterung der Schnittstellenproblematik findet sich in Anhang B

Datenübertragen den kritischen Punkt in der Vernetzung von Entwurfsaktanten darstellt. Erst durch die Schnittstelle ist es den Aktanten möglich miteinander zu kommunizieren und Netzwerke zu bilden. Die definierten Datenklassen und die Datenbank gewährleisten den Informationsfluss und gehen einen ähnlichen Weg wie es früher die Polarkoordinaten oder heute moderne Kommunikationsprotokolle vorgezeigt haben.

DAS ENTWERFEN VON AKTANTEN: DIE ELEMENTE

Mit den Datenbankkomponenten wurde die Grundlage für den Entwurf von Agenten in Grasshopper hergestellt, erst sie haben den dafür notwendigen einfachen Datenaustausch ermöglicht. Projekte, die nach der Standardmethode in Grasshopper entworfen werden, lassen sich meistens auf wenige Parameter zurückführen. Sie erwecken deshalb den Anschein, dass es sich wie bei den Werken von Objectile um völlig assoziative Entwürfe handelt. Während das Büro von Cache, wie in der Population von Trummer, die einzelnen Objekte in Beziehung stellt, produziert die Software Strukturen, in der sich die Teile wie in der Cross Scalar Seriality von Rocker verhalten. Um den Ansatz der Objektorientierung verfolgen zu können, musste ein Weg gefunden werden, die Beziehungen der Bauteile in die Codeebene zu übertragen. Genau so, wie den Objekten eine Geometrie zugeschrieben ist, und ihnen im BIM Daten hinzugefügt werden, wird ihnen im CDN ein Verhalten eingeschrieben. Materielle Eigenschaften können dadurch auf die gleiche Weise integriert werden wie formelle, die von den Entwerfern delegiert werden. Hier hebt sich die Methode des in dieser Arbeit präsentierten Beispiels von heute verfügbaren digitalen Entwurfstechniken ab: die Neuerung, dass jedes Bauteil ein Skript ausführt, wodurch diese mit anderen direkt oder indirekt in Kontakt steht und die Interaktion mit den Planern gefördert wird anstatt sie einzuschränken, lässt das CDN aus dem Kreis der Digitalen Entwurfsmethoden hervorstechen. Mit dem Übertragen spezieller Formfindungsaufgaben an die Bauteile selbst, kann der Entwerfer mehr Ziele verfolgen und in den Entwurfsprozess integrieren. Es können einfacher komplexe Planungs- und Entwurfsaufgaben gelöst werden, ohne dass der Entwerfer sämtliche Faktoren selbst berücksichtigen und mitberechnen muss – und ohne dass komplexe Systeme durch eine vereinfachende Übersetzung in abstrakte Kennzahlen Flexibilität und Veränderbarkeit einbüßen. Das in dieser Arbeit entwickelte CDN zeigt auf, wie neuartige digitale Entwurfsmethoden aussehen können, und wie damit der Entwurfsprozess beeinflusst werden kann.

Die hier beschriebenen Objekte mit eingeschriebenem Verhalten werden in der Informatik auch als Agenten bezeichnet, und nachfolgend wurde für die Aktanten im CDN die gleiche Bezeichnung gewählt. Jeder dieser Agenten kann Informationen aus seiner Umwelt aufnehmen, dazu gehören auch Informationen über andere Agenten und die direkte Kommunikation mit ihnen. Jeder Agent beeinflusst wiederum die Umwelt, und damit die ihm zukünftig zukommende Information. Der Zustand der Welt wird als Menge von Punkten referenziert, auf die sich wiederum die einzelnen Agenten in ihrer Geometrie beziehen. Werden diese Punkte verschoben, verändern sich auch die Agenten. Eine ähnliche Darstellung hat bereits Oosterhuis mit seiner SAT bemüht; das CDN ergänzt dieses Modell nun um ein wichtiges Element: den Vektor. Jeder Agent beeinflusst die, auf die er sich bezieht. Durch Ziele die er verfolgt (Formänderung, statische Optimierung, u.ä.) will er sich selbst, und somit die mit ihm verbundenen Punkte, ändern. Folglich betrifft jede Änderung umgehend alle anderen mit dem Punkt in Verbindung stehenden Agenten. Eine Art den Einfluss, den der Agent auf den Punkt ausübt, darzustellen ist der Vektor. Auf diese Weise kann jeder mit dem Punkt verbundene Agent den Punkt beeinflussen, wobei die Summe aller Vektoren als Resultat die tatsächliche Veränderung des Punktes (vergleichbar mit dem Zusammensetzen von Zielen) verursacht. Demnach ergibt sich ein Entwurfsmodell, das als kontinuierlicher Prozess der Zustandsänderung gesehen werden kann.

> The fourth advantage I see in the word »design" (in addition to its modesty, its attention to detail and the semiotic skills it always carries with it), is that it is never a process that begins from scratch: to design is always to *redesign*. There is always something that exists first as a given, as an issue, as a problem. Design is a task that follows to make that something more lively, more commercial, more usable, more user's friendly, more acceptable, more sustainable, and so on, depending on the various constraints to which the project has to answer. In other words, there is always something *remedial* in design. This is the advantage of the »not only … but also" feature although I criticized it above. This split is a weakness to be sure (…) but it is also an immense advantage when compared to the idea of creation. To design is never to create *ex nihilo*. (Latour B. , 2008, S. 4)

Das beschriebene Modell deckt sich mit den Ansichten Latours, der davon ausgeht, dass Entwerfen nie das Erschaffen aus dem Nichts, sondern immer das Umgestalten von Bestehendem ist. Das Verständnis des Entwurfsprozesses als andauernde Änderung wirkt sich unmittelbar auf den Inhalt des Entwerfens aus: Es werden keine absoluten Werte entworfen, sondern die relativen Veränderungen zur Ausgangssituation. Während in gängigen Verfahren der Architekt (oder das Entwurfskollektiv) Parameterwerte aus dem

Vergleich verschiedener Situationen ableitet, und dann Zahlen, je nach Zielsetzung, größer oder kleiner gestaltet um neue passendere Situationen herbeizuführen, nutzt das CDN die Abweichung als neuen Input. Entwurfsaktanten gestalten Bauteile nicht mehr größer oder kleiner, sie beeinflussen viel mehr ob er größer oder kleiner wird. Aus dem zyklischen Entwerfen wird so eine kontinuierliche Entwicklung.

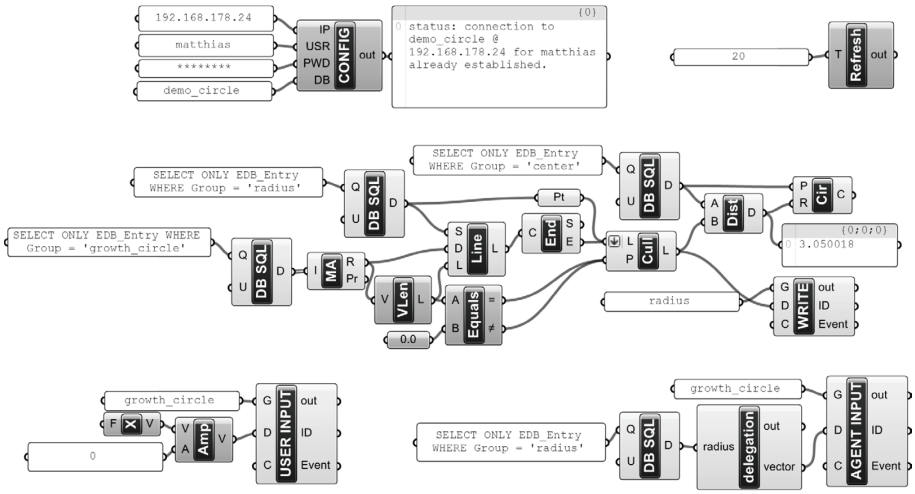

Abb. 5: Dynamische Kreisdefinition

Man kann diesen Vorgang leicht auf das Kreisbeispiel übertragen (der Kreis ist Stellvertreter für alle geschlossenen Bauteilblöcke) und damit auch die Funktionsweise der Datenbankerweiterung darlegen. Im ersten Schritt wird eine Verbindung zur Datenbank `demo_circle` hergestellt, und das globale Aktualisierungsintervall der Lesebefehle auf zwanzig Sekunden gesetzt. Die restlichen (in der Abbildung die unteren drei) Netzwerke umfassen dann die direkte Eingabe durch den menschlichen Benutzer, das an einen Agenten delegierte formale Verhalten und den Kreisagenten selbst. In diesem Beispiel wird ein Mittelpunkt `center` in die Datenbank geschrieben, und zusätzlich ein Punkt `radius` wodurch der Kreis definiert ist. In der Startphase, solange weder Benutzer noch andere Aktanten den Kreis beeinflussen, ergibt sich für ihn folgendes Handlungsschema: (Absatz!) Radiuspunkt und Mittelpunkt werden ausgelesen, der Radius als Abstand der beiden berechnet, der Kreis wird im Arbeitsbereich von Rhinoceros konstruiert und der Radiuspunkt wird wieder (unverändert) in die Datenbank geschrieben.
Will nun ein Benutzer, dass der Kreis größer oder kleiner wird, gibt er die Veränderung als Zahlenwert an (positive Werte führen zu Wachstum, und negative Werte zum Schrump-

fen). Aus der Eingabe wird ein Vektor konstruiert und in der Datenbank gesichert. Beim nächsten Durchlauf nimmt der Kreisagent diesen Einfluss in sein Verhalten mit auf. Er liest die Summe aller Vektoren (also die Einflüsse) auf den Radiuspunkt und gibt ihnen durch das Verschieben des Radiuspunktes nach. Dadurch ergibt sich ein veränderter Radiuspunkt zum Speichern und ein veränderter Radius, der zu einem veränderten Kreis führt. Darüber hinaus können Beeinflussungen des Radius auch an andere Agenten delegiert werden. Im Beispiel soll der Agent (rechts unten) den Radius immer in Richtung der nächsten ganzen Zahl drängen[47]. Ist der Radius knapp unter einem passenden Wert gibt er einen Vektor aus, der den Kreis wachsen lassen würde und umgekehrt. Der Kreisagent setzt die Einflüsse ungewichtet zusammen und reagiert darauf, so dass es sein kann, dass sich der menschliche und der nicht-menschliche Einfluss aufschaukeln, abschwächen oder gegenseitig aufheben. Der Kreis kann theoretisch auf unendlich viele Vektoren reagieren, ob und wie diese zum Tragen kommen und sich zusammenfügen kann über zusätzlichen Code bestimmt werden.

> Reactive agents do not plan their actions, but react directly to sensory input using a simple rule based if-then system. When an input signal exceeds a certain threshold, the agent is triggered into a predefined action. A reactive agent can respond to changes in its environment in near real-time, making it very useful in environments that require quick responses from an agent. A drawback of the reactive agent architecture is the inability of reactive agents to do long-term planning. (Schermer, 2007, S. 26)

Der Kreisagent im Beispiel ist ein rein reaktiver Agent, der die Umwelt, die er über seine Sensoren (Input, Lesekomponente) aufnimmt, mit seinen Effektoren (Output, Schreibkomponente) nach immer gleichen Regeln beeinflusst. Er besitzt keine Fähigkeit sein Verhalten zu planen und besitzt auch kein inneres Abbild seiner Umwelt. Das simple Verhaltensmuster erleichtert auch sein Modellieren, und ist auf diese Weise besonders geeignet um einfache Aufgaben wie die im Beispiel zu übernehmen. Dort erhält er einen Wert, verarbeitet ihn nach einem gewissen Skript, und gibt einen neuen Wert aus. Ähnlich funktioniert das Verhalten des Benutzers, der neben einem Wert aus der Datenbank auch die grafische Darstellung zur Entscheidungsfindung heranziehen kann. Über seine Wahrnehmung und die, für ihn wichtigen, angewandten Regeln kann der Benutzer über den ihm zugänglichen Effektor des Eingabefeldes den gleichen Einfluss setzen wie der Agent. Sie unterscheiden sich nur in der Art der Entscheidungsfindung, und womöglich

47 Ein Verhalten, dass gerade beim Verwenden von Normgrößen von Interesse ist.

in ihrer Ausformung, sind aber für die »Umwelt«, in Form der Datenbank, in ihren Eingabemöglichkeiten gleichgestellt. Ob die Vektoren im Beispiel von einem Menschen oder von einem Agenten stammen ist für den Kreisagenten unwichtig. Das Forschungsfeld rund um die Künstliche Intelligenz hat, im Vergleich zum reaktiven Agenten auch weiter entwickelte Architekturen hervorgebracht:

> Deliberative agents are a more sophisticated class of agents. This type of agent is able to reason about its behavior and adapt to changes in its environment using an internal reasoning model. In addition to such a model, a world model can also be employed. The agent can use this world model to increase its chances of coming up with a successful plan in unforeseen situations. (…) The most successful deliberative agent architecture model is the BDI model and many deliberative agents are based on this model of agency. (Schermer, 2007, S. 26f)

Im Kern geht es bei den Entwicklungen darum, dass Agenten die Fähigkeit zugeschrieben bekommen ihre Entscheidungen zu planen. Sie können zwischen verschiedenen Verhaltensweisen jene auswählen, die ihrer Erfahrung nach die höchsten Erfolgschancen hervorbringen. Mit BDI-Agenten könnten auch vielfältige Planungsaufgaben an die, so weiterentwickelten Bauteilreferenzen delegiert werden. Im Gegensatz zu den reaktiven Agenten haben sie den Nachteil, dass sie erstens aufwendiger zu erstellen sind und zweitens mehr Rechenleistung benötigen. Schermer führt in seinem Text auch weiter aus, wie die Interaktion mehrerer Agenten in einem Multiagenten-System zu einem komplexen Verhalten führen können und so Methoden bei denen nur ein Agent genutzt wird übertreffen. Im Collaborative Design Network wurden analog dazu mehrere Agenten unterschiedlicher Komplexität entwickelt und eingesetzt. Wie ein solcher Entwurfsagent aufgebaut sein kann veranschaulicht das Beispiel eines Trägeragenten:

Abb. 6: Trägeragent des CDN Experimentes

Der Grundaufbau ähnelt dem des Kreisagenten: Über die Lesekomponenten erhält der Trägeragent Information und verarbeitet sie nach gewissen Regeln, um abschließend über die Schreibkomponenten wieder Informationen (Änderungen) auszugeben. Wo sich das Einlesen von Daten nur in der Anzahl der Datensätze unterscheidet, ist die Differenz im

Verhalten etwas größer: Der Träger soll als plane Fläche (eine Vorgabe aus der späteren Fertigung) mehrere Fassadenpunkte, die nicht auf einer Ebene sind, verbinden. Es gäbe nun die Möglichkeit, dass der Agent einfach eine Ebene zwischen dem obersten und dem untersten Punkt spannt und die dazwischen liegenden Punkte darauf projiziert. Manche Punkte könnten bei dieser Lösung jedoch große Abstände zur Ebene aufweisen. Ein anderes Verfahren ist die Methode der kleinsten Quadrate[48]. Das mathematische Standardverfahren zur Ausgleichsrechnung hilft dabei in eine Menge von Punkten eine passende Gerade einzuschreiben.

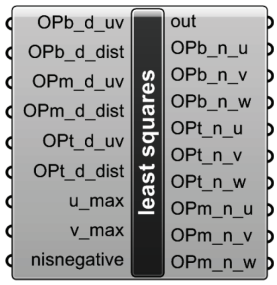

Abb. 7: Trägerteilverhalten "Methode der kleinsten Quadrate"

Dem Träger wurde sie als Verhaltenskomponente »method of the least squares« zu den anderen Knoten hinzugefügt. Im Projekt bezieht sie sich auf drei Punkte, über die zuvor mit Hilfe von Einflussvektoren Sollzustände ermittelt wurden. Die drei werden für die Berechnung einer Geradenfunktion herangezogen, aus der sich innerhalb eines gewissen Toleranzbereiches, die tatsächlich umsetzbaren Sollkoordinaten ergeben. Mit dieser Methode wird, in Anbetracht der auf ihn wirkenden Einflüsse, eine optimale Lage und Dimension für den Träger im Raum ermittelt (der Quelltext zeigt, wie das Verfahren Schritt für Schritt im Inneren der Komponente durchexerziert wird und wie ein Bauteil auf Verhalten verweisen kann). Wären nicht noch andere Agenten mit dem Trägeragenten verbunden, wäre er demnach so isoliert wie der Kreisagent zuvor, und sein Verhalten wäre damit bereits bestimmt. Da er aber noch mit anderen in Kontakt steht, musste sein Programm um zusätzliche Funktionen erweitert werden.

```
01   double vb = OPb_d_uv.Y;
02  double vm = OPm_d_uv.Y;
```

48 Die Methode wurde ungefähr Zeitgleich am Anfang des neunzehnten Jahrhunderts von Carl Friedrich Gauß und Adrien-Marie Legendre entwickelt.

```
03  double vt = OPt_d_uv.Y;
04  double va = (vb + vm + vt) / 3;
05  double ub = OPb_d_uv.X;
06  double um = OPm_d_uv.X;
07  double ut = OPt_d_uv.X;
08  double ua = (ub + um + ut) / 3;
09  double b = ((vb - va) * (ub - ua) + (vm - va) * (um - ua) + (vt - va) *
    (ut - ua)) / ((vb - va) * (vb - va) + (vm - va) * (vm - va) + (vt - va)
    * (vt - va));
10  double a = ua - b * va;
11  double tol = 0.2;
12  if((a + b * 0) < (0 + tol)){a = 0 + tol;}
13  if((a + b * u_max) < (0 + tol)){b = (tol + tol - v_max) / u_max;}
14  if((a + b * 0) > (v_max - tol)){ a = v_max - tol;}
15  if((a + b * u_max) > (v_max - tol)){b = (v_max - tol - a) / u_max;}
16  double vbn = a + b * 0; // absolute bottom
17  double vtn = a + b * u_max; // absolute top
18  OPb_n_u = 0;
19  OPb_n_v = vbn;
20  OPt_n_u = u_max;
21  OPt_n_v = vtn;
22  OPm_n_u = um > (u_max - tol) ? (u_max - tol) : (um < (0 + tol) ? (0 +
    tol) : um);
23  OPm_n_v = vm > (v_max - tol) ? (v_max - tol) : (vm < (0 + tol) ? (0 +
    tol) : vm);
24  if(nisnegative == true)
25  {
26  OPb_n_w = OPb_d_dist > ((-1) * u_max / 10) ? ((-1) * u_max / 10) :
    OPb_d_dist;
27  OPt_n_w = OPt_d_dist > ((-1) * u_max / 10) ? ((-1) * u_max / 10) :
    OPt_d_dist;
28  OPm_n_w = OPm_d_dist > ((-1) * u_max / 10) ? ((-1) * u_max / 10) :
    OPm_d_dist;
29  } else {
30  OPb_n_w = OPb_d_dist < (u_max / 10) ? (u_max / 10) : OPb_d_dist;
31  OPt_n_w = OPt_d_dist < (u_max / 10) ? (u_max / 10) : OPt_d_dist;
32  OPm_n_w = OPm_d_dist < (u_max / 10) ? (u_max / 10) : OPm_d_dist;
33  }
```

Einer der Umstände besteht darin, dass der Trägeragent mit den mit ihm verbundenen Trägern Anschlusspunkte teilt, von denen er sich nicht lösen kann. Die verbundenen Bauteile müssen dementsprechend ihre Eingriffe (oder Zustandsveränderungen) koordinieren. Wie zuvor beim Kreisagenten setzt sich die tatsächliche Änderung aus allen Einflüssen zusammen, die auf verschiedene Art koordiniert werden können. Ein anderer zu berücksichtigender Aspekt ist die Lageabhängigkeit des Verhaltens. Je nachdem, ob der Träger frei auf einer Ebene positioniert ist oder entlang einer Kante verläuft, kann er sich den Einflussvektoren beugen. Für diesen Zweck war es notwendig, dass der Trägeragent über eine ausgeprägte Selbstwahrnehmung verfügt, um das für seine Position geeignete Skript auswählen zu können.

Die Beschreibung des Trägeragenten soll zeigen, wie Bauteile als Entwurfsaktanten gestaltet werden können. Neben den materiellen können auf die gleiche Art und Weise formale Handlungen an ihn delegiert werden. Als eigenständiger Agent kann der digitale Ent-

wurfsaktant gleich dem menschlichen Entwerfer auf das Modell Einfluss ausüben. In einem Netzwerk von mehreren zusammengeschlossenen Rechnern, in dem auf den einen Agenten laufen und auf den anderen Menschen den Entwurf bearbeiten, ist die Unterscheidung zwischen Menschen und Nicht-Menschen nicht länger relevant. Noch dazu sind Agenten, wenn sie dafür programmiert wurden, auf mehrere Entwürfe anwendbar und manche Komponenten von ihnen als Teilverhalten austauschbar.

DAS ENTWERFEN MIT AKTANTEN: DAS KOLLEKTIV

Im Experiment CDN wird das Entwerfen einer einfachen Fassade in Kooperation mit dafür entwickelten digitalen Aktanten getestet. Bereits im Aufbau des Systems wird berücksichtigt, dass die teilnehmenden Aktanten nicht gleicher Art sein müssen. So unterscheiden sich die dreieckigen und rechteckigen Fassadenplatten sowohl in Geometrie als auch Verhalten maßgeblich von den durch fünf Punkte bestimmten Trägern. Die einen erfüllen hauptsächlich eine formale Funktion, während die anderen die (materialistische) Aufgabe ein Tragwerk zu simulieren, bedienen. Das Ziel des Experiments liegt nicht im Programmieren eines möglichst intelligenten Aktanten, sondern eines interagierenden dynamischen Kollektivs.
Die Fassade setzt sich aus drei Ebenen zusammen, die miteinander in Verbindung stehen. Die planen Wandflächen der Innenseite orientieren sich nach einem vom Benutzer eingegebenen Umriss und einer anfangs definierten Höhe. Beide Angaben können, so wie alle anderen Parameter, währen des Entwurfsprozesses manipuliert werden. Im Entwurfsbeispiel sollte das Verhalten einer nicht rechtwinkeligen Innenecke und einer nicht rechtwinkeligen Außenecke untersucht werden. Es wäre genauso möglich als Umfang auf ein geschlossenes, regelmäßiges oder unregelmäßiges Vieleck zurückzugreifen. Über zwei Anschlusspunkte ist jeder Trägeraktant mit der Wandfläche verbunden. Die Punkte orientieren sich an der oberen und unteren Stockwerkskante. Darüber hinaus können sie sich, in gewissen Toleranzbereichen, frei entlang dieser Achse verteilen.

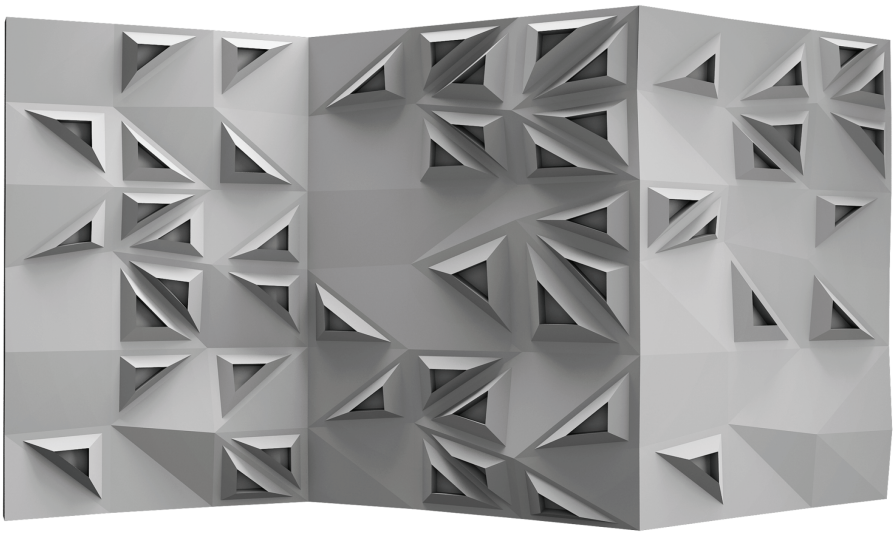

Abb. 8: 19. Generation bei Regel 1-2/1-3, Dimension 18x6, Form F

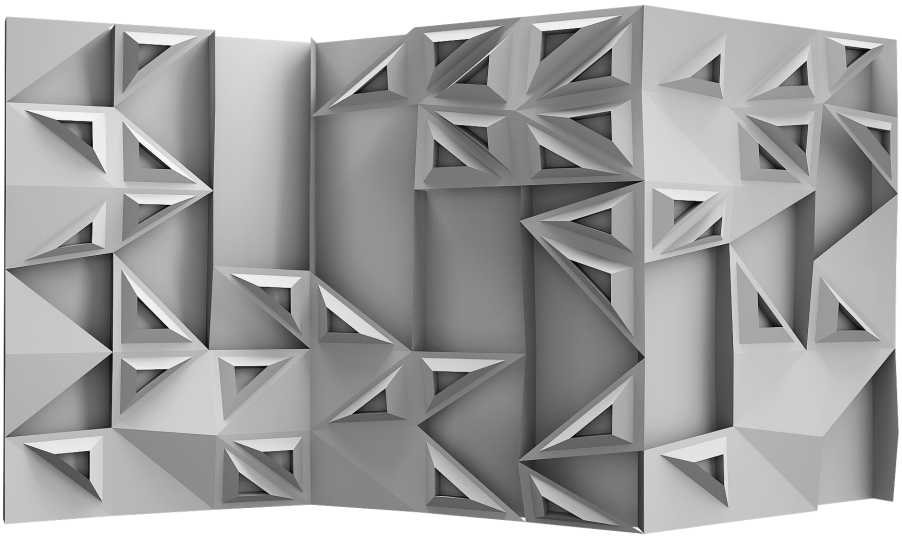

Abb. 9: dreischichtiger Aufbau aus Wand-, Träger- und Fassadenelementen

Befindet sich der Träger an einer Kante, kann er sich im Beispiel nicht bewegen. Eine weitere Vorgabe war, dass der plane Träger immer normal auf die Wandebene stehen muss. Er hat im Beispiel einzig eine minimale Bautiefe die er nicht unterschreiten kann, ist aber in seiner maximalen Tiefe nicht weiter begrenzt. Die diversen Fassadenelemente sind über den Träger mit der Wand verbunden, wie aus der Abbildung ersichtlich ist. Im gezeigten Beispiel interagieren im Kollektive 54 bis 108 Fassadenagenten (je nach Bau-

form), 30 Trägeragenten und 9 Wandagenten. Gesamt sind das bis zu 147 digitale Bauteilaktanten, die als autonome Programme existieren und interagieren.
Bevor die Interaktion dieses Kollektivs mit dem Benutzer analysiert wird, folgen noch Beschreibungen zu seinem »eigenständigen« Verhalten, allen voran die Fassadenagenten-Träger Beziehung. Der Trägeraktant, wie bereits beschrieben, ist eine zentrale Figur im Gestaltungsprozess des Beispiels. Er ist Mittler zwischen den Vorgaben der Wand und den Einflüssen, die von der Fassadenseite kommen.

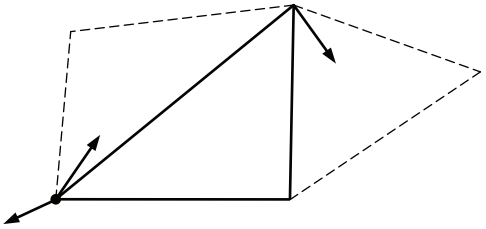

Abb. 10: Einflüsse an den Eckpunkten

Die einzelnen Fassadenelemente wurden mit den Eigenschaften von zellulären Automaten entwickelt. Ob der Agent wachsen oder schrumpfen will, richtet sich nach den Bewegungen seiner Nachbarn. Er liest nicht direkt die Eigenschaften der anderen aus, sondern nimmt die Vektoren wahr, die an seinen Eckpunkten angreifen. Aus der Anzahl der Vektoren die ihn schrumpfen oder wachsen ließen, ermittelt er nach einer eingangs definierten Regel seine Zielsetzung. Dem Fassadenagenten steht nur eine gewisse »Intensität« zur Verfügung, er kann nur einen Vektor mit einer vorgegebenen Amplitude auf einen Eckpunkt wirken lassen. Er berechnet, unter Berücksichtigung der bereits vorhandenen Vektoren, an welchem Eckpunkt er seine Zielsetzung am besten erfüllen kann, und setzt dort seinen Eingriff an[49].

> For the supporters of algorithmic architecture, a new field of exploration has opened up, one that aims at a better understanding and exploration of computation's genuine process and their potential for the production of architecture. They are fascinated by

49 Ist sein Ziel zu schrumpfen, wird ein Vektor formuliert, der als Ausgangspunkt einen der Eckpunkte und als Ziel den Mittelpunkt des Dreiecks hat. Will er wachsen, zeigt der Vektor in die umgekehrte Richtung und teils Normal auf die Ebene des Dreiecks. In beiden Fällen wird für den Umwelteingriff jener der drei Punkte ausgewählt, dessen Resultierende – also die Summe des eigenen und der Nachbareingriffe – die kleinste Richtungsdifferenz zum eigenen Vektor vorweist.

> how complex architecture emerge from simple rules and models like the Turing Machine and cellular automaton, in the moment of their computation, as exemplified here by some of the work produced by Studio Rocker in spring 2004. The promise and limits of such explorations are diverse. The use of genetic algorithms has been traditionally driven by the desire to generate novel forms yet without a deeper consideration of their physical actualization and performance. Most, if not all, explorations into algorithmic process- and form-generation generally neglect the structural and material integrity of architecture. (Rocker I. , 2007, S. 24)

Die Ebene der Fassadenagenten isoliert betrachtet unterscheidet sich nicht von den gängigen formalen Ansätzen in der Disziplin des Digitalen Entwerfens. Die strukturelle Integrität von der Rocker schreibt, erhält das Experiment zum einen durch das Verständnis des digitalen Aktanten als Referenz auf ein »Objekt« und zum anderen durch die Interaktion verschiedener Bauteilsysteme. Sehr gut ist dies am Verhalten und am Einfluss des Trägeragenten zu beobachten.

Abb. 11: Lage- und Formänderung des Trägers

Wie im vorigen Abschnitt beschrieben, nutzt er dazu die Methode der kleinsten Quadrate. Aus der Summe seiner Einflüsse leitet er seine Form- und Lageveränderung ab. In wie weit er auf die Bewegungen der Fassadenelemente eingehen kann und wie diese tatsächlich ihre Geometrie verändern beeinflusst auch wie sie sich im nächsten Durchgang verhalten. Kleine Winkeländerungen können dazu führen, dass ihr Vektor an einem anderen Punkt angreift und damit eine Kettenreaktion in der Formänderung verursacht, die in einfachen Systemen Zellulärer Automaten nicht vorkommt. Dadurch, dass die Träger die Anschlusspunkte der Dreiecke verschieben, entstehen für diese und deren potenzielle Vektoren neue Richtungen. Die veränderten Richtungen können wiederum Einfluss darauf haben, an welchen Punkten sie ansetzen und damit indirekt mitbestimmen, ob ein Nachbardreieck schrumpft oder wächst. Diese einfache Konfiguration lässt bereits ein komplexes Verhalten mehrerer Aktanten in verschiedenen Bezugssystemen beobachten.

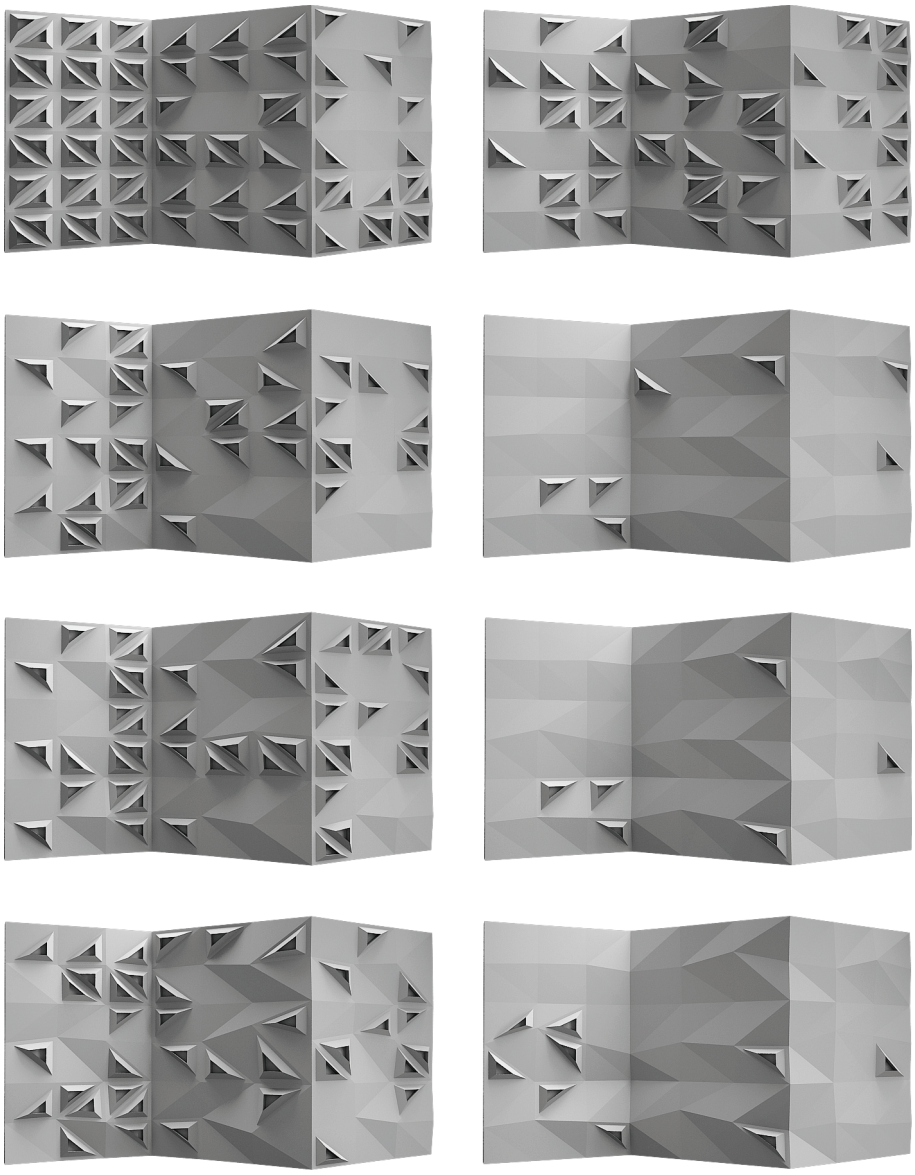

Abb. 12: Generationen 0,5,10,15 der Regeln 1-2/1-3 (li.) und 1-2/3-4 (re.)

Die beiden gegenübergestellten Serien sollen dazu veranschaulichen welche Auswirkungen das Ändern der Regeln hat, nach denen die Fassadenplattenaktanten ihre Zielsetzung bestimmen. Jene Regeln, die zu einem häufigeren Wechseln des Ausdehnungsmodus der Aktanten führen, verursachen zwar eine stärkere Durchmischung, haben aber geringere gestaltverändernde Auswirkungen. Im Gegensatz dazu produzieren jene Regeln, die zu statischen Mustern und damit einer geringeren Fluktuation in der Vektorpositionierung

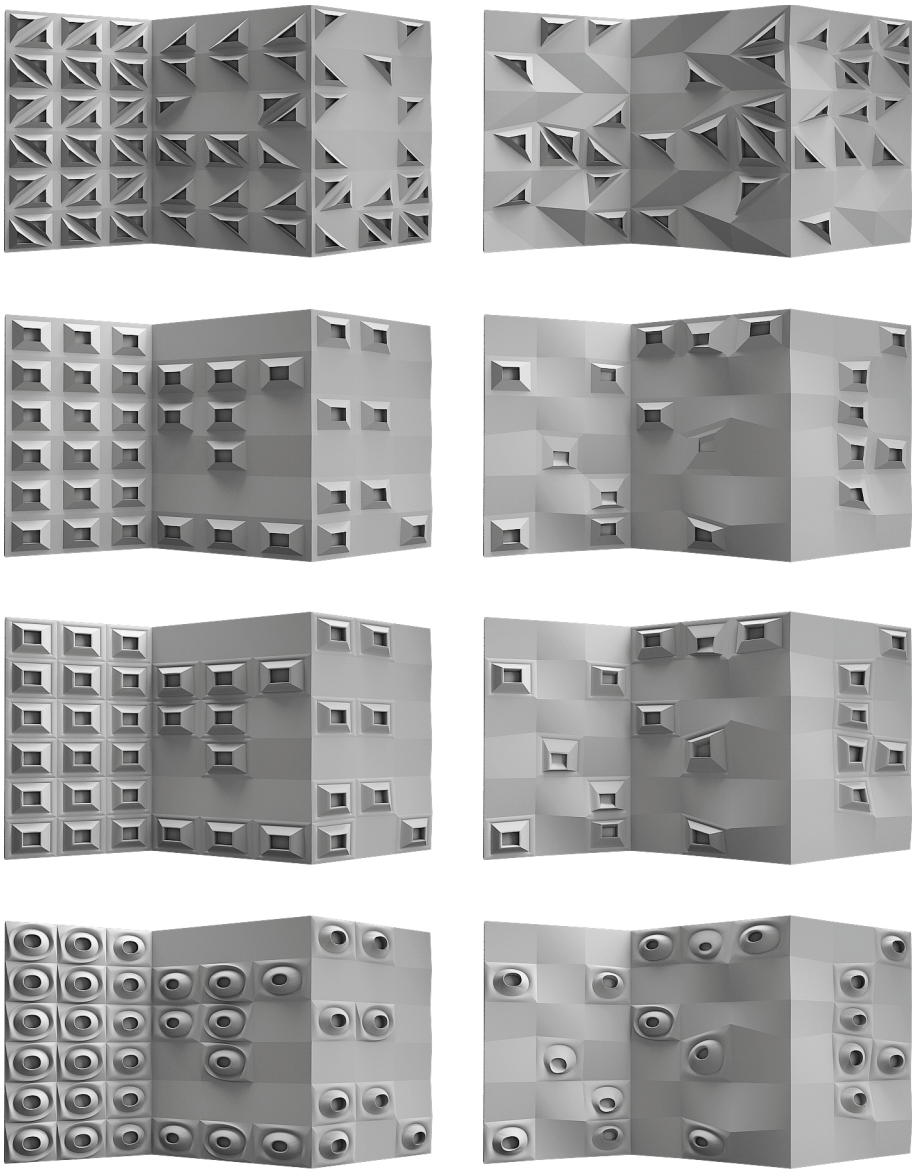

Abb. 13: Fassadentypen F, G, H und I in den Generationen 0 und 20

führen, stärkere Änderungen der Geometrie. Darüber hinaus lässt sich die Auswirkung der Durchmischung der verschiedenen Modi erkennen: Die Serie, in der sich nur wenige ausdehnende Aktanten befinden, ändert sich weniger als die Serie mit vielen sich ausdehnenden Agenten. Ob die Fassadenelemente nur formale Regeln ausführen, für eine bauphysikalische Optimierung zuständig sind, materielle Eigenschaften der Objekte vertreten oder alles auf einmal miteinkalkulieren liegt dabei in der Hand des Entwerfers.

Welche Form die Fassadenelemente annehmen ist genauso frei definierbar wie die Regeln nach denen sie handeln. Die Geometrievarianten können über die gezeigten Beispiele in Detaillierung und Komplexität weit hinausgehen, Fensterelemente sind dabei genauso möglich wie alle anderen parametrisierten Bauteile. Welche Elemente Verwendung finden ist von den diversen Entwurfsaktanten bestimmbar. Entweder bestimmt der menschliche Entwerfer welche Agenten er einsetzen will oder er lässt andere Agenten diese Entscheidung treffen. Zwei benachbarte Dreiecke könnten so unter gewissen Umständen zu einem Viereck werden oder umgekehrt. Der Unterschied des CDN zu anderen Einsätzen von Automaten liegt aber nicht darin, sondern im Fokus auf die Interaktion der Agenten des Entwurfsnetzwerks. Die bislang gezeigten Entwurfsreihen fanden ohne direkte Teilnahme eines menschlichen Entwerfers statt. Sie sind darum als »Leerlauf« ein Sonderfall der Anwendung dieser Technik und nicht Regelfall, vergleichbar mit deterministischen Optimierungsmethoden oder Planungsautomaten.
In den folgenden Abbildungen wird das eigentliche Einsatzszenario des CDN gezeigt. Ein oder mehrere Benutzer können auf das Modell einwirken und den Entwurfsprozess beeinflussen. Neben der Möglichkeit Ausgangsparameter zu verändern, kann direkt eingegriffen werden. Im gezeigten Beispiel wird manchen Fassadenelementen ein anderes Verhalten zugeschrieben (sie sollen sich an jedem Eckpunkt unabhängig von dem auf sie einwirkenden Vektoren immer ausdehnen), oder ihr Verhalten modifiziert (manche werden so eingestellt, dass sie sich immer ausdehnen oder schrumpfen). Die Typen der Fassadenelemente variieren (dreieckige und viereckige), und werden auch im Prozess nach Belieben getauscht. Am augenscheinlichsten sind die direkten Änderungen, wie das Ziehen an manchen Punkten (durch das Setzen von zusätzlichen Vektoren). Alle diese Arten der Einflussnahme werden vom Modell nicht direkt umgesetzt, sondern nach den vorhandenen Beziehungen übersetzt. Zieht man an einem Punkt, beeinflusst man direkt das Verhalten der Träger und damit das Verhalten der anschließenden Fassadenelemente. Zieht man öfter mit der gleichen Intensität, bekommt man zu jedem Zeitpunkt und für jeden Punkt ein anderes Resultat.

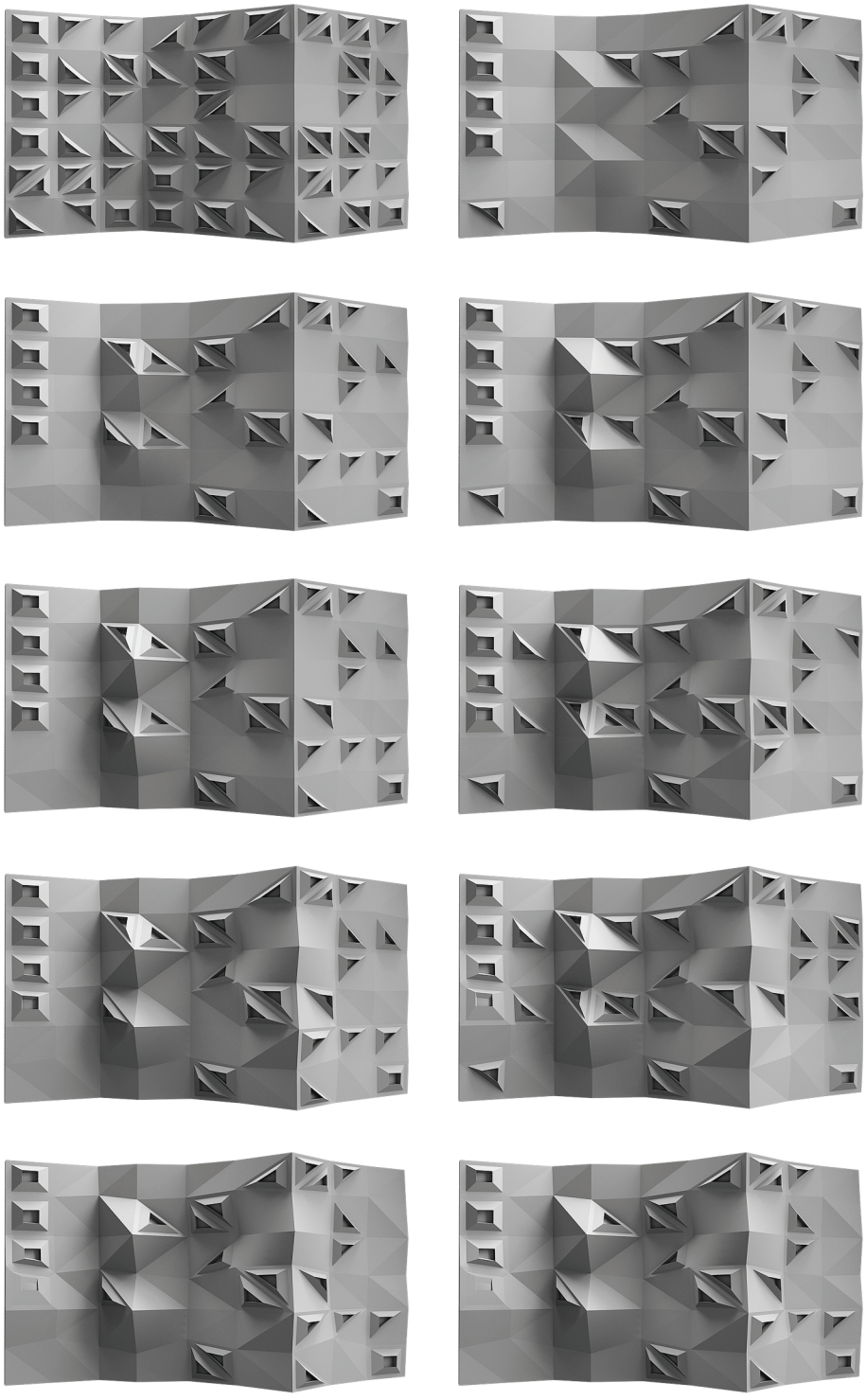

Abb. 14: Versuchsreihe mit Benutzerinteraktion, Generationen 1 bis 10

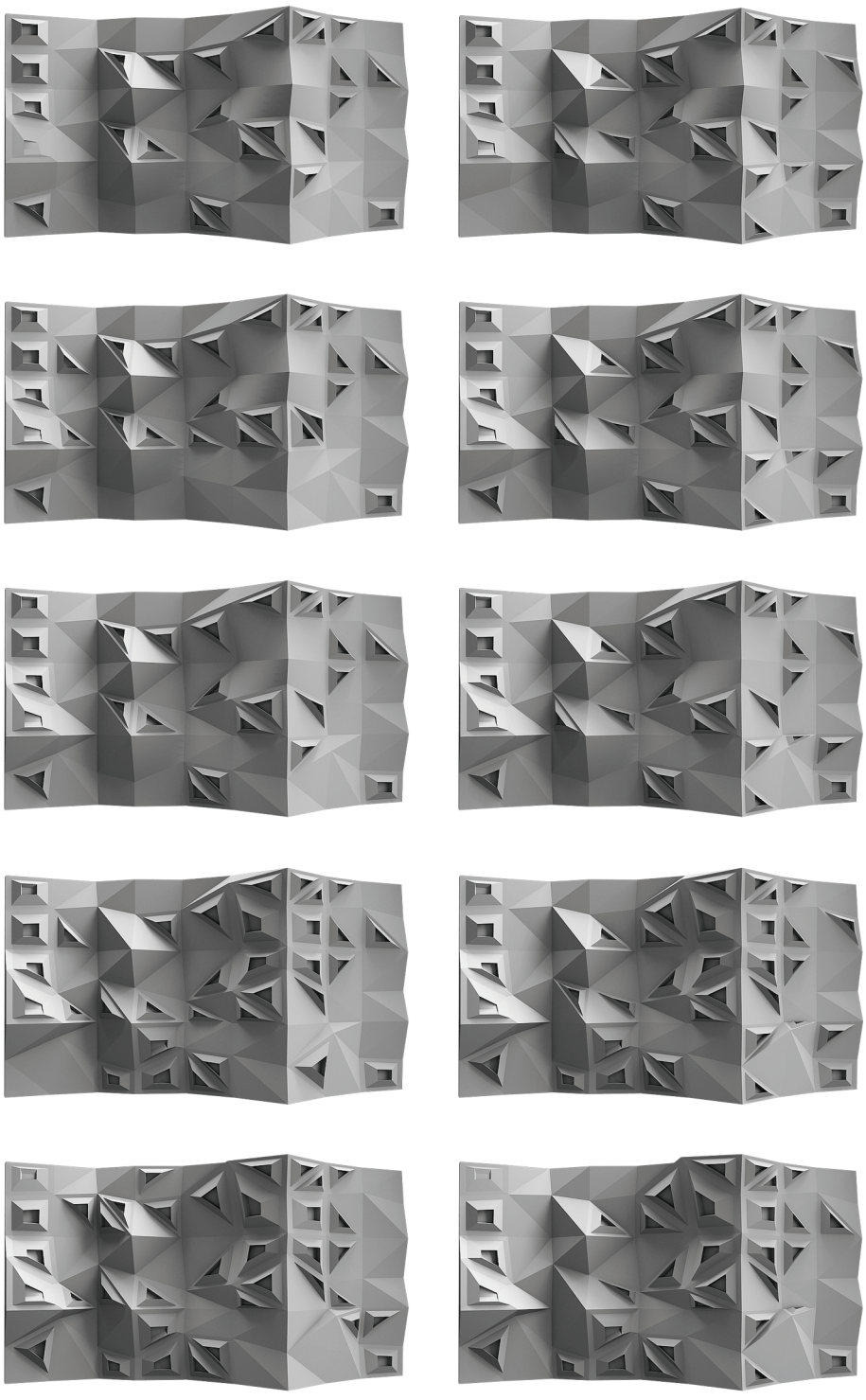

Abb. 15: Versuchsreihe mit Benutzerinteraktion, Generationen 11 bis 20

Der große Unterschied von CDN zu bestehenden Ansätzen ist die prozessuale Konstruktion. Während die Variante meistens als Abweichung vom Archetypus verstanden wird, ist bei der CDN die Variante eine Abweichung der vorangegangenen Variante und das Entwerfen damit iterativ. Da das Ermittlungsverfahren nicht deterministischer Natur ist, können die Varianten nicht gleichzeitig ermittelt werden. Währen in anderen Ansätzen Ergebnisse entworfen werden, entwirft man im CDN immer nur die Abweichung von der vorangegangenen Version, also immer nur einen Zwischenschritt.
Ob eine Iteration in Kennwerte übersetzt besser oder schlechter ist als die vorangegangene ist nicht prognostizierbar. Dadurch kann es sein, dass genau jenes Zwischenergebnis, das man umsetzen möchte, in Kennzahlen ausgedrückt um vieles schlechter geeignet ist als das nur eine Iteration vorher oder nachher. Die Ergebnisse des Experimentes belegen, dass mit derzeit zur Verfügung stehenden technischen Mitteln eine an die ANT angelehnte Methodik im digitalen Entwerfen möglich ist. Bei dem vorgestellten CDN handelt es sich nicht um einen Entwurfsautomaten, denn an jeder Stelle des Projekts ist der Eingriff eines oder mehrerer Planer und Architekten möglich. Sowohl im Erstellen der Quelltexte und dem damit verbundenen Delegieren der Arbeit wie auch in der Auswahl der teilnehmenden Aktanten kann den damit beauftragten Aktanten im Entwerfen eine besondere Rolle zukommen. Darüber hinaus ist es in jedem Iterationsschritt möglich, von außerhalb in den Prozess einzugreifen. Es können zusätzliche Vektoren eingefügt und bestehende Parameter überschrieben oder gelöscht werden. Dabei liegt ein Schwerpunkt des CDN in der Offenheit des Prozesses. Wer wann wie lange mitgestaltet ist im Vorhinein nicht festgelegt. Es scheint schwer das CDN als Weiterentwicklung nur eines der präsentierten Ansätze zu verstehen, haben doch sowohl Trummer, Lynn, Rocker und Oosterhuis bereits Ideen verfolgt, die das CDN wieder aufnimmt. Den wichtigsten Einfluss auf die Gestaltung des CDN Modells hat aber die Actor Network Theory von Bruno Latour, deren praktische Umsetzung mit dem Beispiel dargestellt wurde. Derzeit fehlt noch eine Technik um die Verbindungen der Agenten wie in den Projekten rund um MACOSPOL darstellen zu können. Eine Karte, die die Interaktion der digitalen Aktanten aufzeigt, und ähnlich wie die Netzwerke in Grasshopper bearbeitet werden kann, ist als neue Referenzebene für das Entwerfen möglich.

6. SCHLUSSWORT

Entwerfen als Prozess

Eine Veränderung der Selbstwahrnehmung des Entwerfers, wie sie bereits im Gange ist, weg von der Idee des Alleinentwerfers hin zum Bild des Mitentwerfers in einem Entwurfskollektiv, bereitet die Basis für neue Entwurfsmethoden. Begünstigt durch die einfache Vernetzung der Entwurfsteilnehmer im Digitalen Entwerfen hat die neue Art der Zusammenarbeit zum Formulieren von Theorien wie der SAT inspiriert. Darüber hinaus kann durch die Erkenntnis, dass mit jeder Übersetzung eine Transformation einhergeht, die bidirektionale Interaktion die unidirektionale Anweisung ablösen. Lineare und zyklische Modelle, welche die Grundlage für das derzeitige digitale Entwerfen bilden, können durch parallel ablaufende Prozesse ersetzt werden, wodurch in weiterer Folge die Aktanten und ihre Vernetzungen in den Mittelpunkt der Betrachtung rücken. Die ANT sprengt dabei Hierarchien und egalisiert die Unterschiede zwischen nicht-menschlichen Aktanten und menschlichen Aktanten, wodurch sich der kreative Spielraum zusätzlich vergrößert.
Vor allem müssen wir die Vorstellung hinter uns lassen, Entwerfen produziere Resultate. Entwerfen ist ein Prozess, und diesen an einem Zeitpunkt zu beenden eine beliebige Entscheidung. Gerade die ANT stellt klar, dass Entwerfen immer ein Weiterentwickeln darstellt, und kein Erschaffen aus dem Nichts. Die Ziele der involvierten Aktanten setzen sich zusammen und beeinflussen sich gegenseitig. Sie erreichen keinen stabilen oder womöglich gar idealen Gleichgewichtszustand, sondern können sich diesem nur annähern. Der Zustand der als geeigneter »Abschluss« des Entwurfsprozesses eingestuft wird, ist auf keinen Fall ein zwingendes Endergebnis. Rückkopplungen und andere systemimmanente Effekte lassen keine zukünftigen Zustände prognostizieren und stehen einer deterministischen Anwendung im Weg. Der Entwurf kann ebenso wenig auf ein Ende wie auf eine formale Darstellung reduziert werden, die direkt in Materie übersetzt werden könnte – er ist kein isoliertes Gedankengebäude, sondern ein sich stetig entwickelndes, aus vielen Zielen zusammengesetztes und viele Ziele verfolgendes, unscharfes Gemeinschaftswerk.
Die ANT zeigt, dass über Werkzeuge kollektiv Einfluss ausgeübt wird, und dass Werkzeuge zum Erstellen von Werkzeugen eine Alternative zu Kurven zum Erstellen von Kurven

darstellen. Nicht nur, dass sich damit zahlreiche neue Gestaltungsmöglichkeiten eröffnen und Architekten wie Planer mehr Chancen haben sich in den Prozess einzubringen, es wird damit auch das Diktat der Parameter und die damit verbundene Optimierungsdrift erfolgreich umgangen. Bisher haben isoliert betrachtete Kennwerte, wie städtebauliche oder bauphysikalische Parameter, und die damit verbundenen Techniken oft massiven Einfluss auf den Gestaltungsprozess ausgeübt und dadurch diverse Hierarchiebestrebungen provoziert, die über das Ausüben von Macht die Stellung des Architekten als zentraler Entwerfer bewahren wollten. Dazu bietet die ANT mit dem Gewinnen von Einfluss durch das Delegieren von Arbeit eine Alternative. Das Delegieren lässt dabei jedem einzelnen Aktanten im Kollektiv eine neue, wichtigere Rolle zukommen. Auf die gleiche Weise wie die Ziele dieses Textes durch ein Zurückgreifen auf diverse Zitate abdrifteten und sich im Laufe des Schreibens weiterentwickelten, entwickelt sich der Entwurf im darin beschriebenen Experiment. Buch und Fassade reflektieren die Idee von kollaborativen Entwurfsnetzwerken, in denen sich Form und Materie aus dem Zentrum der sie verbindenden Referenzkette herauskristallisieren.

Als eine von wenigen hat Ingeborg Rocker bereits die Frage ins Spiel gebracht, wie die ANT auf den architektonischen Entwurf angewendet werden könnte. Neben einem Kontextualisieren von verschiedenen Werten und Methoden im Digitalen Entwerfen, sucht man auch neue Gedankenmodelle um diese wissenschaftstheoretischen Erkenntnisse in den Entwurf übersetzen zu können. Die in diesem Werk behandelte Referenzkette des digitalen Entwerfens und die vorgestellte Technik der Multiagenten Systeme sind ein Beitrag zu eben diesem gerade aufblühenden Diskurs. Dass der Einsatz der ANT im Digitalen Entwerfen möglich ist, zeigt das vorgestellte Experiment. Eine Weiterentwicklung von objektorientierten Methoden lässt darin ein kollaboratives Entwurfsnetzwerk mit Planern interagieren und »gemeinsam« eine Fassade gestalten. Die Grenzen, die zu einer Einteilung der Urheberschaft in Architekt und Entwurfssoftware (und den damit verbundenen Entwicklern) dient, verschwimmen dabei zusehens.

Die präsentierten Digitalen Agenten funktionieren symmetrisch in beide Richtungen der Referenzkette, und ihre Übersetzungen ermöglichen neue Einblicke in den Entwurf. Selbst Agenten zu erstellen wird auf diese Weise fundamental für das Entwerfen, während die Kenntnis über die eingesetzten »Werkzeuge« zur Grundlage für die Nachvollziehbarkeit der Einflüsse auf den Entwurfsprozess wird. Der Architekt kann sich selbst dadurch wieder näher ins Zentrum des Entwurfsnetzwerkes rücken. Das Systemmodell des Fassadenbeispiels beansprucht für sich keines Falls eine fertige und direkt anwendbare Entwurfslösung zu sein, es veranschaulicht aber sehr gut in welche Richtung sich das Di-

gitale Entwerfen entwickeln kann und wie wichtig tief greifende Kenntnisse im Adaptieren standardisierter Softwarelösungen sind.
Um für einen praxisnahen Einsatz besser anwendbar zu werden, ist als Weiterentwicklung des CDN in nächster Zeit zum einen eine Loslösung von Grasshopper und Rhino als Ausführungsumgebung geplant und zum anderen das Umsetzen einer Darstellung der Agentennetze analog zu MACOSPOL. Darüber hinaus soll die Einbindung von Techniken wie objektrelationalen Datenbanken (z.B. PostgreSQL) und JADE[50] sowie eine Anbindung an bestehende BIM Lösungen für das CDN evaluiert und gegebenenfalls umgesetzt werden.
Die Anwendung der ANT auf das Entwerfen bietet ein unermessliches Potenzial für Weiterentwicklungen und neue Erkenntnisse in der Architektur. Mit der Chance die sich hier ergibt, kann sie einen nächsten Meilenstein in ihrer Entwicklung erreichen. Der hier vorliegende Text stellt einen kleinen Beitrag in diesem Prozess dar.

50 JADE, Java Agent Development Framework

VERZEICHNISSE

LITERATURVERZEICHNIS

Beaucé, P., & Cache, B. (2004). Towards a Non-Standard Mode of Production. In F. O. Architects, *Phylogenesis: foa's ark* (S. 390-407). Barcelona, New York: Actar Publishers.

Beierle, C., & Kern-Isberner, G. (2006). *Methoden Wissensbasierter Systeme: Grundlagen - Algorithmen - Anwendungen, 3. erweiterte Auflage.* Wiesbaden: Friedr. Vieweg & Sohn Verlagsgesellschaf, GWV Fachverlage.

Benbow, D. W., Elshennawy, A. K., & Walker, H. F. (2003). *The certified quality technician handbook.* Milwaukee: ASQ Quality Press.

Binding, G. (1993). *Baubetrieb im Mittelalter.* Darmstadt: Wissenschaftliche Buchgesellschaft.

Boreham, P., Parker, R., Thompson, P., & Hall, R. (2008). *New Technology @ Work.* New York: Routledge.

Bronstein, I. N., Semendjajew, K. A., Musiol, G., & Muehlig, H. (2008). *Taschenbuch der Mathematik.* Frankfurt am Main: Wissenschaftlicher Vertrag Harri Deutsch.

Burnham, J. u. (1971). *The structure of art.* G. Braziller.

Cache, B. ([1983] 1995). *Earth moves. the Furnishing of Territories.* Cambridge: MIT.

Cache, B. (2003). Towards a Fully Associative Architecture. In B. Kolarevic, *Architecture in the digital age: design and manufacturing* (S. 203-209). New York: Spon Press.

Cantor, G., Dedekind, R., & Fraenkel, A. A. (1932). *Gesammelte Abhandlungen Mathematischen und Philosophischen Inhalts.* Berlin: Verlag von Julius Springer.

Carpo, M. (2009). Aufstieg und Fall der identischen Reproduzierbarkeit. Zu Leon Battista Albertis unzeitgemäßer Entdeckung digitaler Technologie in der Frührenaissance. In D. Gethmann, & S. Hauser, *Kulturtechnik Entwerfen. Praktiken, Konzepte und Medien in Architektur und Design Science.* (S. 49-64). Bielefeld: Transcript Verlag.

Carpo, M. (2010). Das Digitale, "Mouvance" und das Ende der Geschichte. *GAM - Graz Architektur Magazin* , S. 16-29.

Cramer, T. (1998). *Waz hilfet âne sinne kunst? Lyrik im 13. Jahrhundert.* Berlin: Erich Schmidt Verlag.

Daenzer, W. F. (1976). *Systems Engineering.* Zürich: Verlag Industrielle Organisation.

DeBruyn, G., & Reuter, W. (2011). *Das Wissen der Architektur. Vom Geschlossenen Kreis zum Offenen Netz.* Bielefeld: Transcript Verlag.

DeLanda, M. (2004). Deleuze and the Use of the Genetic Algorithm in Architecture. In F. O. Architects, *Phylogenesis: foa's ark* (S. 520-529). Barcelona, New York: Actar Publishers.

DeLanda, M. (1998). Meshowrks, Hierarchies and Interfaces. In J. Beckmann, *The Virtual Dimension: Architecture, Representation, and Crash Culture* (S. 274-285). New York: Princeton Architectural Press.

DeLanda, M. (1998). Meshworks, Hierarchies and Interfaces. In J. Beckmann, *The Virtual Dimension: Architecture, Representation, and Crash Culture* (S. 274-285). New York: Princeton Architectural Press.

DeLanda, M. (1994). Virtual Environments and the Emergence of Synthetic Reason. In M. Dery, *Flame Wars: The Discourse of Cyberculture* (S. 263-286). Durham: Duke University Press.

Deleuze, G. u. (1977). *Rhizom.* Berlin: Merve Verlag.

Detlor, B. (2004). *Towards Knowledge Portals. From Human Issues to Intelligent Agents.* Dordrecht: Kluwer Academic Publishers.

Eastman, C. (2006). New Opportunities for IT Research in Construction. *Intelligent Computing in Engineering and Architecture: 13th EG-ICE Workshop 2006* (S. 163-174). Ascona, Switzerland: Springer.

Etzioni, A. (2009). *Die aktive Gesellschaft. Eine Theorie gesellschaftlicher und politischer Prozesse. 2. Auflage.* Wiesbaden: VS Verlag für Sozialwissenschaften | GWV Fachverlag.

Evers, B., & Thoenes, C. (2003). *Architectural Theory. From the Renaissance to the Present.* Köln: Taschen.

Feng, H. (2009). Quantum Architecture. A non-deterministic and Interactive Computational Design System. *Second International Conference on Critical Digital: Who Cares (?)*, (S. 183-191). Harvard.

Forbes, L. H., & Ahmed, S. M. (2011). *Modern Construction: Productive and Lean Practices.* Boca Raton: Taylor and Francis Group.

Hays, K. M., Ingraham, C., & Kennedy, A. (2003). Computer-Animismus. In G. DeBruyn, & S. Trüby (Hrsg.), *architektur_theorie.doc. texte seit 1960* (S. 148-151). Basel: Birkhäuser.

Heidegger, M. ([1962] 2007). *Die Technik und die Kehre.* Stuttgart: Klett-Cotta.

Heuer, A., Saake, G., & Sattler, K.-U. (2008). *Datenbanken. Konzepte und Sprachen.* Heidelberg: Redline.

Höfler, C. (2005). Form und Feld. In H. Bredekamp, M. Pratschke, M. Bruhn, & G. Werner, *Digitale Form* (S. 64-73). Berlin: Akademie Verlag.

Holland, M. (1989). *From Industry to Alchemy: Burgmaster, a Machine Tool Company (Original: When the Machine Stopped: A Cautionary Tale from Industrial America).* Frederick: Beard Books.

Hütt, M.-T. (2001). *Datenanalyse in der Biologie.* Berling Heidelberg: Springer-Verlag.

Jain, K. C., & Chitale, A. K. (2010). *Textbook of Production Engineering.* New Delhi: PHI Learning Private Limited.

James, J. (2006). *Digital Intermediates for Film and Video.* Burlington: Focal Press (Elsevier).

Joedicke, J., Rietkötter, K. H., Schmöller, K. H., Schulte, H.-O., & Seibert, H.-U. (1975). *Arbeitsberichte zur Planungsmethodik 9.* Stuttgart: Karl Krämer Verlag.

Juergens, E., Deissenboeck, F., Hummel, B., & Wagner, S. (2009). Do Code Clones Matter? *ICSE 2009. IEEE 31st International Conference on Software Engineering* (S. 485-495). Vancouver: IEEE.

Klügl, F. (2001). *Multiagentensimulation. Konzepte, Werkzeuge, Anwendungen.* München: Addison-Wesley Verlag.

Kowol, G. (2009). *Projektive Geometrie und Cayley-Klein Geometrien der Ebene.* Basel Boston Berlin: Birkhäuser Verlag.

Latour, B. (2008). A Cautious Prometheus ? A Few Steps Toward a Philosophy of Design (With Special Attention to Peter Sloterdijk). *Proceedings of the 2008 Annual International Conference of the Design History Society – Falmouth, 3-6 September 2009* (S. 2-10). Boca Raton: Universal Publishers.

Latour, B. (2010). An Attempt at a “Compositionist Manifesto”. *New Literary History; 41, 3* , 471-490.

Latour, B. (2002). *Die Hoffnung der Pandora.* Frankfurt am Main: Suhrkamp Verlag.

Latour, B. (2006). Drawing Things Together: Die Macht der unveränderlichen Elemente. In A. Belliger, & D. J. Krieger (Hrsg.), *ANThology* (S. 259-307). Bielefeld: transcript Verlag.

Latour, B. (1996). On Actor Network Theory. A Few Clarifications. *Soziale Welt 47* , S. 369-381.

Latour, B. u. (2008). Die Analyse der Architektur nach der Actor-Network-Theory. In U. u. Staub, *Explorations in architecture: teaching, design, research* (S. 80-89). Basel: Birkhäuser.

Latour, B. (2008). *Wir sind nie modern gewesen. Versuch einer symmetrischen Anthropologie.* Frankfurt am Main: Suhrkamp Verlag.

Leach, N., & DeLanda, M. (Juli/August 2009). The Limits of Urban Simulation. An Interview with Manuel DeLanda. *Architectural Design Vol 79, No 4* , S. 50-55.

Lynn, G. (1999). *Animate Form.* New York: Princeton Architectural Press.

Lynn, G. (Februar 2005). *Greg Lynn: On Calculus in Architecture.* Abgerufen am 12. Februar 2011 von Video File: http://www.ted.com/talks/lang/eng/greg_lynn_on_organic_design.html

Matrosov, A., Rodionov, E., Harley, D., & Malcho, J. (2011). *Stuxnet Under the Microscope, Revision 1.31.* Bratislava: ESET.

McCormack, J. P., & Cagan, J. (2002). Supporting Designer's Hierarchies through Parametric Shape Recognition. *Environment and Planning B, Volume 29* , S. 913-931.

Menning, G. (2008). *Werkzeugbau in der Kunststoffverarbeitung. Bauarten, Herstellung, Betrieb.* München: Carl Hanser Verlag.

Meyer, B. (2004). The power of abstraction, reuse and simplicity: an object-oriented library for event-driven design. In O. Owe, S. Krogdahl, & T. Lyche, *From Object-Orientation to Formal Methods: Essays in Memory of Ole-Johan Dahl (Lecture Notes)* (S. 236-271). Berlin Heidelberg: Springer-Verlag.

Moore, G. E. (19. April 1965). Cramming more components onto integrated circuits. *Electronics, Volume 38, Number 8* , S. 4-7.

Nerdinger, W. (2002). *Perspektiven der Kunst. Von der Karolingerzeit bis zur Gegenwart. 2. Auflage.* München: Oldenbourg.

Noble, D. F. (1995). *Progress Without People. New Technology, Unemployment, and the Message of Resistance.* Toronto: Between The Lines.

Núnez, R. E., & Lakoff, G. (1998). What did Weierstrass Really Define? *Mathematical Cognition Volume 4 Issue 2* , S. 85-101.

Oberkampf, V. (1976). *Systemtheoretische Grundlagen einer Theorie der Unternehmensplanung.* Berlin: Duncker & Humblot.

Olexa, R. (August 2001). The father of the Second Industrial Revolution (Interview with John Parsons). *Manufacturing Engineering Vol. 127 No. 2* .

Oosterhuis, K. (Oktober 2004). A New Kind of Building. *EAAE News Sheet* , S. 18-31.

Oosterhuis, K. (2003). *Hyperbodies: toward an e-motive architecture.* Basel: Birkhäuser.

Oosterhuis, K. (2006). Swarm Architecture II. In K. Oosterhuis, & L. Freireiss, *GameSetandMatch II. On Computer Games, Advanced Geometries, and Digital Technologies* (S. 14-28). Rotterdam: Episode Publishers.

Palladio, A. ([1570] 2002). *The Four Books on Architecture.* (R. Tavernor, & R. Schofield, Übers.) Cambridge: MIT Press.

Patten, T. (1988). *Systemic Text Generation as Problem Solving.* Cambridge: Cambridge University Press.

Pfahlmann, H. (1974). *Die Industrielle Revolution.* Würzburg: Verlag Ploetz.

Raja, V., & Fernandes, K. J. (2008). *Reverse Engineering: An Industrial Perspective.* London: Springer-Verlag.

Randal, J. C. (2005). *Osama: the making of a terrorist.* London, New York: I. B. Tauris & Co.

Rembold, R. W. (2007). *Einstieg in CATIA V5: objektorientiert konstruieren in Übungen und Beispielen.* München: Carl Hanser Verlag.

Rocker, I. M. (2008). Versioning: Architecture as a Series? *First International Conference on Critical Digital, What Matter (s)?* (S. 157-170). Cambride: Harvard.

Rocker, I. (2002). Versioning: Evolving Architectures - Dissolving Identities 'Nothing is as persistent as Change'. *Architectural Design Vol 72 No 5 Sept/Oct* , 10-17.

Rocker, I. (2007). When Code matters. *Architectural Design No 182 Vol 76 No 4* , S. 16-25.

Saam, N. J. (2002). *Prinzipale, Agenten und Macht.* Tübingen: J.C.B. Mohr (Paul Siebeck).

Schermer, B. W. (2007). *Software Agents, Surveillance, and the Right to Privacy: A Legislative Framework for Agent-Enabled Surveillance.* Leiden: Leiden University Press.

Schumacher, P. S. (2011). *The Autopoiesis of Architecture, Volume I: A New Framework for Architecture.* Chichester, West Sussex: John Wiley & Sons.

Stiny, G. (1980). Introduction to Shape and Shape Grammars. *Environment and Planning B, Volume 7* , S. 343-351.

Stiny, G., & Mitchell, W. J. (1978). The Palladian Grammar. *Environment and Planning B Volume 5* , S. 5-18.

Trummer, P. (5. November 2008). Vom Typus zur Population. *Arch+ 189* , S. 46-51.

Vanderfeesten, E., & DeVries, B. (2006). Confection for the Masses in a Parametric Design of a Modular Favela. In K. Oosterhuis, & L. Freireiss (Ed.), *GameSetandMatch II : On Computer Games, Advanced Geometries, and Digital Technologies* (S. 306-393). Rotterdam: Episode Publishers.

Weber, M. (1984). *Soziologische Grundbegriffe.* Tübingen: J. C. B. Mohr (Paul Siebeck).

Weckherlin, G. (2009). Architekturmaschinen und wissenschaftliches Entwerfen. Entwurfspraktiken und -theorien Ende der sechziger Jahre. In D. Gethmann, & S. Hauser (Hg.), *Kulturtechnik Entwerfen. Praktiken, Konzepte und Medien in Architektur und Design Science* (S. 203-226). Bielefeld: transcript Verlag.

Whitehead, E. J., & Goland, Y. Y. (2002). WebDAV: A network protocol for remote collaborative authoring. *ECSCW '99: proceedings of the Sixth European Conference on Computer Supported Cooperative Work, 12-16 September 1999, Copenhagen, Denmark* (S. 291-310). Copenhagen: Kluwer Academic.

Wirth, N. (2008). *Grundlagen und Techniken des Compilerbaus.* München: Oldenbourg Wissenschaftsverlag.

Woodbury, R. S. (1964). *History of the milling machine: a study in technical development.* Boston: M. I. T. Press.

Yaneva, A. (2009). *MAC. Mapping Architectural Controversies.* Abgerufen am 4. Februar 2011 von http://www.msa.ac.uk/mac/why.html

Yaneva, A. (December 2005). Scaling Up and Down: Extraction Trials in Architectural Design . *Social Studies of Science* , 867-894.

Ziesche, P. (2005). *Nebenläufige & Verteilte Programmierung.* Herdecke Bochum: W3L.

ABBILDUNGSVERZEICHNIS

Alle verwendeten Abbildungen wurden vom Verfasser dieser Arbeit selbst erstellt.

ABKÜRZUNGSVERZEICHNIS

ANT ..... Actor Network Theory
BDI ..... Belief Desire Intention
BIM ..... Building Information Modeling
CAD ..... Computer Aided Design
CAM ..... Computer Aided Manufactoring
CATIA ..... Computer Aided Three-Dimensional Interactive Application
CDN ..... Collaborative Design Network
CNC ..... Computer Numerical Control
JADE ..... Java Agent Development Framework
MACOSPOL ..... Mapping Controversies on Science for Politics
MAS ..... Multi Agent System

NC.....Numerical Control
NURBS..... Non-Uniform Rational Basis Spline
P2P ..... Peer to Peer
SAT.....Swarm Architecture
USAF ..... United States Air Force

INDEX

ANHANG A

Die Funktionen von GHEloqueraDB.gha V 2.1

LIZENZ UND KOMPONENTEN

Das Plug-in wurde in C# .NET 3.5 geschrieben, ist als Open Source lizenziert, und benötigt zum Ausführen folgende Komponenten/Bibliotheken: Eloquera.Client, Eloquera.Common, GH_IO, Grasshopper, Rhino_DotNET, RhinoCommon, System, System.Core, System.Data, System.Drawing, System.Windows.Forms, System.Xml. Folgende Software muss installiert sein: Windows Vista, Rhinoceros 4.0, Grasshopper 0.80001 und der passende Eloquera Server in der Version 3.0.

```
34  /// GHEloqueraDB.gha Version 2.1
35  ///
36  /// This program is free software: you can redistribute it and/or
37  /// modify it under the terms of the GNU General Public License as
38  /// published by the Free Software Foundation, either version 3 of
39  /// the License, or(at your option) any later version.
40  /// This program is distributed in the hope that it will be useful,
41  /// but WITHOUT ANY WARRANTY/ without even the implied warranty of
42  /// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
43  /// GNU General Public License for more details.
44  /// You should have received a copy of the GNU General Public License
45  /// along with this program.
46  /// If not, see http://www.gnu.org/licenses/
47  /// This simple but useful Grasshopper Plugin is brought to you by:
48  ///
49  ///         Matthias Standfest
50  ///         TU Graz
51  ///         Rechbauerstraße 12
52  ///         8010 Graz
53  ///         Austria
54  ///         http://tugraz.at
55  ///
56  /// Created 21/11/2010 by Matthias Standfest
57  ///
58  /// Copyright ©2012 Matthias Standfest
59  ///
60  /// Description:
61  /// This Plugin provides an Eloquera Database extension for
62  /// Grasshopper / Rhinoceros.
```

EINTRAG SPEICHERN

Eingaben für diese Komponente sind der Name der Gruppe zu der der Datensatz hinzugefügt werden soll, die Liste der Daten und die Angabe ob mit dem Schreibvorgang die alten Werte der schreibenden Komponente oder alle Werte der Gruppe überschrieben werden sollen. Die Komponente liefert ausgabeseitig eine Statusmeldung, die Identifikationsnummer der Instanz und ein Ereignis bei erfolgreichem Schreibvorgang.

```
63   using System;
64   using System.Collections.Generic;
65   using System.Text;
66   using System.Collections;
67   using System.Drawing;
68   using System.IO;
69   using System.Linq;
70   using System.Xml.Serialization;
71   using System.Windows.Forms;
72   using Grasshopper.Kernel;
73   using Grasshopper.Kernel.Types;
74   using RMA.OpenNURBS;
75   using Eloquera.Client;
76
77   namespace GHEloqueraDB
78   {
79     public class writeDB : GH_Component
80     {
81       public writeDB()
82         : base("Write Data", "WRITE", "writes data in an eloquera database", "Extra",
     "CCTRL Database") //tooltip
83       {
84       }
85       protected override void RegisterInputParams(GH_Component.GH_InputParamManager
     pManager) //inputs
86       {
87         pManager.Register_StringParam("group", "G", "group data belongs to");
88         pManager.Register_GenericParam("data", "D", "data from database",
     GH_ParamAccess.list);
89         pManager.Register_BooleanParam("delete", "C", "false: delete all other data of
     group; true: delete all other data from ID", true);
90       }
91       protected override void RegisterOutputParams(GH_Component.GH_OutputParamManager
     pManager) //outputs
92       {
93         pManager.Register_StringParam("Status", "out", "whats going on");
94         pManager.Register_StringParam("ID", "ID", "ID of this write element");
95         pManager.Register_BooleanParam("Event", "Event", "return 'true' after
     writing");
96       }
97       protected override void SolveInstance(IGH_DataAccess DA) //function
98       {
99         //definitions
100        List<Object> input = new List<Object>();
101        string groupname = null;
102        bool overwrite = false;
103        // input control
```

```
104       if (((DA.GetData<string>(0, ref groupname) && DA.GetDataList<Object>(1, input))
    && input.Count > 0) && groupname.Length > 0)
105       {
106         DA.GetData<bool>(2, ref overwrite);
107         string statustext = null;
108         // prep DB
109         EDB_Entry placebo = new EDB_Entry();
110         placebo.Group = groupname;
111         placebo.Data = null;
112         placebo.Type = input[0].GetType().ToString();
113         placebo.OriginID = this.InstanceGuid.ToString();
114         GH_Eloquera.db.RegisterType(typeof(EDB_Entry));
115         // check if file is alreadey saved, so ID can be generated
116         string myid = null;
117         var ghDoc = this.OnPingDocument();
118         if (ghDoc != null)
119           myid = ghDoc.FileNameProxy;
120         if (myid.Contains("unnamed") == false)
121         {
122           EDB_Entry item = new EDB_Entry(); // create item 2 write
123           item.Group = groupname;
124           item.Type = input[0].GetType().ToString();
125           item.OriginID = myid + "-" + this.InstanceGuid.ToString();
126           switch (item.Type) // determine data type
127           {
128             case "Grasshopper.Kernel.Types.GH_Boolean":
129               List<bool> inputBool = new List<bool>();
130               DA.GetDataList<bool>(1, inputBool);
131               item.DataBool = inputBool;
132               break;
133             case "Grasshopper.Kernel.Types.GH_Integer":
134               List<int> inputInt = new List<int>();
135               DA.GetDataList<int>(1, inputInt);
136               item.DataInt = inputInt;
137               break;
138             case "Grasshopper.Kernel.Types.GH_Number":
139               List<double> inputDouble = new List<double>();
140               DA.GetDataList<double>(1, inputDouble);
141               item.DataDouble = inputDouble;
142               break;
143             case "Grasshopper.Kernel.Types.GH_String":
144               List<string> inputString = new List<string>();
145               DA.GetDataList<string>(1, inputString);
146               item.DataString = inputString;
147               break;
148             default:
149               item.Data = input;
150               break;
151           }
152           if (!overwrite) // if all content of the group should be reset
153           {
154             string querydel = "SELECT ONLY EDB_Entry WHERE Group = '" + groupname +
    "'";
155             var resdels = GH_Eloquera.db.ExecuteQuery(querydel);
156             foreach (var resdel in resdels)
157             {
158               GH_Eloquera.db.Delete(resdel);
159             }
160             resdels = null;
161           }
162           else // or just the one of the item
```

```
163           {
164             string querytext = "SELECT ONLY EDB_Entry WHERE OriginID ='" +
    item.OriginID + "'";
165             var results = GH_Eloquera.db.ExecuteQuery(querytext);
166             foreach (var result in results)
167             {
168               GH_Eloquera.db.Delete(result);
169             }
170             results = null;
171           }
172           GH_Eloquera.db.Store(item); // data storing
173           // status message
174           statustext = "[" + input.Count.ToString() + "] items of type [" +
    input[0].GetType().ToString() + "] from component [" + item.OriginID + "]
    successfully written in database ["+ GH_Eloquera.dbName +"] group [" + groupname +
    "].";
175           DA.SetData(1, item.OriginID); // output of component set
176           DA.SetData(2, true);
177           item = null;
178           placebo = null;
179           input = null;
180           groupname = null;
181           myid = null;
182         }else{
183           statustext = "error: cannot add correct ID without filename. please save
    file first.";
184         }
185         DA.SetData(0, statustext); // output statustext
186         statustext = null;
187       }
188     }
189     public override Guid ComponentGuid
190     {
191       get
192       {
193         return new Guid("{32cf6078-4ab3-487d-b6eb-3e0e79606417}");
194       }
195     }
196     protected override Bitmap Internal_Icon_24x24
197     {
198       get
199       {
200         return GHEloqueraDB.Properties.Resources.D_write;
201       }
202     }
203     public override GH_Exposure Exposure
204     {
205       get
206       {
207         return GH_Exposure.primary;
208       }
209     }
210   }
211 }
```

EINTRAG AUSLESEN

Eingaben für diese Komponente sind die ID des Ursprungsobjektes des Datensatzes und die Angabe ob der Lesevorgang in einem global definierten Zeitintervall wiederholt werden soll. Die Komponente liefert die ausgelesenen Datensätze.

```
212  using System;
213  using System.Collections.Generic;
214  using System.Text;
215  using System.Collections;
216  using System.Drawing;
217  using System.IO;
218  using System.Linq;
219  using System.Windows.Forms;
220  using Grasshopper.Kernel;
221  using Grasshopper.Kernel.Types;
222  using RMA.OpenNURBS;
223  using Eloquera.Client;
224
225  namespace GHEloqueraDB
226  {
227    public class readDB : GH_Component
228    {
229      public readDB()
230        : base("Read Data", "READ", "reads data from an eloquera database", "Extra",
     "CCTRL Database") // tooltip
231      {
232      }
233      // inputs
234      protected override void RegisterInputParams(GH_Component.GH_InputParamManager
     pManager)
235      {
236        pManager.Register_StringParam("Origin ID", "ID", "id of origin object to read
     from");
237        pManager.Register_BooleanParam("automatic update", "U", "set auto update to
     global refreshrate", false);
238      }
239      // outputs
240      protected override void RegisterOutputParams(GH_Component.GH_OutputParamManager
     pManager)
241      {
242        pManager.Register_GenericParam("data", "D", "data from database");
243      }
244      // autoupdate
245      public void readthis(object sender, EventArgs e)
246      {
247        ExpireSolution(true);
248      }
249      // definitions
250      string myoriginID = null;
251      bool autoupdate = true;
252      // function
253      protected override void SolveInstance(IGH_DataAccess DA)
254      {
255        if ((DA.GetData<string>(0, ref myoriginID) && (DA.GetData<bool>(1, ref
     autoupdate))) && myoriginID.Length > 0)
256        {
```

```
257         string status = null;
258         // get data
259         string querytext = "SELECT ONLY EDB_Entry WHERE OriginID = '" + myoriginID +
    "'";
260         var result = GH_Eloquera.db.ExecuteQuery(querytext);
261         foreach (EDB_Entry item in result)
262         {
263         // write correct data type to output
264         switch (item.Type)
265         {
266           case "Grasshopper.Kernel.Types.GH_Boolean":
267             DA.SetDataList(0, item.DataBool);
268             break;
269           case "Grasshopper.Kernel.Types.GH_Integer":
270             DA.SetDataList(0, item.DataInt);
271             break;
272           case "Grasshopper.Kernel.Types.GH_Number":
273             DA.SetDataList(0, item.DataDouble);
274             break;
275           case "Grasshopper.Kernel.Types.GH_String":
276             DA.SetDataList(0, item.DataString);
277             break;
278           default:
279             DA.SetDataList(0, item.Data);
280             break;
281         }
282         }
283         // definitions
284         result = null;
285         querytext = null;
286         // eventhandler for autoupdate
287         EventHandler reader = new EventHandler(readthis);
288         GH_Eloquera.removeEvent(reader);
289         if (autoupdate == true)
290         {
291           GH_Eloquera.addEvent(reader);
292           status = "status: active (interval " +
    GH_Eloquera.Timer.Interval.ToString() + "msec)";
293         }
294         else
295         {
296           status = "status: inactive";
297         }
298         status = null;
299       }
300     }
301     public override Guid ComponentGuid
302     {
303       get
304       {
305         return new Guid("{02c8f6ec-63ad-41d4-836c-74c3425bee47}");
306       }
307     }
308     protected override Bitmap Internal_Icon_24x24
309     {
310       get
311       {
312         return GHEloqueraDB.Properties.Resources.D_read;
313       }
314     }
315     public override GH_Exposure Exposure
```

```
316      {
317        get
318        {
319          return GH_Exposure.primary;
320        }
321      }
322    }
323  }
```

SQL ABFRAGE

Eingaben für diese Komponente sind die SQL Abfrage als String und die Angabe ob der Lesevorgang in einem global definierten Zeitintervall wiederholt werden soll. Die Komponente liefert die ausgelesenen Datensätze.

```
324  using System;
325  using System.Collections.Generic;
326  using System.Text;
327  using System.Collections;
328  using System.Drawing;
329  using System.IO;
330  using System.Linq;
331  using System.Windows.Forms;
332  using Grasshopper.Kernel;
333  using Grasshopper.Kernel.Types;
334  using RMA.OpenNURBS;
335  using Eloquera.Client;
336
337  namespace GHEloqueraDB
338  {
339    public class sqlqueryDB : GH_Component
340    {
341      public sqlqueryDB()
342        : base("SQL Query", "DB SQL", "sql query for an eloquera database", "Extra",
     "CCTRL Database") //tooltip
343      {
344      }
345      // input
346      protected override void RegisterInputParams(GH_Component.GH_InputParamManager
     pManager)
347      {
348        pManager.Register_StringParam("querytext", "Q", "sql query (string)");
349        pManager.Register_BooleanParam("automatic update", "U", "set auto update to
     global refreshrate", false);
350      }
351      // definition of output
352      protected override void RegisterOutputParams(GH_Component.GH_OutputParamManager
     pManager)
353      {
354        pManager.Register_GenericParam("data", "D", "data from database");
355      }
356      // autoupdater
357      public void readthis(object sender, EventArgs e)
358      {
```

```
359       ExpireSolution(true);
360     }
361     // definitions
362     string querytext = null;
363     bool autoupdate = true;
364     string status = null;
365     // function
366     protected override void SolveInstance(IGH_DataAccess DA)
367     {
368       // input control
369       if ((DA.GetData<string>(0, ref querytext)&&(DA.GetData<bool>(1, ref
    autoupdate))) && querytext.Length > 0)
370       {
371         var result = GH_Eloquera.db.ExecuteQuery(querytext);
372         // query and correct output type
373         foreach (EDB_Entry item in result)
374         {
375           switch (item.Type)
376           {
377             case "Grasshopper.Kernel.Types.GH_Boolean":
378               DA.SetDataList(0, item.DataBool);
379               break;
380             case "Grasshopper.Kernel.Types.GH_Integer":
381               DA.SetDataList(0, item.DataInt);
382               break;
383             case "Grasshopper.Kernel.Types.GH_Number":
384               DA.SetDataList(0, item.DataDouble);
385               break;
386             case "Grasshopper.Kernel.Types.GH_String":
387               DA.SetDataList(0, item.DataString);
388               break;
389             default:
390               DA.SetDataList(0, item.Data);
391               break;
392           }
393         }
394         // nullsetting
395         result = null;
396         querytext = null;
397         // autoupdater
398         EventHandler reader = new EventHandler(readthis);
399         GH_Eloquera.removeEvent(reader);
400         if (autoupdate == true)
401         {
402           GH_Eloquera.addEvent(reader);
403           status = "status: active (interval " +
    GH_Eloquera.Timer.Interval.ToString() + "msec)";
404         }
405         else
406         {
407           status = "status: inactive";
408         }
409         // setting statusstring null too
410         status = null;
411       }
412     }
413     public override Guid ComponentGuid
414     {
415       get
416       {
417         return new Guid("{90e83caa-66ad-4e89-9288-b4cbd6652cac}");
```

```
418         }
419       }
420       protected override Bitmap Internal_Icon_24x24
421       {
422         get
423         {
424           return GHEloqueraDB.Properties.Resources.DB_sqlquery;
425         }
426       }
427       public override GH_Exposure Exposure
428       {
429         get
430         {
431           return (GH_Exposure.dropdown | GH_Exposure.secondary);
432         }
433       }
434     }
435   }
```

DATENBANK SICHERN

Eingaben für diese Komponente sind Dateiname und Pfad für die Sicherungsdatei, das Zeitintervall für einen automatischen Sicherungsvorgang und die Angabe ob automatisch gesichert werden soll. Die Komponente liefert ausgabeseitig eine Statusmeldung über Erfolg oder Misserfolg der Sicherung.

```
436   using System;
437   using System.Collections.Generic;
438   using System.Text;
439   using System.Collections;
440   using System.Drawing;
441   using System.IO;
442   using System.Linq;
443   using System.Windows.Forms;
444   using Grasshopper.Kernel;
445   using Grasshopper.Kernel.Types;
446   using RMA.OpenNURBS;
447   using Eloquera.Client;
448   using Eloquera;
449
450   namespace GHEloqueraDB
451   {
452     public class backup : GH_Component
453     {
454       public backup()
455         : base("Backup Database", "DB BAK", "saves a backup of an eloquera database",
      "Extra", "CCTRL Database") // tooltip
456       {
457       }
458       // inputnodes
459       protected override void RegisterInputParams(GH_Component.GH_InputParamManager
      pManager)
460       {
```

```
461       pManager.Register_StringParam("filename", "SRC", "filename (string) and path of
    the backupfile");
462       pManager.Register_IntegerParam("timespan", "T", "minutes (int) between
    backups");
463       pManager.Register_BooleanParam("active", "A", "activity status (bool) of
    backuptimer",false);
464     }
465     // outputnodes
466     protected override void RegisterOutputParams(GH_Component.GH_OutputParamManager
    pManager)
467     {
468       pManager.Register_StringParam("status", "out", "status");
469     }
470     // function
471     protected override void SolveInstance(IGH_DataAccess DA)
472     {
473       // definition
474       string filename = null;
475       int timespan = 60;
476       bool activitystatus = false;
477       string status = null;
478       // input control
479       if (((DA.GetData<string>(0, ref filename) && DA.GetData<int>(1, ref timespan))
    && DA.GetData<bool>(2, ref activitystatus)) && filename.Length > 0)
480       {
481         System.Windows.Forms.Timer timer = new System.Windows.Forms.Timer();
482         // autobackup
483         if (activitystatus == true)
484         {
485           timer.Interval = timespan * 60 * 1000;
486           GH_Eloquera.dbBakName = filename;
487           timer.Tick += new System.EventHandler(GH_Eloquera.backupDB);
488           timer.Start();
489           status = "status: active (interval " + timespan.ToString() + "min, filename
    c:\\" + filename + ".eq)";
490         }
491         else
492         {
493           timer.Stop();
494           status = "status: inactive";
495         }
496         DA.SetData(0, status);
497         }
498     }
499     public override Guid ComponentGuid
500     {
501       get
502       {
503         return new Guid("{d1c00180-733d-4380-a02c-57263f3eb4e8}");
504       }
505     }
506     protected override Bitmap Internal_Icon_24x24
507     {
508       get
509       {
510         return GHEloqueraDB.Properties.Resources.DB_save;
511       }
512     }
513     public override GH_Exposure Exposure
514     {
515       get
```

```
516         {
517           return (GH_Exposure.dropdown | GH_Exposure.tertiary);
518         }
519       }
520     }
521   }
```

EINTRAG LÖSCHEN

Eingaben für diese Komponente ist die ID des Ursprungsobjektes des zu löschenden Datensatzes. Die Komponente liefert eine Statusmeldung und eine Ereignismeldung.

```
522   using System;
523   using System.Collections.Generic;
524   using System.Text;
525   using System.Collections;
526   using System.Drawing;
527   using System.IO;
528   using System.Linq;
529   using System.Windows.Forms;
530   using Grasshopper.Kernel;
531   using Grasshopper.Kernel.Types;
532   using RMA.OpenNURBS;
533   using Eloquera.Client;
534
535   namespace GHEloqueraDB
536   {
537     public class deleteItem : GH_Component
538     {
539       public deleteItem()
540         : base("Delete Data", "DELETE", "deletes data from an item in an eloquera
      database", "Extra", "CCTRL Database") //tooltip
541       {
542       }
543       // input variables
544       protected override void RegisterInputParams(GH_Component.GH_InputParamManager
      pManager)
545       {
546         pManager.Register_StringParam("ID", "ID", "the data origin's ID");
547       }
548       // output variables
549       protected override void RegisterOutputParams(GH_Component.GH_OutputParamManager
      pManager)
550       {
551         pManager.Register_StringParam("Status", "out", "whats going on");
552         pManager.Register_BooleanParam("Event", "Event", "return 'true' after
      writing");
553       }
554       // function
555       protected override void SolveInstance(IGH_DataAccess DA)
556       {
557         string itemID = null;
558         // input control
559         if (DA.GetData<string>(0, ref itemID) && itemID.Length > 0)
560         {
```

```
561          EDB_Entry placebo = new EDB_Entry();
562          placebo.Group = "test";
563          placebo.Data = null;
564          placebo.Type = "test";
565          placebo.OriginID = itemID;
566          // registrating dummy
567          GH_Eloquera.db.RegisterType(typeof(EDB_Entry));
568          // sql query
569          string querytext = "SELECT ONLY EDB_Entry WHERE OriginID ='" + itemID + "'";
570          var results = GH_Eloquera.db.ExecuteQuery(querytext);
571          foreach (var result in results)
572            {
573              GH_Eloquera.db.Delete(result);
574            }
575          // generating status message
576          string status = "_ items from component _" + itemID + "_ successfully deleted
     from database.";
577          // set output data
578          DA.SetData(0, status);
579          DA.SetData(1, true);
580
581        }
582      }
583      public override Guid ComponentGuid
584      {
585        get
586        {
587          return new Guid("{efa6819b-8271-4585-864c-800e6900a78b}");
588        }
589      }
590      protected override Bitmap Internal_Icon_24x24
591      {
592        get
593        {
594          return GHEloqueraDB.Properties.Resources.D_delete;
595        }
596      }
597      public override GH_Exposure Exposure
598      {
599        get
600        {
601          return GH_Exposure.primary;
602        }
603      }
604    }
605  }
```

DATENBANK ZURÜCKSETZEN

Eingabe für diese Komponente ist ob die Datenbank zurückgesetzt werden soll. Die Komponente liefert ausgabeseitig eine Statusmeldung und ein Ereignis bei erfolgreichem Ausführung.

```
606  using System;
```

```
607  using System.Collections.Generic;
608  using System.Text;
609  using System.Collections;
610  using System.Drawing;
611  using System.IO;
612  using System.Linq;
613  using System.Windows.Forms;
614  using Grasshopper.Kernel;
615  using Grasshopper.Kernel.Types;
616  using RMA.OpenNURBS;
617  using Eloquera.Client;
618
619  namespace GHEloqueraDB
620  {
621    public class resetDB : GH_Component
622    {
623      public resetDB()
624        : base("Reset Database", "DB RESET", "deletes all data from the active
     database", "Extra", "CCTRL Database") // tooltip
625      {
626      }
627      // input variables
628      protected override void RegisterInputParams(GH_Component.GH_InputParamManager
     pManager)
629      {
630        pManager.Register_BooleanParam("Reset", "Reset", "deletes all Data if 'true'");
631      }
632      // output variables
633      protected override void RegisterOutputParams(GH_Component.GH_OutputParamManager
     pManager)
634      {
635        pManager.Register_StringParam("Status", "out", "whats going on");
636        pManager.Register_BooleanParam("Event", "Event", "return 'true' after
     writing");
637      }
638      // function
639      protected override void SolveInstance(IGH_DataAccess DA)
640      {
641        bool checker = false;
642        // input check
643        if (DA.GetData<bool>(0, ref checker) && checker == true)
644        {
645          // deleting and again creating
646          GH_Eloquera.db.DeleteDatabase(GH_Eloquera.dbName, true);
647          GH_Eloquera.db.CreateDatabase(GH_Eloquera.dbName);
648          GH_Eloquera.db.OpenDatabase(GH_Eloquera.dbName);
649          // status message & output
650          string status = "database successfully reset.";
651          DA.SetData(0, status);
652          DA.SetData(1, true);
653        }
654      }
655      public override Guid ComponentGuid
656      {
657        get
658        {
659          return new Guid("{3647422e-1cbd-4cf7-b1ae-b2981cfdeb45}");
660        }
661      }
662      protected override Bitmap Internal_Icon_24x24
663      {
```

```
664         get
665         {
666           return GHEloqueraDB.Properties.Resources.DB_backup;
667         }
668       }
669       public override GH_Exposure Exposure
670       {
671         get
672         {
673           return (GH_Exposure.dropdown | GH_Exposure.secondary);
674         }
675       }
676     }
677   }
```

DATENBANK ÖFFNEN

Eingaben für diese Komponente sind Dateiname und Pfad für die zu öffnende Sicherungsdatei. Die Komponente liefert ausgabeseitig eine Statusmeldung über Erfolg oder Misserfolg des Vorgangs.

```
678   using System;
679   using System.Collections.Generic;
680   using System.Text;
681   using System.Collections;
682   using System.Drawing;
683   using System.IO;
684   using System.Linq;
685   using System.Windows.Forms;
686   using Grasshopper.Kernel;
687   using Grasshopper.Kernel.Types;
688   using RMA.OpenNURBS;
689   using Eloquera.Client;
690
691   namespace GHEloqueraDB
692   {
693     public class open : GH_Component
694     {
695       public open()
696         : base("Open Database", "DB Open", "opens a backup of an eloquera database",
      "Extra", "CCTRL Database") // tooltip
697       {
698       }
699       // input variables
700       protected override void RegisterInputParams(GH_Component.GH_InputParamManager
      pManager)
701       {
702         pManager.Register_StringParam("filename", "SRC", "filename (string) and path of
      the backupfile");
703       }
704       // output variables
705       protected override void RegisterOutputParams(GH_Component.GH_OutputParamManager
      pManager)
706       {
```

```
707       pManager.Register_StringParam("status", "out", "status");
708     }
709     // function
710     protected override void SolveInstance(IGH_DataAccess DA)
711     {
712       // definitions
713       string filename = null;
714       string status = null;
715       // input control
716       if (DA.GetData<string>(0, ref filename)&& filename.Length > 0)
717       {
718         // file check
719         if (System.IO.File.Exists("C:\\" + filename + ".eq") == true)
720         {
721           // closing and opening
722           GH_Eloquera.db.Close();
723           System.IO.File.Replace("C:\\" + filename + ".eq",
    "C:\\grasshopper_general.eq", "C:\\grasshopper_general_BAK.eq");
724           System.IO.File.Copy("C:\\grasshopper_general.eq", "C:\\" + filename +
    ".eq");
725           GH_Eloquera.db.OpenDatabase(GH_Eloquera.dbName);
726           status = "status: c:\\" + filename + ".eq successfully opened.";
727         }
728         else
729         {
730           status = "status: c:\\" + filename + ".eq not found.";
731         }
732         // set status
733         DA.SetData(0, status);
734       }
735     }
736     public override Guid ComponentGuid
737     {
738       get
739       {
740         return new Guid("{d02f6723-2907-4b6b-9b33-08d351058b38}");
741       }
742     }
743     protected override Bitmap Internal_Icon_24x24
744     {
745       get
746       {
747         return GHEloqueraDB.Properties.Resources.DB_open;
748       }
749     }
750     public override GH_Exposure Exposure
751     {
752       get
753       {
754         return (GH_Exposure.dropdown | GH_Exposure.tertiary);
755       }
756     }
757   }
758 }
```

ALLE IDENTIFIKIATIONSNUMMERN EINER GRUPPE LESEN

Eingaben für diese Komponente sind der Name der Gruppe von der alle Mitgliederidentifikationsnummern gelistet werden sollen und Indexnummer einer speziellen Auswahl. Die Komponente liefert ausgabeseitig eine Liste aller Identifikationsnummern der Gruppe sowie die eine weitere mit angegebener Indexnummer zusätzlich.

```
759  using System;
760  using System.Collections.Generic;
761  using System.Text;
762  using System.Collections;
763  using System.Drawing;
764  using System.IO;
765  using System.Linq;
766  using System.Windows.Forms;
767  using Grasshopper.Kernel;
768  using Grasshopper.Kernel.Types;
769  using RMA.OpenNURBS;
770  using Eloquera.Client;
771
772  namespace GHEloqueraDB
773  {
774    public class idsDB : GH_Component
775    {
776      public idsDB()
777        : base("IDs of group", "IDS", "reads all IDs of a group from an eloquera
     database", "Extra", "CCTRL Database") // tooltip
778      {
779      }
780      // input variables
781      protected override void RegisterInputParams(GH_Component.GH_InputParamManager
     pManager)
782      {
783        pManager.Register_StringParam("group", "G", "group data belongs to");
784        pManager.Register_IntegerParam("index", "I", "index of one ID", 0);
785      }
786      // output variables
787      protected override void RegisterOutputParams(GH_Component.GH_OutputParamManager
     pManager)
788      {
789        pManager.Register_StringParam("all IDs", "A", "all IDs of the group");
790        pManager.Register_StringParam("one ID", "ID", "the ID with the given index");
791
792      }
793      // function
794      protected override void SolveInstance(IGH_DataAccess DA)
795      {
796        // definition
797        string groupname = null;
798        int idIndex = 0;
799        List<String> idAll = new List<string>();
800        // input control
801        if ((DA.GetData<string>(0, ref groupname) && DA.GetData<int>(1, ref idIndex))
     && groupname.Length > 0)
802        {
803          string querytext = "SELECT ONLY EDB_Entry WHERE Group = '" + groupname + "'";
804          var result = GH_Eloquera.db.ExecuteQuery(querytext);
805          int i = 0;
806          foreach (EDB_Entry item in result)
807          {
```

```
808             // just concat and print result afterwards
809             idAll.Add(item.OriginID);
810             if (i == idIndex) DA.SetData(1, item.OriginID);
811             i++;
812           }
813           // set output
814           DA.SetDataList(0, idAll);
815         }
816       }
817       public override Guid ComponentGuid
818       {
819         get
820         {
821           return new Guid("{b6405abe-29e1-4933-b0c9-6bd67ba7848f}");
822         }
823       }
824       protected override Bitmap Internal_Icon_24x24
825       {
826         get
827         {
828           return GHEloqueraDB.Properties.Resources.D_groupIDs;
829         }
830       }
831       public override GH_Exposure Exposure
832       {
833         get
834         {
835           return GH_Exposure.primary;
836         }
837       }
838     }
839   }
```

GLOBALE AKTUALISIERUNGSGESCHWINDIGKEIT SETZEN

Eingabe für diese Komponente ist der Zeitabstand in Sekunden zwischen den automatischen Aktualisierungen der Lesebefehle. Die Komponente liefert ausgabeseitig eine Statusmeldung und im Hintergrund das Ereignis an dem sich automatische Aktualisierungen synchronisieren.

```
840   using System;
841   using System.Collections.Generic;
842   using System.Text;
843   using System.Collections;
844   using System.Drawing;
845   using System.IO;
846   using System.Linq;
847   using System.Windows.Forms;
848   using Grasshopper.Kernel;
849   using Grasshopper.Kernel.Types;
850   using RMA.OpenNURBS;
851   using Eloquera.Client;
852
```

```
853  namespace GHEloqueraDB
854  {
855    public class refreshrate : GH_Component
856    {
857      public refreshrate()
858        : base("Refreshrate of Data", "Refresh", "sets the global refreshrate for read
     commands", "Extra", "CCTRL Database") // tooltip
859      {
860      }
861      // registering input variables
862      protected override void RegisterInputParams(GH_Component.GH_InputParamManager
     pManager)
863      {
864        pManager.Register_IntegerParam("timespan", "T", "seconds (int) between read
     commands", 360); // default value 1 hour
865      }
866      // registering output variables
867      protected override void RegisterOutputParams(GH_Component.GH_OutputParamManager
     pManager)
868      {
869        pManager.Register_StringParam("status", "out", "status");
870      }
871      // function
872      protected override void SolveInstance(IGH_DataAccess DA)
873      {
874        // definition and timer set
875        int timespan = GH_Eloquera.Timer.Interval;
876        string status = null;
877        // input control
878        if (DA.GetData<int>(0, ref timespan))
879        {
880          // setting timer
881          GH_Eloquera.Timer.Interval = timespan * 1000;
882          status = "status: global interval time for reading from database set to " +
     GH_Eloquera.Timer.Interval.ToString() + "msec.";
883          // output
884          DA.SetData(0, status);
885        }
886      }
887      public override Guid ComponentGuid
888      {
889        get
890        {
891          return new Guid("{71378377-94c5-469a-ab13-13525a0e1a37}");
892        }
893      }
894      protected override Bitmap Internal_Icon_24x24
895      {
896        get
897        {
898          return GHEloqueraDB.Properties.Resources.D_refreshrate;
899        }
900      }
901      public override GH_Exposure Exposure
902      {
903        get
904        {
905          return (GH_Exposure.dropdown | GH_Exposure.primary);
906        }
907      }
908    }
```

```
909  }
```

DATENBANK VERBINDUNGSKONFIGURATION

Eingaben für diese Komponente sind die IP des Datenbankservers, der Benutzername für die Verbindung, das Password des Benutzers und der Name der Datenbank. Die Komponente liefert ausgabeseitig eine Statusmeldung, und stellt im Hintergrund die Verbindung her.

```
910  using System;
911  using System.Collections.Generic;
912  using System.Text;
913  using System.Collections;
914  using System.Drawing;
915  using System.IO;
916  using System.Linq;
917  using System.Windows.Forms;
918  using Grasshopper.Kernel;
919  using Grasshopper.Kernel.Types;
920  using RMA.OpenNURBS;
921  using Eloquera.Client;
922
923  namespace GHEloqueraDB
924  {
925    public class configDB : GH_Component
926    {
927      public configDB()
928        : base("Config Database", "CONFIG", "settings for the database connection",
     "Extra", "CCTRL Database") // tooltip
929      {
930      }
931      // registering input variables
932      protected override void RegisterInputParams(GH_Component.GH_InputParamManager
     pManager)
933      {
934        pManager.Register_StringParam("Server IP", "IP", "IP of the database server");
935        pManager.Register_StringParam("User", "USR", "user for connecting to the
     server");
936        pManager.Register_StringParam("Password", "PWD", "password for connecting to
     the server");
937        pManager.Register_StringParam("Database", "DB", "name of the database
     connecting to");
938      }
939      // registering output variables
940      protected override void RegisterOutputParams(GH_Component.GH_OutputParamManager
     pManager)
941      {
942        pManager.Register_StringParam("status", "out", "status");
943      }
944      // function
945      protected override void SolveInstance(IGH_DataAccess DA)
946      {
947        // definitions
```

```
948         string status = null;
949         string sip = null;
950         string susr = null;
951         string spwd = null;
952         string dbname = null;
953         string wholedb = null;
954         // input control
955         if (((DA.GetData<string>(0, ref sip) && DA.GetData<string>(1, ref susr)) &&
    DA.GetData<string>(2, ref spwd)) && DA.GetData<string>(3, ref dbname))
956         {
957           // check if connection is already established (importand when multiple files
    are connected)
958           if (GH_Eloquera.db.IsOpen == false)
959           {
960             // connection data
961             GH_Eloquera.dbName = dbname;
962             GH_Eloquera.dbBakName = GH_Eloquera.dbName + "_bak";
963             wholedb = "server=" + sip + ";user=" + susr + ";password=" + spwd +
    ";options=persistent;"; // persistent
964             GH_Eloquera.db = new DB(wholedb);
965             // connecting if database exists
966             if ((Array.IndexOf(GH_Eloquera.db.GetDbList(), GH_Eloquera.dbName) > 0))
967             {
968               GH_Eloquera.db.OpenDatabase(GH_Eloquera.dbName);
969               GH_Eloquera.Timer.Start();
970               status = "status: connection to " + dbname + " @ " + sip + " for " + susr
    + " successfully established.";
971             }
972             else // if file not found
973             {
974               status = "status: connection to " + dbname + " @ " + sip + " for " + susr
    + " not established. File not found.";
975             }
976           }
977           else // if other database is to open
978           {
979             if (GH_Eloquera.dbName != dbname)
980             {
981               string dbold = GH_Eloquera.dbName;
982               GH_Eloquera.db.Close();
983               wholedb = "server=" + sip + ";user=" + susr + ";password=" + spwd +
    ";options=persistent;"; // persistent
984               GH_Eloquera.db = new DB(wholedb);
985               GH_Eloquera.dbName = dbname;
986               GH_Eloquera.dbBakName = GH_Eloquera.dbName + "_bak";
987               if ((Array.IndexOf(GH_Eloquera.db.GetDbList(), GH_Eloquera.dbName) > 0))
988               {
989                 GH_Eloquera.db.OpenDatabase(GH_Eloquera.dbName);
990                 GH_Eloquera.Timer.Start();
991                 status = "status: connection to "+ dbold +" terminated and connection
    to " + dbname + " @ " + sip + " for " + susr + " successfully established.";
992               }
993               else
994               {
995                 status = "status: connection to " + dbold + " terminated and connection
    to " + dbname + " @ " + sip + " for " + susr + " not established. File not found.";
996               }
997
998             }
999             else
1000              {
```

```
1001                status = "status: connection to " + dbname + " @ " + sip + " for " +
     susr + " already established.";
1002              }
1003            }
1004            // status set
1005            DA.SetData(0, status);
1006          }
1007        }
1008        public override Guid ComponentGuid
1009        {
1010          get
1011          {
1012            return new Guid("{2d8fbc0d-1174-4bcb-9d0c-bb88c151f6bc}");
1013          }
1014        }
1015        protected override Bitmap Internal_Icon_24x24
1016        {
1017          get
1018          {
1019            return GHEloqueraDB.Properties.Resources.DB_config;
1020          }
1021        }
1022        public override GH_Exposure Exposure
1023        {
1024          get
1025          {
1026            return GH_Exposure.secondary;
1027          }
1028        }
1029      }
1030    }
1031    public class EDB_Entry // special class of saved items
1032    {
1033      [Index]
1034      public string Group { get; set; }
1035      [Index]
1036      public string Type { get; set; }
1037      [Index]
1038      public string OriginID { get; set; }
1039      [Index]
1040      public List<object> Data { get; set; }
1041      [Index]
1042      public List<bool> DataBool { get; set; }
1043      [Index]
1044      public List<int> DataInt { get; set; }
1045      [Index]
1046      public List<double> DataDouble { get; set; }
1047      [Index]
1048      public List<string> DataString { get; set; }
1049    }
1050    public static class GH_Eloquera // database class
1051    {
1052      private static string mfilename = "empty123";
1053      private static string mbakname = null;
1054      private static DB mdb = new DB(); // "server=(local);options=inmemory,persist;"
     // sonst options=inmemory,persist; oder =none;
1055      private static System.Windows.Forms.Timer mytimer = new
     System.Windows.Forms.Timer();
1056      public static System.Windows.Forms.Timer Timer
1057      {
1058        get
```

```
1059        {
1060          return mytimer;
1061        }
1062        set
1063        {
1064          mytimer = value;
1065        }
1066      }
1067      // database timer
1068      public static void addEvent(EventHandler ev)
1069      {
1070        GH_Eloquera.Timer.Tick += ev;
1071      }
1072      public static void removeEvent(EventHandler ev)
1073      {
1074        GH_Eloquera.Timer.Tick -= ev;
1075      }
1076      public static DB db
1077      {
1078        get
1079        {
1080          return mdb;
1081        }
1082        set
1083        {
1084          mdb = value;
1085        }
1086      }
1087      public static string dbName
1088      {
1089        get
1090        {
1091          return mfilename;
1092        }
1093        set
1094        {
1095          mfilename = value;
1096        }
1097      }
1098      public static string dbBakName
1099      {
1100        get
1101        {
1102          return mbakname;
1103        }
1104        set
1105        {
1106          mbakname = value;
1107        }
1108      }
1109      public static void backupDB(Object myObject, EventArgs myEventArgs)
1110      {
1111        GH_Eloquera.db.Close();
1112        if (System.IO.File.Exists("C:\\" + GH_Eloquera.dbBakName.ToString() + ".eq")
      == true) System.IO.File.Delete("C:\\" + GH_Eloquera.dbBakName.ToString() + ".eq");
1113        System.IO.File.Copy("C:\\" + GH_Eloquera.dbName.ToString() + ".eq", "C:\\" +
      GH_Eloquera.dbBakName.ToString() + ".eq");
1114        GH_Eloquera.db.OpenDatabase(GH_Eloquera.dbName);
1115      }
1116    }
```

DATENBANK BEREINIGEN

Eingabe für diese Komponente ist die Angabe ob der Befehl ausgeführt werden soll. Die Komponente liefert ausgabeseitig eine Statusmeldung und ein Ereignis bei erfolgreicher Ausführung.

```
1117    using System;
1118    using System.Collections.Generic;
1119    using System.Text;
1120    using System.Collections;
1121    using System.Drawing;
1122    using System.IO;
1123    using System.Linq;
1124    using System.Windows.Forms;
1125    using Grasshopper.Kernel;
1126    using Grasshopper.Kernel.Types;
1127    using RMA.OpenNURBS;
1128    using Eloquera.Client;
1129
1130
1131    namespace GHEloqueraDB
1132    {
1133      public class compactDB : GH_Component
1134      {
1135        public compactDB()
1136          : base("Clean Removed Data", "DB COMPACT", "cleans removed entries from the
      database", "Extra", "CCTRL Database") // tolltip
1137        {
1138        }
1139        // registering input variables
1140        protected override void RegisterInputParams(GH_Component.GH_InputParamManager
      pManager)
1141        {
1142          pManager.Register_BooleanParam("Clean", "C", "cleans if 'true'");
1143        }
1144        // registering output variables
1145        protected override void
      RegisterOutputParams(GH_Component.GH_OutputParamManager pManager)
1146        {
1147          pManager.Register_StringParam("Status", "out", "whats going on");
1148          pManager.Register_BooleanParam("Event", "Event", "return 'true' after
      writing");
1149        }
1150        // function
1151        protected override void SolveInstance(IGH_DataAccess DA)
1152        {
1153          bool checker = false;
1154          // user input check
1155          if (DA.GetData<bool>(0, ref checker) && checker == true)
1156          {
1157            GH_Eloquera.db.Compact();
1158            // compact command for cleaning use
1159            string status = "database successfully tidied up.";
1160            // setting output
1161            DA.SetData(0, status);
1162            DA.SetData(1, true);
1163          }
```

```
1164        }
1165        public override Guid ComponentGuid
1166        {
1167          get
1168          {
1169            return new Guid("{52c28d2e-fe76-45dc-8005-5826671740e4}");
1170          }
1171        }
1172        protected override Bitmap Internal_Icon_24x24
1173        {
1174          get
1175          {
1176            return GHEloqueraDB.Properties.Resources.DB_compact;
1177          }
1178        }
1179        public override GH_Exposure Exposure
1180        {
1181          get
1182          {
1183            return (GH_Exposure.dropdown | GH_Exposure.secondary);
1184          }
1185        }
1186      }
1187    }
```

DATENBANK ERSTELLEN

Eingabe für diese Komponente ist der Name der zu erstellenden Datenbank. Die Komponente liefert ausgabeseitig eine Statusmeldung bei erfolgreichem Schreibvorgang.

```
1188    using System;
1189    using System.Collections.Generic;
1190    using System.Text;
1191    using System.Collections;
1192    using System.Drawing;
1193    using System.IO;
1194    using System.Linq;
1195    using System.Windows.Forms;
1196    using Grasshopper.Kernel;
1197    using Grasshopper.Kernel.Types;
1198    using RMA.OpenNURBS;
1199    using Eloquera.Client;
1200
1201    namespace GHEloqueraDB
1202    {
1203      public class newDB : GH_Component
1204      {
1205        public newDB()
1206          : base("Create New Database", "NEW DB", "creates a new eloquera database",
     "Extra", "CCTRL Database") // tooltip
1207        {
1208        }
1209        // registering input variables
1210        protected override void RegisterInputParams(GH_Component.GH_InputParamManager
     pManager)
```

```
1211        {
1212          pManager.Register_StringParam("name", "N", "name (string) of the new
      database");
1213        }
1214        // registering output variables
1215        protected override void
      RegisterOutputParams(GH_Component.GH_OutputParamManager pManager)
1216        {
1217          pManager.Register_StringParam("status", "out", "status");
1218        }
1219        // function
1220        protected override void SolveInstance(IGH_DataAccess DA)
1221        {
1222          // definition
1223          string dbname = null;
1224          string status = null;
1225          // user input control
1226          if (DA.GetData<string>(0, ref dbname) && dbname.Length > 0)
1227          {
1228            if (Array.IndexOf(GH_Eloquera.db.GetDbList(), dbname) < 0)
1229            {
1230              GH_Eloquera.db.CreateDatabase(dbname);
1231              status = "status: database" + dbname + "successfully created.";
1232            }
1233            else
1234            {
1235              status = "status: database" + dbname + "is already existing.";
1236            }
1237            // status message
1238            DA.SetData(0, status);
1239          }
1240        }
1241        public override Guid ComponentGuid
1242        {
1243          get
1244          {
1245            return new Guid("{658eb2e4-768a-4a9f-82a4-708b44a3f9cb}");
1246          }
1247        }
1248        protected override Bitmap Internal_Icon_24x24
1249        {
1250          get
1251          {
1252            return GHEloqueraDB.Properties.Resources.DB_new;
1253          }
1254        }
1255        public override GH_Exposure Exposure
1256        {
1257          get
1258          {
1259            return (GH_Exposure.dropdown | GH_Exposure.tertiary);
1260          }
1261        }
1262      }
1263    }
```

AUFLISTEN ALLER VERFÜGBAREN DATENBANKEN

Eingabe für diese Komponente ist ob mit der Vorgang ausgeführt werden soll. Die Komponente liefert ausgabeseitig eine Statusmeldung und eine Liste mit den Namen aller verfügbaren Datenbanken auf dem Server.

```
1264    using System;
1265    using System.Collections.Generic;
1266    using System.Text;
1267    using System.Collections;
1268    using System.Drawing;
1269    using System.IO;
1270    using System.Linq;
1271    using System.Windows.Forms;
1272    using Grasshopper.Kernel;
1273    using Grasshopper.Kernel.Types;
1274    using RMA.OpenNURBS;
1275    using Eloquera.Client;
1276
1277    namespace GHEloqueraDB
1278    {
1279      public class availableDB : GH_Component
1280      {
1281        public availableDB()
1282          : base("Lists Available Database", "AVABL DB", "lists all available eloquera
      database", "Extra", "CCTRL Database") // tooltip
1283        {
1284        }
1285        // registering input variables
1286        protected override void RegisterInputParams(GH_Component.GH_InputParamManager
      pManager)
1287        {
1288          pManager.Register_BooleanParam("list", "l", "lists all databases if true",
      true);
1289        }
1290        // registering output variables
1291        protected override void
      RegisterOutputParams(GH_Component.GH_OutputParamManager pManager)
1292        {
1293          pManager.Register_StringParam("status", "out", "status");
1294          pManager.Register_StringParam("list of databases", "l", "list of available
      databases found on server.");
1295        }
1296        // function
1297        protected override void SolveInstance(IGH_DataAccess DA)
1298        {
1299          // definition
1300          bool dblist = false;
1301          string status = null;
1302          List<string> names = new List<string>();
1303          int i = 0;
1304          // user input control
1305          if (DA.GetData<bool>(0, ref dblist))
1306          {
1307            foreach (string litem in GH_Eloquera.db.GetDbList())
1308            {
1309              i++;
1310              names.Add(litem);
1311            }
1312            status = "status: "+i+"Databases found and listed.";
1313            // setting output
```

```
1314            DA.SetData(0, status);
1315            DA.SetDataList(1, names);
1316          }
1317        }
1318        public override Guid ComponentGuid
1319        {
1320          get
1321          {
1322            return new Guid("{078ca83f-1ea1-4e5e-b4f4-d77f180d8711}");
1323          }
1324        }
1325        protected override Bitmap Internal_Icon_24x24
1326        {
1327          get
1328          {
1329            return GHEloqueraDB.Properties.Resources.DB_available;
1330          }
1331        }
1332        public override GH_Exposure Exposure
1333        {
1334          get
1335          {
1336            return (GH_Exposure.dropdown | GH_Exposure.tertiary);
1337          }
1338        }
1339      }
1340    }
```

DATENBANK LÖSCHEN

Eingabe für diese Komponente ist der Name der zu löschenden Datenbank. Die Komponente liefert ausgabeseitig eine Statusmeldung.

```
1341    using System;
1342    using System.Collections.Generic;
1343    using System.Text;
1344    using System.Collections;
1345    using System.Drawing;
1346    using System.IO;
1347    using System.Linq;
1348    using System.Windows.Forms;
1349    using Grasshopper.Kernel;
1350    using Grasshopper.Kernel.Types;
1351    using RMA.OpenNURBS;
1352    using Eloquera.Client;
1353
1354    namespace GHEloqueraDB
1355    {
1356      public class deleteDB : GH_Component
1357      {
1358        public deleteDB()
1359          : base("Deletes Database", "DELETE DB", "deletes aneloquera database",
     "Extra", "CCTRL Database") // tooltip
1360        {
1361        }
```

```
1362        // registering input variables
1363        protected override void RegisterInputParams(GH_Component.GH_InputParamManager
      pManager)
1364        {
1365          pManager.Register_StringParam("name", "N", "name (string) of the database to
      delete");
1366        }
1367        // registering output variables
1368        protected override void
      RegisterOutputParams(GH_Component.GH_OutputParamManager pManager)
1369        {
1370          pManager.Register_StringParam("status", "out", "status");
1371        }
1372        // function
1373        protected override void SolveInstance(IGH_DataAccess DA)
1374        {
1375          // definition
1376          string dbname = null;
1377          string status = null;
1378          // user input control
1379          if (DA.GetData<string>(0, ref dbname) && dbname.Length > 0)
1380          {
1381            if (Array.IndexOf(GH_Eloquera.db.GetDbList(), dbname) >= 0)
1382            {
1383              GH_Eloquera.db.DeleteDatabase(dbname, true);
1384              status = "status: database" + dbname + "successfully deleted.";
1385            }
1386            else
1387            {
1388              status = "status: database" + dbname + "is not existing.";
1389            }
1390            DA.SetData(0, status);
1391          }
1392        }
1393        public override Guid ComponentGuid
1394        {
1395          get
1396          {
1397            return new Guid("{e1ddef55-dd23-4d54-83cc-741ebfe85d4a}");
1398          }
1399        }
1400        protected override Bitmap Internal_Icon_24x24
1401        {
1402          get
1403          {
1404            return GHEloqueraDB.Properties.Resources.DB_delete;
1405          }
1406        }
1407        public override GH_Exposure Exposure
1408        {
1409          get
1410          {
1411            return (GH_Exposure.dropdown | GH_Exposure.tertiary);
1412          }
1413        }
1414      }
1415    }
```

ANHANG B

Der Schriftwechsel mit den Entwicklern von Eloquera und Grasshopper zur Übersetzungsproblematik der Objekte

KORRESPONDENZ MIT DEM TEAM VON GRASSHOPPER[1]

i work for the database plugin again. and i need to say, as far as i am it looks pretty cool. but i still have on problem i can't solve myself. i did some network extension, so multiuser can access a database hosted on a server. till now i did extensive testing and i found this: there are some objects i can save like :vectors, points, strings, float and integer and there are lots i can't like: planes, breps (surfaces), lines, circles, ... the items i save can be accessed by many users and are persistent. but the ones i cannot save correctly, i can just use in the single instance of the program with my .dll i created them and stored them in the database - as long as i don't close the program. it's like there is some sort of connected to something going on in the memory. some of the unsaveable items create an error message when i try to access them from a second account like : nullplane, line (0.00mm) or invalid circle, while others just crash rhino (some sort of display error). i know grasshopper has classes for every geometry (like point, plane, line, vector and so on), but for simplification i saved e.g. a List as a List, when using "inmemory mode" this works fine. i thought this would be my mistake now, but in an other test i remove all converting steps (i set all types to GHBrep and tried to save a GHBrep) without success again. maybe on of you has a clue why points and vectors work while breps dont?

DAVID RUTTEN for this to work there would have to be proper (de)serialization in place. Points may be smart enough to deserialize themselves from "0.5, 4.5, -9", but a Brep cannot deserialize itself from "Closed Brep". I don't know how data is written to the database. Is it anything more advanced than calling ToString() on the object? At any rate, Grasshopper cannot serialize Curves, Surfaces, Meshes and Breps either. For this you'll need to use some sort of OpenNurbs functions to convert these types into byte-arrays

1 **Die Diskussion fand im Internet von 24.11.2010 bis 26.11.2010 statt. Diskussionspartner waren: David Rutten und Steve Baer von Robert McNeel & Associates**

which can then be stored on your database. I'll need to pass you on to Steve Baer for this, he knows more about the deserialization stuff than me.

thanks david, now i know what i'm looking for. btw the database stores objects without splitting into tables, e.g:

```
1416        Cinema cinema = new Cinema() {
1417        Location = "Sydney",
1418        OpenDates = new DateTime[] { new DateTime(2003, 12, 10), new
    DateTime(2003, 10, 3) }
1419        };
1420        db.Store(cinema);
```

this simplicity and the fact that eloquera is one of the fastest solutions where reasons i chose this database.

DAVID RUTTEN Steve and me have been looking at the Eloquera docs for a bit. We're pretty certain it's working on Points and Vectors because we decorated those types with the [Serializable()] attribute. We should do the same for Brep, Curve etc. but that will take quite a bit of work. It's also not near the top of the list of things that need to happen before we ship V5. The most important question in the meantime is what sort of control you have over how and when Eloquera serializes and deserializes its data.

i discussed the topic with some eloquera developers, and i need to say that slowly i'm reaching the limit of my coding abilities. here my last info: "The Eloquera DB does not need the class to be marked as [Serializable] in order to be persist its instances. And there is a possibility that one of the base classes uses [NotSerialized] for some of its fields. Then, the database will respect the attribute, and will not persist such fields. Another possibility is that some of the fields are marked as readonly (a C# keyword), and they cannot be updated outside of the class' constructor. There is a way to overcome this, but it is not yet available through public methods as modification of readonly fields can have some nasty side effects in the applications."

i think you know the definitions better than anybody else, do some classes use [NotSerialized] for some of it's fields or are some marked as 'readonly'? i would forward this information to the eloquera team, they seem positive to find a solution.

STEVE BAER I'm pretty sure that these classes need to be made serializable for this to work. Many of the geometry classes in RhinoCommon are layered on top of unmanaged C++ classes which means that Eloquera has no way of figuring out all of the internal pieces of the RhinoCommon geometry class is unless explicitly provide a way for the classes to be serialized.

i'm afraid you're right. but the one thing i can't understand is why e.g. circles, lines, etc won't work either. i looked into the definition, and the only difference i found out is the additional

`DebuggerDisplay("{m_x}, {m_y}, {m_z}")`. *would you please explain this a little further for me. i really try to understand this.many thanks so far*

DAVID RUTTEN The DebuggerDisplay attribute allow you to look at an instance of the type in a debugger window and see what value it has. If we didn't add DebuggerDisplay then all you'd see is the type name and not it's value. It does nothing to help (de)serialization.

thanks for your answers david and steve, you helped me a lot in understanding this matter. theoretically it would be possible to cast data from an original "GH_Brep" (which is {get;] only) into a new class "My_Brep" (which would be {get;set;} and write this back into a new "GH_Brep" per using System.Reflection. like in this simple example:

```
1421    using System;
1422    using System.Collections.Generic;
1423    using System.Text;
1424    using System.Collections;
1425    using Eloquera.Client;
1426    using System.Windows.Forms;
1427    using System.Reflection;
1428
1429    namespace RemoteExample
1430    {
1431    public class my_EDB
1432    {
1433    [Index]
1434    public string Group { get; set; }
1435    [Index]
1436    public string Type { get; set; }
1437    [Index]
1438    public string OriginID { get; set; }
1439    [Index]
1440    public string Data { get; set; }
1441    }
1442    public class EDB
1443    {
1444    private string mGroup = "testgroup";
1445    private string mType = "testtype";
1446    private string mOriginID = "testoriginid";
1447    private string mData = "testdata";
1448    [Index]
1449    public string Group { get{return mGroup;}}
1450    [Index]
1451    public string Type { get { return mType; } }
1452    [Index]
1453    public string OriginID { get { return mOriginID; } }
1454    [Index]
1455    public string Data { get { return mData; } }
1456    }
1457
1458    public class example
1459    {
1460    public static void Main(string[] args)
1461    {
1462    my_EDB test3 = new my_EDB()
```

```
1463    {
1464    Group = "MYGROUPNAME",
1465    Type = "test1",
1466    OriginID = "ecfd2a5c-2585-4b7d-bc4f-8a33ce0a84f6",
1467    Data = "xxxxxxxx"
1468    };
1469    EDB test2 = new EDB();
1470    EDB test4= new EDB();
1471    Console.WriteLine("test:");
1472    Console.WriteLine(test3.Group);
1473    Console.WriteLine(test3.Type);
1474    Console.WriteLine(test3.OriginID);
1475    Console.WriteLine(test3.Data);
1476    Console.WriteLine("-----------------------");
1477    Console.WriteLine("test2:");
1478    Console.WriteLine(test2.Group);
1479    Console.WriteLine(test2.Type);
1480    Console.WriteLine(test2.OriginID);
1481    Console.WriteLine(test2.Data);
1482    FieldInfo field = typeof(EDB).GetField("mGroup",
1483    System.Reflection.BindingFlags.Instance |
1484    System.Reflection.BindingFlags.NonPublic);
1485    field.SetValue(test2, test3.Group);
1486    Console.WriteLine("-----------------------");
1487    Console.WriteLine("test2 NEU:");
1488    Console.WriteLine(test2.Group);
1489    Console.WriteLine(test2.Type);
1490    Console.WriteLine(test2.OriginID);
1491    Console.WriteLine(test2.Data);
1492    Console.WriteLine("-----------------------");
1493    Console.WriteLine("test3:");
1494    Console.WriteLine(test4.Group);
1495    Console.WriteLine(test4.Type);
1496    Console.WriteLine(test4.OriginID);
1497    Console.WriteLine(test4.Data);
1498    MessageBox.Show("closewindow");
1499    }
1500    }
1501    }
```

for sure this would be pretty sloppy, but i think it could do it's job as a workaround, until the other grasshopper types got an added "set{}" to the properties and become writeable too. of course i don't know how complicated this will be, but maybe i give it a shot.

KORRESPONDENZ MIT DEM TEAM VON ELOQUERA[2]

my plugin for rhino3d is able to connect to the database on the server, and as long as just one user per database is online everything works fine. but as soon as i try to run two instances of rhino for testing the multiuser capability one problem emerges: while GetRegisteredTypeList() shows there would be 115 elements, and the filesize from about 22MB indicates there would be data too, every sql query returns empty. (i tried refreshing the objects, but without success. i also tried backing up the db every time i write and restoring it as soon as i log in, but first of all this would be more than sloppy and second the bak.eq just reaches 8MB, where the original.eq got 22MB - maybe this was the reason this approach wasn't successfull either)
the second problem is, that there is no persistence of the data. when i login i cannot read whats already stored, though i see typelist and filesize. at the moment i presume that the error is located somewhere in the code above, or in the server's config file which is pretty standard except WriteThru is set on true.
DMYTRO BABLINYK Can you try 3.0.0.27, in Examples solution run SimpleExample project. Make for simplicity loop at the end like this

```
1502       IEnumerable listOfMovies = db.ExecuteQuery("SELECT Movie WHERE
    Genre> >= 5");
1503       foreach (Movie mov in listOfMovies)
1504       {
1505          Console.WriteLine(mov.Title);
1506       }
```

Do you have data displayed every time you run this code?
i'm afraid the code runs as it is supposed to. i modified the example as you wrote, replaced the 11 lines with yours, and i see all entries every time i run the file. i am using the most recent version 3.0.0.27 of eloquera (32bit for the client and 64bit for the server), and vista 64bit ultimate on both pcs. any clue?
DMYTRO BABLINYK You can start bringing your code into example to replicate the issue. May be you are connecting to the local database, may be something with the queries. You need to make a simple example to replicate the problem.
i did extensive testing and found this: there are some objects i can save like :vectors, points, strings, float and integer and there are lots i can't like: planes, breps (surfaces), lines, circles, ...

2 Die Diskussion fand per Email von 23.11.2010 bis 26.11.2010 statt. Diskussionspartner waren: Dmytrio Bablinyk und Jay Haybatov von Eloquera (Eloquera ist eine australische Firma, mit Büros in Sydney, NSW (Australien) ind Chicago, IL (USA), gegründet 2007 in Sydney.)

the items can be accessed by many users and are persistent. but the ones i cannot save correctly i just can use in the single instance of the program with my .dll i created them and wrote them in the db - as long as i don't close the program - it's like they are some sort of connected to something going on in the memory. some of the unsaveable items create an error message when i try to access them from a second account like : nullplane, null line or invalid circle others just let the program crash (some sort of display error).i know grasshopper (the plugin i write my plugin for) has classes for every geometry (like point, plane, line, vector and so on), but for simplification i saved e.g. a List<GHPoint> as a List<Object>, when using "inmemory mode" this works fine. i thought this would be my mistake now, but in an other test i remove all converting steps (i set all types to GHBrep and tried to save a GHBrep) without success again. do you think the error is located in the structure of the classes itself? and why do some work and others dont?

JAY HAYBATOV As I am not familiar with Grasshopper, could you please send me the definition of any class that has problems with persistence. It can be only the data definition section, without methods, if the class is big. We will try to figure out what can be done to resolve an issue.

i added the definition of the brep class (both complete and compact). Brep represents a 3D polysurface. GH_Brep wraps the functionality of the OpenNURBS OnBrep class. i hope you will find something, many thanks for your help in advance.

JAY HAYBATOV It appears that the class structure for Grasshopper is quite complex, and it is quite difficult to determine the cause of the problem by analysing a single class. Internally, GH_Brep uses a GH_GooProxy<GH_Brep> class, and it may contain something preventing it from being deserialised properly. Could you please send me code for the GH_GooProxy<GH_Brep> class as well? The Eloquera DB internally uses its own serialisation/deserialisation methods, which are quite advanced and do not rely on the standard serialisation mechanism. So, the problem may lay deeper that it may seem at the beginning. Hope that the remote debugging session will crack the problem pretty quickly. :-)

while grasshopper itself is a plugin for rhino, i thought i send you the original rhino brep definition too. additional i added the gh_point definition which is working, and the gh_gooproxy you asked for. it would be pretty awsome if you could crack the problem.

JAY HAYBATOV The Eloquera DB does not need the class to be marked as [Serializable] in order to be persist its instances. And there is a possibility that one of the base classes uses [NotSerialized] for some of its fields. Then, the database will respect the attribute, and will not persist such fields. Another possibility is that some of the fields are marked as readonly (a C# keyword), and they cannot be updated outside of the class' constructor. There is a

way to overcome this, but it is not yet available through public methods as modification of readonly fields can have some nasty side effects in the applications. Could you please confirm whether any of these is the case? Then we can plan how to solve the problem a step further.

DMYTRO BABLINYK [Serializable] attribute to my knowledge is ignored. There is attribute [Ignore] that states not to save the property. There may be issue with readonly fields, readonly fields can be stored, queried but not restored back. Also any pointers, delegates will not be saved as they are meaningless outside of current session.

first i asked the grasshopper team, but i just got this: "I'm pretty sure that these classes need to be made serializable for this to work. Many of the geometry classes in RhinoCommon are layered on top of unmanaged C++ classes which means that Eloquera has no way of figuring out all of the internal pieces of the RhinoCommon geometry class is unless explicitly provide a way for the classes to be serialized." on the other hand i looked trough the definitions for '[NotSerialized]' and 'readonly' and i found nothing, the whole stuff is in the appendix. But I found something else: there are other classes which include [Serializable], but the ones working are the only ones with an additional DebuggerDisplay("{m_x}, {m_y}, {m_z}") . But i don't have anything more

DMYTRO BABLINYK In Arc class, for instance, you have a readonly properties

```
1507        public bool IsValid { get; }
1508        public bool IsCircle { get; }
```

i totally forgot to look this way, sry. but the classes which work, ex point3d and vector3d do have some too:

```
1509        point3d
1510            public static Point3d Origin { get; }
1511            public static Point3d Unset { get; }
1512        public bool IsValid { get; }
1513            public double MinimumCoordinate { get; }
1514            public double MaximumCoordinate { get; }
1515
1516
1517        vector3d
1518            public static Vector3d Zero { get; }
1519            public static Vector3d XAxis { get; }
1520            public static Vector3d YAxis { get; }
1521            public static Vector3d ZAxis { get; }
1522            public static Vector3d Unset { get; }
1523            public bool IsValid { get; }
1524            public double MinimumCoordinate { get; }
1525            public double MaximumCoordinate { get; }
1526            public double Length { get; }
1527            public double SquareLength { get; }
1528            public bool IsUnitVector { get; }
1529            public bool IsZero { get; }
```

there are some get only elements in the working classes origins too, but in the grasshopper's type classes e.g. public class GH_PointProxy : GH_GooProxy<GH_Point> all properteis are get{} set{}, where in GH_BRepProxy : GH_GooProxy<GH_Brep> just the field for ObjectID is. all other important fields, like VertexCount or Volume are {get;} only.for me it seems you finally found the big difference Dmytro, thanks! but what means jay when i writes "There is a way to overcome this, but it is not yet available through public methods as modification of readonly fields can have some nasty side effects in the applications.", which way is he thinking of?

DMYTRO BABLINYK Actually readonly property Jay has noticed, I just wrote :-) On the multi-core/multiprocessor systems each processing core/CPU has its own local cache. The .NET runtime may optimize the code by caching the readonly values in the each core's cache after the object is initialized. Updating the readonly fields via reflection may lead to modification of the value in the cache of the core executing the modification code. Other cores may be unaware about the change, and still may use the previous value (most likely, null).This scenario may lead to intermittent exceptions, which cannot be traced during the debugging process, or to some data damage. Previously, the Eloquera DB was supporting the modification of the readonly fields in MSIL. About 2-3 years ago Microsoft produced a patch for all versions of the .NET framework, that prevented the modification of readonly fields in MSIL, for the very same reason. So, updating the readonly fields is to let the customers shoot their feet off without even suspecting it. Therefore, we decided not to circumvent the .NET security system.

thanks a lot for your help in understanding this things, but may i ask one final question? _theoretically_, would it be possible to cast the data from a original "GH_Brep" (which is {get;] only) into a new class "My_Brep" (which would be {get;set;} and write this back into a "GH_Brep" per using System.Reflection? e.g. this way

```
1530        using System;
1531        using System.Collections.Generic;
1532        using System.Text;
1533        using System.Collections;
1534        using Eloquera.Client;
1535        using System.Windows.Forms;
1536        using System.Reflection;
1537
1538        namespace RemoteExample
1539        {
1540          public class my_EDB
1541          {
1542            [Index]
1543            public string Group { get; set; }
1544            [Index]
1545            public string Type { get; set; }
1546            [Index]
1547            public string OriginID { get; set; }
1548            [Index]
```

```
1549          public string Data { get; set; }
1550        }
1551        public class EDB
1552        {
1553          private string mGroup = "testgroup";
1554          private string mType = "testtype";
1555          private string mOriginID = "testoriginid";
1556          private string mData = "testdata";
1557          [Index]
1558          public string Group { get{return mGroup;}}
1559          [Index]
1560          public string Type { get { return mType; } }
1561          [Index]
1562          public string OriginID { get { return mOriginID; } }
1563          [Index]
1564          public string Data { get { return mData; } }
1565        }
1566        public class example
1567        {
1568          public static void Main(string[] args)
1569          {
1570            my_EDB test3 = new my_EDB()
1571            {
1572              Group = "MYGROUPNAME",
1573              Type = "test1",
1574              OriginID = "ecfd2a5c-2585-4b7d-bc4f-8a33ce0a84f6",
1575              Data = "xxxxxxxx"
1576            };
1577            EDB test2 = new EDB();
1578            EDB test4= new EDB();
1579            Console.WriteLine("test:");
1580            Console.WriteLine(test3.Group);
1581            Console.WriteLine(test3.Type);
1582            Console.WriteLine(test3.OriginID);
1583            Console.WriteLine(test3.Data);
1584            Console.WriteLine("-----------------------");
1585            Console.WriteLine("test2:");
1586            Console.WriteLine(test2.Group);
1587            Console.WriteLine(test2.Type);
1588            Console.WriteLine(test2.OriginID);
1589            Console.WriteLine(test2.Data);
1590
1591             FieldInfo field = typeof(EDB).GetField("mGroup",
1592        System.Reflection.BindingFlags.Instance |
1593        System.Reflection.BindingFlags.NonPublic);
1594             field.SetValue(test2, test3.Group);
1595
1596
1597             Console.WriteLine("-----------------------");
1598             Console.WriteLine("test2 NEU:");
1599             Console.WriteLine(test2.Group);
1600             Console.WriteLine(test2.Type);
1601             Console.WriteLine(test2.OriginID);
1602             Console.WriteLine(test2.Data);
1603             Console.WriteLine("-----------------------");
1604             Console.WriteLine("test3:");
1605             Console.WriteLine(test4.Group);
1606             Console.WriteLine(test4.Type);
1607             Console.WriteLine(test4.OriginID);
1608             Console.WriteLine(test4.Data);
1609             MessageBox.Show("closewindow");
1610          }
1611        }
1612      }
```

for sure it would be pretty sloppy, but just as a workaround until grasshopper git new definitions?

DMYTRO BABLINYK Looks like it's possible http://dedjo.blogspot.com/2007/07/hack-read-only-properties-and-fields.html Just wondering if there is any elegant solution around. Of course ideal is to add set; to the properties.

SCHRIFT: Adobe Garamond Pro, Frutiger 75 Black, Courier New

PAPIER: Clairefontaine Trophée (FSC), Rose, 80g/m²

DRUCK: Techn. Universität Graz, Druck- und Kopierzentrum

BINDUNG: Buchbinderrei Erich Folkhard

GENDERKLAUSEL: Aus Gründen der besseren Lesbarkeit und Vereinfachung wurde in diesem Text auf eine korrekte sprachliche Gleichbehandlung der Geschlechter verzichtet und auf die generelle Verwendung des generischen Maskulinums zurückgegriffen.

STATISTIK: 51.161 Wörter, 369.235 Zeichen