VizBox

- eine webbasierte Bibliothek
für das Design von Visualisierungen

Masterarbeit
zur Erlangung des akademischen Grades
eines Masters of Science (M.Sc.)

Kerstin Alexandra Ludwig
Inselgasse 13
78462 Konstanz

Gutachter: Prof. Dr. Harald Reiterer
Prof. Dr. Oliver Deussen

Universität Konstanz
FB Informatik und Informationswissenschaft
Master-Studiengang Information Engineering
Oktober 2005

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

DTD	Dokumenttypdefinition
GoF	Gang of Four
GUI	Graphical User Interface
GUIDE	Graphical User Interface Design and Evaluation Method
HCI	Human-Computer Interaction
HCIL	Human-Computer Interaction Lab
HI	Human-Interaction
IPL	Interaction Pattern Language
MUSE	Method for Usability Engineering
OLIVE	On-line Library of Information Visualization Environments
OOSE	Object-Oriented Software Engineering
PLoPD	Patterns Languages of Program Design
PNNL	Pacific Northwest National Laboratory
RAD	Rapid Applications Development
RSVP	Rapid Serial Visual Representation
SE	Software Engineering
SPIRE	Spatial Paradigm for Information Retrieval and Exploration
SPL	Structure Pattern Language
TTT	Task by Data Type Taxonomy for Information Visualization
UI	User Interface
VizBox	Visualization Box
XML	Extensible Markup Language
XSD	XML Schema

Abstract

The subject of this thesis is the presentation and explanation of the VizBox, a web-based library for the design of visualizations. The library consists of two components: the visualization patterns and the structured descriptions of visualizations which both assist the designer in the creation of alternative visualization designs.

The goal of the VizBox approach is to combine the main advantages of two prominent design strategies from traditional design disciplines, that are also discussed in detail. First, there is the pattern approach, originally developed by the architect Chistopher Alexander which generates designs of high quality. The second approach is the so-called "Box-of-Ideas" which originates from the context of product design. This approach functions as an assistance for the designer in developing new and innovative proposals of designs while she is designing a visualization.

The thesis discusses how these approaches can be modified and utilized for the visualization of data. The VizBox combines the advantages of both approaches by adapting them for visualization design and embedding them in the generation of a visualization system.

The thesis presents a new pattern format which enables VizBox for the combination of the existing formats. Corresponding to this, a new description format for visualization is specified. Eventually a set of design suggestions for the VizBox as a visualization system is developed which draws profit from the combination of the two approaches and makes the derived approach applicable for the context of data visualization. An added value of the VizBox is the possibility to design a visualization system by the consequent use of user-friendly visualization techniques. The compilation of these techniques in a single library makes the VizBox also usable as a kind of web-based encyclopedia for the visualization of data, because it contains a huge amount of knowledge about this subject.

1 Einleitung

Information ist allgegenwärtig. Ob es sich dabei um Zeitungsartikel, Verkehrsmeldungen, Wetterberichte, Zugfahrpläne, Aktienkurse, Reisekarten oder Geschäftsberichte handelt, Information findet sich in allen Bereichen des täglichen Lebens – im Privaten sowie im Beruflichen. Jedoch wächst die Menge an produzierter Information und damit auch die Schwierigkeiten mit ihrem Umgang zunehmend und ihre nahezu permanente Verfügbarkeit erzeugt einen Effekt, welcher unter dem populären Schlagwort der „Informationsüberflutung" bekannt geworden ist.
Aufgrund dieses neuen Stellenwertes der Information und der Tatsache, dass sich bestimmte Formen von Medien, wie die visuelle Repräsentation, besonders für die Darstellung von Information eignen, wird intensiv an der Entwicklung neuartiger Formen von Visualisierungsmöglichkeiten gearbeitet. Mittels dieser Visualisierungen ist es möglich - unter der Voraussetzung einer qualitativ hochwertigen visuellen Repräsentation - große Mengen von Information rasch interpretieren zu können. Leistungsstärkere Computer und neue Technologien ermöglichen dabei die Entwicklung immer weiterer, neuer Visualisierungstechniken, welche Personen bei dem Umgang mit Information, sei es nun im Arbeits- oder Privatleben, bei der Kommunikation oder Exploration von Information unterstützen.

Jedoch ist die Disziplin des Designs von Visualisierungen - das Visualization Design – im Vergleich zu traditionellen Designdisziplinen, wie der Architektur, dem Grafik oder Produkt Design, ein immer noch relativ junges Gebiet, welches auf nur geringe Erfahrung beim Entwurf zurückblicken kann. Untersuchungen haben gezeigt, dass insbesondere die bewusste Ausführung der kreativen Designphase, in welcher alternative Entwurfsvorschläge erschlossen werden und, welche für das Resultat eines qualitativ hochwertigen Entwurfs als äußerst relevant erachtet wird, innerhalb des Designprozesses einer Visualisierung noch vernachlässigt wird.
Diese Arbeit untersucht Ansätze aus traditionellen Designdisziplinen, welche speziell der Unterstützung dieser Phase des Designprozesses dienen. Zu diesen Ansätzen gehören der Ansatz der *Pattern*, welcher von dem Architekten Christopher Alexander entwickelt wurde, sowie der Ansatz der *Box-of-Ideas*, welcher primär aus dem Produkt Designbereich stammt. Auf Basis dieser Untersuchungen wurde ein Konzept für ein webbasiertes Kreativitätstool entwickelt, welches beide Ansätze für den Bereich des Visualization Design adaptiert und in ein Visualisierungssystem einbettet. Diese webbasierte Bibliothek, die sogenannte Visualization Box (VizBox), hat das Ziel dem Visualization Designer eine Technik zur Verfügung zu stellen, welche speziell in der kreativen Designphase des Entwicklungsprozesses einer Visualisierung zur Generierung alternativer Lösungsideen eingesetzt werden kann und die Entwicklung qualitativ hochwertiger und innovativer Visualisierungen fördert.

2 Visualisierung von Daten

„The world is complex, dynamic, multidimensional; the paper is static, flat. How are we to represent the rich visual world of experience and measurement on mere flatland?"
[Edward R. Tufte]

Die Thematik der Visualisierung von Information bzw. Daten ist ein sehr umfangreiches und komplexes Gebiet, welches über eine lange Tradition in der Geschichte der Menschheit verfügt, jedoch auch und nicht zuletzt aufgrund der zunehmend wachsenden Menge an Information ein Thema, mit sehr aktuellem Bezug. Seit Beginn des Computerzeitalters ist es möglich geworden, visuelle Repräsentationen von Daten in fließenden dynamischen Visualisierungen darzustellen, welche vom Benutzer interaktiv manipuliert werden können und somit seinem Bedürfnis nach unterschiedlichen Darstellungen der Information nachkommen. Aufgrund dessen werden von einer Reihe unterschiedlicher Disziplinen eine Vielzahl an interaktiven Visualisierungsmöglichkeiten erforscht und entwickelt, welche den Benutzer bei dem täglichen Umgang mit Information unterstützen.
Dieses Kapitel gibt eine kurze Einführung in die Thematik der Visualisierung von Daten, wie sie sich von der Pre-Computer-Phase zur Computer-Phase entwickelt hat, weshalb visuelle Repräsentationen von Daten so effektiv sind, wie der Prozess einer Visualisierung aussieht und wie sich qualitativ hochwertige Visualisierungen definieren.

2.1 Historie

2.1.1 Visuelle Artefakte

„Visual artifacts and computers do for the mind what cars do for the feet!"
[Stuart K. Card, Jock D. Mackinlay, Ben Shneiderman]

Die Tätigkeiten des Menschen sind stark mit physischen externen Hilfsmitteln, hier auch Artefakte genannt, verbunden. Die wenigsten Tätigkeiten werden in unserer heutigen Welt ausschließlich mental erledigt. Die Kombination von mentalen Denkprozessen und physischen Hilfsmitteln - also innerer mentaler Aktion und externer wahrnehmbarer Interaktion - ist stark miteinander verknüpft und hat schon eine lange Tradition in der Geschichte der Menschheit. Laut [Card et al. 1999] ist die Verflechtung von innerer mentaler Aktion und externer Wahrnehmung (und Manipulation) kein Zufall. Es ist vielmehr der Schlüssel dazu, wie Menschen ihre mentalen Fähigkeiten zunehmend erweitern und somit ihre Intelligenz steigern. Es gibt eine Reihe von verschiedenartigen externen Hilfsmitteln, jedoch ist die mit Abstand wichtigste Kategorie diejenige der grafischen oder auch visuellen Hilfsmittel. Bei grafischen Hilfsmitteln lassen sich insgesamt zwei unterschiedliche Zwecke unterscheiden. Zum einen dienen grafische Hilfsmittel dazu

eine Idee zu kommunizieren. Damit verbunden wird häufig der bekannte Ausspruch: „*A picture is worth ten thousand words*." Zum anderen ermöglichen grafische Hilfsmittel die Idee selbst zu erzeugen oder zu entdecken: „*Using vision to think*." Diese beiden Zwecke von grafischen Hilfsmitteln sind grundsätzlich verschieden, auch wenn sie eng miteinander verknüpft sind. Ein einfaches informelles Experiment, bei dem Testpersonen einmal Multiplikationen von Zahlen rein mental und beim anderen mal unter Zuhilfenahme von Stift und Papier ausführen sollten, zeigt die Relevanz von grafischen Hilfsmitteln. Die Verwendung von Stift und Papier reduzierte die Zeit, welche für die Erledigung der Aufgaben benötigt wurde, um den Faktor fünf. Dies zeigt, dass selbst einfachste Hilfsmittel, wie Stift und Papier die kognitiven Fähigkeiten eines Menschen um eine Vielfaches erweitern. Durch die Verwendung einer visuellen Repräsentation ist es möglich Teilergebnisse außerhalb des Gedächtnisses festzuhalten und somit das Gedächtnis einer Person zu erweitern. Stift und Papier ist nur ein Beispiel für ein grafisches Hilfsmittel; andere sind zum Beispiel Abakus, Rechenschieber, Nomographen, Taschenrechner, Karten oder auch Diagramme, um nur einige zu nennen. [Card et al. 1999]
Grafische oder auch visuelle Hilfsmittel bzw. Artefakte unterstützen den Denkprozess. Sie sind seit jeher stark verflochten mit kognitiver Aktion. Man kann die Entwicklung der Zivilisation auch an der Erfindung von visuellen Artefakten ablesen: Schrift, Mathematik, Karten, Diagramme und *Visual Computing*. Viele Aktivitäten in unserem Umfeld haben mit der Erzeugung oder Verwendung visueller Artefakte zu tun. Visuelle Artefakte erweitern die kognitiven Fähigkeiten eines Menschen, welche wiederum für die Aneignung oder Verwendung von Wissen verantwortlich sind. [Card et al. 1999]

2.1.2 Die menschliche Wahrnehmung

Das menschliche visuelle System hat sich über zehn Millionen Jahre hinweg als Instrument für die Wahrnehmung und Erkennung unserer natürlichen Umgebung entwickelt. Das gesamte System ist darauf ausgerichtet, Informationen aus unserer Welt, mit ihren ganz bestimmten physikalischen Eigenschaften, zu extrahieren. Laut Colin Ware führt die Interaktion des stets wachsenden Nervensystems mit der täglichen Realität, zu einem mehr oder weniger standardisierten visuellen System, weshalb dieselben visuellen Konzepte oder Entwürfe für alle Menschen gleich effektiv funktionieren.
Das menschliche visuelle System ist speziell darauf ausgerichtet, Muster in seiner Umgebung zu suchen und wiederzuerkennen. Jedoch unterliegt das System eigenen Regeln. Obwohl es sehr flexibel ist, ist es darauf abgestimmt, Daten zu erkennen, welche auf bestimmte Art und Weise präsentiert werden. Gibt es Abweichungen in der Darstellung, so bleiben diese Muster unerkannt. Sind die Funktionsweisen der menschlichen Wahrnehmung bekannt, kann dieses Wissen auf Regeln für die Darstellung von Daten übertragen werden. Werden diese Regeln beachtet, ist es möglich Daten so zu präsentieren, dass für den Menschen informative und vor allem wahrnehmbare Muster entstehen. [Ware 1999]

Die Visualisierung von Daten macht sich bereits seit Jahrhunderten diese Tatsache, dass das menschliche Wahrnehmungssystem stark an die sehr effektive Verarbeitung von visuell kodierten Informationen angepasst ist, zu Nutze. [Tufte 1983]

2.1.3 Pre-Computer-Phase

Bis vor einiger Zeit bedeutete der Begriff „*Visualization*" laut dem *Shorter Oxford English Dictionary* noch: „*constructing a visual image in the mind*." Inzwischen hat sich die Bedeutung dieses Begriffs gewandelt. Heutzutage versteht man unter dem Begriff „*Visualization*": „*A graphical representation of data or concepts*". Demnach wandelte sich Visualisierung von einem internen Konstrukt des Verstands zu einem externen Artefakt, welches der Erweiterung des menschlichen Verstandes dient. [Ware 1999]
Visualisierung von Daten gibt es bereits seit einigen Jahrhunderten. Die Entwicklung der perspektivischen Projektion durch die Florentinischen Architekten während des 15.Jh., zur Zeit der Italienischen Renaissance, war ein Geschenk für die Welt des visuellen Denkens. Sie stellt eine einfache Erweiterung einer zweidimensionalen Oberfläche dar, durch die es Menschen möglich wird, Objekte in einem geometrisch korrekten Kontext darzustellen und welche durch die eigene tägliche Erfahrung im dreidimensionalen Raum verständlich wird. [Tufte 1990][Tufte 1997]
Der Übergang schließlich von verkleinerten bildhaften Darstellungen der physischen Welt, beispielsweise in Form von Karten, hin zu abstrakten Darstellungen, wie zum Beispiel Graphen, war ein enormer konzeptioneller Schritt. Es bedurfte 5000 Jahre um die Namen der Koordinaten von Ost-West und Nord-Süd in empirisch messbare Variablen X und Y zu verwandeln. Parallelen können hierbei auch zur Kunst gezogen werden. In der Kunst wurde das naturalistische Koordinatensystem erst Anfang des 20.Jh. durch die fraktalen Bilder des Kubismus, welche gleichzeitig multiple Blickwinkel in einem Bild vereinen, verzerrt und später schließlich mit *Abstract Painting*, bei dem die zwei Dimensionen der Leinwand nicht länger zu einer weltlichen Szenerie, sondern nur noch zu sich selbst in Bezug stehen, völlig aufgelöst. [Tufte 1997]

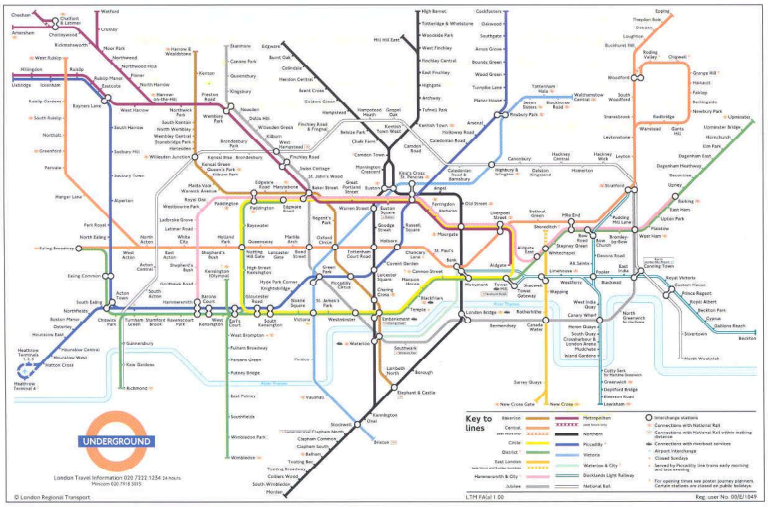

Abb. 1 Karte des Londoner Underground Transportation System [Spence 2001]

Ein sehr bekanntes Beispiel für die Visualisierung von Daten, welches sich die Verzerrung der geografischen Räumlichkeiten auf einer Karte zu Nutze macht, ist die Karte des *Londoner Underground Transportation System (The Tube)* von Harry Beck aus dem Jahr 1931 (Abb. 1). Harry Beck, ein pensionierter Konstruktionszeichner, erkannte damals, dass es keine Relevanz hat, die eigene genaue geografische Lage zu kennen, wenn man unter Tage ist. Er realisierte, dass, solange die Stationen der U-Bahn in der richtigen Reihenfolge präsentiert werden und ihre Verkehrsknotenpunkte klar skizziert sind, man die Skalierung frei verzerren könnte. Der Fokus der Benutzer einer U-Bahn liegt primär auf der geplanten Reiseroute, also auf der gewünschten Abfahrts- und Zielstation und der richtigen U-Bahnlinie. Beck gab seiner Karte die systematische Präzision eines elektrischen Leitungsnetzes und erzeugte somit ein gänzlich neues imaginäres London, welches nur wenig mit der Geografie der Stadt oben drüber zu tun hatte. Seit 1931 wurde die Karte oftmals überarbeitet und modifiziert, beinhaltet aber immer noch Becks brillante Idee, aufgrund derer sie auch mehrfach weltweit kopiert wurde. [Spence 2001]

Oftmals sind Problemstellungen jedoch nicht ausschließlich auf einen zweidimensionalen Raum beschränkt, da die Welt für gewöhnlich multivariant ist. Die schon seit Jahrhunderten bestehende zentrale Fragestellung bei der Darstellung von Daten lautet: Wie präsentiert man drei oder mehr Dimensionen von Daten auf einer zweidimensionalen Oberfläche, wie beispielsweise Wände, Steine, Gewebe, Papier oder Computerbildschirmen? Für diese abstrakteren mehrdimensionalen Daten, welche nicht in unserer dreidimensionalen Welt angesiedelt sind, werden schon seit einiger Zeit, von Personen, die mit diesen Daten zu tun haben, verschiedene Methoden, bzw. Techniken zur Darstellung entwickelt. Zum Teil sind diese Visualisierungen in alltäglichen, manchmal allgegenwärtigen Darstellungen von Daten zu finden, wie zum Beispiel die ausgeklügelte Strukturierung der Periodentabelle der chemischen Elemente (mit Hunderten von verschiedenen vorgeschlagenen Zusammenstellungen, um die damit zusammenhängende Komplexität besser festhalten zu können). [Tufte 1997]

2.1.4 Computer-Phase

Wie bereits erwähnt, haben grafische Hilfsmittel eine sehr lange Geschichte in der Kultur der Menschheit. Neu ist, diese mittels Computern zu erzeugen. Auch wenn die Erzeugung von Grafiken eine grundsätzlich menschliche Aktivität ist, kann sie sehr effektiv von Computern unterstützt werden. [Spence 2001] Das Medium des Computers erlaubt es, grafische Darstellungen zu erzeugen, welche automatisch Tausende von Datenobjekten zu Bildern zusammenbauen und diese interaktiv durch einen Benutzer modifizieren zu lassen. Das bedeutet, dass Visualisierungen nicht länger statisch sind sondern interaktiv von Benutzern beeinflusst werden können. [Card et al. 1999]

2.1.5 Pre-Computer-Phase vs. Computer-Phase

„A graphic is no longer 'drawn' once and for all: it is 'constructed' and reconstructed (manipulated) until all relationships which lie within it have been percieved…a graphic is never an end in itself: it is a moment in the process of decision making."
[Jacques Bertin]

In der Pre-Computer-Phase musste der Erzeuger einer Grafik bereits bei deren Erstellung eine Entscheidung darüber treffen, welche Daten er visualisieren möchte und wie er diese - entsprechend seines eigenen Verständnisses der Aufgabe, welche mit dieser Visualisierung erledigt werden oder der Mitteilung, welche übermittelt werden sollte - repräsentiert. Autor und Betrachter waren zwei verschiedene Personen. Mit Hilfe des Computers ist es nun möglich, dass der „Betrachter", also der Benutzer, - im Rahmen der Freiheiten, die ihm der Erzeuger mit der Gestaltung des Visualisierungstools definiert hat - die interaktive Kontrolle über diese Art von Entscheidungen hat. [Spence 2001]

2.1.6 Visualisierung

„Graphing data needs to be interactive because we often do not know what to expect of the data; a graph can help discover unknown aspects of the data, and once the unknown is known, we frequently find ourselves formulating new questions about the data."
[William S. Cleveland]

In den letzten Jahrzehnten entwickelten sich zunehmend Visualisierungen, welche sich den Computer zu Nutze machen. Die heutige Computertechnologie ist in der Lage äußerst komplexe Visualisierungen zu erzeugen, welche eine extrem verbesserte Wiedergabe besitzen und interaktiv modifizierbar sind. [Card et al. 1999] Auf der menschlichen Seite kann die Visualisierung als dynamisches Artefakt - als Erweiterung des Denkprozesses - agieren. Durch ihre Dynamik erfüllt sie das Bedürfnis des Benutzers nach verschiedenen Sichten oder nach detaillierten Informationen der Daten. [Ware 1999] Eine aufkommende

Sichtweise betrachtet den Menschen und den Computer zusammen als ein Problemlösungssystem. In einem solchen Modell wird die Datenvisualisierung zu einem Teil des Interfaces zwischen den Komponenten Mensch und Maschine. Die Visualisierung ist ein bidirektionales Interface, wenn auch hochgradig asymmetrisch, mit einer weitaus höheren Bandbreite der Kommunikation in Richtung Maschine zu Mensch als umgekehrt.[Ware 1999] Nach [Card et al. 1999] ergibt sich die folgende Definition für Visualisierung:

„Visualization can be defined as the use of computer-supported, interactive, visual representations of data to amplify cognition."

Die Visualisierung von Daten allgemein hat grundsätzlich zwei Facetten: Datenpräsentation und Datenexploration. Der Fokus der Datenpräsentationen ist die Kommunikation von bereits bekannten Fakten durch geeignete Repräsentationsformen. Bei der Datenexploration geht es dagegen darum, mittels angemessener Visualisierungen unbekannte Verknüpfungen zwischen Thematiken aufzudecken, auch genannt „*Visual Data Mining*". Die Übergänge zwischen den beiden Facetten können als fließend betrachtet werden. Sowohl für die Datenpräsentation als auch für die Datenexploration erweist sich die visuelle Repräsentation als vorteilhaft. Im Fall der Präsentation, steht die Kommunikation im Vordergrund, im Fall der Exploration ist es die Entdeckung. [Mann 2002]

Visualisierungen, die sich den Computer zu Nutze machen, haben sich als unabhängige technische Disziplin entwickelt, an der eine Reihe unterschiedlicher Disziplinen beteiligt sind, wie z.B. die Experimentelle oder Kognitive Psychologie, die Statistische bzw. Computer Grafik, das Software Engineering oder die Human-Computer Interaction. Speziell seit den 80er Jahren beeinflussten die Konzepte der Datenvisualisierung viele verschiedene Bereiche. [Mann 2002] Zu dieser Zeit wurden computerbasierte Visualisierungen vor allem im Bereich der Wissenschaft angewandt, was die Entwicklung des Bereichs der *Scientific Visualization* zur Folge hatte. Diesen Visualisierungen liegen in erster Linie physische Daten, wie beispielsweise der menschliche Körper, die Erde, Naturphänomene, Gebäude, technische Konstruktionen oder Moleküle, zugrunde. Der Computer wird dazu verwendet um – mittels oftmals dreidimensionaler Visualisierungen und Animationen - Eigenschaften dieser physischen Elemente darzustellen. Obwohl eine Visualisierung eines physischen Elements an sich abstrakt ist, ist die Information selbst dennoch inhärent geometrisch. [Card et al. 1999]

Seit den 90er Jahren kommen Visualisierungen auch zunehmend in allgemeineren Bereichen, wie Firmen oder Ausbildung zur Anwendung. Diese allgemeinere Anwendung wird *Information Visualization* oder auch Informationsvisualisierung genannt. [Mann 2002][Card et al. 1999] Ebenso wie die visuelle Darstellung physischer Daten ist es oftmals hilfreich, nicht-physische Daten, wie Finanzdaten, Business-Information, Sammlungen von Dokumenten und abstrakten Konzepten zu visualisieren. Jedoch beinhaltet diese Art der Daten keinerlei ersichtlichen räumlichen Bezug. Daher kommt zu der allgemeinen Problematik, wie man sichtbare Eigenschaften von Objekten wiedergibt,

das grundsätzliche Problem hinzu, wie man nichträumliche Abstraktionen in effektive visuelle Formen transformiert. Da in der heutigen Welt eine große Masse solcher abstrakter Information existiert, gibt es zahlreiche Versuche, Visualisierungen in den Bereich des Abstrakten zu erweitern. [Card et al. 1999] Zu diesem Zweck beschäftigt sich die *Information Visualization* mit abstrakten, multi-dimensionalen und multivarianten Daten. Primäres Ziel bei der *Information Visualization* ist, im Gegensatz zur *Scientific Visualization*, nicht die Abbildung selbst, sondern viel mehr die Möglichkeit, Muster, Clusterungen, Lücken oder Sonderfälle in den Daten offen zu legen. [Shneiderman 2001] Der Nutzen die physischen Elemente abzubilden ist in diesem Bereich nicht wichtig, oftmals sogar gänzlich irrelevant. [Spence 2001]

Auch wenn verschiedene Techniken und Methoden innerhalb der *Information Visualization* oder auch der *Scientific Visualization* entwickelt wurden, um mehrdimensionale Daten zu visualisieren und fiktive dreidimensionale Räume zu erschaffen, so bleibt die Welt, welche auf einem Informationsdisplay dargestellt wird – zumindest bisher noch - nach wie vor zweidimensional. D.h. jegliche Kommunikation zwischen den Betrachtern einer Visualisierung und deren Erzeugern muss auf einer zweidimensionalen Oberfläche erfolgen. [Tufte 1990]

„Escaping this flatland is the essential task of envisioning information – for all the interesting worlds (physical, biological, imaginary, human) that we seek to understand are inevitably and happily multivariate in nature. Not flatlands."
[Edward R. Tufte]

2.2 Referenzmodell für Visualisierung

Bei der Erzeugung von Visualisierungen geht es prinzipiell darum, Daten in visuelle Formen umzuwandeln. Ziel dabei ist es, eine entsprechende adäquate Umwandlung zu finden. Um diesen Transformationsprozess konzeptionell zu beschreiben, werden Modelle verwendet. Die Verwendung von Modellen erleichtert die Diskussion über Visualisierungen und ermöglicht es, diese zu beschreiben und einander gegenüber zu stellen. [Card et al. 1999] Es wurden eine Reihe von Referenzmodellen von verschiedenen Forschern, die im Bereich der Visualisierung von Daten tätig sind, vorgeschlagen. Zu den bekanntesten gehört jedoch das Referenzmodell für Visualisierung, das unter der Leitung von Ben Shneiderman am *Human-Computer Interaction Lab (HCIL)* an der Universität Maryland entwickelt wurde. Dieses Modell (Abb. 2) beinhaltet zum einen den Datenfluss, welcher vom Input der originären Daten (Rohdaten) über diverse Transformationsschritte bis zu den endgültigen visuellen Darstellungen verläuft, sowie eine Reihe von Interaktions-möglichkeiten, welche es dem Benutzer ermöglichen auf diese Transformationen einzuwirken.

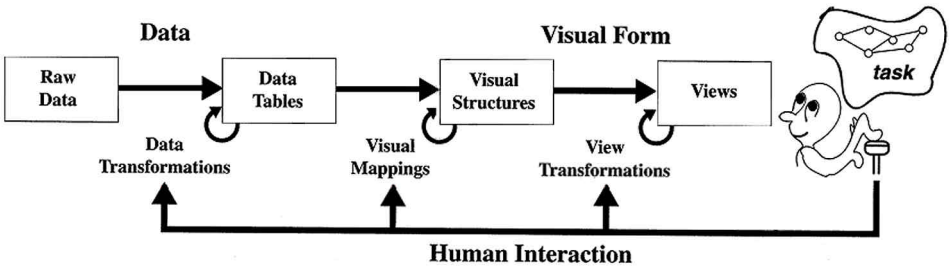

Abb. 2 Referenzmodell für Visualisierung [Card et al. 1999]

Das Referenzmodell besteht aus insgesamt drei Transformationsschritten. Die erste Transformation, die Datentransformation (*Data Transformations*) wandelt die Rohdaten (*Raw Data*) - zumeist unstrukturierte Daten in spezifischen Formaten - in Datentabellen (*Data Tables*) um. Datentabellen sind relationale Beschreibungen der Daten, welche Metadaten enthalten. Das visuelle Mapping (*Visual Mappings*) transformiert die Datentabellen in visuelle Strukturen (*Visual Structures*). Die visuellen Strukturen bestehen aus räumlichen Bereichen, diversen Zeichen (*Marks*) und deren grafischen Eigenschaften. Die Viewtransformation (*View Transformations*) schließlich erzeugt durch das Spezifizieren von grafischen Parametern, wie Position, Skalierung und Ausschnitte, aus den visuellen Strukturen die endgültigen Views.

Der gesamte Prozess wird von der eigentlichen Aufgabe (*Task*) des Benutzers ausgelöst und von der menschlichen Interaktion, also dem Benutzer (*Human Interaction*) manipuliert. Durch die Interaktionsmöglichkeiten ist es dem Benutzer möglich, die Parameter dieser Transformationen zu kontrollieren und dadurch beispielsweise die Sicht (*View*) auf bestimmte Datenbereiche zu begrenzen oder die Art der Transformation zu ändern. Die Visualisierungen und ihre Bedienungselemente dienen der Aufgabenerfüllung. [Card et al. 1999] Im nachfolgenden wird noch mal detaillierter auf die einzelnen Schritte des Referenzmodells und die Interaktionsmöglichkeiten des Benutzers eingegangen, wie es bei [Card et al. 1999] vorgestellt wird.

2.2.1 Data Transformation

Bei der Datentransformation erfolgt die Umwandlung der Rohdaten in Datentabellen. Die Rohdaten, welche in vielfältigen Formaten vorliegen können, wie zum Beispiel als Suchergebnisse oder auch als Text eines Romans, dienen der Visualisierung als Input. Bei der Umwandlung dieser Rohdaten in Datentabellen, werden diese Daten für gewöhnlich in eine Relation oder eine Gruppe von Relationen umgewandelt, welche strukturierter und deshalb einfacher in visuelle Formen transformiert werden kann.

Mathematisch gesehen sind Relationen eine Gruppe von Tupeln {<$Value_{iX}$, $Value_{iY}$, ...>, <$Value_{jX}$, $Value_{jY}$, ...>}. Datentabellen bestehen somit aus Variablen und *Cases*, welche diese Variablen beschreiben, wobei ein *Case* auch wieder als Variable verwendet werden kann und wiederum *Cases* hat. Der Vorteil solch eines mathematischen Ansatzes liegt in der Erweiterbarkeit, Dynamik und Flexibilität dieses Datenmodells. Datentransformationen sind komplex, ebenso wie die Arten von Transformationen, welche man visualisieren und eventuell durch die Visualisierungen kontrollieren möchte. Jede Version der Datentabelle enthüllt verschiedene Aspekte der Daten und führt möglicherweise zu einer anderen Wahl einer visuellen Struktur.

2.2.2 Visual Mapping

Wichtigster Schritt des Modells ist das *Visual Mapping*. Datentabellen basieren auf mathematischen Relationen, visuelle Strukturen basieren dagegen auf grafischen Eigenschaften. Bei der Visualisierung werden Datentabellen in Visuelle Strukturen umgewandelt, welche einen räumlichen Bereich durch Zeichen (*Marks*) und grafische Eigenschaften erweitern um Informationen zu kodieren. Es gibt zumeist mehrere Möglichkeiten Datentabellen in visuelle Strukturen umzuwandeln. Ein gutes Mapping zu finden ist jedoch eine Herausforderung und gelingt nicht immer (vgl. Kapitel 2.3). Visuelle Strukturen bestehen aus insgesamt drei Komponenten:

- der **räumlichen Basis** (Verwendung von Raum),
- den **Zeichen** (visuellen Elementen, die in dem Raum dargestellt werden),
- den **grafischen Eigenschaften** der Zeichen (Gestaltprinzipien, ...).

Räumliche Basis

Die Verwendung von Raum stellt den grundlegendsten Aspekt von visuellen Strukturen dar. Raum ist aus Sicht der Wahrnehmung dominant, daher ist die räumliche Position eine gute visuelle Kodierung für Daten. Aufgrund der Dominanz wird es an dieser Stelle separat von den anderen Features als Bereich behandelt, in welchen andere Teile der visuellen Struktur eingefügt werden, weshalb es hier auch als *Räumliche Basis* bezeichnet wird. Leerer Raum an sich kann behandelt werden, als ob er eine metrische Struktur hat. Man beschreibt diese Struktur mit den Begriffen Achsen und ihren Eigenschaften, wobei man dabei insgesamt vier elementare Typen von Achsen unterscheiden kann:

- die **unstrukturierte** Achse (keine Achse),
- die **nominale** Achse (Aufteilung des Raums in Teilbereiche),
- die **ordinale** Achse (die Ordnung der Teilbereiche ist von Bedeutung) und
- die **quantitative** Achse (der Raum hat eine metrische Struktur).

Lässt sich der quantitative Maßstab der Achse in weiterer Teilbereich aufteilen, so ist eine weitere Unterteilung der Achse möglich. Dies gilt ebenfalls für den Fall, wenn die Achseneinheit eine weiterunterteilbare Kennziffer, physikalische Einheit oder eine sonstige übergeordnete Einheit darstellt. Achsen können linear oder radial verlaufen und sind ein wichtiger Baustein für die Entwicklung von visuellen Strukturen.

Zeichen (Marks)

Zeichen sind sichtbare Elemente, welche innerhalb des Raums erscheinen. Es gibt vier grundsätzliche Typen von Zeichen, wobei die sogenannten „Bereiche", sowohl Oberflächen in drei Dimensionen als auch zweidimensional begrenzte Regionen umfassen:

- **Punkte (0D)**
- **Linien (1D)**
- **Bereiche (2D)**
- **Körper (3D)**

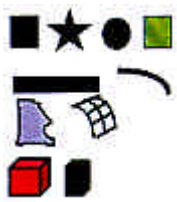

Abb. 3 Zeichentypen [Card et al. 1999]

Grafische Eigenschaften der Zeichen

Die grafische Eigenschaften der Zeichen tangieren eine Reihe unterschiedlicher Disziplinen. Entsprechend existieren diesbezüglich auch eine Reihe von Normen, Regeln und Gesetzten, wie beispielsweise diverse Gestaltungsregeln, Farblehren, Formung neuer Strukturen durch Anordnung, Eigenschaften der Wahrnehmung, temporale Kodierung, etc. Jedoch soll zu diesem Zeitpunkt hier nicht weiter darauf eingegangen werden, da dieser Bereich auch für sich genommen von hohem Umfang ist und weitschweifiger Ausführungen bedarf. Regeln und Guidelines hierzu finden sich in einer Vielzahl an Publikationen.

2.2.3 View Transformations

Viewtransformationen modifizieren und erweitern visuelle Strukturen interaktiv. Sie verwandeln, durch das Etablieren grafischer Parameter, statische Präsentationen in Visualisierungen, um so *Views* von visuellen Strukturen zu erzeugen. Dadurch nutzen Viewtransformationen den Aspekt der Zeit, der es ermöglicht, mehr Informationen von der Visualisierung zu extrahieren, als es bei statischen Präsentationen möglich wäre. Insgesamt lassen sich allgemeine Viewtransformationen unterscheiden:

Location Probes
Location Probes präsentieren zusätzliche Informationen aus der Datentabelle, indem sie spezifische Positionen in einer visuellen Struktur verwenden oder erweitern die visuelle Struktur selbst. Beispiele für *Location Probes* sind: *Details-on-Demand (Pop-up Windows), Brushing und Magic Lenses* oder *Moveable Filters*.

Viewpoint Control
Viewpoint Controls verwenden affine Transformationen, um den Blickwinkel durch *zooming, panning,* oder *clipping* zu ändern. Um Details besser darstellen zu können, vergrößern diese Transformationen visuelle Strukturen oder ändern den Blickwinkel. Dabei können *Viewpoint Controls* sowohl separiert, als auch in die Visualisierung integriert auftreten. Beispiele für *Viewpoint Controls* sind: *Zoom, Overview + Detail* und *Camera Movement.*

Distortion
Distortion modifiziert visuelle Strukturen um *Focus + Context Views* zu erzeugen, welche gewissermaßen *Overview + Detail* in einer einzelnen visuellen Struktur kombinieren. *Distortion* wird insbesondere dann verwendet, wenn dem Benutzer durch die Verzerrung größere entzerrte visuelle Strukturen dargestellt werden können. Dies erweist sich aber nur dann als effektiv, wenn die Eigenschaften oder Muster für die Aufgabe des Benutzers nicht nachteilig verzerrt werden. Beispiele für Distorsion sind: *Perspective Wall, Fisheye-View*.

2.2.4 Interaktions- und Transformationskontrollen

Der Part der menschlichen Interaktion schließt den Kreis zwischen visuellen Formen und der Kontrolle der Visualisierungsparameter, über welche die *Mappings* modifiziert werden. Die naheliegenste Form von Interaktion stellt die direkte Manipulation dar.

2.2.4.1 Transformationen

Wie bereits erwähnt, gibt es insgesamt drei Transformationsarten, auf welche der Benutzer mittels diverser Interaktionstechniken einwirken kann:

Raw Data → Data Tables
Diese Kontrollform ermöglicht die interaktive Kontrolle der *Data Mappings*. Die Veränderungen der Datentabellen kann zum Beispiel über *Range Slider* erfolgen.

Data Tables → Visual Structures

Es gibt zwei Möglichkeiten für die interaktive Kontrolle des *Mappings* von der Datentabelle in eine visuelle Struktur. Entweder wird diese in einem separaten User Interface angeboten oder in die visuelle Struktur integriert. Bei separaten User Interfaces erfolgt die Kontrolle zumeist über Beeinflussung visueller Repräsentanten der Datentabellen und visuellen Strukturen. Integrierte Lösungen erlauben dem Benutzer dagegen direkt auf Teile der visuellen Struktur zu klicken und das *Mapping* zu verändern.

Visual Structures → Views

Auch die interaktiven Kontrollen der *Views* können separiert oder in das Interface integriert werden. Dabei sind *Location Probes* und *Viewpoint* Manipulationen typischerweise integriert, *Distortion* Techniken können sowohl über ein externes User Interface angeboten werden, werden aber auch integriert.

2.2.4.2 Interaktionstechniken

Viele Interaktionstechniken sind im Wesentlichen eine Form der Selektion, d.h. eine Auswahl einer Untermenge von Objekten in der Datentabelle. Dies hat mehrerer Vorteile. So ermöglicht es z.B. mittels dieser Untermenge Daten zu lokalisieren, Muster in den Daten aufzudecken oder die Argumente anderer Transformationen zu selektieren, andere erlauben wiederum die Modifikation der *Data Transformations*.

Data Transformation

Für die Modifikation der *Data Transformation* stehen verschiedene Techniken zur Verfügung: *Dynamic Queries, Direct Walk, Details-on-Demand, Attribute Walk, Brushing* und *Direct Manipulation*.

Visual Mappings

Neben der der Modifikation der *Data Transformation* kann Interaktion auch visuelle *Mappings* modifizieren, welche die Übereinstimmung zwischen Daten und visuellen Formen repräsentieren. Zu diesen Techniken gehören: *Dataflow* und *Pivot Table*.

View Transformation

Auch auf die *View Transformation* kann der Benutzer mittels diverser Interaktionstechniken Einfluss nehmen. Dazu zählen: *Direct Selection, Camera Movement, Magic Lens, Overview* and *Detail und Zoom*.

2.3 Qualität von Visualisierungen

„The purpose of visualization is insight, not pictures."
[Stuart K. Card, Jock D. Mackinlay, Ben Shneiderman]

Die Thematik der Visualisierung von Daten ist, wie bereits erwähnt, ein breitgefächertes Feld, mit dem sich eine Vielzahl verschiedener Disziplinen beschäftigen und welches über zahlreiche Anwendungsmöglichkeiten in vielen unterschiedlichen Bereichen verfügt.
Auch wenn die Visualisierung von Daten prinzipiell viele Vorteile hat, so ist jedoch die Verwendung einer visuellen Darstellung, bzw. Visualisierung anstatt einer andersartigen Darstellungsform allein noch kein Garant für die Verbesserung der Darstellung der Daten selbst. Um die Vorteile von Visualisierungen ausnutzen zu können, ist es entscheidend wie qualitativ hochwertig eine visuelle Repräsentation ist. Allgemein gesagt bedeutet dies, wie gut kann sie den Benutzer in der Erfüllung seiner Aufgabe (*Task*) unterstützen, d.h. wie hoch ist der (kontextspezifische) Nutzen der Visualisierung für den Benutzer.

Denn nicht jede visuelle Darstellung ist geeignet, die in den Daten verborgenen Zusammenhänge für den Betrachter sichtbar zu machen. Unter Umständen kann eine Visualisierung auch zu einer Verschlechterung, d.h. Verschleierung relevanter Informationen bzw. zu einer subjektiven oder verfälschenden Darstellung der Daten führen, welches zu Verständnisproblemen bzw. falschen Interpretationen führen kann. Anschauliche Beispiele für solch qualitativ minderwertige Visualisierungen werden von einer Reihe von Forschern präsentiert. Zu den prominentesten Beispielen gehören dabei die Cholera Epidemie im September 1854 in London und der Start der Challenger am 28. Januar 1986, die Edward Tufte in seinem Buch *Visual Explanations* präsentiert [vgl. Tufte 1997]. Diese beiden extremen Situationen zeigen exemplarisch, welche Konsequenzen mit Entscheidungen verbunden sein können, die aufgrund mangelnder Qualität von Visualisierungen getroffen wurden.
Welche Eigenschaften bestimmen jedoch die Qualität einer visuellen Repräsentation und welche Faktoren beeinflussen diese Qualität? Da der Nutzen und somit auch die Qualität ebenfalls vom Bearbeitungsziel, also den Aufgaben des Benutzers abhängt, definiert sich die Qualität einer Visualisierung laut [Schumann&Müller 2000] allgemein wie folgt:

„Die Qualität einer Visualisierung definiert sich durch den Grad, in dem die bildliche Darstellung das kommunikative Ziel der Präsentation erreicht. Sie lässt sich als das Verhältnis von der vom Betrachter in einem Zeitraum wahrgenommenen Information zu der im gleichen Zeitraum zu vermittelnden Information beschreiben. Die Qualität einer Visualisierung ist somit in starkem Maße abhängig von den Charakteristika der zugrunde liegenden Daten und ihren Eigenschaften, dem Bearbeitungsziel, den Eigenschaften des Darstellungsmediums sowie den Wahrnehmungskapazitäten und den Erfahrungen des Betrachters." [vgl. Schumann et al. 2000, 7]

Ziel bei der Erstellung einer qualitativ hochwertigen Visualisierung ist also immer die Darstellung der Gesamtheit aller in den Daten verborgenen Informationen, die der Benutzer als Grundlage für Entscheidungen nutzen kann. Dabei muss eine Visualisierung prinzipiell sowohl expressiv, effektiv als auch angemessen sein. [vgl. Schumann et al. 2000]

Expressivität

Die Expressivität einer Visualisierung ist die Grundvoraussetzung einer jeden Visualisierung. D.h. sie muss die darzustellende Datenmenge möglichst unverfälscht wiedergeben. Demnach dürfen nur die in den Daten enthaltenen Informationen und tatsächlich nur diese durch die Visualisierung dargestellt werden. [vgl. Schumann et al. 2000]

Effektivität

Jedoch kann es für eine bestimmte Datenmenge durchaus mehrere grafische Darstellungen geben, welche das Kriterium der Expressivität erfüllen. Dabei werden sich die verschiedenen Darstellungsmöglichkeiten jedoch im Bezug auf ihre Effektivität unterscheiden, was bedeutet, dass manche der Visualisierungsmöglichkeiten die Daten für den Benutzer besser präsentieren als andere. Dies ist wiederum nicht nur von den Daten selbst, wie bei der Expressivität, sondern auch von verschiedenen weiteren Einflussfaktoren entscheidend abhängig. Dazu gehören die (visuellen) Fähigkeiten des Betrachters, die charakteristischen Eigenschaften des Ausgabegeräts, die Zielsetzung und der Anwendungskontext. Dabei spielt die Zielsetzung der Visualisierung die wichtigste Rolle. Ziel ist es die Visualisierung zu finden, welche bei der Darstellung die (visuellen) Fähigkeiten des Betrachters und die charakteristischen Eigenschaften des Ausgabegeräts unter Berücksichtigung der Zielsetzung und des Anwendungskontextes optimal ausnutzt. [vgl. Schumann et al. 2000]

Angemessenheit

Die Angemessenheit einer Visualisierung bezieht sich auf die Kosten, bei denen sich Aufwand und Nutzen gegenüberstehen. Sie umfasst den Rechen- und Ressourcenaufwand, der zur Generierung der visuellen Darstellung notwendig ist. Die Angemessenheit einer Visualisierung beschreibt also den Aufwand und die Kosten, die zur Durchführung des Visualisierungsprozesses notwendig sind. Damit beschreibt die Angemessenheit also weniger die Qualität der resultierenden graphischen Darstellung einer Visualisierung und somit den Aufwand für die Interpretation durch den Betrachter im eigentlichen Sinne. Effektivität und Angemessenheit bei der Visualisierung sind in der Praxis oft eng miteinander verknüpft. Denn der Aufwand zur Generierung einer Visualisierung hängt in den meisten Fällen auch mit zeitlicher, physischer oder auch kognitiver Belastung des Benutzers, die wiederum ebenfalls in Kosten ausgedrückt werden kann, zusammen. Demnach können unangemessene Visualisierungen kaum effektiv sein. [vgl. Schumann et al. 2000]

Die Herausforderung bei der Entwicklung einer Visualisierung liegt also darin, die oftmals propagierten Vorzüge der visuellen Darstellung von Daten - Unterstützung der menschlichen Wahrnehmung bei der Informationssuche und Hilfestellung beim Verstehen und Erkennen von Informationsstrukturen [Lin 1998] - mit einer konkreten Visualisierung auch zu erreichen. D.h. eine qualitativ hochwertige Visualisierung muss die Kriterien der Expressivität, der Effektivität und der Angemessenheit erfüllen. Ziel beim Design einer Visualisierung ist es also, unter Beachtung der verschiedenen Einflussfaktoren, ein qualitativ hochwertiges Mapping der Daten in visuelle Repräsentationen zu finden, mit dem Ziel, den größtmöglichen Nutzen für den Benutzer zu erreichen.

3 Design von Visualisierungen

Visualisierungen helfen Benutzern große Mengen an komplexen Daten zu bewältigen. Sie agieren als Tools, die verwendet werden können um die Ziele zu erreichen, die für den jeweiligen Bereich in dem sie angewendet werden relevant sind. Jedoch ist der Entwurf solcher Visualisierungen, wie auch allgemein das Design von interaktiver Software eine komplexe und anspruchsvolle Aufgabe. Die Daten, die Aufgaben, der Anwendungsbereich und die Benutzer sind nur einige Faktoren, die man beim Design von Visualisierungen berücksichtigen muss. Visualisierungen nutzen ebenso Techniken aus den Bereichen wie der Computergrafik, der Human-Computer Interaction, der Wahrnehmungspsychologie, dem Data Mining, dem Software Engineering und dem Grafik Design. Weitere Disziplinen bzw. Personen, die an der Entwicklung beteiligt sind, sind u.a. auch das Marketing, das Technical Writing bzw. Experten bzw. Benutzer aus dem zukünftigen Anwendungsbereich. Dieser Umfang an Wissen und dessen Komplexität, das der Bereich der Visualisierung erfordert macht das Design von qualitativ hochwertigen Visualisierungen zu einer schwierigen Aufgabe. [Wilkins 2003]

Die Komplexität der Designaufgabe erfordert die Beteiligung verschiedener Disziplinen, also ein interdisziplinäres Designteam für das Design von Visualisierungen, auch Visualization Design genannt. Ein solches Designteam kann sich, je nach Projekt, aus Personen der verschiedenen, oben genannten Bereichen zusammensetzen. Jedoch verfügt jede der beteiligten Disziplinen über ihre eigenen Methodologien, welche wiederum ihre eigenen Vorgehensweisen, Techniken bzw. Schwerpunkte für das Vorgehen beim Design von Software beinhalten. Insbesondere beim Verständnis dafür, was eine qualitativ hochwertige Software ausmacht, legt jede dieser Disziplinen ihren Fokus auf unterschiedliche Kriterien.

Da sowohl das Software Engineering (SE) als auch die Human-Computer Interaction (HCI) zu den wichtigsten beteiligten Disziplinen beim Visualization Design gehören, werden nun einige Methodologien, die in diesem Bereich für das Design von Software verwendet werden, kurz aufgeführt. Im Anschluss daran, wird eine Methodologie für das Visualization Design vorgestellt, welche auf der generischen Designmethodologie, welche sich aus Methodologien des Software Designs ableiten lässt, basiert.

3.1 Methodologien Software Design

3.1.1 Methodologie Software Engineering

Das Software Engineering setzt den Fokus auf die Objekte, die entworfen werden sollen - die Hardware und Software - und konzentriert sich damit auf das Verständnis von computerbezogenen Mechanismen und Entwicklungsmethoden, welche die korrekte Funktionsweise des Objekts sichern. Das Hauptanliegen von Software Entwicklern ist es eine spezifische Funktionalität effizient zu implementieren. Der Fokus beim Design von Software liegt dabei auf der korrekten Funktionalität der Software, also dass sie stabil und zuverlässig ist und ihre funktionale Spezifikation erfüllt; also auf elegantem, robustem Code, welcher leicht gepflegt und erweitert werden kann. [Winograd 2002]

Es gibt im Software Engineering verschiedene Ansätze für die Entwicklung von Software, wie z.B. das *Waterfall Life Cycle Modell*, eines der ersten Modelle mit einem im Wesentlichen linearen Ansatz, das beim Software Engineering allgemein bekannt geworden ist und das die Basis vieler *Life Cycles*, welche heutzutage in der Entwicklung verwendet werden bildet. Das Modell besteht aus insgesamt fünf Schritten – *Requirements Analysis, Design, Code, Test* und *Maintenance* – von denen jeder komplett beendet sein muss, bevor der folgende begonnen werden kann. Die Bezeichnung der einzelnen Schritte kann allerdings variieren. Die Idee der Iteration war ursprünglich im Modell nicht vorgesehen, findet sich aber heutzutage in den meisten Versionen des *Waterfall Modells*. *Review Sessions* unter Entwicklern sind heutzutage allgemein üblich, sieht jedoch zumeist keine Überprüfung und Evaluation durch die Benutzer vor.
1988 entwickelte dann Barry Boehm das *Spiral Life Cycle Model of Software Development*. Das *Spiral Model* ist ein iteratives Framework, das es ermöglicht, Ideen und Abläufe/Fortschritte schnell zu überprüfen und zu evaluieren. Jede Iteration um die Spirale herum kann auf einem anderen *Life Cycle Modell* basieren und kann verschiedene Aktivitäten umfassen. Weitere Modelle im Software Engineering sind z.B. das *Rapid Applications Development (RAD) Life Cycle Model of Software Development*, das *Star Model* oder das *Reuse Model*, um nur einige zu nennen. [Preece et al. 2002]

3.1.2 Methodologie Human-Computer Interaction

Reine Softwareansätze für das Design, wie sie im vorherigen Abschnitt beschrieben wurden, tendieren dazu, mehr auf die Daten und die Funktionalität des Systems zu fokussieren als auf den Benutzer. Jedoch ist dieser reine Fokus auf Funktion und Konstruktion sehr einseitig. Denn eine wichtige Komponente bei der Entwicklung von Software ist der zukünftige Benutzer, der später dann mit dieser Software interagieren soll. Software Design ist ein Bereich, der benutzerorientiert ist und wird daher auch immer eher die menschliche Offenheit von Disziplinen wie der Architektur und dem Grafik Design

haben, als die formelhafte Bestimmtheit von technischorientiertem Ingenieursdesign. [Winograd 2002] Ein Ansatz, der dieses bei dem Design von Software beachtet, ist die Human-Computer Interaction (HCI), die Methodologien für das Design von Software entwickelt hat, die speziell auf die Erfüllung der Benutzerbedürfnisse fokussieren, bzw. auf die Benutzerfreundlichkeit der Software (*Usability*). Die Human-Computer Interaction Methoden helfen die Anforderungen und Aufgaben von Benutzern zu eruieren, Interfaces zu gestalten und zu evaluieren und den Benutzer zu modellieren. Entsprechend haben die *Life Cycle Modelle* aus dieser Disziplin einen stärken Fokus auf die Benutzer des zukünftigen Systems.
1989 schlugen Hartson und Hix eines der ersten *Life Cycle Modelle* für die HCI vor, das *Star Life Cycle Model*. Es entstand aus den Beobachtungen, die sie bezüglich der Vorgehensweise von Interface Designern, gemacht haben. Sie identifizierten dabei zwei verschiedene Modi an Aktivität: den Analytischen Modus und den Synthetischen Modus. Der analytische Modus wird durch Begriffe wie, *top-down*, *organisierend*, *bewertend* und *formal* charakterisiert - sich gewissermaßen von der Systemsicht zur Benutzersicht vorarbeitend. Auf den Synthetischen Modus treffen dagegen Begriffe zu, wie *bottom-up*, *freies Denken*, *kreativ* und *ad hoc* - sich also genau entgegengesetzt von der Benutzersicht zur Systemsicht vorarbeitend. Interface Designer wechseln während des Designvorgangs laufend von einem Modus zum anderen. Ein ähnliches Verhalten wurde auch bei Software Designern beobachtet. [Preece et al. 2002]
Das *Star Life Cycle Model* besteht aus den Aktivitäten *Implementation, Task Analysis / Functional Analysis, Evaluation, Prototyping, Requirements / Specification* und *Conceptual Design / Formal Design Representation*. Jedoch gibt es keine spezifische Vorgehensweise innerhalb des Modells. Die einzelnen Aktivitäten sind jedoch stark miteinander verknüpft und die Phase der Evaluation ist zentral für alle Aktivitäten. D.h. prinzipiell kann man von jeder Aktivität zu einer beliebigen anderen wechseln, jedoch führt der Weg stets über die Phase der Evaluation. Mit welcher Phase begonnen wird, ist jedoch frei wählbar. [Preece et al. 2002]

Ein wesentlich bekannteres *Life Cycle Modell* aus dem Bereich der HCI ist das *Usability Engineering Life Cycle* von Deborah Mayhew von 1999 [Mayhew 1999]. Es bietet einen ganzheitlichen Blick auf das Usability Engineering und eine detaillierte Beschreibung wie Usability Aufgaben durchgeführt werden und spezifiziert, wie Usability Aufgaben (*Tasks*) in die traditionellen *Life Cycles* der Softwareentwicklung integriert werden können. Beispielsweise sind die verschiedenen Stadien des *Life Cycles* mit einem allgemeinen Entwicklungsansatz, dem *Rapid Prototyping* und einer spezifischen Methode, dem *Object-Oriented Software Engineering (OOSE)*, die aus dem Software Engineering stammt, verknüpft. Der *Usability Life Cycle* selbst hat im Wesentlichen drei Aufgaben. *Requirements Analysis, Design / Testing / Development* und *Installation*. Dabei umfasst die zweite Aufgabe mehrere Unteraufgaben und stellt damit die umfangreichste der drei Aufgaben dar. Es umfasst Stadien für das Identifizieren der Anforderungen, für das Design, die Evaluation und das Bauen von Prototypen. Außerdem beinhaltet der *Life Cycle* einen *Style Guide*,

welcher als Mechanismus dient, die Usability Ziele des Projekts zu definieren und publik zu machen. [Preece et al. 2002]
Weitere *User-centered* Designansätze sind z.B. *Method for Usability Engineering (MUSE)*, *TRIDENT Methodology, Graphical User Interface Design and Evaluation (GUIDE) Method oder die UNION Methodology.*

3.1.3 Generische Design Methodologie

Wie im oberen Abschnitt exemplarisch gezeigt, arbeiten die verschiedenen Disziplinen, mit den unterschiedlichsten Methoden und Entwicklungs-, Vorgehens- oder Prozessmodellen, die den Entwicklungszyklus der Software beschreiben. Wie Barry Wilkins in seiner Dissertation *MELD: A Pattern Supported Methodology for Visualisation Design* nach Analyse diverser Methodologien aus dem Bereich des Software Design festgestellt hat, liegt diesen Methodologien ein gemeinsames generisches Basismodell, genannt Generische Designmethodologie (vgl. Abb. 4), zugrunde. Er betont, dass jede der von ihm untersuchten Methodologien die folgenden vier Phasen beinhaltet - Analyse, Design, Implementierung und Evaluation - die Techniken, die zur Durchführung der einzelnen Phasen verwendet werden jedoch Disziplinspezifisch sind. Diese Phasen können dabei stets iterativ durchlaufen werden. Diese Techniken und die Wertesysteme, wonach die jeweilige Methodologie ausgerichtet ist, sind die Aspekte worin sich die jeweiligen Methodologien voneinander unterscheiden. Jede Disziplin hat ihr eigenes Wertesystem, auf das die jeweilige Methodologie ausgerichtet ist und setzt zur Erreichung dieses Wertesystems unterschiedliche Techniken ein. Entsprechend sind auch die einzelnen Phasen je nach Methodologie unterschiedlich stark gewichtet.

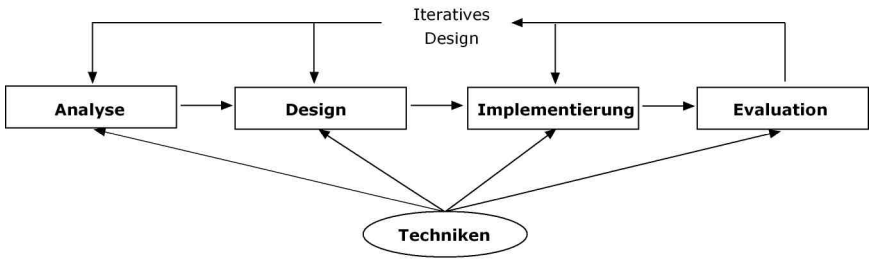

Abb. 4 Generische Design Methodologie [vgl. Wilkins 2003]

So gehören zu den Techniken, die beispielsweise beim Software Engineering in der Evaluationsphase angewendet werden u.a. das *Black-Box Testing, Dry Runs, Code Walkthroughs* oder *Unit-Testing*. Zu den Techniken, die in dieser Phase in der HCI angewendet werden dagegen *Interviews, Questionnaires, Cognitive Walkthoughs* oder *Usability Tests*. Wie Wilkins offen legte, hängen die Techniken, die bei jeder Methodologie zum

Einsatz kommen, stark vom jeweiligen Wertesystem ab, auf welches die Methodologie zugeschnitten ist. Bezüglich der Wertesysteme sind Methodologien, die im Software Engineering eingesetzt werden, darauf ausgerichtet möglichst eleganten und robusten Code zu erzeugen, der einfach erweitert und gepflegt werden kann. In der HCI wird dagegen die Entwicklung einer Software angestrebt, die über eine hohe Usability verfügt, also besonders benutzerfreundlich ist und somit die Benutzerbedürfnisse möglichst optimal unterstützt. Entsprechend sind auch die Phasen der jeweiligen Methodologien in den einzelnen Disziplinen verschieden stark ausgeprägt. [Wilkins 2003]

3.1.4 Methodologie Visualization Design

Im Bereich der Visualisierung werden, wie in Kapitel 2.2 beschrieben, primär Referenzmodelle verwendet, um die Erzeugung einer Visualisierung zu beschreiben. Jedoch fokussieren diese primär auf den Visualisierungsprozess und nicht auf das Visualisierungsdesign. Sie bieten für den Visualization Designer keine Anleitung zur Vorgehensweise bei der Entwicklung einer Visualisierung oder helfen nicht bei Designentscheidungen, wie beispielsweise der Wahl der geeignetsten Transformation, visuellen Struktur oder Interaktionstechnik. Referenzmodelle sind für den Designprozess jedoch insoweit nützlich, als dass sie es ermöglichen die komplexe Aufgabe des Visualization Design in Teilbereiche aufzuteilen, da sie alle wesentlichen Gestaltungskomponenten beinhalten. So ermöglichen sie beispielsweise eine Aufteilung des komplexen Gestaltungsbereichs in überschaubare Einzelkomponenten und beinhalten die Abfolge und Abhängigkeiten der einzelnen Teilschritte bei der Erzeugung einer Visualisierung. [Reiterer 2004]
Das Design von Visualisierungen - Visualization Design genannt - ist eine spezifische Form des Software Designs. Demnach liegt die generische Designmethodologie auch als Basisform einer visualisierungsspezifischen Methodologie für den Designprozess einer Visualisierung zugrunde. Barry Wilkins schlägt für das Design von Visualisierungen eine visualisierungsspezifischen Methodologie vor die aus den vier Phasen der generischen Designmethodologie für Software Design besteht und einer zusätzlichen visualisierungsspezifischen Phase, der Datenerfassung, welche dem eigentlichen Designprozess vorausgeht (vgl. Abb. 5).

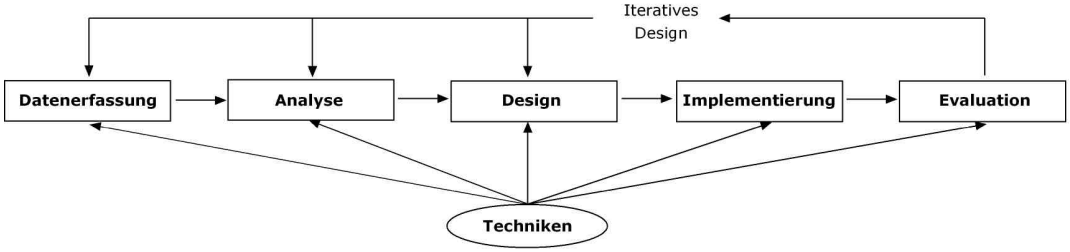

Abb. 5 Visualisierungsspezifische Methodologie [vgl. Wilkins 2003]

3.1.4.1 Datenerfassung

Diese erste Phase der Datenerfassung ist notwendig, da eine Visualisierung eine visuelle Repräsentation von Daten darstellt und deshalb vor dem eigentlichen Design der Visualisierung diese gesammelt und strukturiert werden müssen. Daten, die visualisiert werden sollen, können jedoch sehr heterogen sein. Dabei kann es sich ebenso um Messdaten bezüglich der jährlichen Flächenreduktion der Polarkappen, um die Verbraucherdaten einer großen Handelkette als auch um das Netzwerk der Pariser Metro oder die Baukomponenten eines Automobils handeln. [Spence 2001] D.h. Daten können aus einer Vielzahl unterschiedlicher Quellen stammen und entsprechend unterschiedliche Formate haben. Daher müssen diese Rohdaten zunächst strukturiert werden, bevor sie in einer Visualisierung dargestellt werden können (vgl. Kapitel 2.2.1). So wird beispielsweise der Rohtext in einer Dokumentensammlung als eine Gruppe von Dokument-Feature-vektoren repräsentiert oder medizinische Daten in spezifische Gruppen strukturiert. Ziel der Rohdatentransformation ist es, die Rohdaten in irgendeine vordefinierte, typischerweise in Datentabellen, zu konvertieren. Datentabellen haben den Vorteil, dass diese in Datenbanken gespeichert werden können, welche z.B. schnelles Suchen oder das Zusammenführen von Tabellen ermöglichen. [Wilkins 2003]

3.1.4.2 Analyse

Für das Design einer Visualisierung muss bestimmt werden, welche Transformationen angewendet werden sollen, um die strukturierten Daten in visuelle Strukturen zu konvertieren, als auch welche Interaktionstechniken eingesetzt und wie jede einzelne Visualisierung oder View angelegt werden soll.
Dazu ist es notwenig, zunächst die Hauptfaktoren, welche für eine Visualisierung relevant sind, zu analysieren. Zu diesen Faktoren gehören laut [Wilkins 2003]: die Daten, die Aufgaben (*Tasks*), die Usability Faktoren / Usability Ziele und die Benutzercharakteristiken. Die Analysephase hebt die Randbedingungen hervor, die für jeden einzelnen dieser Faktoren existieren und welche dann vom Visualization Designer abgewogen werden müssen.

1. Daten

Daten werden in den verschiedensten Bereichen gemessen, beobachtet oder berechnet. Entsprechend unterscheiden sich Daten hinsichtlich ihrer Struktur, ihrer Dimensionalität und anderen wichtigen Eigenschaften. Diese Eigenschaften sind für die Visualisierung äußerst relevant, weil damit visuelle Darstellungen erzeugt werden können, welche das Kriterium der Expressivität erfüllen und damit für den Benutzer intuitiv interpretierbar sind. Daher sind die Charakteristika der Daten selbst, sowie der Bezugsraum, in dem die Daten erhoben wurden - die Datenquelle - für das Design von Visualisierungen relevant.

[Schumann&Müller 2000] Zu der Analyse der Daten gehören beispielsweise die Betrachtung der Datenquelle, der Datentyp, die Variablentypen oder die Quantität der Daten.

- **Datenquelle**
  Datenquellen können ganz allgemein in drei verschiedene Typen unterschieden werden: die reale Welt – hier werden die Daten durch Messgeräte erfasst oder durch Beobachtungen gewonnen, die theoretische Welt, in der die Daten auf der Grundlage mathematischer Modelle berechnet werden und die künstliche Welt, deren Daten entworfen werden z.B. für Film, Kunst und Fernsehen. [Schumann&Müller 2000] Die Kenntnis von der Datenquelle kann beispielsweise ein nützlicher Startpunkt für die ersten visuellen Strukturen sein, die in der Visualisierung verwendet werden. Ein Beispiel hierfür wäre, den Text eines Dokuments unter Verwendung des Layouts des Originaldokuments darzustellen oder die Daten eines Verkehrsflusses auf der Karte der Stadt, in der sie gesammelt wurden, zu kodieren. D.h. das physische Objekt von dem die Daten gesammelt wurden, kann als Template für die visuelle Struktur verwendet werden, die in der Visualisierung verwendet wird.
  Ein ähnlicher Ansatz birgt die Idee der Metapher, eine Technik, die Objekte und Konzepte, die dem Benutzer bekannt sind, für das Design der Visualisierung verwendet. [Wilkins 2003] Bekannte Metaphern sind beispielsweise die Buch-Metapher oder die Landschafts- oder Raummetapher mit deren Hilfe es dem Benutzer erleichtert werden soll, sich in dem Softwaresystem zu orientieren und zu navigieren.

  Das Analysieren der Datenquelle und das Anwenden einer Metapher kann dem Designer helfen eine Visualisierung zu entwickeln, die dem mentalen Modell des Benutzers entspricht und daher leichter erlernt und begriffen werden kann. [Wilkins 2003] Denn Metaphern haben das Ziel die Komplexität des User Interfaces anhand von spezifischem Wissen, welches Benutzer zu einem früheren Zeitpunkt in anderen Bereichen erworben haben, zu minimieren.
  Thomas Mann präsentiert in seiner Dissertation *Visualization of search results from the World Wide Web* [vgl. Mann 2002] beispielsweise eine ganze Reihe gängiger Metaphern und führt dazu Systeme an, welche sich dieser Metaphern bedienen.

- **Datentyp**
  Der Datentyp und die Datenquelle hängen stark miteinander zusammen. So stellt der Datentyp eine Klassifikation der Quelle dar, unabhängig von ihrem Ursprungsgebiet. Jedoch ist bereits die Terminologie von Daten in der Literatur nicht konsistent, da diese von einer Vielzahl von Disziplinen erzeugt werden – Mathematik, Statistik, Ingenieurswissenschaften oder den Computerwissenschaften. [Card et al. 1999] Entsprechend existiert auch keine eindeutige Klassifizierung von Daten, da dies stark mit der Klassifizierung von Wissen zusammenhängt, welches eine sehr sensible Aufgabe darstellt [Ware 1999]. Aus diesem Grund gibt es verschiedene Ansätze in der Literatur, welche die Klassifizierung von Daten behandeln. Einer der bekanntesten aus dem

Bereich der Informationsvisualisierung stammt von Ben Shneiderman, der die so genannte *Task by Data Type Taxonomy for Information Visualization*, kurz *TTT* entwickelte, welche die Datentypen von Visualisierungen und die Aufgaben, die durch sie unterstützt werden müssen, identifiziert.

[Shneiderman 1996] setzt für seine Taxonomie, die er 1996 in seinem Paper *The Eyes Have It: A Task by Data Type Taxonomy of Information Visualizations* publizierte, voraus, dass Benutzer Objektsammlungen betrachten, in denen die Objekte mehrere Attribute besitzen. Attribute eines Objektes können zum Beispiel die Länge eines Spielfilms oder die Sprache sein. Bei allen sieben Datentypen der Taxonomie - eindimensionale (1D), zweidimensionale (2D), dreidimensionale (3D), temporale, multidimensionale (MultiD), Baumstrukturierte (Tree) und Netzwerk Daten - besitzen die Objekte Attribute und das Selektieren aller Objekte, welche einer Gruppe von Attributen entsprechen, stellt eine grundsätzliche Suchaufgabe dar.

So werden beispielsweise lineare Daten, die zumeist in Form von – oftmals textbasierten – sequentiellen Listen auftreten als eindimensional klassifiziert. Dies können beispielsweise Textdokumente, Programmquellcode, Programmverzeichnisse, Dokumente mit Suchergebnissen oder alphabetische Namenslisten sein. [Shneiderman 1996] Flächen- oder Kartenbasierte Daten, wie z.B. geografische Karten, Grundrisse, Lagepläne oder Layouts von Zeitungen werden als zweidimensional klassifiziert, Daten von räumlichen oftmals physikalischen Objekten, wie beispielsweise Molekülen, dem menschlichen Körper oder Gebäuden als dreidimensional. Daten, die über einen Zeitraum hinweg gesammelt wurden können als temporal klassifiziert werden. Dies können beispielsweise Zeitachsen sein, wie sie in Krankenblättern, im Projektmanagement oder in historischen Repräsentationen verwendet werden. Jedoch können Daten, die aus einer einzelnen Quelle stammen, auch aus multiplen strukturellen Klassifikationen bestehen. Solche Daten werden als multidimensional klassifiziert. Dies sind zumeist relationale oder statistische Datenbestände, bei denen jedes Objekt mehr als drei Attribute besitzt. [Shneiderman 1996] Beispiele für Anwendungen von multidimensionalen Datentypen können Aktienmarkt-Statistiken, eine Gruppe von Büchern in einer Bibliothek oder eine Filmdatenbank sein. Objektsammlungen, in denen jedes einzelne der Objekte in einer hierarchischen Beziehung zu einem anderen Objekt steht, werden mit dem Datentyp Tree klassifiziert. Hierarchien tauchen natürlicherweise in Taxonomien, in Strukturen von Organisationen, bei der Verwaltung von Speicherplatz, in der Genealogie und dem Dewey Dezimalsystem auf. [OLIVE 2005] Der Datentyp, bei dem die Objekte zu einer beliebigen Anzahl anderer Objekte in Verbindung stehen können, wird Netzwerk genannt. [Wilkins 2003][Mann 2002] Netzwerk-Visualisierungen werden oftmals benutzt um Kommunikationsnetzwerke wie Telefonsysteme oder das World Wide Web zu beschreiben. [Card et al. 1999]

Variationen zu diesen Datentypen können beispielsweise 2 ½ oder 4D Daten aber auch so genannte *Multi-Trees*, sein. Darüber hinaus kommt es auch häufig vor, dass Visualisierungen Kombinationen dieser Datentypen verwenden. [Shneiderman 1996]

- **Variablentyp**
  Während die Datenquelle und die Datentypen zu Datensätzen verweisen, verweist der Variablentyp zu den individuellen Attributen von jedem Objekt, wie z.B. den Eigenschaften des Objekts, die ein Datenobjekt repräsentiert. Variablen können dabei prinzipiell in drei Basistypen unterschieden werden: nominale, ordinale und quantitative Variablen. Eine nominale Variable stellt dabei eine ungeordnete Gruppe dar, wie z.B. Filmtitel; eine ordinale Variable eine geordnete Gruppe, wie beispielweise Filmbewertungen oder Wochentage; eine quantitative Variable einen numerischen Bereich, wie z.B. Spielfilmlängen. Darüber hinaus können zum Beispiel quantitative Variablen in ordinale Daten transformiert werden und weiter auch ordinale Variablen in nominale. Dies führt zu einer weiteren Unterscheidung von Variablentypen: *Classing* ist eine allgemeine Transformation, welche Werte in Klassen von Werten einteilt. Dabei kann es zu einem Wechsel des Variablentyps kommen, wie beispielsweise bei der Unterteilung der Spielfilmlänge in Klassen: [0,360] in <Short, Medium, Long>. Zusätzlich wird noch zwischen wichtigen Subtypen, wie räumlichen oder geografischen quantitativen Daten oder quantitativer oder ordinaler Zeit, unterschieden. Diese prinzipielle Unterscheidungen sind wichtig, da sie den Typ der Achse bestimmen, welcher bei der visuellen Struktur verwendet werden soll, bzw. da die Subtypen als Eigenschaften der realen Welt normalerweise mit speziellen visuellen Konventionen verbunden werden. [vgl. Card et al. 1999]

- **Quantität der Daten**
  Die Quantität der Daten hat einen signifikanten Einfluss darauf, welche visuelle Struktur und welche Interaktionstechniken sich für die Visualisierung eignen, da sich nicht jede Ausprägung auch für alle Datenmengen eignet. [Wilkins 2003]

2. Aufgaben (Tasks)

Die Analyse der Aufgaben der Benutzer bestimmt die Ziele, die der Benutzer mit Hilfe der Visualisierung versucht zu erreichen und die Methoden, die er dafür einsetzt. Entsprechend kann der Visualization Designer die identifizierten Aufgaben und Aufgabenprozesse verwenden, um sowohl die Daten als auch die benötigten Interaktionsmechanismen zu bestimmen, die benötigt werden um die Benutzer bei der Erfüllung der Aufgaben bestmöglich zu unterstützen.

Aufgaben, die der Benutzer mit Hilfe von Visualisierungen erfüllen möchte, können in verschiedene Kategorien klassifiziert werden. Eine Reihe von Forschern, wie z.B. Ben Shneiderman (1996), Barry Wilkins (2003), Heidrun Schumann und Wolfgang Müller (2000) (vgl. Bearbeitungsziele) haben eine Reihe von Klassifikationsmöglichkeiten für diese Aufgaben vorgeschlagen.

Zu den bekanntesten Klassifizierungsmöglichkeiten von Aufgaben gehört die von Ben Shneiderman entwickelte *Task by Data Type Taxonomy for Information Visualization*, kurz *TTT*, die bereits oben erwähnt wurde. Im Rahmen dieser Taxonomie entwickelte Shneiderman insgesamt sieben Aufgaben, die sich prinzipiell auf einem hohen Grad der Abstraktion befinden, die Visualisierungen prinzipiell unterstützen sollten: *Overview*, *Zoom*, *Filter*, *Details-on-Demand*, *Relate*, *History* und *Extract*. *Overview* bietet dabei einen Überblick über den gesamten Datensatz; *Zoom* ermöglicht das Hineinzoomen in Objekte, die von Interesse sind; *Filter* ermöglicht das Herausfiltern von uninteressanten Objekten; *Details-on-Demand* bieten die Funktion ein einzelnes Objekt oder eine Gruppe von Objekten auszuwählen und Details dazu abzurufen, wenn diese benötigt werden; *Relate* zeigt die Beziehungen der Objekte untereinander; *History* dokumentiert den Verlauf der Aktionen um die Möglichkeit zu bieten, diese wieder rückgängig zu machen, zu wiederholen und sie schrittweise zu verfeinern; *Extract* ermöglicht die Extraktion von Untergruppen des Datensatzes und der Anfrageparameter. [Shneiderman 1996] Die Kategorie der Aufgabe, die eine Visualisierung unterstützen muss, bestimmt die Gestaltung einer Visualisierung. [Wilkins 2003]

3. Usability Faktoren / Usability Ziele

In der HCI wird die Effektivität eines GUIs durch das Messen einer Reihe von Usability Faktoren bestimmt. Dabei gibt es eine Reihe unterschiedlicher Usability Faktoren, welche für die Bewertung der Benutzerfreundlichkeit von Systemen verwendet werden. Dies kann beispielsweise die Zeit sein, die ein Benutzer benötigt um ein System zu erlernen, die Geschwindigkeit mit welcher die Aufgaben vollständig erledigt werden, die Anzahl und Rate der Fehler, die der Benutzer dabei macht, wie gut sich der Benutzer an das System erinnern kann oder wie zufrieden der Benutzer allgemein mit dem System ist. Insgesamt muss die Visualisierung dem mentalen Modell und den Erwartungen des Benutzers entsprechen und die Usability Ziele für die Visualisierung klar definiert und gewichtet werden. [Wilkins 2003] Zu den bekanntesten Usability Faktoren gehört die Norm für HCI und Usability, die ISO 9241, welche die Gestaltung von Bildschirmarbeit und –plätzen beschreibt und Auskunft gibt über die Anforderungen an Mensch-Computer-Schnittstellen. Laut dieser Norm gehören zu den Grundsätzen der Dialoggestaltung - auch Kriterien der Benutzerfreundlichkeit genannt - Aufgabenangemessenheit, Selbstbeschreibungsfähigkeit, Erwartungskonformität, Lernförderlichkeit, Steuerbarkeit, Fehlertoleranz und Individualisierbarkeit. [vgl. Reeps 2004] Beim Design einer Visualisierung müssen die definier-

ten Usability Faktoren beachtet und entsprechend die Usability Ziele für die Visualisierung definiert werden.

4. Benutzercharakteristiken

In der HCI wurden Techniken entwickelt, die speziell darauf ausgerichtet sind Benutzermodelle zu entwickeln, die beispielsweise den Grad ihrer Fachkenntnis bezüglich dem Umgang mit Software oder ihre Präferenzen und physischen Fähigkeiten bestimmen. Einige dieser Benutzerattribute hängen mit den Usability Faktoren zusammen, so brauchen Benutzer auf dem Anfängerlevel länger um ein System zu erlernen und sind fehleranfälliger, wogegen andere eventuell unter Farbenblindheit leiden (rot-grün). Entsprechend müssen diese Charakteristiken beim Design der Visualisierung beachtet werden. [Wilkins 2003]
Diese detaillierte Analyse definiert die Struktur und den Inhalt der Visualisierung, die Aufgaben, welche die Visualisierung unterstützen muss, die Gruppe der zukünftigen Benutzer und die Usability Ziele, welche die Visualisierung erfüllen muss. [Wilkins 2003]

3.1.4.3 Design

Der Visualisierungsprozess, der Daten in eine *View* konvertiert, wurde in Kapitel 2.2 vorgestellt. Im Wesentlichen besteht der Prozess aus einer Anzahl von Transformationen und den Interaktionstechniken, mit deren Hilfe der Benutzer auf die Transformationen einwirken kann. Nach der Analyse der Faktoren, welche beim Design einer Visualisierung beachtet werden sollten, werden in der Designphase die Komponenten, die bei einer Visualisierung gestaltet werden müssen, ermittelt und definiert. Zu diesen Komponenten zählen die Transformationen, die eingesetzt werden sollen, die Interaktionstechniken und die Kompositionselemente, die für mehrfach verlinkte Visualisierungen erforderlich sind. Deren Ausprägungen definieren die grafische Szene und wie der Benutzer damit interagieren kann – entweder, um sie zu initiieren oder sie zu modifizieren. Bei multiplen *Views* oder multiplen Visualisierungen muss der Designer sowohl das Layout dieser *Views* als auch jede Kommunikation zwischen den Visualisierungen gestalten.

Zu den Transformationen, die der Designer gestalten muss, gehören die *Data Transformations*, das *Visual Mapping* und die *View Transformations*. Die Datentransformation bestimmt den Inhalt der Visualisierung, wogegen die anderen beiden Transformationsarten die visuellen Aspekte der Visualisierung modifizieren. Die Gestaltung der Interaktionstechniken definiert dagegen, welche Interaktionsmöglichkeiten der Benutzer mit der Visualisierung hat und wie die daraus resultierenden Transformationen aussehen.

In der Designphase ist es hilfreich, wenn der Designer durch geeignete Techniken unterstützt wird, welche ihm effektives Designwissen anbieten, um zu qualitativ hochwertigen Designergebnissen zu gelangen. [Wilkins 2003]

3.1.4.4 Implementierung

Nachdem eine Reihe von Entwurfsvorschlägen erarbeitet wurden, kann mit der Implementierung der Visualisierung begonnen werden. Dazu kann jede der üblichen Software Engineering Methoden, wie in Kapitel 3.1.1 beschrieben, sowie auch der Einsatz von Software Patterns verwendet werden um die Visualisierung zu implementieren. [Wilkins 2003] In diesem Stadium können aber auch zunächst mal, je nach Projekt, Prototypen bzw. auch Paper-Mockups entstehen, aus denen erst zu einem späteren Zeitpunkt die eigentliche Implementierung der Visualisierung entsteht.

3.1.4.5 Evaluation

Nachdem ein oder mehrere Visualisierungsentwürfe erarbeitet und implementiert wurden, erfolgt die Evaluation des Entwurfs, um zu überprüfen ob er in der Lage ist, den Benutzer bei seinen Aufgaben zu unterstützen und ob alle gewünschten Usability Faktoren erfüllt werden. In der HCI wurden eine Reihe von Evaluationstechniken entwickelt, wie z.B. *Cognitive Walkthroughs, Focus Groups, Usability Inspection, Heuristic Evaluation*, etc., welche je nach Bedarf in der Evaluationsphase eingesetzt werden können. Die Entscheidung ist auch häufig von den Rahmenbedingungen des Projekts, wie Zeit- und Kostenaspekten, Typen von Benutzern und den Usability Faktoren an denen man am meisten interessiert ist, abhängig.

3.1.4.6 Iteratives Design

Wie in Abb. 5 zu sehen war, ist die Entwicklung einer Visualisierung ein iterativer Prozess. D.h. die Evaluation kann wieder zu früheren Phasen innerhalb des Designprozesses, wie der Datenerfassung, der Analyse oder dem Design zurückführen. In früheren Phasen des Designprozesses, nach der Datenerfassung und anfänglichen Analysen, sollten Prototyping und Tests mit Benutzern als Möglichkeit verwendet werden, Anforderungen an die Visualisierung zu etablieren und Belange hinsichtlich der Benutzerfreundlichkeit zu identifizieren. Die dabei entwickelten Prototypen müssen nicht vollständig funktional sein oder besonders schnell oder stabil, man kann zu diesem Zeitpunkt auch Paper-Mockups nutzen, um dieses Feedback von den Benutzern einzuholen. Nach der Generierung eines anfänglichen Prototypendesigns und dem Sammeln des daraus entstandenen Feedbacks während der Evaluation kann das Designteam zur Datenerfassung, den Analysen oder der

Designphase zurückkehren um die Ursachen von jedem Problem, das identifiziert wurde zu bestimmen und versuchen es zu lösen. Dieser Prototypingprozess kann beliebig oft wiederholt werden, solange bis die gewünschten Ergebnisse erzielt werden. [Wilkins 2003]

3.2 Allgemeiner Designprozess

Laut Jennifer Preece liegen der allgemeinen Designtätigkeit und zwar unabhängig von ihrer Disziplin, egal ob Architektur, Grafik oder Produkt Design oder Interface oder Software Design, vier grundsätzliche Aktivitäten zu Grunde: das Identifizieren von Bedürfnissen und das Festsetzen von Anforderungen, das Entwickeln alternativer Designs, welche diese Anforderungen erfüllen, das Erzeugen interaktiver Versionen, so dass diese kommuniziert und bewertet werden können und deren Evaluation, z.B. anhand dem Grad ihrer Akzeptanz. Auch wenn die Bezeichnungen oftmals im Einzelnen divergieren, so sind die Aktivitäten im Wesentlichen dieselben. Diese allgemeingültigen Phasen der Designtätigkeit entsprechen den vier grundsätzlichen Phasen, die auch Barry Wilkins für die Generische Designmethodologie, welche dem Software Design im Allgemeinen zugrunde liegt, ermittelt hat (vgl. Kapitel 3.1.3). In diesem generischen Designprozess gehört, laut Jennifer Preece, das Erzeugen alternativer Designs zur eigentlichen Kernaktivität im Design und ist damit eine äußerst wichtige Phase im Designprozess. [Preece et al. 2002]

Erzeugung alternativer Designs

Eine allgemeine menschliche Tendenz ist, sich an die Dinge zu halten, von denen man weiß, dass sie funktionieren. Auch wenn man erkennt, dass wahrscheinlich mögliche bessere Lösungen existieren, als das was man selber kennt und als gut erachtet, ist es wesentlich einfacher und bequemer diese eine Lösung zu akzeptieren, da man sicher sein kann, dass sie in jedem Fall funktioniert. Dies ist nicht notwendigerweise „schlecht", jedoch werden so, gute und eventuell bessere Alternativen niemals betrachtet und das Betrachten alternativer Lösungen, ist ein äußerst wichtiger Schritt im Designprozess. Wie erzeugt man jedoch alternative Designideen? [Preece et al. 2002]

Sicherlich ist es von der Person des einzelnen Designers abhängig, wie viele und wie gut die jeweiligen Ideen sind, die innerhalb eines Designprozesses generiert werden. Erfahrene Designer blicken auf mehrjährige Designerfahrung zurück und können aus einem Pool von Ideen und Erfahrungen schöpfen, wogegen unerfahrene Designer sich diese oftmals aufwändig erarbeiten müssen.
Die Problematik liegt darin, dass selbst bei erfahrenen Designern die Designerfahrung sehr einseitiger Natur sein kann, wie im Fall des Software Designs beispielsweise begrenzt auf einen bestimmten Typ von Software oder Software eines bestimmten Unternehmens. Diesem Modus des Denkens entkommt man niemals, selbst bewusst nicht. Bewusst oder unbewusst, wenden Designer das an, was sie kennen und verwenden Lösungen, welche

sie als gut erachten und die sie zuvor gesehen haben, wieder. Unerfahrene Designer müssen sich ihre Ideen erarbeiten, wie z.B. durch das Betrachten erfolgreicher Designlösungen anderer Designer. Die Problematik hierbei kann jedoch in der Wahl von minderwertigen Vorbildern liegen. [Tidwell 1999]

Dieser eher intuitiven, nicht gesteuerten Vorgehensweise für die Ideenfindung, wie sie beim Software Design - welches im Vergleich zu anderen Designdisziplinen noch eine relativ junge Disziplin ist - noch eher verbreitet ist, stehen Methoden und Techniken aus Designdisziplinen gegenüber, die sich schon länger mit dem Designprozess und der Ideenfindung beschäftigen, wie beispielsweise dem Produkt Design oder der Architektur. In diesen Designdisziplinen nimmt die Phase der Ideenfindung - wenn sie denn beim Software Design überhaupt bewusst ausgeführt wird - einen wesentlich umfangreicheren und bedeutenderen Part im Designprozess ein. In diesen Designdisziplinen wird ein sehr großer Teil der Zeit und des Aufwands in das anfängliche Design und hierbei insbesondere in die Phase der Ideenfindung investiert. [Preece et al. 2002] Die intensive und bewusste Suche nach möglichst vielen alternativen und guten Lösungsmöglichkeiten und Designvorschlägen wird sorgfältig ausgeführt. Zur Unterstützung der Ideenfindungsphase des Designprozesses neuer Produkte oder auch der Überarbeitung bestehender Produkte werden oftmals Kreativitätstechniken verwendet, die helfen aus den gewohnten Denkmustern auszubrechen, die jedem zueigen sind. Tatsächlich gibt es, obwohl Kreativität und Erfindung oft als schwer greifbar empfunden werden, Methoden und Techniken, die dazu beitragen Kreativität zu steigern oder anzuregen.

Es gibt eine Vielzahl solcher Kreativitätstechniken, wie z.B. die *Morphologische Matrix*, die *Osborn-Checkliste*, die *Visuelle Synektik* [vgl. Pricken 2005] oder aber die *Mind-Matrix* [vgl. Pricken 2003], und entsprechend groß ist die Menge an verfügbarer Literatur dazu. Zum Teil erfolgt die Archivierung solcher Techniken auch in Form von Online-Sammlungen (z.B. Sammlung für Kreativitätstechniken: Andreas Roser[1]), welche eine Vielzahl von Kreativitätstechniken aufführen und diese z.T. auch erläutern.
Das nachfolgende Kapitel stellt zwei dieser Kreativitätstechniken, welche von den interdisziplinären Ansätzen aus Kapitel 4 verwendet werden, kurz vor. Jedoch kann die Benennungen der einzelnen Kreativitätstechniken oftmals variieren, weshalb die hier beschriebenen Techniken auch unter anderem Namen bekannt sein können.

[1] Online-Sammlung für Kreativitätstechniken: Andreas Roser
URL: http://www.andreas-roser.com/index.html

3.3 Kreativitätstechniken

Eine Kreativitätstechnik, welche häufig unbewusst benutzt wird (vgl. Kapitel 3.3), ist das so genannte *Case-based Reasoning*. Diese Technik entstand aus der Beobachtung, dass Designer neue Probleme durch das Heranziehen von Wissen lösen, welches sie von der Lösung früherer ähnlicher Probleme erworben haben. Dabei kann es sich um Erfahrungen handeln, die sich der Designer selbst erworben hat, jedoch auch um Erfahrungen, die von anderen Designern bei früheren Projekten erworben wurden. D.h. dieses Designwissen kann auch anderen Designern gleichermaßen zugute kommen. Dieses zurate ziehen von Erfahrungen kann beispielsweise in Form von Durchstöbern einer Sammlung von Designs erfolgen, durch das der Designer dazu inspiriert wird, sich alternative Möglichkeiten und damit alternative Lösungen zu betrachten. Prinzipiell ist das ganz bewusste Aufspüren passender Quellen der Inspiration ein wertvoller Schritt in jedem Designprozess. Diese Quellen können unterschiedlicher Art sein, z.B. können sie dem geplanten neuen Produkt sehr ähnlich sein, wie beispielsweise Konkurrenzprodukte oder sie können frühere Versionen ähnlicher Systeme sein oder etwas gänzlich anderes. Letztere Technik wird *Cross-Fertilization* genannt, denn tatsächlich ist nur sehr wenig wirklich neu in dieser Welt. Neue Ideen oder Lösungsvorschläge, oftmals mit dem populären Begriff „Innovation" bezeichnet, entstehen auch durch Transferieren von Ideen, Techniken oder Materialien von einem Anwendungsbereich in einen anderen. Also durch Ändern des Kontextes in dem etwas angewendet wird. Dazu werden Lösungen von Bereichen analysiert, die dem eigenen fachfremd sind und entsprechende Designprinzipien auf den eigenen Bereich angewendet. [Preece et al. 2002]

4 Interdisziplinäre Ansätze für das Visualization Design

„*The real voyage of discovery consists not in seeking new landscapes*
but in having new eyes."
[Marcel Proust]

Da der Bereich des Software Design noch eine sehr junge Disziplin ist, lohnt es sich andere traditionelle Designdisziplinen, die schon länger bestehen wie z.B. die Architektur oder das Produkt bzw. Industrie Design, welches die Gestaltung dreidimensionaler Objekte, die industriell gefertigt werden umfasst und sich von einer anfänglich reinen Ingenieurskunst seit dem 19. Jh. zu einer eigenen Designdisziplin entwickelt hat, zu betrachten und zu analysieren. Es ist interessant zu beobachten wie diese Designdisziplinen zu ihren alternativen Designideen bzw. -vorschlägen, die in jedem Gestaltungsprozess - egal welcher Disziplin – generiert werden, gelangen.
Insbesondere im Designbereich gibt es viele Kreativitätstechniken, die helfen können, die Fähigkeit alternative Ideen zu entwickeln, zu erlernen bzw. zu trainieren. Sicherlich gibt es Menschen, die von Natur aus kreativer veranlagt sind als andere, aber dennoch ist Kreativität auch eine Übungssache und kann daher auch mittels Training gesteigert werden. Darüber hinaus gibt es Hilfsmittel – die so genannten Kreativitätstools - die helfen können den Kreativitätsprozess zu unterstützen und daher in den Designprozess integriert werden.

Nachfolgend werden nun zwei Ansätze aus diesen traditionellen Designbereichen, die der Generierung alternativer Designideen dienen, näher betrachtet und vorgestellt, mit dem Ziel diese auf eine etwaige Adaptierungsmöglichkeit für den Bereich des Visualization Design hin zu überprüfen - und hierbei insbesondere zur Ideenfindung in der Designphase von Visualisierungen. Diese Phase wird, obwohl sie eine äußerst relevante Phase des Designprozesses einer Visualisierung darstellt, bisher in der Regel nur unbewusst ausgeführt und findet demnach innerhalb des Visualization Design noch viel zu wenig Beachtung.
Zu den betrachteten Ansätzen gehören der Ansatz der *Pattern*, der von Christopher Alexander vorgeschlagen und ursprünglich für den Entwurf von Architektur entwickelt wurde, jedoch in vielen anderen Disziplinen bereits Anwendung gefunden hat, sowie der Ansatz der *Box-of-Ideas*, eine Kreativitätstechnik, die primär im Produkt Design Bereich verwendet wird und von der amerikanischen Designagentur IDEO in Form der *Tech Box* nahezu perfektioniert wurde. Beide Ansätze dienen der Ideenfindung innerhalb des Designprozesses.

4.1 Vorstellung des Pattern Ansatzes

Der folgende Abschnitt erläutert die grundlegende Idee der Patterns und ihre Entwicklung von den Architektur Patterns von Christopher Alexander über die Software Patterns, auch Software Design Patterns genannt, aus dem Bereich des Software Engineering bis zu den Ansätzen für die Human-Computer Interaction und den Visualization Patterns aus dem Bereich der Visualisierung von Daten. Zu den im Rahmen der vorliegenden Arbeit betrachteten Patternsammlungen gehören die Architektur Patterns von Christopher Alexander, die Patternsammlung für Software Engineering von der sog. *Gang of Four*, zu denen Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides gehören, diverse Patternsammlungen aus dem Bereich der HCI, wie *Common Ground* von Jenifer Tidwell[2], die Usability Patternsammlung von Kimberly Perzel und David Kane[3] und die Patternsammlungen für Web Design, GUI Design und MobileUI Design von Martijn van Welie[4], sowie die Patternsammlung von Jan Borchers[5] und die Visualization Patterns von Barry Wilkins[6]. Die Vorstellung der Patternsammlung[7] von Christopher Alexander führt das Konzept der Patterns detailliert ein, da dieser Sammlung viele zentrale Konzepte und Eigenschaften der Patterns zugrunde liegen, die sich auf andere Disziplinen, wie das Software Engineering, die Human-Computer Interaction oder das Visualization Design übertragen haben. Anschließend wird kurz auf die Entwicklung der Patterns in einigen dieser Disziplinen eingegangen, um einen Überblick über die verschiedenen Ausprägungen des Pattern-Ansatzes im Allgemeinen zu geben. Des weiteren wird repräsentativ aus jeder der Disziplinen jeweils eine Patternsammlung, die zu den bekanntesten gehört, etwas näher vorgestellt. Jedoch können an dieser Stelle nicht alle der im Rahmen dieser Arbeit betrachteten Patternsammlungen präsentiert werden, da dies den Umfang dieser Arbeit überschreiten würde. Die Resultate ihrer Betrachtung flossen aber in die Entwicklung der VizBox mit ein.

---

[2] Jenifer Tidwell: Common Ground: A Pattern Language for Human-Computer Interface Design
URL: http://www.mit.edu/~jtidwell/interaction_patterns.html

[3] Kimberly Perzel und David Kane
URL: http://jerry.cs.uiuc.edu/~plop/plop99/proceedings/Kane/perzel_kane.pdf

[4] Martijn van Welie: Amsterdam-Collection
URL: http://www.welie.com/index.html

[5] Jan Borchers
URL: http://www.hcipatterns.org/patterns/borchers/patternIndex.html

[6] Barry Wilkins
URL: http://www.cs.bham.ac.uk/~bxw/vispatts/ (30.11.2004)

[7] *Anmerkung: innerhalb dieser Arbeit wird der Begriff „Patternsammlung" als Oberbegriff für Patternsprachen als auch für Patternkataloge, deren Pattern keine explizite Patternsprache bilden, verstanden*

4.1.1 Architektur Pattern

Die ersten Anfänge Designwissen zu dokumentieren, werden auf die Zeit der Renaissance zurückgeführt. So wird angenommen, dass einer der Hauptgründe weshalb die Architektur, wie auch andere Wissenschaften und Künste der damaligen Epoche, eine Blütezeit erlebte, darin begründet liegt, dass die Baumeister der damaligen Zeit damit begonnen hatten ihr Designwissen systematisch zu sammeln, zu dokumentieren und zu strukturieren. [Borchers 2001]

Basierend auf dieser allgemeingültigen Idee entwickelte in den 1970ern der Architekt und Mathematiker Christopher Alexander eine neue revolutionäre Theorie der Architektur, des Bauens und der Planung. [Borchers 2001] Neben seinen umstrittenen Vorschlägen zum Prozess des Bauens selbst löste v.a. seine Theorie zur Auflösung der Architektur in elementare Patterns und ihre Beziehungen heftige Diskussionen aus, die selbst bis heute noch andauern und deren revolutionäre grundsätzliche Idee bereits in viele andere Bereiche vorgedrungen ist. [Donath 2004]

Bereits 1964 stellte Christopher Alexander in seiner Doktorarbeit *Notes on the Synthesis of Form*, in der er sich grundsätzlich mit der komplexen Entscheidungsfindung beim Entwurf beschäftigt, fest, dass es selbst unter Einsatz von Computern unmöglich ist, alle Anforderungen eines Entwurfs gleichzeitig zu lösen. Seiner Ansicht nach, lässt sich dieses Problem jedoch durch Teillösungen, die für Gruppen mit starken Wechselwirkungen gefunden wurden, überwinden. [Alexander et al. 1995] Er beschreibt daher die Theorie der Zerlegung eines Gesamten und komplexen Problems (= Designaufgabe) in Einzelprobleme (= Teilgruppen) und begründet, dass mit der Lösung der Einzelprobleme auch das Gesamtproblem gelöst ist. [vgl. Donath 2004]

Basierend auf diesem Ansatz veröffentlichte Christopher Alexander 1977 dann im dem Buch *A Pattern Language* 253 sogenannte Designpatterns. Von diesen beschreibt jedes einzelne eine erfolgreiche mögliche Lösung zu einem allgemeinen wiederkehrenden partiellen Designproblem innerhalb des Architekturentwurfs und zwar mittels der Charakteristik der unveränderlichen Qualitäten dieser Lösung. Wie der Name Pattern, also Muster, schon besagt, hat ein Pattern mit sich wiederholenden Elementen zu tun. [van Welie et al. 2001] Alexander selbst definiert ein Pattern wie folgt: „*Each pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution*". [Alexander 1979, 247] Somit besteht jedes einzelne dieser Pattern im Wesentlichen aus einer Problemstellung, also einer Beschreibung eines immer wieder auftretenden Problems in unserer Umwelt, einer Erörterung des Problems unter Zuhilfenahme einer Skizze und aus der Beschreibung einer Lösung und zwar so, dass man diese beliebig oft anwenden kann, ohne sich je zu wiederholen.

Patternformat

Aus Gründen der Handhabbarkeit und Klarheit verfügt jedes der Pattern über das gleiche Format. D.h. jedes der Architektur Pattern besteht aus denselben Abschnitten (nachfolgend Elemente genannt), welche stets in der gleichen Reihenfolge und Form präsentiert werden. Jedoch markiert Alexander diese Elemente nicht durch explizite Überschriften, sondern die Information ihrer Strukturierung ist implizit - mit Hilfe von festgelegten Regeln bezüglich der Typographie - in den Patterns enthalten. So besteht nicht nur jedes der Pattern aus denselben Elementen in derselben Ordnung, sondern jedes Patternelement hat seine eigenen festgelegten Typographieregeln, wie z.B. der Name des Pattern in Großbuchstaben, das Problem und die Lösung in fetter Schrift als separierter Absatz, usw. [Alexander et al. 1995][Borchers 2001]
Der nachfolgende Abschnitt gibt einen kurzen Überblick über das Format über das jedes der Architektur Pattern verfügt [vgl. Borchers 2001] und geht anschließend nochmals detaillierter auf jedes der einzelnen Elemente ein.

Format der Architektur Pattern

- der **Name** des Pattern
- ein **Ranking** seiner Gültigkeit
- ein **Bild** als ein Beispiel seiner Anwendung
- der **Kontext**, in welchem es verwendet wird
- eine kurze **Problemstellungnahme**
- eine detaillierte **Problembeschreibung** mit empirischem Background
- eine zentrale **Lösung** des Pattern
- eine **Skizze**, welche die Lösung illustriert
- und schließlich **Referenzen** zu verwandten Pattern

Wie bereits oben erwähnt, setzt sich jedes der Pattern aus drei größeren Teilen zusammen. Dabei bilden Name, Ranking, Bild und Kontext den einführenden Teil, Problemstellungnahme, Problembeschreibung, Lösung und Skizze stellen den zentralen Teil dar und die Referenzen formen den abschließenden Teil jedes Pattern. Diese Hauptteile werden optisch jeweils durch eine Linie von drei Sternchen von einander getrennt dargestellt. [Borchers 2001]

Name

Der Name eines Pattern, wie z.B. SITTING WALL, gehört zu den essentiellen Elementen jedes Pattern, da er die Idee des Pattern in einem oder ein paar wenigen Worten vermitteln muss. So bleibt es besser in Erinnerung und man kann während des Design-prozesses z.B. innerhalb von Diskussionen leichter darauf Bezug nehmen. [Borchers 2001]

Ranking

Ein Ranking für jedes Pattern gibt an, wie der Autor des Pattern selbst den Grad der Zuverlässigkeit der Lösung einschätzt, denn die Qualität der Pattern ist nicht immer gleich. Manche Lösungen sind profunder als andere. Dargestellt wird das Ranking in Form von Sternchen, in der Anzahl von null bis zwei, hinter dem Namen des Pattern.

Von den Patterns mit zwei Sternchen wird angenommen, dass es echte Invarianten darstellen. D.h. dass die gegebene Lösung Merkmale zusammenfasst, die allen möglichen Arten, das Problem zu lösen, gemeinsam ist. Eine richtige Lösung des gegebenen Problems ist also nur möglich, wenn die Umwelt entsprechend dem vorgegebenen Pattern gestaltet wird. Das Pattern beschreibt in diesem Fällen prägnante und unverzichtbare Merkmale einer wohlgestalteten Umwelt.

Patterns mit einem Sternchen kommen dem Ziel eine solche Invariante identifiziert zu haben nahe, es ist aber möglich, die Lösung noch zu verbessern. Für die Verwendung des Pattern wird daher geraten, die Lösung mit einer gewissen Skepsis zu behandeln und Varianten zu dieser Lösung zu suchen, da es mit Sicherheit weitere Lösungen gibt, die in den Beschreibungen des Patterns nicht enthalten sind.

Kein Sternchen bedeutet, dass es sicherlich nicht gelungen ist, eine Definition einer echten Invariante zu formulieren und es daher mit Gewissheit noch andere Lösungen für das Problem gibt als nur die angegebene. Die angegebene Lösung zeigt aber zumindest einen möglichen Weg auf, das Problem zu lösen. Die wahre Invariante, also die echten Merkmale als Kern aller möglichen Lösungen des Problems, muss aber noch gefunden werden. [Alexander et al. 1995]

Bild

Nach dem Kopfteil des Pattern folgt ein Bild, das ein archetypisches Beispiel des betreffenden Pattern zeigt. Für gewöhnlich ist es eine Fotografie, die ein gutes Beispiel der Idee des Pattern zeigt. [Alexander et al. 1995]

Kontext

Ein einführender Absatz stellt den Zusammenhang des Pattern zu größer skalierten Patterns her. D.h. der Beitrag dieses Pattern wird zur Vervollständigung bestimmter größerer Pattern umrissen. Dadurch wird das Pattern zu anderen Pattern auf einem höheren Level verlinkt. [Borchers 2001][Alexander et al. 1995]

***

Drei Sternchen markieren den nun folgenden Problemabschnitt, welcher den zentralen Bestandteil jedes Pattern darstellt.

Problemstellungnahme

Eine fett hervorgehobene Schlagzeile fasst in einem oder zwei Sätzen das Wesen des Problems, das dieses Pattern adressiert zusammen. [Alexander et al. 1995]

Problembeschreibung

In einer ausführlichen Problembeschreibung folgt der längste Teil des Pattern, der eigentliche Inhalt. Hier wird beispielsweise der empirische Hintergrund des Pattern beschrieben, seine Gültigkeit begründet, die Spannweite verschiedener Formen aufgezeigt, die das Pattern in einem Gebäude annehmen kann, etc. [Alexander et al. 1995] Die Problembeschreibung nennt oftmals das Problem unter Hervorhebung der in Konflikt stehenden Randbedingungen, den sog. *Forces*. Ziel der Patterns ist es, diese zu klären und sie optimal im gegebenen Kontext auszubalancieren.

„As an element in the world, each pattern is a relationship between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain spatial configuration which allows these forces to resolve themselves. As an element of language, a pattern is an instruction, which shows how this spatial configuration can be used, over and over again, to resolve the given system of forces, whereever the context makes it relevant." [Alexander 1979, 247]

Lösung

Die Lösung, also die Essenz des Pattern, ist das zentrale Element jedes Pattern und erfolgt wieder im Fettdruck. Sie ist gewissermaßen das Destillat aus all den zuvor beschriebenen Beispielen, in Form einer allgemeingültigen Lösung zu dem vorliegenden Problem. [Borchers 2001] Sie löst im Allgemeinen ein Problem von Konflikten zwischen den verschiedenen Randbedingungen. Beispielsweise werden Menschen innerhalb eines Raums natürlicherweise in Richtung Licht gezogen, also z.B. zum Fenster. Halten sie sich länger in dem Raum auf möchten sie ebenfalls nach einer Weile hinsetzen. Hat ein Raum nur in der Mitte Sitzgelegenheiten, so können diese konkurrierenden Randbedingungen nicht gelöst werden. Die Lösung liegt darin eine Art von Sitzgelegenheit im Bereich des Fensters zu bauen (vgl. Pattern WINDOW PLACE), die es Menschen ermöglicht sehr nah oder innerhalb des Fensters zu sitzen [Alexander et al. 1977, 833]. Diese Randbedingungen, die mit der Lösung des gestellten Problems im Zusammenhang stehen können unterschiedlicher Natur sein, wie z.B. sozialer, wirtschaftlicher, natürlicher oder physischer. [Borchers 2001]

Die Lösung ist dabei so allgemein formuliert, so dass jeder auf seine eigene persönliche Art und Weise das Problem selbst lösen kann. Dazu muss er lediglich die Lösung den eigenen Präferenzen und den örtlichen Bedingungen anpassen. Deshalb hat die Lösung stets die Form einer positiv formulierten Anweisung, ohne jedoch aufzwingend zu sein, sodass man genau weiß, wie man das Pattern umzusetzen hat. Die Lösung enthält dabei lediglich die wesentlichen Punkte, auf die bei einer wirklichen Lösung des Problems nicht verzichtet werden kann. D.h. jede Lösung versucht die unveränderlichen Merkmale zu erfassen, die überall dort vorkommen, wo das Problem gelöst worden ist. [Alexander et al. 1995] Damit ist die Lösung eine klar formulierte, jedoch generische Gruppe von Instruktionen, die in verschiedenen Situationen angewendet werden kann und Designern hilft neue Instanzen zu konstruieren. [Borchers 2001][van Welie et al. 2001]

Skizze
Eine Skizze visualisiert die zentrale Idee der Lösung grafisch, so dass sie einfacher zu erfassen ist und man sich leichter an sie erinnern kann. Die wesentlichen Elemente der Skizze sind beschriftet. Da jedes Pattern eine räumliche Anordnung beschreiben soll um ein bestimmtes Problem zu lösen, muss das Skizzieren eines Pattern immer möglich sein: *„If you can't draw a diagram of it, it isn't a pattern.*" [Alexander 1979, 267] Ein Bild oder Foto kann immer nur eine spezielle Anwendung der Lösung präsentieren, die Skizze aber muss das für alle geltende Prinzip der Lösung verdeutlichen. [Borchers 2001]

***
Das Ende des Hauptteils des Pattern wird von drei weiteren Sternchen begrenzt.

Referenzen
Das letzte Element des Pattern stellt die Referenzen dar, welche den Leser zu kleineren Pattern verweisen. Dieser Absatz setzt das Pattern mit allen kleineren Pattern der Sprache in Beziehung, mit deren Hilfe es ergänzt, verschönert und ausgefüllt wird. [Alexander et al. 1995]

Dieser Aufbau der Pattern verfolgt im Wesentlichen zwei Absichten. Zum einen geht es Christopher Alexander darum, jedes Pattern in Verbindung mit anderen Pattern zu zeigen, sodass man die Sammlung aller 253 Pattern als ein Ganzes begreift, als eine Sprache, in der eine unendliche Vielfalt von Kombinationen geschaffen werden kann. Und zum anderen soll das Problem und die Lösung jedes Pattern so dargestellt werden, dass man es selbst beurteilen und modifizieren kann, ohne die zentrale Idee zu verlieren. [Alexander et al. 1995]
Dabei kann das Entwurfsproblem oder -ziel sowohl technischer als auch ästhetischer oder sozialer Natur sein. Diese Patterns sind, laut Christopher Alexander, der Schlüssel zur Formfindung. [Donath 2004][Alexander et al. 1995]

Patternsprache

Patterns beinhalten die Lösungen zu Problemen von größer dimensionierten Maßstäben, wie dem Entwurf von Städten oder Nachbarschaften, über kleiner skalierte, wie beispielsweise dem Entwurf einer Promenade bis hin zu Lösungen hinsichtlich des Designs von einzelnen Gebäuden, Räumen und Ausstattungsdetails. Dadurch ergibt sich eine Schachtelung der Maßstäbe dieser Geometrie, so dass jedes Ganze kleinere Ganzheiten in sich birgt und sich mit anderen zu einem neuen, größeren Ganzen zusammenschließt. Dabei greifen die Patterns stufenweise ineinander, wodurch den Patterns eine implizite hierarchische Ordnung innewohnt. [Alexander et al. 1995] Durch das Anwenden einer Sequenz solcher Patterns werden schließlich ganze Gebäude, Nachbarschaften und Städte schrittweise erzeugt. Diese werden so iterativ vergrößert, fixiert und verbessert. Alle 253

Pattern zusammen schaffen ein in sich geschlossenes Bild einer ganzen Region. [Borchers 2001][Alexander et al. 1995]

Alexander ist der Auffassung, dass man schöne und beständige Architektur als Ganzes und den Zusammenhang der Bestandteile als Sprache auffassen kann. Entsprechend bilden für ihn die Patterns zusammen eine sog. Patternsprache, in der sich die einzelnen Patterns wie "*wie Worte in einem Satz...*" verhalten. Für ihre Benutzung gibt es eine Ordnung und Struktur, wie z.B. die Benutzung der Pattern entsprechend ihrer hierarchischen Ordnung vom größeren zum kleineren Maßstab hin. [Donath 2004] Diese hierarchische Ordnung ist für die Funktionsweise der Sprache wesentlich. Sie stellt sich als lineare Abfolge dar, die auf der Beziehung zwischen den Pattern beruht. Jedes Pattern beinhaltet Verweise, die sich auf bestimmte „größere" Pattern, die in der Sprache an höherer Stelle stehen, und auf bestimmte „kleinere" Pattern, die in der Sprache an untergeordneter Stelle stehen, beziehen. Das Pattern selbst trägt somit zur Vervollständigung der „über" ihm stehenden Pattern bei und wird selbst vervollständigt durch die kleineren Pattern, die „unter" ihm stehen. D.h. kein Pattern ist eine abgetrennte Einheit. Jedes Pattern wird gestützt durch andere Patterns: von den größeren Pattern, in die es eingebettet ist, von den Pattern gleichen Maßstabs, die es umgeben, und von den kleineren Pattern, die in ihm eingebettet sind. [Alexander et al. 1995]

Da auch jede kurze Sequenz von Patterns aus dieser Sprache, selbst wieder eine Sprache für einen kleineren Teil der Umwelt bildet, gibt es gemäß der unzähligen Kombinations-möglichkeiten dieser 253 Patterns entsprechend viele Patternsprachen. Die Charakteristik des jeweilig erzeugten Objekts, sei es nun ein Raum oder eine ganze Stadt, entsteht durch die Zusammenstellung der entsprechenden Patterns, also die jeweilige Patternsprache, die darin eingebaut werden sollen. Der Charakter dessen, was gebaut wird, wird durch die Sprache der Pattern entstehen, die man zu seiner Erzeugung verwendet. Daher ist die Auswahl einer Sprache für ein Projekt von grundlegender Bedeutung. [Alexander et al. 1995]

Durch die kombinierte Anwendung der einzelnen Patterns ist es möglich, Regionen in unendlich verschiedenen Formen mit der unzählbaren Vielfalt aller Einzelheiten zu erzeugen - also der Entwurf komplexer Architektur. [Alexander et al. 1995][Borchers 2001]

Dichte der Pattern

Durch die kombinierte Verwendung der Patterns entsteht eine sog. Patternsprache deren synergetischer Effekt, laut Alexander, das Ganze mehr werden lässt als die bloße Summe seiner Teile. [Tidwell 1999] Dabei kann man Gebäude durch das lose Aneinanderreihen von Pattern errichten. Ein so konstruiertes Gebäude stellt eine Ansammlung von Pattern dar und besitzt keinen inneren Zusammenhalt. Fügt man aber Patterns so zusammen, so

dass sich möglichst viele Pattern innerhalb desselben Raums überlagern, besitzt das Gebäude einen inneren Zusammenhalt und besitzt viele Bedeutungen, die auf kleinem Raum zusammengefasst sind. Durch diesen Zusammenhalt gewinnt es an Substanz. [Gamma et al. 2004] Jedes Gebäude, jeder Raum, jeder Garten besitzt eine höhere Qualität, wenn alle erforderlichen Muster soweit wie möglich verdichtet sind. Beim Anwenden der Sprache sollte man also darauf achten möglichst viele Pattern auf kleinstem Raum zu verdichten, denn die besten Entwürfe verwenden viele Patterns, die sich ergänzen und miteinander verweben, um ein größeres Etwas zu schaffen. [Alexander et al. 1995][Gamma et al. 2004]

Netzstruktur

Wie Christopher Alexander in seinem 1979 erschienen Buch *The Timeless Way of Building* ausführlich beschreibt hat die Patternsprache selbst eine Netzstruktur. Wenn man die Patternsprache benutzt, so resultiert dies immer in einer bestimmten Abfolge der Patterns: von den größeren und den kleineren Patterns, von jenen die Strukturen schaffen, zu jenen, die Strukturen verfeinern, und dann zu jenen, welche die Verfeinerungen verfeinern, usw. Da die Sprache aber ein Netz ist, gibt es keine Reihenfolge, in der sie vollkommen erfasst werden kann. Die Reihenfolge, in der Alexander die Patterns in seinem Buch *A Pattern Language* präsentiert erfasst jedoch „den großen Schwung des ganzen Netzes". Die präsentierte Reihenfolge der Patterns ist sowohl eine Zusammenfassung der Sprache als auch zugleich ein Inhaltsverzeichnis der Patterns. [Alexander et al. 1995]

Anwenden der Sprache

Um einen Überblick über die gesamte Sprache zu erhalten, ist es hilfreich die Zwischensätze zu lesen, welche die einzelnen Gruppen von Patterns miteinander verbinden. Hat man diesen Überblick gewonnen, so kann man nach den Patterns suchen, die für ein bestimmtes Projekt relevant sind, wobei man über die Verweise innerhalb der Patternbeschreibung zu den nächsten relevanten Patterns geführt wird. Die Reihenfolge der Patterns stellt auch den Grundplan dar, von dem aus man eine Sprache für das eigene Projekt bilden kann. Dazu wählt man die zutreffendsten Pattern aus, lässt sie aber mehr oder weniger in der Reihenfolge, in der sie Alexander in seinem Buch *A Pattern Language* präsentiert. Alexander verwendet dieses organisierende Prinzip, da es die Ordnung reflektiert, die er auch für das Vorgehen beim Design vorschlägt. Dieses beginnt bei einem groben Entwurf, welcher dann durch die Verwendung kleinerer Patterns zunehmend verfeinert wird. [Borchers 2001][Alexander et al. 1995]

Philosophie und Qualität

Hinter der Theorie von Christopher Alexander steht seine Überzeugung, dass es eine nahezu selbstverständliche Fähigkeit der Menschen gibt, zeitlose und damit qualitativ hochwertige Dinge zu erschaffen, die heutzutage jedoch weitestgehend – und vor allem bei professionellen Gestaltern – verloren gegangen ist, welche jedoch mit Hilfe der Patterns wiedergewonnen werden kann. [Alexander et al. 1995] Alexander gewinnt seine Pattern aus den Beobachtungen bewährter, beständiger und "guter" Architektur, wie z.B. alte englische Universitäten, italienische Dörfer oder orientalische Tempel. Seine Patterns berücksichtigen daher nicht nur die Architektur verschiedener Gesellschaften bzw. Gemeinschaften sondern auch verschiedener Epochen. In dieser zeitlosen Architektur suchte er nach einer Sprache, die aus archetypischen Formen und Beziehungen besteht. Diese Archetypen, die sog. Pattern, befriedigen, laut Alexander, elementare menschliche Bedürfnisse und streben somit den Zustand der emotionalen Erfüllung an. Diese Qualität beschreibt Alexander in seinem 1979 erschienen Buch *The Timeless Way of Building* mit dem Ausdruck *„Quality without a name*". Er argumentiert, dass Gebäude und Städte in denen Menschen gerne leben eine bestimmte, zeitlose Qualität innewohnt, die sich weder konkret benennen, noch sich auf ein einzelnes Maß reduzieren lässt. Damit geht sein theoretischer Ansatz teilweise auf die Philosophie und Naturbetrachtungen östlicher und asiatischer Kulturen zurück. [Donath 2004][Borchers 2001]

Alexanders These besagt, dass man durch das Erlernen und Verwenden dieser sorgfältig definierten Gruppe von Designregeln - den Pattern - in denen die architektonische Erfahrung (Designwissen) bisheriger Generationen von Architekten kodiert ist, hervorragende Leistungen in der Architektur erzielen könnte, in welcher die verlorenen gegangenen Qualitäten früherer Baukunst steckt. [Tidwell 1999][Alexander et al. 1995]

Und obwohl die Qualität eines gut gestalteten Gebäudes differenziert und schwer in Worte zu fassen ist, sind die Patterns selbst, die dieses Gebäude ausmachen, bemerkenswert einfach und allgemeinverständlich geschrieben und somit auch von Laien leicht zu verstehen. [Tidwell 1999] Bewohner gestalten heutzutage nicht mehr selbst ihre Umgebung sondern haben dies an Architekten und Bauunternehmer abgegeben. Da diese jedoch in der Regel nicht persönlich oder anderweitig eng zu den zukünftigen Bewohnern und den Objekten in Beziehung stehen, unterstützen die resultierenden Entwürfe oftmals nicht auf optimale Weise die Lebensgewohnheiten, Aufgaben und Ereignisse der Bewohner. Alexanders Ziel ist es, mit Hilfe der Patterns die Bewohner wieder an der Gestaltung ihrer Umgebung teil haben zu lassen. Ihnen das nötige Vokabular, in Form von natürlichsprachlich formulierten Pattern, an die Hand zu geben um ihre Ideen und Entwürfe auszudrücken und um sie mit anderen diskutieren zu können. [Borchers 2001] Die Patternsprache ist aber auch in höchstem Maße praxisbezogen. Alexander selbst versteht die Patternsprache als Planungshilfe für Bau und Planung. D.h. sie dient sowohl als Regelwerk, das wie eine Checkliste beim Entwerfen verwendbar ist, da es Antworten auf

Entwurfsprobleme liefert, wie „Wie hoch soll eine Fensterbrüstung sein? Wie viele Geschosse soll ein Gebäude haben?", als auch als Anleitung im tatsächlichen Bauvorgang. Die Patternsprache erlaubt sowohl das Einbringen individueller Wünsche und Erfahrungen als auch kollektiver Errungenschaften und Traditionen und dient als Quelltext zum zeitlosen Bauen. [Alexander et al. 1995]

Tatsächlich betrachtet Alexander seine Pattern lediglich als Hilfsmittel um Menschen an die wichtigen Gesichtspunkte bei der Gestaltung ihrer Umgebung zu erinnern; einmal verstanden, haben die Patterns ihren Zweck erfüllt und werden nicht länger als ein formales Framework benötigt [Borchers 2001]: *„The language, and the process which stem from it, merely release the fundamental order which is native to us. They do not teach us, they only remind us of what we know already […].*" [Alexander et al. 1977, 531]

Ziel von Christopher Alexander ist es, Menschen eine gemeinsame Patternsprache zu geben, welche selbst lebendig ist, d.h. neue Pattern gefunden und bestehende stets verbessert werden. So dass die wahren Invarianten mit der Zeit entdeckt werden. Jedoch stellen alle von Christopher Alexander präsentierten Patterns, wie er selbst betont, Hypothesen dar. D.h. sie sind alle provisorisch und können sich aufgrund neuer Erfahrungen und Beobachtungen entwickeln. [Alexander et al. 1995] Alexander regt an, was von Kritikern häufig übersehen wird, die im Buch beschriebenen Patterns als Vorschlag zu sehen und sich seine eigene Sprache je nach Aufgabe oder Projekt aufzubauen. [Donath 2004] Da die Patterns autonome Teile eines ganzen Großen sind, können sie jedes für sich studiert und verbessert werden, so dass ihre Entwicklung schrittweise und kumulativ vor sich geht. Alexander betont, dass man jedes Pattern wie eine Hypothese in der Wissenschaft betrachten kann. In diesem Sinne stellt jedes Pattern die derzeit beste Annahme darüber dar, durch welche Anordnung der physischen Umwelt das gegebene Problem am besten gelöst wird. Laut Alexander sind viele der gefundenen Patterns archetypisch. D.h. sie sind so profund, dass sie vermutlich noch in fünfhundert Jahren ebenso Teil der menschlichen Natur und des menschlichen Handelns sein werden, wie sie es heute sind.

Letztendlich ist Christopher Alexander vermutlich der einzige, der versuchte einfache, generische Formen und Beziehungen zu finden, die gewöhnlich in den meisten Gemeinschaften vorkommen, aus einem sozialen Gefüge heraus gewachsen und durch die meisten Menschen einfach zu verstehen sind. Christopher Alexanders Theorie wird daher auch als humane Architektur, welche ihre Basis in den elementaren Bedürfnissen (*human needs*) der Menschen hat, bezeichnet." [Donath 2004] Damit spürte Christopher Alexander auf rationale Weise jenen Qualitäten der gebauten Umwelt nach, die vielfach als irrational angesehen werden. [Alexander et al. 1995]

Kreativität

Christopher Alexanders Pattern sind eine gute Methode um Designwissen, welches erfolgreichen Lösungen für wiederkehrende Probleme im (Architektur-)Entwurf zugrunde liegt, zu kommunizieren, zu archivieren und zu strukturieren. Jedoch, wie er selbst angibt, ersetzen sie nicht die Notwendigkeit von Kreativität. [Gamma et al. 2004] Denn Patterns dokumentieren bereits existentes, bzw. auch verloren gegangenes Designwissen, bringen jedoch selbst keinen neuen theoretischen Boden hervor oder präsentieren neue innovative Techniken. Sie sind lediglich die Ergebnisse vorangegangener Argumente und Gedankengänge. [Alexander et al. 1995]

In den zurückliegenden Jahren haben Christopher Alexander und zahlreiche andere Architekten in praktischen Vorhaben unter Beweis gestellt, dass Alexanders Theorien prinzipiell realisierbar sind. Jedoch blieb bedingt durch die ungewöhnlichen und oft auch radikal anmutenden Theorien von Alexander die Kritik nicht aus. [Donath 2004] Insbesondere bei seinen Kollegen in der Architektur stieß er mit seinen Ideen oftmals auf wenig Sympathie. Dabei bezieht sich diese Kritik allerdings in erster Linie auf den von Alexander vorgeschlagenen Prozess des architektonischen Entwerfens und nicht auf seine Pattern Theorie. Die Kritik seiner Kollegen liegt aber sicherlich auch darin begründet, dass er mit seinen Konzepten den zukünftigen Bewohnern mehr Möglichkeiten verlieh, den Bauprozess zu beeinflussen und ihnen damit ein Teil der Macht, der bisher ausschließlich den Architekten vorbehalten blieb, übertrug. [Borchers 2001]

Auch wenn die Idee der Patterns von Christopher Alexander in der Architektur Community mit unterschiedlicher Resonanz aufgenommen wurde, so setzte sich sein Kerngedanke jedoch in verschiedenen anderen Disziplinen fort. Viele weitere Disziplinen haben seine Idee - positives und neuerdings auch negatives Designwissen zu archivieren - adaptiert. Manche der Grundgedanken Alexanders sind in den vielen Versuchen die Idee der Patterns zu adaptieren verloren gegangen. Auch das Format der Patterns wurde immer wieder verändert und neu angepasst, so dass der Begriff Pattern in jeder Disziplin eventuell eine eigene Definition bräuchte. Der Kerngedanke Designwissen in einem strukturiertem Format zu dokumentieren, zu archivieren und zu kommunizieren ist jedoch erhalten geblieben.

4.1.2 Software Pattern

Zum ersten mal wurde 1987 an der OOPSLA Konferenz über Objektorientierung die Idee der Architektur Patterns vom Software Engineering aufgenommen, um Erfahrungen im Softwareentwurf auszudrücken. Kent Beck war einer der ersten in der Software Community, der Christopher Alexanders Arbeit beachtete und im Software Engineering

bekannt machte. Initiiert durch die Konferenz begann daraufhin ein reger Austausch über die Verwendung von Software Patterns. [Borchers 2001]

Denn obwohl Christopher Alexander über Patterns in Gebäuden und Städten spricht, trifft seine Definition auch für objektorientierte Software Patterns zu. [Gamma et al. 2004] Zwar werden im Software Engineering Lösungen mit Hilfe von Objekten und Schnittstellen anstatt mit Wänden und Türen beschrieben, jedoch kann man beide Arten von Patterns im Prinzip als Problemlösungen für bestimmte Situationen verstehen. Software Patterns beschreiben einfache und präzise Lösungen für wiederkehrende spezifische Entwurfsprobleme im objektorientierten Software Engineering. Wie die Architektur Patterns bieten die Software Patterns Lösungen, die konkret genug sind, um mit guten Resultaten unmittelbar in der Praxis angewendet zu werden und sind gleichzeitig abstrakt genug, um bei einer Vielzahl von Situationen eingesetzt zu werden, allein limitiert durch den Ideenreichtum und die Fähigkeiten der Menschen, welche die Pattern einsetzen. [Tidwell 1999] Wie die Architektur Patterns erfassen die Software Patterns Problemlösungen, welche erst im Laufe der Zeit gefunden wurden, sich weiterentwickelt haben und auf vielen Entwurfsrevisionen und erneutem Programmieren, die Entwickler auf der Suche nach größerer Wiederverwendung und Flexibilität vorgenommen haben, basieren. Allgemeines Ziel der Software Pattern ist es, Entwürfe flexibler, modularer, einfacher, wiederverwendbar und verständlicher zu machen und es dadurch zu ermöglichen, dass Entwickler einen Entwurf auf Anhieb schneller „richtig" machen. [Gamma et al. 2004]

Gang of Four (GoF) – Pattern für objektorientiertes Softwaredesign

Einer der bekanntesten und wohl beeinflussendsten Sammlungen von Patterns für objektorientiertes Softwaredesign wurde 1995 in dem Buch *Design Patterns: Elements of Reusable Object-Oriented Software* von der sog. *Gang of Four*, bestehend aus Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides publiziert, das mittlerweile als Standardwerk im Bereich der Softwarearchitektur angesehen wird. [Borchers 2001] Das Buches enthält einen Katalog mit 23 Patterns, welche in insgesamt drei Hauptkategorien – klassifiziert nach Aufgabe und Gültigkeitsbereich - eingeteilt sind: Erzeugungspattern, Strukturpattern und Verhaltenspattern.

Dabei beschreibt das Pattern den Kontext und die Randbedingungen, unter denen es angewendet werden kann, sowie welche Vor- und Nachteile seine Anwendung hat. Wie die Architektur Patterns, ist jedes der GoF Pattern nach einem einheitlichen Format beschrieben, welches das Verstehen, Vergleichen und Verwenden erleichtern soll, auch wenn sich die einzelnen Pattern hinsichtlich ihrer Granularität und ihres Abstraktionsgrades unterscheiden. Allerdings weicht diese Struktur relativ stark vom Format der Architektur Patterns ab und sind den Bedürfnissen des Software Engineering angepasst. Das Format der GoF Patterns umfasst die folgenden Elemente: Name und Klassifizierung, Zweck, Auch bekannt als, Motivation, Anwendbarkeit, Struktur, Teilnehmer, Interaktionen, Konse-

quenzen, Implementierung, Beispielcode, Bekannte Verwendungen und Verwandte Patterns.
Eine umfassende Beschreibung der Struktur der GoF Patterns findet sich in [Gamma et al. 2004]. Alle GoF Patterns wurden aus real existierenden Systemen abgeleitet und können in bekannten Frameworks gefunden werden. Die Erläuterung jedes Pattern findet mit Hilfe von Beispielcode statt, der anschaulich darstellt, wie das Pattern in objektorientierten Programmiersprachen wie C++ und Smalltalk implementiert werden kann. Die Patterns der *Gang of Four* wurden seit ihrem erstmaligen Erscheinen 1995 als Buch mehrfach überarbeitet, erweitert und verfeinert. Sie geben selbst an, dass ihre Sammlung von Software Patterns sehr wahrscheinlich weder vollständig noch endgültig ist, sondern eher eine Momentaufnahme ihrer Gedanken über guten Entwurf. [Gamma et al. 2004]

Es gibt einige Gemeinsamkeiten zwischen den Patterns von Christopher Alexander und denen der *Gang of Four*. Beide Ansätze gewinnen ihre Patterns durch die Beobachtung existierender Systeme, verwenden ein einheitliches Format um die Patterns zu beschreiben, auch wenn sich diese Formate deutlich von einander unterscheiden. Beide Beschreibungsarten basieren auf natürlicher Sprache, anstatt eine formale Sprache zu verwenden. Und beide Patternarten bieten dem Benutzer viele Beispiele für das Pattern in der Anwendung und erklären ihre Patterns, anstatt sie nur zu beschreiben. [Gamma et al. 2004]. Auch wenn sich das Format der Patterns von den Architektur Pattern zu den GoF Patterns vom Grundprinzip her nicht viel verändert hat - Name, Kontext, Problem, Lösung, Beispiele, Diagramme, und Verweise sind immer noch essentielle Bestandteile von jedem Pattern – so sind jedoch einige der grundsätzlichen Aspekte Alexanders in dem Prozess der Adaptierung verloren gegangen, womit sich auf relevante Weise die Ziele geändert haben.

Software Patterns werden als nützliche Sprache für die Kommunikation betrachtet und als praktisches Hilfsmittel um Entwickler, die in objektorientierter Programmierung weniger erfahren sind in diesen Bereich einzuführen. Jedoch ist der Patternkatalog weder für Einsteiger gedacht, die über allgemein wenig Programmiererfahrung verfügen – es wird erwartet, dass der Benutzer sich bereits in mindestens einer objektorientierten Programmiersprache relativ gut auskennt und auch über einige Erfahrung im objektorientierten Entwurf verfügt - und schon gar nicht für Laien, die sich im Bereich des Software Engineering nicht auskennen. Ziel des Patternkatalogs ist es, neuen Entwicklern den objektorientierten Entwurf zu vermitteln. Die wesentliche Idee Alexanders, dass die Endbenutzer ihre eigene (Software) Architektur entwerfen wurde nicht übernommen. Auch liegt bei Alexanders Pattern der Schwerpunkt auf den von ihnen adressierten Problemen, während die Entwurfsmuster von [Gamma et al. 2004] sich stärker auf die Lösungen konzentrieren und sie herausarbeiten.

Auch stellen, wie [Gamma et al. 2004] selbst anmerken, die Softwarepatterns lediglich ein Katalog von Patterns dar und keine eigene Patternsprache. [Borchers 2001] begründet dies damit, dass der Patternkatalog und das Verlinken zwischen den einzelnen Patterns nicht komplett genug ist, um eine Sprache zu sein. Wie sie selbst betonen, ist es für sie schwer erkennbar, wie man angesichts der Bandbreite an Softwaresystemen, einen „kompletten" Satz an Pattern bereitstellen könnte, der schrittweise Instruktionen zum Entwurf von Anwendungen bietet. Dagegen gibt Christopher Alexander an, dass seine Pattern vollständige Gebäude erzeugen können. Die *Gang of Four* stellt nicht den Anspruch, dass ihre Patterns vollständige Programme erzeugen werden.

Darüber hinaus stellt [Borchers 2001] fest, dass viele der GoF Patterns nicht das Wissen und die Konzepte von gutem objektorientiertem Design repräsentieren, wie man es durch Expertenwissen erhält, sondern viele Patterns stellen Abhilfen dar um objektorientierte Konzepte zu implementieren. Typische Patterns aus der Sammlung wenden über 50% ihres Textes für die Implementierungsdetails und Beispielcodelisten auf. Sicherlich kann ein professioneller Software Engineer viel von diesen Patterns lernen, um seine Software Entwürfe und Implementierungsfähigkeiten zu verbessern, jedoch sind sie nicht für eine allgemeinere Zielgruppe gedacht. [Borchers 2001]

Auch gibt Alexander eine Reihenfolge für die Verwendung der Patterns vor, *Gang of Four* tun dies nicht. Sie gehen sogar soweit zu sagen, dass es unwahrscheinlich ist, dass es jemals eine komplette Patternsprache für Software geben wird. Allerdings räumen sie ein, dass es sicherlich möglich ist, eine Patternsprache zu erstellen, die umfassender ist als ihr eigener Katalog. Erweiterungen des Katalogs müssten Frameworks und ihre Verwendung umfassen, Patterns für den Entwurf von Benutzungsschnittstellen und alle anderen Aspekte, die zum Entwurf von Software gehören. Software Patterns sind lediglich nur ein Teil einer größeren Patternsprache für Software. [Gamma et al. 2004]

Wie bereits erwähnt, fordert Christopher Alexander, dass sich gute Patterns mit der Zeit und einem weitläufigen Gebrauch verfeinern und verbessern. Auch [Gamma et al. 2004] sehen wie Christopher Alexander ihren Katalog lediglich als einen Beginn des Dokumentierens von guten Entwürfen. Er enthält einige der bekanntesten Patterns, die von Experten im objektorientierten Entwurf verwendet werden. Diesen Katalog soll der Anfang einer Bewegung sein, welche die Expertise von erfahrenen Softwareentwicklern dokumentieren wird. Man soll ein kritischer Pattern-Konsument sein und Autoren von Patterns Feedback geben, bzgl. Problemen mit Patterns oder Anregungen zu Erweiterungen oder tiefer gehenden Erläuterungen, Patterns diskutieren, bzw. sogar eigenen Patternkatalog schreiben.
Die objekt-orientierte Software Engineering Community bewies, dass dies erreicht werden kann, da sich an der Entwicklung eigener Patternsprachen nun viele Leute beteiligten und sie durch die Community noch mal überprüfen ließen. Die Diskussion um Patterns im Software Engineering setzte sich weltweit auf Konferenzen, in Magazin-Kolumnen, über

Mailing-Listen und speziellen lokalen Interessensgruppen fort. [Tidwell 1999] Im Zuge dessen haben auch viele andere Forscher Ideen entwickelt wie man das Prinzip der Patterns auf das Software Engineering adaptieren kann. Die Etablierung eines gänzlich neuen und nützlichen Forums um geprüfte, allgemeingültige Lösungen für wiederauftauchende Softwaredesignprobleme in der Community auszutauschen, das im Rahmen der jährlich stattfindenden *Pattern Languages of Programming (PLoP)* Konferenzen entstand, wurde erfolgreich aufgenommen.
Seit der erfolgreichen Patternsammlung der *Gang of Four*, welches die Software Community nachhaltig beeinflusste, wurden viele Bücher über Softwarepatterns veröffentlicht, wie die *Patterns Languages of Program Design (PLoPD) Serie*, Sammlungen von Papern von verschiedenen Software Patternkonferenzen und der *Pattern-Oriented Software Architecture Serie* mit seinen Mehrzweck-, Gleichzeitigkeits- und Netzwerkpatterns. [Borchers 2001]

Wie auch Christopher Alexander, bemerken [Gamma et al. 2004], dass keines der GoF Patterns neuartige Entwürfe beschreibt. Es werden keine neuen Algorithmen oder Programmiertechniken präsentiert, nichts was nicht schon zuvor bekannt war und genutzt wurde. Die Patterns berücksichtigen lediglich solche Entwürfe, die bereits mehrfach angewendet wurden und sich in unterschiedlichen Systemen bewährt haben. Dennoch wurden viele dieser Entwürfe noch nie dokumentiert, da sie entweder zum Allgemeinwissen objektorientierter Entwickler gehören oder Teil erfolgreicher objektorientierter Systeme, die sich aber Anfänger zumeist nicht leicht erschließen können. Somit erfasst der Software Pattern Katalog zwar keine neuen Entwürfe, jedoch in einer neuen und leicht zugänglichen Weise: als ein Katalog von Pattern mit einem einheitlichen Format. [Gamma et al. 2004]

4.1.3 Human-Computer Interaction Pattern

Auch im Bereich der HCI stieß die Patternidee von Christopher Alexander auf reges Interesse. Allerdings erhielt sie im akademischen und professionellen Feld der HCI weitaus weniger Aufmerksamkeit als im Software Engineering. Entgegen der langläufigen Meinung, wurde laut [Borchers 2001] die Idee der Patterns früher noch als im Software Engineering referenziert. Erste Referenzen tauchen in Norman und Drapers wegweisendem Buch über *User-Centered Systemdesign* auf, welches sowohl bei Forschern als auch bei Praktikern des User Interface Design als Standardwerk gilt. Eine der ersten Homepages im HCI Bereich, die sich mit Patterns beschäftigte und Publikationen und Patternsammlungen präsentierte, wurde 1998 von Thomas Erickson[8] etabliert.

[8] The Interaction Design Patterns Page
URL: http://www.visi.com/~snowfall/InteractionPatterns.html

Seit 1997 erfährt das Thema der Patternsprachen in der HCI zunehmendes Interesse und der erste Workshop in welchem Pattern Communitys von Software Engineering und HCI in direkten Kontakt zueinander traten, fand auf der *ChiliPLop'99*, eine der drei jährlichen Konferenzen in der *PLoP Konferenzserien*, statt. [Borchers 2001] Im Zuge dessen entstanden mehrere Patternsammlungen für die HCI, wie die Usability Patternsammlung von Kimberly Perzel und David Kane, die Patternsammlungen für Web Design, GUI Design und MobileUI Design von Martijn van Welie, die Patternsammlung von Jan Borchers und die häufig zitierte Patternsammlung von Jenifer Tidwell, *Common Ground*.

Die Patternsammlung von Kimberly Perzel und David Kane, welche speziell für das Usability Engineering von Web Applikationen ausgerichtet ist wurde 1999 in dem Artikel *Usability Patterns for Applications on the World Wide Web* veröffentlicht und beinhaltet lediglich fünf ausgearbeitete Patterns. Obwohl die Sammlung nur wenige Patterns umfasst, wird sie häufig referenziert. Perzel und Kane unterteilen die Patterns in zwei Hauptkategorien: *User Interface Patterns*, welche sich mit der Gestaltung des Aussehens einer Webseite befassen und *System Patterns*, welche sich mit Usability Fragen beschäftigen, die nur weitläufig zum UI gehören, wie z.B. *Policy Statement*, *Server-Side Validation* oder *Client-Side Validation*. [Pfründer 2002][Perzel&Kane 1999]

Im Jahr 2000 veröffentlichten van Welie und Troetteberg 20 *User Interaction Patterns*, genannt *Amsterdam-Collection*, die speziell darauf ausgerichtet sind, die Usability eines Systems zu verbessern. Jedes der Pattern bezieht sich auf eines der folgenden Usability Prinzipien: *Visibility, Affordance, Natural Mapping, Constraints, Conceptual Model, Feedback, Safety und Flexibility*. Diese GUI Design Patternsammlung, welche bisher um sechs weitere Patterns erweitert wurde, und weitere Patterns, die sich speziell mit Web Design (113 Patterns) und MobileUI Design (sieben Patterns) befassen, veröffentlichte van Welie auf einer Webseite, die es dem Benutzer auch erlaubt sich via Postings auf der Webseite aktiv an der Weiterentwicklung und Verbesserung dieser Patterns zu beteiligen. Jedoch spricht van Welie bzgl. seiner Patternsammlung noch nicht von einer Patternsprache, da er der Meinung ist, dass eine solche erst entstehen kann, wenn genügend Patterns gefunden und verifiziert wurden. Denn Gültigkeit und Zustimmung sind Bedingungen für eine Patternsprache und nicht alles, was in einem Patternformat niedergeschrieben wurde, ist tatsächlich ein Pattern und sollte als solches akzeptiert werden. [van Welie et al. 2001][Pfründer 2002]

Jan Borchers sieht im Vergleich zu den anderen Patternautoren die Idee der Patterns wesentlich umfassender. Er ist der Meinung, dass Pattern in jedem beliebigen Bereich angewendet werden können und hat im Zuge dessen ein allgemeingültiges Modell entwickelt, um die Designbelange des HCI, des Software Engineerings und des Anwendungsbereichs eines Projekts auszudrücken. Seiner eigenen Ansicht nach ist sein Ansatz der erste, welcher eine Cross-disziplinäre Methode vorschlägt, um Patterns im Designprozess von interaktiven Systemen auf strukturierte und gleichförmige Weise zu

verwenden. Darüber hinaus erkannte er, dass das Patternformat nicht nur eine geeignete Form darstellt, Designerfahrung für Folgeprojekte zu dokumentieren, sondern auch um unerfahrene Designer in wichtige HCI Konzepte einzuführen und diese auch als Unterrichtshilfe in HCI Designkursen verwenden zu können. Dabei adressiert Borchers mit seinen Patterns auch Laien - die im Sinne des User Interface Design -, wie z.B. Anwendungsbereichesexperten oder auch Software Engineers.

Als Ergebnis seines allgemeinen Pattern-basierten Ansatzes präsentiert er drei Patternsprachen, die HCI-, Software Design- und Anwendungsbereichskonzepte beschreiben und zusammen ein interdisziplinäres Patternsprachen Framework bilden, welches das Ziel hat, Designerfahrung in der HCI, dem Software Engineering und dem Anwendungsbereich eines Software Projekts zu sammeln und auszudrücken. Es ist sowohl speziell an die HCI angepasst, als auch an das Software Engineering sowie an den jeweiligen Anwendungsbereich eines Projekts. Letzteres ist ein neuer Ansatz, der Konzepte aus der Disziplin, in welcher die Software verwendet werden wird, in Form eines Pattern ausdrückt, welche wiederum in einer Patternsprache geordnet sind.

In seinem Buch *A Pattern Approach to Interaction Design* präsentiert er ein Beispiel dieses Frameworks für die interaktive Musikausstellung *WorldBeat*. *WorldBeat* lässt die Benutzer in gänzlich neuen Wegen mit Musikkonzepten interagieren. Dies umfasst beispielweise das Spielen virtueller Trommeln, das Auffinden von Liedern lediglich durch Summen von Melodien oder das Improvisieren einer Bluesband unter Verwendung eines Paars Infrarot-Taktstöcken. Das Framework enthält eine HCI Patternsprache, welche die Aufgabe des Entwurfs von interaktiven Systemen adressiert, die in öffentlichen Räumen verwendet werden. Typische Systeme, die in diese Kategorie fallen, beihalten interaktive Ausstellungen in Museen und Ausstellungszentren, wie auch andere öffentliche Systeme, die allgemein als Kiosksysteme bezeichnet werden. Sie ist mit 17 Patterns die größte der drei Sprachen. Des weiteren präsentiert er eine aus vier Pattern bestehende Software Patternsprache, welche sich auf Patterns konzentriert, die speziell zum Software Design für interaktive Musiksysteme im Bezug stehen. Und eine Musik Patternsprache, die aus insgesamt 11 Anwendungsbereichs-Patterns besteht, welche die wichtigsten Prinzipien des Musikstils Blues beschreiben, die in der *WorldBeat* Ausstellung verwendet werden.

Für jede der drei Sprachen präsentiert Borchers einen Graphen, der am Anfang jeder Sprache zu finden ist. Er zeigt die Namen aller Pattern dieser Sprache und ihre gegenseitige Kopplung durch Referenzen. Die Hierarchie der Patterns entspricht dabei der Dimension ihrer Belange, die sie adressieren. Zusätzlich sind die Patterns der Musik Patternsprache von links nach rechts in Gruppen arrangiert, die harmonische, melodische and rhythmische Aspekte eines Stückes adressieren. Im Fall der Musik Patternsprache wird die Aufgabe des Komponierens oder Improvisierens eines Bluesstückes als Designaktivität betrachtet, deren Ergebnis die Musik ist. Der Benutzer wird von größer dimensionierten Belangen zu kleiner dimensionierten Belangen geführt. Auch wenn die tat-

sächliche Durchführung nicht einem ausgearbeiteten *top-down* Designprozess folgen wird, so wird durch die Strukturierung der Patterns in eine Patternhierarchie doch das Erlernen von Blues vereinfacht.

Borchers Ansatz fokussiert explizit auf die Benutzerbedürfnisse, die Benutzerteilnahme und die Modellierung einer Umgebung, in welcher Menschen sich wohl fühlen. Somit ist für ihn auch die Allgemeinverständlichkeit der Patterns und die Bildung einer Patternsprache oberstes Kriterium, so dass es Benutzern möglich ist, aktiv an der Entwicklung eines Systems Teil zu haben. Damit orientiert sich Borchers allgemein sehr stark an der ursprünglichen Idee des Patternansatzes von Christopher Alexander. Er verwendet auch, bis auf einige wenige Änderungen, die selbe Formatvorgabe für die Strukturierung der Patterns wie Alexander. Jedoch, im Gegensatz zu den anderen Autoren, die sich mit der Entwicklung von Patterns beschäftigen, verwendet Jan Borchers für alle Sorten seiner Patterns das gleiche Format und passt dieses nicht der jeweiligen Disziplin an. Die Patternsprachen, welche Borchers während des Entwurfs der *WorldBeat* entwickelt hat, wurden in einer Reihe von Folge-Projekten wiederverwendet und weiter verfeinert. Auch Jan Borchers fordert auf, die gegeben Patterns weiter zu verfeinern und neue zu finden und sie durch die Community validieren zu lassen. Insbesondere die Idee das Prinzip der Patterns auf einen Anwendungsbereich zu übertragen, sollte seiner Ansicht nach durch eine erneute Anwendung auf einen anderen Anwendungsbereich validiert werden, um die generelle Gültigkeit dieses Ansatzes zu unterstreichen. Allerdings müssen diese Bereiche einige Aspekte einer Problemlösungs- oder „Design-Aktiviät" aufweisen, um diesen Ansatz zu ermöglichen.[Borchers 2001]

Jenifer Tidwell: Common Ground
A Pattern Language for Human-Computer Interface Design

Zu den bekanntesten Patternsammlungen in der HCI gehört die von Jenifer Tidwell publizierte Patternsprache für Human-Computer Interface Design, *Common Ground*. Diese Sammlung adressiert das allgemeine Problem des Entwerfens von komplexen interaktiven Softwareartefakten. [Borchers 2001]
Jenifer Tidwell orientiert sich bei der Entwicklung ihrer Patterns stark an der ursprünglichen Idee Christopher Alexanders. Auch sie hat das Ziel, wie Alexander, eine eigene Patternsprache zu entwickeln, in der zeitlose Prinzipien von gutem Interaktionsdesign in einer hierarchisch strukturierten, verlinkten Gruppe von Patterns repräsentiert werden, mit dem Ziel, Designer auf verschiedenen Leveln der Abstraktion durch den Designprozess zu führen. Ganz wie Christopher Alexander sollen die Patterns dabei synergetisch zusammen verwendet werden, so dass das Ganze mehr ist als die Summe seiner Teile. [Borchers 2001][Tidwell 1999]

Die Sammlung von Jenifer Tidwell, welche derzeit 53 veröffentlichte und sieben bisher unveröffentlichte Patterns umfasst, ist, obwohl das Format nicht vollständig uniform verwendet wird und nicht alle Pattern ausgearbeitet und komplett sind, für Außenstehende gut lesbar geschrieben. Ihre Patterns sind primär für Personen gedacht, die traditionelle User Interfaces, Webseiten, Online Dokumentationen, Videospiele und Ähnliches gestalten, aber auch sekundär für Personen, die für die Implementierung zuständig sind, Usability Tests durchführen oder aber mit der Leitung des Designteams betraut sind.

Jede Patternbeschreibung definiert einen Kontext of Use, ein Problem, das der Designer lösen möchte, ein Gruppe von Randbedingungen, die den Designer in verschiedene Richtungen lenken und eine Hauptregel – und manchmal zusätzliche Sekundärregeln – wie diese Randbedingungen geklärt werden sollten, mit dem Ziel das Problem auf die bestmöglichste Weise zu lösen. Zusätzlich werden Beispiele angeboten und zwar sowohl gute als auch manchmal schlechte, wobei die schlechten Beispiele unsachgemäße Verwendung des Pattern zeigen können als auch Situationen, in denen das Pattern hätte verwendet werden sollen, aber es nicht getan wurde. Das Format der *Common Ground* Patterns sieht wie folgt aus: *Name, Examples, Context, Problem, Forces, Solution, Resulting Context, Notes*. Eine genaue Beschreibung der Patterns findet sich auf der Webseite der Patternsammlung (s.o.).

Tidwell verwendet für ihre Patternsprache eine Hierarchie, die sich gut mit einem typischen *top-down* User Interface Designprozess deckt.[Borchers 2001] Dabei ist ihre Hierarchie von oben nach unten, aber auch von unten nach oben anwendbar. Die Patterns gruppieren sich dabei in verschiedene Problem-Unterbereiche mit variierenden Detaillevel und adressieren zumeist entweder eine Frage der Präsentation des Inhalts oder der Verfügungsstellung von Aktionen. Was jedoch atypisch für eine Patternsprache in Alexanders Sinne ist, ist die Tatsache, dass die meisten dieser Patterns auf verschiedenen Ebenen verwendet werden können. Was bedeutet, dass beispielsweise das Pattern „Form" in einem Artefakt das dominierende Pattern sein kann, wogegen es in einem anderen unbedeutende Helferaufgabe inne hat. [Tidwell 1999]

Wie schon Christopher Alexander und Gang of Four betont Jenifer Tidwell, dass die von ihr präsentierte Patternsammlung lediglich ein Anfang einer Patternsprache darstellt, da diese, nach ihren eigenen Aussagen, weit davon entfernt ist komplett zu sein und die bestehenden Pattern weiter verfeinert und neue Patterns hinzugefügt werden sollten. Auch sie fordert die Nutzer und Leser der Patterns auf, diese kritisch zu überdenken, Verbesserungsvorschläge zu machen, sie zu diskutieren und neue Patterns zu finden. [Tidwell 1999]

Wie beschrieben, wurden in der HCI von Forschern bereits einige Versuche unternommen Patterns zu erzeugen. Jedoch bestand bislang kein Konsens darüber, wie Patterns für die HCI niedergeschrieben, wie sie strukturiert und welchen Fokus sie haben sollten. Denn ein

Pattern für die HCI ist nicht notwendigerweise auf dieselbe Weise strukturiert wie ein Architekturpattern und es gilt ein Format zu finden, das für speziell für UID entwickelt wurde und die richtige Sicht auf die wichtigen Belange, wie beispielsweise Probleme hinsichtlich der Usability, in UID hat. [van Welie et al. 2001] Bislang hat jeder Forscher für das Schreiben von Patterns sein eigenes Format verwendet. Trotz der existierenden Unterschiede enthalten die meisten der vorgeschlagenen Patternformate die folgenden Elemente: einen Patternnamen, anhand dessen das Pattern referenziert werden kann, eine Problembeschreibung, ein Kontext in dem das Problem auftaucht, die Randbedingungen, die eine Lösung behindern können, eine geprüfte Lösung und Beispiele der Lösung in der Praxis und schließlich Links zu anderen verwandten Patterns, die verwendet werden können, um die Lösung zu verfeinern. Jedoch können einige der Elemente als optional betrachtet werden.[Wilkins 2003]
Die verschiedensten Forscher griffen die Idee der Patterns auf und publizierten Pattern-sammlungen für den Bereich der Human-Computer Interaction. Bislang wurde jedoch laut [van Welie et al. 2001] noch keine maßgebende und anerkannte Sammlung solcher Patterns entwickelt, wie sie mit der Patternsammlung der Gang of Four im Software Engineering vergleichbar wäre.

Prinzipiell existieren große Parallelen zwischen HCI Design und Architektur bzw. urbanen Design. Sowohl in der Architektur als auch in der HCI liegen die zentralen Belange in der Beziehung zwischen Benutzer und Umgebung, ob physisch oder virtuell. Dies betrifft z.B. die folgenden Aspekte: wie erfährt der Benutzer seine Umgebung, wie beeinflusst eine Umgebung das Verhalten des einzelnen, wie integrieren sich einzelne Umgebungen in einen größeren Kontext und wie entwickeln und verändern sich diese kleineren und größeren Umgebungen im Laufe der Zeit. Patterns beziehen sich auf Beziehungen zwischen physischen Elementen und den Events, die in dieser Umgebung geschehen. Sowohl Interface Designer als auch Architekten haben das Ziel Umgebungen zu erzeugen, die bestimmtes Verhaltenspattern etablieren mit einem positiven Effekt auf diese Menschen „innerhalb" dieser Umgebungen. Die *Quality Without a Name*, die Alexander fordert als gute, "zeitlose" Architektur ist vergleichbar mit den Interfacequalitäten, wie *Transparent* und *Natural*. [Borchers 2001]

Der wesentliche Unterschied im HCI Design und der Architektur ist jedoch die temporale Dimension, welche dem UI Design, im Gegensatz zur Architektur zu eigen ist. Das User Interface eines interaktiven Computersystems ist eine dynamische Umgebung, welche für gewöhnlich seine Erscheinung und sein Verhalten im Verlauf der Interaktion wesentlich verändert. Die Interaktion ist wesentlich dynamischer und Kontext und System der Rahmenbedingungen wechseln während des Ablaufs der Interaktion häufig. Alexanders Pattern beschäftigen sich dagegen nahezu exklusiv mit räumlichen Konfigurationen. [Borchers 2001]

4.1.4 Visualization Pattern

Da eine Visualisierung als eine visuelle Repräsentation eines Datensatzes betrachtet werden kann, mit welcher der Benutzer via *Graphical User Interface (GUI)* interagieren kann, gelten für deren Design auch HCI Patterns. Jedoch beinhaltet die Thematik der Visualisierung von Daten noch weitere Themen als im Standard Interfacedesign gefunden werden können. Themen wie Daten- oder Variablentyp, Quantität der Daten, visuelle Strukturen und Layout oder Interaktionsmechanismen, wie Selektion, Navigation und Manipulation der Views, müssen beim Visualization Design zusätzlich beachtet werden.
Daher wurde auch im Bereich der Visualisierung von Daten, die Idee der Pattern von Christopher Alexander aufgenommen und für das Visualization Design adaptiert, wenn auch bislang auch noch verhalten. So beschreibt beispielsweise Barry Wilkins in seiner Doktorarbeit *MELD: A Pattern Supported Methodology for Visualisation Design* einen visualisierungsspezifischen Ansatz für die Verwendung von Patterns.

Barry Wilkins entwickelte seine Visualization Patterns zum einen auf der Basis von visualisierungsspezifischen Heuristiken als auch auf der von Techniken, die im Bereich der Visualisierung von Daten immer wieder verwendet werden. Dabei sind sowohl die entwickelten Heuristiken als auch die Techniken domänenunabhängig.[Wilkins 2003]

Die erste Basis, die visualisierungsspezifischen Heuristiken, sind mit den Heuristiken oder Guidelines aus dem Interface Design vergleichbar und basieren auf den Erfahrungen, die beim Design, Implementieren und Evaluieren von Visualisierungen, von Visualisierungs-forschern über Jahre hinweg gesammelt wurden. Über die Vorteile und Unterscheide von Patterns gegenüber Heuristiken oder Guidelines haben verschiedene Forscher, wie z.B. Martijn van Welie oder auch Jan Borchers reflektiert. [vgl. Welie et al. 2000]. Dabei lassen sich allgemein drei wichtige Unterschiede zusammenfassen: Zum einen erfassen Patterns alle Informationen, die normalerweise erforderlich sind, um eine Guideline zu benutzen. Sowohl Kontext, Problem als auch Lösung werden explizit gemacht mit einer zusätzlichen Begründung, die erläutert weshalb die Lösung funktioniert. Folglich kann ein Pattern also verschiedene Guidelines kombinieren, die oftmals die Basis vieler Patterns formen. Zum anderen müssen Patterns eine geprüfte Problemlösung anbieten und Beispiele der Lösung in der Praxis. Der dritte Unterschied besteht darin, dass Patterns in Patternsprachen organisiert werden, die dem Designer erlauben schnell zu navigieren und Lösungen zu verfeinern. Eine solche hierarchische Organisation ist mit Guidelines bzw. Heuristiken nicht möglich. [Wilkins 2003]

Die zweite Basis leitet sich aus der Tatsache ab, dass viele der Techniken, die im Bereich der Visualisierung von Daten entwickelt wurden, - wie beispielsweise die *Overview and Detail* Technik, welche in vielen Visualisierungen immer wieder bevorzugt eingesetzt wird - immer wieder verwendet werden, um wiederkehrende Probleme zu lösen, was als solches der Definition eines Pattern entspricht. Daher sollte es möglich sein, diese Techniken in

Patterns zu formalisieren und sie zu organisieren um eine Patternsprache zu formen. Visualisierungsspezifische Patterns können verwendet werden um Entwürfe zu erzeugen, welche diese gut entwickelten Techniken berücksichtigen.[Wilkins 2003]

Visualization Pattern

Wilkins teilt die von ihm entwickelten Visualization Patterns in drei Hauptkategorien ein, wobei es dabei zu Überschneidungen kommen kann [Wilkins 2003]:

- **Strukturpatterns**
  Strukturpatterns fokussieren in erster Linie auf die Definition der Form und den Inhalt einer Visualisierung.
- **Interaktionspatterns**
  Interaktionspatterns fokussieren auf die Interaktionsmechanismen, die der Benutzer verwenden kann, um die Aufgaben (*Tasks*) auszuführen und die visuellen Effekte, die sie erzeugen.
- **Kompositionspatterns**
  Kompositionspatterns beinhalten Lösungen für das effektive Layout multipler Visualisierungen und die Kommunikation zwischen ihnen.

Format der Visualization Pattern

Die Struktur, die Wilkins für seine Visualization Patterns gewählt hat, wird unten aufgeführt. Jedoch gibt er innerhalb seiner Dissertation keine genaue Beschreibung dazu, wie sich die einzelnen Elemente im Detail definieren.

- **Titel**
- **Context**
- **Problem**
- **Forces**
- **Solution**
- **Examples**
- **Related Patterns**

Patternsprache

Wilkins formte aus den, im Rahmen seiner Dissertation, entwickelten Visualization Patterns beispielhaft zwei Patternsprachen, die *Structure Pattern Language (SPL)* und die *Interaction Pattern Language (IPL)*. Die SPL besteht hauptsächlich aus Struktur Patterns, weshalb sie auch als *Structure Pattern Language* bezeichnet wird. Verschiedene der Visualization Patterns befinden sich auf derselben Ebene und können bei den meisten Visualisierungsdesigns verwendet werden. Ebenso können einige der von Jenifer Tidwell entwickelten GUI Designpatterns in diese Sprache eingefügt werden. Wie Wilkins angibt, ist die Sprache nicht komplett, da es zweifellos weitere Patterns auf und zwischen jeder

Ebene gibt. Auch die konstante Entwicklung neuer Techniken und Lösungen und das Entdecken neuer Patterns führen dazu, dass existierende Patternsprachen permanent verfeinert und weiterentwickelt werden müssen. Jedoch bietet sie dem Visualization Designer, laut seinen eigenen Angaben, bereits in diesem Stadium, ungeachtet der offensichtlichen Lücken, nützliches Designwissen.
Die *IPL* besteht, die im Gegensatz zur *SPL*, hauptsächlich aus Interaktionspatterns und wird daher *Interaction Pattern Language (IPL)* genannt, obwohl es Verbindungen zur *SPL* gibt. Zusätzlich in den Graphen eingefügte Knoten, welche keine tatsächlichen Patterns darstellen, sollen dem Designer helfen den Kontext der Patterns, auf den weiter unten liegenden Ebenen zu definieren und ihm somit bei der Navigation in der Patternsprache helfen. So wie die *SPL* ist auch die *IPL*, laut Wilkins, nicht komplett. Auch hier können neben weiteren Visualization Patterns GUI Designpatterns von Jenifer Tidwell mit in die Sprache eingefügt werden, wie beispielsweise das GUI Designpattern *„Controls*", um Parameterwerte für Filterfunktionen zu setzen oder das Pattern *„Scrollbars*", um verschiedene Bereiche des Datenraums anzuzeigen. [Wilkins 2003]

Auch Wilkins betont, wie auch die Forscher aus den anderen Disziplinen, dass die von ihm entwickelten Patterns ein erster Versuch sind, eine Gruppe von Visualisierungsspezifischen Patterns zu erzeugen, die jedoch weder als definitiv verstanden werden sollten noch einen Anspruch auf Vollständigkeit erheben. Folglich sind nach eigenen Aussagen, eine Reihe der Patterns sehr unpräzise und könnten sogar überhaupt keine Patterns sein. Jedoch helfen diese eine komplettere Patternsprache zu formen, da sie oftmals den Kontext und die Randbedingungen festlegen, welche bei Patterns auf weiter unten liegenden Ebenen angewendet werden. Eventuell sollten einige Patterns aus der Sammlung entfernt oder kombiniert werden, in jedem Fall sollten aber neue hinzugefügt werden. Jedoch ist dieses so genannte *Pattern Mining* eine schwierige Aufgabe, da es häufig schwierig zu beurteilen ist, ob ein Pattern tatsächlich ein Pattern ist. Darüber hinaus wird neuerdings von Forschern angeregt, ebenfalls nach sogenannten Anti-Patterns zu suchen und diese auf ähnlich schriftliche Weise zu belegen. Dies würde Designer davor bewahren ungeeignete Designtechniken für ihre Entwürfe zu verwenden. Die Verifikation der Patterns soll durch andere Forscher und durch ihre Anwendung geschehen. [Wilkins 2003]

4.1.5 Anti-Pattern

Der allgemeine Erfolg der Patterns führte zu der Entwicklung von Anti-Patterns. Diese erläutern, weshalb Dinge nicht richtig funktionieren, um im Anschluss daran eine Lösung für das Problem zu bieten. Man kann ein Anti-Pattern demnach als ein Pattern betrachten, dem ein Beispiel für schlechtes Design vorausgeht. Für den Bereich des User Interface Design wurden bereits eine Reihe von Beispielen für schlechtes Design dokumentiert (z.B.

Interface Hall of Shame[9]). Auch wenn das Betrachten von schlechtem Design inspirierend sein mag, so stellt es jedoch keine direkte Hilfe für das Lösen von Problemen dar. Van Welie regte die Idee an, Patterns und Anti-Patterns miteinander zu kombinieren, indem das Pattern mit einem Beispiel von dem was sehr wahrscheinlich geschieht, wenn das Pattern nicht verwendet wird, erweitert wird. Insbesondere beim UI Design mag dies sehr anschaulich sein, da hier die Problematik zumeist schon durch das Betrachten eines einzelnen Screenshots erkannt werden kann.[van Welie et al. 2001]

4.1.6 Pattern und XML

Wie bereits erwähnt, sind die verwendeten Patternformate sehr heterogen. Sie unterscheiden sich nicht nur von Disziplin zu Disziplin sondern auch von Autor zu Autor. Dieser Problematik wird versucht mit Hilfe der Auszeichnungssprache *XML (Extensible Markup Language)* entgegen zu wirken, um mit der Verwendung eines XML-Standards ein einheitliches Patternformat zu schaffen. XML ist ein Standard zur Erstellung maschinen- und menschenlesbarer Dokumente in Form einer Baumstruktur, der die Regeln für den Aufbau solcher Dokumente definiert. Diese Sprache besteht aus Elementen, welche ganz unterschiedliche Daten enthalten und beschreiben können, wie z.B. Text, Grafiken oder abstraktes Wissen und den entsprechenden Regeln. Für die Beschreibung der Struktur von XML-Dokumenten werden Metasprachen, die sog. Schemasprachen genutzt. Zu den bekanntesten gehören hierbei DTD (Dokumenttypdefinition) und XML Schema (XSD). Der Grundgedanke von XML ist, Daten und ihre visuelle Repräsentation voneinander zu trennen. Dies ermöglicht beispielsweise, aus einer einzelnen, im XML-Format definierten Datenbasis unterschiedliche Darstellungsformen gewinnen zu können. [Wikipedia 2005]

Manche der Patternsammlungen, die online verfügbar sind, nutzen bereits XML, um ein konstantes und standardisiertes Format für die Veröffentlichung der Patterns zu erzeugen. XML bietet verschiedene Möglichkeiten des automatischen Indexieren und Kategorisieren der Patterns. Eine erste solche Spezifikation für User Interface Designpatterns wurde von Martijn van Welie entwickelt. Den Autoren der Patterns wird somit die Möglichkeit geboten, diese unter Verwendung des Pattern DTD, welches bewirkt, dass alle Pattern in einem konsistenten Weg dargestellt werden, zu übermitteln. [van Welie et al. 2001] [Borchers 2001] Von den im Zuge dieser Arbeit betrachteten Patternsammlungen nutzen derzeit Jan Borchers und Martijn van Welie XML für ihre Patternsammlungen.

[9] Interface Hall of Shame
URL: http://www.iarchitect.com/shame.htm

4.2 Vorstellung des Box-of-Ideas Ansatzes

Wie bereits erwähnt, gibt es im Designbereich eine ganze Reihe von Kreativitätstechniken und -tools, die den Designprozess neuer Produkte oder auch die Überarbeitung bestehender Produkte, insbesondere in der Ideenfindungsphase unterstützen. Eine der Tools, die vor allem durch die renommierte amerikanische Designagentur IDEO[10] perfektioniert und bekannt wurde, ist das Prinzip der Box-of-Ideas, der sog. *Tech Box*. Diese stellt eine physische und webbasierte Bibliothek interessanter Objekte dar, die IDEO als Inspiration für Innovation im Designprozess dienen. [IDEO 2005a]

IDEO entstand aus einem Zusammenschluss der beiden Firmen David Kelley Design, einer Firma für mechanisches Design und der Industrial Design Firma ID2 von Bill Moggridge. ID2 rief in den 80er Jahren den Begriff des Interaction Design ins Leben um seine Arbeit bezüglich grafischen User Interfaces und Smart-Produkten zu beschreiben. Mittlerweile gehört IDEO zu den weltgrößten und erfolgreichsten Agenturen für Produktdesign, Engineering und Designberatung mit Büros in London, San Francisco, Palo Alto, Boston, Chicago und Tokio. In interdisziplinären Teams entwickeln sie in enger Zusammenarbeit mit dem Kunden neue Produkte, Dienstleistungen und Lösungen. [Preece et al. 2002] [IDEO 2005a] Dabei reicht die Produktpalette von IDEO von medizinischer Hightech-Ausrüstung, über das physische Design vieler bekannter Computersysteme, inklusive dem Apple Computer Gehäuse und die farbenreichen Silicon Graphics Workstations, die erste kommerzielle Computermaus von Apple, den Palm Pilot V bis zu einem mechanischen Wal, der für Spezialeffekte in dem Film Free Willy verwendet wurde. Die Fähigkeit die ungleichen Kulturen von Engineering und Design miteinander zu integrieren wird als unverkennbares Markenzeichen von IDEO betrachtet. IDEO kombiniert routinemäßig kreative Innovationen mit pragmatischem Engineering und Produktion. [Winograd 2002]

4.2.1 Designprozess bei IDEO

Laut David Kelley, dem Gründer von IDEO, beruht der Erfolg von IDEO nicht nur auf den Fähigkeiten von einzelnen Designern, sondern vielmehr auf dem Gesamtansatz bezüglich des Design, welcher die Gruppenarbeit und die cross-funktionale Entwicklung betont. IDEO agiert als ein Verband kleinerer Designbüros mit verschiedenen Stilen, welcher ihm die Vielfalt gibt, die Breite seiner Designinteressen in einem gemeinsamen Ansatz bezüglich der Methodologie zu unterstützen. Der stark am Benutzer orientierte Designprozess von IDEO erfordert einen intensiven Fokus sowohl auf das Verstehen der Welt des Benutzers, sowie auf die gegenseitige Befruchtung der Disziplinen, Stile und Persönlichkeit als auch auf die Fähigkeit sich zur selben Zeit auf Funktionalität, Ästhetik und Produzierbarkeit

[10] IDEO
URL: http://www.ideo.com

konzentrieren zu können. [Winograd 2002] Diese Methodologie wurde über die Jahre erforscht und dient als Grundlage zahlreicher erfolgreicher Produktentwicklungen in verschiedenen Anwendungsbereichen und hat IDEO die weltweit höchste Anzahl international anerkannter Designauszeichnungen eingebracht. [Form 2005] Sie basiert auf einer kollaborativen Methodik, die zugleich die Benutzerbedürfnisse, die technische Machbarkeit und die wirtschaftliche Realisierbarkeit überprüft und welche die Endbenutzer des Produkts während des gesamten Designprozesses mit einbindet, um die Attraktivität neuer Ideen und möglicher Lösungen zu evaluieren. [IDEO 2005b] Abb. 6 zeigt diesen iterativen Designprozess, welcher als eine Sequenz von fünf unabhängigen Phasen zusammengefasst werden kann: *Understand*, *Observe*, *Visualize and Predict*, *Evaluate and Refine* und *Implementation*. Innerhalb des Prozesses werden eine Reihe von Techniken zum Visualisieren, Evaluieren und Verfeinern von Möglichkeiten für das Design und die Entwicklung angewendet. [IDEO 2005b]

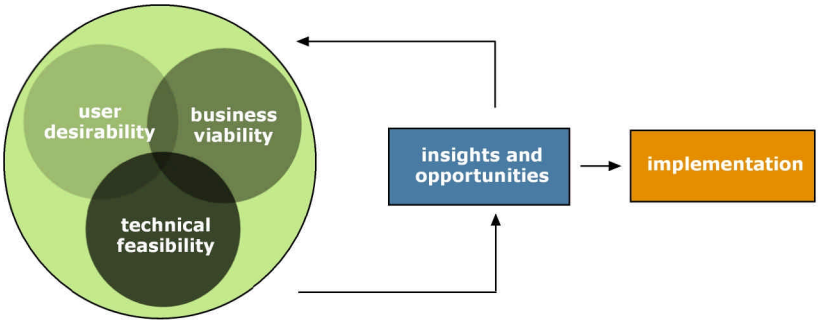

Abb. 6 Designprozess bei IDEO [vgl. IDEO 2005b]

Understand

Bevor der eigentliche Designprozess beginnt, muss der Designer sich zunächst Wissen über die Kontextsituation verschaffen. D.h. wie sieht beispielsweise die Wettbewerbssituation aus, welche vergleichbaren Konkurrenzprodukte gibt es auf dem Markt, welche potentiellen Marktsegmente gibt es, gibt es spezielle Interessen oder Technologien, die für den Bereich, in dem das Produkt erscheinen wird, relevant sein könnten, etc. Dies erfordert eine *State of the Art* Analyse, welche den Austausch mit Kollegen, das Interviewen relevanter Experten und Verbraucher am potentiellen Markt und das Einarbeiten in den Anwendungsbereich erfordert. Durch diesen Schritt erhalten die Designer Einblick in den Problembereich und erzeugen bereits eine kleine Menge von Schlüsselideen, die in das Produkt Design mit einfließen. [Winograd 2002]

Observe

Der eigentliche Designprozess beginnt mit der Fokussierung auf die potentiellen Benutzer und Kunden des Produkts. Diese Phase beinhaltet bei IDEO beispielsweise Methoden, wie *Focus Groups* oder *Interviews*. Dabei ist es wichtig einen weitläufigen Blick zu behalten. D.h. nicht nur Aktivitäten beobachtet, welche unmittelbar mit dem geplanten Entwurf in

Zusammenhang stehen, sondern ebenso zu beachten, was die Menschen in den in Beziehung stehenden Bereichen tun, was sie gewohnt sind zu tun und was für sie von Bedeutung ist. Der Benutzer und der Anwendungsbereich des Produkts, wie beispielsweise die Arbeits- oder Lebensumgebung des Benutzers, in dem das Produkt angewendet wird, wird beobachtet. Diese Phase der Beobachtung erfolgt unter Anleitung von *Human Factors*-Spezialisten. Diese Beobachtungen werden anschließend dokumentiert, um die folgende Arbeit daran auszurichten. Eine Technik, die hierbei eingesetzt wird, sind die sogenannten *Character Maps*. Diese stellen detaillierte Persönlichkeits- und Aktivitätsbeschreibungen für eine kleine Gruppe von angenommenen typischen Benutzern dar. Diese fiktiven Charaktere werden erzeugt, um eine breite Palette verschiedener Charakteristiken darzustellen, welche das Team bei potentiellen Benutzern des Produkts beobachtet habt. Das Visualisieren dieser Charaktere hilft den Designern sich vorzustellen, wie die geplanten Produkte in der Praxis von verschiedenen Menschen verwendet werden. [IDEO 2005b][Winograd 2002]

Visualize and Predict

In der dritten Phase tritt das Produkt, das entworfen werden soll, in den Fokus. Dabei kommen Techniken für *Brainstorming*, *Skizzierung*, *Prototyping*, *Simulationen* oder *analysierende Designs* zum Einsatz, die je nach Projekt eingesetzt und auch miteinander vermischt werden können. Aber auch dabei bleibt der Hauptaugenmerk stets auf dem Benutzer. Eine gute Möglichkeit die Details des Designs auszuarbeiten ist, detaillierte Szenarien oder Storybords einer Person zu entwickeln, die das neue Produkt benutzt. Szenarien sind fiktive Geschichten mit Charakteren, Events, Produkten und Umgebungen, die Produktideen und Themen in den Kontext einer realistischen Zukunft projizieren. Storyboards sind dagegen hochgradig visuell. Sie verwenden Fotografien oder Comicbuch-Sequenzen von Zeichnungen um die Interaktion zwischen Mensch und Produkt bildlich darzustellen. Bei beiden Techniken ist es die Gegenständlichkeit, die den Designer dabei unterstützt Aspekte des Design zu enthüllen, die sich ihm andernfalls nicht offenbart hätten. [Winograd 2002]

Evaluate and refine

Ist die grundsätzliche Struktur des Designs angelegt, werden die Details ausgearbeitet und Tests mit Benutzern durchgeführt um Feedback zu erhalten. Das Designteam führt wiederholte Zyklen der *Analyse*, *Beobachtung*, *Skizzierung* und des *Prototyping* aus. Insbesondere *Prototyping* wird als die Sprache der Innovation bei IDEO angesehen. Dabei können Prototypen sowohl von neuen Produkten, als auch von Services, einer Webseite oder auch von einem neuen Raum erstellt werden. Diese Prototypen können verschiedene Formen annehmen, je nach Projekt - von einfachen Modellen bis hin zu Modellen, welche praktisch schon fertiggestellte Produkte darstellen. *Prototyping* lässt den Designer schnell erkennen, welche Probleme im Zusammenhang mit dem Produkt auftauchen, so dass diese frühzeitig behoben werden können. In dieser Phase des Designprozesses sind die cross-funktionalen, interdisziplinären Aspekte äußerst relevant, da bei der Evaluation

jedes Aspekts des Designs unterschiedliche Evaluationskriterien berücksichtigt werden müssen. [IDEO 2005b][Winograd 2002]

Implement

Die Phase der Implementierung komplettiert den Gedankenzyklus und damit auch den Designprozess, indem es das Konzept in seine finale Form bringt. Alle Möglichkeiten wurden evaluiert und die Prototypen validiert und verfeinert. In dieser Phase fokussiert der Designer mehr auf die pragmatischen Aspekte der Erzeugung des Produkts. D.h. die Kosten, die Produzierbarkeit, die Lebensdauer, die Qualitätskontrolle, die Pflege und Instandhaltung des Produkts, u.s.w. Das Projektteam legt die letzten Design- und Konstruktionsdetails fest, wählt, wenn nötig Produktionspartner aus und arbeitet mit den Kunden zusammen um eine rechtzeitige und erfolgreiche Markteinführung zu gewährleisten.
Aufgrund der Beteiligung eines cross-funktionales Teams während des Designprozesses, kann sichergestellt werden, dass diese Aspekte in den früheren Schritten nicht ignoriert wurden, da sie Auswirkungen auf das gesamte Design des Produkts haben. Wurde der Designprozess akkurat durchgeführt, ist in dieser Phase des Prozesses mit keinen großen Überraschungen mehr zu rechnen, auch wenn es erforderlich sein kann, einige Modifikationen von Designdetails vorzunehmen. [IDEO 2005b][Winograd 2002]

Diese Methodologie kann auch in Variationen angewendet werden. D.h. es gibt eine Version mit drei, eine mit vier und eine mit sieben Phasen, wobei jede die Prozesselemente unterschiedlich neu arrangiert und verknüpft. Das Wesentliche an einer Designmethodologie ist nicht ein Vorgehensweise zu haben, der man strikt folgt, sondern eher eine Struktur zu haben, auf die man zurückgreifen kann – gewissermaßen als eine Gruppe von Referenzpunkten. Eine strukturierte Prozessmethode kann auch als eine Form von Selbstreflektion genutzt werden, um den aktuellen Designprozess sichtbar zu machen und um den Designprozess selbst kontinuierlich überarbeiten und verbessern zu können – als integraler Teil des Designs. [Winograd 2002]

4.2.2 Tech Box

Wie bereits erwähnt, werden die verschiedenen Phasen des Designprozesses von diversen Techniken unterstützt, die je nach Projekt eingesetzt und miteinander kombiniert werden können. Eine erfolgreiche Methode, welche von Designern bevorzugt für die Ideenfindung innerhalb des Designprozesses eingesetzt wird, ist der Ansatz der Box-of-Ideas. Hierbei geht es darum, sich durch Rückgriff auf Ideen, Techniken, Mechanismen, Materialien oder Produkten entweder aus dem projekteigenen oder aber aus projektfremden Anwendungsbereichen für das Design oder Redesign neuer oder bestehender Produkte inspirieren zu lassen. Die Kreativitätstechniken, die bei diesem Ansatz zum Einsatz kommen, sind das *Case-based Reasoning*, also das Lösen von neuen Problemen durch das Heranziehen von Wissen, entweder des eigenen oder das anderer Designer, welches von der Lösung

früherer ähnlicher Probleme erworben wurde und die *Cross-Fertilization*, das Transferieren von Ideen, Techniken oder Materialien von einem Anwendungsbereich in einen anderen. Dies beinhaltet eine Analyse von Lösungen aus Bereichen, die dem Projekteigenen fachfremd sind. Durch Adaption, Kombination oder Modifikation dieser Ideen auf neue Anwendungsbereiche lassen sich so neue kreative Ideen kreieren um so spannende Lösungen finden, welche - zunächst als unmöglich erscheinend - eventuell im Vorfeld als Lösungsweg ausgeschlossen worden wären.
IDEO verwendet den Ansatz der Box-of-Ideas in Form der Tech Box, ein Kreativitätstool das speziell zur Unterstützung des Brainstormings in der *Visualize and Predict* Phase des Designprozesses entwickelt wurde. Die Tech Box ist sowohl eine physische Objektsammlung, als auch eine digitale Datenbank in Form eines Online-Katalogs sowie das organisatorisches Gedächtnis der IDEO und gehört damit für IDEO, laut eigenen Aussagen, zu den hilfreichsten Tools innerhalb des Designprozesses. [IDEO 2005a]

Physische Objektsammlung

Die physische Ausprägung der Tech Box umfasst eine Sammlung von etwa 200 verschiedenartigen interessanten Objekten. Dies können beispielsweise pfiffige Materialien, eleganten Mechanismen, interessante Gewebestrukturen aber auch intelligente Spielzeuge sein. Andere Objekte in der Box sind z.B. metallummanteltes Holz, Materialien mit oder ohne Hohlräume, die dehnbar sind, sich krümmen und bei verschiedenen Temperaturen ihre Form oder Farbe ändern. [Preece et al. 2002] Aufbewahrt werden die Objekte in einem Schrank (vgl. Abb. 7), der viele verschiedene Schubfächer beinhaltet, welche wiederum in einzelne Abteilungen (vgl. Bento Box) unterteilt sind. Eingeteilt sind die verschiedenartigen Objekte in insgesamt fünf Kategorien, wobei die einzelnen Schubfächer nach Kategorien sortiert sind [IDEO 2005b]: *Amazing Materials*, *Cool Mechanisms*, *Interesting Manufacturing Processes*, *Electronic Technologies* und *Thermal and Optical*. [Preece et al. 2002] Jedes dieser Objekte oder auch Artefakte ist klar mit seinem Namen, seiner Kategorie und einer Nummer beschriftet. [IDEO 2005a]

Abb. 7 Tech Box [IDEO 2005a]

Online-Katalog

Detailliertere Informationen zu den einzelnen Objekten der Tech Box werden in einer zugehörigen digitalen Datenbank in Form eines Online-Katalog gesammelt und umfassend dokumentiert. Dieser Online-Katalog ist allen IDEO Mitarbeitern via Intranet intern zugänglich, wird aber auch als Teil des Innovationsservices interessierten Kunden gegen Bezahlung zur Verfügung gestellt. [Reiterer 2004][IDEO 2005a] Dabei verfügt jedes der Objekt über seine eigene Webseite, auf welcher genau seine Spezifikationen beschrieben sind, welche Hersteller es gibt, den Preis und wenn möglich, eine zusätzliche IDEO-spezifische Anekdote mit Angaben zu dem verantwortlichen Designer und Projektinformation. [IDEO 2005b] Die Webseite enthält die folgenden Elemente: ein Sucheingabefeld für eine Schlagwortsuche, eine Funktion für eine erweiterte Suche, die Möglichkeit die Liste der Objekte einzusehen, die Funktion den Kurator der Tech Box zu kontaktieren und die Möglichkeit die Webseite des nächstfolgenden bzw. des vorherigen Objekts anzeigen zu lassen. Zu dem jeweiligen Objekt gibt es die folgenden Informationen: Name des Objekts, Objektnummer, eine kleine Abbildung des Objekts mit Vergrößerungsmöglichkeit über einen Mausklick, *What is it?*, *What is it made of?*, *How does it work?, Why is it cool?, IDEO* (bei welchem Projekt von IDEO wurde das Objekt bereits eingesetzt und Ansprechpartner dafür). Darüber hinaus sind die folgende zusätzlichen Informationen verfügbar: *IDEO Contact, Category, Demo Instructions, Size of our Specimen(s), Appearance, Applications, Cost, Additional Info und Suppliers*. [Preece et al. 2002]

Verwendung

Die Tech Box dient IDEO als Kreativitätstool im Designprozess, um sich zu neuen Gestaltungsideen inspirieren zu lassen. Dazu kann jeder Mitarbeiter von IDEO die Fächer der Tech Box im Rahmen des Designprozesses, z.B. zur Unterstützung des *Brainstormings* durchstöbern und sich mit den Objekten beschäftigen und sie für ihr aktuelles Projekt einsetzen. [IDEO 2005a] Dabei können die Objekte von einem einzelnen Mitarbeiter allein zur Inspiration genutzt werden, können aber auch zu *Brainstorming Meetings* mit mehreren Personen mitgenommen werden, um innerhalb einer Gruppe darüber zu diskutieren und deren Anwendbarkeit für das konkrete Designproblem zu überlegen. Die Auswahl der Objekte bleibt den jeweiligen Personen überlassen. Dies können nützliche visuelle Requisiten oder mögliche Lösungen für ein bestimmtes Problem sein oder aber einfach nur Objekte, welche im gegebenen Kontext interessant erscheinen. Die notwendigen Hintergrundinformationen zu jedem einzelnen der Objekte sind über die Webseite des jeweilige Objekts abrufbar. Dadurch ist es möglich zu innovativen Lösungen zu gelangen, insbesondere durch Übertragung von Ideen, Techniken, Materialien oder auch Mechanismen aus einem – zunächst als unmöglich erscheinenden Anwendungsbereich – in einen anderen. [Reiterer 2004][Preece et al. 2002]

Die Tech Box ist eine wertvolle Quelle, welche Designer und Ingenieure dazu verwenden sich inspirieren zu lassen, um aus einem Muster auszubrechen oder einfach nur, um nicht das Rad nochmals neu zu erfinden. [IDEO 2005a]

Organisatorisches Gedächtnis

Da jeder IDEO Mitarbeiter dazu angehalten ist, konstant Ausschau zu halten nach möglichen neuen Kandidaten für die Tech Box, um diese dann zur Betrachtung einzureichen, fungiert die Tech Box auch als organisatorisches Gedächtnis der Firma. [Preece et al. 2002] Die Tech Box ermöglicht IDEO seine weitgefasste Reihe von Erfahrungen, welche aus den unterschiedlichen Projekten erworben wurde, zu archivieren und sie mit allen Studios in ihrem weltweiten Netzwerk zu teilen. [IDEO 2005a] Denn ganz bewusste Aufspüren passender Quellen der Inspiration ist, wie bereits erwähnt, ein wertvoller Schritt in jedem Designprozess.
Aufgrund des weitgefächerten Leistungsspektrums von IDEO und ihrer umfangreichen Designkompetenz verfügen sie über weitläufige Erfahrungen in vielen Industriebereichen, was sie befähigt, Techniken und Entdeckungen von dem einen in den anderen Bereich einzuführen, eine ideale Vorrausetzung für die Kreativitätstechnik der *Cross-Fertilization*, welche bei IDEO auch *Technology Brokering* oder *Knowledge Brokering* genannt wird. Dieses *Technology Brokering* oder *Knowledge Brokering* ist ein wichtiger Teil des Innovationsprozesses von IDEO. [IDEO 2005a]

Alle Hauptbüros von IDEO führen eine doppelte Tech Box, wobei jede von ihnen über ihren eigenen Kurator verfügt, welcher das Hinzufügen und Ergänzen neuer Objekte überwacht und verantwortlich ist für die Instandhaltung, Ergänzung, Pflege und Katalogisierung der Objekte und für das Fördern ihrer tagtäglichen Verwendung innerhalb des Büros. Jedes erdenkliche Objekt kann in die Tech Box eingebracht werden, sollte aber, sobald es allgemein bekannt ist, wieder entfernt und durch neue faszinierende Objekte ersetzt werden. [Preece et al. 2002] Die Idee einer solchen Box-of-Ideas oder auch Kreativbox lebt also von ihrer steten Weiterentwicklung, indem fortlaufend neue Elemente gesammelt und hinzugefügt werden, um so einen stetig wachsenden Pool an Ideen zu schaffen.

4.3 Pattern vs. Box-of-Ideas

Das nun folgende Kapitel fasst nochmals knapp die wesentlichen Vor- bzw. Nachteile der beiden Ansätze der Patterns und der Box-of-Ideas zusammen und wägt sie hinsichtlich ihrer geplanten Verwendung für die Unterstützung der Ideenfindungsphase innerhalb des Designprozesses einer Visualisierung, ab.

Pattern

Mit Hilfe von Patterns ist es möglich, geprüftes Designwissen, das auf vielen Jahren praktischer Erfahrung basieren kann, in einem einheitlich strukturiertem Format zu dokumentieren, zu archivieren und zu kommunizieren. Patterns reduzieren sich dabei auf das Wesentliche; auf die Kernidee, die jedem Entwurf innewohnt, und beschreiben daher generische Lösungen, die unzählige Male angewendet werden können, ohne sich zu wiederholen. Patterns werden auch als Instanzen von gutem Design bezeichnet. [vgl. Tidwell 1999] Patterns fokussieren explizit auf den Kontext und teilen dem Designer mit wann, wie und warum die Lösung angewendet werden kann. Patterns sind verordnend formuliert und helfen Designern neue Instanzen von gutem Design zu erzeugen. Patterns lassen sich in Patternsprachen organisieren, welche dem Designer als Anleitung zur Vorgehensweise beim Entwurf dienen. Patternsprachen lassen sich wiederum in Form von Graphen darstellen.

Die kurze prägnante Benennung der Patterns, die den Kern der Idee wiedergibt, sowie das Aussparen von Fachjargon in den Beschreibungen der Patterns unterstützt die Nutzung der Patterns durch disziplinunabhängige Personen. Dadurch kann ein allgemeines Vokabular, eine gemeinsame Sprache für den Designprozess aufgebaut werden, welche die Kommunikation der einzelnen Personen eines Designteams untereinander erleichtert und es ermöglicht, Entwürfe und Konzepte auf einer abstrakteren Ebene zu diskutieren. Die allgemeine Verständlichkeit der Patterns wird zusätzlich durch den Schreibstil der Patterns in Prosaform unterstützt. Die in den Patterns aufgeführten zahlreichen Beispielanwendungen erlauben Nicht-Experten die Idee, die hinter dem Pattern steckt, leichter zu verstehen. [Wilkins 2003] Die Beispiele dienen ebenso dazu, Benutzern potentielles Design zu zeigen. Somit wird auch die Integration von zukünftigen Benutzern, die in der Regel keine Experten für den Entwurf darstellen, in den Designprozess erleichtert. Diese Integration unterstützt den Ansatz des *User-Centered Design*, welches die Grundlage für ein erfolgreiches und qualitativ hochwertiges Design bietet. Damit lösen Patterns eines der größten Probleme, die beim interdisziplinären Design, wie es für Visualisierungen erforderlich ist, bestehen: die effektive Kommunikation. Eine effektive Kommunikation beschleunigt den Designprozess und kommt der Qualität des resultierenden Produkts zugute.

Patterns können sowohl von unerfahrenen als auch von erfahrenen Designer genutzt werden um Designvorschläge zu erarbeiten und Lösungen zu verfeinern. Denn Patterns sammeln die kollektive Erfahrung einer Vielzahl von Designern in einer Form, die es erlaubt unmittelbar angewendet zu werden. Patternsammlungen kommen im Allgemeinen sowohl einem einzelnen Designer zugute aber auch einer Community als Ganzes.

Neben den Vorteilen von Patterns gehören zu ihren beiden wesentlichen Nachteilen: sie hemmen die Kreativität eines Designers und sind aufgrund ihrer üblichen Präsentationsform als reine Textsammlung für den Designer nur unkomfortabel anzuwenden.

Patterns dokumentieren immer nur bereits bestehendes, bzw. auch verloren gegangenes Designwissen und bringen selbst keinen neuen theoretischen Boden hervor oder präsentieren keine neue innovative Techniken. Sie sind lediglich die Ergebnisse vorangegangener Argumente und Gedankengänge und ersetzen damit nicht die Notwendigkeit von Kreativität. [Alexander et al. 1995][Gamma et al. 2004] Ganz im Gegenteil, denn obwohl Patterns zu effektiven Designs führen, können sie die Kreativität eines Designers hemmen. Denn das Anbieten von leicht anzuwendenden und bekanntermaßen gut funktionierenden Lösungen kann dazu führen, dass der Designer keinen Anreiz hat, für sich selbst eine neuartige Lösung zu entwickeln.
Der zweite Nachteil von Patterns liegt in der mangelnden Komfortabilität der Anwendung von Patterns für den Designer im Zuge des Designentwurfs. So erfolgt die Dokumentation und Präsentation von Patterns im Allgemeinen in rein textueller Form - entweder analog, in Form eines Buches oder virtuell, in Form von Webseiten mit Hypertextfunktionen. Jedoch bieten selbst die derzeitigen Formen der virtuellen Versionen nur einen geringen Mehrwert gegenüber der analogen Form. Diese Art der Präsentation bietet dem Designer jedoch nur unkomfortable Möglichkeiten mit den Patterns zu interagieren um sie für den Entwurf eines Objekts zu nutzen. Damit kann der Entwurfsprozess für den Designer zu einem mühseligen Vorgang werden, der von ihm viel Geduld erfordert, wodurch die Verwendung von Patterns für den Entwurfsprozess im Allgemeinen eher gehemmt wird.

Box-of-Ideas

Die Box-of-Ideas – Ansatz, der in Form der Tech Box von IDEO vorgestellt wurde, hat das Ziel den Designer, durch Rückgriff auf Ideen, Techniken, Mechanismen, Materialien oder Produkte aus den unterschiedlichsten Anwendungsbereichen für das Design oder Redesign neuer oder bestehender Produkte zu inspirieren.

Dabei kombiniert die Tech Box die beiden Kreativitätstechniken des *Case-based Reasoning* und des *Cross-Fertilization*. Das *Case-based Reasoning* hat das Ziel, neue Probleme durch das Heranziehen von Wissen, entweder des eigenen oder das anderer Designer, das von der Lösung früherer ähnlicher Probleme erworben wurde, zu lösen. Beim *Cross-Fertilization* kommt es zu einer Befruchtung aus anderen Disziplinen und Anwendungsbereichen, indem Ideen oder Konzepte aus diesen anderen Bereichen auf das eigene Projekt übertragen werden. Durch Adaption, Kombination oder Modifikation dieser Ideen auf neue Anwendungsbereiche lassen sich so neue und innovative Ideen kreieren und sich so spannende Lösungen finden, welche - zunächst als unmöglich erscheinend - eventuell im Vorfeld als Lösungsweg ausgeschlossen worden wären.

Durch das Anbieten verschiedener Nutzungsmöglichkeiten der Tech Box, ist es dem Designer möglich, sich auf unterschiedliche Art und Weise von den verschiedenen Objekten der Tech Box inspirieren zu lassen. Die Tech Box unterstützt dabei eine zielgerichtete als auch ein ungerichtete Vorgehensweise des Designers und kann sowohl von

einzelnen Personen, aber auch innerhalb einer Gruppe, z.B. in Brainstorming-Meetings zur Inspiration genutzt verwendet werden. Dabei bietet der synchrone Webkatalog zum einen die nötigen Hintergrundinformationen zu den jeweiligen Objekten, dient aber auch der Zentrierung, Dokumentierung und Archivierung und Kommunikation von Designwissen und bildet somit das organisatorische Gedächtnis von IDEO. Dieses Wissen wird dem Designer in einem einheitlich formatierten und übersichtlich strukturiertem Format präsentiert, welches ihm ermöglicht, die für ihn relevanten Informationen schnell und bequem abzurufen. Die zusätzliche Angabe von Ansprechpartnern für das jeweilige Objekt bietet die Möglichkeit, weitere Informationen, welche nicht im Webkatalog vorhanden sind, mittels direkter Kontaktaufnahme via Email, zu erhalten.

Die Bestimmung eines verantwortlichen Kurators für die Tech Box sorgt dafür, dass deren Inhalte gepflegt, durch neue Objekte erweitert und innerhalb des Designprozesses auch genutzt werden. Dadurch werden die Mitarbeiter von IDEO auf der einen Seite sensibilisiert, für neue Ideen aus ihrem eigenen Umfeld offen zu sein und auf der anderen Seite wird so gewährleistet, dass die Tech Box stets auf dem neuesten Stand bleibt. Die Idee einer solchen Kreativbox lebt also von ihrer steten Weiterentwicklung, indem neue Elemente gesammelt und entsprechend hinzugefügt werden, um so einen stetig wachsenden Pool an Ideen zu schaffen.

Zusammenfassend kann man also sagen, dass die Verwendung von Patterns innerhalb des Designprozesses zu nachweislich qualitativ hochwertigen Entwürfen führt, welche jedoch immer nur auf bereits existierenden Lösungsmöglichkeiten basieren. Dem entgegen ist der Ansatz der Box-of-Ideas gezielt darauf ausgerichtet, den Designer innerhalb des Entwurfs-prozesses zu kreativen und neuartigen Designlösungen zu führen, welche so in dieser Form noch nicht existieren. Somit würden sich die beiden Ansätze innerhalb des Designprozesses gewinnbringend ergänzen und deren kombinierte Anwendung innerhalb der Entwurfsphase eines Artfakts hätte einen synergetischen Effekt, welcher zu qualitativ hochwertigen sowie innovativen Designlösungen führt.

5 Visualization Box (VizBox)

Die vorangegangenen Kapitel haben die Vor- und Nachteile der beiden beschriebenen Ansätze für die kreative Ideenfindung innerhalb des Designprozesses - Patterns und Box-of-Ideas - aufgezeigt. Offensichtlich haben beide Ansätze ihre Berechtigung für das Design und genießen ein Recht auf Eigenständigkeit. Die Untersuchung hat auch gezeigt, dass durch die kombinierte Verwendung der beiden Ansätze, es möglich wäre diese synergetisch miteinander zu verknüpfen, wodurch für den Benutzer ein Mehrwert entsteht, den ein Ansatz für sich allein genommen nicht erbringen kann. Durch diese ergänzende Kombination der Ansätze und die zusätzliche Nutzung der Vorzüge von visuellen Darstellungen und den damit verbundenen Interaktionsmöglichkeiten in Form von Visualisierungen, ist es möglich ein Tool zu schaffen, das gewinnbringend zur Unterstützung der kreativen Ideenfindung in der Designphase eines Entwicklungsprozesses für Visualisierungen eingesetzt werden kann. Dieses Kreativitätstool würde die beiden Ansätze der Patterns und der Box-of-Ideas synergetisch miteinander verknüpfen, so dass sowohl qualitativ hochwertige als auch innovative Designlösungen entwickelt werden können und diese gleichzeitig in ein Visualisierungssystem einbetten, welches dem Designer die Möglichkeit gibt, diese auf bequeme und Art und Weise innerhalb des Designprozesses zu nutzen. Damit stellt dieses Kreativitätstool selbst ein Visualisierungssystem dar, welches wiederum für das Design von Visualisierungen genutzt werden kann. Damit basiert dieses Tool auf einem interdisziplinären Ansatz, welcher von den traditionellen Designbereichen - der Architektur und dem Produkt Design - abgeleitet und inspiriert wurde und präsentiert diesen mittels aktueller und komfortabler Visualisierungstechniken. Basierend auf diesen Erkenntnissen, die aus vorangegangenen Kapiteln erworben wurden, wurde der Ansatz für die Visualization Box, kurz VizBox, entwickelt.

Die VizBox stellt eine webbasierte Bibliothek dar, welche eine Sammlung von Visualization Patterns und detaillierten Beschreibungen mit zugehörigen multimedialen Elementen zu Visualisierungen enthält und mit Hilfe von Visualisierungen und deren Interaktionsmöglichkeiten exploriert werden kann. Ziel der VizBox ist es, den Visualization Designer innerhalb des Entwicklungsprozesses einer Visualisierung, bzw. eines Visualisierungssystems – und dabei speziell die kreative Ideenfindung in der Designphase – mit einem Kreativitätstool zu unterstützen, welches das Erzeugen alternativer Lösungsideen fördert und zu qualitativ hochwertigen und innovativen Designlösungen führt.
Die VizBox vereint dabei zwei Ansätze, die aus traditionellen Designdisziplinen entstammen – der Ansatz der Box-of-Ideas, welcher primär aus dem Produkt Design Bereich stammt und der Patternansatz, der seine Ursprünge im Architekturbereich hat. Die Kernideen, die diesen beiden Ansätzen zugrunde liegen und in Kapitel 4.2 nochmals kurz zusammengefasst wurden, werden in dem interdisziplinären Ansatz der VizBox auf den Bereich der Visualisierung von Daten transferiert. Daher beinhaltet die VizBox sowohl

visualisierungsspezifische Patterns, die so genannten Visualization Patterns, welche auf abstrakter Ebene wiederkehrende geprüfte Lösungen für das Design von Visualisierungen bieten als auch einen breitgefächerten Ideenpool existenter Visualisierungsmöglichkeiten aus den unterschiedlichsten Anwendungsbereichen, die sowohl als Beispiellösungen für die Visualization Patterns dienen als auch zur Inspiration für neue Visualisierungsideen. Die VizBox nutzt damit sowohl die Vorteile von Patterns als auch die Vorteile des Box-of-Ideas Ansatzes, kombiniert diese beiden mit synergetischem Effekt für den Visualization Designer, bettet diese in ein Visualisierungssystem ein und macht sich somit auch die Vorteile interaktiver visueller Darstellungen zu Nutze.

Die nachfolgenden Kapitel zeigen, in welcher Phase die VizBox im Designprozess einer Visualisierung eingesetzt werden kann, wie die Strukturierung der einzelnen Komponenten – Visualization Patterns und Visualisierungsbeschreibungen - der VizBox aufgebaut ist und wie sie zusammenhängen, und wie eine mögliche Umsetzung der VizBox als Visualisierungssystem aussehen könnte, in Form von möglichen Designvorschlägen. Abschließend werden nochmals kurz die Vorteile der VizBox gegenüber den Ansätzen der Pattern und der Box-of-Ideas als alleinstehende Formen der Kreativitätsförderung innerhalb des Designprozesses aufgezeigt.

5.1 Einsatz der VizBox im Designprozess

Wie bereits in Kapitel 3.1.4 erwähnt, besteht der Designprozess einer Visualisierung laut der visualisierungsspezifischen Methodologie von Barry Wilkins aus insgesamt fünf Phasen, wobei die erste Phase des Modells als visualisierungsspezifisch und die folgenden vier Phasen als generisch für das Software Design gelten. Die Unterscheidung der spezifischen Methodologien der jeweiligen Disziplinen findet über die unterschiedlichen Wertesysteme und Techniken statt, welche für den Designprozess einer Software eingesetzt werden. Die VizBox stellt hierbei eine Technik dar, welche in der Designphase der Entwicklung einer Visualisierung oder eines Visualisierungssystems zur kreativen Ideenfindung und zum Erzeugen alternativer Designvorschläge unterstützend eingesetzt werden kann (vgl. Abb. 8). Damit stellt die VizBox eine rein visualisierungsspezifische Technik dar.

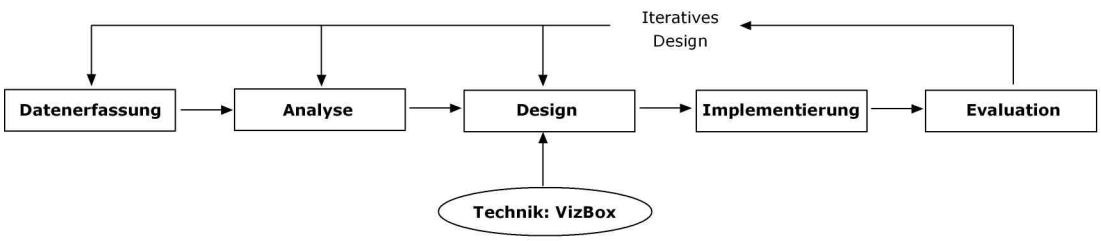

Abb. 8 Die Technik der VizBox im Designprozess

5.2 Gliederungskonzept der VizBox

Der folgende Abschnitt definiert das Gliederungskonzept der VizBox basierend auf den Ergebnissen der vorangegangenen Kapitel. Dieses Konzept umfasst den konzeptionellen Aufbau der VizBox, also aus welchen Komponenten die VizBox besteht, wie sie zueinander in Beziehung stehen und wie sich die Strukturierung der einzelnen Komponenten definiert.

5.2.1 Komponenten der VizBox

Das Gliederungskonzept der VizBox sieht die Integration der beiden Ansätze – Pattern und Box-of-Ideas - in Form zweier Sorten von Komponenten vor, aus denen die VizBox sich zusammensetzt: zum einen die Visualization Patterns, also visualisierungsspezifische Patterns und zum anderen Visualisierungen, bzw. Visualisierungsbeschreibungen, welche mit diesen Visualization Patterns mittels Referenzen verknüpft sind und einerseits als Beispielanwendungen für die Visualization Patterns dienen und andererseits den Box-of-Ideas Ansatz unterstützen, indem sie einen umfangreichen Ideenpool an existenten Visualisierungen aus den unterschiedlichsten Anwendungsbereichen bieten, welcher dem Visualization Designer zur Verfügung gestellt wird.

Die Visualisierungsbeschreibung selbst wird durch die Aufzählung der in der Visualisierung vorkommenden Visualization Patterns, in Form von Referenzen, im Gegenzug komplementiert. Zusätzlich kann es neben diesen heterogenen Referenzarten auch homogene Arten geben. Dies bedeutet, sowohl Visualization Patterns als auch Visualisierungsbeschreibungen können sich auch untereinander referenzieren, z.B. in Form von verwandten Patterns oder weiteren Visualisierungen eines Systems. Abb. 9 zeigt diesen Aufbau schematisch[11]. Wie man in der Abbildung sieht, referenzieren sich die beiden Komponentenarten sowohl untereinander als auch gegenseitig, wobei diese stets Referenzen bidirektional angelegt sind.

[11] *Anmerkung*: Zugunsten der Übersichtlichkeit wurde in der Abbildung auf die Darstellung aller vorhandenen Referenzen verzichtet.

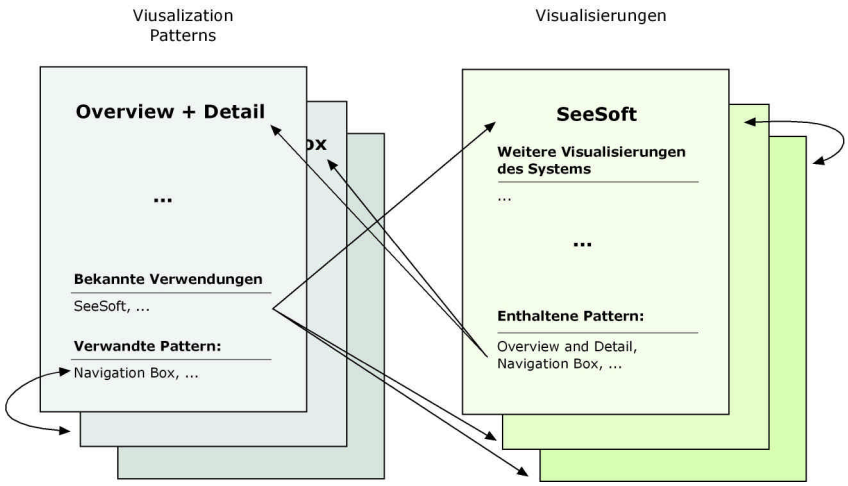

Abb. 9 Komponenten der VizBox

5.2.2 Strukturierung der Komponenten

Der folgende Abschnitt definiert, wie die Strukturierung der einzelnen Komponenten der VizBox – die Visualization Patterns und die Visualisierungen, bzw. Visualisierungsbeschreibungen – im Detail aussehen. Dabei wird zunächst die Formatvorlage für die Visualization Patterns, welche die VizBox enthält, präsentiert und anschließend die Formatvorlage für die Präsentation der Visualisierungen, bzw. Visualisierungsbeschreibungen erläutert.

5.2.2.1 Visualization Pattern

Visualization Patterns sind visualisierungsspezifische Patterns. Wie bereits erwähnt, hat sich mit der Entwicklung von Visualization Patterns bereits Barry Wilkins in seiner Dissertation *MELD: A Pattern Supported Methodology für Visualisation Design*. beschäftigt. Nachfolgend wird nun ein Strukturierungsvorschlag für Visualization Patterns gegeben, der von dem Barry Wilkins teilweise abweicht. Diese Abänderung des Formats für die Visualization Patterns basiert zum einen auf den Erkenntnissen, die während der Beschäftigung mit den diversen Patternsammlungen (vgl. Kapitel 4.1) und der Tech Box von IDEO (vgl. Kapitel 4.2) gewonnen wurden und zum anderen auf der Absicht die Visualization Patterns in das Visualisierungssystem der VizBox mit einzubetten.

5.2.2.1.1 Format des Visualization Pattern

Entsprechend der prinzipiellen Definition für das Format eines Pattern, sieht die Strukturierung für jedes einzelne der Visualization Pattern identisch aus. Der folgende Formatvorschlag wurde aufgrund der in Kapitel 4.1 betrachteten Patternsammlungen, hinsichtlich ihrer Formate für die Patterns selbst als auch hinsichtlich der Klassifizierung der Patterns in Untergruppen, erarbeitet. Zusätzlich unterliegt der Vorschlag auch dem Einfluss der Erkenntnisse aus der Betrachtung der Tech Box von IDEO (vgl. Kapitel 4.2). Von den betrachteten Patternsammlungen wurde insbesondere die Patternsammlung von Christopher Alexander beachtet, da diese die Basis für alle anderen Patternsammlungen legt und sich viele Definitionen für die einzelnen Elemente der Patterns daraus ableiten. Wie auch alle anderen Patterns bestehen die Visualization Patterns der VizBox prinzipiell aus den drei Teilen: Kontext, Problem und Lösung. Dabei beschreibt der Kontext das immer wieder auftauchende Problem in unserer Umwelt, der Problemabschnitt erörtert das Problem und der Lösungsabschnitt beschreibt schließlich eine mögliche Lösung des Problems und zwar so generisch, dass deren Anwendung beliebig oft vollzogen werden kann, ohne sich zu wiederholen. Die vorgeschlagene Formatvorlage für die Visualization Patterns der VizBox wird zunächst überblickartig dargestellt und anschließend jedes Element des Pattern nochmals detaillierter erläutert.

EINFÜHRENDER TEIL	Name
	Ranking
	Auch bekannt als
	Klassifizierung
	Autor
	Kontext
	***
PROBLEMABSCHNITT	Problemstellungnahme
	Problembeschreibung
	Lösung
	Skizze
	***
ABSCHLIEßENDER TEIL	Bekannte Verwendungen
	Verwandte Pattern
	Kurator

Nachfolgend werden die einzelnen Elemente des Visualization Patterns nochmals detaillierter erläutert. Dabei werden an dieser Stelle zum Teil nur noch mal kurz die wesentlichen Merkmale des jeweiligen Patternelement genannt, da diese größtenteils bereits in Kapitel 4.1.1 detailliert beschrieben wurden.

EINFÜHRENDER TEIL

- **Name**
  Der einführende Teil des Visualization Pattern beginnt mit dem Namen, einer der essentiellen Elemente des Visualization Pattern. Dieser sollte kurz, d.h. zwischen zwei bis maximal vier Worten lang, einprägsam und aussagekräftig sein und die Kernidee des Pattern beschreiben. [Borchers 2001] Dadurch kann die Idee schneller kommuniziert und innerhalb des Designprozesses darauf Bezug genommen werden und ist leichter erinnerbar. Zudem sollte er keine fachspezifischen Worte enthalten um möglichst allgemein verständlich zu bleiben. Namen können auch aus Analogien entstehen, dabei muss aber darauf geachtet werden, dass diese auch allgemeinverständlich sind. Ziel ist es so ein aussagekräftiges und möglichst allgemein verständliches Vokabular für den Entwurf von Visualisierungen zu erzeugen, welches es ermöglicht auf einer höheren Abstraktionsebene zu entwerfen und innerhalb eines interdisziplinären Designteams oder der Design Community im Allgemeinen zur Kommunikation verwendet werden kann.

- **Ranking**
  Das Ranking für jedes Pattern gibt den Grad der Zuverlässigkeit der jeweiligen Lösung an, also inwieweit der Autor des Pattern selbst glaubt, dass das Pattern tatsächlich einer Invariante entspricht. Denn die Qualität der einzelnen Patterns ist nicht immer gleich. Wie bereits erwähnt, ist das Schreiben von Pattern keine triviale Aufgabe und bedarf daher einer steten Revision. Für das Ranking wird hier die Einteilung von null bis zwei Sternchen hinter dem Namen des Pattern nach Christopher Alexander vorgeschlagen (vgl. Kapitel 4.1.1). Dieses Ranking wird als sehr wichtig erachtet, da es dem Benutzer des Patterns mitteilt, wie vertrauensvoll er mit der dargebotenen Lösung umgehen kann, insbesondere da ein Pattern einer steten Überarbeitung ausgesetzt ist. Derzeit nutzen diese Form der Bewertungsmöglichkeit eines Pattern nur die wenigsten der betrachteten Patternsammlungen.

- **Auch bekannt als**
  Da die Entwicklung von Patterns an unterschiedlichen Stellen vorangetrieben wird, kann nicht ausgeschlossen werden, dass die Kernidee, die dieses Pattern adressiert schon von einem anderen Patternautor unter einem anderen Patternamen beschrieben wurde. Dieses kann auch nach der Erstellung des vorliegenden Patterns geschehen sein, daher bietet Abschnitt die Möglichkeit, diese anderen Namen für das Pattern, sofern sie

existieren, aufzuführen, idealer Weise in Kombination mit Referenzangaben zur jeweiligen Quelle bzw. Autor des Pattern.

- **Klassifizierung**
  Die Klassifizierung gibt an, in welche Hauptkategorie das Pattern fällt. Für diese Art der Klassifizierung von Patterns wurden die Kategorisierungsmöglichkeiten nach Barry Wilkins übernommen - Strukturpattern, Interaktionspattern und Kompositionspattern. Diese Adaptierung der Klassifizierung wird zum einen damit begründet, dass für eine eigen erzeugte Klassifizierungsmöglichkeit der Patterns erst eine hinreichende Anzahl von Visualization Patterns erwickelt sein müssen, auf deren Grundlage dann eine Einteilung der Patterns vorgenommen werden kann. Zum anderen erscheint die Klassifizierung nach Wilkins, welche über insgesamt drei Kategorien verfügt, für eine erste grobe Klassifizierung eingängig und gut nachvollziehbar. Es kann aber nicht ausgeschlossen werden, dass bei der Entwicklung eigener Visualization Patterns auch weitere bzw. andere Klassifizierungsmöglichkeiten entdeckt werden.

- **Autor**
  In diesem Abschnitt wird der Name des Autors des Patterns angegeben. Da davon ausgegangen wird, dass an der Erstellung der Patternsammlung mehr als eine Person beteiligt sein wird, wird es als sinnvoll erachtet, den Autor mitanzugeben, wie dies auch van Welie für die GUI Design Patterns der Amsterdam-Collection[12] vorgesehen hat. Damit wird auch der Ansatz, welcher von vielen Patternautoren vorgeschlagen wurde, Patternsammlungen innerhalb der Community erstellen, überarbeiten und überprüfen zu lassen, unterstützt. Dieser hat das Ziel so eine Patternsprache zu erzeugen, welche stets weiter verbessert und erweitert wird und über eine hohe Validität verfügt.

- **Kontext**
  Unter dem Begriff „Kontext" kommt es bei den betrachteten Patternsammlungen zu einigen Abweichungen hinsichtlich der Definition dessen, was in diesem Abschnitt des Patterns aufgeführt wird. Im ursprünglichen Sinne, wie er von Christopher Alexander verwendet wird, stellt dieser Abschnitt den Zusammenhang zu den größer skalierten Patterns in der Sammlung her, was bedeutet, dass der Beitrag dieses Pattern zur Vervollständigung bestimmter größerer Pattern umrissen wird. Dadurch wird das Pattern zu anderen Pattern, welche in der Hierarchie auf einem höheren Level liegen, verlinkt. Dies ist aber nur möglich, wenn die Patterns in einer Hierarchie geordnet sind. Dies ist aber hier derzeit noch nicht der Fall. Daher wird dieser Abschnitt, wie auch bei anderen Patternautoren, allgemeiner verstanden und mittels Fließtext in Prosaform wird die Situation, in der das Pattern auftaucht, beschrieben. So wird dieser Abschnitt vor-

[12] Amsterdam-Collection: GUI Design Patterns
URL: http://www.welie.com/patterns/gui/index.html

erst auch hier verstanden, da noch keine Hierarchisierung für die Visualization Patterns vorliegt.

PROBLEMABSCHNITT

- **Problemstellungnahme**
  Die Problemstellungnahme leitet den Problemabschnitt, den zentralen Teil des Pattern ein. Sie fasst in einem oder zwei prägnanten Sätzen das Wesen des Problems, welches das Pattern adressiert, zusammen. Diese Problemstellungnahme wird fett hervorgehoben, vergleichbar einer Schlagzeile.

- **Problembeschreibung**
  Die ausführliche Problembeschreibung erläutert den empirischen Hintergrund des Pattern, begründet seine Gültigkeit, zeigt die verschiedener Formen, die das Pattern annehmen kann, usw. Es erläutert das Problem unter Verwendung des Konzepts der konkurrierenden Randbedingungen, weshalb dieser Abschnitt des Patterns in anderen Patternsammlungen, wie z.B. bei Jan Borchers oder Barry Wilkins zum Teil auch „*Forces*" genannt wird. Ziel des Patterns ist es, diese konkurrierenden Randbedingungen zu klären oder sie optimal im gegebenen Kontext auszubalancieren. Diese treten für gewöhnlich in Paaren auf, welche sich gegenseitig widersprechen. Die Problembeschreibung diskutiert ebenso existierende Lösungen. Dieser Part des Pattern ist oftmals am schwierigsten zu schreiben. Hat man Schwierigkeiten exakt auszudrücken, welches Problem ein potentielles Pattern löst, zeigt, dass das Problem nicht konkret genug definiert wurde. [Borchers 2001]

- **Lösung**
  Die Lösung stellt die Essenz jedes Patterns dar und erfolgt wieder im Fettdruck. Sie bietet eine allgemeingültige, generische geprüfte Lösung zu dem vorliegenden Problem, indem sie die vorliegenden Randbedingungen optimal für den gegebenen Designkontext ausbalanciert. Sie ist gewissermaßen das Destillat aus all den beschriebenen Beispielen und generalisiert dieses zu einem konstruktiven Designvorschlag, welcher in verschiedenen Situationen angewendet werden kann, wann immer ein Problem, wie es im Kontext beschrieben ist, angetroffen wird. Dabei ist sie jedoch so allgemein formuliert, dass sie den jeweiligen Bedingungen des aktuellen Problems angepasst werden kann. In jedem Fall sollte die Lösung nicht mehr als ein paar kurze und prägnante Sätze umfassen, da dieser Abschnitt des Pattern oftmals gelesen wird, um herauszufinden was die Kernaussage des Pattern ist.

- **Skizze**
  Eine Skizze visualisiert die zentrale Idee der Lösung auf schematische Art und Weise, so dass sie einfacher zu erfassen ist und man sich leichter an sie erinnern kann. Die

wesentlichen Elemente der Skizze sind beschriftet, während unwichtige Details zugunsten der Prägnanz weggelassen werden. Während ein Bild oder ein Foto immer nur eine spezifische Anwendung der Lösung präsentieren kann, muss die Skizze das für alle geltende Prinzip der Lösung verdeutlichen. Da jedes Pattern eine räumliche Anordnung beschreiben soll um ein bestimmtes Problem zu lösen, muss das Skizzieren eines Pattern immer möglich sein. Da in Visualization Patterns jedoch auch temporale Aspekte enthalten sind, kann es u.U. schwierig sein, diese festzuhalten. Jan Borchers schlägt zur Lösung dieses Problems daher die Verwendung einer Storyboard-Skizze vor, welche die temporären Designaspekten eines Patterns einfängt. [Borchers 20001]

ABSCHLIEßENDER TEIL

- **Bekannte Verwendungen**

  Der Abschließende Teil des Pattern beginnt mit der Aufzählung von bekannten Verwendungen, in welchen das beschriebene Pattern auftaucht. Diese dienen als praktische Beispiele für das Pattern und referenzieren die jeweiligen Visualisierungen, bzw. Visualisierungsbeschreibungen, die dieses Pattern beinhalten. Und zeigen, wo das vorliegende Problem vorkommen kann, wie es im einzelnen Fall gelöst wurde und wie die verschiedenen Randbedingungen ausbalanciert wurden. Die Aufzählung der bekannten Verwendungen sollte in Form des Namens der Visualisierung und einem zugehörigen Screenshots in Form eines *Thumbnails* erfolgen. Diese Bildbeispiele übernehmen auch die Funktion des sensibilisierenden Beispiels, welches im Kopfteil von Christopher Alexanders Pattern vorkommt. Dabei ist es wichtig, dass möglichst viele Beispielanwendungen referenziert werden. Dies dient zum einen als Nachweis der Validität des Patterns und zum anderen unterstützt es den Box-of-Ideas Ansatz.

- **Verwandte Pattern**

  Der Abschnitt der verwandten Pattern stellt den Bezug zu anderen Pattern her. Dies bedeutet, dass verwandte Patterns referenziert werden, welche mit dem aktuell beschriebenen häufig gemeinsam auftauchen. Laut Christopher Alexander und weiteren Pattern-Forschern, wie beispielsweise Jenifer Tidwell, ist die Qualität eines Entwurfs umso höher je mehr Patterns darin enthalten sind.

- **Kurator**

  Die Angabe eines Kurators und dessen Kontaktmöglichkeit in Form einer Referenz ermöglicht anderen Forschern und Anwendern Feedback zum aktuellen Stand des Pattern zu geben. Dadurch wird es möglich, bestehende Patterns stets durch die Community, wie von vielen Autoren von Patterns gefordert, zu überarbeiten und verbessern zu lassen (s.o.). Die Ernennung einer für die „Pflege" verantwortlichen Person entstammt dem Tech Box Ansatz, bzw. von OLIVE (vgl. „*Page Maintainers*", Kapitel 5.3).

Die drei Teile *Einführender Teil*, *Problemabschnitt* und *Abschließender Teil* werden jeweils optisch von einander getrennt dargestellt, wie dies auch bei den Architekturpatterns mit Hilfe der Sternchen der Fall ist.

Wie bereits oben erwähnt, erfolgt die Klassifizierung der Visualization Patterns nach Barry Wilkins. Grundsätzlich verfügen alle betrachteten Patternsammlungen oder Patternsprachen über keine allzu tiefe Hierarchisierungsebenen, bzw. Anzahl an Klassifizierungskategorien. Selbst Christopher Alexander strukturiert, wie oben gesehen, seine 253 Patterns lediglich in insgesamt drei Kategorien – Stadt oder Gemeinde (Städte), Gebäude und Konstruktion. Laut Barry Wilkins fallen Visualization Patterns in insgesamt drei Hauptkategorien, wobei es dabei auch zu Überschneidungen kommen kann [vgl. Wilkins 2004]: Strukturpatterns, Interaktionspatterns und Kompositionspatterns. Die Definition zu den einzelnen Patternkategorien findet sich in Kapitel 4.1.4.

5.2.2.2 Visualisierungen

Visualisierungen stellen die zweite Komponente dar, aus denen die VizBox besteht. Sie dienen zum einen als Beispielanwendungen der Patterns und zum anderen als Ideenpool existenter Lösungsmöglichkeiten aus den verschiedensten Anwendungsbereichen, durch die sich der Benutzer zu neuartigen Ideen für Entwurfsmöglichkeiten inspirieren lassen kann. Dieser Aspekt nimmt den Ansatz der Box-of-Ideas, bzw. Tech Box auf, bei welchem Kreativitätstechniken, wie das *Case-based Reasoning* und die *Cross-Fertilization* zum Einsatz kommen. Diese beinhalten die Inspiration durch Wissen, welches zum einen von der Lösung früherer ähnlicher Probleme als auch von Lösungsmöglichkeiten aus anderen Anwendungsbereichen stammt. Um diesen Ansatz zu gewährleisten, ist es erforderlich, dass eine möglichst breite Vielfalt an unterschiedlichen Visualisierungsmöglichkeiten in der VizBox enthalten sind. Um die Kreativitätstechnik der *Cross-Fertilization* bestmöglich zu unterstützen, ist es wichtig, Visualisierungen unterschiedlicher Anwendungsbereiche einzupflegen.
Dies bedeutet, dass die Inhalte der VizBox nicht auf Visualisierungen beschränkt sein dürfen, wie sie in der Regel im wissenschaftlichen Bereich entstehen, sondern auch bewusst Visualisierungsformen umfassen muss, wie sie in allen Bereichen des täglichen Lebens vorkommen. Dieser Ansatz wird von IDEO in der Art unterstützt, dass alle Mitarbeiter stets dazu angehalten sind, bewusst nach Objekten Ausschau zu halten, die in die Box-of-Ideas, die Tech Box, miteingebracht werden können. Wie bereits zu Beginn erwähnt, ist der Bereich der Visualisierung von Daten ein umfassendes Gebiet und eine Thematik, welche in vielen Bereichen des Lebens ihre Anwendung findet, gleich ob dies der Fahrplan der U-Bahn ist, die Statistik der Aktienwerte oder andere alltägliche Objekte, mit welchen man in Kontakt tritt. Mittlerweile nutzen auch viele Produkte des alltäglichen Lebens digitale Displays, welche ihre Informationen oftmals in visuellen Repräsentationen

darstellen. Ein gutes Beispiel hierfür sind Mobilfunkgeräte, welche in immer kürzer werdenden Produktzyklen zunehmend mehr Funktionen beinhalten und auf kleinstem Raum dem Benutzer zur Verfügung stellen. Auch Bereiche wie beispielsweise Webseiten, Softwaresysteme, Grafik- oder Bildbearbeitungsprogramme oder multimediale Ausbildungssoftware, Videospiele, digitale Enzyklopädien, Online-Dokumentationen, Terminals oder andere Geräte mit digitalen Interfaces, wie z.B. Palmtops, Handhelds oder MP3-Player, bieten die unterschiedlichsten Techniken für das Visualisieren von Daten. Diese Techniken müssen in die VizBox miteingepflegt werden um dem Visualization Designer einen möglichst breitgefächerten und umfassenden Ideen- und Wissenspool für die Thematik der Visualisierung von Daten zu bieten. Somit dient die Visualisierungsbeschreibung der VizBox der umfassenden Dokumentation von Visualisierungsmöglichkeiten in Form von strukturiertem Hintergrundwissen und Informationsmaterial zu den einzelnen Visualisierungen.

Nachfolgend wird die Strukturierung der Visualisierungsbeschreibung überblickartig aufgeführt, denn ähnlich der Visualization Patterns, gibt es auch für die Beschreibung der Visualisierungen ein festgelegtes Format, nach welchem diese dokumentiert werden. Dabei ist zu beachten, dass prinzipiell zwischen Visualisierungen und Visualisierungssystemen unterschieden wird. Ein Visualisierungssystem setzt sich aus verschiedenen Visualisierungen zusammen (s.u.), wobei die VizBox für jede Visualisierung eine eigene Visualisierungsbeschreibung vorsieht. Der Aufbau der Strukturierung erfolgt aus den Erkenntnissen, wie sie insbesondere aus Kapitel 3.1.4, aber auch aus Kapitel 4 und 5.3 gewonnen wurden. Dies umfasst die Faktoren, welche in der Analysephase des Designprozesses einer Visualisierung erhoben werden und zusätzliche Formatelemente, welche sich zum einen aus den zuvor genannten Kapitel ableiten lassen sowie auf eigenen Erfahrungen aus dem Bereich der Visualisierung von Daten basieren. Anschließend werden die einzelnen Formatabschnitte der Visualisierungsbeschreibung nochmals detaillierter beschrieben.

5.2.1.2.1 Format der Visualisierungsbeschreibung

EINFÜHRENDER TEIL	Name
	Autor(en)
	Entstehungsjahr
	Gehört zum Visualisierungssystem
	Weitere Visualisierungen des Systems
	Abbildung
	Enthaltene Visualization Pattern
	Anwendungsbereich(e)
	Kurzbeschreibung

DATEN	Datenquelle(n) Datentyp(en) Variablentyp(en) Quantität der Daten
VISUELLE STRUKTUR / LAYOUT	Dimensionalität Interaktionstechnik(en) Metapher(n) Element(e)
AUFGABEN / U-FAKTOREN / ZIELGRUPPE	Aufgab(en) Usability Faktor(en) Zielgruppe(n)
BESCHREIBUNG	Ausführliche Beschreibung
TECHNIK	Technische Anforderungen Implementierung / Quellcode
WEITERFÜHRENDE INFORMATIONEN	Weitere Abbildungen Literatur Video(s) / Animation(en) / Demoversion(en) Projekt(e) Produkt(e) Kurator

Der nachfolgende Teil beschreibt nochmals detailliert die einzelnen Strukturierungspunkte der Formatvorlage nach welcher die Visualisierungen beschrieben werden. Grundsätzlich gliedert sich die Visualisierungsbeschreibung in sieben Hauptabschnitte, den EINFÜHRENDER TEIL, die Abschnitte über DATEN, VISUELLE STRUKTUR / LAYOUT, AUFGABEN / USABILITY FAKTOREN / BENUTZER, BESCHREIBUNG, TECHNIK und WEITERFÜHRENDE INFORMATIONEN.

Der einführende Teil der Beschreibung hat das Ziel dem Visualization Designer einen ersten informativen Eindruck über die Visualisierung zu verschaffen, aufgrund dessen er sein weiteres Vorgehen planen kann, wie z.B. detaillierte Informationen zu der Visualisierung abzurufen oder zu einer anderen Visualisierung zu wechseln, bzw. zu einem in der Visualisierung enthaltenen Pattern zu wechseln. Die Informationen der folgenden drei Abschnitte DATEN, VISUELLE STRUKTUR / LAYOUT und AUFGABEN / USABILITY FAKTOREN / BENUTZER hängen mit den Hauptfaktoren aus Kapitel 3.1.4 zusammen, die für eine Visualisierung als relevant eingestuft und in der Analysephase des Design-prozesses eruiert werden. An dieser Stelle wurden auch zum Teil bereits Informationen zu

den einzelnen Faktoren genannt, welche in dem folgenden Abschnitt nicht nochmals detailliert erklärt werden.

Der Abschnitt der Visualisierungsbeschreibung zu den DATEN bietet dem Benutzer Informationen darüber aus welcher Quelle die Daten stammen, welche Datentypen der Visualisierung zugrunde liegen, welcher Variablentyp von der Visualisierung unterstützt bzw. für welche Menge von Daten die Visualisierung geeignet ist. Der Abschnitt VISUELLE STRUKTUR / LAYOUT macht Angaben zur Dimensionalität der Visualisierung, den enthaltenen Interaktionstechniken, ob und wenn ja, welche Metapher die Visualisierung nutzt und führt listenartig die in der Visualisierung verwendeten Elemente auf. Der nächste Abschnitt AUFGABEN / USABILITY FAKTOREN / BENUTZER führt die Aufgaben an, welche die Visualisierung unterstützt, die Usability Faktoren und für welche Art von Benutzer, also Zielgruppe die Visualisierung geeignet ist. Die nachfolgende AUSFÜHRLICHE BESCHREIBUNG der Visualisierung liefert dem Designer bzw. Benutzer der VizBox alle nötigen Hintergrundinformationen, die erforderlich sind um die Visualisierung gänzlich zu verstehen. Innerhalb des TECHNISCHEN ABSCHNITTS werden Angaben zu den technischen Anforderungen der Visualisierung gemacht und Hinweise zu Implementierung, inklusive Quellcode gegeben. Der abschließende Teil der Visualisierungsbeschreibung hat das Ziel den Benutzer mit WEITERFÜHRENDEN INFORMATIONEN zu der Visualisierung zu versorgen. Dies geschieht in Form von weiteren Abbildungen, die zu der Visualisierung existieren, Videos, Animationen und Demoversionen, weiterführender Literatur und Projekte und Produkte, welche zu dieser Visualisierung in Bezug stehen. Darüber hinaus wird auch ein Kurator bestimmt, welcher für die Betreuung der Visualisierungsbeschreibung verantwortlich ist und bei Fragen auch kontaktiert werden kann. Nachfolgend werden diese einzelnen Elemente der Visualisierungsbeschreibung nochmals detaillierter ausgeführt.

EINFÜHRENDER TEIL

- **Name**

  Der einführende Teil der Visualisierungsbeschreibung beginnt mit dem Namen unter dem die Visualisierung bekannt ist und gehört zu den essentiellen Komponenten der Beschreibung. Anhand des Namens der Visualisierung kann diese referenziert und während des Designprozesses innerhalb von Diskussionen des Designteams, aber auch innerhalb der Community zur Kommunikation genutzt werden. Daher sollte der Namen der Visualisierung eindeutig, d.h. innerhalb der VizBox nicht mehrfach vorkommen und prägnant sein, um leicht erinnerbar zu sein. Entsprechend sollte der Name der Visualisierung nicht zu lang gewählt werden und im Idealfall nur wenige Worte enthalten. Dies betrifft in erster Linie Visualisierungen, welche nicht im Rahmen von Forschungs- oder Entwicklungsarbeiten entstanden sind und daher einen beschreibenden Namen benötigen, der in der Regel erst noch vergeben werden muss.

Visualisierungen, welche im Rahmen von Forschungs- oder Entwicklungsarbeiten entwickelt wurden, wie z.B. *ThemeView* besitzen bereits einen eigenen Namen. Hierbei muss jedoch das Kriterium der Eindeutigkeit berücksichtigt werden, da Visualisierungssysteme zwar in der Regel einen eigenen Namen besitzen, die einzelnen verschiedenen Visualisierungen, die darin enthalten sind, jedoch manchmal über keinen eigenständigen Namen verfügen. Auf diesen Punkt wird weiter unten nochmals eingegangen (siehe „Gehört zum Visualisierungssystem").

- **Autor(en)**
  In diesem Abschnitt werden der Autor, bzw. die Autoren der Visualisierung aufgeführt, sofern sie bekannt sind. Zusätzlich können an dieser Stelle auch Referenzen zu personenbezogenen Homepages oder ähnlichem, welche weitere Informationen zu dem oder den Autoren bieten, eingefügt werden.

- **Entstehungsjahr**
  Das Entstehungsjahr der Visualisierung bezeichnet den zeitlichen Punkt, an dem die Visualisierung erstmalig publiziert wurde, bzw. in Fall von Visualisierungen, welche nicht im Rahmen von Forschungs- oder Entwicklungsarbeiten entstanden sind, erstmalig entdeckt wurde. Dies bietet die Möglichkeit eine zeitliche Abfolge über die Entstehungen von Visualisierungen zu erstellen.

- **Gehört zum Visualisierungssystem**
  Dieser Abschnitt bietet die Möglichkeit die Visualisierungen zu anderen Visualisierungen bzw. Visualisierungssystemen, zu welchen sie gehören in Bezug zu setzen. Entsprechend dem mehrdimensionalen Ansatz für die Beschreibung von Visualisierungen, den Thomas Mann in seiner Dissertation *Visualization of search results from the World Wide Web* vorstellt, wird prinzipiell zwischen Visualisierungen und Visualisierungssystemen unterschieden ([vgl. Mann 2002]: Komponenten und Systeme). Dies bedeutet, dass sich ein Visualisierungssystem aus einem oder mehreren Visualisierungen zusammensetzen kann. Ein Beispiel für ein Visualisierungssystem, welches aus mehreren verschiedenen Visualisierungen besteht, ist das visuelle Suchsystem *IN-SPIRE*[13], einer windowsbasierten Version von *SPIRE (Spatial Paradigm for Information Retrieval and Exploration)*, welches von der *Pacific Northwest National Laboratory (PNNL)*[14] entwickelt wurde und über mehrere Visualisierungen, wie z.B. der *Galaxies* Visualisierung und der *ThemeView* Visualisierung, verfügt. Diese Aufteilung eignet sich um die einzelnen Visualisierungen besser beschreiben zu können, da so jede Visualisierung einzeln mit ihren spezifischen Merkmalen beschrieben werden kann.

[13] IN-SPIRE™
URL: http://www.pnl.gov/infoviz/

[14] Pacific Northwest National Laboratory (PNNL)
URL: http://www.pnl.gov/

- **Weitere Visualisierungen des Visualisierungssystems**
  An dieser Stelle können Referenzen zu weiteren Visualisierungen, über die das zugehörige Visualisierungssystem verfügt, gesetzt werden. Dies stellt den Bezug her, mit welchen Visualisierungen die vorliegende Visualisierung in Kombination auftreten kann und entspricht dem Abschnitt „Verwandte Patterns" in der Beschreibung der Visualization Patterns.

- **Abbildung**
  Die Abbildung zeigt eine einprägsame und aussagekräftige Darstellung der Visualisierung, welche einen ersten Eindruck vermitteln und leicht erinnerbar sein soll. Zudem ist der Abbildung eine Beschriftung beigefügt, welche diese kurz beschreibt. In der Regel wird dies ein Screenshot oder Bild der Visualisierung sein. Dies kann aber auch, im Fall der Umsetzung der VizBox als Visualisierungssystem, eine kurze Animation der Visualisierung sein.

- **Enthaltene Visualization Pattern**
  In diesem Abschnitt werden die Visualization Patterns in Form von Referenzen - also die Namen der Patterns - aufgeführt, welche in dieser Visualisierung identifiziert wurden. Somit dient die Visualisierung auch als praktisches Beispiel für die referenzierten Patterns, in welchem dem Benutzer gezeigt wird, wo das vorliegende Problem vorkommt, wie es gelöst wurde und wie die einzelnen konkurrierenden Randbedingungen in dem vorliegenden Fall ausbalanciert wurden.

- **Anwendungsbereich(e)**
  Dieser Abschnitt der Visualisierungsbeschreibung führt die Anwendungsbereiche, in welchen die Visualisierung bereits eingesetzt wurde auf. Auch hier können zusätzlich Referenzen, wie im Abschnitt des Autors der Visualisierung, eingefügt werden.

- **Kurzbeschreibung**
  Die Kurzbeschreibung der Visualisierung beschreibt mit wenigen klaren Sätzen die Charakteristika der Visualisierung. Dies dient dazu, dem Benutzer einen ersten Überblick über die Visualisierung und ihre Funktionen zu vermitteln. Aufgrund dieses Eindrucks kann er beurteilen, inwieweit die vorliegende Visualisierung für ihn selbst bzw. ein bestimmtes Projekt relevant ist und ob er dazu weitergehende Informationen wünscht. Die Kurzbeschreibung eignet sich auch, um unerfahrenen Visualization Designern eine Möglichkeit zu bieten, sich schnell einen Überblick über mehrere Visualisierungsmöglichkeiten zu verschaffen. Man könnte diesen Abschnitt der Visualisierungsbeschreibung mit einem Abstract vergleichen.

DATEN

- **Datenquelle(n)**
  In diesem Abschnitt werden Angaben zum Bezugsraum der Daten, in dem die Daten gesammelt wurden, also über die Herkunft der Daten, gemacht. Diese Datenquellen können sehr verschiedenartig sein. Jedoch kann diese Information, wie bereits erwähnt, innerhalb des Designprozesses ein nützlicher Startpunkt für die Wahl der visuellen Struktur sein, da das physische Objekt von dem die Daten gesammelt wurden als Template für die visuelle Struktur der Visualisierung dienen kann. Beispiele hierfür wurden bereits in Kapitel 3.1.4 genannt.

- **Datentyp(en)**
  Dieser Abschnitt der Visualisierungsbeschreibung bestimmt den oder die Datentypen, die von der Visualisierung unterstützt werden. Der Datentyp hat Auswirkungen auf die Anzahl der Dimensionen, die von der visuellen Struktur gefordert werden können und definiert häufig ihre Beziehung zueinander und ist daher sehr essentiell für das Design von Visualisierungen. Wie bereits erwähnt, gibt es viele Ansätze für die Klassifizierung von Daten. Für diesen Abschnitt der Visualisierungsbeschreibung wird die Klassifizierung nach der *Task by Data Type Taxonomy of Information Visualizations*, kurz *TTT* von Ben Shneiderman vorgeschlagen, der zufolge insgesamt sieben verschiedene Datentypen unterschieden werden: eindimensionale (1D), zweidimensionale (2D), dreidimensionale (3D), temporale, multidimensionale (MultiD), Baumstrukturierte (Tree) und Netzwerk Daten. Dabei eignen sich manche Datentypen leichter für visuelle Strukturen als andere. So werden beispielsweise räumliche Daten häufig unter Verwendung von 2D oder 3D Koordinatensystemen gezeigt und hierarchische Daten oder Netzwerkdaten lassen sich gut in Linkdiagrammen darstellen. Multidimensionale und temporale Daten haben dagegen keine offensichtliche visuelle Form. Für deren Darstellung werden dann abstraktere Strukturen oder spezialisierte Techniken verwendet. Das Analysieren des Datentyps kann den Designer auf visuelle Strukturen und Techniken aufmerksam machen, welche in anderen Visualisierungen des gleichen Datentyps bereits erfolgreich verwendet wurden. [Wilkins 2003]

- **Variablentyp(en)**
  Dieser Abschnitt der Visualisierungsbeschreibung gibt an, welche Variablentypen (vgl. Kapitel 3.1.4) die Visualisierung unterstützt. Wie bereits erwähnt, unterscheidet man hierbei im Allgemeinen die folgenden drei Basistypen für Variablen: nominale, ordinale und quantitative Variablen, die zusätzlich transformiert, bzw. in diverse Subtypen, wie räumlichen oder geografischen quantitativen Daten oder quantitativer oder ordinaler Zeit unterschieden werden können. Die Variablen, welche der Visualisierung zugrunde liegen, sind für das Design einer Visualisierung äußerst relevant, da diese das Erscheinungsbild, d.h. die Wahl des Achsentyps der Visualisierung (elementare Achsentypen: unstrukturierte, nominale, ordinale oder quantitative Achse [vgl. Card et al. 1999])

beeinflussen. So erfordern beispielweise Scatterplots einen quantitativen Datentyp oder Bar Charts einen quantitativen in Kombination mit einem ordinalen, bzw. nominalen Datentyp.

- **Quantität der Daten**
  Der Abschnitt der Quantität der Daten macht Angaben darüber wie groß die Menge der Daten ist, welche die Visualisierung unterstützt. Dies bezieht sich in diesem Fall auf die Anzahl der Datenobjekte, welche durch die Visualisierung sinnvoll dargestellt werden können. Darüber hinaus ist die tatsächliche Datenmenge noch von weiteren Faktoren abhängig, wie der Anzahl der interessierenden Parameter, der pro Parameter erhobenen Werte und der Anzahl der Punkte, an denen die Parameter erhoben werden. [Schumann&Müller 2000] Prinzipiell gibt es bei den Darstellungsfähigkeiten von Visualisierungen und den damit verbundenen Interaktionstechniken große Unterschiede. Manche Visualisierungen und Interaktionstechniken eignen sich nur für die Darstellung geringer Datenmengen - also weniger Datenobjekte - wogegen andere erst bei großen Datenmengen sinnvoll eingesetzt werden können. Entsprechend signifikant ist der Einfluss dieser Angaben auf die Wahl der visuellen Struktur und der Interaktionstechniken beim Design einer Visualisierung.

VISUELLE STRUKTUR / LAYOUT

- **Dimensionalität**
  Dieser Abschnitt beschreibt, wie viele Dimensionen die Visualisierung in der Lage ist darzustellen. Die Dimensionalität der Visualisierung und der Datentyp hängen stark miteinander zusammen. Oftmals bestimmt der Datentyp die Anzahl der Dimensionen einer Visualisierung und deren Beziehung zueinander. Jedoch kann es auch vorkommen, dass die zugrundeliegenden Datentypen in einer anders dimensionierten Umgebung präsentiert werden, wie zum Beispiel multidimensionale oder dreidimensionale Daten in einer zweidimensionalen visuellen Struktur. Für das Design einer Visualisierung bzgl. der Dimensionalität gilt in der Regel: je höher die Datendimensionalität liegt, welche die Visualisierung unterstützen muss, desto schwieriger ist es, eine visuelle Struktur zu entwickeln, welche vom Benutzer schnell und exakt interpretiert werden kann. Das Analysieren der erforderlichen Datentypen und das Vergleichen der Dimensionen, welche von existierenden Visualisierungen unterstützt werden, kann den Visualization Designer zu geeigneten Techniken für das Bewältigen hochdimensionaler Systeme führen. [Wilkins 2003]

- **Interaktionstechnik(en)**
  In diesem Abschnitt werden die Interaktionstechniken aufgeführt, welche die Visualisierung unterstützt. Hierbei kann es zu Redundanzen innerhalb der Visualisierungsbeschreibung kommen, da Interaktionstechniken auch Visualization Patterns sein können (vgl. z.B. *Dynamic Queries*). Jedoch wird dieser Abschnitt als notwenig empfunden, da zu diesem Zeitpunkt nicht abgeschätzt werden kann, dass auch jede identifizierte Interaktionstechnik einem Visualization Pattern entspricht - auch wenn davon ausgegangen werden kann. Da aber Interaktionstechniken zu essentiell für Visualisierungen und deren Design sind, wird hier eine etwaige Redundanz innerhalb der Visualisierungsbeschreibung nicht als störend empfunden. Dieser Abschnitt betont die Relevanz von Interaktionstechniken und hebt sie daher nochmals gesondert für den Benutzer hervor. Die hier gewählte Klassifizierung der Interaktionstechniken entspricht, wie auch schon die Datentypen der *Task by Data Type Taxonomy of Information Visualizations* (siehe Kapitel 3.1.4) von Ben Shneiderman. Zu diesen Interaktionstechniken gehören: *Dynamic Queries, Direct Walk, Details-on-Demand, Attribute Walk, Brushing, Direct Manipulation, Dataflow, Pivot Table, Direct Selection, Camera Movement, MagicLens, Overview & Detail* und *Zoom* (vgl. Kapitel 2.2).

- **Metapher(n)**
  Ein ähnlicher Ansatz wie die Datenquelle stellt die Metapher dar. Dies ist eine Technik, welche Objekte und Konzepte, welche dem Benutzer aus anderen Bereichen bekannt sind, wie z.B. das Bewegen durch Landschaften oder Interagieren in Räumen, für das Design der Visualisierung verwendet. Thomas Mann beschreibt beispielsweise in seiner Dissertation einige Metaphern, die von Visualisierungen genutzt werden [vgl. Mann 2002]. Auch hier ist es das Ziel, wie bei der Datenquelle, durch das Unterstützen des mentalen Modells des Benutzers, die Erlernbarkeit und Interaktion mit der Visualisierung für ihn zu erleichtern.

- **Element(e)**
  In diesem Abschnitt werden die Elemente aufgezählt, welche die Visualisierung beinhaltet. Man könnte dies als eine Art der Verschlagwortung der Visualisierung bezeichnen. An dieser Stelle können beispielsweise die Grammatikelemente für Grafiken nach Leland Wilkinson aufgeführt werden, wie er sie in seinem Buch *The Grammar of Graphics* beschreibt [vgl. Wilkinson 1999]. Wilkinsons Grundidee war, durch Trennung von mathematischen und ästhetischen Regeln, die Grammatik von Grafiken zu identifizieren und somit ein allgemeingültiges System zu erzeugen. Dem Benutzer soll so die Möglichkeit gegeben werden auf Basis der „erlernten" Grammatik auch neuartig erscheinende Grafiken richtig klassifizieren zu können. So unterscheidet er beispielsweise diverse Graphtypen, wie *Punktgraph, Liniengraph, Bargraph, Histobargraph* oder *Konturgraph*. Aber auch andere verwendete Elemente der Visualisierung, wie z.B. *Scatterplot*, *Range Slider*, die Art der verwendeten *Glyphen*, *Radiobuttons*, *Comboboxen*, etc. können an dieser Stelle genannt werden.

AUFGABEN / USABILITY FAKTOREN / BENUTZER

- **Aufgab(en)**
  Dieser Abschnitt der Visualisierungsbeschreibung listet alle Aufgaben auf, welche die Visualisierung unterstützt. Die Aufgaben, die der Benutzer mit der Visualisierung erfüllen kann, sind äußerst relevant, da diese die Ziele des Benutzers darstellen, welche die Visualisierung bestmöglich unterstützen muss (vgl. Kapitel 2.3). Aufgaben lassen sich in Kategorien klassifizieren. Die für diesen Abschnitt vorgeschlagene Klassifizierungsmöglichkeit gehört zu der bereits mehrfach angeführten *Task by Data Type Taxonomy of Information Visualizations* von Ben Shneiderman. Diese umfasst insgesamt sieben Aufgaben, die sich prinzipiell auf einem hohen Level der Abstraktion befinden: *Overview, Zoom, Filter, Details-on-Demand, Relate, History* und *Extract*. Das Kennen der Aufgabenkategorien, welche eine Visualisierung unterstützen muss, kann dem Designer helfen, die Wahl der visuellen Struktur und insbesondere der Interaktionstechniken beim Design einer Visualisierung zu bestimmen.

- **Usability Faktor(en)**
  Dieser Abschnitt der Visualisierungsbeschreibung bewertet inwieweit die festgelegten Usability Faktoren von der Visualisierung erfüllt werden. Es gibt eine Reihe unterschiedlicher Usability Faktoren, welche für die Bewertung der Benutzerfreundlichkeit von Systemen verwendet werden. Die hier vorgeschlagenen Faktoren entsprechen der Norm für HCI und Usability, der ISO 9241, welche die Gestaltung von Bildschirmarbeit und –plätzen beschreibt und über die Anforderungen an Mensch-Computer-Schnittstellen Auskunft gibt. Laut dieser Norm gehören zu den Grundsätzen der Dialoggestaltung - auch Kriterien der Benutzerfreundlichkeit genannt – Aufgabenangemessenheit, Selbstbeschreibungsfähigkeit, Erwartungskonformität, Lernförderlichkeit, Steuerbarkeit, Fehlertoleranz und Individualisierbarkeit. [vgl. Reeps 2004] An dieser Stelle wird nun bewertet, inwieweit die beschriebene Visualisierung die vorgegebenen Usability Faktoren erfüllt. Das vorgeschlagene Bewertungsschema umfasst insgesamt vier Bewertungsschritte, rangierend von „die Visualisierung erfüllt den Usability Faktor überhaupt nicht" bis „die Visualisierung erfüllt den Usability Faktor vollständig". Die gerade Anzahl der Bewertungsschritte macht es erforderlich, dass Stellung dazu bezogen wird, ob die beschriebene Visualisierung prinzipiell benutzerfreundlich ist oder nicht. Da nicht davon ausgegangen werden kann, dass immer alle Informationen vorliegen, um jeden Faktor bewerten zu können, ist auch die Möglichkeit vorgesehen, den Usability Faktor mit „Keine Angabe" zu bewerten. Dieser Abschnitt der Visualisierungsbeschreibung dient dazu, eine kurze Einschätzung der Gebrauchstauglichkeit der Visualisierung zu erhalten, kann jedoch nicht die Evaluationsphase des Designprozesses einer Visualisierung ersetzten.

- **Benutzer**
  Dieser Abschnitt gibt an, für welche Art von Benutzer die Visualisierung geeignet ist. Dabei wird grob unterschieden in: Anfänger, Fortgeschrittener und Experte. Dies bezieht sich ganz allgemein auf den Grad der Fachkenntnis bezüglich dem Umgang mit Softwaresystemen. Auch dieser Abschnitt soll, wie schon der Abschnitt der Usability Faktoren, dazu dienen eine grobe Einschätzung der beschriebenen Visualisierung vornehmen zu können.

BESCHREIBUNG

- **Ausführliche Beschreibung**
  In der ausführlichen Beschreibung wird die Visualisierung detailliert in Prosaform beschrieben, so dass der Benutzer, bzw. Visualization Designer genau nachvollziehen kann, was die Visualisierung darstellt, wie die Interaktionstechniken genau funktionieren und welches Ziel die Visualisierung hat. Diese Beschreibung liefert sämtliche Hintergrundinformationen, die der Benutzer der VizBox benötigt, um die beschriebene Visualisierung gänzlich zu verstehen und beurteilen zu können.

TECHNIK

- **Technische Anforderungen**
  Innerhalb dieses Abschnitts werden die technischen Anforderungen, welche die Visualisierung erfordert - so weit sie bekannt sind - aufgeführt. Dazu zählen beispielsweise der Rechen- und Ressourcenaufwand, die verwendeten Hardware-, Software- und Peripheriekomponenten, die Systemanforderungen, etc.

- **Implementierung / Quellcode**
  In diesem Abschnitt wird die Möglichkeit geboten, Hinweise zur Implementierung der Visualisierung anzugeben, wobei dies durchaus auch Quellcode mit einschließen kann. Da dies voraussichtlich den dafür vorgesehen Raum für diesen Abschnitt überschreiten würde, kann dieses auch in Form von Referenzen geschehen.

WEITERFÜHRENDE INFORMATIONEN

- **Weitere Abbildungen**
  An dieser Stelle können weitere Abbildungen, welche zu dieser Visualisierung existieren referenziert werden. Dies hat das Ziel, dem Benutzer einen möglichst umfassenden und eindrücklichen visuellen Eindruck von der Visualisierung zu geben, weshalb hier auch möglichst viele Bildbeispiele referenziert werden sollten.

- **Video(s) / Animation(en) / Demoversion(en)**
  In diesem Abschnitt der Visualisierungsbeschreibung werden Videos, Animationen oder auch Demoversionen, welche zu der Visualisierung existieren referenziert. Dies soll, wie auch schon der Abschnitt zuvor, dazu dienen, dem Benutzer einen möglichst umfassenden visuellen Eindruck von der Visualisierung zu bieten und fördert darüber hinaus das Verständnis für die Funktionsweise der Visualisierung.

- **Literatur**
  Unter dem Abschnitt der Literatur sind Referenzen zu weiterführender Literatur, wie z.B. Bücher, Paper oder Webseiten zu finden. Daher erweist es sich als sinnvoll, diesen Abschnitt nochmals in die zuvor genannten Kategorien zu unterteilen. Dem Benutzer wird so die Möglichkeit geboten, sich über die vorliegende Visualisierungsbeschreibung hinaus zu informieren.

- **Projekt(e)**
  Dieser Abschnitt referenziert Projekte, welche mit der vorgestellten Visualisierung in Beziehung stehen. Dies hat das Ziel, den Eindruck, den der Benutzer von der Visualisierung erhalten hat, abzurunden und um ihm die Möglichkeit zu bieten sich über die Informationen der Visualisierungsbeschreibung hinaus noch weiter zu informieren.

- **Produkt(e)**
  Auch der letzte Abschnitt der Visualisierungsbeschreibung hat dasselbe Ziel wie der Abschnitt zuvor – dem Benutzer, bzw. dem Visualization Designer die Möglichkeit zu geben sich weiter zu informieren. Dies erfolgt an dieser Stelle in Form von Referenzen, die auf Produkte verweisen, welche die vorliegende Form der Visualisierung nutzen. Vorwiegend werden dies Produkte aus dem kommerziellen Bereich sein.

- **Kurator**
  Wie auch bei der Patternbeschreibung zuvor, wird für die Betreuung der Visualisierungsbeschreibung ein sogenannter Kurator bestimmt, welcher auch kontaktiert werden kann. Dies gibt anderen Forschern und Visualization Designern die Möglichkeit, Feedback zum aktuellen Stand der Visualisierungsbeschreibung abzugeben. Dadurch werden, wie bei den Visualization Patterns, bestehende Beschreibungen durch die Community fortlaufend überarbeitet, verbessert und deren Inhalte evaluiert.

5.3 Umsetzung als Visualisierungssystem

Die im Rahmen dieser Arbeit betrachteten Patternsammlungen, welche allgemein zu den bekanntesten gehören, liegen bisher entweder in rein analoger Textform - wie z.B. in Form eines Buches - vor oder sind virtuell, im Form von HTML-basierten Webseiten zugänglich, welche die Nutzung von Hypertexttechniken ermöglichen. Auch Sammlungen von Visualisierungen liegen in der Regel in analoger Textform, entweder in Form von Büchern oder Papern vor. Allerdings ist hier das Wissen selten strukturiert und übersichtlich aufbereitet und beinhaltet überwiegend Text in Prosaform. Vereinzelt gibt es auch digitale Visualisierungssammlungen. Dazu gehört beispielsweise *OLIVE*[15] *, On-line Library of Information Visualization Environments*, der Universität Maryland. Jedoch beschreibt diese Sammlungen primär nicht die Visualisierungen selbst, sondern bietet eine überblickartige Zusammenstellung in Form von Hyperlinks zu Informationen über die einzelnen Visualisierungen. Dies können beispielsweise Paper, Literatur bzw. diverse Projekte oder Produkte sein, welche mit diesen Visualisierungen in Zusammenhang stehen. OLIVE ist frei online verfügbar, wurde jedoch die letzten Jahre nicht mehr aktualisiert. Nachfolgend wird diese Sammlung nochmals kurz näher vorgestellt, um einen Eindruck zu gewinnen, in welcher Form diese ihren Inhalt präsentiert.

OLIVE - On-line Library of Information Visualization Environments

Die *On-line Library of Information Visualization Environments*, kurz OLIVE, entstand 1997 als Klassenprojekt innerhalb eines Kurses über *Information Visualization* an der Universität Maryland, welcher von Dr. Ben Shneiderman abgehalten wurde. Das Ergebnis dieses Kurses war eine Webseite, welche mittels Hyperlinks referenzierend diverse Visualisierungsprojekte und zugehörige Informationen aufführt. Die Kategorisierung der einzelnen Visualisierungen erfolgt anhand der Taxonomie von Chris North (Universität Maryland) *A Taxonomy of Information Visualization User-Interfaces*. Diese basiert auf Ben Shneidermans *Taxonomie für Information Visualization Environments*, welche bereits oben erwähnt wurde, erweitert diese jedoch um eine weitere Kategorie, genannt *Workspace*. Diese Kategorie adressiert das Problem der Organisation des Arbeitsplatzes, mit dem Ziel dem Benutzer möglichst viele der Informationen visuell zu zeigen. [OLIVE 2005] Auf der Webseite sind diese acht Kategorien über die gleichnamigen Navigationspunkte auf der linken Seite der Startseite erreichbar. Abb. 10 zeigt auf der linken Seite die Startseite der Webseite von *OLIVE*.

[15] OLIVE - On-line Library of Information Visualization Environments
URL: http://otal.umd.edu/Olive/

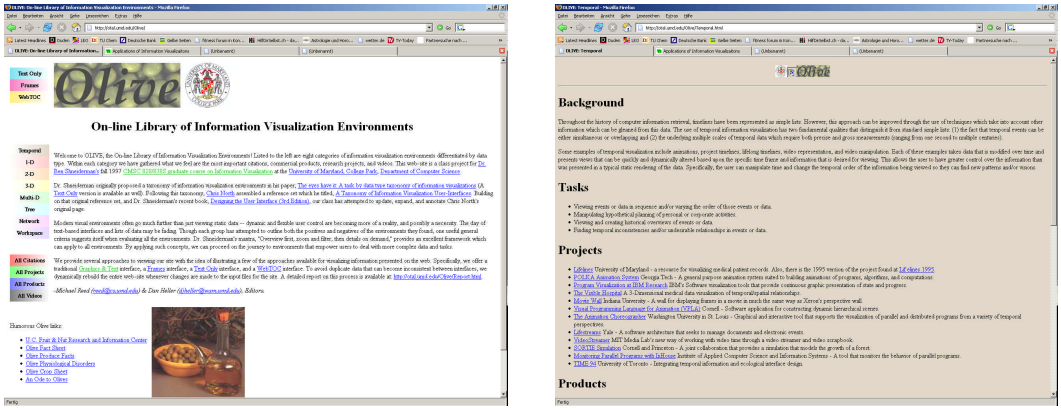

Abb. 10 Startseite (li.) und Unterseite *Temporal* (re.) [OLIVE 2005]

Auf den einzelnen Unterseiten gibt es unter *Background* einen kurzen Abschnitt mit Hintergrundinformationen zu dem jeweiligen Datentyp und unter *Tasks* werden Aufgaben aufgelistet, die mit Hilfe von Visualisierungen, welche in diese Kategorie fallen, vom Benutzer erfüllt werden können. Unter *Projects, Products, Citations* und *Videos* werden die, nach Meinung der Betreiber, jeweils wichtigsten Literaturstellen, kommerziellen Produkte, Forschungsprojekte und Videos zu dem jeweiligen Datentyp aufgelistet. Darüber hinaus werden auf jeder Unterseite die Verantwortlichen für die jeweilige Seite mit Emailadresse aufgeführt. Die aufgeführten Fakten, wie Projekte, Produkte, Literaturstellen und Videos sind zu den Originalquellen verlinkt. Nachteil hierbei ist, dass die Pflege der Links einen hohen Arbeitsaufwand erfordert und daher auch einige der Links nicht mehr aktuell sind. Die Sammlung enthält also selbst keine eigentlichen Visualisierungen, sondern ist streng genommen lediglich eine Linksammlung für Visualisierungen. Abb. 10 zeigt auf der rechten Seite beispielhaft eine Unterseite zu der Kategorie *Temporal*.
Über die Navigationspunkte *All Citations*, *All Projects*, *All Products* und *All Videos* auf der Startseite gelangt man zu Übersichtsseiten, auf welchen alle dazu gesammelten Fakten in alphabetischer Ordnung aufgelistet sind. Darüber hinaus verfügt die Webseite über keine weitere Strukturierung der Sammlung.

Die vorangegangenen Kapitel haben exemplarisch gezeigt, dass sowohl Patternsammlungen als auch Sammlungen zu Visualisierungen bislang entweder in rein analoger Textform oder vereinzelt auch in Form von Onlinesammlungen unter sporadischer Verwendung von Hypertextfunktionalitäten vorliegen. Dies bedeutet, dass der Präsentationsform solcher Sammlungen im Allgemeinen bislang wenig Aufmerksamkeit geschenkt wurde, weshalb sich deren Verwendung für den Benutzer oftmals als unattraktiv, unkomfortabel und zum Teil sogar als mühselig erweist.

Die geplante Umsetzung der VizBox als Visualisierungssystem wirkt genau diesem Manko entgegen. Die Verwendung von Visualisierungstechniken für die Umsetzung der VizBox bietet die Möglichkeit, die allgemeinen Vorzüge von visuellen Darstellungen und den damit verbundenen Interaktionsmöglichkeiten für den Benutzer zu nutzen. Aufgrund der viel-

fältigen Nutzungsmöglichkeiten der VizBox bietet sich für die Umsetzung ein mehrdimensionaler Ansatz an, was bedeutet, dass dem Benutzer mehrere unterschiedliche Visualisierungen und Interaktionsmöglichkeiten angeboten werden, um die Inhalte der VizBox explorativ zu erkunden. Damit stellt die VizBox selbst ein Visualisierungssystem dar, welches die zuvor beschriebenen Visualization Patterns und Visualisierungsbeschreibungen (vgl. Kapitel 5.2.2) als Inhalte hat. Bei der Umsetzung als solches sollten prinzipiell sowohl Ansätze unterstützt werden, welche dem Benutzer ermöglichen zielgerichtet nach Objekten zu suchen als auch Ansätze, welche ihm erlauben sich spielerisch und intuitiv mit den Inhalten der VizBox zu beschäftigen (*Browsing*-Paradigma), um somit beiden enthaltenen Ansätzen der VizBox – Pattern und Box-of-Ideas - gerecht zu werden.

Gerichtete Suche

Eine gerichtete Suchmöglichkeit stellt beispielsweise die Schlagwortsuche dar, welche dem Benutzer die Möglichkeit gibt, die Inhalte der VizBox nach Schlagworten zu durchsuchen, die für ihn relevant erscheinen (vgl. Online-Katalog der Tech Box). Dabei kann unterschieden werden zwischen einem einfachen Suchmodus und einem erweiterten Suchmodus, welcher beispielsweise mit Boleschen Operatoren arbeitet, bzw. eine Spezifizierung des Suchraums (z.B. Beschränkung auf eine Komponentenart oder bestimmte Formatelemente, wie Namen oder Lösung, etc.) zulässt. Damit könnte die erhaltene Treffermenge weiter spezifiziert werden.
Eine gerichtete Suche wäre aber auch durch Filterfunktionen möglich. Dabei werden alle Objekte der VizBox ausgeschlossen, welche nicht den definierten Suchkriterien entsprechen. Die so erhaltene Ergebnismenge würde somit nur Objekte enthalten, welche beispielsweise einen bestimmten Daten- oder Variablentyp unterstützen oder einen bestimmten Rankingtyp besitzen.
Eine weitere Möglichkeit einer gerichteten Suche, könnte über die Auswahl spezifischer Kategorien der einzelnen Komponenten stattfinden. Dabei wird eine bestimmte Kategorie, wie z.B. *Interaktionspattern* ausgewählt, ein geeignet erscheinendes Pattern davon ausgewählt und über die darin aufgeführten Referenzen wird der Benutzer zu anderen Objekten, welche mit diesem in Zusammenhang stehen weitergeleitet. Dieser Ansatz würde in Bezug auf die Visualization Patterns im Idealfall die Umsetzung der Patternsammlung als Patternsprache vorsehen.
Aber auch Sortiermöglichkeiten, wie es bei OLIVE der Fall ist (vgl. *Alle Projekte*, *Alle Videos*, etc.), sollten angeboten werden, um den Benutzer die Möglichkeit zu geben, Kategorienspezifisch zu suchen. Dies würde beispielsweise auch die Sortierung nach den Angaben, die in den einzelnen Elementabschnitten der Komponenten der VizBox gemacht wurden mit einschließen; wie beispielsweise die Sortierung der Visualisierungen nach Datentyp oder verwendeten Variablentypen oder die Sortierung der Visualization Patterns nach Rankingtyp oder Autor.

Ungerichtete Suche

Zum anderen muss die Umsetzung der VizBox aber auch den *Browsing*-Ansatz unterstützen, der eine ungerichtet, eher spielerische Herangehensweise des Benutzers, bzw. des Visualization Designers mit den Inhalten der VizBox ermöglicht. Das Durchstöbern des Inhalts der VizBox ermöglicht dem Benutzer eine nicht zielgerichtete Suche nach Objekten, welche für ihn interessant erscheinen und der Inspiration für neue Designvorschläge und insbesondere neuartige Lösungsmöglichkeiten dienen (vgl. physische Ausprägung der Tech Box von IDEO). Diese Art der Explorationstechniken der VizBox sollten sowohl über eine hohe Usability verfügen als auch einen unterhaltenden Charakter haben, um den Benutzer, bzw. Visualization Designer zu einer solchen Vorgehensweise anzuregen und so das Defizit zu mindern, das eine rein virtuelle Objektsammlung gegenüber einer physischen, wie die der Tech Box, hat. Bekannte Beispiele für Visualisierungstechniken, welche einen solchen Ansatz unterstützen sind beispielsweise die *Fisheye-Technik* oder die Technik der *Rapid Serial Visual Representation (RSVP)* von Oscar de Brujin und Robert Spence.

Sekundäres Ziel der VizBox ist, neben dem primären Ziel der Unterstützung der Designtätigkeit während des Entwicklungsprozesses einer Visualisierung, auch gleichzeitig die Form der interessanten Wissenspräsentation, welche der digitaler Enzyklopädien ähnelt (vgl. beispielsweise *Encarta Enzyklopädie*). Durch die große Menge an Information zu der Thematik von Visualisierungen und multimedialer Inhalte, welche die VizBox enthält, wird diese zu einer Art von Wissensdatenbank. Durch entsprechende Visualisierungsformen und die angenehme und informative Präsentation der Inhalte der VizBox, bietet sie dem Benutzer die Möglichkeit sich auf bequeme Art und Weise umfassend über die Thematik der Visualisierungen zu informieren.

Somit stellt die VizBox auf der einen Seite ein Kreativitätstool dar, welches der Unterstützung der Idenfindungsphase im Designprozess einer Visualisierung dient und auf der anderen Seite erfüllt sie die Funktion einer digitalen Enzyklopädie, welche ihre Wissensdatenbank dem Benutzer auf ansprechende und informative Weise präsentiert und mittels Interaktionstechniken exploriert werden kann.

Um die Ziele und Inhalte der VizBox als Visualisierungssystem umzusetzen gibt es eine Vielzahl von Lösungsmöglichkeiten. Wie beschrieben, ist der Entwurf einer Visualisierung und eines Visualisierungssystems im Besonderen, eine komplexe und anspruchsvolle Aufgabe, welche eines mehrstufigen Entwicklungsprozesses unter der Beteiligung eines interdisziplinären Designteams bedarf. Dies gilt selbstverständlich auch für den Entwurf der VizBox als Visualisierungssystem. Da eine solche Vorgehensweise jedoch nicht im Rahmen dieser Arbeit möglich ist, aber dennoch ein Eindruck vermittelt werden soll, wie eine visuelle Ausprägung der VizBox aussehen könnte - um auch deren Vorzüge als Visualisierungssystem ersichtlich werden zu lassen - werden an dieser Stelle exemplarisch

auch einige Designvorschläge vorgestellt. Die vorangegangenen Kapitel haben gezeigt aus welchen Komponenten, die VizBox sich zusammensetzt und wie die Struktur dieser Komponenten im Detail aussieht. Das nachfolgende Kapitel präsentiert einzelne mögliche Visualisierungsformen, wie sie in der VizBox vorkommen könnten. Zugunsten der Präsentation mehrerer Lösungsvorschläge wurde auf eine detailgetreue Ausarbeitung verzichtet, weshalb sie einen stark schematischen Charakter besitzen. Diese Designvorschläge dienen dazu, den Ansatz der VizBox als Visualisierungssystem zu verdeutlichen und um Anregungen zu geben, wie einige visuelle Ausprägungen der VizBox aussehen könnte.

5.4 Designvorschläge

5.4.1 Designvorschlag - Netzwerk

Markant für die Charakteristik des Zusammenspiels der beiden Komponenten der VizBox - Visualization Pattern und Visualisierungsbeschreibungen - sind die vielfältigen Referenzen untereinander, welche in einer virtuellen Ausprägung der VizBox als Visualisierungssystem, in Form von Hyperlinks dargestellt werden können. Aufgrund dieser Referenzen entsteht ein Netzwerk, welches sowohl aus Visualization Patterns als auch aus Visualisierungsbeschreibungen besteht. Da sich beide Komponentenarten auch untereinander selbst referenzieren, lassen sich aus dem gesamten Netzwerk auch Teilmengen bilden. Dies bedeutet, dass es neben einem heterogenen Netzwerk auch zwei homogene Netzwerke, welche nur Komponenten einer Sorte bestehen, gibt. Prinzipiell kann natürlich auch immer aus Teilmengen der Komponenten ein Netzwerk gebildet werden. Diese charakteristische Beziehung der Komponenten untereinander ermöglicht es, diese in einer Netzwerkdarstellung zu visualisieren, welche dem Benutzer die Möglichkeit gibt, sowohl überblickartig als auch im einzelnen zu erkennen, welche Objekte miteinander in Beziehung stehen. Abb. 11 zeigt eine solche Netzwerkvisualisierung der beiden Komponenten der VizBox schematisch.

Zu dieser einen Ausführung gibt es sicherlich eine Vielzahl von Variationsmöglichkeiten. Die Abbildung zeigt eine Variation der Netzwerkdarstellung, welche prinzipiell aus Visualization Patterns besteht, für das zuvorderst dargestellte Pattern jedoch zusätzlich noch die referenzierten Visualisierungen mit abbildet, welche das Pattern beinhalten. Diese positionieren sich automatisch gleichmäßig um das vorderste Objekt herum, um möglichst Verdeckungen zu verhindern. Kommt es dennoch zu Verdeckungen aufgrund mangelnden Darstellungsraums, könnte man zusätzlich die Technik der *Fisheye-View* verwenden, welche diese beispielsweise beim Überfahren mit dem Mauszeiger optisch entzerrt. Optional könnte man auch dem Benutzer die Möglichkeit geben via Range Sliders die Menge an dargestellten Beispielen selbst zu bestimmen. Die visuelle Angabe über die Sorte der Komponenten findet über eine farbliche Kodierung der einzelnen Objekte und deren

Querverbindungen statt. Gleichzeitig bietet ein Detailfenster unterhalb des Netzwerkes zusätzliche Informationen über das ausgewählte Objekt (siehe blauer Rahmen). Der Benutzer hat nun die Möglichkeit mit der Visualisierung zu interagieren. Zum einen kann er weiter durch das Netzwerk navigieren, in dem er ein anderes Pattern auswählt und damit in den Vordergrund der Darstellung rückt, zum anderen bietet ihm das Detailfenster beim Überfahren eines Objektes mit dem Mauszeiger Elemente aus den Beschreibungen der Komponenten, die er zuvor spezifiziert hat, an. Hiermit würde die VizBox dem Benutzer die Möglichkeit zur Individualisierung der Anwendung geben. Die hier spezifizierten Elemente sind neben dem Namen und einer Abbildung der Visualisierung: Autor(en), Anwendungsbereich(e), Datentyp(en) und Kurzbeschreibung. Dies bietet ihm die Möglichkeit schnell und bequem durch die Sammlung zu navigieren und gleichzeitig Detailinformationen zu den Objekten abzurufen.
Denkbar wäre beispielsweise auch eine Ausklappfunktion des Detailfensters, um die Menge an dargestellten Informationen zu erhöhen oder eine Bildlaufleiste, welche dem Benutzer beim Überfahren mit dem Mauszeiger zusätzliches Bildmaterial der Visualisierung bietet bzw. die Wahlmöglichkeit für den Benutzer zwischen einer dreidimensionalen Darstellung des Netzwerkes - wie hier gezeigt - oder einer zweidimensionalen zu wählen. Zu der definitiven Ausprägung der Visualisierung gibt es zahlreiche Variationsmöglichkeiten. Diese sind nur einige wenige davon.

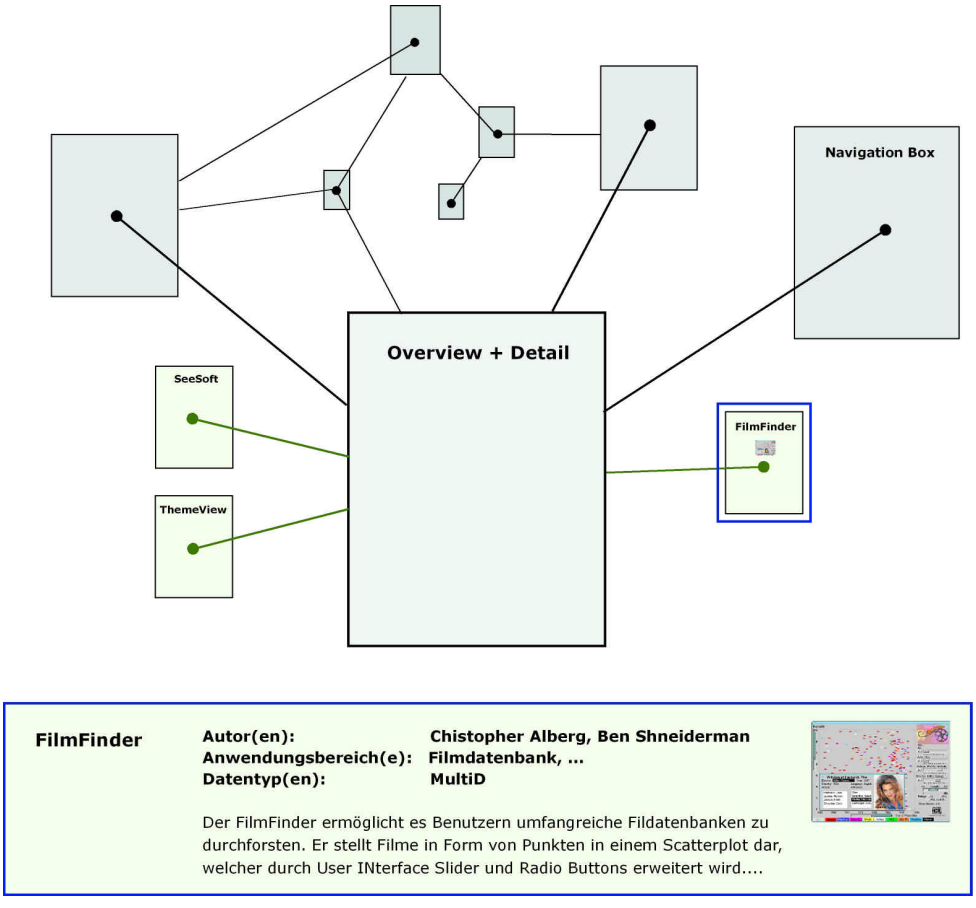

Abb. 11 Designvorschlag – Netzwerk

5.4.2 Designvorschlag - TableLens

Eine gute Möglichkeit listenartige Ergebnismengen übersichtlicher darzustellen und dem Benutzer gleichzeitig Einblick in bestimmte Details der einzelnen Objekte zu geben bietet die *Table Lens*, welche von Ramana Rao und Stuart Card entwickelt wurde. Durch die Verwendung der *Fisheye-Technik* ist es möglich, eine oder mehrere Zeilen der dargestellten Tabelle zu öffnen, um dem Benutzer weitere Informationen zu den einzelnen Objekten bieten zu können, ohne dabei aber die Darstellung verlassen zu müssen. Mittels des Mauszeigers fährt der Benutzer über die einzelnen Zeilen der Tabelle, welche sich dadurch automatisch öffnen und ihm so einen schnellen Einblick in die Details der Objekte bieten. Durch einen Mausklick lassen sich diese auch im geöffneten Stadium fixieren. Dieser Prozess kann auch stufenweise vollzogen werden, so dass es möglich ist, gleichzeitig mehrere Objekte in unterschiedlichen Granularitätsstufen darzustellen (vgl. *INVISIP*[16]). Der Benutzer hat die Möglichkeit für ihn interessante Objekte visuell zu markieren, um sie später leichter wiederfinden zu können. Die Belegung der Spalten kann vom Benutzer selbst spezifiziert werden (vgl. oben: Individualisierung des Systems).

In dieser Version (siehe Abb. 12) sind die Spalten mit Name, Abbildung, Autor(en), Kurzbeschreibung und Datentyp der Visualisierungsbeschreibung belegt. Für heterogene Ergebnismengen können nur Elemente der Komponenten gewählt werden, welche beide gemeinsam haben. Ein großer Vorteil dieser Visualisierung liegt darin, dass sie dem Benutzer ermöglich, die einzelnen Spalten auf- oder absteigend zu sortieren. Dadurch kann er auf bequeme Weise die erhaltene Treffermenge explorieren und mittels Funktionen, wie dem Markieren oder Verstecken einzelner Objekte, diese auch weiter verfeinern. Die eigentliche Suchanfrage kann dabei über eine Schlagwortsuche ausgeführt worden sein, wäre aber beispielweise auch in Kombination mit der Technik der *Dynamic Queries* denkbar, welche die direkte und dynamische Manipulation der Treffermenge mittels *User Interface Slidern*, *Radio Buttons* und *Check Boxes* ermöglicht. Bevorzugt wird die Technik der *TableLens* auch in Kombination mit einem Detailfenster oder anderen Visualisierungen verwendet, mit denen sie dann gekoppelt wird. Diese Möglichkeit wäre auch hier denkbar.

[16] INVISIP - Information Visualisation for Site Planning
URL: http://hci.uni-konstanz.de/index.php?a=research&b=projects&c=198153&lang=de

	Name	Abbildung	Autor(en)	Kurzbeschreibung	Datentyp
○	CircleSegments	...	...	...	...
○	ThemeView		James A.Wise, James J.Thomas, Kelly Pernnock	Die ThemeView des SPIRE Systems ist eine von der Galaxies (SPIRE) abgeleitete Visualisierung, in welcher die Dominanz der Themen auf einer zusätzlichen Achse – der Zeit – dargestellt wird. Diese ThemeView stellt die ...	1D
●	WebBook	...	...	...	...
●	SeeSoft	...	...	...	...
●	FilmFinder		Christopher Alberg, Ben Shneiderman	Der FilmFinder ermöglicht es Benutzern umfangreiche Fildatenbanken zu durchforsten. Er stellt Filme in Form von Punkten in einem Scatterplot dar, welcher durch User INterface Slider und Radio Buttons erweitert wird. Diese Punkte kodieren farbig zusätzlich die jeweiligen Filmkategorien, wie zum Beispiel „Action" oder „Sience Fiction". Mittels der Interaktionstechnik der Dynamic Queries, ist es dem Benutzer möglich, mittels direkter Manipulation eine Anfrage rasch zu spezifizieren. Durch Kopplung dieser Anfrage mit ...	MultiD
○	Galaxies	...	...	...	...
	...	...	...	...	...

Abb. 12 Designvorschlag - TableLens

5.4.3 Designvorschlag - RSVP

Eine sehr intuitive und bequeme Möglichkeit einen Datenbestand zu durchstöbern stellt die Technik der *Rapid Serial Visual Presentation (RSVP)* von Oscar de Brujin und Robert Spence dar. Diese entspricht der elektronischen Version des „Durchblätterns" eines Buches um seinen Inhalt einschätzen zu können. Diese Technik eignet sich sowohl für die Darstellung von viel Information auf kleinem Raum als auch um das *Browsing*-Paradigma zu unterstützen. Diese Information kann dabei aus Text- als auch aus Bildmaterial bestehen. Daher eignet sich diese Technik besonders gut als Visualisierungsmöglichkeit für die VizBox und könnte sicherlich an vielen Stellen eingesetzt werden. Abb. 13 zeigt eine dieser möglichen Anwendungen, bei welcher der Benutzer die Möglichkeit hat durch die Sammlung der Visualisierungsbeschreibungen zu navigieren. Dabei rotieren die einzelnen Objekte kreisförmig sobald der Benutzer mit dem Mauszeiger darüber fährt. Stößt er auf ein Objekt seines Interesses, kann er diese Rotation anhalten, bzw. verlangsamen, um die Details des Objektes zu betrachten. Somit hat er mit Hilfe des Mauszeigers die Möglichkeit die Rotationsgeschwindigkeit zu beeinflussen.

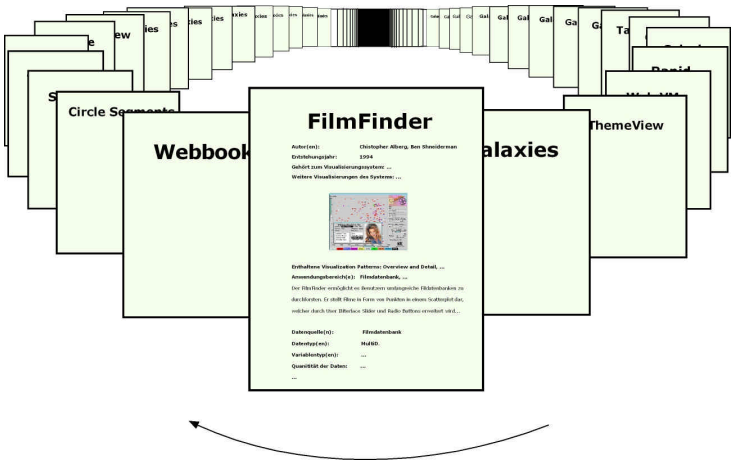

Abb. 13 Designvorschlag - RSVP

Variationen dazu wären z.B. das Navigieren durch Teile der Visualisierungsbeschreibungen, im Sinne von ID Karten, auf welchen nur die wichtigsten Aspekte der Visualisierung notiert sind, zugunsten einer größeren Darstellung der Inhalte. Diese Variation könnte beispielsweise auch wieder mit einem Detailfenster kombiniert werden, in welchem weiterführende Angaben zu der jeweiligen Visualisierung erscheinen, sobald diese in den Vordergrund rückt. Die gleiche Technik ist auch für das Durchstöbern der Visualization Patterns denkbar. Weiterhin sind auch Kombinationsmöglichkeiten denkbar, bei welchen beispielsweise ein bestimmtes Visualization Pattern betrachtet wird und die Visualisierungen, die diesem Visualization Patterns als Beispielanwendungen zugeordnet sind, mittels dieser Technik vom Benutzer, bzw. Visualization Designer exploriert werden können. Umgekehrt ist dies natürlich ebenso möglich.

Dieses Kapitel hat exemplarisch einen kleinen Einblick gegeben, wie Visualisierungstechniken dazu genutzt werden können, die Inhalte der VizBox zu explorieren. Außer diesen gibt es noch viele weitere Visualisierungsmöglichkeiten, wie z.B. eine temporal geordnete Darstellung der Visualisierungen auf einer Zeitleiste – entsprechend ihrem Entstehungsjahr – durch welche der Benutzer interaktiv navigieren kann oder die Nutzung der Technik des *Webbooks* von Stuart K. Card, Georg G. Robertson und William York, welches dem Benutzer unter Verwendung einer Technik, welche *Document Lens* genannt wird, es durch „Auseinanderziehen" der Seiten ermöglicht alle Seiten parallel zu betrachten. Diese vielfältigen Visualisierungsmöglichkeiten und damit auch alternativen Designlösungen müssten im Rahmen eines kompletten Designprozesses für die VizBox iterativ gesammelt, implementiert und evaluiert werden, um anschließend die bestmöglichen Designlösungen als Visualisierungssystem für die VizBox umzusetzen.

5.5 Die VizBox als webbasierte Lösung

Die Umsetzung der VizBox sieht eine Anwendung auf Basis von Web-Technologien vor. Dies hat den Vorteil, dass die Anwendung der VizBox somit über das Internet bzw., vergleichbar der Tech Box, über das Intranet verfügbar gemacht werden kann. Die Vorteile webbasierter Anwendungen liegen zum einen in der Einfachheit der Verteilung der Anwendung, da so aufwändige Installationsroutinen oder die Problematik älterer Programmversionen wegfallen, und zum anderen in der Zentralisierung der Anwendung auf dem Webserver und somit dem Benutzern über den Web-Browser zur Verfügung gestellt werden kann, womit auch Plattformunabhängigkeit erreicht wird. Ein zentraler Datenbestand erleichtert auch die Pflege der VizBox, da immer nur an einer Stelle Veränderungen des Datenbestandes vorgenommen werden und es somit dabei zu keinen Inkonsistenzen kommen kann.
Diese Vorteile bieten dem Benutzer die Möglichkeit, bequem und ortsungebunden auf die VizBox zuzugreifen und mit dieser zu interagieren. Dies unterstützt auch den Ansatz der VizBox für ihre Verwendung innerhalb des Designprozesses einer Visualisierung. Die

VizBox unterstützt den Visualization Designer innerhalb der kreativen Designphase des Prozesses bei der Generierung alternativer Designlösungen. In dieser bewusst ausgeführten Phase der Ideenfindung kann der Visualization Designer entweder eigenständig nach möglichen Lösungsvorschlägen suchen, aber auch gemeinsam mit anderen im Rahmen von Brainstorming-Meetings innerhalb des Designteams. Um hierbei die VizBox nutzen zu können, ist die Ortsungebundenheit und der zentrale Datenbestand besonders wichtig, da dies ihren Einsatz insbesondere in Meetings mit mehreren Personen vorteilhaft unterstützt. Darüber hinaus bietet eine Umsetzung der VizBox auf Basis von Web-Technologien zusätzliche Funktionsmöglichkeiten, wie z.B. die Nutzung von Emailfunktionen zur einfachen Kontaktaufnahme mit dem Kurator der jeweiligen Komponente, Verlinken von externen Inhalten, die aktive Beteiligung von einzelnen Designern, bzw. der Community im Allgemeinen an der Pflege und Weiterentwicklung der Komponenten der VizBox, sowie die Verwendungsmöglichkeit eines XML-Standards für die einheitliche Beschreibung der einzelnen Komponenten.

5.6 Erweiterte Funktionalitäten der VizBox

Die Umsetzung der VizBox als webbasiertes Visualisierungssystem bietet Möglichkeiten für zusätzliche Funktionalitäten, welche für den Umgang mit der VizBox und insbesondere für ihren Einsatz als Arbeitstool nützlich sind. Diese erweiterten Funktionalitäten hängen z.T. nicht alle unmittelbar mit der Nutzung von Web-Technologien zusammen, werden aber dadurch begünstigt. Der folgende Abschnitt beschreibt einige dieser möglichen Funktionalitäten, aber auch hier dienen diese, wie schon die Designvorschläge aus Kapitel 5.4 zuvor, als Denkanstöße um die Möglichkeiten einer solchen Anwendung wie der VizBox aufzuzeigen.

Eine mögliche Funktionalität, welche den Einsatz der VizBox als Arbeitstool begünstigt, ist den Benutzern eine Möglichkeit zu bieten, ihre Erkenntnisse, welche sie aufgrund ihrer Recherche innerhalb der VizBox erhalten haben, zu archivieren, zu dokumentieren und auch zu kommunizieren. Dies könnte durch einen Ansatz gelöst werden, welcher die Einrichtung von persönlichen Bereichen in der Anwendung ermöglicht, wie z.B. einem *Personal Workspace* und einem *Shared Workspace*. Innerhalb eines solchen *Personal Workspace* hat der Benutzer beispielsweise die Möglichkeit, Rechercheergebnisse bzw. spezifische Visualisierungsbeschreibungen oder Visualization Patterns abzuspeichern, diese in eigen angelegten Kategorien wie z.B. *Interessante Interaktionsmöglichkeiten* oder *Ungewöhnliche Trefferanzeigen* (vgl. IDEO Web-Katalog) abzulegen und sie mittels einer Notizfunktion zu dokumentieren. Diese von ihm angelegte Auswahl kann er fortlaufend durch Hinzufügen oder Löschen von Objekten aktualisieren. Dies ermöglicht es Benutzern Arbeitsfortschritte und Teilergebnisse bzw. Objekte, welche für sie interessant erscheinen abzuspeichern.

Im Sinne einer Backup-Funktion sollten sich Zwischenstände dieses *Personal Workspace* in Form einer Datei zusätzlich abspeichern lassen und könnten so auch per Email an andere

Teammitglieder verschickt werden. Auf den *Personal Workspace* hat nur derjenige Benutzer Zugriff, der dafür auch autorisiert wurde. Jedoch hat der Inhaber des *Personal Workspace* die Möglichkeit andere Benutzer in seinen *Workspace* mit einzuladen und ihnen somit das Recht geben, diesen einzusehen bzw. darauf zuzugreifen. Dabei könnte unterschieden werden in Lese- und Schreibrechte. Dies bedeutet, es ist personenspezifisch, ob ein eingeladener Benutzer die Inhalte des *Personal Workspace* lediglich lesen oder diese auch modifizieren kann.
Diese Funktion des Einladens von Personen in einen *Workspace* ermöglicht es auch so genannte *Shared Workspaces* anzulegen. Diese könnten beispielsweise projektspezifisch oder auch arbeitsgruppenspezifisch sein. Durch das Nutzen der Funktion des *Shared Workspace* innerhalb von Brainstorming-Meetings können so gemeinsam Lösungen entwickelt, vorangetrieben, dokumentiert und kommuniziert werden. Zusätzlich ist es denkbar, weiterführende Materialien, wie z.B. Protokolle oder anderweitige Dokumente bzw. Objekte an dieser Stelle mit einzubinden zu können, um den Teammitgliedern eine Plattform zu bieten, welche es ermöglicht Materialien zu einem spezifischen Projekt zentral abrufen zu können. Darüber hinaus sollte die VizBox auch über ganz allgemeine Funktionen verfügen, wie z.B. dem Anbieten von Druckversionen der Visualization Patterns oder Visualisierungsbeschreibungen oder das Abspeichern von Rechercheergebnissen in Form einer Datei, z.B. im PDF-Format.

5.7 Vorteile der VizBox

Das nun folgende Kapitel fasst nochmals kurz und überblickartig die Vorteile der kombinierten Verwendung der beiden Ansätze - der Patterns und der Box-of-Ideas – und deren Einbettung in ein webbasiertes Visualisierungssystem für die VizBox zusammen.

Pattern

Mit Hilfe der Visualization Patterns ist es möglich, bestehendes und geprüftes Designwissen aus dem Bereich der Visualisierung von Daten, das auf mehreren Jahren praktischer Erfahrung basieren kann, in einem einheitlich strukturiertem Format zu dokumentieren, zu archivieren und zu kommunizieren. Sie unterstützen den Visualization Designer effektiv bei der Generierung qualitativ hochwertiger Designentwürfe, welche auch mit dem Ausdruck von „zeitlosem Design" bezeichnet werden können. Dabei sind die Visualization Patterns möglichst allgemeinverständlich geschrieben, was bedeutet, dass sie auf die Verwendung von so genannten Fachjargon verzichten. Dies begünstigt ihre Verwendung in einem interdisziplinären Designteam, wie es für den Entwurf einer Visualisierung erforderlich ist, insbesondere wenn im Zuge des Ansatzes des *User-Centered Designs*, welcher für ein erfolgreiches Design notwendig ist, Benutzer während des gesamten Entwicklungsprozesses beteiligt sind.

Durch die kurze prägnante Benennung der Pattern, welche den Kern der Idee wiedergibt, kann so ein allgemeines Vokabular, eine gemeinsame Sprache für den Designprozess einer Visualisierung aufgebaut werden, was die Kommunikation der einzelnen Personen des Designteams untereinander erleichtert und es ermöglicht, Entwürfe und Konzepte auf einer abstrakteren Ebene zu diskutieren. Damit lösen Pattern eines der größten Probleme, die beim interdisziplinären Design, wie dem von Visualisierungen, bestehen: die effektive Kommunikation. Somit stellen Patterns eine gute Methode dar, Designwissen zwischen Designteammitgliedern von unterschiedlichen Disziplinen zu kommunizieren und damit das Erzeugen qualitativer Designs zu unterstützen. [Borchers 2001]
Patterns bieten dem Designer umfassendes Wissen darüber, wann, wie und insbesondere warum die vorliegende Lösung anwendet werden kann. Da Patterns sich auf die Kernidee konzentrieren, die jedem Entwurf innewohnt, sind die angebotenen Lösungen solch generischer Natur, dass es möglich ist, diese unzählige Male zu verwenden ohne sich wiederholen. Die Anzahl der Vielfalt hängt dabei allein vom Designer ab. Im Idealfall unterliegen die Patterns innerhalb der Sammlung einer inhärenten hierarchischen Ordnung (von groß nach klein) und referenzieren somit sowohl Patterns, welche größer dimensionierte Bezüge adressieren, in welche die vorliegende Lösung eingebunden werden kann als auch Patterns, welche kleine dimensionierte Bezüge adressieren und somit die vorgeschlagene Lösung weiter verfeinern. Somit ergibt sich durch die Patternsprache für den Visualization Designer eine geordnete Vorgehensweise beim Entwurf einer Visualisierung, die auch einem typischen *top-down* Designprozess entspricht. Zudem ist es möglich, die Patternsprache mit Hilfe eines Graphen visuell darzustellen. Durch eine Patternsprache ist es möglich, ganze komplexe Objekte zu beschreiben.

Ein weiterer Vorteil der Verwendung von visualisierungsspezifischen Patterns für das Design einer Visualisierung ist, dass sie von unerfahrenen als auch von erfahrenen Designer genutzt werden können um Designvorschläge zu erarbeiten und Lösungen zu verfeinern. Denn die kollektive Erfahrung andere Designer wird durch Patterns auf eine Weise gesammelt, die es erlaubt unmittelbar angewendet zu werden. Das Lesen der Begründung des Patterns fördert das Verständnis dafür, warum Patterns besser funktionieren als das lediglich blinde Verfolgen einer Reihe von Guidelines.
Mit Hilfe der im Pattern referenzierten Beispiele (vgl. Bekannte Verwendungen), welche auch dazu dienen Benutzern potentielle Designlösungen zu zeigen, erhält der Visualization Designer einen praktischen Eindruck davon, wie und wo die Lösung des Pattern bereits angewandt wurde. Dabei gilt, je mehr Beispiele referenziert werden, desto höher liegt auch die Validität des Pattern. Aufgrund der generischen Form der Lösungsvorschläge der Patterns, bieten diese auch die Möglichkeit Visualisierungsmöglichkeiten aus vielen verschiedenen Bereichen, wie z.B. Webseiten, Softwaresysteme, Grafik- oder Bildbearbeitungsprogramme oder multimediale Ausbildungssoftware, Videospiele, digitale Enzyklopädien, Online-Dokumentationen, Terminals oder andere Geräte mit digitalen Interfaces, wie beispielsweise Palmtops, Handhelds oder MP3-Player, zu dokumentieren. Denn wenn ein Pattern für einen spezifischen Anwendungsbereich gilt, so gilt es auch für einen anderen.

Dieser Transfer wurde intuitiv oder per Zufall schon oft vollzogen, ohne jedoch, dass der Ausdruck „Pattern" verwendet worden wäre. Jedoch ist dieser Vorgang wesentlich effektiver, wenn man ihn bewusst, auf einem hohen Level der Abstraktion, ausführt. Da Visualization Patterns zeitloses Designwissen speichern, welches auch noch in vielen Jahren gültig sein wird - auch wenn die einzelnen Ausprägungen, wie sie in den heutigen Medien existieren, überholt sein werden - werden die Patterns, welche in ihnen stecken immer noch Bestand haben. Aufgrund dessen legen Visualization Patterns die solide praktische Grundlage, auf der neue Produkte entstehen. [Tidwell 1999].

Neben all den Vorteilen der Verwendung von Visualization Patterns für das Design von Visualisierungen, gibt es auch Nachteile, die mit den Patterns verbunden sind. Dabei lassen sich im Wesentlichen die folgenden beiden Hauptprobleme identifizieren.

Nachteil 1: Patterns hemmen die Kreativität

Wie Christopher Alexander selbst und auch alle anderen Autoren, der im Rahmen dieser Arbeit betrachten Patternsammlungen betonen, dokumentieren Patterns immer nur bereits bestehendes, bzw. auch verloren gegangenes Designwissen, bringen jedoch selbst keinen neuen theoretischen Boden hervor oder präsentieren neue innovative Techniken. Sie sind lediglich die Ergebnisse vorangegangener Argumente und Gedankengänge. Dies bedeutet, sie ersetzen nicht die Notwendigkeit von Kreativität. [Alexander et al. 1995][Gamma et al. 2004] Ganz im Gegenteil, denn obwohl Visualization Patterns zu effektiven Designs führen, hemmen sie die Kreativität eines Designers. Denn das Anbieten von bereits existenten, leicht anzuwendenden Lösungen kann dazu führen, dass der Visualization Designer weniger dazu geneigt ist, für sich selbst eine neuartige Lösung zu entwickeln.

Erfordernis der Entwicklung neuartiger Visualisierungen

Jedoch haben Designdisziplinen allgemein immer mit einem Medium der Konstruktion zu tun und die Entwicklung neuer Technologien bringen auch immer neue Bereiche für das Design mit sich. So hat beispielsweise die Architektur immer mit der Technologie des Bauens zu tun, Grafik Design mit den Technologien des Druckens und Produkt Design erreicht immer wieder neue Ausprägungen mit Hilfe neuer Technologien, wie beispielsweise mit der Entwicklung von Kunststoffen, welches die Vielfalt von möglichen Formen der alltäglichen Produkte erweitert. Die Entwicklung neuer Technologien und Techniken im Computerbereich beeinflusst auch die Disziplinen, die ihn als Medium haben, wie z.B. das Visualization Design. Diese Tatsache und die Problematik, dass die Quantität und Komplexität der Daten, mit denen sich Benutzer beschäftigen müssen zunehmend steigt und die Aufgaben, die mit Visualisierungen bearbeitet werden sollen zunehmend anspruchsvoller und spezifischer werden, macht es erforderlich neuartige und effektivere Formen der Visualisierung zu entwickeln. [Preece et al. 2002]

Nachteil 2: Unkomfortable Verwendung von Patterns

Die zweite Problematik liegt in der Komfortabilität der Verwendung von Patterns für den Designer im Zuge des Designentwurfs allgemein. So erfolgt die Dokumentation und Präsentation von Patterns prinzipiell in rein textueller Form entweder analog, in Form eines Buches oder virtuell, in Form von Webseiten mit Hypertextfunktionen. Jedoch bieten selbst die virtuellen Versionen kaum einen Mehrwert gegenüber der analogen Form, welche von Christopher Alexander initiiert wurde. Die Mehrwerte treten zumeist in Form von Hyperlinks zu den in der Patternbeschreibung referenzierten Patterns (vgl. *Common Ground* von Jenifer Tidwell) auf und in Form von Posting-Möglichkeiten, wie bei Martijn van Welie. Die virtuelle Form bietet des weiteren die Möglichkeit der Verwendung von XML zur einheitlichen Strukturierung und Darstellung der einzelnen Patterns. Dies wird, bzgl. der im Rahmen dieser Arbeit betrachteten Patternsammlungen, bislang von Jan Borchers und Martijn van Welie für ihre Sammlungen genutzt. Diese Art der Präsentation und Interaktion mit den Patterns ist jedoch für den Designer sehr mühselig und fordert viel Geduld von ihm ab. Sicherlich wächst mit der vermehrten Nutzung der Patternsammlung auch die Übung mit deren Umgang, jedoch stellt dies dennoch ein umständlicher und beschwerlicher Zugang zum Designwissen dar, was manchen Designer davon abhalten wird, diese Form der Wissensrepräsentation, trotz ihres nachweislichen Mehrwertes zu verwenden.

Box-of-Ideas / Visualisierungen

Durch Adaption des Ansatzes der Box-of-Ideas für die VizBox ist es möglich, genau den zuvor beschriebenen Nachteil, dass Visualization Patterns die Kreativität eines Designers hemmen, auszugleichen. Dies erfolgt mittels eines umfangreichen Ideenpools an Visualisierungen, welcher sowohl Visualisierungsbeschreibungen enthält, die innerhalb von Forschungs- oder Entwicklungsarbeiten entstanden sind, aber auch Beschreibungen von Visualisierungen, welche aus dem allgemeinen täglichen Umfeld stammen, wie z.B. Grafik- oder Bildbearbeitungsprogramme oder multimediale Ausbildungssoftware, Videospiele, Geräte mit digitalen Interfaces, digitale Enzyklopädien, Terminals, etc. Durch diese Formen der Visualisierungsmöglichkeiten kann sich der Designer zu neuen Entwürfen inspirieren lassen. Damit unterstützt die VizBox sowohl die Kreativitätstechnik des *Cased-Based Reasoning* als auch die der *Cross-Fertilization*. Dabei hat das *Case-based Reasoning* das Ziel, neue Probleme durch das Heranziehen von Wissen, entweder des eigenen oder das anderer Designer, welches von der Lösung früherer ähnlicher Probleme erworben wurde zu lösen. Beim *Cross-Fertilization* dagegen kommt es zu einer Befruchtung aus anderen Disziplinen und Anwendungsbereichen, indem Ideen oder Konzepte aus diesen fachfremden Gebieten auf das eigene Projekt übertragen werden. Durch Adaption, Kombination oder Modifikation dieser Ideen auf neue Anwendungsbereiche lassen sich so neue kreative Ideen finden und sich so spannende Lösungen erarbeiten, welche - zunächst als unmöglich erscheinend - eventuell im Vorfeld als Lösungsweg ausgeschlossen worden

wären. Durch das Betrachten und sich Beschäftigen mit Designlösungen aus dem Visualisierungsbereich sollen so viele und insbesondere neuartige und innovative Ideen generiert werden. Dieser Teil der VizBox hat somit das Ziel, Designern dabei zu helfen die eigenen Denkmuster, die jedem, nicht zuletzt aufgrund der eigenen Projekterfahrungen, zueigen sind, zu überwinden und offen zu bleiben für neue und ungewöhnliche Ideen, aus denen innovative Designs entstehen. Dieser Ansatz der kreativen Ideenfindung ist eine erfolgreiche Methode, welche von Designern bereits bevorzugt innerhalb eines Designprozesses eingesetzt wird.

Neben dem Ansatz der Box-of-Ideas fungiert dieser Ideenpool an Visualisierungen auch gleichzeitig als Beispielsammlung für die Visualization Patterns. Diese referenzieren die Visualisierungen, um dem Visualization Designer einen praktischen Eindruck davon zu geben, wie und wo die Lösung des Pattern bereits angewandt wurde. Dieser Part der Visualization Pattern stellt die Verknüpfung zum Ansatz der Box-of-Ideas her, sowie dieses wiederum von der Visualisierungsbeschreibung selbst referenziert wird. Die Visualisierungsbeschreibung selbst dient dabei der umfassenden Dokumentation der einzelnen Visualisierungen in Form von strukturiertem Hintergrundwissen (vgl. Web-Katalog von IDEO), welches für das Design von Visualisierungen relevant ist und referenziert zu weiterführenden Informationsquellen sowie multimedialen Inhalten. Dieses Wissen wird dem Visualization Designer in einem einheitlichen und übersichtlichen Format dargestellt, welches ihm ermöglicht, die für ihn relevanten Informationen schnell und bequem abzurufen. Damit fungiert die VizBox der Zentrierung, Dokumentierung, Archivierung und Kommunikation von Designwissen für den Bereich der Visualisierung von Daten.

Die Umsetzung der VizBox als webbasiertes Visualisierungssystem

Die Umsetzung der VizBox als webbasiertes Visualisierungssystem bietet gleich mehrere Vorteile. Zum einen bietet sie dem Visualization Designer durch die Verwendung von Visualisierungen und den damit verbundenen Interaktionstechniken die Möglichkeit auf bequeme und schnelle Art und Weise die Inhalte der VizBox – also die Visualization Patterns als auch den Ideenpool an Visualisierungen – zu explorieren.
Die Verwendung eines mehrdimensionalen Ansatzes für die Umsetzung, also der Einsatz mehrerer unterschiedlicher Visualisierungen für die Exploration der Inhalte der VizBox, hat den zusätzlichen Vorteil dem Designer mehrere Sichten auf den Datenbestand und ihm somit auch unterschiedliche Techniken für die Interaktion mit den Inhalten der VizBox geben zu können. Damit unterstützt die VizBox sowohl Techniken für die gerichtete Suche nach interessanten und für den Designprozess relevanten Objekten als auch Techniken, welche einen eher spielerischen Umgang für die Exploration der Objekte der VizBox – im Sinne des Box-of-Ideas Ansatzes - ermöglicht. Durch die große Menge an Information zu der Thematik von Visualisierungen, mit zusätzlichen multimedialen Inhalten und deren ansprechende Präsentation wird die VizBox zusätzlich zu einer Art von Wissensdatenbank für diesen Themenbereich. Somit stellt die VizBox auf der einen Seite ein Kreativitätstool dar,

welches der Unterstützung der Ideenfindungsphase im Designprozess einer Visualisierung dient und auf der anderen Seite erfüllt sie die Funktion einer Art digitalen Enzyklopädie, welche ihre Wissensdatenbank dem Benutzer auf ansprechende und informative Weise präsentiert und mittels Interaktionstechniken exploriert werden kann.

Der Einsatz von Web-Technologien für die Umsetzung der VizBox beinhaltet zum einen die allgemeinen Vorzüge solcher Anwendungen, wie beispielsweise die Einfachheit der Verteilung der Anwendung, die Zentralisierung der Anwendung auf dem Webserver (Zugriff via Web-Browser, zentraler Datenbestand, Plattformunabhängigkeit), die Ortsungebundenheit, welches insbesondere die Verwendung der VizBox in Meetings begünstigt, sowie Funktionsmöglichkeiten, wie die Nutzung von Emailfunktionen zur einfachen Kontaktaufnahme mit dem Kurator der jeweiligen Komponente, Verlinken von externen Inhalten, die aktive Beteiligung von einzelnen Designern, bzw. der Community im Allgemeinen an der Pflege und Weiterentwicklung der Komponenten der VizBox, sowie die Verwendungsmöglichkeit eines XML-Standards für die einheitliche Beschreibung der einzelnen Komponenten. Zum anderen begünstigt die Verwendung solcher Technologien zusätzliche Funktionalitäten, welche die Verwendung der VizBox als Arbeitstool ermöglichen, wie z.B. das Anlegen persönlicher Bereiche (vgl. *Personal Workspace* und *Shared Workspace*), Archivierung, Dokumentation und Kommunikation von Rechercheergebnissen, wie Abspeichern und Verwalten von Ergebnismengen oder einzelner Objekte, Notizfunktion, Einbinden von Objekten, Printmodi oder das Abspeichern bzw. Verschicken einer Auswahl per Email.
Dieser Vorteil der Verwendung von Visualisierungstechniken für die Umsetzung der VizBox bietet die Möglichkeit, die allgemeinen Vorzüge von visuellen Darstellungen und den damit verbundenen Interaktionsmöglichkeiten für den Designer, bzw. Anwender zu nutzen. Damit wirkt sie der zuvor genannten Problematik der unkomfortablen Verwendung der Patterns und der, in voran gegangenen Kapitel beschrieben für den Benutzer unattraktiven Präsentationsform von Visualisierungssammlungen, entgegen und ermöglicht die Realisierung des Ansatzes der Box-of-Ideas in einem rein virtuellen Bereich.

Die beiden Ansätze der Visualization Box (Viz Box) - die Visualization Patterns und die Box-of-Ideas – stellen somit zwei eigenständige und gleichberechtigte Ansätze dar, welche sich gegenseitig synergetisch ergänzen und in Form eines webbasierten Visualisierungssystems als Kreativitätstool zur Inspiration von Lösungsideen innerhalb des Designprozesses angewendet werden können, um qualitativ hochwertige und innovative Visualisierungen zu entwickeln. Durch ansprechende Visualisierungsformen und die informative Präsentation der Inhalte ist das sekundäre Ziel dieses Ansatzes, der Aufbau einer Art von Wissensdatenbank für die Thematik der Visualisierung von Daten, welcher dem digitaler Enzyklopädien (vgl. beispielsweise *Encarta Enzyklopädie*) ähnelt und es dem Benutzer ermöglicht, sich auf bequeme Art und Weise umfassend zu informieren.

6 Auswahl von Designvorschlägen

„To get a good idea, get lots of ideas."
[Marc Rettig]

Qualitativ hochwertiges Design

Beim Design von Objekten allgemein stellt sich die Frage, wie man zu qualitativ hochwertigen Designlösungen gelangt. Tatsächlich wird das Erzeugen einer großen Anzahl an Lösungen in Form von Brainstorming, als bester Weg zur Ideengenerierung betrachtet. Denn je mehr Ideen gefunden werden, desto wahrscheinlicher ist es, eine gute Idee dabei zu isolieren. Wie definiert sich eine gute Idee, bzw. „gutes" Design jedoch? Terry Winograd formuliert gutes Design wie folgt:

„Good design produces an object that works for people in a context of values and needs, to produce quality results and a satisfying experience." [Winograd 2002, XVI]

Für Visualisierungen bedeutet gutes Design, dass die Qualitätsansprüche an das Design einer Visualisierung, welche in Kapitel 2.3 beschrieben wurden, erfüllt werden. Da Visualisierungen bzw. Visualisierungssysteme jedoch interaktive Softwaresysteme sind, wird das Visualization Design auch stark von den Methoden, Techniken und Wertesystemen beeinflusst, welche im Software Engineering und der HCI verwendet werden. [Wilkins 2003]

Prinzipiell besteht ein großer Unterschied darin, über „gutes" Design nur zu schreiben und Kriterien festzulegen, die gutes Design ausmachen, und gutes Design tatsächlich zu erzeugen. Es gibt eine ganze Reihe von Literatur, welche sich mit systematischen Methoden für das Design und insbesondere für das Design von Interfaces und iterativen Systemen befasst. Auch wenn diese eine nützliche Anleitung bieten, so ist das *„Designen"* selbst eine kreative Aktivität, welche nicht vollständig auf Standardschritte reduziert werden kann. Es existiert keine formale, konsistente oder umfassende Theorie des Designs oder eine universale Methodologie, welche einfach angewendet werden könnte. Zwar ist es möglich und auch notwendig, systematische Prinzipien und Methoden während des Designprozesses zeitweise anzuwenden und den Prozess selbst in zuvor festgelegten Strukturen ablaufen zu lassen, jedoch ist die *„Aktivität des Designens"* selbst inhärent komplex und chaotisch. Jede Entscheidung, welche der Designer trifft, hat sowohl beabsichtigte als auch unbeabsichtigte Effekte. Die *„Aktivität des Designens"* ist somit kein Prozess der sorgfältigen Planung und Ausführung sondern mehr eine Konversation zwischen Designer, dem Designobjekt selbst und ihrer Umgebung. Designbewusstsein geht immer einher mit Intuition, taktischem Wissen und Bauchgefühl. Beim Design geht es nicht nur darum, bei der Lösung von Problemen kreativ zu sein, sondern vor allem kreativ

zu sein hinsichtlich der Identifizierung von Problemen - indem man sich die Bedürfnisse, welche Menschen haben, jedoch selbst noch nicht erkannt haben, vergegenwärtigt. Die Fähigkeit des Designers ist nicht auf eine Gruppe von Methoden reduzierbar und kann nicht einfach durch einen gut durchstrukturierten Lehrplan erlernt werden, wie dies in anderen Disziplinen möglich ist. Jedoch kann man sich die Fähigkeiten des „*Designens*" durch die Interaktion zwischen Lernendem und Lehrer, Designer und Kritik aneignen, wie dies schon seit langem in der Tradition der Ausbildung in den Designbereichen der Fall ist. [Winograd 2002]

Auswahl von Designlösungen

Um zu einer qualitativ hochwertigen Lösung zu gelangen müssen im Laufe des Designprozesses einer Visualisierung Entscheidungen getroffen werden, bezüglich dessen welche der verschiedenen Designlösungen weiter entwickelt werden soll. Allgemein werden diese Entscheidungen von der Information, die über die Daten, die Benutzer und ihre Aufgaben welche in der Analysephase des Designprozesses gesammelt wurden und von der technischen Machbarkeit einer Idee, beeinflusst. Bei dem Design von Visualisierungen stellt die Art, wie der Benutzer mit dem System interagiert, die zentrale Komponente dar. Daher ist eine gute Möglichkeit zwischen alternativen Designvorschlägen zu wählen und damit zu Designentscheidungen zu gelangen, Benutzer mit ihnen interagieren zu lassen und ihre Erfahrungen, Präferenzen und Vorschläge für Verbesserungen zu diskutieren. Dies ist fundamental für die Entwicklung eines benutzerorientierten Ansatzes. [Wilkins 2003]

Evaluation

Die Evaluation von Designlösungen stellt eine gute Möglichkeit der Entscheidungsfindung innerhalb des Designprozesses dar. Dabei ist die Evaluation die Methode des Bestimmens der Usability und Akzeptanz des Produkts oder des Designs. Diese wird anhand einer Auswahl an verschiedenen Kriterien, wie beispielsweise die Anzahl der Fehler, die ein Benutzer bei dessen Verwendung macht, wie ansprechend es ist, wie gut es die Anforderungen erfüllt, etc. gemessen. [Preece et al. 2002]

Dazu gibt es eine Reihe von Evaluationstechniken, welche im Bereich der HCI (vgl. Kapitel 3.1.2) entwickelt wurden. Darüber hinaus werden zur Evaluation von Visualisierungslösungen auch speziell für den Bereich der Visualisierung von Daten entwickelte Techniken eingesetzt. So schlägt Barry Wilkins beispielsweise für die Evaluation von Visualisierungen diverse visualisierungsspezifische Methoden vor. Eine dieser Methoden diverse Designvorschläge zu evaluieren ist, jede Visualisierung mit Benutzern unter kontrollierten Bedingungen zu testen. Die gesammelten Daten werden dann statistischen Analysen unterzogen, um Usability Fragen zu beantworten. Dies können beispielsweise Fragen sein, wie: welche Visualisierung beansprucht die wenigste Zeit um erlernt zu werden, welche hat den

höchsten Prozentsatz an richtigen Fragen in der kürzesten Zeit, welche bevorzugen die Benutzer usw.
Eine weitere Methode ist die so genannte *Mapping Evaluation*, welche auf den Resultaten der Bewertung von diversen Wahrnehmungs-Kodierungstechniken, wie z.B. das Ranking des Datentyps zur visuellen Eigenschaft basiert. Diese Bewertungen sollen als simpler Maßstab für die perzeptuelle Effektivität einer Visualisierung dienen. Ein weiterer Vorschlag ist die *Visualization Heuristic Evaluation*, welche eine Evaluationsmethode darstellt, die eine Klassifikation der Visualisierungsheuristiken bezüglich ihres Effekts auf die Usability nutzt. Dabei wägt der Visualization Designer verschiedene Usability Faktoren, basierend auf ihrer Relevanz für den Anwendungsbereich und für die Usability Ziele, ab. Für jeden Designvorschlag und für jede Heuristik bestimmt der Designer eine Punktzahl, basierend darauf, wie gut die Heuristik seiner Meinung nach umgesetzt wurde. Um zu einer finalen Visualisierungsbewertung zu erlangen, wird die gewichtete Summe jedes Heuristikeffekts auf die Usability Faktoren mit den Punktständen kombiniert. Somit ist es möglich alle Designlösungen anhand dieser Bewertungen miteinander zu vergleichen.
Eine dritte Methode ist die *Metric Evaluation*, welche anhand von Visualisierungsmetriken verschiedene Designvorschläge miteinander vergleicht. [Wilkins 2003] Entsprechend dem spezifischen Projekt können eine oder oftmals auch mehrere dieser Evaluationsmethoden angewandt werden, da die Verwendung einer einzelnen Evaluationstechnik oftmals nicht alle Faktoren abdecken kann, welche für ein Design relevant sind.

Jede Form der Evaluation erfordert eine Art interaktive Version des Designs. Es gibt verschiedene Techniken um diese „Interaktion" zu erreichen. Dies können beispielsweise papierbasierte Prototypen sein, welche sehr schnell und kostengünstig herzustellen, und sehr effektiv für das Identifizieren von Problemen in den frühen Stadien des Designs sind. Durch den Einsatz von rollenspielenden Benutzern kann ein realistisches Gefühl erlangt werden, bezüglich dessen, wie es sein wird mit dem Produkt zu interagieren. [Preece et al. 2002]

Die Aktivitäten der Entwicklung alternativer Designs, das Erzeugen interaktiver Versionen des Designs und die Evaluation sind stark miteinander verflochten: Alternativen werden mittels interaktiver Versionen des Designs evaluiert und die Ergebnisse fließen in weiteres Design ein. Diese Iteration ist eine der Schlüsselcharakteristiken des Designprozesses. [Preece et al. 2002]

7 Zusammenfassung und Ausblick

Diese Arbeit stellt das Konzept der VizBox vor, einer webbasierten Bibliothek für das Design von Visualisierungen. Kapitel eins und zwei geben eine kurze Einführung in die Thematik der Visualisierung von Daten, wie sie sich von der Pre-Computer-Phase zur Computer-Phase entwickelt hat und stellen dar, weshalb visuelle Repräsentation für die Darstellung von Daten so effektiv funktionieren. Anhand des Referenzmodells für Visualisierung von Ben Shneiderman wird der Erzeugungsprozess einer Visualisierung konzeptionell beschrieben und anschließend erläutert, wie sich qualitativ hochwertige Visualisierung definieren. Kapitel drei führt in den Designprozess von Visualisierungen ein, welcher auf diversen Methodologien aus dem Software Design beruht. An dieser Stelle werden die verschiedenen Phasen des Prozesses vorgestellt und die, für das Design von Visualisierungen, relevanten Faktoren definiert. Dieser Prozess wird dann mit dem allgemeinen Designprozess, wie er allen Designdisziplinen zugrunde liegt, verglichen und damit im Zusammenhang stehende Kreativitätstechniken vorgestellt.
Kapitel vier stellt die beiden interdisziplinären Ansätze für das Visualization Design – Pattern und Box-of-Ideas – vor, welche aus den traditionellen Designbereichen der Architektur und des Produkt Design stammen und speziell auf die Generierung qualitativ hochwertiger und innovativer Designlösungen ausgerichtet sind. Zusammenfassend werden dann sowohl die Nach- als auch die Vorteile der beiden Ansätze aufgezeigt.
In Kapitel fünf wird die Visualization Box vorgestellt, welche diese beiden Ansätze für das Visualization Design in Form von Visualization Patterns und strukturierten Visualisierungsbeschreibungen adaptiert, modifiziert und in einem Visualisierungssystem synergetisch miteinander kombiniert. Dazu wurden, auf Basis der vorangegangenen Untersuchungen, sowohl ein eigenes Patternformat als auch ein eigener Strukturierungsvorschlag für die einheitliche Beschreibung von Visualisierungen entwickelt, sowie eine Möglichkeit für deren kombinierte synergetische Anwendung erarbeitet. Anschließend werden die Vorteile der Umsetzung der VizBox als webbasiertes Visualisierungssystem hervorgehoben und mögliche Designvorschläge für deren visuelle Ausprägung gemacht.
Die Arbeit schließt mit einem zusammenfassenden Überblick über die Vorteile des Ansatzes der VizBox, deren Ziel es ist, den Visualization Designer innerhalb des Entwicklungsprozesses einer Visualisierung – und dabei speziell die kreative Ideenfindung in der Designphase – mit einem Kreativitätstool zu unterstützen, welches das Erzeugen alternativer Lösungsideen fördert und zu qualitativ hochwertigen und innovativen Designlösungen führt.

Aufgrund der Aktualität der Thematik der Daten und der Relevanz der Entwicklung qualitativ hochwertiger und innovativer Visualisierungen, stellt die Umsetzung der VizBox als Kreativitätstool für den Entwurfsprozess einer Visualisierung, ein spannendes Thema dar. Basierend auf den Erfahrungen, die aus dem praktischen Umgang mit der VizBox, gewonnen werden, wird es sicherlich noch zu Verbesserungen bzw. Erweiterungen hinsichtlich der Strukturierungsansätze der VizBox kommen. Jedoch wurde mit dieser Arbeit die Basis hierfür gelegt. Diese Erfahrungen werden sowohl aus der Umsetzung der VizBox als Visualisierungssystem gewonnen werden, als auch durch die Entwicklung der Visualization Pattern und die Einpflege der Visualisierungsbeschreibungen, sowie durch deren Verwendung als Kreativitätstool und Wissensdatenbank für die Thematik der Visualisierung von Daten.

Insbesondere durch die Einbindung der Community für die Visualisierung von Daten in die Entwicklung der Visualization Patterns und der Visualisierungsbeschreibungen der VizBox, wird ein Mehrwert erwartet, den eine begrenzte Gruppe von Visualisierungsforschern in diesem Umfang nicht leisten kann. Die erfolgreiche Etablierung der Pattern durch die Software Engineering Community - mittels Diskussionsplattformen, Workshops auf Konferenzen, Foren etc. - hat gezeigt, dass es möglich ist, Synergieeffekte, welche durch die gemeinsame Arbeit innerhalb der Community zustande kommen, vorteilhaft zu nutzen. Eine Erweiterung würde dieser Ansatz der VizBox noch durch die Nutzung des Konzepts der sehr erfolgreichen Wiki-Technik - der Wikipedia Enzyklopädie[17] – erfahren, in welcher alle Beiträge gemeinsam von den Benutzern erarbeitet und frei genutzt werden können. So entstünde eine wertvolle Wissensdatenbank über die Thematik der Visualisierung von Daten und für das Design qualitativ hochwertiger und innovativer Visualisierungen.

[17] Wikipedia – Die freie Enzyklopädie
URL: http://de.wikipedia.org/wiki/Hauptseite

8 Quellenverzeichnis

[Alexander et al. 1977]
Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, Shlomo Angel. *A Pattern Language.* Oxford University Press, New York, 1977.

[Alexander 1979]
Christopher Alexander. *The Timless Way of Building.* Oxford University Press, New York, 1979.

[Alexander et al. 1995]
Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, Shlomo Angel. *Eine Muster-Sprache: Städte, Gebäude, Konstruktion.* Hermann Czech (Hrsg.), Löcker Verlag Wien, 1995.

[Borchers 2001]
Jan Borchers. *A Pattern Approach to Interaction Design.* Chichester [u.a.]: Wiley, 2001.

[Card 2003]
Stuart Card. *Information Visualization.* In: The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications, Lawrence Erlbaum Associates, Inc., S. 544-582.

[Card et al. 1999]
Stuart K. Card, Jock D. Mackinlay, Ben Shneiderman. *Readings in Information Visualization. Using Vision to Think.* San Francisco, CA (Morgan Kaufmann), 1999.

[Chi 2000]
Ed H. Chi. *A Taxonomy of Visualization Techniques Using the Data State Reference Model.* In: Roth, Steven F.; Keim, Daniel A. (Eds.): Proceedings of IEEE Information Visualization 2000. Conference: Salt Lake City, UT, October 9-10 2000. Los Alamitos, CA (IEEE Computer Soc. Press) 2000. S. 69 –75.

[Däßler 1999]
Rolf Däßler. *Informationsvisualisierung. Stand, Kritik und Perspektiven.* In: Methoden/Strategien der Visualisierung in Medien, Wissenschaft und Kunst, Wissenschaftlicher Verlag Trier (WVT), 1999.
URL: http://fabdp.fh-potsdam.de/daessler/paper/InfoVis99.pdf (26.04.04)

[de Haas 1999]
Stephan de Haas. *Softwarearchitektur?! Ein Vergleich mit dem Bauwesen.*
OBJEKTspektrum 6, 1999, S. 60-70.

[Donath 2004]
Dirk Donath*. Christopher Alexander, seine Theorien und Bauten.*
URL: http://www.uni-weimar.de/~donath/c-alexander98/ca98-html.htm (17.06.2004)

[Form 2005]
Form.de > Profile > Online > I > Ideo Product Development
URL: http://www.form.de/w3.rDefault.php?objId=88 (21.08.05)

[Gamma et al. 2004]
Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software.* Addison-Wesley, München, 2004.

[IDEO 2005a]
Ideo > Our Work > view by client: Ideo > Tech Box
URL: http://www.ideo.com/portfolio/re.asp?x=50035 (09.03.05)

[IDEO 2005b]
Ideo > About Us > Methods
URL: http://www.ideo.com/about/index.asp?x=3&y=1 (09.03.05)

[Furuhata 2005]
Information Visualization and Visualization Techniques - A Collection of Visualizations. Rika Furuhata, Fujishiro Ichikawa Lab, Japan, 2000.
URL: http://pfp7.cc.yamaguchi-u.ac.jp/~ichikawa/iv/index.html (05.07.05)

[Lin 1998]
Xia Lin. *Information Visualization: Ten Years in Review.*
URL: http://research.cis.drexel.edu/~xlin/presentations/iv.ppt (12.04.04)

[Mann 2002]
Thomas M. Mann. *Visualization of search results from the World Wide Web.* Dissertation, Universität Konstanz, 2002.
URL: http://www.ub.uni-konstanz.de/kops/volltexte/2002/751/ (02.04.04)

[Mayhew 1999]
Deborah J. Mayhew. *The Usability Engineering Lifecycle.* San Francisco: Morgan Kaufmann, 1999.

[OLIVE 2005]
On-line Library of Information Visualization Environments (OLIVE). University of Maryland, 1997.
URL: http://otal.umd.edu/Olive/ (23.06.05)

[Perzel&Kane 1999]
Kimberly Perzel und David Kane. *Usability Patterns for Applications on the World Wide Web*. PloP Konferenz 1999.
URL: http://jerry.cs.uiuc.edu/~plop/plop99/proceedings/Kane/perzel_kane.pdf (28.07.05)

[Pfründer 2002]
Jörg Pfründer. *HCI Patterns*. Seminararbeit im Bereich Mensch-Computer Interaktion. Universität Konstanz, FB Informatik und Informationswissenschaft. 2002.

[Preece et al. 2002]
Jennifer Preece, Yvonne Rogers, Helen Sharp. *Interaction Design: Beyond Human-Computer Interaction.* New York: Wiley, 2002.

[Pricken 2003]
Mario Pricken. *Visuelle Kreativität. Kreativitätstechniken für neue Bildwelten in Werbung, 3-D-Animation & Computer-Games.* Verlag Hermann Schmidt Mainz, 2003.

[Pricken 2005]
Mario Pricken. *Kribbeln im Kopf. Kreativitätstechniken & Brain-Tools für Werbung & Design.* Verlag Hermann Schmidt Mainz, 2005.

[Reiterer 2004]
Harald Reiterer. *Visuelle Exploration digitaler Datenbestände*. Knowledge Media Design - Grundlagen und Perspektiven einer neuen Gestaltungsdisziplin, in: Thissen F., Stefan P.F., Springer, 2004.

[Reeps 2004]
Inga Elisabeth Reeps. *Joy-of-Use – eine neue Qualität für interaktive Produkte.* Masterarbeit, Universität Konstanz, 2004.
URL: http://www.ub.uni-konstanz.de/kops/volltexte/2004/1386/ (08.01.05)

[Schumann&Müller 2000]
Heidrun Schumann, Wolfgang Müller. *Visualisierung. Grundlagen und allgemeine Methoden.* Springer-Verlag Berlin Heidelberg, 2000.

[Shneiderman 2001]
Ben Shneiderman. *Supporting Creativity with Advanced nformation-Abundant User Interfaces*. In Earnshaw, R., Guedj, R., Van Dam, A., Vince J. (eds.), Human-Centred Computing, Online Communities, and Virtual Environments, Springer-Verlag, London (2001), S. 469-480.

[Shneiderman 1996]
Ben Shneiderman. *The Eyes Have It: A Task by Data Type Taxonomy of Information Visualizations.* Proc. 1996 IEEE Conference on Visual Languages, Boulder, Colo. (Sept. 3-6, 1996), S. 336-343.

[Spence 2001]
Robert Spence. *Information Visualization*. Harlow, ACM Press (Addison-Wesley), 2001.

[Tidwell 1999]
Jenifer Tidwell. *Common Ground: A Pattern Language for Human-Computer Interface Design.* 1999.
URL: http://www.mit.edu/~jtidwell/interaction_patterns.html (25.05.2004)

[Tufte 1983]
Edward R. Tufte. *The Visual Display of Quanititive Information.* Cheshire, CT (Graphics Press), 1983.

[Tufte 1997]
Edward R. Tufte. *Visual Explanations.* Cheshire, CT (Graphics Press), 1997.

[Tufte 1990]
Edward R. Tufte. *Envisioning Information*. Cheshire, CT (Graphics Press), 1990.

[van Welie et al. 2001]
Martijn van Welie, Gerrit C. van der Veer, and Anton Eliens. *Patterns as Tools for User Interface Design.* In: Tools for Working with Guidelines, Springer Verlag London, 2001.

[Ware 1999]
Colin Ware. *Information Visualization. Perception for Design*.
San Francisco, CA (Morgan Kaufmann), 1999.

[Wikipedia 2005]
Wikipedia. Die freie Enzyklopädie. 2005.
URL: http://de.wikipedia.org/wiki/XML (18.10.05)

[Wilkins 2003]
Barry Wilkins. *MELD: A Pattern Supported Methodology for Visualisation Design.* Dissertation, University of Birmingham, 2003.
URL: http://www.cs.bham.ac.uk/~bxw/thesis/MELD_PhD_Thesis.pdf (17.02.05)

[Wilkins 2004]
Barry Wilkins. *Visualisation Patterns.* 2004.
URL: http://www.cs.bham.ac.uk/~bxw/ (30.11.04)

[Wilkinson 1999]
Leland Wilkinson. *The Grammar of Graphics.* New York: Springer, 1999.

[Winograd 2002]
Terry A. Winograd. *Bringing Design to Software.* Ed. by Terry Winograd. New York, N.Y.: ACM, 2002.