Universität Hamburg
FB Informatik
Datenbanken und Informationssysteme

STUDIENARBEIT

PolitIcon: Entwurf einer Bilddatenbank für den Index zur politischen Ikonographie

September 1995

Andreas Bösch
Oberstraße 18F
20144 Hamburg

Tel.: 040 / 4205148

Betreuer:
Prof. Dr. Joachim W. Schmidt

Inhaltsverzeichnis

1. Einleitung und Motivation **1**
1.1 Der Index zur politischen Ikonographie 1
1.2 Zielsetzung und Aufbau der Arbeit 2

2. Analyse mit OMT **4**
2.1 OMT - ein Überblick 4
2.1.1 Die vier Phasen der Entwicklung 4
2.1.2 Die Analysephase 5
2.2 StP OMT 7
2.3 Analyse des Index zur politischen Ikonographie 9
2.3.1 Die Problembeschreibung 9
2.3.2 Objekte und Klassen 9
2.3.3 Verbindungen und Assoziationen 11
2.3.4 Verbindungsattribute 12
2.3.5 Generalisierung und Vererbung 13
2.3.6 Das Objektmodell für den Index zur politischen Ikonographie 14

3. Die Datenbank **16**
3.1 Umsetzung des OMT-Objektmodells in ein relationales Datenbankschema . . 16
3.1.1 Übersetzung von Objektklassen in Tabellen 17
3.1.2 Übersetzung von binären Assoziationen in Tabellen 17
3.1.3 Übersetzung von Generalisierungsbeziehungen in Tabellen 19
3.2 Access - ein Überblick 22
3.2.1 Der Access-Tabelleneditor 23
3.2.2 Der Access-Beziehungseditor 24
3.3 Die Jet Datenbank-Engine 25

4. Die Benutzerschnittstelle **28**
4.1 VisualBASIC - ein Überblick 28
4.2 Der Protoyp 30
4.2.1 Der Stichwort-Index 31
4.2.2 Der Künstler-Index 34

5. Zusammenfassung und Ausblick **36**
5.1 Erfahrungen mit OMT 36
5.2 Erfahrungen mit Access 36
5.3 Erfahrungen mit VisualBASIC 37
5.4 Der Prototyp – eine Abschlußbetrachtung 37

A. Anforderungsdefinitionen **39**
A.1 Kooperationsperspektiven zwischen dem Index zur politischen Ikonographie und dem Fachbereich Informatik an der Universität Hamburg 39
A.1.1 Beschreibung des Bestandes 39
A.1.2 Erwartungen an eine Digitalisierung des Index 41
A.2 Beschreibung der für den Index zur politischen Ikonographie zu erstellenden Software 41

B. Objektdiagramme **43**
B.1 Wichtige Klassen im Index zur politischen Ikonograhie 43
B.2 Klassen und Objekte 43
B.3 Attribute und Werte 44
B.4 Verbindungen und Assoziationen 44
B.5 Verbindungsattribut 44
B.6 Generalisierung 45
B.7 OMT Objektmodell Stichworte 45
B.8 OMT Objektmodell Karten 46

Literaturverzeichnis **47**

1. Einleitung und Motivation

Bildinformationen werden digital aufbereitet und gespeichert, um sie möglichst vielen Menschen möglichst schnell und individuell zugänglich zu machen. Herkömmliche nicht digitale Techniken sind Bildbände, Kataloge oder Fotosammlungen auf Karteikarten (wie beim Index zur politischen Ikonographie, siehe Abschnitt 1.1). Der Nachteil dieser Speicherungsformen von Bildinformationen ist, daß die Bilder nur in einem festgelegten Kontext erscheinen und der Betrachter die Bilder nicht in beliebig viele ihm sinnvoll erscheinende Zusammenhänge bringen kann, denn dies würde ein mehrfaches physisches Vorhandensein des Bildmaterials erfordern, was wiederum hohe Kosten nach sich zöge.
Es gilt also, das mehrfache Vorhalten von Daten zu vermeiden und stattdessen deren Zugreifbarkeit zu verbessern. Dies wird erreicht, indem Daten kontextunabhängig gespeichert werden und der (individuelle) Kontext durch Verweise auf die Daten, aber nicht durch die physische Anordnung der Daten selbst entsteht.

1.1 Der Index zur politischen Ikonographie

Die politische Ikonographie untersucht politische Botschaften bildlicher Darstellungen und geht dabei von der Annahme aus, daß sich die politische Wirkung vergangener und gegenwärtiger Machthaber nicht nur in Akten und Verträgen niederschlägt, sondern in Zeremonien, Gebäuden, Denkmälern, Medaillen und eben Bildern [Warnke 93]. Der Index zur politischen Ikonograhie ist eine Sammlung von ca. 200.000 Photographien und Abbildungen, die auf Karten im Postkartenformat aufgeklebt und nach etwa einhundert Hauptstichworten geordnet sind. So sind z. B. unter dem Stichwort *Frieden* 1300 Bildkarten versammelt, die wiederum Unterstichworten zugeordnet sind, so etwa 400 Bildkarten dem Unterstichwort *Allegorien* und 300 Bildkarten dem Unterstichwort *Friedensschlüsse*. Das Zugriffskriterium auf die Bildkarten ist allein der aus Haupt- und Unterstichworten bestehende Index. Ein Problem, das sich im Laufe der Arbeit mit dem Index ergeben hat, ist, daß eine Bildkarte oft mehreren Stichworten zugeordnet werden könnte, man jedoch nur ein Exemplar hat. Umgangen wird dieser Mißstand mit Hilfe von Kopien der jeweiligen Karten. Auf diesen Kopien wird vermerkt, wo sich die jeweilige Originalbildkarte befindet. Eine weitere Unzulänglichkeit ist das Fehlen sekundärer Indizes. Ein rechnergestützter Bildindex, der die Bild- und Textdaten digital abspeichert, könnte die vorgestellten Probleme lösen sowie die Arbeit mit dem Index wesentlich erleichtern und beschleunigen.

1.2 Zielsetzung und Aufbau der Arbeit

Ziel der vorliegenden Arbeit ist eine Software für den Index zur politischen Ikonographie, die die in Abschnitt 1.1 beschriebenen Probleme löst.
Eine Software basiert in jedem Fall auf einem genauen Verständnis des zu lösenden Problems. Um dieses Verständnis zu erlangen, ist es nötig, eine genaue Analyse der Anwendungsdomäne durchzuführen. Das Analysedokument ist in enger Zusammenarbeit mit Fachleuten aus der Anwendungsdomäne zu erarbeiten, da bei ungenauer oder unvollständiger Analyse die gewünschte Funktionalität der Software nicht erreicht werden kann. Die Analyse des Index zur politischen Ikonographie wurde mit Hilfe der *Object Modeling Technique* nach [Rumbaugh et al. 91] und dem darauf basierenden CASE[1]-Werkzeug Software through Pictures / Object Modeling Technique (StP OMT) durchgeführt (siehe Kapitel 2). OMT ermöglicht es, einen Teilbereich der realen Welt so objektorientiert zu modellieren, daß das Modell

- verständlich ist,
- die Anforderungen an das System widerspiegelt,
- als Basis für die Abstimmung zwischen Anwender und Softwareentwickler dient.

Im Laufe der Arbeit mit den Kunsthistorikern hat sich gezeigt, daß die Verständigung zwischen Softwareentwickler und späteren Anwendern nur bis zu einem gewissen Grad mit Hilfe von Beschreibungen bzw. Objektdiagrammen sinnvoll ist. Gerade bei den Anwendern tritt relativ schnell eine Ermüdung ein, da für sie das Arbeitsgebiet, das sie dem Softwareentwickler nahebringen müssen, selbstverständlich ist und sie sich ihres impliziten Wissens darüber nicht bewußt sind. Ohne dieses Wissen aber hat der Softwareentwickler Schwierigkeiten, sich in die Probleme der Anwender hineinzudenken. Um diese Kommunikationsprobleme zu beseitigen und den Kunsthistorikern eine erste Idee eines computergestützten Index zur politischen Ikonographie zu geben, lag es nahe, einen Prototypen der Software zu entwickeln. Nach [Floyd 84] und [Grønbæk 88] werden in der Softwareentwicklung zwei Arten von Prototypen unterschieden:

Horizontaler Prototyp: Bei einem *horizontalen* Protoyp sind alle sichtbaren Teile der Benutzerschnittstelle eines neuen Computersystems implementiert, d.h. es können Bildschirmdialoge simuliert werden, ohne tatsächlich Daten zu berechnen.

Vertikaler Prototyp: Bei einem *vertikalen* Protoyp sind einige der Systemfunktionen in ihrer beabsichtigten endgültigen Form implementiert, so daß eine in der Realität vorkommende Arbeitsaufgabe bearbeitet werden kann.

Es wurde ein vertikaler Prototyp implementiert, um den späteren Anwendern schon früh eine Vorstellung von der Funktionsweise der Software zu geben, da diese bis zu diesem Zeitpunkt außer mit einem Textverarbeitungsprogramm unter dem Betriebssystem MS-DOS noch nie mit Rechnern gearbeitet hatten. Der Protoyp besteht aus folgenden Komponenten:

[1] Computer Aided Software Engineering

- Datenverwaltung
- Applikationsalgorithmen
- Benutzerschnittstelle

Diese Aufteilung wurde vorgenommen, um die Komplexität des Problems zu reduzieren. Außerdem ist es so möglich, einzelne Komponenten (wie z.B. die Benuzerschnittstelle) zu ersetzen ohne große Veränderungen an den anderen Teillösungen vorzunehmen.
Kapitel 3 beschreibt die Datenverwaltung des Prototypen für den Index zur politischen Ikonographie. Zuerst wird die Abbildung des bei der Analyse entstandenen OMT-Objektmodells auf ein relationales Datenbankschema untersucht. In den zwei weiteren Abschnitten werden mit dem Datenbankprogrammiersystem Access und der Jet Datenbank-Engine die technischen Grundlagen der Datenverwaltung erläutert. Access wurde zur Realisierung der Datenverwaltungsebene ausgewählt, da es über sehr einfach zu bedienende Editoren zur Erstellung von Relationen und relationalen Datenbankschemata verfügt.
Bei den Applikationsalgorithmen handelt es sich im wesentlichen um Standardalgorithmen für Datenbankzugriff und Bilddarstellung. Diese werden daher nicht näher untersucht. Da der Übergang von den Applikationsalgorithmen zur Benutzerschnittstelle einer Anwendung problematisch ist, wurde zur Implementierung dieser Komponenten die Programmiersprache VisualBASIC gewählt. Bei VisualBASIC handelt es sich um eine Sprache der vierten Generation (4GL, *fourth generation language*), mit der sich einerseits leicht die o.g. Applikationsalgorithmen realisieren lassen und die es andererseits ermöglicht, ohne Programmieraufwand Benutzerschnittstellen zu erzeugen (siehe Kapitel 4).
Die Erfahrungen im Umgang mit OMT, dem Datenbankprogrammiersystem Access und der Programmiersprache VisualBASIC und eine kritische Abschlußbewertung meiner Arbeit mit Vorschlägen zu weitergehenden Verbesserungen des Prototypen und möglichen Anschlußarbeiten sind die Bestandteile des abschließenden Kapitels 5.

2. Analyse mit OMT

Das Ziel der objektorientierten Analyse ist, einen Teilbereich der realen Welt so zu modellieren, daß das Modell

- verständlich ist,
- die Anforderungen an das System widerspiegelt,
- als Basis für die Abstimmung zwischen Anwender und Softwareentwickler dient.

In diesem Kapitel wird in Abschnitt 2.1 allgemein die *Object Modelling Technique* (OMT) aus [Rumbaugh et al. 91] und in Abschnitt 2.2 das darauf basierende CASE[1]-Werkzeug Software through Pictures / Object Modeling Technique (StP OMT) aus [Int 94] der Firma Interactive Development Environments vorgestellt. Nach dieser Einführung folgt in Abschnitt 2.3 anhand der Analyse des Index zur politischen Ikonographie eine vertiefende, aber nicht vollständige Darstellung des Objektmodells von OMT.

2.1 OMT - ein Überblick

OMT ist eine aus einer ganzen Reihe von Methoden zur objektorientierten Entwicklung von Software (vgl. [Stein 93] oder [Fowler 93]). Es handelt sich hierbei nicht um eine Programmiertechnik, sondern um ein Modellierungskonzept, das von einer Programmiersprache unabhängig ist. Zu der Methode gehört eine Notation, die objektorientierte Konzepte grafisch abbildet (siehe Abschnitt 2.3).

2.1.1 Die vier Phasen der Entwicklung

Die OMT-Methode basiert darauf, ein Modell einer Anwendungsdomäne zu entwickeln und dann während des Designs des Systems Implementationsdetails hinzuzufügen. Die Methode umfaßt vier Phasen:

Analyse: Ausgehend von einer informalen Problembeschreibung wird in der Analyse ein Bereich der realen Welt möglichst kurz, präzise und verständlich modelliert. Das Analysemodell legt fest, welche Funktionalität das gewünschte System erfüllt; es sollte keinerlei Implementationsentscheidungen beinhalten.

[1] Computer Aided Software Engineering

Systementwurf: Beim Systementwurf werden Entscheidungen über die Gesamtarchitektur getroffen, d.h., das geplante System wird in seine Hauptkomponenten, die in OMT Subsysteme genannt werden, unterteilt. Desweiteren müssen die Anforderungen an die vom System zu erbringende Leistung und die sich daraus ergebenden Hardwareanforderungen abgeschätzt werden.

Objektentwurf: Diese Phase besteht aus der Erweiterung des Analysemodells um implementierungsbezogene Objekte, Klassen, Datenstrukturen und Algorithmen [Matthes 94].

Implementierung: Die Objektklassen und Beziehungen, die während des Objektentwurfes entwickelt wurden, werden in Konzepten von Programmiersprachen oder Datenbanken oder als Hardwareimplementierung realisiert. Die Programmierung sollte ein relativ kleiner und mechanischer Teil des Entwicklungsprozesses sein, da alle wichtigen Entscheidungen bereits während des Designs getroffen werden sollten.

Von besonderer Wichigkeit ist die Analysephase, da in dieser die Weichen für das spätere System gestellt werden, d.h. Fehler, die in dieser Phase gemacht werden, können in den nachfolgenden Phasen nur noch mit hohem Aufwand beseitigt werden. Die Analyse wird deshalb im folgenden Abschnitt genauer vorgestellt.

2.1.2 Die Analysephase

Analyse ist der erste Schritt der OMT-Methode. Das Ziel dieser Phase ist ein präzises, verständliches und korrektes Modell der realen Welt.
In Abbildung 2.1 wird das Vorgehen deutlich. Am Anfang des Analyseprozesses steht eine Problembeschreibung, die in erster Linie vom Benutzer, möglicherweise aber auch in Zusammenarbeit mit dem Entwickler bzw. Manager formuliert wird. Die Problembeschreibung ist eine textuelle Beschreibung der Systemanforderungen und soll die Frage beantworten:

> *Was* soll das System leisten?

und nicht

> *Wie* soll das System arbeiten?

Es ist in der Analysephase nicht wichtig, Probleme, die das Design und die Implementation betreffen, zu formulieren (z.B. Algorithmen, Datenstrukturen, Gesamtarchitekur, ...). Am Anfang des Analyseprozesses ist ein generelles Verständnis des Problems anzustreben. Darum sollten die folgenden Punkte in einer Problembeschreibung Beachtung finden:

- Umfang des Problems
- Benötigte Ressourcen
- Kontext der Anwendung

- Annahmen
- Leistungsanforderungen

In der Analysephase des Index zur politischen Ikonographie haben die Kunsthistoriker ohne Kenntnis dieser in [Rumbaugh et al. 91] aufgestellten Anforderungen eine Problembeschreibung erstellt, die die meisten dieser Punkte beinhaltet (siehe Anhang A.1). Das spricht für die Natürlichkeit der o.g. Anforderungen an eine Problembeschreibung.

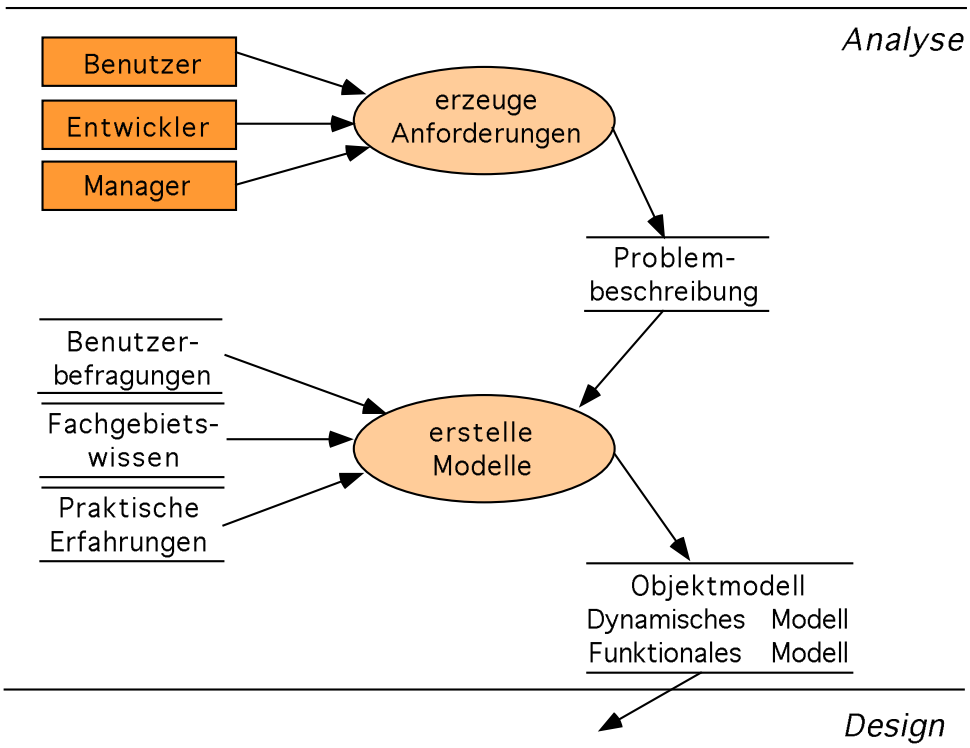


Abbildung 2.1: Der Analyseprozeß im Überblick

Ausgehend von einer Problembeschreibung, die in der Regel von den Benutzern erstellt wird, wird nun mit Hilfe von Benutzerbefragungen, Fachgebietswissen und praktischer Erfahrung ein Analysedokument erstellt. Hierbei versteht man unter Fachgebietswissen das Wissen über die Anwendungsdomäne. Das Analysedokument dient dann als Grundlage für die späteren Designphasen. Dadurch, daß am Anfang des Softwareentwicklungsprozesses ein genaues Modell der Anwendungsdomäne erstellt werden muß, wird der Entwickler gezwungen, Mißverständnisse in dieser sehr frühen Phase auszuräumen.

Das Analysedokument besteht aus drei Modellen:

Objektmodell: Das *Objektmodell* beschreibt die statische Struktur der Objekte im System und ihre Beziehungen zueinander. Es enthält *Objektdiagramme*. Ein Objektdiagramm ist ein Graph, in dem die Knoten Klassen und die Kanten Beziehungen zwischen Klassen sind.

Dynamisches Modell: Das *dynamische Modell* beschreibt die Veränderungen des Systems über die Zeit. Es enthält *Zustandsdiagramme*. Ein Zustandsdiagramm ist ein Graph, in dem die Knoten Objektzustände und die Kanten von Ereignissen ausgelöste Transitionen zwischen Objektzuständen sind.

Funktionales Modell: Das *funktionale Modell* beschreibt die Transformation von Daten durch Prozesse. Es enthält *Datenflußdiagramme*. Ein Datenflußdiagramm ist ein Graph, in dem die Knoten Prozesse und die Kanten Datenflüsse sind.

Das Objektmodell ist die Basis der beiden anderen Modelle, da es nötig ist, zuerst zu beschreiben, *was* sich ändert, bevor man beschreibt, *wann* oder *wie* es sich ändert. Das Objektmodell wird in Abschnitt 2.3 näher erläutert.

2.2 StP OMT

Während sich 90 Prozent aller CASE-Werkzeuge auf strukturierte Methoden wie z.B. 'Strukturierte Analyse' nach [DeMarco 79] und Entity-Relationship-Modelle nach [Chen 76] stützen, die strikt zwischen Daten- und Funktionssicht trennen, werden diese Sichten in objektorientierten Methoden wie z.B. OMT integriert [Versteegen 93].
In diesem Abschnitt wird das CASE-Werkzeug StP OMT vorgestellt, mit dem die Objektmodelle für das PolitIcon-Projekt entwickelt wurden. Diese Objektmodelle waren Diskussionsgrundlage für Benutzerbefragungen und dienten der Verständigung zwischen Benutzer und Entwickler über das vorhandene Datenmaterial und dessen Struktur.
StP OMT erfüllt folgende Anforderungen:

- interaktiver Grafikeditor
- mehrbenutzerfähiger Datenspeicher (*repository*)
- Versionsmanagement
- Zugangskontrollmechanismen
- Fähigkeiten zum Speichern von verschiedenen Konfigurationen und Voreinstellungen für Benutzer, Gruppen oder Projekte
- Einbettung in die Implementierungsphase, z. B. automatische Codegenerierung

Wie in Abbildung 2.2 aus [Int 94] gezeigt wird, bietet StP OMT eine Reihe von Editoren und Hilfsmitteln an, die es erlauben, OMT Modelle zu entwickeln. Wichtig für diese Arbeit waren im wesentlichen die beiden folgenden Editoren:

Objektmodell Editor: Mit diesem Editor lassen sich OMT-Objektmodelle erzeugen und verändern, da die von [Rumbaugh et al. 91] entwickelte Notation vollständig unterstützt wird.

Klassentabellen Editor: Im Klassentabellen Editor können Klassen mit ihren Attributen und Operationen (siehe Abschnitt 2.3.2) vollständig definiert werden.

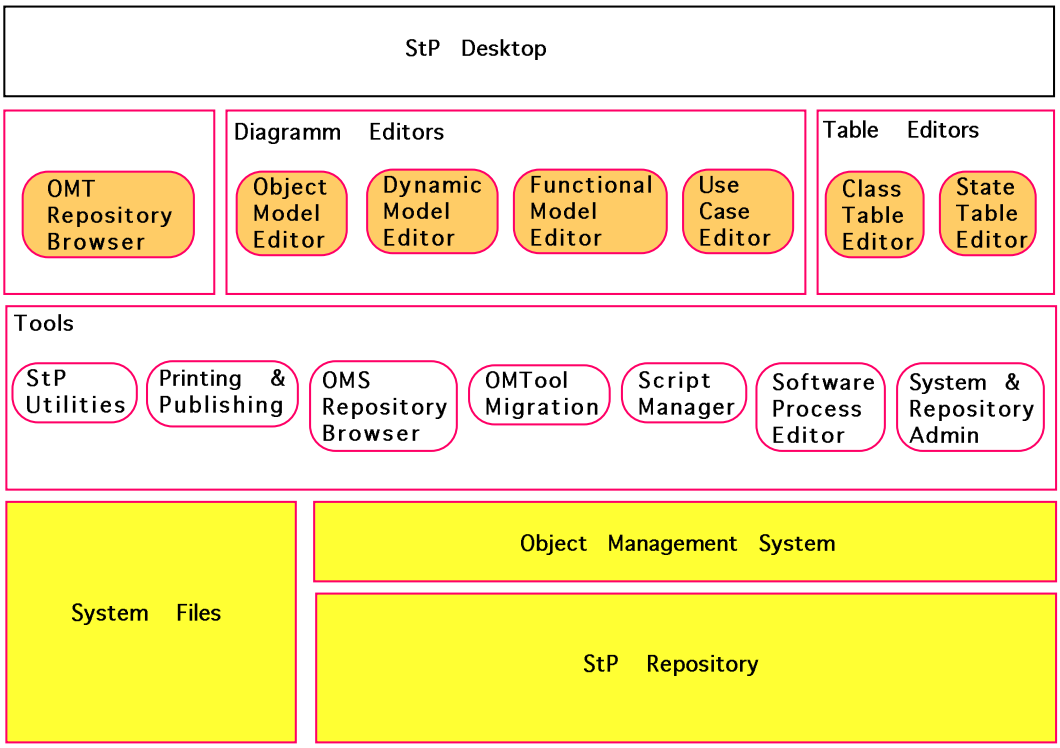


Abbildung 2.2: Die Systemarchitektur von StP OMT

Die Struktur und alle Details eines OMT Modells werden im StP Repository gespeichert und sind über das Object Management System zugreifbar. Erwähnenswert ist an dieser Stelle noch der integrierte Script Manager, mit dem sowohl die Generierung von C++ Kode als auch die Prüfung auf Konsistenz und semantische Korrektheit möglich ist. Gerade die Möglichkeit, solche Überprüfungen automatisch durchführen zu können, ist in großen Projekten mit mehreren Entwicklern wichtig.

2.3 Analyse des Index zur politischen Ikonographie

Die Analyse wurde in zwei Schritten durchgeführt:

1. Erstellung einer Problembeschreibung
2. Umsetzung der Problembeschreibung in ein OMT-Objektmodell

Ein *Objektmodell* beschreibt die statische Struktur eines Systems. Zur statischen Struktur eines Systems gehören:

- Objekte
- Beziehungen zwischen Objekten
- Attribute und Operationen, die eine Klasse von Objekten charakterisieren

Hier sollen nun die Konzepte und die dazugehörige Notation des OMT-Objektmodells als das grundlegende Modell der OMT Methodik erläutert werden. Am Beispiel des Index zur politischen Ikonographie wird die Entwicklung eines Objektmodells verdeutlicht.

2.3.1 Die Problembeschreibung

Wie in Abschnitt 2.1.2 bereits dargestellt, steht vor der Erstellung eines Objektmodells eine adäquate Problembeschreibung, die die Anforderungen an das System erläutert. Ein Benutzerhandbuch für das gewünschte System ist eine gute Problembeschreibung.
Die erste Problembeschreibung ist von den Mitarbeitern der politischen Ikonographie entworfen worden (siehe Anhang A.1). Sie enthält bereits alle für eine Problembeschreibung wichtigen Anforderungen (Abschnitt 2.1.2), wenn sich auch einige der enthaltenen Angaben im weiteren Analyseprozess als unvollständig herausgestellt haben. So fehlte z.B. eine wichtige Eigenschaft der Bildkarten völlig, nämlich die Auftragsnummer der Fotostelle. Diese Auftragsnummer wird meist auf der Rückseite der Bildkarten eingetragen und dient dem Wiederfinden der nach diesen Nummern abgelegten Negative zu den Bildern.
Diese Problembeschreibung wurde mit den Mitarbeitern der Politischen Ikonographie besprochen und eine zweite Version erstellt (siehe Anhang A.2). Beide Dokumente dienten dann zusätzlich zu den Notizen der Gespräche mit den Kunsthistorikern als Grundlage für das Objektmodell. Insgesamt fanden in der Analysephase fünf jeweils ca. zweistündige Treffen mit den Kunsthistorikern statt, bei denen die Problembeschreibungen und darauf aufbauend die Entwürfe zu den OMT-Objektmodellen besprochen und diskutiert wurden.

2.3.2 Objekte und Klassen

Ein *Objekt* ist ein Konzept, eine Abstraktion oder eine Sache mit klar umrissenen Grenzen, das in einem Anwendungszusammenhang sinnvoll ist. So ist z. B. im Index zur politischen Ikonographie eine Bildkarte oder ein Stichwort, unter dem diese Bildkarte abgelegt ist, ein

Objekt. Objekte haben den Zweck, einerseits das Verständnis der realen Welt zu fördern und andererseits als Basis für die Implementation zu dienen.
Die Dekomposition eines Problems in Objekte hängt vom individuellen Blickwinkel des Betrachters ab. Es gibt nicht *die* korrekte Repräsentation.
Alle Objekte haben eine Identität und sind unterscheidbar, auch wenn sie eventuell gleich aussehen und sich auch im Verhalten nicht unterscheiden. *Objektidentität* ist eines der wichtigen Konzepte in der Objektorientierung [Khoshafian, Abnous 90], z.B. im Gegensatz zum relationalen Modell [Lockemann, Schmidt 87].
Eine *Objektklasse* beschreibt eine Menge von Objekten mit gleichartigen Eigenschaften (Attributen), gemeinsamem Verhalten, gleichartigen Beziehungen zu anderen Objekten und derselben Semantik. In unserem Fall sind z.B. die Menge der Bildkarten und die Menge der Stichworte Objektklassen.

2.3.2.1 Objektdiagramme

Mit Hilfe von *Objektdiagrammen* lassen sich Objekte, Klassen und die Beziehungen zwischen diesen formal grafisch abbilden. Man unterscheidet zwei Arten von Objektdiagrammen:

Klassendiagramm: Ein Klassendiagramm ist ein Schema - ähnlich dem eines Entity-Relationship Modells [Fowler 93] - zur Beschreibung vieler verschiedener Instanzen von Daten. Ein Klassendiagramm beschreibt Objektklassen.

Instanzendiagramm: Ein Instanzendiagramm stellt eine bestimmte Menge von Objekten und ihre Beziehungen zueinander dar. Instanzendiagramme sind für die Dokumentation von Testläufen und Diskussionsbeispielen sinnvoll. Zu einem gegebenen Klassendiagramm gibt es eine unendliche Menge von Instanzendiagrammen. Ein Instanzendiagramm beschreibt Objektinstanzen.

Abbildung 2.3 zeigt ein Klassendiagramm (links) und ein mögliches Instanzendiagramm (rechts). Die Objekte *Ludwig XIV. und B. Franklin* und *Tetrarchen-Denkmal an San Marco* sind Instanzen der Klasse *Bildkarte*.

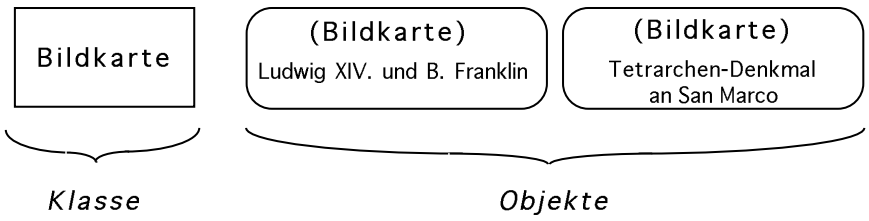


Abbildung 2.3: Klassen und Objekte

In Anhang B.1 ist eine Sammlung der für den Index zur politischen Ikonographie wichtigen Klassen abgebildet.

2.3.2.2 Attribute

Ein *Attribut* ist ein Datenwert, der zu jedem Objekt einer Klasse gehört. So sind z.B. Bildtitel und die Entstehungszeit des Bildes Attribute der Klasse Bildkarte. Jedes Attribut hat einen Wert für jede Instanz eines Objektes. Verschiedene Objektinstanzen können gleiche oder verschiedene Werte für ein gegebenes Attribut haben.
Abbildung 2.4 zeigt die Klasse *Bildkarte* mit den Attributen *Bildtitel* und *Bildentstehungszeit*, die beide vom Typ String sind. Das eine Objekt der Klasse *Bildkarte* hat den Wert *Ludwig XIV. und B. Franklin* als Bildtitel, das andere *Tetrarchen-Denkmal an San Marco*. Entsprechend ist es bei den Werten für das Attribut *Bildentstehungszeit*.

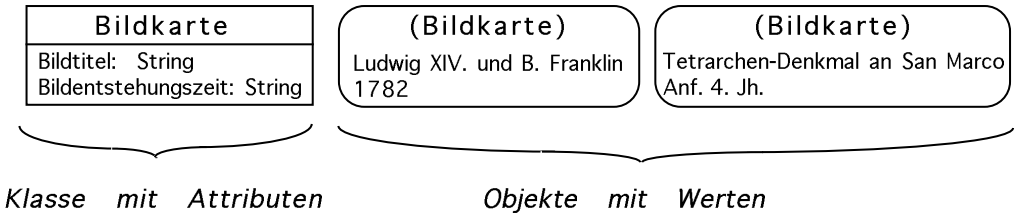


Abbildung 2.4: Attribute und Werte

Neben den Attributen hat jede Klasse von Objekten auch ihre eigenen *Operationen*, die auf oder von den Objekten der Klasse angewendet werden können. Auf die Objekte der Klasse Bildkarte könnte z.B. die Operation *Bildkarte anzeigen* angewendet werden. Nach der OMT-Methodik werden Operationen erst mit der dynamischen bzw. funktionalen Modellierung in das Objektmodell eingefügt, da ja erst dann die benötigten Operationen bekannt sind, die ein bestimmtes dynamisches oder funktionales Verhalten produzieren sollen. Weil in dieser Arbeit keine dynamische und funktionale Modellierung durchgeführt wurde, sind in den folgenden Objektdiagrammen nur die Attribute der jeweiligen Klassen aufgeführt.

2.3.3 Verbindungen und Assoziationen

Eine *Verbindung* ist eine physische oder konzeptuelle Beziehung zwischen Objektinstanzen. So ist z.B. die Bildkarte 'Ludwig XIV. und B. Franklin' *abgelegt unter* dem Stichwort 'politische Begegnungen'. Eine Verbindung ist die Instanz einer Assoziation.
Eine *Assoziation* beschreibt eine Menge von Verbindungen mit gleicher Struktur und Semantik. So ist z.B. eine Bildkarte *abgelegt unter* einem Stichwort. Alle Verbindungen in einer Assoziation verbinden Objekte der selben Klassen. Eine Assoziation beschreibt eine Menge potentieller Verbindungen, so wie eine Klasse eine Menge potentieller Objekte beschreibt.
Assoziationen sind bidirektional, d.h. binäre Assoziationen gelten in beiden Richtungen. So wird z.B. aus der Beziehung 'eine Bildkarte *ist unter* einem Stichwort *abgelegt*' in umgekehrter Richtung '*unter* einem Stichwort *sind* Bildkarten *abgelegt*'. In Wirklichkeit ändert sich an der Assoziation nur der Name, die Bedeutung bleibt gleich.

Abbildung 2.5 zeigt eine n:m Assoziation mit entsprechenden Verbindungen. Sowohl Assoziationen als auch Verbindungen werden als Linien dargestellt.

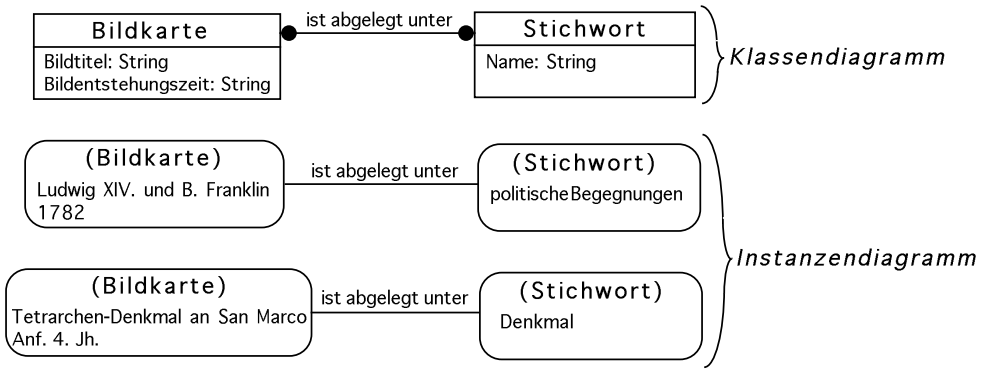


Abbildung 2.5: Verbindungen und Assoziationen

Kardinalität

Die *Kardinalität* legt fest, wieviele Instanzen einer Klasse zu einer Instanz einer anderen Klasse in Verbindung stehen dürfen. Objektdiagramme visualisieren Kardinalität mit speziellen Symbolen an den Enden der Assoziationslinien. In Abbildung 2.5 handelt es sich um eine n:m Assoziation, d.h eine Bildkarte kann unter mehreren Stichworten abgelegt werden und ein Stichwort kann mehrere Bildkarten enthalten.
Die Notation der möglichen Kardinalitäten ist in Abbildung 2.6 dargestellt.

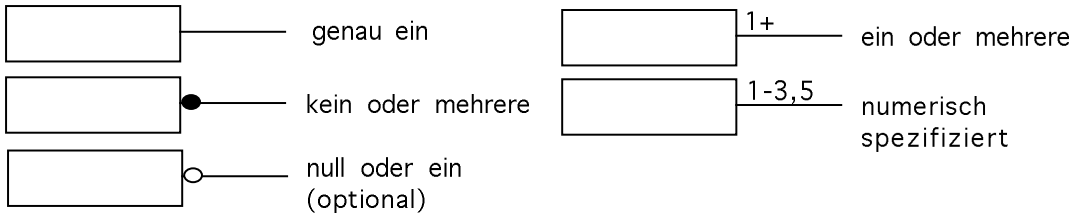


Abbildung 2.6: Kardinalitäten

2.3.4 Verbindungsattribute

Ein Attribut ist eine Eigenschaft von Objekten einer Klasse. Dementsprechend ist ein *Verbindungsattribut* die Eigenschaft einer Verbindung in einer Assoziation. So ist z.B. in Abbildung 2.7 *istOriginal* ein Attribut von *abgelegt unter*. Die Notation betont die Ähnlichkeit

von Objektattributen und Verbindungsattributen. Die n:m Assoziation in der Abbildung 2.7 bietet zwingende Gründe für ein Verbindungsattribut. Eine Karte kann unter mehreren Unterstichworten abgelegt werden und ein Unterstichwort kann mehrere Karten enthalten. Da der Benutzer aber gerne wissen möchte, wo sich die Originalkarte im physischen Index zur politischen Ikonographie befindet, ist das Verbindungsattribut *istOriginal* notwendig und kann ohne Informationsverlust nicht einer der beiden Klassen zugordnet werden.

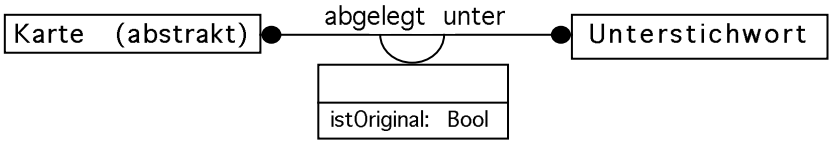


Abbildung 2.7: Verbindungsattribut

2.3.5 Generalisierung und Vererbung

Generalisierung ist die Beziehung zwischen einer Klasse und einer oder mehrerer ihrer verfeinerten Versionen. Die Klasse, die verfeinert wird, nennt man *Superklasse*, jede ihrer Verfeinerungen wird als *Subklasse* bezeichnet. In Abbildung 2.8 ist *Stichwort* die Superklasse der Klassen *Hauptstichwort* und *Unterstichwort*. Gemeinsame Attribute und Operationen einer Gruppe von Subklassen werden der Superklasse zugeordnet und gelten dann für jede Subklasse, ohne daß sie dort noch einmal aufgeführt werden. Man sagt, jede Subklasse *erbt* die Kennzeichen ihrer Superklasse.

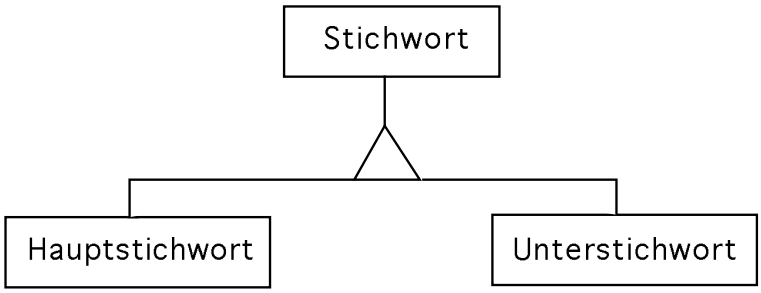


Abbildung 2.8: Generalisierung

2.3.6 Das Objektmodell für den Index zur politischen Ikonographie

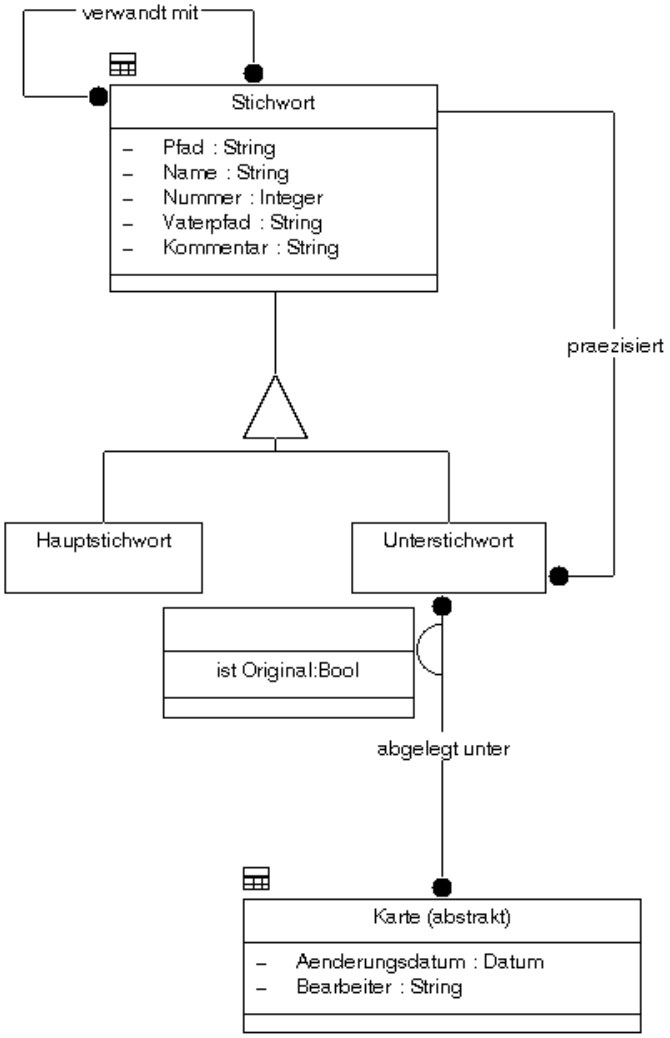


Abbildung 2.9: OMT Objektmodell Stichworte

Das Objektmodell für den Index zur politischen Ikonographie wurde der besseren Übersicht wegen in zwei Teile geteilt. Die Abbildung 2.9 beschreibt die Struktur des Index selbst, also die Struktur von Haupt- und Unterstichworten. Man sieht in der Abbildung deutlich, daß ein Stichwort entweder Hauptstichwort oder Unterstichwort sein kann und daß mehrere Unterstichworte ein beliebiges Stichwort präzisieren können. Außerdem können beliebig viele Karten unter beliebig vielen Unterstichworten abgelegt werden. Wenn eine Karte mehreren Unterstichworten zugeordnet wurde, enthält das Assoszіationsattribut *istOriginal* die Information, welchem Unterstichwort die Originalkarte zugordnet wurde. Das ist wichtig, wenn

neben der später möglichen Bildschirmdarstellung der Karte auf das physische Original zugegriffen werden soll.
In Abbildung 2.10 ist das Modell unterhalb der Stichwort-Ebene zu sehen.

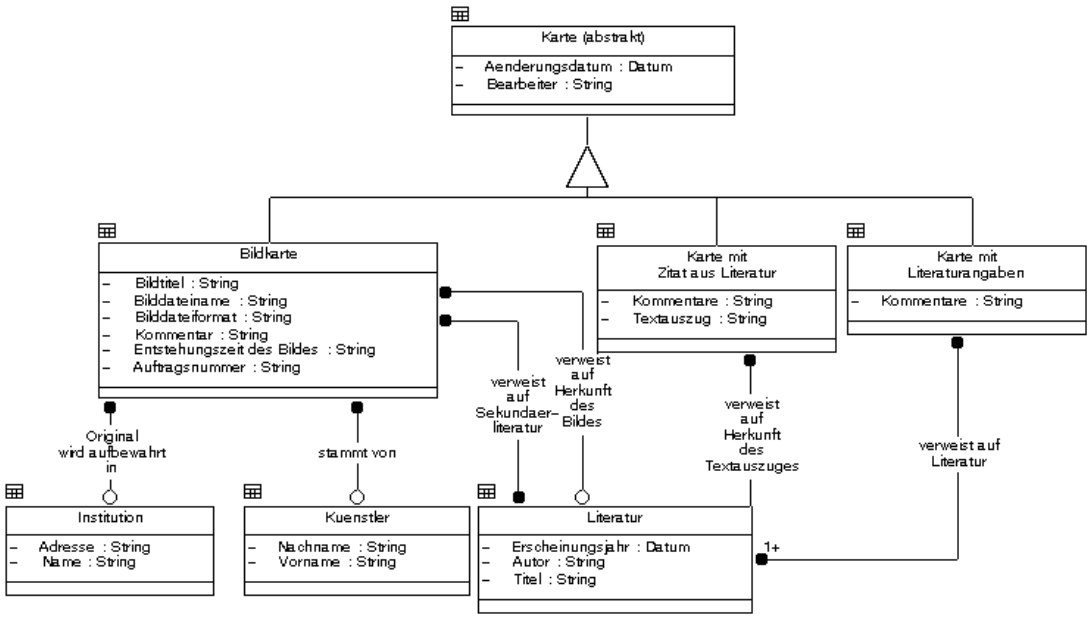


Abbildung 2.10: OMT Objektmodell Karten

Die Klasse *Karte* ist eine abstrakte Klasse, d.h. es gibt keine Instanzen dieser Klasse. Die Klasse *Karte* dient als Superklasse für die Klassen *Bildkarte*, *Karte mit Zitat aus Literatur* und *Karte mit Literaturangaben*. Die abstrakte Klasse *Karte* vererbt also ihre Attribute *Änderungsdatum* und *Bearbeiter* an ihre drei Subklassen (siehe Abschnitt 2.3.5). Die Generalisierung ist hier disjunktiv, d.h. eine Karte kann entweder eine *Bildkarte* oder eine *Karte mit Zitat aus Literatur* oder eine *Karte mit Literaturangaben* sein, niemals jedoch sowohl *Bildkarte* als auch *Karte mit Zitat aus Literatur*.
Über die Klasse *Literatur* könnte eine Verbindung zum Datenbestand der Literaturdatenbank der politischen Ikonographie, die mit dem Programm Allegro verwaltet wird, hergestellt werden. Außerdem gibt es in der Marburger Bibliothek bereits eine große Künstlerdatenbank, von der Teile bereits in den Prototyp eingearbeitet wurden. Diese Daten werden in Marburg mit Hilfe von MIDAS, dem Marburger Inventarisations-, Dokumentations- und Administrations-System [Heusinger 94] erfaßt und hier in der Klasse *Künstler* verwaltet.

3. Die Datenbank

Die Datenbank ist der Kern des für den Index zur politischen Ikonographie entwickelten Protoypen. Der physische Index zur politischen Ikonographie besteht aus einer großen Sammlung von Karteikarten mit aufgeklebten Bildern, die nach Stichworten geordnet sind. Die Datenbank des Prototypen wurde in Form von miteinander verknüpften Tabellen für die textuellen Daten (z.B. Stichworte oder Dateinamen von zu Bildkarten gehörigen Bildern) und einem Dateibaum, in dem die Bilder in Verzeichnissen mit jeweils hundert Bildern abgelegt sind, realisiert. Die Bilder liegen jeweils einmal im Kleinformat (ca. 50KB) und einmal im Großformat (ca. 1MB) vor. Dies hat den Vorteil, daß bei Anfragen an den Datenbestand zuerst eine Übersicht der in Frage kommenden Bilder im Kleinformat angezeigt werden kann und somit mehrere Bilder gleichzeitig auf dem Bildschirm sichtbar sind. Ist danach eines der Bilder von größerem Interesse, kann es im Großformat angesehen werden.
Im Abschnitt 3.1 wird die Abbildung des bei der Analyse entstandenen OMT-Objektmodells auf ein relationales Datenbankschema nach [Rumbaugh et al. 91] erläutert, bevor in Abschnitt 3.2 das für den Prototyp verwendete Datenbankprogrammiersystem Access aus [Baloui 94], [Hüskes, Kurzidim 94] und [Zerbe 93] kurz vorgestellt wird. Danach wird in Abschnitt 3.3 ein Überblick über die Jet Datenbank-Engine gegeben. Diese sorgt für die Kommunikation zwischen der Datenbank und der Benutzerschnittstelle (siehe Kapitel 4).

3.1 Umsetzung des OMT-Objektmodells in ein relationales Datenbankschema

Das relationale Datenmodell basiert auf dem Konzept der Tabelle. Ein RDBMS (*Relational Database Management System*) wie Access ist ein Computerprogramm, daß Tabellen administriert. Es setzt sich nach [Rumbaugh et al. 91] aus drei Hauptteilen zusammen:

Daten: Daten werden in Tabellen repräsentiert. Tabellen haben eine festgelegte Anzahl von Spalten und eine beliebige Anzahl von Zeilen. Die Spalten einer Tabelle werden *Attribute* genannt und entsprechen den Attributen in Objektmodellen. Die Zeilen werden *Tupel* genannt und entsprechen den Objektinstanzen und den Verbindungen.

Operatoren: Operatoren dienen der Manipulation von Tabellen. Der Standard sowohl nach ANSI (*American National Standards Institute*) als auch ISO (*International Standards Organisation*) ist hier die Anfragesprache SQL. Mit Hilfe des SQL *select state-*

ment können Daten aus Tabellen gelesen werden. Andere SQL-Operatoren können Daten aus Tabellen löschen, Tabellen erzeugen, Zeilen in Tabellen einfügen oder löschen, etc.

Integritätsbedingungen: Man unterscheidet zwei Arten von Integrität. Die *existentielle* Integrität besagt, daß jede Tabelle genau einen Primärschlüssel hat. Ein Primärschlüssel ist eine Kombination von einem oder mehreren Attributen, deren Wert jede Zeile der Tabelle eindeutig identifiziert. *Referentielle* Integrität erfordert, daß das RDBMS die Konsistenz jedes Fremdschlüssels mit seinem zugehörigen Primärschlüssel überwacht. Ein Fremdschlüssel ist ein Primärschlüssel einer Tabelle, der in eine andere Tabelle integriert ist.

3.1.1 Übersetzung von Objektklassen in Tabellen

Jede Klasse in einem OMT-Objektmodell entspricht einer oder mehrerer Tabellen. Eine Tabelle kann aber ebenfalls einer oder mehreren Klassen entsprechen (siehe Abschnitt 3.1.2). In Abbildung 3.1 ist die Konvertierung einer Objektklasse in OMT-Notation in eine Tabelle dargestellt. Die Klasse *Karte mit Zitat aus Literatur* hat die Attribute *Zitat* und *Kommentar*. Die zugehörige Tabelle enthält diese Attribute und zusätzlich die implizite Objekt-ID, die eine eindeutige Identifzierung der einzelnen Datensätze ermöglicht und daher als Primärschlüssel dient.

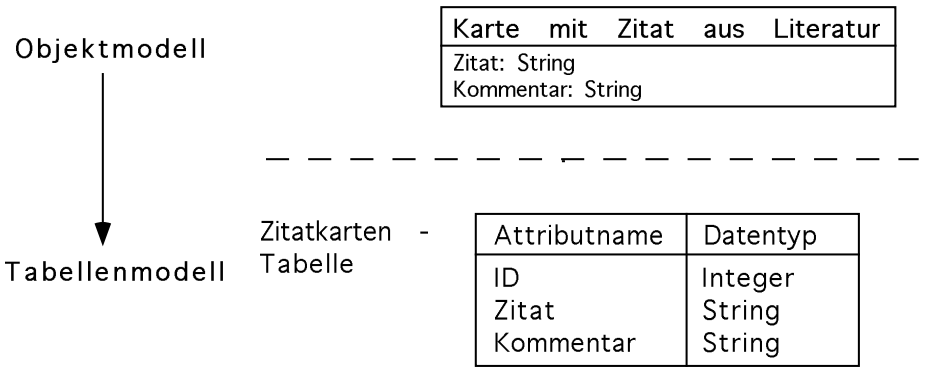


Abbildung 3.1: Konvertierung einer Klasse in eine Tabelle

3.1.2 Übersetzung von binären Assoziationen in Tabellen

Es kann keine generelle Aussage gemacht werden, ob eine Assoziation einer Tabelle entspricht oder nicht, denn dies hängt vom Typ und der Kardinalität der Assoziation und den Präferenzen des Datenbankentwicklers in Bezug auf Erweiterbarkeit, Anzahl der Tabellen und Leistungsanforderungen ab.

Jede n:m Assoziation entspricht einer eigenen Tabelle. Hierbei werden die Primärschlüssel der beiden in Beziehung stehenden Klassen und alle Verbindungsattribute Attribute der Assoziationstabelle (siehe Abbildung 3.2). Die Kombination der Attribute *KartenID* und *Stichwortpfad* ergibt den Primärschlüssel der Tabelle *abgelegtUnter*. Hierbei ist die Überwachung der referentiellen Integrität der beiden Attribute wichtig. Jedes Tupel der Tabelle *abgelegtUnter* muß eine *Karte* und ein *Stichwort* referenzieren, die in ihren eigenen Tabellen definiert werden. Die Tabellen für die Klassen *Karte* und *Stichwort* ähneln der *Zitatkarten*-Tabelle aus Abbildung 3.1.

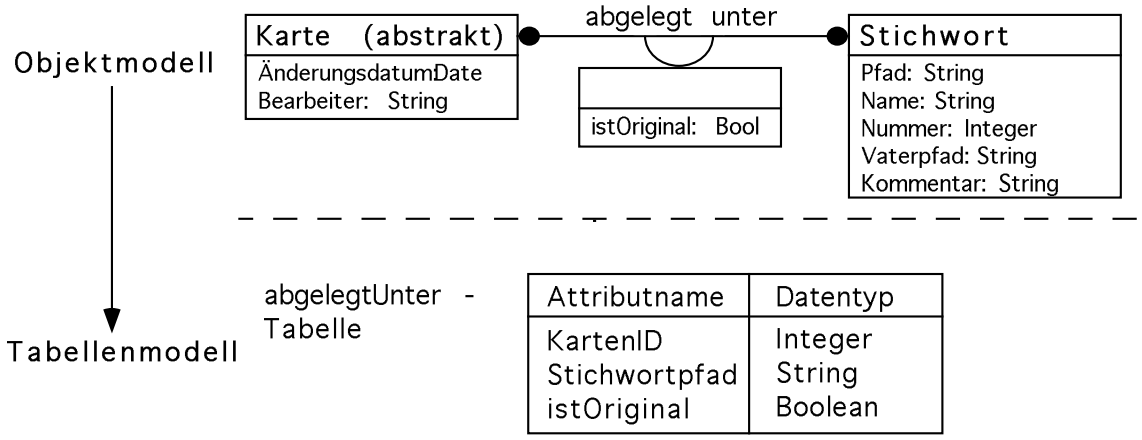


Abbildung 3.2: Konvertierung einer n:m Assoziation in eine Tabelle

Die Abbildungen 3.3 und 3.4 zeigen zwei Möglichkeiten, eine 1:n Assoziation in eine Tabelle umzuwandeln. Man kann eine separate Tabelle erzeugen oder einen Fremdschlüssel in die Tabelle für die *Bildkarten*-Klasse einbetten. Die Vorteile der Integration einer Assoziation in eine Klasse sind:

- weniger Tabellen
- schnellerer Zugriff (in diesem Fall von der *Bildkarten*-Tabelle aus)

Die Nachteile, eine Assoziation in eine Klasse zu integrieren, sind:

- *Geringere Designgenauigkeit*: Assoziationen werden zwischen voneinander unabhängigen, gleichwertigen Objekten definiert. Daher ist es normalerweise nicht angemessen, Objekte mit dem Wissen über andere Objekte zu erweitern.
- *Reduzierte Erweiterbarkeit*: Änderungen der Kardinalitäten von Beziehungen können weitreichende Auswirkungen auf das Datenbankschema haben.
- *Größere Komplexität*: Eine asymmetrische Repräsentation verkompliziert Such- und Änderungsoperationen.

Die Tabellen für die Klassen *Bildkarte* und *Künstler* aus der Abbildung 3.3 ähneln der *Zitatkarten*-Tabelle aus Abbildung 3.1.
Die Entscheidung, ob eine Assoziation in eine Klasse integriert wird oder nicht, hängt letztlich von der Anwendung ab.

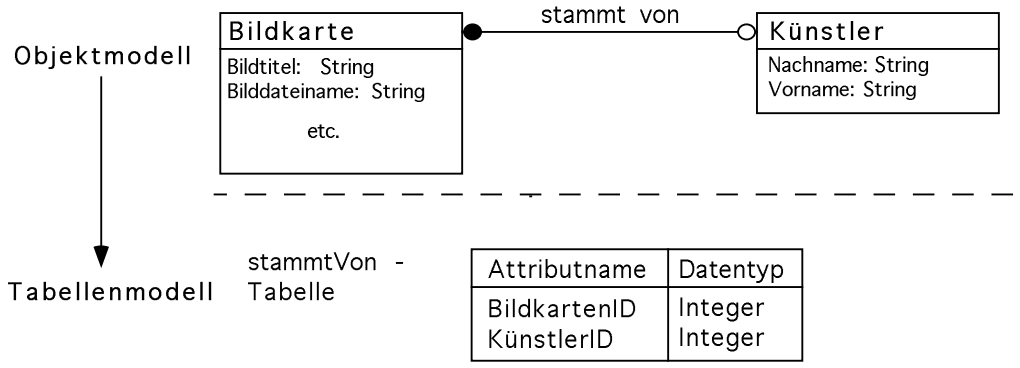


Abbildung 3.3: Konvertierung einer 1:n Assoziation in eine Tabelle (a)

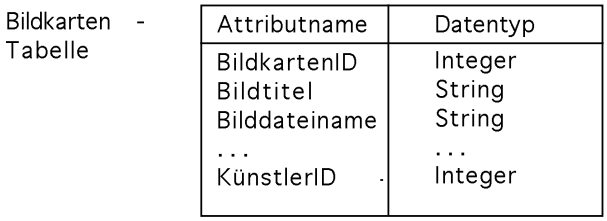


Abbildung 3.4: Integration einer 1:n Assoziation in eine Tabelle (b)

3.1.3 Übersetzung von Generalisierungsbeziehungen in Tabellen

Es gibt drei Ansätze für die Übersetzung von Generalisierungsbeziehungen in Tabellen. Als Grundlage für die Erläuterungen dieser Strategien dient Abbildung 3.5, die besagt, daß eine Instanz der abstrakten Klasse *Karte* entweder eine Instanz der Klasse *Bildkarte*, der Klasse *Zitatkarte* oder der Klasse *Literaturkarte* ist. Außerdem werden die Attribute der abstrakten Superklasse *Karte* auf ihre drei Subklassen vererbt.
Abbildung 3.6 zeigt den üblichen Ansatz. Sowohl die Superklasse als auch die Subklassen werden jeweils in eine Tabelle übertragen. Die Identität eines Objektes wird durch die gemeinsame *KartenID* sichergestellt. Dieser Ansatz ist zwar logisch klar und erweiterbar,

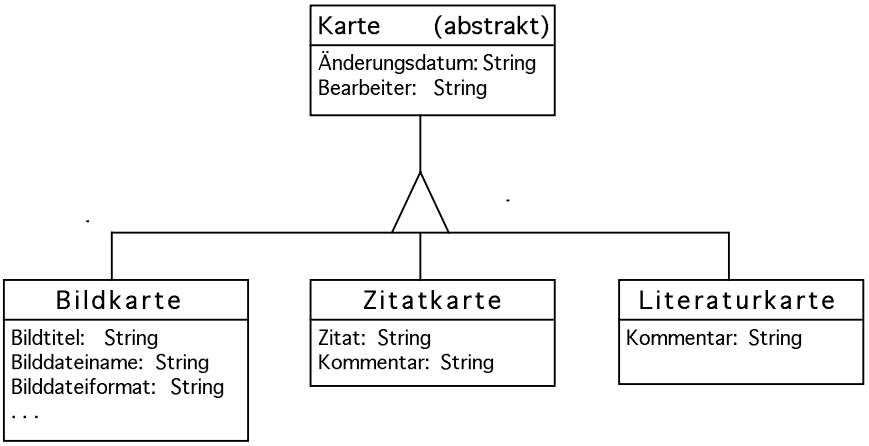


Abbildung 3.5: Objektmodell einer Generalisierung

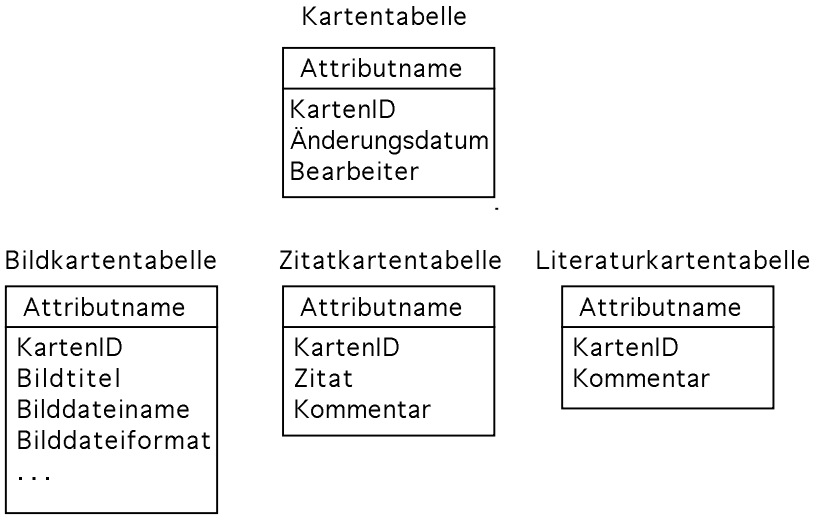


Abbildung 3.6: Tabellenmodell einer Generalisierung – Super- und Subklassen

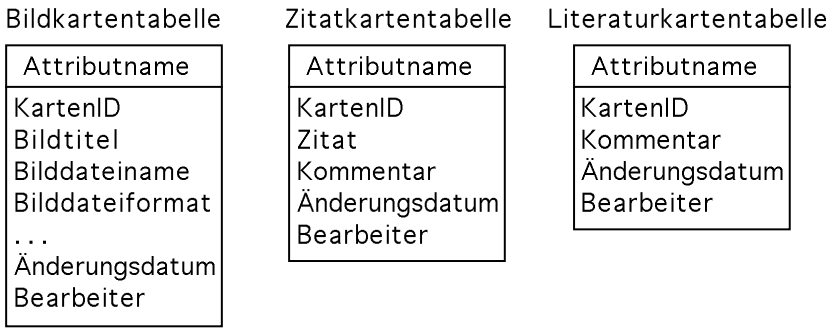


Abbildung 3.7: Tabellenmodell einer Generalisierung – mehrere Subklassen

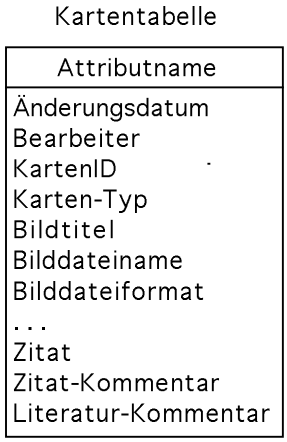


Abbildung 3.8: Tabellenmodell einer Generalisierung – eine Superklasse

erzeugt aber viele Tabellen und kann die Performanz der Datenbank bei Such- oder Änderungsoperationen beeinträchtigen.
Die Abbildungen 3.7 und 3.8 zeigen alternative Ansätze, die die Navigation zwischen Superklasse und Subklassen vermeiden und dadurch Geschwindigkeitsvorteile erbringen können. Die Variante aus Abbildung 3.7 kann sinnvoll sein, wenn die Superklasse nur eine geringe Anzahl von Attributen hat und im Anwendungsprogramm ermittelt wird, welcher Typ von Karte gesucht werden soll. Der Ansatz in Abbildung 3.8 integriert alle Subklassenattribute in die Tabelle der Superklasse. Jedes Tupel in dieser Tabelle enthält nur die Attribute, die für den jeweiligen Typ von Karte vorgesehen sind. Dieser Ansatz mag beim Bearbeiten einer *Bildkarte* noch sinnvoll sein, wenn aber eine *Literaturkarte* bearbeitet wird, enthält der Großteil der Attribute nur Nullwerte. Dieser Ansatz ist sinnvoll, wenn es nur zwei oder drei Subklassen mit wenigen Attributen gibt.

3.2 Access - ein Überblick

Access ist ein Datenbankprogrammiersystem zum Anlegen und Bearbeiten von relationalen Datenbanken unter Windows. Zu einer Access-Datenbank gehören folgende Objekte:

Tabellen: Eine Tabelle ist eine Sammlung von Daten einer Kategorie, z.B. eine Menge von Stichworten oder ein Menge von Bildkarten. Tabellen sind in Zeilen und Spalten unterteilt, wobei jede Zeile einer Tabelle genau einen aus mehreren Feldern bestehenden Datensatz enthält.

Abfragen: Abfragen stellen eine eingeschränkte Sicht auf eine Verknüpfung mehrerer Tabellen durch einen Filter dar. So können Daten nach vom Benutzer ausgewählten Kriterien aus einer oder mehreren Tabellen angezeigt werden.

Formulare: Formulare basieren auf Tabellen oder Abfragen und zeigen die darin enthaltenen Daten an. Während Tabellen und Abfragen optisch einfach als Listen dargestellt werden, entsprechen Formulare eher Karteikarten und zeigen meist nur einen Datensatz an. In Formularen können beliebige Formatierungen der Daten vorgenommen werden.

Berichte: Berichte basieren wie Formulare auf Tabellen oder Abfragen und ermöglichen ebenfalls die optisch ansprechende Gestaltung der darin enthaltenen Daten. Die angezeigten Daten können in einem Bericht nicht bearbeitet, sondern nur ausgedruckt werden.

Makros: Makros werden zur Automatisierung häufig benötigter Aktionen, wie z.B. dem Öffnen einer Tabelle, verwendet.

Module: Access enthält die Programmiersprache Access-BASIC. Access-BASIC ist eine Variante der Programmiersprache VisualBASIC (siehe Abschnitt 4.1), mit der Prozeduren und Funktionen zum Lösen spezieller Datenbankprobleme erstellt werden können. Diese Prozeduren und Funktionen werden dann in Modulen zusammengefaßt.

Diese Objekte sind über das sogannte Datenbankfenster zugänglich und werden zusammen mit den Daten in einer Datei abgelegt. Zur Unterstützung des Anwenders gibt es ein umfangreiches interaktives und kontextabhängiges Hilfesystem. Sogenannte *Assistenten* dienen der nahezu automatischen Generierung von Tabellen, Abfragen, Formularen und Berichten. Ein *Ratgeber* führt schrittweise zu Problemlösungen und gibt dem Anwender detailliert Anleitungen zur Vorgehensweise.
Access verfügt über umfassende, flexible Im- und Export-Möglichkeiten. Für den Im- und Export von Textdateien lassen sich Text- und Spaltentrennzeichen sowie das Datumsformat festlegen. Dies wurde genutzt, um große Textdatenbestände eines Künstlerindex in die Datenbank zu integrieren und somit systematisch zugänglich zu machen.

Neben dem Import bietet Access die Möglichkeit der Einbindung von Tabellen anderer Datenbanken (z.B. im dBase-Format). Dabei erscheint dem Anwender die eingebundene Tabelle, als wäre sie Bestandteil der aktuellen Datenbank; in Wirklichkeit führt Access jedoch alle Operationen auf den Originaldaten direkt aus. Der Zugriff auf Datenbestände in Datenbanken anderer Hersteller erfolgt über die ODBC-Schnittstelle (*Open Database Connectivity*). Daten aus Access-Datenbanken können über diese Schnittstelle ebenfalls von anderen Anwendungen gelesen werden.
Die Kommunikation zwischen Windows-Anwendungen erfolgt über die OLE-Schnittstelle (*Objekt Linking and Embedding*). Das bedeutet, daß Daten aus Access mit einem Mausklick in andere Windows-Anwendungen (z.B. Textverarbeitung), die ebenfalls OLE unterstützen, übernommen werden können und umgekehrt.
Die Möglichkeiten von Access in Bezug auf die Einbindung von Tabellen anderer Datenbanken und auf die Kommunikation zwischen Windows-Anwendungen waren während dieser Arbeit nicht von Belang. Access bietet mit ihnen aber eine große Offenheit gegenüber späteren Erweiterungen.

3.2.1 Der Access-Tabelleneditor

Im Zusammenhang mit dieser Arbeit waren der Tabelleneditor zur Definition der Tabellen und der Beziehungseditor (siehe Abschnitt 3.2.2) zur Definition der Relationen zwischen den Tabellen besonders wichtig. Im Tabelleneditor enthält jede Zeile des Entwurfsformulars genau eine *Felddefinition*, bestehend aus

- dem Feldnamen,
- dem Felddatentyp und
- der Feldbeschreibung.

Außerdem werden für jedes Feld die Feldeigenschaften festgelegt. Diese bestehen z.B. aus der Feldgröße, dem Feldformat, dem Standardwert, einer Gültigkeitsregel, etc. Mit diesen Feldeigenschaften können also Bedingungen für den Feldinhalt definiert werden, die später bei jeder Änderung eines Datensatzes automatisch von Access überprüft werden und bei deren Nichteinhaltung eine Fehlermeldung erzeugt wird. Im Tabelleneditor wird auch der Primärschlüssel einer Tabelle festgelegt.
In Abbildung 3.9 ist die Tabelle *Bildkarten* im Access-Tabelleneditor dargestellt. Der Primärschlüssel ist hier das Feld *ID*, erkennbar an dem kleinen Schlüsselsymbol links vom Feldnamen. In der Spalte rechts neben dem Feldnamen wird der Felddatentyp festgelegt, in diesem Fall der Typ *Zahl*. Die Länge des Feldes wird bei den Feldeigenschaften festgelegt. Eine Zahl vom Datentyp *Long Integer* ist eine ganze Zahl zwischen -2.2147.483.648 und 2.147.483.647 und belegt 4 Byte Speicherplatz. Das heißt, mit der genauen Bestimmung der Größe eines Feldes läßt sich der Speicherplatzbedarf optimieren.

Abbildung 3.9: Der Access-Tabelleneditor

3.2.2 Der Access-Beziehungseditor

Im Beziehungseditor können Relationen zwischen Tabellen grafisch definiert werden, indem man mit der Maus ein Feld einer Tabelle über ein mit diesem zu verknüpfendes Feld einer anderern Tabelle zieht und es dort fallenläßt (*drag and drop*). Es können drei Arten von Beziehungen definiert werden:

1:1-Beziehungen: Zu jedem Datensatz einer Tabelle gibt es genau einen passenden Datensatz in einer anderen Tabelle. Die Verbindung zwischen den Tabellen wird dadurch hergestellt, daß beide Tabellen den gleichen Primärschlüssel haben. 1:1-Beziehungen sind meistens nur der Übersicht wegen sinnvoll. Ebensogut wäre es möglich, beide Tabellen zu einer zusammenzufassen.

1:n-Beziehungen: Ein Datensatz in einer Tabelle ist mit einem oder mehreren Datensätzen einer zweiten Tabelle verknüpft. 1:n-Beziehungen werden hergestellt, indem der Primärschlüssel der Tabelle *1* in die Tabelle *n* eingefügt wird. 1:n-Beziehungen sind die am häufigsten auftretenden Beziehungen.

n:m-Beziehungen: Zu mehreren Datensätzen einer Tabelle gibt es mehrere Datensätze in einer anderen Tabelle. n:m-Beziehungen können in Access nicht direkt abgebildet

werden. Die Realisierung erfolgt mit Hilfe einer dritten Tabelle, die die n:m-Beziehung in zwei 1:n-Beziehungen auflöst.

Die Access-Datenbankmaschine (Jet Datenbank-Engine, siehe Abschnitt 3.3) sorgt bei Bedarf für die Überwachnung der referentiellen Integrität der Daten und bietet außerdem Funktionen zur automatischen Aktualisierungs- und Löschweitergabe an: werden also Einträge einer Tabelle geändert oder gelöscht, überträgt Access die Änderungen automatisch auf die von ihr abhängige Tabelle.
In Abbildung 3.10 ist die n:m-Beziehung zwischen den *Stichworten* und den ihnen zugeordneten *Karten* im Access-Beziehungseditor dargestellt. Die Abbildung auf ein relationales Schema erfordert die Tabelle *abgelegtunter*, in der nicht nur die beiden Primärschlüssel der beiden Tabellen enthalten sind, sondern auch das Verbindungsattribut *IstOriginal*, das festlegt, unter welchem Stichwort die physische Originalkarte im Index zur politischen Ikonographie eingeordnet ist.

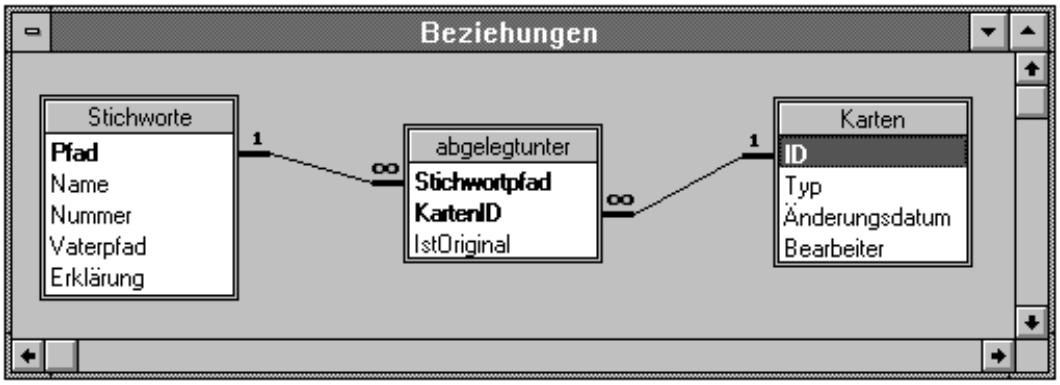


Abbildung 3.10: Der Access-Beziehungseditor

3.3 Die Jet Datenbank-Engine

Der Austausch von Daten zwischen dem Anwendungsprogramm und der Datenbank erfolgt im Falle des Protoypen für den Index zur politischen Ikonographie über die Jet Datenbank-Engine der Microsoft Corp. Sie wird in Access in der Version 2.0 verwendet, in VisualBASIC hingegen in Version 1.1. Daher können VisualBASIC-Applikationen erst nach der Installation einer Kompatibilitätsschicht auf Access-Datenbanken der Version 2.0 zugreifen. Diese Kompatibilitätsschicht ist im Internet frei verfügbar und kann mit Hilfe des FTP (*File Transfer Protocol*) auf den lokalen Rechner übertragen werden. Nach der Installation der Kompatibilitätsschicht erfolgt die Kommunikation zwischen VisualBASIC und Access problemlos. Da jedoch der Funktionsumfang der Jet Datenbank-Engine mit der Version 2.0 erweitert wurde,

gibt VisualBASIC bei Fehlern, die auf diesen Neuerungen beruhen, nur Fehlernummern aus und nicht wie sonst Hinweise auf die Ursache des aufgetretenen Fehlers.

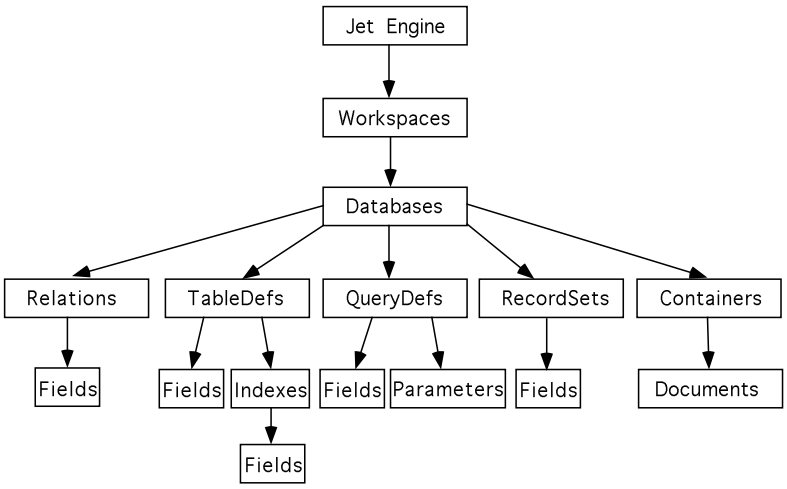


Abbildung 3.11: Hierarchie der Datenzugriffsobjekte der Jet Datenbank-Engine

Die Jet Datenbank-Engine besteht aus einer Hierarchie von unterschiedlichen Objekten. Die Pfeile in Abbildung 3.11 sind als Inklusionsbeziehung zwischen den Objekten zu verstehen, d.h. die *Jet Engine* enthält das Unterobjekt *Workspaces*, die *Workspaces* enthalten *Databases*, etc. Die Objekte im einzelnen:

Workspaces: Ein *Workspace*-Objekt ist eine Art Behälter für die aktuelle Sitzung eines Anwenders und enthält eine den Anwender identifizierende Benutzerkennung und sein Paßwort. In Mehrbenutzer-Umgebungen existieren mehrere dieser *Workspace*-Objekte.

Databases: Jedes *Database*-Objekt repräsentiert eine komplette Datenbank, inkl. Tabellen, Relationen, Abfragen, etc.

Relations: Alle Beziehungsdefinitionen zwischen den in einer Datenbank enthaltenen Tabellen werden *Relations* genannt.

TableDefs: Die Definitionen aller in einer Datenbank enthaltenen Tabellen mit ihren Feldern (*Fields*) und Indizes (*Indexes*) sind in dem Objekt *TableDefs* zusammengefaßt.

QueryDefs: Jede *QueryDef* ist die Definition einer in der Datenbank enthaltenen Abfrage.

RecordSets: *RecordSet*-Objekte enthalten Datensätze, die auf Tabellen, Abfragen oder SQL-Anweisungen basieren. Diese Datensätze können editiert und gelöscht werden; auch das Hinzufügen weiterer Datensätze ist möglich.

Containers: Ein *Container* enthält die Beschreibungen aller Objekte in einer Datenbank.

Das wichtigste im Zusammenhang mit dieser Arbeit ist die Steuerung des Zugriffs auf Datensätze in Tabellen (*RecordSets*), deren Felder (*Fields*) und die Beziehungen zwischen den Tabellen (*Relations*). Mit Hilfe von *QueryDefs* lassen sich auch Anfragen in der Datenbank selbst abspeichern. Allerdings könnte beim Aufruf solcher Abfragen vom Anwendungsprogramm aus die Transparenz des Programmablaufes für den Programmierer verloren gehen. Außerdem können Anfragen an die Datenbank teilweise erst zur Laufzeit des Programms mit Hilfe von Benutzereingaben formuliert werden, z. B. wenn es darum geht, nur die Bilder zu einem bestimmten Stichwort anzuzeigen. Aus diesem Grund wurden die *QueryDefs* nicht genutzt und die Anfragen an die Datenbank mit Hilfe von SQL im Anwendungsprogramm formuliert.

4. Die Benutzerschnittstelle

Die Mitarbeiter des Index zur politischen Ikonographie hatten bis zum Zeitpunkt der Demonstration des Prototyps noch nicht mit einer Benutzerschnittstelle mit Fenstertechnik gearbeitet. Sie konnten jedoch mit einem Textverarbeitungsprogramm unter dem Betriebssytem MS-DOS umgehen und waren auch bereit, sich in eine für sie neue Bedientechnik einzuarbeiten.
Die Programmiersprache VisualBASIC für Windows verfügt über Eigenschaften, die sie für diesen Prototyp geeignet erscheinen ließen (siehe Abschnitt 4.1). Mit der Jet Datenbank-Engine verfügt VisualBASIC über eine direkte Schnittstelle zu dem Datenbankprogrammiersystem Access, die in Abschnitt 3.3 vorgestellt wurde.
Eine weitere Tatsache, die für VisualBASIC sprach, war die große internationale Benutzergemeinde. So gibt es umfangreiche Shareware-Archive mit Erweiterungen und Programmierbeispielen. Dies sollte sich während der Arbeit als große Hilfe erweisen, da VisualBASIC erstmalig eingesetzt wurde und dementsprechend relativ wenige Datailkenntnisse vorhanden waren.
In Abschnitt 4.2 wird anhand von einigen Bildschirmabbildungen des laufenden Programms die Benutzerschnittstelle des Prototypen für den Index zur politischen Ikonographie vorgestellt.

4.1 VisualBASIC - ein Überblick

VisualBASIC besitzt, wie in Abbildung 4.1 dargestellt, eine integrierte Entwicklungsumgebung (*Integrated Development Environment*, IDE), die einen Maskengenerator (*Form Edit*), einen Quelltext-Editor (*Source Edit*), eine interaktive und kontextsensitive Hilfe (*Online Help*), eine Projektverwaltung (*Projects*), einen Interpreter, einen Übersetzer (*Compiler*), einen Debugger und einen Werkzeugkasten (*Toolbox*), der sich mit externen Werkzeugen (*Custom Controls*) ergänzen läßt, umfaßt [Maslo, Dittrich 93; Müßig 94].
Programme können in der Entwicklungsumgebung editiert, übersetzt, getestet und ausgeführt werden. Ein ohne die Entwicklungsumgebung lauffähiges Programm kann mit Hilfe des Übersetzers erzeugt werden. Dieses benötigt allerdings zur Laufzeit zusätzlich ein Laufzeitsystem in Form einer dynamischen Bibliothek (*Dynamic Link Library*, DLL).
Im Gegensatz zu herkömmlichen BASIC-Dialekten gibt es in VisualBASIC kein Hauptprogramm. Die Programmsteuerung erfolgt über Ereignisse, die im Windows-System vom

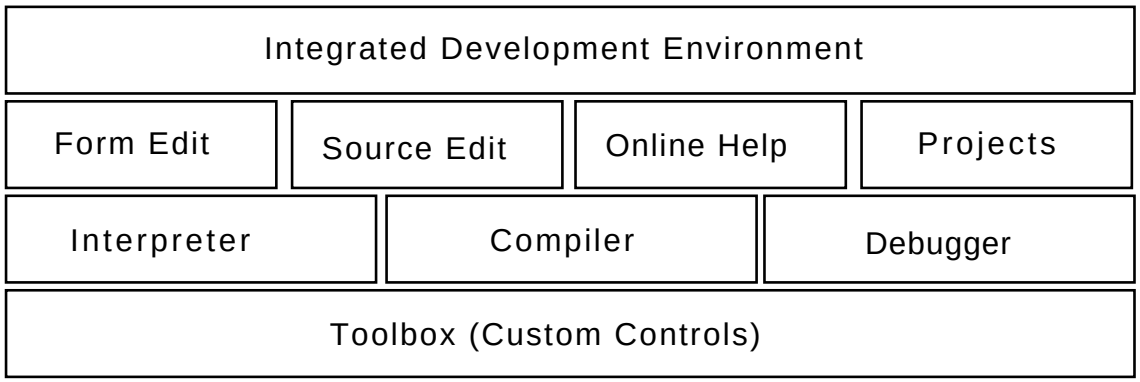


Abbildung 4.1: Komponenten der VisualBASIC-Entwicklungsumgebung

Benutzer oder automatisch (zB. zeitabhängig) ausgelöst werden. Man spricht daher bei VisualBASIC auch von ereignisorientierter Programmierung. Die Ursachen für die Unterschiede zu anderen BASIC-Programmiersystemen sind zum großen Teil in Windows selbst begründet. Programmme unter Windows können alle Windows-Schnittstellenkomponenten nutzen und auch auf Systemfunktionen, die sogenannte WindowsAPI (*Application Programming Interface*), zugreifen. Der Standardfunktionsumfang kann mit Hilfe von DLLs und den VisualBASIC-Erweiterungen (*VisualBASIC-Extension*, VBX) ergänzt werden. Diese Erweiterbarkeit hat zu einem großen Angebot von frei verfügbaren und kommerziellen DLLs und VBXs für die unterschiedlichsten Anwendungen geführt. Mit VisualBASIC selbst lassen sich keine DLLs, bzw. VBXs erzeugen. Neben der Einbindung von DLLs stehen Windows Funktionen zur vereinfachten Grafikprogrammierung (*Graphics Device Interface*, GDI), für den dynamischen Datenaustausch (*Dynamic Data Exchange*, DDE) und für die Verknüpfung und Einbettung von Objekten (*Object Linking and Embedding*, OLE) zur Verfügung.

Anwendungsentwicklung mit VisualBASIC

Nachdem mit den Konzepten aus Abschnitt 3.1 die relationale Datenbankstruktur in Access aufgebaut wurde, erfolgte mit VisualBASIC die Gestaltung der Benutzeroberfläche und die Steuerung der Zugriffe auf die Datenbank. Hierbei kann das Vorgehen in mehrere Schritte zerlegt werden [Mic 93]:

Erzeugen einer Oberfläche: Man beginnt mit einem leeren Fenster, in VisualBASIC Formular (*form*) genannt. Mit dem Maskengenerator wird das optische Erscheinungsbild einer Anwendung entworfen, indem die gewünschten Schnittstellenkomponenten (zB. Text- oder Listenfelder, Befehlsschaltflächen, Auswahlboxen, etc.) aus dem Werkzeugkasten mit Hilfe der Maus in das gewünschte Formular gezogen, dort positioniert (*drag and drop*) und in der Größe angepaßt werden. Jedem Formular kann ein individuelles Menü mit Hilfe eines speziellen Menüentwurfsfensters zugeordnet werden.

Festlegen von Eigenschaften: Nach der Definition der Benutzerschnittstelle werden allen Objekten in Abhängigkeit von ihrer späteren Aufgabe individuelle Eigenschaften zugewiesen. Zu diesen Eigenschaften gehören z. B. Benennung des Objektes, Vorder- und Hintergrundfarben, Beschriftungen, etc. Diese Eigenschaften können einerseits durch einen Eintrag in einer Eigenschaften-Liste, die jedes Objekt besitzt, geändert werden; andererseits ist es auch möglich, sie durch Programmkode zur Laufzeit der Anwendung zu ändern.

Schreiben von Kode: In Modulen, Prozeduren, Funktionen und Anweisungen wird nun die Funktionalität der Oberfläche mit Hilfe von VisualBASIC-Kode bestimmt. Hierbei werden die Objekte der Oberfläche durch ihre Namen referenziert, die als Eigenschaft festgelegt werden. Außerdem werden im Kode auch die Anfragen an die Datenbank mit Hilfe von SQL formuliert.

4.2 Der Protoyp

Im Laufe der Arbeit mit den Kunsthistorikern hat sich gezeigt, daß die Verständigung zwischen Softwareentwickler und späteren Anwendern nur bis zu einem gewissen Grad mit Hilfe von Beschreibungen bzw. Objektdiagrammen sinnvoll ist. Gerade bei den Anwendern tritt relativ schnell eine Ermüdung ein, da für sie das Arbeitsgebiet, das sie dem Softwareentwickler nahebringen müssen, selbstverständlich ist und sie sich ihres impliziten Wissens darüber nicht bewußt sind. Ohne dieses Wissen aber hat der Softwareentwickler Schwierigkeiten, sich in die Probleme der Anwender hineinzudenken.

Um diese Kommunikationsprobleme zu beseitigen und den Kunsthistorikern eine erste Idee eines computergestützten Index zur politischen Ikonographie zu geben, lag es nahe, einen Prototypen zu entwickeln. Der Protoyp wurde mit Hilfe des Datenbankprogrammiersystems Access (siehe Abschnitt 3.2 und der Programmiersprache VisualBASIC (siehe Abschnitt 4.1) der Microsoft Corp. entwickelt. Während Access dazu genutzt wurde, das relationale Datenbankschema (siehe Abschnitt 3.1) aufzubauen, ist mit Hilfe von VisualBASIC die Benutzerschnittstelle implementiert worden. Die Kommunikation zwischen der Datenbank und dem Anwendungsprogramm erfolgt mit Hilfe von SQL (*Standard Query Language*) über die Jet Datenbank-Engine (siehe Abschnitt 3.3).

Hier soll nun die Benutzerschnittstelle des Prototypen für den Index zur politischen Ikonographie vorgestellt werden. Die Anwendung wurde als sogenannte MDI-Anwendung (MDI = *Multiple Document Interface*) realisiert. Innerhalb des Arbeitsbereiches eines MDI-Elternfensters können Kindfenster geöffnet, positioniert und bearbeitet werden. Kindfenster können nicht aus dem Arbeitsbereich des MDI-Elternfensters herausbewegt werden, sind also immer an jenes gebunden. In Abbildung 4.2 enthält das MDI-Elternfenster *PolitIcon* das Kindfenster *Stichwort-Index*.

4.2.1 Der Stichwort-Index

Der Stichwort-Index im Index zur politischen Ikonographie besteht aus einer Reihe von Hauptstichworten, denen wiederum Unterstichworte zugeordnet sind. In Abbildung 4.2 ist die Repräsentation dieses Index im Prototyp *PolitIcon* zu sehen.
In der ersten Ebene sind die Hauptstichwörter mit ihren jeweiligen Nummern erkennbar (z.B. *65 Begegnungen* oder *180 Frieden*). Mit einem Mausklick auf das Pluszeichen vor der Nummer des Hauptstichwortes läßt sich dieses wie ein Ordner öffnen und es werden – wie im Fall von *180 Frieden* – die Unterstichworte mit ihren jeweiligen Nummern sichtbar. Die Stichworthierarchie im Index zur politischen Ikonographie ist derzeit nur zweistufig. Im Protoyp ist die Verschachtelungstiefe theoretisch nicht begrenzt, so daß weitere Verfeinerungen des Index ohne Probleme möglich sind.

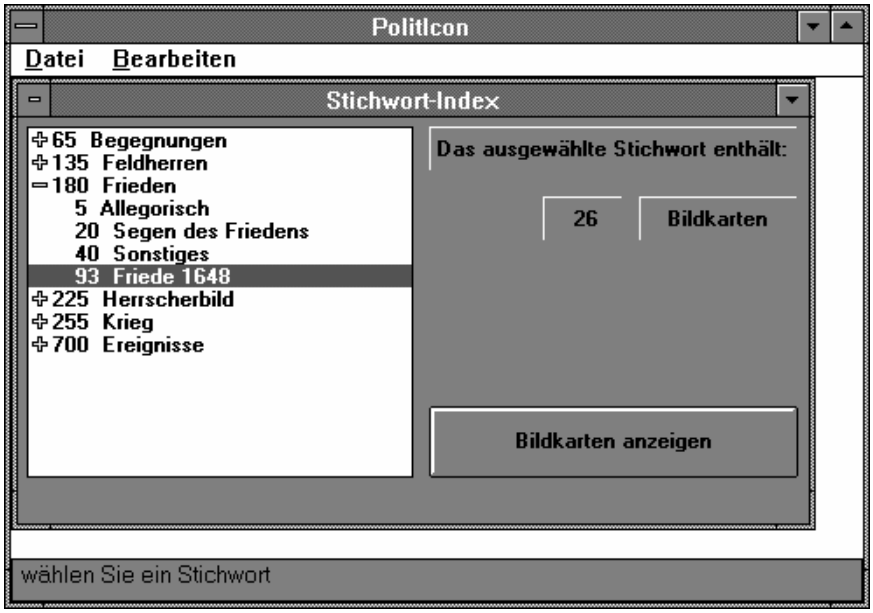


Abbildung 4.2: PolitIcon: Stichwort-Index

Klickt der Benutzer auf ein Unterstichwort, wird automatisch die Anzahl der unter diesem Stichwort abgelegten Bildkarten bestimmt und angezeigt, um dem Benutzer eine Vorstellung vom Umfang des Datenbestandes zu dem jeweiligen Thema zu geben. Mit einem Klick auf den Befehlsknopf (*command button*) *Bildkarten anzeigen* wird ein anderes Fenster geladen, in dem die Bilder zu dem ausgewählten Unterstichwort im Kleinformat (75 dpi, 16 Farben) in einer Tabelle angezeigt werden (siehe Abbildung 4.3). Als Fensterüberschrift für die Bildauswahl dient zur Orientierung der Pfad des ausgewählten Unterstichwortes.
Mit Hilfe von vertikalen und horizontalen Rollbalken (*scrollbars*) kann der Benutzer alle Bilder, die dem Stichwort zugeordnet sind, sichtbar machen. Mit einem Klick auf ein Bild

Abbildung 4.3: PolitIcon: Bildauswahl

Abbildung 4.4: PolitIcon: Bilddetails

wählt der Benutzer dieses aus und bekommt die Bilddetails (siehe Abbildung 4.4) angezeigt, die das Bild und die dazugehörigen textuellen Informationen enthalten. Der Titel des ausgewählten Bildes dient als Fensterüberschrift. Die textuellen Informationen zum Bild setzen sich zusammen aus:

- Bild-ID
- Bildtitel
- Bildentstehungszeit
- Auftragsnummer der Fotostelle
- Kommentare
- Künstler (Referenz auf Künstler-Index, siehe Abschnitt 4.2.2)

Außerdem ist an dem Feld *Original* zu erkennen, ob die jeweilige Karte unter diesem Stichwort als Original oder als Kopie abgelegt wurde. Mit dieser Information und der *Bild-ID* ist es möglich, die Original-Bildkarte im physischen Index zur politischen Ikonographie zu finden.

Abbildung 4.5: PolitIcon: Bild im Großformat

Die Bilddetails können jederzeit geändert, ergänzt und mit Hilfe des Befehlsknopfes *Änderungen speichern* abgespeichert werden. Mit dem Befehlsknopf *Schließen* wird das Fenster

geschlossen, und die Daten bleiben unverändert. Die große Version des Bildes (300 dpi, 256 Farben) läßt sich mit einem weiteren Mausklick auf das Bild im Bilddetailfenster aufrufen (siehe Abbildung 4.5).
Wenn dem Benutzer nun bei dem Bild im Großformat Details auffallen, die er kommentieren möchte, kann er mit einem Mausklick auf das Bild wieder das Bilddetailfenster (siehe Abbildung 4.4) öffnen und seine Anmerkungen dort eintragen.

4.2.2 Der Künstler-Index

Bei dem Künstler-Index des Protoypen *PolitIcon* handelt es sich um einen Auszug aus dem Datenbestand des Künstler-Lexikons des Bildarchiv Foto Marburg. In der *Marburger Bibliothek maschinell nutzbarer Veröffentlichungen* werden ausgewählte Datenbestände als sequentielle, unformatierte Datei angeboten. Neben dem Künstler-Lexikon stehen verschiedene Thesauri (z.B. Thesaurus antiker keramischer Gefäße, Thesaurus der Buchmalerei) und Lexika (z.B. Historisch-geographisches Lexikon, Ikonographisches Lexikon) zur Verfügung. Diese Datenbestände sind kostenlos verfügbar und dienen der Verbesserung der wissenschaftlichen Zusammerarbeit zwischen Kulturhistorikern [IBM 89].

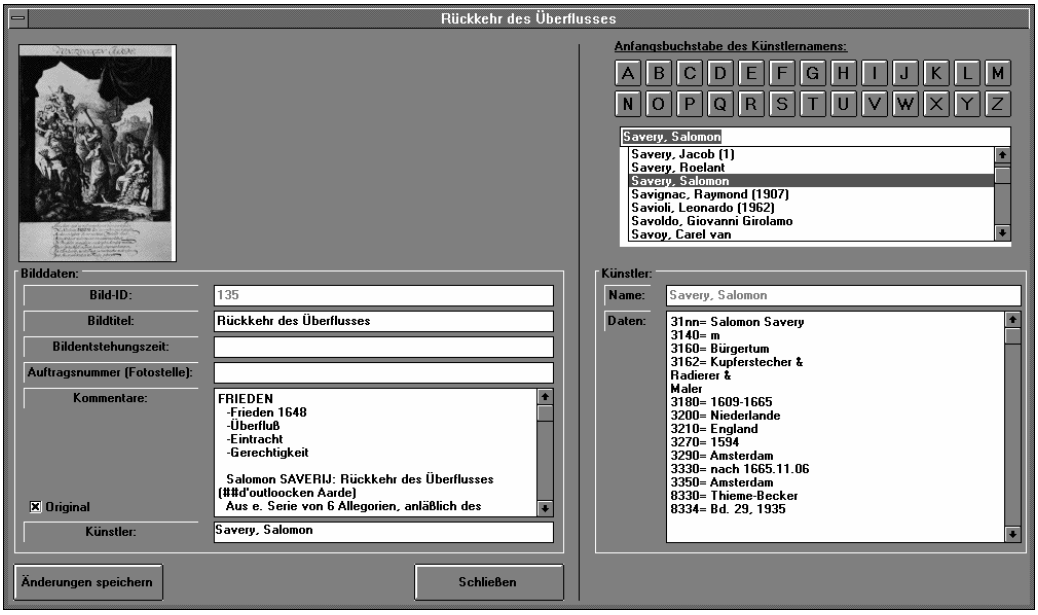


Abbildung 4.6: PolitIcon: Künstler-Index

Der Künstler-Index ist mit einem Mausklick auf das Feld *Künstler* im Bilddetails-Fenster aufrufbar (siehe Abbildung 4.6). Der Benutzer gibt dann den Anfangsbuchstaben des Künstlernamens an. Im Listenfeld der Kombinationsbox erscheinen sämtliche Künstler mit dem gewählten Anfangsbuchstaben. Wählt der Benutzer einen Namen aus dem Listenfeld aus,

wird dieser invertiert dargestellt und im rechten unteren Teil des Fensters werden dessen Daten in einem Listenfeld angezeigt. Die Zahlen in diesem Listenfeld (z.B. 31nn, 3140, etc.) stehen für bestimmte Felder im Künstler-Lexikon. So steht *31nn* für den Namen des Künstlers und *3140* für sein Geschlecht. Gleichzeitig mit der Auswahl eines Künstlers wird dieser im Feld *Künstler* des Bilddetailfensters eingetragen. Mit einem Mausklick auf den Befehlsknopf *Änderungen speichern* wird die Beziehung zwischen Künstler und Bildkarte gespeichert.
Mit der Einbindung des Künstler-Index in *PolitIcon* sollte gezeigt werden, daß es ohne weiteres möglich ist, externe Datenbestände in die Datenbank einzubinden. In einem endgültigen Programm wäre es natürlich wünschenswert, die aus dem Marburger Künstler-Index stammenden Nummern durch aussagekräftige Feldbezeichungen zu ersetzen. Ausserdem sollte es dann möglich sein, daß der Benutzer einfach durch eintippen eines beliebigen Namens in das Textfeld der Kombinationsbox des Künstler-Index den gewünschten Künstler sucht und daß er nicht zuerst den Anfangsbuchstaben angeben muß. Dies war allerdings bei diesem Prototyp noch nicht möglich, da VisualBASIC nur eine bestimmte Anzahl von Einträgen innerhalb eines Listenfeldes verwalten kann und diese von den Einträgen im Künstler-Lexikon bereits überschritten wurde.

5. Zusammenfassung und Ausblick

In diesem abschließenden Kapitel soll auf meine Erfahrungen mit den benutzten Entwurfswerkzeugen näher eingegangen werden. Hierbei werden neben OMT insbesondere das Datenbankprogrammiersystem Access und die Programmiersprache VisualBASIC näher betrachtet. Das Ende des Kapitels bildet eine kritische Beurteilung meiner Arbeit, sowie eine Aufstellung möglicher sinnvoller Erweiterungen und Verbesserungen des Prototypen *PolitIcon*.

5.1 Erfahrungen mit OMT

OMT-Objektmodelle dienten in dieser Arbeit zur Visualisierung der statischen Struktur des Index zur politischen Ikonographie. Sie waren Grundlage der Diskussionen mit den Kunsthistorikern und wurden von diesen auch akzeptiert und verstanden. Das ursprünglich geplante Bearbeiten aller vier OMT-Phasen mußte aus Termingründen entfallen. Außerdem war bei den Kunsthistorikern nach ca. 8 Wochen Diskussion (insgesamt fünf zweistündige Treffen) über Problembeschreibungen und OMT-Objektdiagramme ein Nachlassen des Engagements zu bemerken. Dies kann auch daran gelegen haben, daß diese Treffen mit entsprechender Vor- und Nachbereitung von den Kunsthistorikern zusätzlich zu ihrem normalen Arbeitspensum bewältigt werden mußten.
Kritisch zu beurteilen ist in diesem Rahmen der Einsatz des CASE-Werkzeuges StP OMT, da hier der Einarbeitungsaufwand und der tatsächliche Nutzen in einem ungünstigen Verhältnis zueinander standen. In größeren Projekten, bei denen alle vier Phasen der OMT-Methode durchlaufen werden und an denen mehrere Entwickler gleichzeitig arbeiten, kann dieses Werkzeug jedoch sehr nützlich sein.

5.2 Erfahrungen mit Access

Die Benutzung des Datenbankprogrammiersystems Access hat den Entwurf des relationalen Datenbankschemas wesentlich vereinfacht, da sich die Tabellen und ihre Beziehungen zueinander mit relativ geringem Aufwand erstellen ließen. Die einfach zu benutzenden (teilweise grafischen) Editoren beschleunigten die Entwicklung um ein Vielfaches.
Die Einbindung des Künstler-Lexikons des Bildarchiv Foto Marburg wurde durch die Fähigkeit von Access zum Import von Textdateien in unkomplizierter Weise möglich.

In dieser Arbeit wurde nur ein geringer Teil der von Access zur Verfügung gestellten Funktionalität genutzt. Nicht benutzt wurden z.B. die Funktionen zum Entwurf von Benutzerschnittstellen. Mit diesen Funktionen hätte der Prototyp *PolitIcon* auch ohne Zuhilfenahme der Programmiersprache VisualBASIC realisiert werden können. In dieser Arbeit sollte jedoch gerade auch das Zusammenspiel dieser beiden Komponenten erprobt werden, da dieses dem Hersteller Microsoft als Marketing-Argument dient. Nach kurzer Einarbeitungszeit und der Beschaffung der sogenannten Kompatibilitätsschicht, die für die Kommunikation zwischen Access Version 2.0 und VisualBASIC Version 3.0 notwendig ist, erfolgte die Zusammenarbeit nahezu reibungslos. Negativ zu bemerken sind hier nur die zum Teil nicht näher erläuterten Fehlermeldungen beim Datenbankzugriff von VisualBASIC aus.

5.3 Erfahrungen mit VisualBASIC

Das Erstellen einer grafischen Benutzeroberfläche mit VisualBASIC ist sehr einfach, da sich die gewünschten Schnittstellenkomponenten (z. B. Text- oder Listenfelder, Befehlsschaltflächen, Auswahlboxen, etc.) einfach mit der Maus positionieren und in der Größe anpassen lassen.
Prozeduren und Funktionen werden in VisualBASIC in Modulen zusammengefaßt. Das Aufrufen solcher Unterprogramme erfolgt aber ohne Bezugnahme auf die jeweiligen Modulnamen. Dadurch wird beim Vorhandensein von vielen Modulen das Verfolgen eines Programmlaufes sehr zeitaufwendig, wenn nicht gar unmöglich.
Ein weiteres Problem von VisualBASIC-Programmen ist die im Vergleich zu in der Programmiersprache C geschriebenen Anwendungen niedrige Performanz, d.h. VisualBASIC-Programme sind relativ langsam.
Außerdem war es im Rahmen des Künstler-Index nicht möglich, beliebig viele Datensätze in einem Listenfeld anzuzeigen, da die Speicherverwaltung von VisualBASIC nur einen Speicherbereich von 64 KB (entspricht 5440 Einträgen in einer Liste) pro Listenfeld bereitstellt.

5.4 Der Prototyp – eine Abschlußbetrachtung

Zusammenfassend kann gesagt werden, daß die bestehende Form des Prototypen seine Aufgabe erfüllt. Es ist hier festzuhalten, daß es sich bei *PolitIcon* um einen Prototypen im Sinne der Definition aus Abschnitt 1.2 handelt und nicht um ein dem Benutzer zu übergebendes Endprodukt. Daher ist es ganz natürlich, daß nicht alle in den Problembeschreibungen erarbeiteten Anforderungen realisiert wurden, da dies den Rahmen meiner Arbeit weit überschritten hätte. Aufgrund der Tatsache, daß sich die Kunsthistoriker während der ca. 3 Monate dauernden Programmierung für den Einsatz einer kommerziellen Bilddatenbanksoftware entschieden haben, fand leider keine Diskussion mit den vorgesehenen Anwendern über den vorliegenden Prototyp statt.
Nachfolgend sind eine Reihe von möglichen Erweiterungen und Änderungen aufgeführt, die den Umgang mit dem Datenbanksystem effizienter gestalten könnten und zugleich den Bedienungskomfort erhöhen würden:

- Eine erweiterte Zugreifbarkeit des Bildmaterials über mehrere Indizes ist erforderlich. Das heißt, es sollte möglich sein, das Bildmaterial nicht nur über die Haupt- und Unterstichworte zu erschließen, sondern z.B. auch über den Künstlernamen, das Bildentstehungsjahr oder den Bildtitel. Die dabei entstehenden Bildzusammenstellungen würden neue Perspektiven auf das Forschungsmaterial erlauben.

- Ein Problem im jetzigen Prototyp ist die Neuverschlagwortung von Bildern, da diese nur für jeweils ein Bild zur Zeit erfolgen kann. Erforderlich wäre hier eine Funktion mit entsprechender Benutzeroberfläche, die es dem Benutzer erlaubt, mehrere Bilder gleichzeitig einem oder mehreren Stichworten zuzuordnen.

- Die Performanz einer VisualBASIC-Anwendung, die letztlich mehr als 200.000 Datensätze samt Bilddateien verwaltet, wird für professionelle Ansprüche nicht ausreichend sein. Daher ist zu überlegen, ob ein Endprodukt nicht doch mit einer Programmiersprache wie C oder C++ realisiert werden sollte.

- Kunsthistoriker arbeiten vielfach so, daß sie die von ihnen unter verschiedenen Kriterien zusammengestellten Bildkombinationen auf dem Schreibtisch nebeneinander ausbreiten, um sie dann vergleichend zu untersuchen. Diese Arbeitsweise sollte auch von einer EDV-Lösung unterstützt werden. Dies könnte so realisiert werden, daß jeder Benutzer Einzelbilder markieren, in seiner individuellen Bildmappe sammeln und später diese neue Zusammenstellung betrachten kann.

Während die o.g. Erweiterungen und Änderungen sehr auf eine Abbildung des derzeitigen Arbeitsplatzes der Kunsthistoriker hinzielen, sind darüber hinaus noch viel weitergehende Ansätze denkbar. So wären z.B. computergestützte Werkzeuge zum interaktiven Entwurf von kunsthistorischen Präsentationen oder Ausstellungen denkbar, die auf einem digitalen Index zur politischen Ikonographie basieren. Außerdem kommt der weltweiten Vernetzung von Computern in letzer Zeit immer größere Bedeutung zu, so daß bei einer Vernetzung der digitalen Bildbestände in aller Welt eine viel umfassendere und vielschichtigere Forschung betrieben werden könnte als es derzeit der Fall ist. Auch die Einbindung und die Verknüpfung weiterer Forschungsmaterialien (Texte, Filme, Audiosequenzen, Hypertext- und Hypermediadokumente) bietet noch viel Raum für weitere Forschungsaktivitäten.
Bei allen diesen Visionen würden sich allerdings auch die Arbeitsabläufe der Kunsthistoriker entscheidend verändern und da gerade im Bereich Kunst und Kultur die Schwellenangst vor Computern sehr groß ist, können solche Ideen nur in enger Zusammenarbeit mit den Betroffenen ausgearbeitet und realisiert werden.

A. Anforderungsdefinitionen

Die erste Anforderungsdefinition in Abschnitt A.1 stammt von Mitarbeitern der politischen Ikonographie und wird hier nur auszugsweise wiedergegeben. Die zweite Anforderungsdefinition in Abschnitt A.2 basiert auf der ersten und ist im Laufe der Gespräche mit den Kunsthistorikern entstanden.

A.1 Kooperationsperspektiven zwischen dem Index zur politischen Ikonographie und dem Fachbereich Informatik an der Universität Hamburg

A.1.1 Beschreibung des Bestandes

Der Index zur politischen Ikonographie ist eine Sammlung von rund 200.000 Photographien und Abbildungen, die auf Karten in Postkartenformat aufgeklebt und nach etwa hundert Hauptstichworten geordnet sind. So sind z.B. unter dem Stichwort FRIEDEN etwa 1.300 Bildkarten versammelt, die wiederum Unterstichworten zugeordnet sind, so etwa 400 Bildkarten dem Unterstichwort *Allegorien*, etwa 300 Bildkarten dem Unterstichwort *Friedensschlüsse*. Entsprechend befinden sich unter dem Stichwort KRIEG etwa 3000 Bildkarten, die wiederum auf Unterstichworte wie *Kriegerdenkmale, Schlachten, Sieg, Soldaten* usw. verteilt sind. Jedes Stichwort hat eine arabische Nummer als Kennziffer, so 180 Frieden, 180.18 Propaganda, 255 Krieg, 255.15 Feindbilder.
Die Bildkarten enthalten recht heterogene Informationen. Da ist zunächst das Bild. Dieses dient hier nicht ästhetischen, formanalytischen oder farbtheoretischen Zwecken, sondern ausschließlich ikonographischen Zwecken, d.h. es muß lediglich die auf Bildern gezeigten Gegenstände und Symbole erkennbar machen. Das Bild ist in den meisten Fällen eine Photographie im Format bis zu 10cm x 7cm. Sie ist aufgeklebt auf einen Zettel Normalpapier im Format 14,5cm x 10,5cm. Die weitaus meisten Photos sind Schwarzweißaufnahmen, zum großen Teil nach Abbildungen in Publikationen, Büchern und Originalphotographien. Die Qualität der Reproduktionen hat mit der automatischen Herstellung etwas nachgelassen, ist aber in den meisten Fällen recht zufriedenstellend und für ikonographische Belange vollkommen ausreichend. Photokopien erfüllen den ikonographischen Informationsbedarf nur unvollkommen und tauchen deshalb in dem Index nur selten auf. Postkarten erfüllen diesen Bedarf ideal, doch sind deren Hersteller selten an politischen Motiven interessiert; doch

enthält das Stichwort STADT die Postkartensammlung des bedeutenden Städtehistorikers Erich Kayser. Die Postkarten sich oft farbig, ebenso zahlreiche aus Zeitschriften, Prospekten und anderen Materialbereichen ausgeschnittene und aufgeklebte Abbildungen. Schätzungsweise sind von den 200.000 Bildern höchstens 2% farbig.

Neben den aufgeklebten Bildern enthalten die Bildkarten TEXTE. Diese sind nicht streng standardisiert und fast durchgehend handschriftlich. Doch in der Regel trägt jede Karte oben links oder rechts zumeist in Druckschrift den Namen des Hauptstichwortes und der ihm zugeordneten Kennziffer. Außerdem wird der Künstler mit Name und Vorname genannt. Die wichtigste Information ist der Bildtitel, da hier die vor allem interessierende politische Information meistens mitenthalten ist. Es folgt die Entstehungszeit des Bildes, eine ebenfalls wichtige Information, da etwa ein Gemälde vom Jahre 1881 ein Ereignis vom Jahre 1648 wiedergeben kann. Für weitere Recherchen ist auch der Aufbewahrungsort des Bildes wichtig, da oft Museumskataloge weitere Recherchen erleichtern. (Wenn diese Daten mitphotographiert und lesbar sind, wurden sie nicht nochmals aufgeschrieben.)

Schließlich ist angegeben die Literaturstelle, der die Abbildung entnommen ist. Dieser Nachweis verweist in der Regel auf den letzten Forschungsstand. Nicht selten kommt ein Bild öfters vor, da es an verschiedenen Stellen erwähnt, besprochen und gedeutet wurde. Das heißt: tendenziell dokumentiert das Stichwort den Bild- und Forschungsbestand zur Sache: So findet man unter Friedensallegorien nicht nur die entsprechenden Bilder, sondern auch die Hinweise auf die Literatur, in der sie besprochen und abgebildet sind. Dies kann auch dadurch erweitert werden, daß ZETTEL mit Literaturangaben eingestellt sind oder Zettel mit Zitaten aus Literatur oder Quellen, die geeignet sind, den Gehalt des Stichwortes auszufüllen.

Die Bildkarten enthalten keine Querverweise: Ich erfahre also z.B. unter dem Stichwort 465.25 Todesformen nicht, daß unter dem Stichwort 225.215 HERRSCHER/*Tod des Herrschers* ebenfalls Erhängungen vorkamen oder daß unter 43 Attentate wiederum sowohl Todesformen wie auch Herrschertode vorkommen können. Hierauf vor allem beziehen sich die Erwartungen und Hoffnungen einer digitalen Erfassung des Materials.

Eine zweite Abteilung des Archivs zur politischen Ikongraphie ist die FORSCHUNGSLITERATUR ZUR POLITISCHEN IKONOGRAPHIE. Die Abteilung besteht aus einer Sammlung von Fotokopien und Büchern zur politischen Ikonographie. Das Material ist in derzeit 110 Bänden gebunden bereitgestellt, wobei jedem Band ein Inhaltsverzeichnis vorangestellt ist. Große Teile der Aufsätze sind in den Bildindex eingearbeitet. Der zweite Komplex dieser Abteilung besteht aus Büchern (derzeit etwa 10.000 Stück). Diese werden nach den Stichworten des Bildindex aufgestellt und tragen auch als Signatur deren Kennziffern: So kann man, wenn man im Bildindex 255.63 *Kriegsspielzeug* durchsieht, danach in der Buchabteilung ebenfalls unter 255.63 nachsehen, ob es Bücher zum Thema gibt. Der ganze Bestand an fotokopiertem Material ist auf Disketten gespeichert. Die Bücher werden über das Programm Allegro erschlossen. Wünschbar wäre, daß diese Daten systematisch in den Bildindex eingehen könnten.

A.1.2 Erwartungen an eine Digitalisierung des Index

- Jedes Bild sollte unter verschiedenen ikonographischen Stichworten und den PI-Nummern abfragbar sein. Beispiel: Kriegspostkarte mit Landespersonifikation *Italia* sollte unter Krieg, Personifikation von *Italia*, aber auch nach der Jahreszahl (z.B. '1914') auf dem Bildschirm abfragbar und aufrufbar sein.

- Verweise und Hinweise auf mögliche Vergleichsbeispiele unter anderen PI-Schlagworten sollen per Text **und** per Bild so möglich sein, daß man diese Vergleiche neben dem Ausgangsbeispiel auf den Bildschirm holen und sehen kann. Beispiel: Von Stichwort 'Krieg und Frieden' zu Stichwort 'Stadt'.

- Aus dem eingegebenen Bildtitel sollten alle Begriffe und Künstler, historische Personen, Orte, Jahreszahlen und historische Begebenheiten, Aufbewahrungsort per Register erschließbar **und** abfragbar sein. Die Register sollten also beliebig in Listen zusammengestellt und auch ausgedruckt werden können.

- Titel und sonstige Angaben sowie ikonographische Schlagworte sollten in einer Maske eingegeben werden können, welche in Form einer 'offenen' Karteikarte mit dynamischen Feldern jederzeit Ergänzungen und längere Kommentare zuläßt. Dabei sollte auch möglich sein, bestimmte Angaben entfallen zu lassen oder später auszufüllen. (Dies sollte auch für Computerlaien möglich sein.) Eine Maske könnte folgendermaßen strukturiert sein: Künstler: Titel: Entstehungsjahr: Standort: Sekundärliteratur: PI-Schlagworte und Nummern: Kommentare:

- Ebenso sollen Verweise auf eingegebene Texte möglich und aufrufbar sein, die Quellen oder Kommentare zu den PI-Schlagworten sind.

- Ferner sollte es möglich sein, Abbildungen von Mikrofiche oder Mikrofilm (negativ oder positiv) einzuscannen.

- Wünschenswert ist die Verknüpfung mit der Literatur-Datenbank der politischen Ikonographie, aufgenommen unter der Berücksichtigung der PI-Schlagworte in 'Allegro' und 'Hidas Midas'.

A.2 Beschreibung der für den Index zur politischen Ikonographie zu erstellenden Software

Die zu entwerfende Software soll Zugriff auf eine Sammlung von rund 200.000 Abbildungen ermöglichen, zwischen denen sich Zettel mit Literaturangaben oder Zettel mit Zitaten aus Literatur oder Quellen befinden können. Diese Sammlung ist nach den sogenannten PI-Stichworten geordnet. Ein PI-Stichwort ist entweder ein PI-Hauptstichwort oder ein dem PI-Hauptstichwort zugeordnetes PI-Unterstichwort. Jedes PI-Stichwort hat eine (eventuell zusammengesetzte) arabische Nummer als Kennziffer. Diese sogenannte PI-Nummer ist

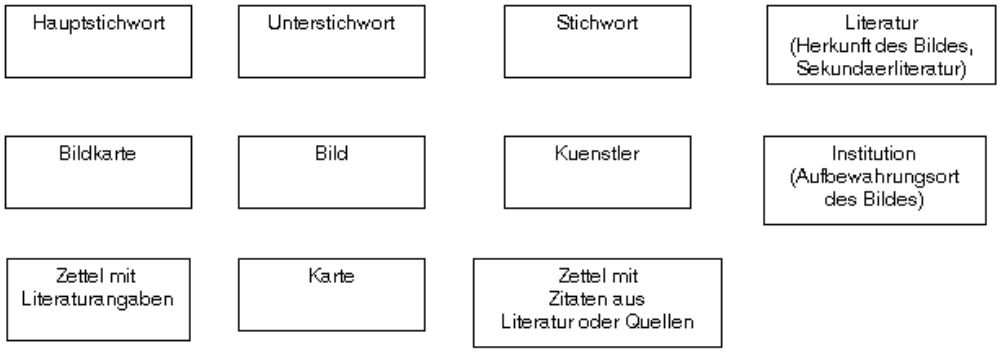
entweder die Nummer eines PI-Hauptstichwortes, oder sie ist aus der Nummer des PI-Hauptstichwortes und der Nummer des PI-Unterstichwortes zusammengesetzt (z. B. 255 Krieg, 255.15 Feindbilder). Es gibt jedoch Ausnahmen von der Systematik der Haupt- und Unterstichworte. So sind z.B. den Unterstichworten des Stichwortes 200 Gesten keine Nummern zugeordnet. Außerdem gibt es z.B. zum Stichwort 225.215 Herrscherbild/Tod des Herrschers noch weitere Unterstichworte, die keine Nummern haben.
Die Abbildungen werden im folgenden 'die Bildkarten' genannt. Eine Bildkarte besteht aus einer Abbildung (schwarz-weiß oder farbig) und aus einer Reihe von textuellen Informationen:

- PI-Stichworte
- PI-Nummer
- Künstlername und -vorname
- Bildtitel
- Entstehungszeit des Bildes
- Aufbewahrungsort des Bildes
- Literaturstelle der Bildentnahme
- Auftragsnummer der Fotostelle
- Kommentare
- Verweise auf andere PI-Stichworte, Bildkarten, Zettel mit Literaturangaben, Zettel mit Zitaten aus Literatur oder Quellen

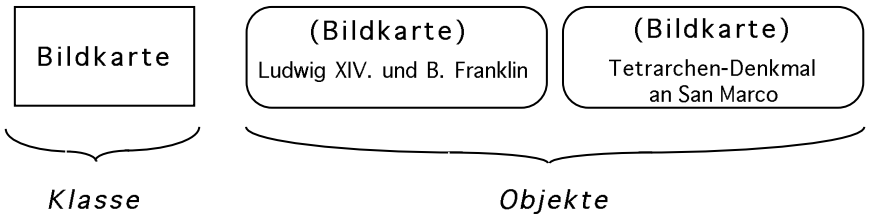
Hierbei sind nur PI-Stichwort, PI-Nummer und der Bildtitel immer auszufüllen. Die restlichen Informationen sind optional. Sämtliche Einträge können jederzeit geändert werden. Jedes Bild ist nach jedem dieser Einträge abfragbar und aufrufbar. Verweise von einer Bildkarte auf Vergleichsbeispiele sind derzeit noch nicht vorhanden. Diese Verweise sollen so möglich sein, daß man die Vergleichsbeispiele neben dem Ausgangsbeispiel auf den Bildschirm holen kann.
Desweiteren soll es Register geben, die nach den verschiedenen Einträgen (PI-Stichworte, PI-Nummern, Künstlername und -vorname, etc.) geordnet sind. Diese Register sollen beliebig in Listen zusammengestellt und auch ausgedruckt werden können.
Die Zettel mit Literaturangaben oder Zettel mit Zitaten aus Literatur oder Quellen sind unter ihren PI-Stichworten bzw. PI-Nummern abfragbar und aufrufbar. Es soll möglich sein, Bilder auf dem Bildschirm zu vergrößern und auszudrucken.
Eine Verknüpfung mit der Literaturdatenbank der politischen Ikonographie ist wünschenswert. Diese ist nach den PI-Stichworten geordnet und bisher über die Programme 'Allegro' und 'Hidas Midas' zugänglich.

B. Objektdiagramme

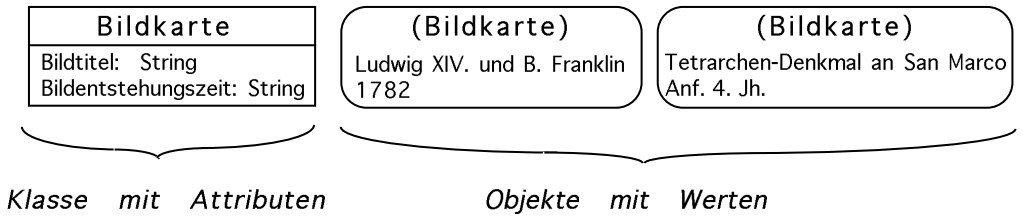
B.1 Wichtige Klassen im Index zur politischen Ikonograhie



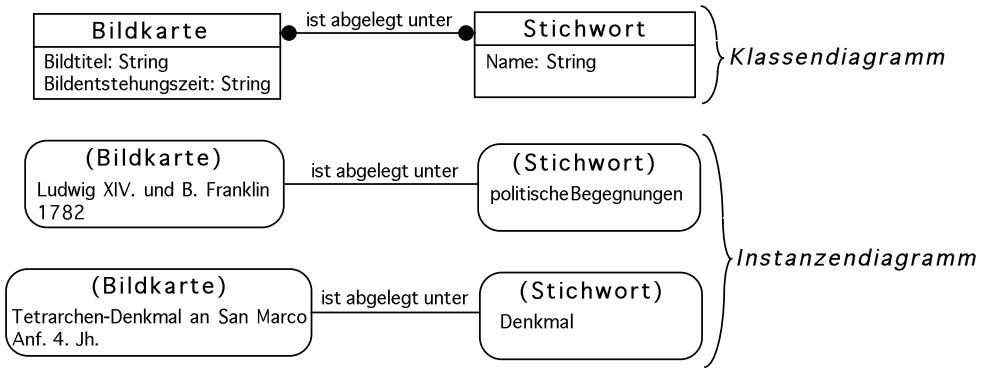
B.2 Klassen und Objekte



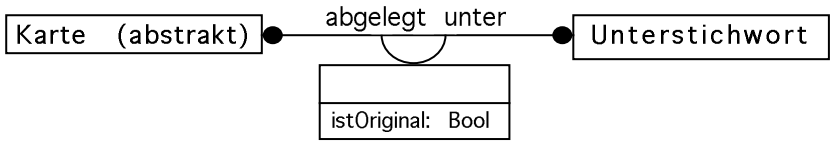
B.3 Attribute und Werte



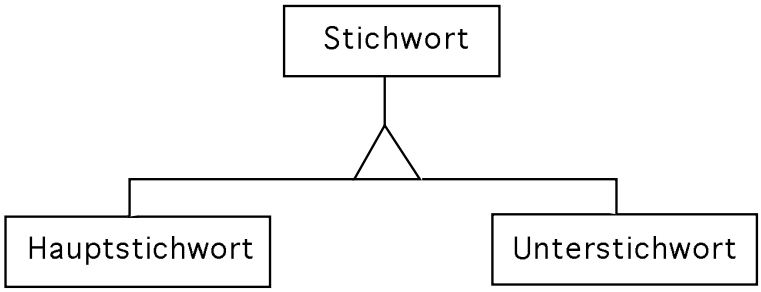
B.4 Verbindungen und Assoziationen



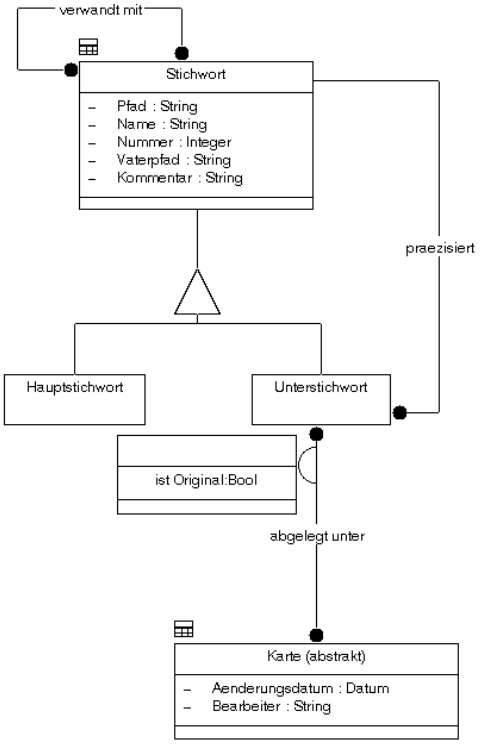
B.5 Verbindungsattribut



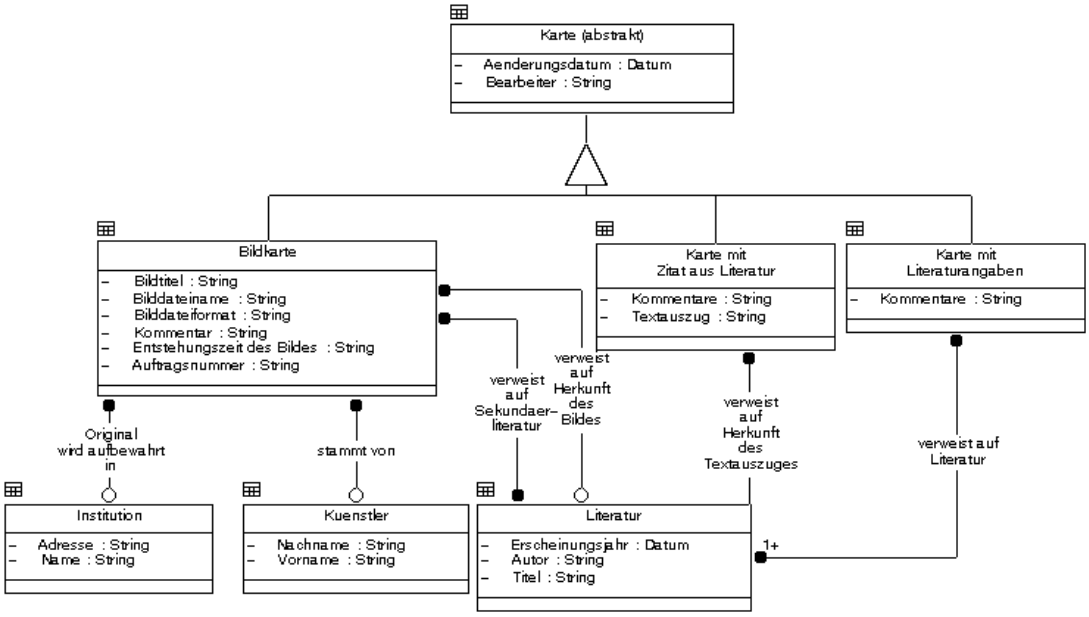
B.6 Generalisierung



B.7 OMT Objektmodell Stichworte



B.8 OMT Objektmodell Karten



Literaturverzeichnis

Baloui 94: Baloui, S. *Access 2.0, Das Kompendium.* Markt und Technik, 1994.

Chen 76: Chen, P. *The entity-relationship model – toward a unified view of data.* ACM Transactions on Database System 1, 1976, Nr. 1.

DeMarco 79: DeMarco, T. *Structured Analysis and System Specification.* Prentice Hall, 1979.

Floyd 84: Floyd, C. *A systematic look at prototyping.* In: Budde, R., Kuhlenkamp, K., Mathiassen, L., und H., Züllighoven (Hrsg.). *Approaches to Protyping.* Springer-Verlag, Berlin u.a., 1984, S. 1–18.

Fowler 93: Fowler, M. *A Comparison of Object-Oriented Analysis and Design Methods.* In: Carmichael, A. (Hrsg.). *Approaches to object-oriented analysis and design.* Ashgate, Aldershot, UK, 1993.

Grønbæk 88: Grønbæk, K. *Rapid Prototyping with Fourth Generation Systems.* Technical Report DAIMI PB - 270, Computer Science Department, Aarhus University, 1988.

Heusinger 94: Heusinger, L. *MIDAS-Handbuch.* K. G. Saur Verlag, München – New Providence – London – Paris, 1994.

Hüskes, Kurzidim 94: Hüskes, R. und Kurzidim, M. *Glaubenskrieg, Datenbankprogrammiersysteme unter Windows.* c't, Magazin für Computertechnik, 1994, Nr. 5, S. 150 – 164.

IBM 89: IBM. *Bildarchiv Foto Marburg: Von der gedruckten zur digitalen Überlieferung*, 1989.

Int 94: Interactive Develepment Environments. *Software through Pictures / Object Modeling Technique - Creating OMT Models, Release 2*, 1994.

Khoshafian, Abnous 90: Khoshafian, S. und Abnous, R. *Objekt Orientation Concepts, Languages, Databases, User Interfaces.* John Wiley & Sons, Inc., 1990.

Lockemann, Schmidt 87: Lockemann, P.C. und Schmidt, J.W. (Hrsg.). *Datenbankhandbuch.* Springer-Verlag, Berlin u.a., 1987.

Maslo, Dittrich 93: Maslo, Pia und Dittrich, Stefan. *Das große Buch zu Visual Basic 3.0 für Windows.* Data Becker, 1993.

Matthes 94: Matthes, F. *Objektorientierte Daten- und Prozeßmodellierung.* Vorlesungsskript, Fachbereich Informatik, Universität Hamburg, 1994.

Mic 93: Microsoft Corporation. *Microsoft Visual Basic Programmer's Guide*, 1993.

Müßig 94: Müßig, S. *Beiträge zur typsicheren generischen Datenvisualisierung.* Diplomarbeit, Fachbereich Informatik, Universität Hamburg, 1994.

Rumbaugh et al. 91: Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., und Lorensen, W. *Object-Oriented Modeling And Design.* Prentice-Hall, 1991.

Stein 93: Stein, W. *Objektorientierte Analysemethoden - ein Vergleich.* Informatik Spektrum, Jg. 16, 1993, Nr. 6, S. 317–332.

Versteegen 93: Versteegen, G. *Modellieren nach Plan.* iX, Multiuser Multitasking Magazin, 1993, Nr. 11, S. 142–146.

Warnke 93: Warnke, M. *Politische Ikonographie.* In: *Bildindex zur Politischen Ikonographie*, 1993, S. 5–12.

Zerbe 93: Zerbe, K. *Datenbanken ausreizen, Teil1: Programmierung und Applikationserstellung mit Access.* c't, Magazin für Computertechnik, 1993, Nr. 9, S. 200 – 206.