Detlef Wingerath

XML_Anwendung: Schnittmuster für Mobiltelefone

apparently interacting daily used object need clothes.

CAAD

INDIVIDUAL THESIS:

Xml_Anwendung: Schnittmuster für Mobiltelefone

apparently interacting daily used object need clothes

NDS 2004

„Over time, cyberspace will hold a mirror image of our physical world, and the distinctions between Virtual Reality and every day consensus reality will begin to blur. Our living environments will be saturated with technology. Many objects will recognize us as we come near them. Things we now think of as inanimate will seem almost alive as they respond to our movements and even to our moods.“

Lawrence Hagerty, The Spirit of the Internet, Volume I:
Speculations on the Envolution of Global Consciousness

“Computation is intrinsically different from existing media because it is the only medium where the material and the process for shaping the material coexist in the same entity:numbers. The only other medium where a similar phenomenon occurs is pure thought. It naturally follows that computational media could eventually present the rare opportunity to express a conceptual art that is not polluted by textual or other visual representation.”

John Maeda, Design by Numbers

Die Verifizierung der These “apparently interacting daily used object need clothes” ist das Ziel der vorliegenden Arbeit. Die Definitionen der Begriffe “Ubiquitous Computing”, “Smart Objects”, “Smart Behavior”, Smart...bilden den Hintergrund für die angewendete Methode.
Im Sinne eines digitalen Arbeitsprozesses werden Schnittmuster, die für die Herstellung der Kleidungsstücke benötigt werden, in XML generiert, in XSLT transformiert und als Vektorgrafik ausgegeben. Die Schnittmustergrafiken werden anschließend über eine Druckerschnittstelle an einer Modellaserschneidmaschine “ausgeplottet”. Beispielhaft wird die Methode bei Mobiltelefonen angewendet. Dank der Firma “Max Gimmel AG” konnte Leder als Material verwendet werden.
Als Beweis des Konzeptes wird in einer kurzen Animation zwei smarten Objekten eine Persönlichkeit gegeben: http://edu.caad.hbt.arch.ethz.ch/nds2004/623
Da jede Persönlichkeit Kleider bedarf und Kleider erst Leute machen, werden in einem weiteren Schritt Muster vorgestellt, welche in VectorWorks generiert, in VectorScript programmiert worden sind, um wirklich smarten Objekten die Möglichkeit einer Gefühlsäußerung zu geben.
Ein Angebot dieser Dienstleistungen wäre mittels einer Internetseite mit angehängter Datenbank denkbar: Maßgeschneiderte Handy-Kleider mit unterschiedlichen Mustern je nach Gemütslage.

Zürich, den 27.09.2004

Inhalt

1. Einführung

Die Professur für CAAD des Departements Architektur der ETH Zürich wendet aktuelle Informationstechnologien in der architektonischen Praxis an. Das Interesse reicht von der Entwurfsunterstützung durch die digitalen Medien über die Produktion mit computergesteuerten Maschinen bis hin zum intelligenten Gebäudebetrieb.
Das Nachdiplomstudium CAAD wendet sich an diplomierte und praktizierende Architekten und Architektinnen sowie an Absolventen und Absolventinnen verwandter Studienrichtungen aus dem In- und Ausland. Der Ausbildungsschwerpunkt liegt auf dem computergestützen Entwurf (CAD) und seiner automatisierten Produktion (CAM).
Das Nachdiplomstudium ist ein Vollzeitstudium. Das Programm unterteilt sich in Modulen, die in seminaristischer Form durchgeführt werden. Das Studium endet mit einer Individuellen Thesis und einem Gruppenprojekt.

2. Definitionen

2.1 Ubiquitous Computing

Unter dem 1988 von Mark Weiser, Xerox-Forschungszentrum in Palo Alto, geprägten Begriff „Ubiquitous Computing“ wird die Allgegenwärtigkeit von Informationstechnik und Computerleistung verstanden. Vorstellbar sind zahllose kleinste, miteinander über Funk kommunizierende Mikroprozessoren, die unsichtbar in Dinge eingebaut werden können. Mit Sensoren ausgestattet, können diese kleinen Computer die Umwelt des Gegenstandes, in dem sie eingebaut sind, erfassen und diesen mit Informations-verarbeitungs- und Kommunikationsfähigkeiten ausstatten.
Die Vision des Ubiquitous Computing sieht vor, Alltagsdinge an das Internet anzuschließen und mit Sensorik auszustatten. Der berüchtigte Kühlschrank, der selbständig Milch nachbestellt, ist ein oft zitierter Vertreter einer solchen Welt. Drahtlos kommunizierende Prozessoren und Sensoren können aufgrund ihrer geringen Größe und ihres vernachlässigbaren Preises und Energiebedarfs bald in viele Gegenstände integriert werden.
Statt Experimente z.B. in einem Labor voller Instrumente durchzuführen, soll es dann möglich sein, die extrem miniaturisierten Beobachtungsinstrumente am Vorgang in der Natur selbst anzubringen. Der Technologietrend zeigt eindeutig in Richtung einer umfassenden Informatisierung der Welt. Durch lichtemittierende Polymere, Displays aus dünnen und hochflexiblen Plastikfolien, können zum Beispiel Windschutzscheiben, Preisschilder auf Warenregalen, aber auch Milchtüten oder Müslipackungen in dynamische Anzeigetafeln verwandelt werden.
Schon heute sind sich nur wenige darüber im Klaren, dass beim Surfen

Abb. 1

im Internet nicht nur jeder Einkauf, sondern im Allgemeinen auch alle Mausklicks protokolliert werden und potentiell über ihre Vorlieben Auskunft geben können. Sollten sich smarte Umgebungen und schlaue Alltagsgegenstände durchsetzen, wäre mit dem Ausschalten des Pc`s keineswegs auch die elektronische Datensammlung beendet: Smarte Möbel und Kleidungsstücke würden fast immer aktiv sein und selbst in den eigenen vier Wänden genau wahrnehmen können, was man gerade tut, eine smarte Armbanduhr würde ständig die aktuelle Position des Benutzers ermitteln und weitermelden, um so ortsbezogene Dienste z.B. die lokale Wettervorhersage oder die Anzeige des Rückwegs zum Hotel, nutzen zu können.
Was ist in dem Zusammenhang Datenschutz und Privatsphäre? Die Privatsphäre eines Individuums ist wohl der Anspruch den Prozess der Informationssammlung, -speicherung, -verarbeitung, und –weitergabe betreffend seiner persönlichen Daten zu kontrollieren. Wer aber überprüft die Richtigkeit von Aussagen smarter Objekte? Wieviel Handlungsautonomie gibt man smarten Objekten? Stelle man sich nicht nur einen Kopierer vor, der in eigener Verantwortung Papier nachbestellt,"sondern auch zum Beispiel eine Barbie-Puppen, die sich programmgesteuert über eine kabellose Internetverbindung neue Kleider von ihrem „Taschengeld" kauft..."[1]

2.2 Mooresche Gesetz

Der stetige Fortschritt der Mikroelektronik beschreibt das Mooresche Gesetz. Mit erstaunlicher Präzision und Konstanz gilt das bereits 1965 von Gordon Moore aufgestellte Gesetz: etwa alle 18 Monate verdoppelt sich die Verarbeitungsgeschwindigkeit von Computern, und ähnlich hohe Effizienzsteigerung sind bei der Speicherkapazität und im Bereich der Vernetzung zu beobachten.
Dieser Trend führt dazu, dass elektronische Komponenten in Zukunft noch wesentlich leistungsfähiger, kleiner und billiger werden, womit Computerleistung bald im Überfluß vorhanden sein dürfte. „Die kurz vor der Masseneinführung stehenden „smart labels" sind erste Hinweise auf die zu erwartenden Myriaden von „Wegwerfcomputern"."[2]

„It`s been 18 months and my computer`s power hasn`t doubled!"

Abb. 2

2.3 Smart Labels

Bei „smart labels“ oder „RFID tags“ (Radio Frequency Identification) handelt es sich um papierdünne, quadratmillimetergroße Chips ohne eigene Energiequelle, welche mit einem Hochfrequenz-Signal bestrahlt werden, diesen decodieren, aus ihm Energie beziehen und dann eine Antwort z.B. einen Produktcode zurückfunken. Sie funktionieren so ähnlich wie die bekannten Diebstahlsicherungen und Türschleusen von Kaufhäusern, allerdings geht es hier nicht mehr um die Information `bezahlt/gestohlen`, sondern es können auf eine Distanz von einigen Metern und innerhalb von Millisekunden Hunderte von Zeichen gespeichert bzw. ausgelesen werden.
Mit `smart labels`, die auf Paletten und Produktverpackungen angebracht sind, kann eine lückenlose Verfolgung der Warenströme über die gesamte Lieferkette hinweg sichergestellt werden, indem Lesestationen an Laderampen und Hochregallagern den Zustand und Ort von Gütern ohne menschliche Intervention erkennen und direkt dieses Wissen in betriebliche Informationssysteme speisen. Sie benötigen keine Sichtverbindung zur Lesestation.

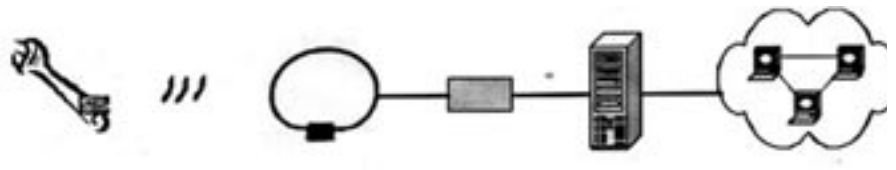


Abb. 3

2.4 Smart Objects

Sind „Smart Objects“ eine eingetragene Handelsmarke?[3] Sind sie intelligente Dinge, interaktive Dinge? Ist das Betätigen eines Lichtschalters, welcher das Leuchten einer Lampe schaltet, eine Interaktion? Ist es dann eine Interaktion, wenn eine Rückmeldung bestätigt, dass die Lampe leuchtet?
Dinge werden als smart bezeichned, weil sie mit Technologien zur automatischen Identifikation, zur Lokalisierung bzw. mit Sensortechnologie ausgestattet sind. Es gibt „smarte“ Krankenkassen-Karten, Lokalisierungssysteme für gestohlene Autos und entlaufende Haustiere sowie Mobiltelefone mit integriertem Notizbuch, Kamera, Terminplaner und MP3Player.
Auf der CES (International Consumers Electronic Show in Las Vegas) stellte Microsofts Chairman Bill Gates die brandneue „Smart Personal Object Technologie „ SPOT und erste Geräte vor, welche den Alltag des Menschen nun neben den PC’s vollends erobern möchten. Das Anhängsel „Smart“ der Technologie zeigt, dass ihre bekannten Alltagsgeräte nun langsam ihren eigenen Willen bzw. ihre „Intelligenz“ bekommen. So kommen Wecker und Armbänder mit kleinen Prozessoren auf den deutschen Markt, welche ihnen Staumeldungen, Termine oder die Wettervorhersage direkt aufs Handgelenk bringen.

Abb. 4

2.5 Smart Behavior

Beim smarten Verhalten von Virtuellen Objekten bzw. Agenten sind Vorbilder in der Tier- und Pflanzenwelt gefunden worden. Mit einigen wenigen Regeln, wie der konstante Abstand zum Vordermann und eine gleiche Ausrichtung, kann ein Schwarmverhalten von Fischen oder Vögeln simuliert werden.
Desweiteren können digitale Hormone bzw. Pheremone zur Selbstorganisation, zu einem selbständigen Verhalten eingesetzt werden. Ein virtueller Agent, der mit einem digitalen Farbsensor ausgestattet ist, kann so programiert werden, dass er z.B. seine Farbe verändert, wenn sein Sensor die Farbe Rot erfasst.
Im Bildbeispiel laufen programmierte Agenten zufällig gesteuert über eine Bildumgebung. Sie überprüfen die Größe des Environments. Sie erkennen Farbwerte, welche Gebäudehöhen representieren, Verkehrswege für Fussgänger, Straßenbahn und Autos, Wasser- und Grünflächen. Wenn sie ein Gebäude erkennen, springen sie zufällig gesteuert zu einer anderen Position. Die Agenten überprüfen weiterführend die Umgebung, indem sie ein Quadrat von 15 x 15 Pixel lesen, und wenn alle vier Eckpunkte des Quadrats den Farbwert Grün haben, bewegen sie sich in die Mitte des Quadrats und beenden die Überprüfung. Da ein Agent mindestens 15 x 15 Pixel benötigt, um sich zu stoppen, ist er ein Identikator für ausgedehnte Grünflächen.

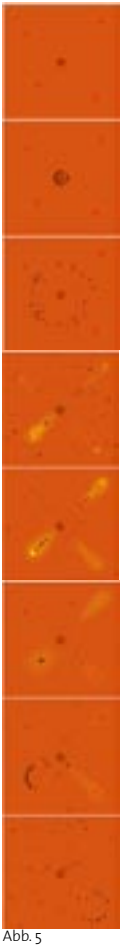


Abb. 5

Ameisen sind smart. Sie entfernen sich für die Futtersuche vom Bau.Wenn sie Futter finden und dieses zurück in den Bau tragen, markieren sie den Weg vom Bau zum Futter mit Pheremonen, so daß andere Ameisen schneller das Futter finden und es ebenfalls zum Bau transportieren können. Je gößer die Pheremonenwolke, um so mehr Ameisen werden angezogen, falls die Futterquelle abgetragen ist, beginnt die Futtersuche von vorne.

2.6 Smart via Datenbank

Der berüchtigte Datenschatten ist eine Datenbank, welche auf einem Webserver liegt. Über ein Netzwerk lassen sich die Daten dieser Datenbank austauschen. Mit Datenbanken kann Dingen ein Wissen zur Verfügung gestellt werden. Mittels logisch verknüpfter Tabellen, in denen die notwendigen Daten enthalten bzw. gespeichert werden können, ist eine Interaktion über eine Schnittstelle mit dem Objekt möglich. Im folgenden Beispiel ist die Datenbank mit SQL – Structured Query Language realisiert worden. Als Datenbanksystem wurde PostgreSQL[4] benutzt. Das Verwaltungswerkzeug war phpPgAdmin.

Abb. 6

In der Tabelle “statues” befinden sich alle wichtigen Daten einiger Skulpturen, besonders wichtig ein Link zu den einzelnen Bildern. Die Bilder liegen auf einem Server und erlauben so, die Datenmenge in der Datenbank gering zu halten.
In der Tabelle “boxofbricks” werden alle Skulpturvarianten gespeichert, Anwender der InternetSeite „my loveliest“ können über eine Flash-Benutzeroberfläche Köpfe, Beine und Rumpf von Skulpturen variieren und sich ihre Lieblingsskulptur neu zusammensetzen. Mit der Möglichkeit dieser Kreation einen Namen zu geben, kann diese gespeichert und wieder abgerufen werden. Beim Aussuchen werden einem die Informationen automatisch aus der Datenbank geliefert. Im Bildbeispiel sind es die wesentlichen Daten der Skulpturen wie Entstehungsjahr, Maße, Material.

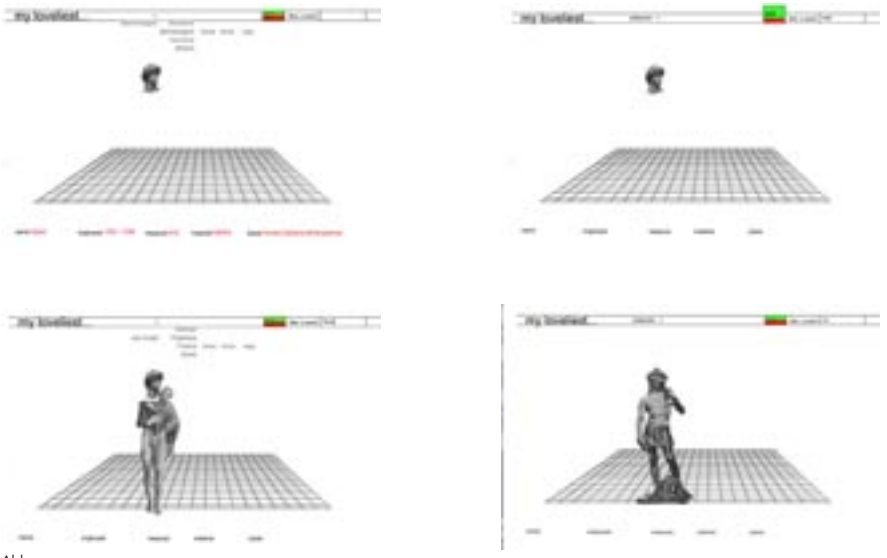


Abb. 7

3. Idea

3.1 Lehrstuhl für CAAD

Das Interesse des Lehrstuhls reicht von der Entwurfsunterstützung durch die digitalen Medien über die Produktion mit computergesteuerten Maschinen bis hin zum intelligenten Gebäudebetrieb. Das Experiment mit multimedialen Darstellungsmöglickeiten ist ein Schwerpunkt der Lehre der Professur für CAAD.
In meiner Diplomarbeit beschäftige ich mich mit Regelsystemen, welche ich beispielhaft in eine computerbasierte Beschreibungsform übersetze. Wobei das humorvolle Experiment im Mittelpunkt steht.
Häufig aufgegriffene Regelsysteme sind die klassischen Säulenordnungen. In Form von Architekturtraktaten sind immer wieder die antiken Architekturvorbilder analysiert, ihre Elemente systematisiert und katalogisiert worden.
Diese Systematisierung findet in computergenerierten Verfahren eine zeitgemäße Entsprechung. Im Rahmen des Diplomwahlfaches “replay atlas 03/04” unter der Leitung von Katharina Bosch, Markus Braach und Susanne Schumacher wurde der Versuch einer Übersetzung unternommen. Ausgehend von der Analyse historischer Traktate konnte eine computerbasierte Beschreibungsform für Säulenordnungen entwickelt werden. Die Konstruktionsanweisungen aus den Quellen wurden in XML (eXtensible Markup Language) übersetzt, anschließend in das grafische Format SVG (Scalable Vector Grafics) transformiert und zu einem Säulenatlas zusammengeführt. Das Buch zeigt die klassische dorische und ionische Säulenordnung von zehn wichtigen Autoren der Architekturtheorie: Vitruv, Alberti, Serlio, Blum, Vignola, Palladio, Perrault, Durand, Stuart und Revett, Chitham.

Abb. 8

3.2 Xml-Anwendungsbeispiel

Die Idee, Dingen und deren Elemente in Regeln zu fassen und systematisch zu beschreiben, ist so alt wie die wissenschaftliche, bzw. kunsthistorische Betrachtung der Objekte selbst. Gerade unsere zweite Haut unser Kleidungsstück ist in Regeln erfaßt und systematisiert. Smarte Objekte benötigen Kleider, ist eine humorvolle Behauptung, die die Ausgangsfrage meiner Thesis bilded.
Smarte Objekte benötigen nicht nur Kleider, sondern Kleidungsstücke, welche sie sich selber aussuchen, bestellen und kaufen können.
Ein erster Schritt der Verifizierung ist die Bereitstellung von Schnittmuster für smarte Objekte in einer Sprache, die sie verstehen: XML.
Durch XML werden Web-Informationen Maschinenlesbar.

3.1.2 XML

XML heißt eXtensible Markup Language und liefert die Regeln, die beim Definieren von Dokumenttypen angewendet werden. Diese Sprache ist durch eine einfache Syntax definiert, in der sich Daten als Text mit lesbaren

Abb. 9

„Tags“ markieren lassen. XML ist eine für andere Programme lesbare Sprache. XML Dokumente lassen sich umformen, sie sind transformierbar. XML bezieht sich im Wesentlichen auf die Struktur, also auf die Beschreibung eines Gegenstandes. Die Grundidee von XML ist, dass XML-Dokumente, die alle in ihrem Aufbau gewissen Grundmustern folgen, durch andere Programme automatisiert werden können. Wenn diese Grundmuster eingehalten werden, ist es möglich, Programme zu schreiben, die die Dokumente automatisch verarbeiten. Eine solche Anwendung ist XSLT.

3.1.3 XSLT

XSLT heißt eXtensible Stylesheet Language Transformations. XSLT ist eine XML-Anwendung. Diese legt Umformungen fest, nach denen ein XML Dokument in ein anderes XML Dokument transformiert wird. Ein XSLT Dokument ist als eine Layout-Datei zu verstehen, welche Schablonen enthält. Ein XSLT Prozessor vergleicht diese Vorlagen mit den Elementen in dem zugrundegelegtem XML Dokument und erzeugt daraus ein neues Dokument.
Mit einer XSL-Transformation läßt sich ein XML Dokument in eine SVG Datei umwandeln.

3.1.4 SVG

SVG bedeutet Scalable Vector Graphics. SVG ist eine Sprache zur Beschreibung zweidimensionaler Vektorgrafiken. SVG-Dateien können wie XML und XSLT Dateien mit einem einfachen Texteditor erstellt und bearbeitet werden. SVG Dateien können durch Web-Browser oder mittels Zeichenprogramme dargestellt werden. Bemerkenswert ist die Durchgängigkeit der Spache, denn sowohl die Grunddaten in XML, als auch die Transformationsanweisungen in XSLT und die Grafikbeschreibungen in SVG sind die Syntax von XML. Mit der Trennung von Inhalt und Form kann ein Inhalt verschiedene Formen bestimmen. Ein XML Dokument beschreibt abstrakt den Inhalt und die Struktur, während XSLT Dateien die Formen erzeugen, welche dann via SVG sichtbar werden.

3.1.5 Schnittmuster für Mobiltelefone

Das „interacting daily used object“ ist derzeit ein mobiles Telefon (cellphone, mobile, Handy, Natel...). Mobiltelefone sind mit integriertem Notizbuch, Kamera, Terminplaner und MP3Player erhältlich.
Für eine erste Beschreibung in XML generierte ich ein einfaches T-Shirt - Schnittmuster.

Abb. 10

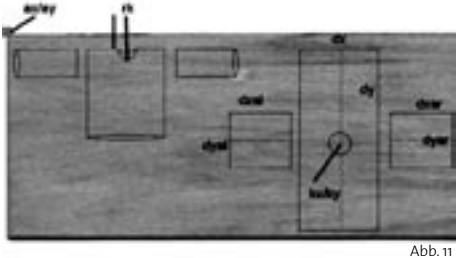


Abb. 11

Der Ursprung des virtuellen Zeichenpapieres wird durch „ax“ und „ay“ bestimmt. Im Style-Sheet wird die Größe des Papieres festgelegt, width, height, viewbox. Die Werte „dx“ und „dy“ stehen für die Länge und Breite des T-Shirts. Mit den Werten „dxal“ und „dyal“ läßt sich die Größe des linken Ärmels verändern, ebenso mit „dxar“ und „dyar“ den rechten Ämel. Mit „kx“ und „ky“ wird der Ursprung des Ausschnitts bestimmt mit „rk“ dessen Größe. Im folgenden Style-Sheet werden bestimmte Regeln angewandt. Mit einfachen mathematischen Aussagen, die sich hinter den Befehlen „rect“, „line“ und „circle“ verbergen können so leicht Varianten eines T-Shirts erzeugt werden.

XML_Regeln für ein T-Shirt:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<sewingpatternXML>
                    <element type="hauptteil"
ax="250.000"
ay="50.0000"
dx="200.0000"
dy="500.0000"
dxal="150.0000"
dyal="120.0000"
dxar="150.0000"
dyar="120.0000"
/>
    <kreis type="ausschnitt"
kx="350.000"
ky="300.0000"
radius="40.000"
 />
    <linie type="faltlinie"
a1="350.000"
a2="50.0000"
b1="350.000"
b2="260.000"
/>
 <linie type="faltlinie"
a1="350.000"
a2="340.0000"
b1="350.000"
b2="550.000"
/>
  <linie type="faltlinie"
a1="90.000"
a2="300.0000"
b1="240.000"
b2="300.000"
/>
   <linie type="faltlinie"
a1="460.000"
a2="300.0000"
b1="610.000"
b2="300.000"
/>
</sewingpatternXML>
```

Style-Sheet_XSLT:

```
<?xml version=”1.0”?>

<xsl:stylesheet version=”1.0” xmlns:xsl=”http://www.w3.org/1999/XSL/Transform”>


<xsl:template match=”sewingpatternXML”>

<svg width=”29.7cm” height=”21cm” viewBox=”0 0 1200 400”

    xmlns=”http://www.w3.org/2000/svg” version=”1.1”>

                <xsl:apply-templates select=”element”/>
  <xsl:apply-templates select=”kreis”/>
  <xsl:apply-templates select=”linie”/>

</svg>

</xsl:template>


<xsl:template match=”element”>

                <xsl:element name=”rect” namespace=”http://www.w3.org/2000/svg”>

                                <xsl:attribute name=”x”>

                                                <xsl:value-of select=”@ax”/>

                                </xsl:attribute>

                                <xsl:attribute name=”y”>

                                                <xsl:value-of select=”@ay”/>

                                </xsl:attribute>

                                <xsl:attribute name=”width”>

                                                <xsl:value-of select=”@dx”/>

                                </xsl:attribute>

                                <xsl:attribute name=”height”>
```

```
                    <xsl:value-of select="@dy"/>
                </xsl:attribute>
                <xsl:attribute name="fill">none</xsl:attribute>
                <xsl:attribute name="stroke">black</xsl:attribute>
                <xsl:attribute name="stroke-width">3</xsl:attribute>
            </xsl:element>
<xsl:element name="rect" namespace="http://www.w3.org/2000/svg">
                <xsl:attribute name="x">
                    <xsl:value-of select="(@ax)-(@dxal)-10"/>
                </xsl:attribute>
                <xsl:attribute name="y">
                    <xsl:value-of select="@dyal*(-0.5)+@ay+@dy*0.5"/>
                </xsl:attribute>
                <xsl:attribute name="width">
                    <xsl:value-of select="@dxal"/>
                </xsl:attribute>
                <xsl:attribute name="height">
                    <xsl:value-of select="@dyal"/>
                </xsl:attribute>
                <xsl:attribute name="fill">none</xsl:attribute>
                <xsl:attribute name="stroke">black</xsl:attribute>

                <xsl:attribute name="stroke-width">3</xsl:attribute>
```

```
		</xsl:element>
<xsl:element name="rect" namespace="http://www.w3.org/2000/svg">
				<xsl:attribute name="x">
					<xsl:value-of select="((@ax)+(@dx))+10"/>
				</xsl:attribute>
				<xsl:attribute name="y">
					<xsl:value-of select="@dyar*(-0.5)+@ay+@dy*0.5"/>
				</xsl:attribute>
				<xsl:attribute name="width">
					<xsl:value-of select="@dxar"/>
				</xsl:attribute>
				<xsl:attribute name="height">
					<xsl:value-of select="@dyar"/>
				</xsl:attribute>
				<xsl:attribute name="fill">none</xsl:attribute>
				<xsl:attribute name="stroke">black</xsl:attribute>
				<xsl:attribute name="stroke-width">3</xsl:attribute>
		</xsl:element>
</xsl:template>
<xsl:template match="kreis">
		<xsl:element name="circle" namespace="http://www.w3.org/2000/svg">
```

```
    <xsl:attribute name="cx">

                                        <xsl:value-of select="@kx"/>

                                </xsl:attribute>
    <xsl:attribute name="cy">

                                        <xsl:value-of select="@ky"/>

                                </xsl:attribute>
    <xsl:attribute name="r">

                                        <xsl:value-of select="@radius"/>

                                </xsl:attribute>
        <xsl:attribute name="fill">none</xsl:attribute>

                                <xsl:attribute name="stroke">black</xsl:attribute>

                                <xsl:attribute name="stroke-width">3</xsl:attribute>

                </xsl:element>

</xsl:template>


<xsl:template match="linie">

                <xsl:element name="line" namespace="http://www.w3.org/2000/svg">

    <xsl:attribute name="x1">

                                        <xsl:value-of select="@a1"/>

                                </xsl:attribute>
    <xsl:attribute name="y1">

                                        <xsl:value-of select="@a2"/>

                                </xsl:attribute>
    <xsl:attribute name="x2">

                                        <xsl:value-of select="@b1"/>

                                </xsl:attribute>
```

```
<xsl:attribute name="y2">
                                        <xsl:value-of select="@b2"/>
                                </xsl:attribute>
    <xsl:attribute name="fill">none</xsl:attribute>
                                <xsl:attribute name="stroke">blue</xsl:attribute>
                                <xsl:attribute name="stroke-width">1</xsl:attribute>
                </xsl:element>
</xsl:template>
  </xsl:stylesheet>
```

Mit dem XSLT - Dokument werden die Punkte der Vektoren ermittelt, die sich dann via SVG darstellen lassen. Auf der nächsten Seite einige Varianten der ersten Beschreibung eines Bekleidungsstücks für Smarte Objekte. Im Folgenden der PunkteCode in SVG:

<?xml version="1.0" encoding="UTF-8"?><svg xmlns="http://www.w3.org/2000/svg" width="29.7cm" height="21cm" viewBox="0 0 1200 400" version="1.1"><ns_1:rect xmlns="" xmlns:ns_1="http://www.w3.org/2000/svg" x="230.000" y="10.0000" width="200.0000" height="500.0000" fill="none" stroke="black" stroke-width="3"/><ns_2:line xmlns:ns_2="http://www.w3.org/2000/svg" x1="350.000" y1="300.0000" x2="430" y2="300.0000" fill="none" stroke="blue" stroke-width="1"/><ns_3:line xmlns:ns_3="http://www.w3.org/2000/svg" x1="350.000" y1="300.0000" x2="230.000" y2="300.0000" fill="none" stroke="blue" stroke-width="1"/><ns_4:circle xmlns:ns_4="http://www.w3.org/2000/svg" cx="350.000" cy="300.0000" r="40.000" fill="none" stroke="black" stroke-width="3"/></svg>

Abb. 12

Vektor-Grafik Varianten:

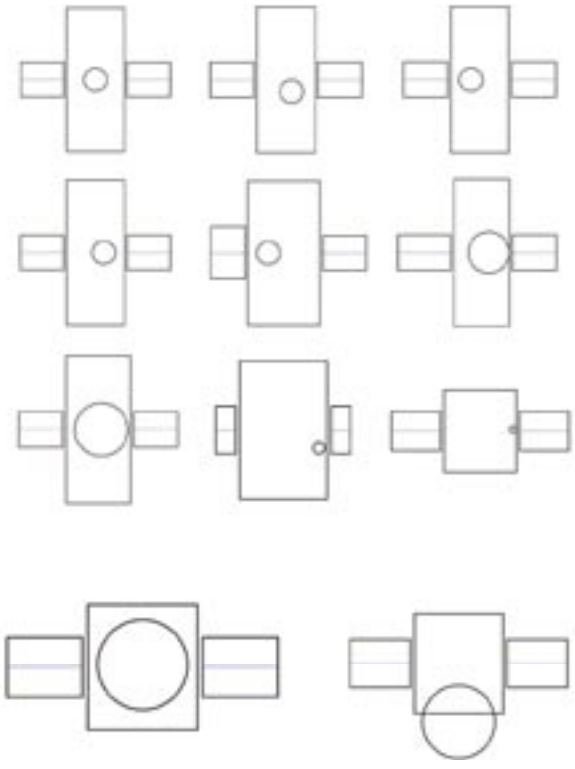
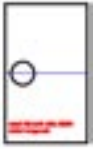


Abb. 13

3.1.6 Ponchoschnittmuster

Da Mobiletelefone keine Arme haben, sind Poncho`s das Kleidungstück für Mobiltelefone. In dieser Diplomarbeit beschränke ich die Auswahl auf sechs produzierte Poncho-Varianten. Die Schnittmuster können aber mittels Variabeln im Xml-Dokument individuell bestimmt werden. Neben der Ponchogröße, die durch die Höhe, Breite und Tiefe des smarten Objekts vorgeben wird, kann der Ausschnitt so variiert werden, passend zur Antenne, rechts, links oder mittig. Falls keine sichtbare Antenne vorhanden ist, ist der ovale, mittige Ausschnitt geeignet, um ein Mobiltelefon zu kleiden.

Abb. 14



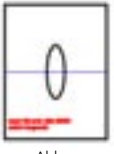


Abb. 15

Xml, Xslt und Svg-Code eines Ponchoschnittmusters

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<sewingpatternXML>
                <element type="hauptteil"
ax="100.000"
ay="10.0000"
dx="70.0000"
dy="70.0000"
akx="142.000"
aky="52.0000"
dkx="12.0000"
dky="12.0000"
/>
</sewingpatternXML>
```

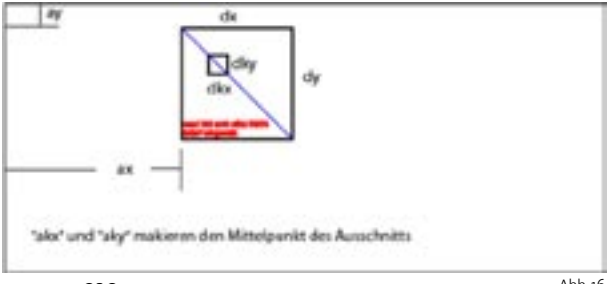


Abb. 16

```
<?xml version="1.0"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">


<xsl:template match="sewingpatternXML">

<svg width="29.7cm" height="21cm" viewBox="0 0 700 300"

    xmlns="http://www.w3.org/2000/svg" version="1.1">
   <text>
   <tspan x="100" y="72" style="font-family:Arial;
   font-size:5; fill:red; stroke:red;y">caad hbt arch ethz 03/04
   </tspan>
   </text>
   <text>
   <tspan x="100" y="78" style="font-family:Arial;
   font-size:5; fill:red; stroke:red;y">detlef wingerath
   </tspan>
   </text>

                <xsl:apply-templates select="element"/>

</svg>

</xsl:template>


<xsl:template match="element">

                <xsl:element name="rect" namespace="http://www.w3.org/2000/svg">

                                <xsl:attribute name="x">

                                                <xsl:value-of select="@ax"/>

                                </xsl:attribute>

                                <xsl:attribute name="y">

                                                <xsl:value-of select="@ay"/>

                                </xsl:attribute>
```

```
                    <xsl:attribute name="width">
                        <xsl:value-of select="@dx"/>
                    </xsl:attribute>
                    <xsl:attribute name="height">
                        <xsl:value-of select="@dy"/>
                    </xsl:attribute>
                    <xsl:attribute name="fill">none</xsl:attribute>
                    <xsl:attribute name="stroke">black</xsl:attribute>
                    <xsl:attribute name="stroke-width">2</xsl:attribute>
                </xsl:element>
<xsl:element name="rect" namespace="http://www.w3.org/2000/svg">
                    <xsl:attribute name="x">
                        <xsl:value-of select="@akx"/>
                    </xsl:attribute>
                    <xsl:attribute name="y">
                        <xsl:value-of select="@aky"/>
                    </xsl:attribute>
                    <xsl:attribute name="width">
                        <xsl:value-of select="@dkx"/>
                    </xsl:attribute>
                    <xsl:attribute name="height">
                        <xsl:value-of select="@dky"/>
                    </xsl:attribute>
                    <xsl:attribute name="fill">none</xsl:attribute>
```

```
                    <xsl:attribute name="stroke">black</xsl:attribute>

                    <xsl:attribute name="stroke-width">2</xsl:attribute>

            </xsl:element>

<xsl:element name="line" namespace="http://www.w3.org/2000/svg">

  <xsl:attribute name="x1">

                        <xsl:value-of select="@ax"/>

                    </xsl:attribute>
  <xsl:attribute name="y1">

                        <xsl:value-of select="@ay"/>

                    </xsl:attribute>
  <xsl:attribute name="x2">

                        <xsl:value-of select="@ax+@dx"/>

                    </xsl:attribute>
  <xsl:attribute name="y2">

                        <xsl:value-of select="@ay+@dy"/>

                    </xsl:attribute>
    <xsl:attribute name="fill">none</xsl:attribute>

                    <xsl:attribute name="stroke">blue</xsl:attribute>

                    <xsl:attribute name="stroke-width">1</xsl:attribute>

            </xsl:element>


 </xsl:template>
</xsl:stylesheet>
```

<?xml version="1.0" encoding="UTF-8"?><svg xmlns="http://www.w3.org/2000/svg" width="29.7cm" height="21cm" viewBox="0 0 1200 400" version="1.1"><ns_1:rect xmlns="" xmlns:ns_1="http://www.w3.org/2000/svg" x="230.000" y="10.0000" width="200.0000" height="500.0000" fill="none" stroke="black" stroke-width="3"/><ns_2:line xmlns:ns_2="http://www.w3.org/2000/svg" x1="350.000" y1="300.0000" x2="430" y2="300.0000" fill="none" stroke="blue" stroke-width="1"/><ns_3:line xmlns:ns_3="http://www.w3.org/2000/svg" x1="350.000" y1="300.0000" x2="230.000" y2="300.0000" fill="none" stroke="blue" stroke-width="1"/><ns_4:circle xmlns:ns_4="http://www.w3.org/2000/svg" cx="350.000" cy="300.0000" r="40.000" fill="none" stroke="black" stroke-width="3"/></svg>

4. Produktion

4.1 CAD/CAM und CNC Technik

CAD, Abk. für engl. Computer Aided Design, rechnergestütztes Konstruieren, die Konstruktion techn. Erzeugnisse mit Hilfe eines Rechners unter der Voraussetzung, daß sich beschreibbare Vorgänge wiederholen. Bei der Konstruktion von Varianten kann eine Grundausführung mit Hilfe des Rechners variiert werden, z.B. ein Getriebe mit baukastenförmigen Elementen (Wellen, Zahnräder, Lager). Der Rechner führt selbständig alle Routinearbeiten durch, die der Konstrukteur sonst ausführen müßte.
CNC, Abk. für engl. Computer Numerically Control, eine Werkzeugmaschinensteuerung, bei der die Funktionen nicht festgelegt sind, sondern an der Maschine durch den Bedienenden mit einem Rechner frei programmiert werden können. Im Gegensatz zu DNC ist jedoch nicht eine Vielzahl von Programmen, sondern nur das jeweilige Programm verfügbar. CNC soll insbes. kleineren Betrieben die Einführung numerischer Steuerungen erleichtern.
numerische Steuerung, NC- Steuerung, eine Steuerung von Werkzeugmaschinen, bei der die Weginformation (z.B. Schlittenwege) nicht analog (durch Steuerkurven, Nockenschienen...), sondern digital, d.h. in Form von Zahlen (numerisch), eingegeben werden. Der Weg, den z.B. der Schlitten (mit dem Drehmeißel) einer Drehmaschine zurücklegen muß, um eine bestimmte Werkstücklänge zu drehen, wird als Zahl eingegeben und nicht als Abstand zwischen zwei Schaltnocken. Neben den Weginformationen werden auch die Schaltinformationen in Zahlenform eingegeben: Drehzahlen und Vorschübe, Werkzeugwahl. Man unterscheidet Punktsteuerung, Streckensteuerung und Bahnsteuerung. Als Datenspeicher dienen Lochstreifen, Magnetbänder, Halbleiterspeicher. Als Werkstückprogrammierung bezeichnet man

die Erstellung der von Werkstückform und -werkstoff abhängigen Steuerdaten. Sie kann von Hand oder mittels Datenverarbeitungsanlagen erfolgen, wobei besondere Programmiersprachen benutzt werden. Numerische Steuerungen ergeben gegenüber anderen Steuerungen sehr kurze Rüstzeiten an den Werkzeugmaschinen selbst, so daß auch wenige oder einzelne Werkstücke automatisch bearbeitet werden können. (Berteslmann-Lexikon, Gütersloh 1987)
Als Speichermedien sind Lochstreifen und Magnetbänder inzwischen Geschichte: auch vor 17 Jahren mögen sie schon etwas anachronistisch gewesen sein. Grundlegendes hat sich aber nicht verändert. Heutige CNC-Maschinen sind Bedienerfreundlicher geworden, sie können größere Datensätze verarbeiten, man muß keine Programiersprache lernen, sondern man nutzt eine Software bzw. man installiert einen "Treiber", wie beim Tintenstrahldrucker.
Die folgenden CNC- Maschinen sind für das Projekt am Lehrstuhl verfügbar:
eine 3 achsige Fräse (Produktionstisch 2.40m x 1.55m), eine Industrielaserschneidmasc hine (Produktionstisch 3.00m x 2.00m), ein 3D- Gipsdrucker (Produktionsgröße 0.20m x 0.20m x 0.25m) und eine Modellaserschneidmaschine (Produktionstisch 0.45m x 0.81m).
Mit diesem Modellbaulaser können schnell, filigran und präzise flächige, dünne Materialien geschnitten und graviert werden. Nahezu jedes Material lässt sich verarbeiten. Die Maschine funktioniert wie ein Tintenstrahldrucker.

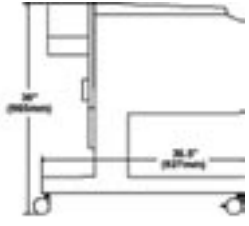


Abb. 17

Abb. 18

"Anfang der neunziger Jahre verglichen William H. Davidow und Michael S. Malone in ihrem Buch über „Das virtuelle Unternehmen" die Tragweite der bevorstehenden „Kulturrevolution" mit dem Einschnitt, den die industriellen Revolution vor rund 150 Jahren verursacht hatte: „Die industrielle Revolution zerriß das Gewebe der Gesellschaft in kleinste Teile und setzte sie zu einem völlig neuen Muster wieder zusammen."[5] Die Strukturen des täglichen Lebens änderten sich für immer; und mit ihnen das Wesen der Familien, der Regierungen, der Städte, der Bauernhöfe, der Sprache, der Kunst, ja selbst unseres Zeitbegriffs...Dagegen gewinnt unsere Generation heute ihre dynamische Kraft aus der Informationsverarbeitung. Die Entwicklung der digitalen Informations- und Kommunikations Technologie und computergesteuerter Werkzeuge wie CNC-Fräse und Biegemaschine, Laser- und Wasserstrahlschneider bis hin zu den Technologien wie 3D-Printing stellt die Gestalter heute vor Aufgaben, die in vielerlei Hinsicht mit der aus der Industriealisierung erwachsenden Herausforderungen vergleichbar ist. Seit der ersten industriellen Revolution bis zum Auftauchen der digitalen Werkzeuge gab es im Prinzip nur die Alternative zwischen einer konkurrenzlos günstigen industriellen Massenfertigung und der sich zunehmend verteuernden handwerklichen Einzelstück- und Kleinserienherstellung. Mit der Verbreitung digitaler Produktionstechnologien verändern sich die Produktionsbedingungen. Digitale Herstellungsprogramme und computergestützte Werkzeuge erlauben den Einstieg in die Produktion mit Kleinserien zu günstigen Preisen"[6] - computer aided manufactoring (CAM).

4.2 Das Produkt

Material

Abb. 19

Prozess

Abb. 20

Abb. 21

erster Prototyp

Baumwollstoff, 7 x 13 cm

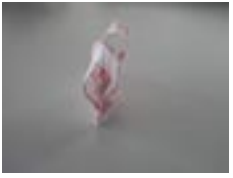


Abb. 22

Model ETH-Zürich

Abb. 23

Storyboard

Abb. 24

Set

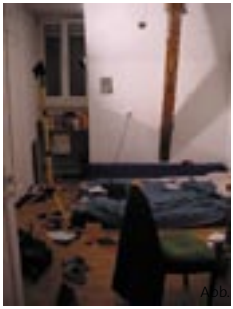


Abb. 25

5. Ausblick - VektorScript

Als Erweiterung der Thesis ist ein Flash- Interface mit anghängter Datenbank denkbar. Falls man eine Vermarktung anstrebt, müsste man nur auf einer Internet-Seite die Maße seines Mobiltelefons eingeben und sich ein Kleidungsmodel aussuchen, Name eingeben, bezahlen, speichern - und über die Datenbank wird das maßgenaue Schnittmuster ermittelt und kann an ein entsprechendes Ausgabegerät weitergeleitet werden z.B. eine CNC- Webmaschine.
Mit der im ersten Kapitel angedeuteten Persönlichkeitserlangung von Dingen könnten Objekte über eine drahtlose Internetverbindung ein Verhalten vortäuschen bzw. leben, welches an eine Handlung eines Subjektes erinnert, z.B Kleidungsstücke aussuchen und kaufen. Weiterhin wären kompliziertere Schnittmuster denkbar. Auch ist eine Ausweitung auf andere Objekte möglich. So könnten z.B. Abdeckplanen für Automobile maßgeschneidert werden. Die Wahl individueller Muster wäre durch die Generierung eines Stoffmusters gegeben.
Vielleicht sind Automobile und Mobiltelefone bald auch smart genug um eine Art Gefühlsleben zu haben und dies äußert sich dann über ein sich veränderndes Muster. Die folgenden Zeichnungen sind in VectorWorks generiert, in VectorScript programmiert und könnten als Mustervorlagen für Gemütslagen smarter Objekte dienen.
Die Programmiersprache, in der in VectorWorks Scripts programmiert werden, heißt VectorScript. VectorScript ist eine an Pascal angelehnte Sprache, die jedoch nicht über alle Befehle eines echten Pascals (wie z.B. Think Pascal oder Turbo Pascal) verfügt. Eine ausführliche Beschreibung sämtlicher Funktionen und Prozeduren, zur Erstellung von intelligenten Objekten gibt es in der VectorScript-Online-Hilfe. Dort befindet sich auch eine kurz gehaltene Einführung in die Programmiersprache Pascal als VectorScript PDF Handbuch.

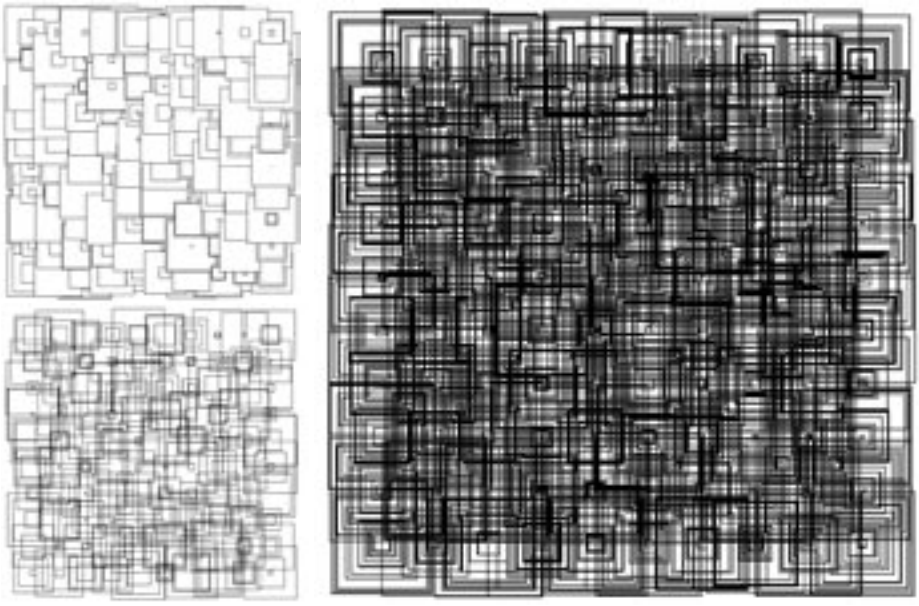


Abb. 26

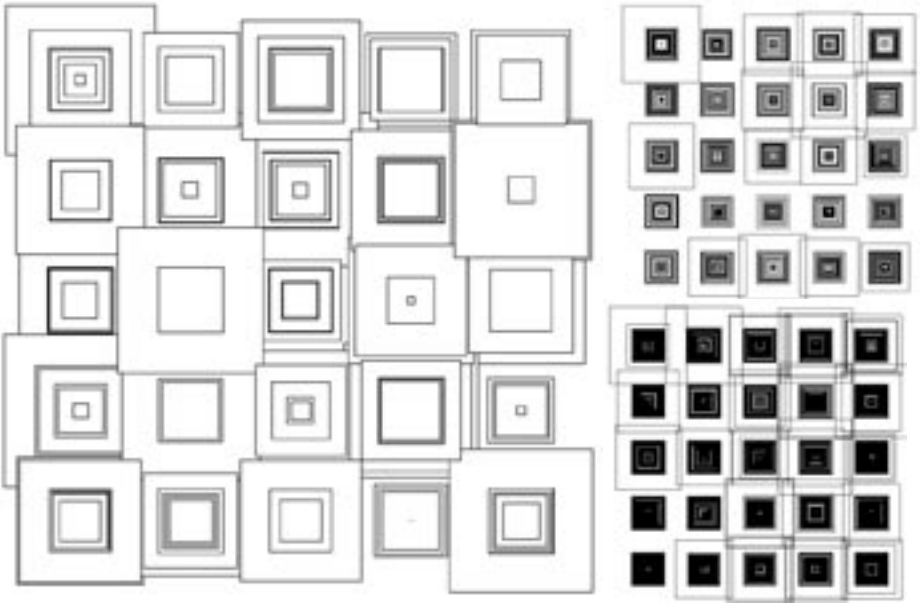


Abb. 27

```
PROCEDURE rechteck;

VAR
                x,y,x1,y1:INTEGER;
    d,h,h1:REAL;
    R:HANDLE;

BEGIN
    d:=2.5;

                For x:= 1 TO 10 DO BEGIN
    For y:= 1 TO 10 DO BEGIN

 h:=Random*5;

                Rect((x*d)-h/2,(y*d)+h/2,(x*d)+h/2,(y*d)-h/2);

  h:=Random*5;

                Rect((x*d)-h/2,(y*d)+h/2,(x*d)+h/2,(y*d)-h/2);

h:=Random*5;

                Rect((x*d)-h/2,(y*d)+h/2,(x*d)+h/2,(y*d)-h/2);


   END;
   END;
END;
run(rechteck);
```

```
PROCEDURE rechteck2;

VAR
                x,y,x1,y1:INTEGER;
    d,h:REAL;
    R:HANDLE;
BEGIN
    d:=2.5;

                For x:= 1 TO 5 DO BEGIN
    For y:= 1 TO 5 DO BEGIN

 h:=Random*7;

                Rect(((x*10)-h)+d,(y*10)-h,((x*10)+h)+d,(y*10)+
h);

    For x1:= 1 TO 5 DO BEGIN
    For y1:= 1 TO 5 DO BEGIN

 h:=Random*3;

    Rect(((x1*10)-h)+d,(y1*10)-h,((x1*10)+h)+d,(y1*10)+h);

    END;
    END;
END;
END;
END;
run(rechteck2);
```

Abb. 28

```
PROCEDURE einF;
      VAR
          currentObject:HANDLE;
          rotationAngle,centerX,centerY,dx,dy,a,i:INTEGER;
          myrandomx,myrandomy:REAL;

PROCEDURE F(startX,startY:REAL);


BEGIN

    BeginPoly;
   AddPoint(startX,startY);
 relative;
     AddPoint(0,100);
     AddPoint(80,0);
     AddPoint(0,-10);
     AddPoint(-70,0);
     AddPoint(0,-30);
     AddPoint(50,0);
     AddPoint(0,-10);
     AddPoint(-50,0);
     AddPoint(0,-50);
   closePoly;
 EndPoly;
   absolute;
END;

PROCEDURE Fmodul(startX,startY:REAL);

 BEGIN

 rotationAngle:=180;
 centerX:=0;
 centerY:=0;

   F(0,0);
    Duplicate(-90,-100);
     currentObject:=LNewobj;
     HRotate(currentObject,centerX, centerY,rotationAngle);
    Duplicate(80,-100);
     {FlipHor;}
    Duplicate(-80,0);
     FlipVer;
     {FlipHor;}

   SelectAll;
```

```
      FOR dx:=1 TO 9 DO BEGIN
      Duplicate(0,-200);


END;
END;
PROCEDURE Freihe(startX,startY:REAL);

BEGIN
  FOR a:= 10 TO 30 DO BEGIN


     myrandomy:=40+Sin(((a+10)/11)*10);
END;

  BeginXtrd(0,100+myrandomy);
  Fmodul(0,0);
  EndXtrd;
END;
PROCEDURE Fplatz(startX,startY:REAL);

BEGIN
  FOR i:= 1 TO 2 DO BEGIN
  Freihe(160,0);
  Duplicate(160*i,0);
 END;


END;
PROCEDURE Fwachsen(startX,startY:REAL);

BEGIN
FOR i:= 1 TO 3 DO BEGIN
  Fplatz(80,0);
  Duplicate(80*i,0);
 END;
 FlipVer;

END;
PROCEDURE Fhoch(startX,startY:REAL);

BEGIN

 FOR i:= 1 TO 3 DO BEGIN

     Fwachsen(80,0);
  Duplicate(80*i,0);
```

```
 END;
END;
PROCEDURE Fpattern(startX,startY:REAL);

BEGIN

FOR i:= 1 TO 3 DO BEGIN

 Fhoch(0,0);

 Duplicate(10*i,0);
 END;
END;
BEGIN

FOR i:= 1 TO 3 DO BEGIN

 Fpattern(0,0);

 Duplicate(124.3*i,0);
 END;
END;
run(einF);
```

6. Fazit

Kurz umschrieben sei zunächst das Ideal Programmbasierter Lösungsstrategien als theoretische Zielvorstellung: Der Prozeß der Lösungsfindung für ein Gestaltungs -problem basiert beim Programmieren auf der Methode von Versuch und Irrtum, wobei Regeln benutzt werden, um versuchsweise Lösungen zu generieren und dann in einem fortgeschritten Programmierstadium Prädikate zu berechnen, um festzustellen, ob diese Lösungen akzeptabel Lösungen sind. Es handelt sich dabei um einen Prozeß von Generieren und Testen in einem Lösungsraum.
"Die syntaktischen Regeln, die die abstrakte Welt der Entwürfe beherrschen, etablieren so einen Typus, und die in einer kritischenSprache ausgedrückten Prädikate etablieren die Anforderungen eines bestimmten Moments und Kontexts. Die Aufgabe des Gestaltens ist es, den Typus in einer Form zu instantiieren, die diesem gegebenen Moment und Kontext gerecht wird.
Um einen Typ in einem konkreten Kontext instantiieren zu können, müsse ein intelligents Design-System über eine kritische Sprache verfügen, die den Zusammenhang zwischen den Formen und ihrem Funktionieren im jeweiligen Kontext beschreibt und damit den Entwurfsprozeß steuert"[7]. - Soweit das Ideal.
Die Erfahrung in der Auseinandersetzung mit Algorithmen-kontrolliertem Entwerfen hat gezeigt, dass nicht Funktion, Kontext und Gestaltungswillen die prädikativen Regeln bestimmen, sondern die begrenzten Möglichkeiten einer programmierten "Gestaltungsmaschine", welche zudem eine lange Entwicklungszeit bedarf, um befriedigende Ergebnisse zu erzielen.
Beim vorliegenden Lösungsansatz handelt sich um eine individuelle Lösung für ein spezifisches Problem. Eine davon geringfügig abweichende Problemstellung macht auch eine erneute Programmierung erforderlich. Formengrammatiken (Algorithmen) haben so als Werkzeuge zur spielerischen Erzeugung von Formen ihren Wert, aber ein Gestaltungsprozeß läßt sich insgesamt nicht auf sie reduzieren.

7. Anhang

Anmerkungen

1. vgl. Allgegenwart und Verschwinden des Computers, S.16
2. vgl. Die Informatisierung des Alltags, S.15
3. Smart Objects, Smart Shapes, Smart Profiles sind Handelsmarken von think3 Inc. Smart Objects – ein Ensemble aus Softwaretools und Inhalten, das dem Anwender ermöglicht, vordefinierte Designelemente aus „intelligenten Teilen“ per „Drag-and-Drop“ dazu verwenden, ein parametrisches Volumenmodell für mechanisches Design zu erstellen
4. PostgreSQL und dessen kompletter Quellcode sind frei und öffentlich verfügbar. Die englische Aussprache ist “Post-Gres-Q-L”
7. vgl. Das virtuelle Unternehmen, S.52
6. vgl. C-MOEBEL, S.93
7. vgl. Stilverzicht, S.113

Literatur

Xiaoyuan Tu, Artifical animals for computer animation: biomechanics, locomotion, perception, and behavior, Berlin Heidelberg, 1999

William H. Davidow, Michael S. Malone, Das virtuelle Unternehmen, Der Kunde als Co-Produzent, Frankfurt am Main 1993

Dagmar Steffen, C-MOEBEL. Digitale Machart und gestalterische Eigenart, Giessen, 2003

John Maeda, Design By Numbers, Massachusetts Institute of Technology, 1999

Christian Kühn, Stilverzicht, Typologie und CAAD als Werkzeuge einer autonomen Architektur, Braunschweig/Wiesbaden 1998

Branko Kolarevic, Architecture in the digital age: design and manufacturing, New York, 2003

Zeitschriften

Allgegenwärtiges Rechnen, Friedemann Mattern, in LOG IN, Heft Nr. 125, 2003

Bauwelt, Heft Nr. 45, 1997

Digitalisierung des Alltags. Was ist Pervasive Computing?, Marc Langheinrich, Friedemann Mattern, Aus Politik und Zeitgeschichte, B 42/2003, 13. Oktober 2003

Die Informatisierung des Alltags, Friedemann Mattern und Marc Langheinrich, Bulletin ETH Zürich Nr.291 November 2003

Aufsätze

From Animals to Animats7, Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior, Massachusetts Institute of Technologie, 2002

Vom Verschwinden des Computers-Die Vision des Ubiquitous Computing, Institut für Pervasive Computing, ETH Zürich, Friedemann Mattern, 2003

Allgegenwart und Verschwinden des Computers, Leben in einer Welt smarter Alltagsdinge, Jürgen Bohn, Vlad Coroama, Marc Langheinrich, Friedemann Mattern, Michael Rohs, Institut für Pervasive Computing ETH Zürich, 2003

Die Privatsphäre im Ubiquitous Computing-Datenschutzaspekte der RFID-Technologie, Marc Langheinrich, Institut für Pervasive Computing ETH Zürich, 2003

Bildnachweis

Abb.1: Allgegenwart und Verschwinden des Computers, Leben in einer Welt smarter Alltagsdinge, Jürgen Bohn, Vlad Coroama, Marc Langheinrich, Friedemann Mattern, Michael Rohs, Institut für Pervasive Computing ETH-Zürich, 2003

Abb.2: Vom Verschwinden des Computers-Die Vision des Ubiquitous Computing, Institut für Pervasive Computing, ETH Zürich, Friedemann Mattern, 2003

Abb.3: Digitalisierung des Alltags. Was ist Pervasive Computing?, Marc Langhein rich, Friedemann Mattern, Aus Politik und Zeitgeschichte, B 42/2003, 13. Oktober 2003

Abb.4: caad ethz nds 2003/04 | oop/squeak | Detlef Wingerath, Hanne Sommer, Michelangelo Ribaudo
Abb.5: caad ethz nds 2003/04 | oop/squeak | Karsten Droste
Abb.6: caad ethz nds 2003/04 | phpPgAdmin | Detlef Wingerath
Abb.7: caad ethz nds 2003/04 | Flash | Detlef Wingerath
Abb.8: http://images.google.com/images?q=schnittmuster&hl=de&lr=&ie=UTF-8&start=280&sa=N
Abb.9: replay<säulen/>atlas, Diplomwahlfach CAAD , D-Arch, ETH Zürich Wintersemester 2003/04
Abb.10: http://images.google.com/images?q=handy&hl=de&lr=&ie=UTF-8&sa=N&tab=wi
Abb.11: Detlef Wingerath
Abb.12: http://images.google.com/images?q=handy&hl=de&lr=&ie=UTF-8&sa=N&tab=wi
Abb.13: Detlef Wingerath
Abb.14: Detlef Wingerath
Abb.15: Detlef Wingerath
Abb.16: Detlef Wingerath
Abb.17: http://www.f-hagemann.de/content/Laser
Abb.18: http://www.f-hagemann.de/content/Laser
Abb.19: Detlef Wingerath
Abb.20: Detlef Wingerath
Abb.21: Detlef Wingerath
Abb.22: Detlef Wingerath
Abb.23: Detlef Wingerath
Abb.24: Detlef Wingerath
Abb.25: Detlef Wingerath
Abb.26: caad ethz nds 2003/04 | VectorScript | Detlef Wingerath
Abb.27: caad ethz nds 2003/04 | VectorScript | Detlef Wingerath
Abb.28: caad ethz nds 2003/04 | VectorScript | Detlef Wingerath

Danksagung

Lehrstuhl für CAAD	Ludger Hovestadt
	Sibylla Spycher
	Markus Braach
	Silke Berit Lang
	Russell Loveridge
	Ulrike Bahr
	Mathias Ochsendorf
	Oliver Fritz
	Christoph Schindler
	Karsten Droste
	Oskar Zieta
	Odilo Schoch
	Susanne Schumacher
	Torsten Spindler
	Kai Strehlke
	Andrea Gleiniger
	Pia Fricker
	Kai Rüdenauer
	Fabian Scheurer
	Maja Dzieglewska
NDS Kurs Koordinator	Philipp Schaerer
Betreuung Nachdiplomstudien	Anita Buchschacher
	Bruno Dobler
firma: gimmel Leder	Max Gimmel AG
	Gerberei CH-9320