FH JOANNEUM Gesellschaft mbH - Fachhochschule

Lower-CASE
Evaluierung von Datenbank CASE-Tools für die Lehre

Bachelorarbeit 2
zur Erlangung des akademischen Grades „Bachelor of Science in Engineering“
eingereicht am Studiengang Informationsmanagement

Verfasser:
Mario Ranftl

Betreuer:
Dipl. -Ing. Ingo Farcher

Graz, 30.05.2011

FH JOANNEUM
INFORMATIONSMANAGEMENT

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Graz, 30.05.2011 _____

Mario Ranftl

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

CASE	Computer-Aided Software Engineering
DBMS	Datenbankmanagementsystem
DBS	Datenbanksystem
ERD	Entity Relationship Diagram
ERM	Entity Relationship Model
FAQ	Frequently Asked Questions
GB	Gigabyte (10^9 Byte = 1.000.000.000 Byte)
GUI	Graphical User Interface (grafische Benutzeroberfläche)
I-CASE	Integrated Computer-Aided Software Engineering
IDE	Integrated Development Environment (Entwicklungsumgebung)
PDF	Portable Document Format
PL/SQL	Procedural Language/SQL
SQL	Structured Query Language
T-SQL	Transact-SQL

Kurzfassung

In der modernen Software-Entwicklung werden aufgrund der steigenden Komplexität der zu entwickelnden Anwendungen und zur Erleichterung einer zielgerichteten Entwicklungsvorgangsweise zunehmend CASE-Tools zur Unterstützung des Entwicklungsprozesses eingesetzt (vgl. [Atlas 1999], S.4f.). Computer Aided Software Engineering (CASE) ist ein Konzept, das sich mit der Verwendung von Software-Werkzeugen zur rechnergestützten Entwicklung von Software auseinandersetzt (vgl. [Jenkins 2008], S.63). Durch Einsatz eines CASE-Tools kann der Kosten- und Zeitaufwand des Entwicklungsprozesses reduziert werden, während die Qualität des entwickelten Systems verbessert werden kann (vgl. [Lending 1998], S.49).

In der Datenbankentwicklung ist es mittlerweile üblich CASE-Tools einzusetzen. Die Entscheidung, welches Software-Produkt schlussendlich eingesetzt wird, ist sowohl im unternehmerischen als auch im universitären Bereich aufgrund der schieren Masse an verfügbaren Anwendungen, eine anspruchsvolle Aufgabe (vgl. [Jenkins 2008], S.63).

Das Ziel dieser Arbeit ist die Erarbeitung eines Kriterienkatalogs der die, für den Bereich der Datenbanklehre relevanten, Evaluierungskriterien zur Evaluierung von Datenbank-Lower-CASE-Tools definiert. Der Bereich Lower-CASE steht für Software die zur Unterstützung der späteren Entwicklungsphasen Implementierung und Test eingesetzt werden (vgl. [Pleumann 2007], S.34).

In einer Evaluierung werden bestimmte Objekte zielgerichtet und auf der Grundlage eines Systems von Kriterien mit festgelegten Messmethoden beurteilt. Neben der praktischen Evaluierung muss auch die Bestimmung der Kriterien methodisch erfolgen (vgl. [Lutz 2009], S.392ff.). Diese Arbeit definiert daher ein Vorgehensmodell das die erforderlichen Schritte enthält.

Die Eignung eines Tools für die Datenbanklehre bildet sich in den ermittelten Evaluierungskriterien u.a. durch den Fokus auf die Themen Benutzer-freundlichkeit und Funktionalität ab.

Abstract

Due to the increasing complexity in modern software development, CASE tools are used to support a goal-oriented software engineering process (see [Atlas 1999], p.4f.). The concept of Computer Aided Software Engineering (CASE) deals with software tools that support the process of building software (see [Jenkins 2008], p.63). By introducing CASE tools, costs and time of the development process can be reduced, while the quality of the systems developed can be improved (see [Lending 1998], p.49).

Nowadays, it is a common practice to use CASE tools for developing database systems. However, taking the final decision which CASE-tool to introduce at companies or universities is a challenging task, because of the huge amount of available possibilities (see [Jenkins 2008], p.63).

The aim of this degree dissertation is to gather a set of criteria for evaluating the optimal database Lower-CASE tool, which can be used for teaching databases at the FH-Joanneum. The term Lower-CASE refers to tools that support later phases of a software development process, such as implementation and testing (see [Pleumann 2007], p.34).

In an evaluation, specific objects are judged in a goal-oriented manner based on a system of criteria and fixed measurement methodologies. Apart from applied evaluation, also the determination of criteria has to occur systematically (see [Lutz 2009], p.392ff.). Consequently, the present paper defines a process model that covers the necessary steps of the evaluation process.

The ability for teaching databases at the FH-Joanneum with such tools is represented by focusing the criteria on subjects like usability and functionality.

1. Einleitung

Mittlerweile ist es nahezu unmöglich geworden, komplexe Softwareprodukte ohne Verwendung von unterstützenden Software-Werkzeugen zu entwickeln. Diese sogenannten CASE-Tools sind in unzähligen Ausführungen vorhanden und verfolgen das Konzept von Computer Aided Software Engineering (CASE). In der Lehre ist die praktische Verwendung dieser CASE-Tools und die Vermittlung des Konzepts, das CASE verfolgt, ein wichtiger Bereich im Lehrinhalt von Studiengängen mit einer Orientierung auf die Softwareentwicklung geworden. Sowohl im unternehmerischen, als auch im universitären Bereich, ist die Entscheidung welches Produkt schlussendlich eingesetzt werden soll aufgrund der schieren Masse an verfügbaren Anwendungen eine anspruchsvolle Aufgabe (vgl. [Jenkins 2008], S.63).

1.1. Problemstellung und Relevanz

Im Rahmen eines Semesterprojekts des Bachelorstudiengangs Informationsmanagement werden Datenbank-CASE-Tools mit dem Fokus auf die Eignung des Einsatzes innerhalb der Lehre an der FH Joanneum evaluiert. Das Projektteam besteht aus drei Studierenden dieses Studiengangs, das Projekt selbst gliedert sich in zwei Abschnitte. Im ersten Abschnitt werden die Kriterienkataloge für die Evaluierung der Software-Werkzeuge innerhalb von drei miteinander gekoppelten Bachelorarbeiten definiert. Diese Arbeiten decken die Evaluierungskriterien für die Themenbereiche „Upper-CASE-Tools ohne ER-Modellierung“, „ER-Modellierung“ und „Lower-CASE-Tools“ in der Datenbankentwicklung ab. Die vorliegende Arbeit untersucht die Merkmale von Datenbank-Lower-CASE-Tools. Im zweiten Abschnitt des Projekts wird die praktische Evaluierung durchgeführt. Die Ergebnisse der praktischen Evaluierung der ausgewählten Datenbank-CASE-Tools sind Gegenstand einer separaten Projektarbeit, in der die zuvor spezifizierten Evaluierungskriterien aller drei Bachelorarbeiten zur Beurteilung herangezogen werden.

Der Begriff CASE steht in unmittelbarer Verbindung mit der Entwicklung von Software und ist als ein sehr komplexer Themenbereich zu betrachten. Bereits die Definition des Begriffs wird zudem durch den Mangel an aktueller qualitativ hochwertiger Literatur und der Abwesenheit eines einheitlichen Klassifizierungsschemas von CASE-Tools erschwert (vgl. [Jenkins 2008], S.63). Infolgedessen sind bei diesem Projekt besonders die Absprache und gemeinsame Bearbeitung kritischer Themenbereiche durch das Projektteam erforderlich.

Die erarbeiteten Evaluierungskriterien der Datenbank-CASE-Tools sollen sowohl die allgemeine Eignung im Bereich der Lehre, als auch speziell die Eignung für die Bachelor- und Masterstudiengänge Informationsmanagement an der FH-Joanneum einfangen können.

1.2. Ziel der Bachelorarbeit

Ziel dieser Arbeit ist die Erstellung eines Kriterienkatalogs, der für die Lehre wesentliche Kriterien von Datenbank-Lower-CASE-Tools beinhaltet und für eine praktische Evaluierung im Rahmen eines Semesterprojektes eingesetzt werden kann. Durch den Fokus auf die Einsatzfähigkeit im Lehrbereich sind diese Evaluierungskriterien daher nur eingeschränkt für den Einsatz im unternehmerischen Bereich geeignet.

Die Definition des Vorgehensmodells ist ein essentielles Ziel dieser Arbeit. Die systematische Durchführung der Arbeitsschritte dieses Modells führt einerseits zur Erarbeitung des Kriterienkatalogs, andererseits zur Identifizierung der nötigen Phasen für die spätere praktische Evaluierung.

1.3. Aufbau der Arbeit

Im Rahmen dieser Arbeit müssen die Themengebiete CASE, Datenbankentwicklung und Evaluierung miteinander in Zusammenhang gebracht werden. Infolgedessen gliedert sich die Arbeit in fünf aufeinander aufbauende Teile.

Zu Beginn der Arbeit werden für das Verständnis relevante Themen, wie der Software Life-Cycle und CASE, vorgestellt. Im darauffolgenden Abschnitt

wird der Aufbau von Datenbanksystemen beleuchtet. Dies beinhaltet die Vorstellung der eingesetzten Datenbanksprachen und der wichtigen grafischen Komponenten einer Datenbankanwendung. Auf dieser Grundlage wird die Verbindung zu den Merkmalen von Lower-CASE-Tools hergestellt.

Im dritten Teil wird der Begriff Evaluierung definiert und erläutert. Durch die Definition eines Vorgehensmodells zur Evaluierung und der zielgerichteten Durchführung des vorgestellten Ablaufs ist das Ergebnis dieser Arbeit der Kriterienkatalog, der Kernteil dieser Arbeit, der im vierten Teil präsentiert wird.

Im abschließenden fünften Teil werden die gewonnenen Erkenntnisse diskutiert.

2. Begriffsdefinitionen

Innerhalb der folgenden aufeinander aufbauenden Abschnitte werden Begriffe, die für das Verständnis der Arbeit relevant sind, definiert und erläutert. Die adäquate Darlegung des Zusammenspiels der einzelnen Begriffe ist dabei von besonderer Bedeutung.

Im ersten Teil dieses Kapitels wird der Software- bzw. Datenbank-Entwicklungsprozess definiert. Auf Basis dieser Informationen wird im zweiten Teil dann der Begriff CASE definiert und ein Klassifikationsschema zur Abgrenzung der CASE-Tools vorgestellt.

2.1. Software Life-Cycle

Software und andere technische Artefakte unterliegen einem Lebenszyklus. Sie werden geplant, entworfen, hergestellt, in Betrieb genommen, benutzt und schließlich abgelöst und außer Dienst gestellt (vgl. [Störrle 2005], S.30).

„Beim Software-Lebenszyklus handelt es sich um den Zeitraum von der Idee der Software, seine [sic!] *Entwicklung und Wartung bis zur Ablösung der Software durch ein neues leistungsfähigeres Produkt.“* ([Atlas 1999], S.5). Es existieren verschiedene Life-Cycle-Modelle, welche diese Abschnitte, zumeist als Phasen bezeichnet, teils unterschiedlich genau definieren (vgl. [Störrle 2005], S.30).

Abstrahiert dargestellt unterliegt Software beispielsweise folgenden fünf grundlegenden Life-Cycle-Phasen ([Atlas 1999], S.5): *„Analyse, Design, Implementierung, Test, Wartung.“* Dieser einfache Aufbau lässt sich jedoch beliebig erweitern. Störrle stellt den Life-Cycle im Vergleich dazu in acht Phasen dar.

Abbildung 1 zeigt den Aufbau des Software Life-Cycles nach Störrle. Dieser veranschaulicht durch die Pfeilnotation das Verschwimmen der Grenzen der definierten Phasen und verdeutlicht damit die Schwierigkeit, alle Phasen unfehlbar abzubilden. Die geschlossene Kreisstruktur bringt zudem zum

Ausdruck, dass sich im Betrieb befindliche Software auch später in eine analytische bzw. entwerfende Phase zurückbegeben kann.

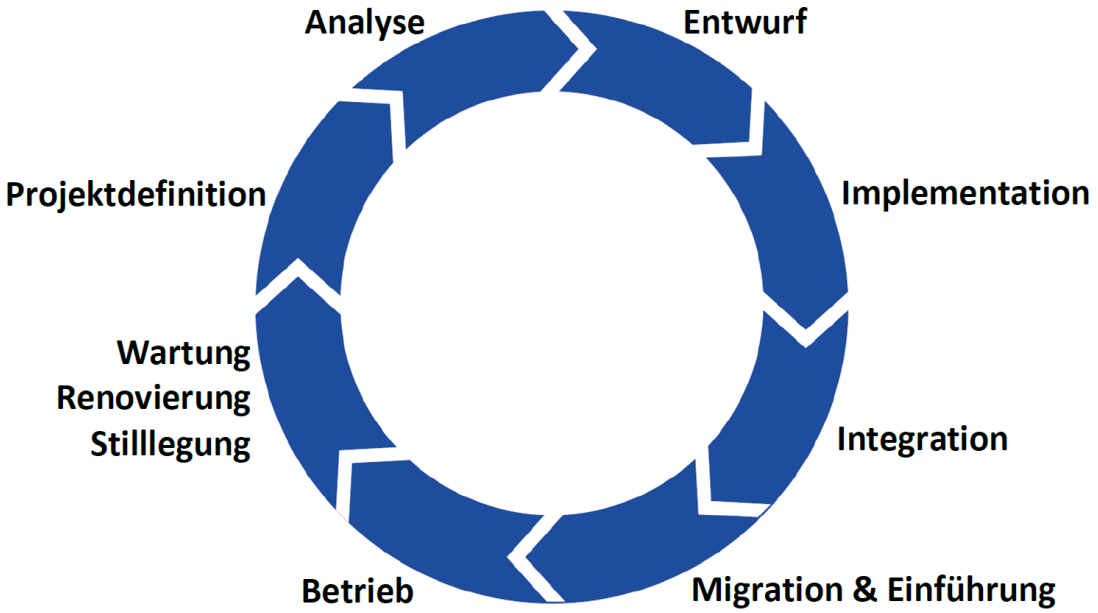

Abbildung 1: Software Life Cycle (vgl. [Störrle 2005], S.30)

2.1.1. Software-Entwicklungsprozess

Während der Software Life-Cycle die Lebensabschnitte einer Software repräsentiert, wird die Vorgangsweise zur Herstellung eines Softwareproduktes Entwicklungsprozess oder Softwareprozess genannt (vgl. [Störrle 2005], S.30). Es ist zwar möglich den Entwicklungsprozess generisch zu definieren, der tatsächliche Ablauf der Herstellungsphasen ist jedoch zumeist vom Softwaretyp abhängig (vgl. [Sommerville 2007], S.8). Ein sequentieller Entwicklungsprozess ist bei komplexeren Software-Produkten zudem schwer durchführbar. Dies würde bedeuten, dass die Software bereits in der Spezifikationsphase (also sehr früh im Entwicklungsprozess) vollständig beschrieben wird (vgl. [Hansen 2009], S.369).

Ein weit verbreitetes schematisches Vorgehensmodell ist das „Submodell Systemerstellung (SE)“ des Vorgehensmodells des Bundes (V-Modell), ein Phasenmodell, das den Entwicklungsablauf von Gesamtsystemen (Hard- und Software) darstellt (vgl. [Störrle 2005], S.30f.).

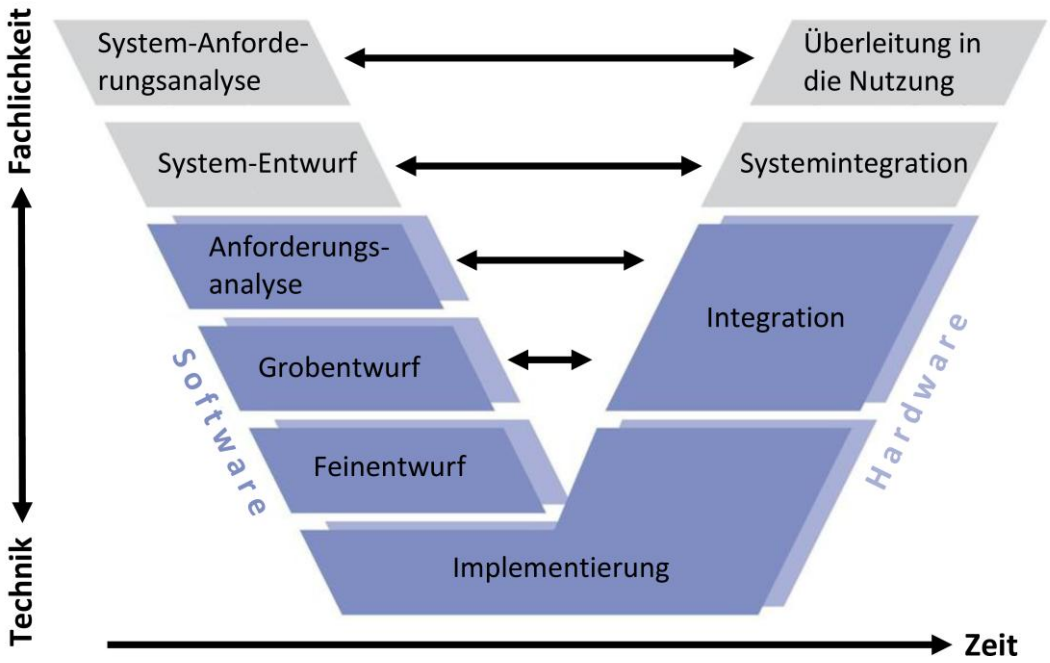

Abbildung 2: V-Modell Submodell SE (vgl. [Störrle 2005], S.31)

Abbildung 2 zeigt eine vereinfachte Darstellung dieses Modells. Der Ablauf startet beim Schritt „System-Anforderungsanalyse“ am linken Ast des Modells. Dieser Ast beherbergt die analytischen bzw. entwerfenden Phasen eines Entwicklungsprozesses. Über die Phase „Implementierung“ bewegt sich der Ablauf zum rechten Ast, in dem die konstruktiven Tätigkeiten untergebracht sind, weiter und endet mit der Phase „Überleitung in die Nutzung“. Die Überlappung der Aufgaben zur Zeitachse soll verdeutlichen, dass der Ablauf allerdings nicht streng sequentiell zu erfolgen hat. Die Beziehung zwischen den beiden Ästen, dargestellt durch vier Pfeile, veranschaulicht die Aus- bzw. Wechselwirkung der Phasen zueinander. Der untere Teil des „V“ ist zudem doppelt entworfen, da diese Phasen beim Entwicklungsprozess eines Gesamtsystems, sowohl für Software- als auch für Hardware-Komponenten ausgeführt werden müssen (vgl. [Störrle 2005], S.30f.).

Aufgrund des allgemein gültigen Aufbaus des beschriebenen Modells sind die Entwicklungsphasen, jenen eines Datenbank-Entwicklungsprozesses sehr ähnlich.

2.1.2. Datenbank-Entwicklungsprozess

Wie bereits angesprochen unterscheidet sich der Software-Entwicklungsprozess je nach Typ der Zielsoftware. Es ist außerdem wichtig festzuhalten, dass der folgende vorgestellte Datenbank-Entwicklungsprozess nicht die alleinige Wahrheit ist. Das bedeutet, es existieren auch andere Phasenmodelle, die den Ablauf der Datenbankentwicklung feiner bzw. größer darstellen.

Atlas gliedert den Entwicklungsprozess einer Datenbank in folgende Phasen (vgl. [Atlas 1999], S.6):

1. Anforderungsanalyse,
2. konzeptioneller Entwurf,
3. logischer Entwurf,
4. physischer Entwurf,
5. Implementierungsphase.

Im Vergleich zum vorherigen Modell wird ersichtlich, dass der rechte Ast des Submodells SE, der die konstruktiven Tätigkeiten beinhaltet, durch die Implementierungsphase innerhalb dieses Modells repräsentiert wird. Die Phasen Grob- bzw. Feinentwurf des linken Astes des Submodells SE erweitern sich zu drei Phasen. Dessen ungeachtet korrespondiert das grundlegende Ablaufs-Schema bei beiden Modellen.

Dieser Datenbank-Entwicklungsprozess beschränkt sich in seiner ursprünglichen Form auf den Datenbankentwurf (vgl. [Atlas 1999], S.5). Der Entwicklungsvorgang der eigentlichen Software, der Datenbankanwendung, wird in weiterer Folge der Implementierungsphase zugeordnet.

Die folgenden Abschnitte beleuchten die zuvor vorgestellten Datenbank-Entwicklungsphasen.

2.1.2.1. Anforderungsanalyse

Innerhalb dieser Phase werden die Anforderungen des Kunden bzw. Nutzers durch Interviews oder Fragebogenaktionen ermittelt. Die Ergebnisse der Analyse werden in einem Anforderungsdokument (User Requirements

Document) festgehalten und bilden die Grundlage des zu entwerfenden Datenbanksystems (DBS) (vgl. [Atlas 1999], S.6). Folglich sollte die Anforderungsanalyse gewissenhaft durchgeführt werden, da alle folgenden Phasen von der Definition dieser (Kunden-) Anforderungen beeinflusst werden.

2.1.2.2. Konzeptioneller Entwurf

Im konzeptionellen Entwurf erfolgt die Beschreibung des Ausschnitts der realen Welt, der in der Datenbank gespeichert werden soll, mit Hilfe eines Datenmodells (vgl. [Atlas 1999], S.6). Die, in der vorigen Phase definierten, Anforderungen müssen durch das Datenmodell erfüllt werden.

Mittels Entity-Relationship Modell (ERM) wird dieser Ausschnitt durch Entitäten, die Attribute besitzen und in Beziehung zueinander stehen, beschrieben. Durch diese natürliche Sichtweise können die wichtigsten semantischen Informationen der realen Welt eingefangen werden (vgl. [Chen 1976], S.9f.).

Durch das Entity-Relationship Diagramm (ERD) wird das ERM grafisch dargestellt. Der Einsatz des ERD bzw. ERM ist im konzeptionellen Entwurf von Datenbanken weit verbreitet (vgl. [Song 1995], S.2). Es kann daher als De-facto-Standard in der Datenmodellierung betrachtet werden.

2.1.2.3. Logischer Entwurf

Innerhalb dieser Phase erfolgt die Überführung des konzeptionellen Datenmodells in das zu verwendende Datenbankschema. In der Regel wird hierfür das relationale Modell eingesetzt. Dies verwendet relationale Tabellen zur Abbildung der im ERM definierten Entitäten, Beziehungen und Attributen (vgl. [Atlas 1999], S.6).

Nachdem die relationalen Tabellen aus dem ERM überführt wurden, müssen diese auf Einhaltung von spezifizierten Regeln, den Normalformen, überprüft werden. Regelverstöße könnten zu Inkonsistenzen des Datenbankinhalts führen. Relationale Tabellen, die diese Regeln nicht erfüllen, müssen daher umgeformt werden (vgl. [Atlas 1999], S.6).

2.1.2.4. Physischer Entwurf

In der Phase des physischen Entwurfs wird das zuvor generierte relationale Modell in den Datenbankcode des Ziel-DBS konvertiert (vgl. [Atlas 1999], S.6). In der Regel erfolgt dieser Überführungsvorgang – nach der Spezifizierung aller notwendigen Konvertierungsparameter – automatisch.

Da sich das zuvor festgelegte Datenbankmodell nicht direkt auf die physischen Speicherstrukturen (z. B. Dateien) umsetzen lässt, sind im Rahmen dieser Entwurfsphase bestimmte Entscheidungen zu treffen, die später eventuell wieder korrigiert werden müssen. Dies wird als „Tuning“ bezeichnet (vgl. [Vossen 2000], S.77).

Aufgrund dieser tendenziell entwerfenden Tätigkeit überschneidet sich die physische Entwurfsphase im DB-Entwurf bereits teilweise mit der Phase Implementierung des Submodells SE.

2.1.2.5. Implementierungsphase

An dieser Stelle wird die tatsächliche Implementierung und Installation der Datenbank auf dem verwendeten DBS durchgeführt (vgl. [Atlas 1999], S.6).

Sollte in weiterer Folge die Entwicklung einer Datenbankanwendung erfolgen, wird dies ebenfalls dieser Phase zugeordnet.

2.1.3. Reflexion

Obwohl die vorgestellten Modelle nur überblicksartig die Phasen des Software- bzw. Datenbank-Entwicklungsprozesses beschreiben, zeigen sie durch die Komplexität der Phasen und der Wechselwirkung zwischen diesen bereits die Schwierigkeit auf, Software ohne den Einsatz unterstützender Werkzeuge oder Methoden zu entwickeln. Darüber hinaus wird eine zielgerichtete Vorgehensweise durch den jährlich steigenden Umfang der Softwareprodukte (mehrere Millionen Programmcode-Zeilen), erschwert (vgl. [Atlas 1999], S.4). Dieser Problematik nimmt sich CASE an.

2.2. CASE

Das Akronym CASE steht für Computer-Aided Software Engineering und bezeichnet die Verwendung von Software-Werkzeugen zur rechnergestützten Entwicklung von Software (vgl. [Jenkins 2008], S.63). CASE verfolgt den Ansatz, alle Phasen eines ingenieurmäßigen Software-Entwicklungsprozesses zu unterstützen. Die ingenieurmäßige Vorgehensweise definiert sich durch eindeutige, koordinierte und wiederholbare Tätigkeiten, deren Repräsentation, Qualitätsstandard und Entwurfsregeln weitgehend akzeptiert sind (vgl. [Forte 1992], S.28).

CASE beschränkt sich dabei allerdings nicht auf ein spezielles Software-Genre. Vielmehr repräsentiert CASE einen ausgedehnten Bereich verschiedenster Typen von Programmen, die spezielle oder mehrere Aktivitäten eines Software-Entwicklungsprozesses unterstützen (vgl. [Sommerville 2007], S.12). Pleumann führt diesen Gedanken folgendermaßen weiter ([Pleumann 2007], S.33): *„Wenn der CASE-Begriff in der oben beschriebenen Allgemeinheit verwendet wird, umfasst er praktisch jedes Werkzeug, das (sinnvoll) im Rahmen der Software-Entwicklung eingesetzt werden kann.“*

Infolgedessen sollte CASE grundsätzlich nur als allgemeines Konzept zur automatisierten Unterstützung der Software-Entwicklungsphasen gesehen werden. Durch diese Vielseitigkeit sind präzisere Spezifizierungen des Begriffs schwierig. Für eine feinere Begriffsbestimmung werden daher direkt CASE-Tools, Software-Werkzeuge die den CASE-Ansatz verfolgen, betrachtet und klassifiziert.

2.2.1. CASE-Tool

CASE-Tools definieren sich durch die Umsetzung des CASE-Konzepts, die Unterstützung der ingenieurmäßigen Vorgehensweise im Software-Entwicklungsprozess, unabhängig vom Typ der unterstützten Entwicklungsphase, Aufgabe oder eingesetzten Notation (vgl. [Forte 1992], S.28).

Die Motivation derartige Werkzeuge innerhalb der Software-Entwicklung einzusetzen, ist die Reduzierung des Kosten- und Zeitaufwands des Entwicklungsprozesses, bei gleichzeitiger Verbesserung der Qualität der entwickelten Systeme (vgl. [Lending 1998], S.49). Infolgedessen steigern sich diese Vorteile bei zunehmender Komplexität eines Projekts. Aus diesem Grund ist die lückenlose Nutzung von CASE-Tools, innerhalb aller Life-Cycle-Phasen, bei zunehmender Größe eines Projekts, umso wichtiger (vgl. [Atlas 1999], S.5). Studien belegen, dass der Einsatz von CASE-Tools maßgeblich zur Steigerung der Produktivität im Entwicklungsprozess beiträgt (vgl. [Post 1998], S.1).

Als CASE-Tool werden sowohl jene Werkzeuge bezeichnet, die nur ausgewählte Aufgaben des Software Life-Cycles, als auch jene, die mehrere Life-Cycle-Phasen automatisiert unterstützen (vgl. [ISO/IEC 14102:2008], S.2). Aus diesem Grund werden zur Differenzierung von CASE-Tools Klassifizierungen verwendet.

2.2.2. Klassifizierung der CASE-Tools

Das bekannteste Klassifikationsschema stammt von Fuggetta und betrachtet primär den Umfang der angebotenen Unterstützung innerhalb des Software-Entwicklungsprozesses (vgl. [Pleumann 2007]. S.33f.). Die Einteilung der Werkzeuge erfolgt in drei Kategorien (vgl. [Fuggetta 1993], S.29, S.32, S.34):

- Als **CASE-Tool** werden Software Komponenten bezeichnet, die eine spezifische Aufgabe innerhalb einer Entwicklungsphase unterstützen.
- **CASE-Workbenches** implementieren verschiedene CASE-Tools in einer einzigen Anwendung und unterstützen einzelne Phasen innerhalb eines Entwicklungsprozesses.
- **CASE-Environments** setzen sich aus einer Menge von CASE-Workbenches und CASE-Tools zusammen und unterstützen große Teile bzw. den gesamten Entwicklungsprozess.

Neben dieser Klassifizierung, die sich auf die angebotene Funktionalität bezieht, differenziert Fuggetta zudem Upper-CASE- und Lower-CASE-Tools. Dieses Klassifikationsschema bezieht sich auf den Typ der Phase,

analytisch/entwerfend oder konstruktiv, die das Werkzeug unterstützt (vgl. [Pleumann 2007], S.34).

2.2.2.1. Upper-CASE

Als Upper-CASE-Tool werden Werkzeuge bezeichnet, die die frühen Phasen des Entwicklungsprozesses (Anforderung, Analyse, Entwurf) unterstützen (vgl. [Pleumann 2007], S.34). Das heißt, diese Klasse von Tools unterstützt die analytischen bzw. entwerfenden Phasen des Software-Entwicklungsprozesses.

In Hinblick auf den Datenbank-Entwicklungsprozess würden diese Werkzeuge demnach die Anforderungsanalyse, sowie Teile der darauffolgenden Entwurfsphasen unterstützen. Laut Chen ermöglichen Upper-CASE-Tools im Datenbankentwurf die rechnergestützte konzeptionelle Modellierung durch beispielsweise Funktionshierarchie-, Datenfluss- Prozess- und ER-Diagrammen (vgl. [Chen 1989], S.9).

Ein Teil des Upper-CASE-Bereichs sind ER-Modellierungstools, Softwaretools die die Erstellung des ERMs unterstützen (vgl. [Atlas 1999], S.9). Datenbankentwicklungstools, die neben ER-Modellierung auch andere Analyse- und Entwurf-Aufgaben unterstützen, kommen der Definition eines CASE-Environments nahe.

Der Bereich Upper-CASE bezieht sich im Bereich der Datenbankentwicklung also auf die Unterstützung der Phasen Anforderungsanalyse und des konzeptionellen und logischen Entwurfs. Derartige Programme besitzen jedoch zudem häufig Funktionalitäten zur automatischen Erzeugung des relationalen Modells und des Datenbankcodes aus den ERMs. Aufgrund der Unterstützung dieser konstruktiven Tätigkeiten im Bereich des physischen DB-Entwurfs, besitzen diese Datenbank-Upper-CASE-Tools eigentlich bereits Lower-CASE-Charakteristika.

Angesichts der Größe des Upper-CASE-Bereichs in der Datenbankentwicklung erfolgt die Definition der Evaluierungskriterien für Datenbank-Upper-CASE-Tools in den beiden anderen Bachelorarbeiten. Diese thematisieren die Ausschnitte „Upper-CASE ohne ER-Modellierung“

und „ER-Modellierung“. Themenbereiche der Arbeit „Upper-CASE ohne ER-Modellierung“ sind primär die Tätigkeiten innerhalb der Anforderungsanalyse während „ER-Modellierung“ sich mit den Datenbank-Entwurfsphasen beschäftigt.

2.2.2.2. Lower-CASE

Lower-CASE-Tools unterstützen die späteren Phasen (Implementierung, Test) des Software-Entwicklungsprozesses (vgl. [Pleumann 2007], S.34). Derartige Tools sind daher den konstruktiven Phasen des Software-Entwicklungsprozesses zuzuordnen.

Tools dieses Bereiches unterstützen folglich den physischen Entwurf und die Implementationsphase des Datenbank-Entwicklungsprozesses. Sie sind in der Lage automatisiert Programmcode zu erzeugen. Die Datenbank-anwendungsentwicklung und die Wartung ist dem Lower-CASE-Bereich zuzuordnen (vgl. [Chen 1989], S.9). Datenbank-Lower-CASE-Tools unterstützen Entwickler sowohl bei der Implementierung der Funktionalität als auch beim Entwurf des grafischen Designs einer Datenbankanwendung.

Die Abgrenzung zwischen Upper-CASE und Lower-CASE in der Datenbankentwicklung wird durch die aufgezeigten Beobachtungen folgendermaßen getroffen:

- **Upper-CASE** in der Datenbankentwicklung bezieht sich auf die Phasen Anforderungsanalyse und den konzeptionellen und logischen Entwurf.
- **Lower-CASE** in der Datenbankentwicklung unterstützt den physischen Entwurf und die Implementierungsphase des Datenbank-Entwicklungsprozesses.

Abbildung 3 veranschaulicht diese Abgrenzung zwischen Upper-CASE und Lower-CASE im Bereich Datenbanken.

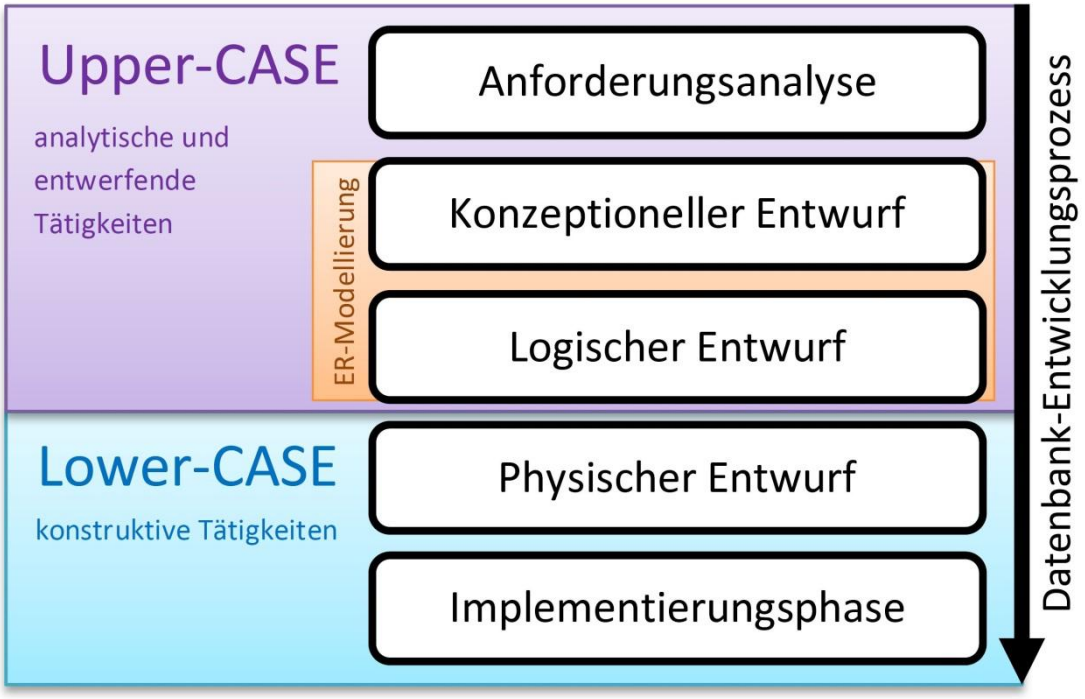

Abbildung 3: Abgrenzung Upper-CASE zu Lower-CASE

Die Implementierung des Datenbankcodes auf dem DBS und die Entwicklung der Datenbankanwendung ist folglich der primäre Unterstützungsbereich von Lower-CASE. Im Kontext der Lehre sind der Wartungsaspekt und die „Tuning"-Tätigkeiten bei Datenbank-Lower-CASE-Tools nicht von Bedeutung. Das Hauptaugenmerk liegt hier auf der Entwicklung von Datenbankanwendungen. Die Merkmale und Fähigkeiten eines derartigen Datenbank-Lower-CASE-Tools werden im Kapitel 3 (Entwicklung von Datenbankanwendungen) eingehend erläutert.

2.2.2.3. I-CASE

I-CASE steht für Integrated-CASE und bezeichnet Werkzeuge, die den vollständigen Software Life-Cycle unterstützen ([Beckworth 1993], S.22). Insofern konvergiert diese Klassifizierung prinzipiell mit der Klasse CASE-Environments des Klassifikationsschemas von Fuggetta.

Neben der Unterstützung des kompletten Entwicklungsprozesses besitzen I-CASE-Tools zudem einen entscheidenden Vorteil: Sie ermöglichen die Erstellung kompletter Anwendungen auf Grundlage konzeptioneller

Spezifikationen. Durch diese Weiterentwicklung wird eine neue Möglichkeit geschaffen: Rapid-Prototyping (vgl. [http://www.npd-solutions.com][1]).

Durch Rapid-Prototyping ist es möglich, bereits innerhalb der frühen Phasen des Software-Entwicklungsprozesses, eingeschränkt lauffähige Prototypen zu erstellen (vgl. [Kirchner 2001], S.199).

2.2.3. Reflexion

Das Ziel von CASE bzw. der Tools, die das CASE-Konzept verfolgen, ist die Steigerung der Entwicklungsgeschwindigkeit und der Qualität der zu entwickelnden Software. CASE ist ein allgemeines Konzept zur Unterstützung des Entwicklungsprozesses und resultierte aus der Problematik der steigenden Komplexität der Software. Die zwei Klassifizierungen Upper- bzw. Lower-CASE beziehen sich auf die analytischen/entwerfenden bzw. konstruktiven Phasen eines Software-Entwicklungsprozesses. Im Bereich der Lehre sind primär die unterstützenden Fähigkeiten von Datenbank-Lower-CASE-Tools bei der Entwicklung von Datenbankanwendungen relevant.

Somit konnte der zu evaluierende Bereich Lower-CASE formal definiert werden. Zur Feststellung der Aufgaben, die bei der Entwicklung von Datenbankanwendungen jedoch unterstützt werden müssen, ist eine genauere Betrachtung dieser notwendig.

3. Entwicklung von Datenbankanwendungen

Der Begriff Datenbankanwendung bezeichnet Programme, die die Abfrage oder Modifizierung von Daten, die in einer persistenten Form in einem DBS aufbewahrt werden, ermöglichen (vgl. [Chatterjee 2004], S.315). Folglich baut eine Datenbankanwendung auf ein bestehendes DBS auf. Das heißt, die Implementierung des Datenbankcodes, der aus dem Datenbankschema für ein bestimmtes DBS konvertiert wurde, hat zum Zeitpunkt der Datenbank-anwendungsentwicklung bereits stattgefunden. Es können nun Daten, in der im Datenmodell definierten Form, abgelegt werden.

Datenbankanwendungen können die unterschiedlichsten Ausprägungen annehmen. Die Bandbreite reicht von einfachen Anwendungen, bis hin zu komplexen Client/Server Anwendungen. Gemeinsam haben diese lediglich Funktionen der Datenmanipulation und –abfrage. Beispielsweise sehen E-Mail-Programme nicht wie typische Datenbankanwendungen aus, zur Speicherung von Adressen und Nachrichten wird dennoch eine Datenbank verwendet (vgl. [http://msdn.microsoft.com][1]). Durch diesen Umstand fällt es schwer, allgemeine Bestandteile einer Datenbankanwendung direkt zu identifizieren. Nur durch Betrachten der Endanwendungen können keine Rückschlüsse auf die Entwicklung und den unterstützenden Fähigkeiten mittels Datenbank-Lower-CASE-Tools gezogen werden. Daher müssen die Merkmale dieser Tools durch Erläuterung der Tätigkeiten innerhalb der Anwendungsentwicklung gewonnen werden.

In den folgenden Abschnitten wird die Architektur eines DBS erläutert, die Menge an möglichen Datenbanksprachen und die Benutzeroberfläche einer Datenbankanwendung, mit ihren grafischen Komponenten beschrieben. Dieser Aufbau ist zweckmäßig, da die Rolle, die Lower-CASE-Tools in diesen Themenbereichen einnehmen, begleitend vermittelt werden kann.

3.1. Architektur eines DBS mit einer Datenbankanwendung

Im Zuge der Kombination des DBS mit einer Datenbankanwendung ergibt sich eine Architektur, die aus mehreren Ebenen besteht. Generell teilt sich dieser Aufbau in zwei Bereiche auf (vgl. [Chatterjee 2004], S.315):

- Der Bereich **Backend** bezeichnet das Datenbanksystem. Dieses ergibt sich durch die Kombination aus Datenbank und Datenbankmanagementsystem (DBMS).
- Der Bereich **Frontend** bezieht sich auf die Anwendungen, die auf das Datenbanksystem über Schnittstellen (die das DBMS definiert) zugreifen und dadurch Daten abfragen und manipulieren können.

Die zentralen Aufgaben eines DBMS sind die Bereitstellung von Sichten (Views) auf die Daten, die Sicherung der Integrität durch Konsistenzprüfung der Daten, die Authorisierungssprüfung, die Sicherstellung der Synchronisation der Daten bei gleichzeitigem Zugriff von verschiedenen Benutzern und das Bereitstellen von Möglichkeiten zur Datensicherung (vgl. [https://www.bsi.bund.de][1]).

Der weitere Aufbau des Frontends ist vom Typ der Software abhängig, die auf den Client-Systemen – Systeme mit denen Endbenutzer/innen interagieren – eingesetzt werden (vgl. [Chatterjee 2004], S.315). „Thick“-Client bezeichnet Software, die als „Stand-Alone“-Programm direkt auf dem Client ausgeführt wird. Im Gegensatz dazu sind „Thin“-Clients von einem zusätzlichen Applikationsserver abhängig. Die Klassifizierung dieses, durch einen derartigen Server erweiterten, Gesamtaufbaus wird durch die zusätzliche Ebene als 3-Tier-Architektur bezeichnet. Bei 2-Tier-Architekturen kommuniziert der „Thick“-Client direkt mit dem DBS.

Der angesprochene Applikationsserver dient zur Bereitstellung der Datenbank-Anwendung für den Client und wird als Anwendungsschnittstelle zwischen Client-Software und DBS eingesetzt (vgl. [https://www.bsi.bund.de][1]). „Thin“-Clients sind beispielsweise Webbrowser, die mit dem Applikationsserver kommunizieren können. Die Erweiterung des Frontends durch einen zusätzlichen Applikationsserver kommt, durch die

zusätzlich geschaffene Anwendungsschnittstelle, allerdings auch „Thick“-Clients zugute, da spezielle Funktionalitäten dadurch ausgelagert werden können (vgl. [Chatterjee 2004], S.315).

Die Entscheidung, ob die Datenbankanwendung als „Thin“- oder „Thick“-Client ausgelegt wird, schränkt die Auswahl an verwendbaren Lower-CASE-Tools ein. Beispielsweise werden mit Oracle Application Express (APEX) Web-Applikationen entwickelt. Die Ausführung und selbst die Entwicklung dieser findet komplett im Web-Browser statt (vgl. [http://www.oracle.com][1]). Dies schließt die Entwicklung eines „Thick“-Clients im Vorhinein aus.

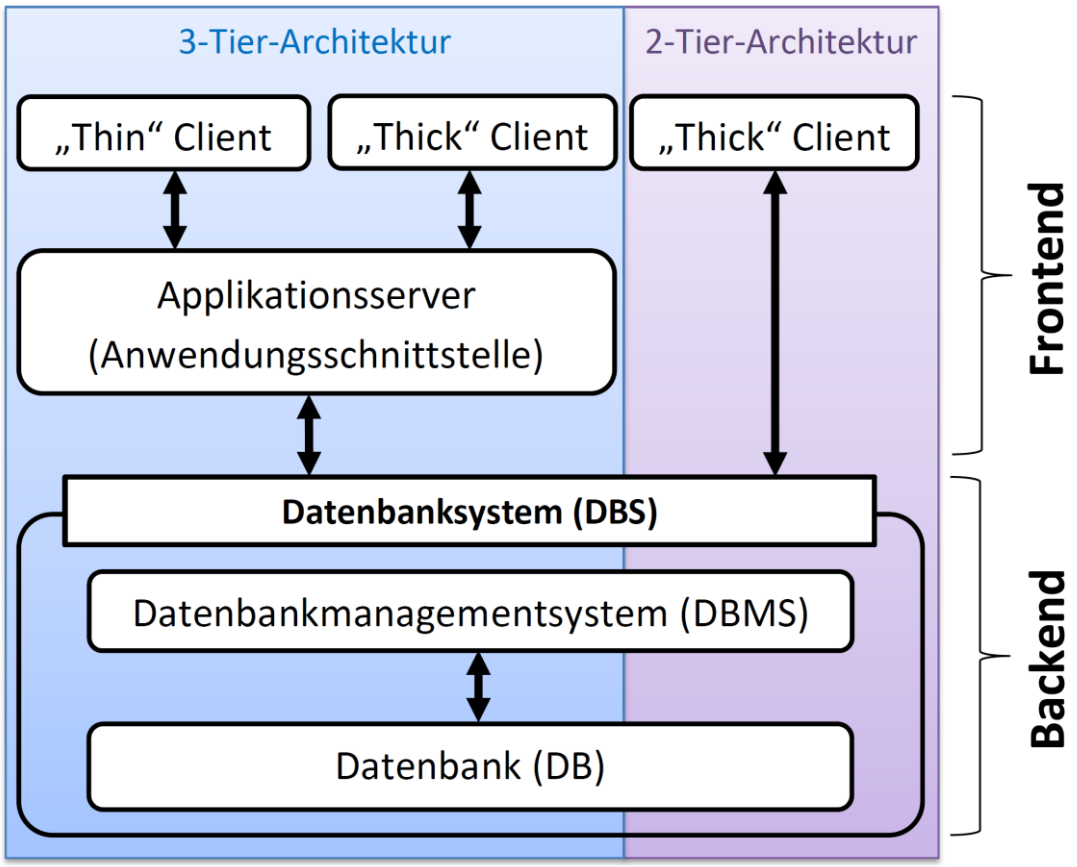

Abbildung 4: Architektur DBS mit DB-Anwendungen (vgl. [Chatterjee 2004], S.315 und vgl. [https://www.bsi.bund.de][1])

Abbildung 4 zeigt eine vereinfachte, schematische Darstellung des zuvor beschriebenen Aufbaus. Ein Datenbank-Lower-CASE-Tool wird, bezogen auf die vorgestellte Architektur, zur Entwicklung der Frontend-Applikation

eingesetzt. Zusätzlich sollte es die Features des Backends unterstützen und die Kommunikation mit diesem erleichtern.

3.2. Datenbanksprachen der Implementierungsphase

Das Schreiben des Programmcodes der Datenbankanwendung ist der primäre Bestandteil der Implementierungsphase. Folglich unterstützen Lower-CASE-Tools den Entwickler bzw. die Entwicklerin bei der Erstellung dieses Programmcodes. Insofern lohnt sich daher die Betrachtung der Sprachen, die bei der Datenbankanwendungsentwicklung eingesetzt werden.

Je nach Ebene (Tier) des zuvor vorgestellten Gesamtaufbaus können unterschiedliche Programmiersprachen verwendet werden. Die Einsatzmöglichkeit einer Sprache ist jedoch vom Softwaresystem abhängig, das innerhalb des Back- bzw. Frontend-Bereichs implementiert wird. Zur Einteilung der Sprachen wird folgendes Klassifizierungsschema verwendet (vgl. [Chatterjee 2004], S.315):

- **Backend-Sprachen** werden primär für die Datenabfrage bzw. -manipulation verwendet.
- **Frontend-Sprachen** werden primär zur Schaffung der Funktionalität einer Datenbankanwendung eingesetzt.

In den folgenden Abschnitten werden verbreitete Backend- bzw. Frontend-Sprachen vorgestellt, Features dieser Sprachen beschrieben und Unterstützungsmöglichkeiten durch Lower-CASE-Tools bestimmt.

3.2.1. Backend-Sprachen

Das Datenbanksystem wird in der Regel mit SQL bzw. einer Datenbank-Programmiersprache wie beispielsweise PL/SQL, Transact-SQL (T-SQL) oder Java angesprochen (vgl. [Chatterjee 2004], S.315). SQL steht für Structured Query Language, eine Datenbanksprache, die von den meisten DBMS unterstützt wird und als Industriestandard angesehen werden kann. Neben den Möglichkeiten der Datenabfrage und -manipulation umfasst der SQL-Wortschatz Statements (Anweisungen) zur Definition von Tabellen,

Beziehungen, Integritätseinschränkungen (Regeln zur Begrenzung der zugelassenen Daten) und Autorisierungsrechten (vgl. [Mannino 2007], S.7).

Eine Datenbank-Programmiersprache (wie beispielsweise PL/SQL oder T-SQL) definiert sich durch einen prozeduralen Sprachgebrauch und dem Vorhandensein einer Schnittstelle zu einem oder mehreren DBMSen. Diese Schnittstelle befähigt Programme prozedurale Anweisungen auf das DBS auszuführen, trotz der üblicherweise nicht-prozeduralen Datenbank-Zugriffsmöglichkeiten. Im Gegensatz zur reinen Abfragesprache SQL ermöglichen diese prozeduralen Sprachen sequentielle, verzweigte Operationen, die direkt abgearbeitet oder auf dem DBMS gespeichert werden können (vgl. [Mannino 2007], S.376ff.).

Lower-CASE-Tools müssen Funktionalitäten bieten, die die Erstellung und Überprüfung des Codes dieser Backend-Sprachen erleichtern. Beispielsweise besitzt Oracle APEX die Programmkomponente „Query Builder“. Diese ermöglicht (PL/)SQL-Anweisungen, ohne direktes Schreiben des Codes, innerhalb einer grafischen Benutzeroberfläche zu erstellen (vgl. [http://www.oracle.com][2]).

Innerhalb des Bachelorstudiengangs Informationsmanagement an der FH-Joanneum wird das Programmieren mit PL/SQL (Procedural Language/SQL) unterrichtet. PL/SQL ist eine proprietäre Datenbank-Programmiersprache des Datenbankherstellers Oracle. Seit der Einführung 1992 wurde die Funktionalität permanent erweitert, sodass die Sprache sowohl SQL Statements beherrscht, als auch Features von modernen Programmiersprachen aufweist (vgl. [Mannino 2007], S.380f.). In Bezug auf die Lehre an der FH-Joanneum ist die Unterstützung der Features und die Bereitstellung von Eingabehilfen für diese bzw. ähnliche Datenbank-Programmiersprachen (z.B. T-SQL) durch Lower-CASE-Tools ein wichtiger Faktor, der innerhalb der Evaluierungskriterien zu berücksichtigen ist.

Im folgenden Abschnitt werden zwei wichtige Features von PL/SQL vorgestellt.

3.2.1.1. Stored Procedures

Stored Procedures (gespeicherte Prozeduren) sind definiert als Auslagerung einer Sammlung von (PL/SQL)-Anweisungen direkt auf dem DBMS. Diese Prozeduren werden somit direkt am DBMS ausgeführt und gespeichert. Dies bedeutet für die Entwicklung zusätzliche Flexibilität, da die Prozeduren nicht am „Thin“-, „Thick“-Client oder Applikationsserver definiert bzw. ausgeführt werden müssen.

Zudem erlauben Stored Procedures durch den mächtigeren PL/SQL Wortschatz die Abwicklung von komplexeren Operationen, als mit reinem SQL möglich wäre. Bei Änderungen am Datenmodell kann das DBMS automatisch eine Neukompilierung der Stored Procedures vornehmen (vgl. [Mannino 2007], S.388, S.692).

Die Anzeige und Manipulation der Stored Procedures durch grafische Benutzeroberflächen innerhalb eines Lower-Case-Tools ist für eine erleichterte Entwicklung äußerst wünschenswert.

3.2.1.2. Trigger

Trigger bestehen ebenfalls aus einer Abfolge von Code-Anweisungen, jedoch werden diese automatisch beim Eintritt eines definierten Ereignisses (beispielsweise bei der Änderung von Daten) und einer bestimmten Bedingung vom DBMS ausgeführt. Durch Trigger können z.B. komplexe, tabellenübergreifende Integritätsüberprüfungen automatisch beim Eintritt eines Ereignisses durchgeführt werden (vgl. [Mannino 2007], S.402f.).

In Bezug auf Lower-CASE-Tools treffen dieselben Überlegungen wie bei Stored Procedures zu.

3.2.2. Frontend-Sprachen

Die Software der Applikationsserver und „Thick“-Clients ist typischerweise in höheren Programmiersprachen wie Java, Basic, Perl oder C++ geschrieben. „Thin“-Clients benutzen webbasierte Technologien oder Java-Applets zur Kommunikation mit dem Applikationsserver. Der Zugriff auf den Backend-Bereich durch die Frontend-Sprache erfolgt meist durch Einbindung der

Schnittstellen „Open Database Connectivity“ (ODBC) oder „Java Database Connectivity“ (JDBC) (vgl. [Chatterjee 2004], S.315).

Da unterschiedlichste Sprachen für die Entwicklung eines Datenbank-Frontends verwendet werden können, ist es schwierig grundlegende Funktionalitäten der Frontend-Sprachen einzufangen. Vielmehr lohnt sich allerdings der Blick auf die Funktionalitäten der Lower-CASE-Tools, die das Programmieren mit der Frontend-Sprache unterstützen.

Da prinzipiell alle modernen höheren Programmiersprachen – beispielsweise Java – die Kommunikation mit einem oder mehreren DBSen ermöglichen, trifft die Bezeichnung Lower-CASE-Tool in der Datenbankanwendungs-entwicklung, auch auf Integrated Development Environments (IDEs) zu, da diese die Entwicklung der Endanwendung unterstützen. Die Fähigkeiten bestimmter Programm-Komponenten dieser Tools können daher ohne Probleme auf die Features eines Datenbank-Lower-CASE-Tools umgelegt werden. Folgende Fähigkeiten sind bezogen auf den Einsatz in der Lehre wünschenswert (vgl. [Wihsböck 2007], S.6-8, S.18):

- Durch „**Syntax-Highlighting**“ werden Schlüsselwörter einer Programmiersprache farblich hervorgehoben.
- „**Code Completion**“ stellt eine Liste von, an der aktuellen Cursorposition, möglichen Ausdrücken zur Verfügung.
- Durch einen „**User Interface Designer**“ kann das grafische Benutzerinterface (GUI) des zu entwickelnden Software-Produkts intuitiv angeordnet und verändert werden. Wichtige Elemente einer GUI sind bereits vordefiniert.
- **Debugging**-Fähigkeiten erlauben die Überprüfung und Manipulation des Codes zur Laufzeit der Anwendung.

Der Grad der Unterstützung der Datenbank-Lower-CASE-Tools reicht von Hilfestellungen beim Schreiben des Codes der Frontend-Sprache über vordefinierte, wichtige (Anzeige-)Elemente bis hin zur kompletten Generierung einer Anwendungskomponente durch „Wizards“. Das heißt, entsprechende Datenbank-Lower-CASE-Tools bieten bei der Entwicklung

der Anwendungen auch Möglichkeiten, Programmteile ohne Kenntnisse der Frontend-Programmiersprache zu erstellen. Beispielsweise bietet Oracle APEX „Wizards“ und Eigenschaftsblätter zur Gestaltung der Komponenten der Datenbankanwendung (vgl. [http://www.oracle.com][3]).

3.3. Grafische Komponenten einer Datenbankanwendung

Nachdem die eingesetzten Programmiersprachen bei der Entwicklung von Datenbankanwendungen aufgezeigt wurden, beschäftigt sich dieser Abschnitt mit den wichtigsten Komponenten der grafischen Benutzeroberfläche der Datenbankwendungen und zeigt inwiefern Lower-CASE-Tools die Entwicklung dieser unterstützen.

Laut Mannino stellen Formulare und Berichte die Eckpfeiler der meisten Datenbankanwendungen dar. Formulare ermöglichen Endbenutzern eine einfachere Eingabe und Änderung von Daten. Berichte bereiten Daten strukturiert und formatiert für die Anzeige am Bildschirm oder den Ausdruck auf ([Mannino 2007], S.9, S.353, S.359). Folglich ist die Unterstützung der Formular- bzw. Berichterstellung eine substanzielle Funktionalität von Datenbank-Lower-CASE-Tools.

Am Beispiel des Lower-CASE-Tools Oracle APEX können folgende hilfreiche Funktionen identifiziert werden (vgl. [http://www.oracle.com][4]):

- „Wizards“ führen schrittweise durch den Erstellungsprozess der verschiedenen vordefinierten Formular- und Berichtstypen. Dies eignet sich zur schnellen Erstellung von eingeschränkt lauffähiger Prototypen (Rapid-Prototyping).
- Objekte und Eigenschaften der Formulare und Berichte können übersichtlich in einer grafischen Oberfläche textuell oder interaktiv manipuliert werden.
- Die Erstellung der Entwicklungs-Dokumentation und einer kontext-sensitiven Benutzerhilfe für die Datenbankanwendung wird durch vordefinierte Komponenten erleichtert.

- Die Definition des Grundaufbaus („Master-Layout“) für die Formulare und Berichte wird ermöglicht und die Erstellung der Navigation zwischen diesen wird erleichtert.

3.4. Reflexion

Neben den grundlegenden Erklärungen zur Architektur, der möglichen Programmiersprachen und der wichtigsten Komponenten der Datenbank-anwendungs-Benutzeroberfläche wurden die Verbindungen zu den Fähigkeiten eines Lower-CASE-Tools in diesen Bereichen aufgezeigt. Die Darstellung der Merkmale dieser Software-Produkte in der Entwicklung von Datenbankanwendungen ist von großer Bedeutung, da diese direkt in den Kriterienkatalog der Evaluierung einfließen.

Ein zusätzliches Merkmal, das Lower-CASE-Tools aufgrund der Einsatz-fähigkeit in der Lehre bzw. bei großen Projekten besitzen, ist die Möglichkeit der parallelen Anwendungsentwicklung durch mehrere Mitglieder eines Projektteams. Moderne Lower-CASE-Tools ermöglichen außerdem die Entwicklung von plattformunabhängigen Anwendungen.

Da der Lower-CASE-Bereich nun entsprechend festgelegt wurde, kann nun die Bestimmung des Evaluierungsmodells in Angriff genommen werden.

4. Methoden der Software Evaluierung

Im universitären Umfeld offeriert die Evaluierung der Leistung in den Bereichen Forschung und Lehre die Chance diese Bereiche nachhaltig zu verbessern. Da die Qualität der Lehre durch verschiedenste Rahmenbedingungen beeinflusst wird, ist es essentiell die wichtigsten dieser Einflussfaktoren bei der Evaluierung zu berücksichtigen. Zudem sollte der Gegenstand einer Evaluierung auf einen speziellen Bereich eingeschränkt werden, um auf Basis von detaillierten Analysen des Bereichs später konkrete Entscheidungen treffen zu können. Die Ergebnisse solcher Evaluierungen werden immer häufiger für die Vergabe von finanziellen Mitteln herangezogen (vgl. [Frank 2006], S.1).

Um die Evaluierung der Datenbank-CASE-Tools durchführen zu können, ist es essentiell, gezielt und methodisch vorzugehen. In den folgenden Abschnitten werden Grundlagen der Evaluierung vorgestellt und ein Überblick zur allgemeinen, als auch zur technischen Herangehensweise nach ISO/IEC 14102:2008 gegeben. Dies führt danach zur Ableitung und praktischen Anwendung des Vorgehensmodells, das die benötigten Phasen ausführlich behandelt.

4.1. Grundlagen der Evaluierung

„Evaluierung oder Evaluation heißt, bestimmte Objekte auf der Grundlage eines Systems von Kriterien zielbezogen zu beurteilen.“ ([Lutz 2009], S.392). Das Hauptergebnis dieser Arbeit, der Kriterienkatalog, definiert folglich dieses System von Kriterien. Evaluierungskriterien sind Eigenschaften eines Evaluierungsobjekts, die unter Berücksichtigung des Evaluierungsziels festlegen was evaluiert wird (vgl. [Lutz 2009], S.394f.).

Im Bereich des Informationsmanagements sind Evaluierungen von Produkten, Dienstleistungen, Methoden und anderen Objekten eine Notwendigkeit zur Bewältigung der strategischen und administrativen Aufgaben geworden. Besonders bei Vorhaben, die aus dem alltäglichen

Geschehen herausragen und Projekt-Charakteristika besitzen, besteht Bedarf an Evaluierungen (vgl. [Lutz 2009], S.392).

Die Ziele einer Evaluierung grenzen sich klar von denen einer Bewertung ab. Eine Bewertung verfolgt das Ziel den Geldwert bzw. die Kosten von Produkten zu ermitteln. Hingegen behandeln Evaluierungen die Beurteilung von Objekten nach beliebigen, systematisch festgelegten Zielen. Der Evaluierungsprozess muss zielgerichtet und methodisch erfolgen, auf ein System von Regeln aufbauen und intersubjektiv nachvollziehbar sein (vgl. [Lutz 2009], S.392f.). Daraus folgt, dass zur Evaluierung der Datenbank-CASE-Tools erst ein Modell aufgestellt werden muss, das die durchzuführenden Arbeitsschritte definiert.

4.1.1. Allgemeines Evaluierungsmodell

Lutz führt nach der Norm EN-1325-1 (Wertanalyse-Arbeitsplan) aus, dass eine Evaluierung nach einem bestimmten und dokumentierten Arbeitsplan zu erfolgen hat, der sich in mehrere Phasen gliedert (vgl. [Lutz 2009], S.393f.):

1. Festlegen der Evaluierungsobjekte,
2. Formulieren des Evaluierungsziels,
3. Ableiten der Evaluierungskriterien,
4. Gewichten der Evaluierungskriterien,
5. Abbilden der Evaluierungskriterien in Messgrößen,
6. Auswählen von Messmethoden und
7. Messen der Kriterienerträge und Organisation des Evaluierungsprozesses.

Bis auf die letzte Phase dieser Vorgehensweise befassen sich alle Phasen mit der Konzeption der Evaluierung. Erst in der siebenten Phase dieses Modells werden die Evaluierungsobjekte ausgewählt und die zuvor definierten Messmethoden auf diese angewandt. Diese Phase beschäftigt sich mit der Alternativensuche, Grobauswahl und Optimumbestimmung, bezieht sich daher auf die praktische Evaluierung, die innerhalb der Projekt-arbeit durchgeführt wird (vgl. [Lutz 2009], S.393-399).

Bei Betrachtung der ersten sechs Phasen wird ersichtlich, dass die praktische Evaluierung neben der Aufstellung der Evaluierungskriterien auch die Definition deren Gewichtung, Messgröße und -methode voraussetzt. Diese Punkte müssen daher in den Kriterienkatalog einfließen.

Da die Evaluierungsobjekte hinsichtlich ihrer Eignung in der Datenbanklehre an der FH-Joanneum beurteilt werden, muss außerdem ein Auswahlmodell gefunden werden, das schlussendlich die Bestimmung des optimalen Datenbank-CASE-Tools des jeweiligen Bereichs innerhalb der Phase ermöglicht. Laut Lutz werden Entscheidungsprobleme dieser Art durch eine Nutzwertanalyse, ein Modell für die systematische und nachvollziehbare Erarbeitung von Informationen zur Entscheidungsunterstützung, bearbeitet (vgl. [Lutz 2009], S.380).

4.1.2. Nutzwertmodell

Die Entscheidungsfindung ist ein häufig anzutreffender Aufgabenbereich im Informationsmanagement: Aus verschiedenen Handlungsalternativen soll unter Berücksichtigung mehrerer Ziele die optimale Alternative ausgewählt werden. Die Möglichkeiten müssen nach ihrer Vorziehenswürdigkeit, unter Beachtung der Präferenzen der Entscheidungsträger, angeordnet werden. Durch die Nutzwertanalyse kann die Vorziehenswürdigkeit eines Evaluierungsobjekts, folglich der Grad der Eignung in der Lehre, durch den sogenannten Nutzwert quantifiziert werden. Das Ergebnis wird in vier Arbeitsschritten ermittelt (vgl. [Lutz 2009], S.380):

1. Festlegen des Zielsystems
2. Ermitteln der Zielerträge
3. Ermitteln der Zielwerte
4. Durchführen der Wertsynthese

Die Methode wendet innerhalb der Phase der Wertsynthese die Rangordnungssummenregel an. Diese bestimmt den Nutzwert eines Evaluierungsobjekts anhand der, im ersten Arbeitsschritt festgelegten, Kriteriengewichtung und der gemessenen Kriterienerträge (vgl. [Lutz 2009], S.386). Um die Nutzwertanalyse innerhalb des zuvor vorgeführten

allgemeinen Evaluierungsmodells anwenden zu können, müssen daher alle konzeptionellen Phasen und Teile der praktischen Phase der Evaluierung durchgeführt worden sein. Innerhalb des Schritts Optimumbestimmung der letzten Phase „Messen der Kriterienerträge und Organisation des Evaluierungsprozesses“ könnte der Nutzwert eines Datenbank-CASE-Tools bestimmt werden. Die ersten beiden Arbeitsschritte werden jedoch bereits früher im allgemeinen Modell durchgeführt. Beispielsweise überschneidet sich die Definition der Evaluierungskriterien und der Kriteriengewichtung mit dem Arbeitsschritt „Festlegen des Zielsystems“ der Nutzwertanalyse.

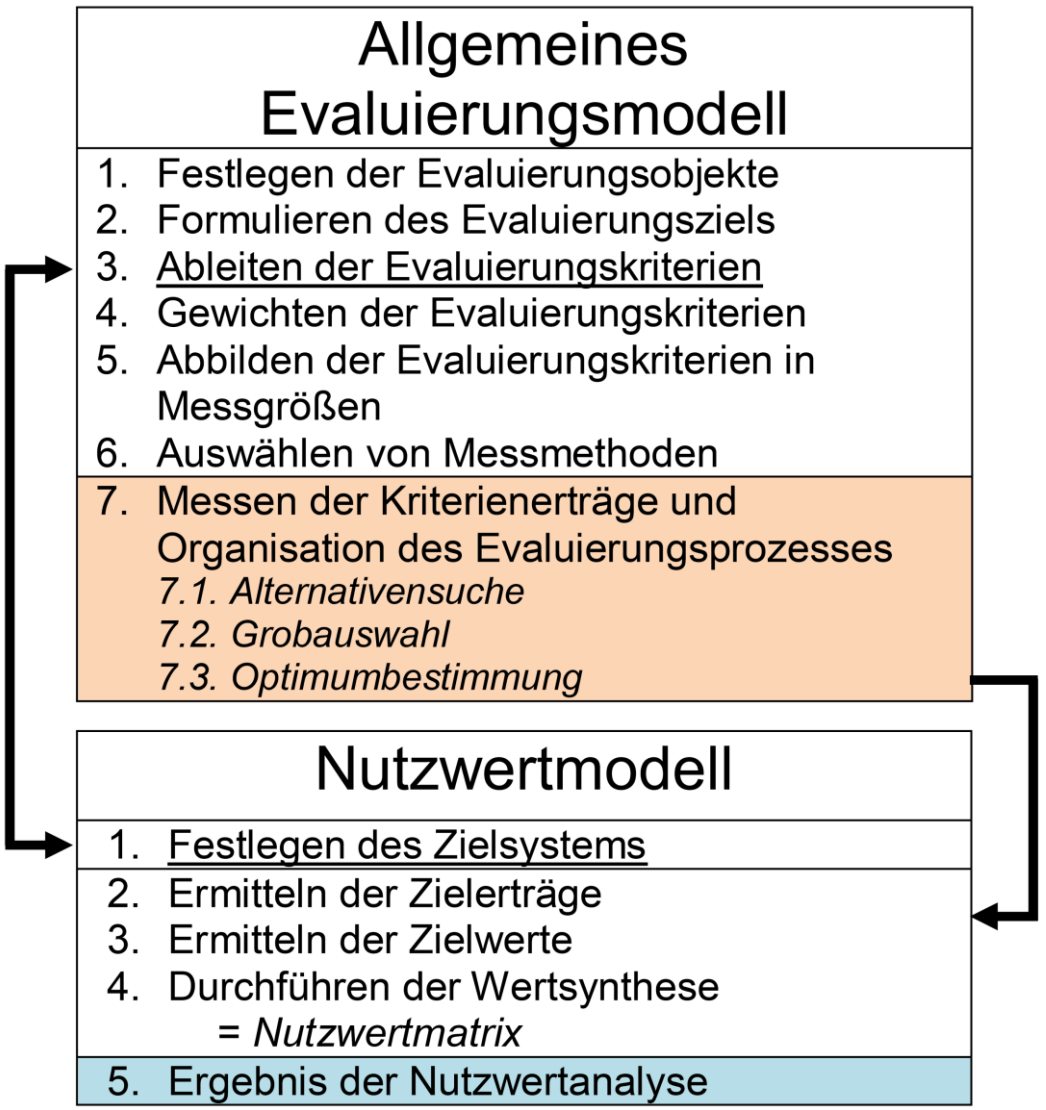

Abbildung 5: Kombiniertes Evaluierungsmodell (vgl. [Lutz 2009], S.380 und S.393f.)

Abbildung 5 verdeutlicht die zuvor beschriebenen Zusammenhänge mit der Nutzwertanalyse und veranschaulicht den Ablauf der Evaluierung.

4.1.3. Reflexion

Evaluierung ist ein zielgerichteter und methodischer Prozess. Um den angestrebten Kriterienkatalog für Lower-CASE-Tools zu bestimmen, muss daher zuvor ein klares Vorgehensmodell definiert werden. Durch die Durchführung der ersten sechs Phasen des allgemeinen Evaluierungs-modells entsteht der Kriterienkatalog durch den die praktische Evaluierung, die in der Projektarbeit thematisiert wird, durchgeführt werden kann. Da die Eignung eines bestimmten Datenbank-CASE-Tools in der Datenbanklehre jedoch noch nicht durch dieses Modell aufgezeigt werden kann, muss am Schluss der praktischen Evaluierung eine Nutzwertanalyse durchgeführt werden.

Durch Kombination des allgemeinen Evaluierungsmodells mit der Nutzwertanalyse entsteht ein Vorgehensmodell, um die Evaluierung der Datenbank-CASE-Tools zielgerichtet und methodisch durchzuführen. Durch die zusätzliche Einbeziehung der Merkmale einer technischen Evaluierung, kann dieses Modell für den praktischen Einsatz vorbereitet werden.

4.2. Evaluierung von CASE-Tools nach ISO/IEC 14102:2008

Diese ISO-Norm legt einen Leitfaden für die technische Evaluierung und Auswahl von CASE-Tools vor. Eine technische Evaluierung definiert sich durch die Zielsetzung, quantitative Resultate zu liefern, durch die die Auswahl des optimalen Evaluierungsobjekts getroffen werden kann. Durch Anwendung der zuvor vorgestellten Nutzwertanalyse könnte dieses Ergebnis also bereits erreicht werden. Zusätzlich sollte das finale Evaluierungsergebnis jedoch auch objektiv, nachvollziehbar und vorurteilsfrei sein. Dies hängt zu einem großen Teil von den verfügbaren Ressourcen und der Größe des Evaluierungsteams ab. Das Risiko des Eintritts von derartigen Kapazitätsproblemen und anderen Schwierigkeiten sollte bereits früh minimiert werden. Die Betrachtung des Ablaufs der technischen Evaluierung von CASE-Tools ist daher für die erfolgreiche Durchführung des zuvor

vorgestellten allgemeinen Vorgehensmodells überaus relevant (vgl. [ISO/IEC 14102:2008], S.1 und S.4).

Die technische Evaluierung des optimalen CASE-Tools gliedert sich in die folgenden vier Hauptprozesse (vgl. [ISO/IEC 14102:2008], S.4):

1. Prozess der Vorbereitung
2. Prozess der Strukturierung
3. Prozess der Evaluierung
4. Prozess der Selektion

Diese müssen jedoch nicht sequentiell durchgeführt werden. Die Auswahl der relevanten Prozesse und der damit verbundenen Aktivitäten ist Aufgabe des Evaluierungsteams bzw. des Auftraggebers der Evaluierung (vgl. [ISO/IEC 14102:2008], S.5). Dies verstärkt den Leitfaden-Charakter dieser Norm. In den folgenden Abschnitten werden die Prozesse beleuchtet und das bisherige Vorgehensmodell, falls erforderlich, entsprechend angepasst.

4.2.1. Prozess der Vorbereitung

In diesem Prozess werden die Ziele der Evaluierung und die K.o.-Kriterien definiert und ein Projektplan zur planmäßigen Durchführung der Evaluierung erstellt. Das Evaluierungsziel berücksichtigt die Interessen des Auftraggebers und soll folgende Fragen beantworten (vgl. [ISO/IEC 14102:2008], S.6):

- Warum wird ein CASE-Tool benötigt?
- Was muss das CASE-Tool können?

Die Auswahl an möglichen Evaluierungsobjekten kann bereits vorab durch K.o.-Kriterien, Eigenschaften die das Evaluierungsobjekt unbedingt aufweisen muss, eingeschränkt werden. Diese Eigenschaften resultieren direkt aus den Zielvorstellungen. Im Projektplan werden u.a. die Verantwortlichkeiten der Projektmitglieder, die durchzuführenden Aktivitäten und Monitoring-Mechanismen für die Ausführung der Evaluierung festgelegt (vgl. [ISO/IEC 14102:2008], S.7).

In Bezug auf das allgemeine Evaluierungsmodell sind keine Anpassungen erforderlich, da die Zielfindung in der zweiten Phase und die Festlegung der

K.o.-Kriterien im Schritt „Grobauswahl“ der praktischen Evaluierung vorgenommen werden (vgl. [Lutz 2009], S.399). Die Projektplanung und Dokumentation ist eine vorgeschriebene Tätigkeit dieses Semesterprojekts an der FH-Joanneum und wird in dieser Arbeit nicht behandelt.

4.2.2. Prozess der Strukturierung

Durch Sammeln von Informationen zu den CASE-Tools und einer Anforderungsanalyse wird die Definition der Evaluierungskriterien und deren Messmethode und Gewicht ermöglicht. Durch die Recherchetätigkeit können außerdem potentielle Evaluierungsobjekte identifiziert werden. Das Kriteriengewicht ist die relative Wichtigkeit eines bestimmten Evaluierungskriteriums im Vergleich zu der Gesamtheit der Evaluierungskriterien. Die Messmethodik bestimmt die Vorgangsweise zur Erhebung der Daten eines Kriteriums am Evaluierungsobjekt (vgl. [ISO/IEC 14102:2008], S.7ff.).

Die Merkmale, die Datenbank-Lower-CASE-Tools für den Lehrbereich besitzen müssen, wurden in dieser Arbeit bereits im Rahmen der vorherigen Kapitel angeschnitten. Durch Einbeziehung des Evaluierungsziels sollte es folglich möglich sein, den Kriterienkatalog im Rahmen des Vorgehensmodells zu definieren. Die angesprochenen Tätigkeiten im Prozess der Strukturierung sind Bestandteile der Phasen drei bis sechs im allgemeinen Evaluierungsmodell.

4.2.3. Prozess der Evaluierung

In diesem Prozess werden die finalen Kandidaten der Evaluierung durch Anwendung der zuvor definierten K.o.-Kriterien identifiziert. Danach wird ein Evaluierungsplan erstellt, der den zeitlichen Ablauf der praktischen Evaluierung spezifiziert. Die Kriterienerträge der CASE-Tools werden dann gemessen und innerhalb eines Evaluierungsberichts dokumentiert (vgl. [ISO/IEC 14102:2008], S.13).

Der Prozess der Evaluierung bezieht sich folglich auf die siebte Phase des allgemeinen Evaluierungsmodells, in der die praktische Evaluierung erfolgt.

4.2.4. Prozess der Selektion

Innerhalb dieser Phase wird ein Selektionsalgorithmus angewandt, um auf Basis der gemessenen Kriterienerträge das optimale CASE-Tool zu ermitteln. Danach kann die Entscheidung getroffen werden, ob das CASE-Tool tatsächlich innerhalb der Organisation eingesetzt wird (vgl. [ISO/IEC 14102:2008], S.14). Im zuvor präsentierten Vorgehensmodell wurde zur Bestimmung des optimalen Evaluierungsobjekts die Nutzwertanalyse eingeführt. Weitere Anpassungen des Vorgehensmodells sind daher nicht notwendig.

4.2.5. Reflexion

Die vorgestellte ISO-Norm ist ein Leitfaden für die technische Evaluierung von CASE-Tools und führt vier Hauptprozesse für die Durchführung einer Evaluierung auf, deren Ablauf nicht unbedingt sequentiell zu erfolgen hat. Die Tätigkeiten innerhalb dieser Prozesse überschneiden sich mit den Phasen des allgemeinen Evaluierungsmodells, weshalb keine weiteren Anpassungen des zuvor vorgestellten Vorgehensmodells notwendig sind.

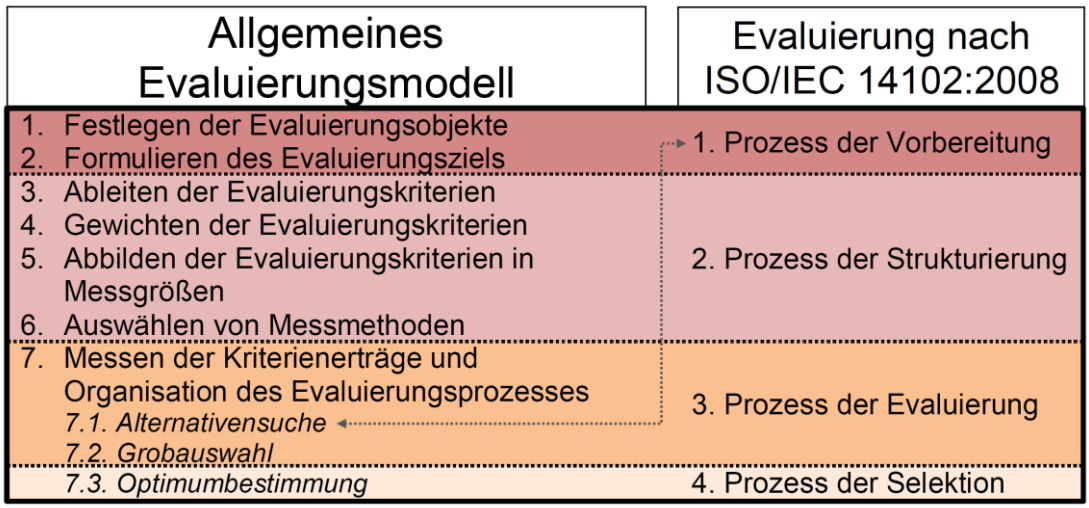

Abbildung 6: Überschneidungen mit ISO/IEC 14102:2008 (vgl. [Lutz 2009], S.393-399 und [ISO/IEC 14102:2008], S.3-15)

Abbildung 6 visualisiert die Überschneidungen der Phasen des allgemeinen Evaluierungsmodells mit den vier Hauptprozessen der ISO/IEC 14102:2008.

4.3. Vorgehensmodell

Die von Lutz beschriebene allgemeine Herangehensweise zur Evaluierung ist, durch die Erkenntnisse des letzten Abschnittes, auch für die technische Evaluierung von CASE-Tools geeignet. Daher können die Phasen dieses Arbeitsplans auch für die Evaluation von Datenbank-Lower-CASE-Tools eingesetzt werden. Da nun sowohl Aspekte der allgemeinen als auch der technischen Evaluierung in das Vorgehensmodell einbezogen wurden, kann mit der feinen Erläuterung der Phasen und der gleichzeitigen Durchführung, der für die Konzeption der Evaluierung essentiellen Phasen, begonnen werden.

In den folgenden Abschnitten werden die einzelnen Phasen beleuchtet und praktisch angewandt, falls diese für die Definition des Kriterienkatalogs Evaluierung erforderlich sind. Abbildung 7 zeigt überblicksartig alle notwendigen Phasen mit deren Teilschritten und grenzt die praktische Evaluierung von der Konzeption der Evaluierung ab.

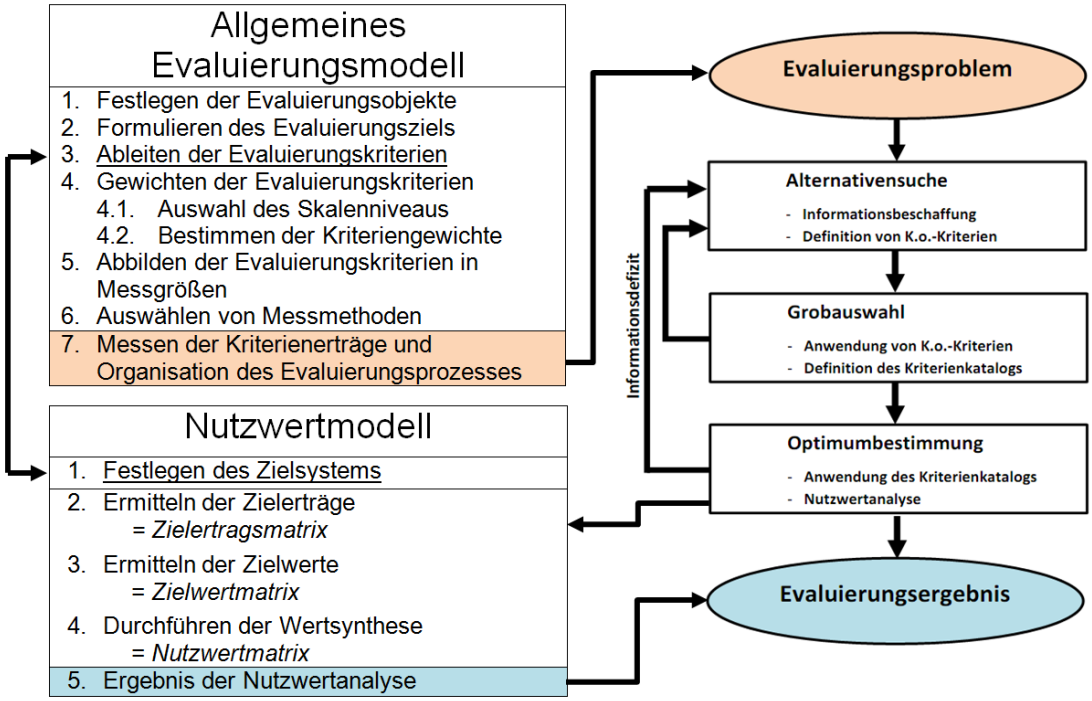

Abbildung 7: Vorgehensmodell für die Evaluierung der Datenbank-CASE-Tools (vgl. [Lutz 2009], S.393-399)

Der Prozess startet bei der ersten Phase des allgemeinen Evaluierungsmodells. In der vierten Phase „Gewichten der Evaluierungskriterien“ muss neben der tatsächlichen Gewichtung der Kriterien zuvor ein Skalenniveau für die Gewichtung gefunden werden. Diese Phase wurde daher um die entsprechenden Teilschritte erweitert. Die Konzeption der Evaluierung endet mit der sechsten Phase des allgemeinen Evaluierungsprozesses. Alle weiteren Phasen werden in dieser Arbeit folglich nur theoretisch betrachtet.

Der rechte Teil der Abbildung zeigt den Ablauf der siebten Phase „Messen der Kriterienerträge und Organisation des Evaluierungsprozesses“ und steht daher für die Vorgehensweise innerhalb der praktischen Evaluierung. Der Ablauf gliedert sich in die Arbeitsschritte Alternativensuche, Grobauswahl und Optimumbestimmung. Da die Informationsrecherche innerhalb der Alternativensuche bei fehlenden Informationen mehrmals erforderlich ist, werden die nachfolgenden Schritte, Grobauswahl und Optimumbestimmung, zu diesem Schritt rückgekoppelt. Innerhalb des letzten Schritts der praktischen Evaluierung, der Optimumsbestimmung, wird das Nutzwertmodell angewandt, welches in weiterer Folge in das Evaluierungsergebnis mündet (vgl. [Lutz 2009], S.399f.).

Dieser Ablauf stimmt aufgrund der gemeinsamen Projektarbeit mit den Vorgehensmodellen der anderen beiden Bachelorarbeiten überein.

4.3.1. Festlegen der Evaluierungsobjekte

In der ersten Phase werden die Evaluierungsobjekte festgelegt. Jede mögliche Konfiguration ist als Evaluierungsobjekt anzusehen. Daher wird in weiterer Folge auch für jedes Angebot bzw. alternative Konfiguration der Evaluierungsprozess durchlaufen (vgl. [Lutz 2009], S.394).

Die Evaluierungsobjekte dieser Evaluierung sind alle Datenbank-Lower-CASE-Tools, die zur Entwicklung von Datenbankanwendungen innerhalb der Lehre eingesetzt werden können.

4.3.2. Formulieren des Evaluierungsziels

Das Evaluierungsziel gibt einen zusammenfassenden Überblick, welche Informationen sich der Auftraggeber von der Evaluierung bzw. vom Ergebnis erwartet. Der geplante Verwendungszweck des Ergebnisses der Evaluierung ist für das Evaluierungsziel dabei von großer Bedeutung. Das Evaluierungsziel formuliert sich meist aus der Zielsetzung, absolute Aussagen je Objekt (z.B. der Nutzwert eines Evaluierungsobjekts), bzw. relative Aussagen (z.B. der maximale Nutzwert einer zur Evaluierung anstehenden Objektmenge) zu treffen (vgl. [Lutz 2009], S.394).

Das Evaluierungsziel dieser Evaluierung ist das Aufzeigen von alternativen Datenbank-Lower-CASE-Tools für den Lehreinsatz an der FH-Joanneum. Der Grad der Eignung (repräsentiert durch den Nutzwert) ist zu ermitteln. Das Ergebnis der praktischen Evaluierung in der Projektarbeit ist die strukturierte Übersicht von möglichen Tools geordnet nach dem ermittelten Nutzwert.

4.3.3. Ableiten der Evaluierungskriterien

Unter Berücksichtigung des Evaluierungsziels werden die Evaluierungs-kriterien – Eigenschaften der Evaluierungsobjekte – abgeleitet. Diese Kriterien legen daher fest, was praktisch evaluiert wird. Es sollten nur Evaluierungskriterien ausgewählt werden, wo abschätzbar ist, dass die verschiedenen Evaluierungsobjekte unterschiedliche Ausprägungen bezüglich dieses Kriteriums einnehmen werden. Wenn der Kriterienertrag eines Evaluierungskriteriums bei allen Evaluierungsobjekten gleich ist, trägt dieses Kriterium nicht zur Bestimmung des optimalen Evaluierungsobjekts bei (vgl. [Lutz 2009], S.394f.).

Für die Evaluierung der Datenbank-Lower-CASE-Tools werden Evaluierungskriterien zu den Hauptbereichen Allgemein, Benutzer-freundlichkeit und Funktionalität definiert. Diese umfassen die bereits im Kapitel 3 (Entwicklung von Datenbankanwendungen) erläuterten Merkmale eines Datenbank-CASE-Tools für den Lehrbereich. Da die praktische Evaluierung gemeinsam von allen drei Projektmitgliedern durchgeführt wird,

gleichen sich die Evaluierungskriterien der Hauptbereiche bzw. der Hauptkriterien „Allgemeine Kriterien“ und „Kriterien zur Benutzerfreundlichkeit“ in allen drei Bachelorarbeiten. In „Kriterien zur Funktionalität“ werden die jeweiligen, zum Themenbereich spezifischen, Kriterien behandelt, dieser Abschnitt behandelt folglich nur spezifische Kriterien zu Datenbank-Lower-CASE-Tools. Die genaue Beschreibung der Evaluierungskriterien ist aus dem Kriterienkatalog (Kapitel 5) zu entnehmen.

4.3.4. Gewichten der Evaluierungskriterien

In der Regel spielen die Evaluierungskriterien für die Ermittlung des Gesamtwertes eines Evaluierungsobjekts eine unterschiedlich große Rolle. Diese Wichtigkeit wird durch die Gewichtung eines Evaluierungskriteriums ausgedrückt. Durch die Herstellung einer Präferenzordnung können die Kriterienerträge bei der Ermittlung der Werte mit unterschiedlichem Gewicht berücksichtigt werden. In weiterer Folge ist daher die Definition des Skalenniveaus und des Verfahrens zur Bestimmung der Kriteriengewichte nötig (vgl. [Lutz 2009], S.383f. und S.395f.). Durch die Methode des paarweisen Vergleichs der Kriterien untereinander, mit Hilfe einer Präferenz-Matrix, können die Kriterien systematisch gewichtet werden (vgl. [Weber 2005], S.8). Dieses Verfahren wird daher genauer erläutert. Die folgenden zwei Abschnitte repräsentieren die essentiellen Arbeitsschritte zur Durchführung dieses Verfahrens.

4.3.4.1. Auswahl des Skalenniveaus

Das Gewicht eines Kriteriums, in Bezug auf die Gesamtheit der zu gewichtenden Kriterien, wird bei der Präferenzmatrix in Prozentzahlen ausgedrückt. Prinzipiell stehen die ordinale und kardinale Skala für die Definition der Gewichtung zur Verfügung (vgl. [Weber 2005], S.9 und [Lutz 2009], S.384).

In Bezug auf diese Evaluierung wurde vom Projektteam folgendes festgelegt:

- Es wird ein ordinales Skalenniveau mit 100 Ausprägungen für die Gewichtung der Kriterien verwendet. Das heißt, jedes Kriterium erhält durch Anwendung des Verfahrens der Präferenzmatrix einen

ganzzahligen Anteil der 100 Punkte, der das Gewicht eines Kriteriums bezogen auf das Gewicht aller Kriterien repräsentiert.

- Da der Paarvergleich mittels Präferenzmatrix auf eine möglichst kleine Anzahl von Kriterien ausgelegt ist, wird die Methodik nur auf die Kriterien eines jeweiligen Hauptkriteriums angewandt (vgl. [Lutz 2009], S.385f.). Die Punkte werden daher zuvor auf die drei Hauptkriterien verteilt:
  - „Allgemeine Kriterien“ hat ein Gewicht von 25 Punkten,
  - „Kriterien zur Benutzerfreundlichkeit“ erhält 40 Punkte und
  - „Kriterien zur Funktionalität“ bekommt 35 Punkte.
- Das Gewicht eines Kriteriums kann folglich auch im Verhältnis zum Gewicht des Hauptkriteriums abgebildet werden.

Der nächste Abschnitt erklärt die Anwendung des Paarvergleichs.

4.3.4.2. Bestimmen der Kriteriengewichte

Im den paarweisen Vergleich wird ermittelt, wie häufig ein bestimmtes Kriterium gegenüber anderen Kriterien, von der Gesamtheit der Entscheidungsträger, als wichtiger eingeschätzt wurde. Diese Verzugshäufigkeit ist die Grundlage für die Gewichtung der Kriterien. Auf eine erhaltene Vorzugshäufigkeit wird danach folgende Formel angewandt (vgl. [Weber 2005], S.9):

$$GwF_i = VH_i * 100 : \sum VH$$

Abbildung 8: Formale Struktur Paarvergleich (vgl. [Weber 2005], S.9)

Durch die in der Abbildung 8 präsentierten Formel wird die erhaltene Vorzugshäufigkeit eines Kriteriums (VH_i) mit 100 multipliziert und dann durch die Summe der möglichen Vorzugshäufigkeiten (Anzahl der Kriterien multipliziert mit der Anzahl der Entscheidungsträger) dividiert. Das Ergebnis der Präferenzmatrix ist die Prozentzahl der Verzugshäufigkeit im Vergleich zu den anderen Kriterien. Durch Strukturierung der erhaltenen Kriterien kann

daher die benötigte Präferenzordnung hergestellt werden (vgl. [Weber 2005], S.9f.).

Die Entscheidungsträger dieser Evaluierung sind die drei Projektmitglieder des Projektteams. Für die Evaluierung der CASE-Tools innerhalb der Projektarbeit muss bei der Durchführung dieses Verfahrens noch die zuvor festgelegte 100-Punkte-Gewichtungsskala berücksichtigt werden. Die, im Rahmen der Präferenzmatrix, erhaltene Prozentzahl eines Evaluierungskriteriums wird daher zusätzlich mit der Gewichtung des dazugehörigen Hauptkriteriums (25, 40 oder 35 Punkte) multipliziert und dann durch Ab- oder Aufrunden in eine Ganzzahl umgewandelt. Sollten sich etwaige Probleme bei dieser Umwandlung ergeben, obliegt es den Projektmitgliedern, das Verhältnis zwischen den Evaluierungskriterien durch einstimmige Entscheidungen entsprechend zu korrigieren. Durch den praktischen Charakter dieser Tätigkeit wird die tatsächliche Gewichtung aller Kriterien innerhalb der Projektarbeit durchgeführt. Die Ergebnisse der Phase „Gewichten der Evaluierungskriterien“ werden daher nicht in dieser Arbeit betrachtet.

4.3.5. Abbilden der Evaluierungskriterien in Messgrößen

„Jedem Kriterium, das sich nicht in weitere Sub-Kriterien teilen lässt, muss eine Messgröße assoziiert werden. [...] Die Skala muss dabei mindestens ordinal skaliert sein [...] um ausdrücken zu können, welcher Wert von zwei Möglichkeiten bevorzugt wird.“[1] ([Stamelos 2000], S.5).

Laut Lutz sind Messgrößen zu definieren, deren erhaltenes Messerergebnis durch eine definierte Messmethode intersubjektiv überprüft werden kann. Außerdem sollen die Messgrößen mit einem angemessenen Aufwand verwendbar sein (vgl. [Lutz 2009], S.396).

Diese Evaluierung verwendet daher eine ordinale Skala mit drei Ausprägungen, die durch die +/-/o-Notation dargestellt werden:

1 Übersetzung durch den Verfasser.

- Das Symbol „**+**“ bezeichnet einen Kriterienertrag eines Evaluierungsobjekts dessen angesprochenes Evaluierungskriterium **komplett erfüllt** wird.
- Das Symbol „**o**“ bezeichnet einen Kriterienertrag eines Evaluierungsobjekts dessen angesprochenes Evaluierungskriterium **teils erfüllt** wird.
- Das Symbol „**-**“ bezeichnet einen Kriterienertrag eines Evaluierungsobjekts dessen angesprochenes Evaluierungskriterium **nicht erfüllt** wird.

Da die Lower-CASE-Tools bezüglich der Unterstützung von bestimmten Fähigkeiten untersucht werden, wirkt sich die Limitierung auf drei mögliche Ausprägungen positiv auf den Aufwand zur Bestimmung des Kriterienertrags aus. Zur Unterstützung der Nachvollziehbarkeit der Ergebnisse muss der Weg, der zum jeweilig gemessenen Kriterienertrag geführt hat, innerhalb der Projektarbeit festgehalten werden.

4.3.6. Auswählen von Messmethoden

Kriterienerträge können direkt, wie z. B. durch Beobachten oder indirekt, wie z. B. durch Befragen, erfasst werden. Beobachten bedeutet dabei methodisches, selektives und systematisches Vorgehen zu einem bestimmten Evaluierungszweck (vgl. [Lutz 2009], S.397f.).

Diese Evaluierung setzt neben der Literaturrecherche zur Ermittlung der Kriterienerträge der allgemeinen Kriterien, direktes Beobachten am Datenbank-Lower-CASE-Tool ein. Die Evaluierung wird gemeinsam von den drei Projektmitgliedern durchgeführt, die die definierten Kriterien am Evaluierungsobjekt messen.

4.3.7. Messen der Kriterienerträge und Organisation des Evaluierungsprozesses

Die letzte Phase des allgemeinen Evaluierungsmodells „Messen der Kriterienerträge und Organisation des Evaluierungsprozesses“ beschäftigt sich mit der praktischen Evaluierung und definiert hierzu drei Arbeitsschritte

(vgl. [Lutz 2009], S.399f.). Diese werden in den folgenden Abschnitten in der Theorie und in Bezug auf diese Evaluierung vorgestellt, die praktische Anwendung erfolgt jedoch in der Projektarbeit.

4.3.7.1. Alternativensuche

Der erste Arbeitsschnitt der praktischen Evaluierung umfasst die Ausschreibung bzw. Recherchetätigkeit zu Informationen von potenziellen Evaluierungsobjekten. Diese Informationen stellen dann die Grundlage für die Definition der K.o.-Kriterien dar (vgl. [Lutz 2009], S.399). Innerhalb der Projektarbeit müssen folglich die Ergebnisse der Recherche nach potentiellen Evaluierungsobjekten und herausgearbeiteten K.o.-Kriterien dokumentiert werden.

4.3.7.2. Grobauswahl

Die K.o.-Kriterien können nun zur Beurteilung der Erträge an den potentiellen Evaluierungsobjekten angewandt werden. Durch den Limitierungscharakter der spezifizierten K.o.-Kriterien kann die Menge der zulässigen Objekte reduziert werden. Dies führt dazu, dass nur noch Evaluierungsobjekte die alle angestrebten Eigenschaften besitzen, für die weitere Evaluierung in Betracht gezogen werden. Nach Bestimmung der zulässigen Evaluierungsobjekte können etwaige zusätzliche Evaluierungskriterien spezifiziert werden. Sollten noch nicht alle Informationen zur Bestimmung der Evaluierungsobjekte eingeflossen sein, muss die Phase „Alternativensuche“ erneut aufgesucht werden (vgl. [Lutz 2009], S.399f.).

Die Projektarbeit muss folglich die Anwendung dieser K.o.-Kriterien und die Bestimmung der letztlich zulässigen Datenbank-Lower-CASE-Tools dokumentieren. Sollten etwaige Änderungen am Kriterienkatalog nötig sein, muss dies ebenfalls behandelt werden.

4.3.7.3. Optimumbestimmung

Die Erträge der Evaluierungskriterien werden an den Evaluierungsobjekten gemessen und das optimale Objekt wird bestimmt. Falls die bisher verwendeten Informationsquellen nicht ausreichen, muss zur Phase

„Alternativensuche“ zurückgesprungen werden und die Evaluierungskriterien eventuell angepasst werden (vgl. [Lutz 2009], S.400).

Für die Bestimmung des optimalen Evaluierungsobjekts eines evaluierten Bereichs wird die Nutzwertanalyse in dieser Evaluierung eingesetzt. Die Ermittlung wird in vier Teilschritten durchgeführt. Diese Schritte werden innerhalb der folgenden Abschnitte erläutert.

4.3.7.3.1. Ermitteln der Zielerträge

In diesem Schritt werden die ausgewählten Messmethoden angewandt und die Messungen am Evaluierungsobjekt durchgeführt. Nach dieser Datenerhebung müssen alle Zielerträge der Evaluierungsobjekte, repräsentiert durch ihren Messgrößen-konformen Wert, in eine Zielertragsmatrix eingetragen werden (vgl. [Lutz 2009], S.383).

Das Ergebnis dieser Evaluierung wird daher eine Übersichtstabelle sein, in der die jeweiligen Kriterienerträge aller Lower-CASE-Tools in ihren definierten Ausprägungen („+“, „o“ oder „-“) eingetragen sind.

4.3.7.3.2. Ermitteln der Zielwerte

Unter Berücksichtigung der Gewichtung eines Evaluierungskriteriums werden die gemessen Kriterienerträge in einen numerischen Wert überführt. Hierzu ist es zunächst erforderlich ein Zielskalenniveau festzulegen (vgl. [Lutz 2009], S.383f.).

Diese Evaluierung definierte eine ordinale Gewichtungsskala, die 100 mögliche Ausprägungen annehmen kann. Das Gewicht eines Kriteriums wird durch einen ganzzahligen Wert dargestellt mit dem gerechnet werden könnte. Die gemessenen Erträge sind derzeit jedoch in der +/-/o-Notation abgebildet und werden daher in multiplizierbare Faktoren umgewandelt:

- Der Zielertrag „+“ erhält den Faktor 1.
- Der Zielertrag „o“ erhält den Faktor 0,5.
- Der Zielertrag „-“ erhält den Faktor 0.

Durch Multiplikation eines transformierten Zielertrags mit dessen festgelegten ganzzahligen Kriteriengewicht-Punkten können die jeweiligen Zielwerte

ermittelt werden. Die Zielskala bleibt ordinal skaliert mit ihrem Wertebereich von 0 bis 100, besitzt nun jedoch, durch die drei definierten Ausprägungen der Zielerträge und infolge des Transformationsfaktors „0,5“, 200 Ausprägungen im Abstand von 0,5er-Schritten.

4.3.7.3.3. Durchführen der Wertsynthese

Die Rangordnungssummenregel ist eine sehr operationale Entscheidungsregel und wird häufig bei ordinal skalierten Zielskalen eingesetzt. Sie ermittelt den Nutzwert eines Evaluierungsobjekts (N_i) durch Summierung aller zuvor überführten Zielwerte (Kriteriumertrag n_{ij} * Kriteriumgewicht g_j). Abbildung 9 zeigt die formale Struktur dieser Entscheidungsregel (vgl. [Lutz 2009], S.386).

$$N_i = \sum_{j=1}^{m} n_{ij} \cdot g_j$$

N_i Nutzwert der Alternative i
n_{ij} Nutzwert der Alternative i bezüglich Kriterium j
g_j Gewicht des Kriteriums j

Abbildung 9: Formale Struktur Rangordnungssummenregel ([Lutz 2009], S.386)

In Bezug auf diese Evaluierung würde somit als Ergebnis der Nutzwert von allen evaluierten Datenbank-Lower-CASE-Tools feststehen.

4.3.7.3.4. Ergebnis der Nutzwertanalyse

Die Handlungsalternativen müssen nun in eine widerspruchsfreie und vollständige Ordnung gebracht werden. Dies erfolgt durch Beurteilung des ermittelten Nutzwertes (vgl. [Lutz 2009], S.386).

Das Lower-CASE-Tool mit dem höchsten ermittelten Nutzwert wäre folglich allen anderen vorzuziehen.

4.4. Reflexion

Durch die Definition eines eigenen Vorgehensmodells für diese Evaluierung, das sowohl Aspekte der allgemeinen, als auch der technischen Evaluierung von CASE-Tools in sich vereint, konnten alle konzeptionellen Vorgänge durchgeführt und die weitere Vorgangsweise bzw. die durchzuführenden Tätigkeiten bei der praktischen Evaluierung beleuchtet werden. Die

tatsächliche Gewichtung der Evaluierungskriterien wird, aufgrund des gemeinschaftlichen Charakters der Paarvergleichs-Methode, im Rahmen der Projektarbeit von den Projektmitgliedern durchgeführt. Im nächsten Abschnitt werden die ermittelten Evaluierungskriterien der Datenbank-Lower-CASE-Tools vorgestellt.

5. Kriterienkatalog

Der folgende Abschnitt umfasst neben den ermittelten Evaluierungskriterien die Dokumentation der Aspekte, die zu deren Definition geführt haben. Die Gliederung der Kriterien erfolgt in die drei Hauptbereiche „Allgemeine Kriterien“, „Kriterien zur Benutzerfreundlichkeit“ und „Kriterien zur Funktionalität“. Nur der Bereich „Kriterien zur Funktionalität“ betrachtet spezifische Kriterien der Datenbank-Lower-CASE-Tools. Die beiden anderen Hauptbereiche können auf alle Arten von Datenbank-CASE-Tools angewandt werden, daher wurden diese Evaluierungskriterien in allen drei Bachelorarbeiten deckungsgleich formuliert.

5.1. Allgemeine Kriterien

Datenbank-CASE-Tools sind allgemein gesehen, zunächst Softwareprodukte zur Erreichung von definierten Zielen eines bestimmten Einsatzbereichs. Die üblichen, allgemeinen Kriterien von Software-Produkten sind folglich auch für die Datenbank-Lower-CASE-Tools anwendbar (vgl. [Kirchner 2001], S.35). Die jeweiligen sechs Subkriterien dieses Hauptkriteriums beschäftigen sich daher mit der Bestimmung dieser allgemeinen Aspekte. Im Rahmen der praktischen Evaluierung werden insgesamt 25 Punkte auf die Evaluierungskriterien verteilt. Diese repräsentieren das Gewicht des jeweiligen Kriteriums.

5.1.1. Kosten

Universitäre Einrichtungen haben meist ein vorgegebenes Budget für Neuanschaffungen, das nicht überschritten werden darf. (vgl. [Kirchner 2001], S.35). Daher sind die anfallenden Kosten, unter Berücksichtigung eventueller Rabatte, für den Einsatz in der FH-Joanneum ein wichtiger Faktor. Dieses Kriterium beurteilt die Kosten pro Arbeitsplatz, die beim Kauf einer Einzellizenz für die günstigste Produktvariante anfallen.

Beurteilung:
Das Projektteam definierte drei Beurteilungsbereiche, um die Kosten in die benötigte Messgröße der Kriterienerträge zu konvertieren:

- Bei einem Preis unter EUR 199,00 wird mit „+“ bewertet.
- Zwischen EUR 200,00 und EUR 999,00 wird mit „o“ beurteilt.
- Ab EUR 1000,00 resultiert die Beurteilung mit „-“.

5.1.2. Lizenzmodelle / Softwarevarianten

Die Lizenzierungsmöglichkeiten von Softwareprodukten unterscheiden sich hinsichtlich der Art der Ausstellung der Lizenzen. Die Anzahl der Lizenzen kann beispielsweise personenbezogen („named user“) oder mengenbezogen („concurrent user“) vergeben werden (vgl. [Nüttgens 2002], S.3). Es ist wichtig ein Lizenzmodell zu wählen, das zum geplanten Einsatzzweck passt. Die Auswahl einer ungeeigneten Lizenzvariante stellt sich sonst später als produktivitätsmindernder Faktor heraus (vgl. [Kirchner 2001], S.36). Dieses Kriterium betrachtet daher die angebotenen Lizenzmodelle oder Softwarevarianten eines Evaluierungsobjekts und prüft aufgrund des Einsatzes an der FH-Joanneum zusätzlich das Vorhandensein spezieller universitärer Lizenzierungsprogramme.

Beurteilung:
Für die bestmögliche Beurteilung „+“ müssen unterschiedliche Lizenzmodelle, von denen mindestens eines eine mengenbezogene Option anbietet, vorhanden sein und Vergünstigungen für den universitären Bereich ermöglicht werden. Wenn eine dieser Eigenschaften am Evaluierungsobjekt fehlt, muss „o“ vergeben werden. Ab zwei Verstößen resultiert der Kriterientrag in „-“.

5.1.3. Systemvoraussetzungen

Die Mindestanforderung an die Hardware setzt sich üblicherweise aus Prozessorleistung, Hauptspeicher und Festplattenspeichervolumen zusammen (vgl. [Nüttgens 2002], S.6). Die Betrachtung der Hardwarevoraussetzungen ist lohnend, da dadurch beurteilt werden kann, ob das Tool auf bestehender Hardware eingesetzt werden kann oder ob Nachrüstungen

bzw. völlig neue Systeme nötig sind. Die Informationen setzen sich einerseits aus recherchierten Herstellerangaben und andererseits aus eventuellen praktischen Erfahrungen mit dem Tool zusammen (vgl. [Kirchner 2001], S.36).

Das Projektteam spezifizierte für dieses Kriterium gemeinsam einen Standard-PC, der den Ziel-Hardwarekomponenten an der FH-Joanneum entspricht. Dieser besteht aus einem Dual Core Prozessor, zwei GB Arbeitsspeicher und 70 GB Festplattenspeicher.

Beurteilung:
Wenn die kommunizierten Hersteller-Mindestanforderungen unter den jeweiligen Ausprägungen am Standard-PC liegen, resultiert der Kriterienertrag mit „+“. Sollte nur eine Komponente, wie beispielsweise der Arbeitsspeicher, den Voraussetzungen nicht entsprechen, wird mit „o“ beurteilt. Ansonsten wird „-“ vergeben.

5.1.4. Installation

Innerhalb dieses Kriteriums wird die Komplexität des Installationsvorgangs eines Evaluierungsobjekts beurteilt (vgl. [ISO/IEC 14102:2008], S.29). Bei der Installation können zudem eventuelle Aktivierungs- bzw. Kopierschutz-Verfahren auftreten. Eine „Einfachheit“ ist somit nicht immer gegeben (vgl. [Nüttgens 2002], S.6). Im Studienalltag kann eine leicht durchzuführende Installation die Austauschbarkeit bzw. Flexibilität der eingesetzten Systeme erhöhen.

Beurteilung:
Für die bestmögliche Beurteilung „+“ müssen innerhalb des Installationsprozesses der Speicherort und die Funktionalität der Softwarekomponenten anpassbar sein. Außerdem müssen eventuell vorhandene Aktivierungsvorgänge schnell und nachvollziehbar durchgeführt werden können. Fehlt eine entsprechende Konfigurationsfähigkeit resultiert der Kriterienertrag mit „o“. Sollen umfassende Recherchetätigkeiten bzw. Supportanfragen an den Hersteller nötig sein, um das Tool zu installieren, wird mit „-“ beurteilt.

5.1.5. Programmdokumentation

Die Vollständigkeit, Korrektheit, Konsistenz und verständliche Formulierung der Dokumentation ist ein wichtiger Faktor bei CASE-Tools (vgl. [ISO/IEC 14102:2008], S.29). Diese kann beispielsweise in Papierform, auf elektronischen Medien (CDs) oder Onlineplattformen zur Verfügung gestellt werden (vgl. [Kirchner 2001], S.42). Für den Lehreinsatz sind daher beispielsweise kontextsensitive Online-Hilfen, Tutorials und die Verfügbarkeit von Online-Foren wünschenswert, da diese die zeitliche Dimension des Einarbeitungsaufwands verringern.

Beurteilung:
Der Kriterienertrag resultiert in „+“, wenn der Hersteller ein moderiertes und möglichst aktives Online-Forum, eine kontextsensitive Hilfe, Tutorials und umfangreiches Referenzmaterial zur Verfügung stellt. Fehlen zwei dieser Ausprägungen wird „o“ vergeben. Ansonsten wird mit „-“ beurteilt.

5.1.6. Support

Das Evaluierungskriterium Support betrachtet neben den seitens des Herstellers zur Verfügung stehenden kostenlosen bzw. kostenpflichtigen Supportmöglichkeiten bzw. -arten, auch die Aktualität der verfügbaren Updates bzw. Softwareversionen (vgl. [Kirchner 2001], S.42f.).

Beurteilung:
Der Kriterienertrag resultiert in „+“ wenn verschiedene kostenlose und kostenpflichtige Supportmöglichkeiten zur Verfügung stehen und unterschiedliche Supportkanäle genutzt werden können. Dies beinhaltet beispielsweise Onlinesupport, Telefonsupport, FAQs sowie die Übermittlung von aktuellen Updates. Sollte nur kostenpflichtiger Support angeboten werden wird mit „o“ beurteilt. Wenn nur wenige Supportkanäle zur Verfügung stehen wird mit „-“ beurteilt.

5.2. Kriterien zur Benutzerfreundlichkeit

Die Bedienung eines Programmes sollte einfach sein, leicht erlernbar sein und sich an die Denk- und Arbeitsweisen der Benutzer richten (vgl. [Richter

1997], S.4). Die Arbeitsweisen bei Studierenden ändern sich im Laufe des Studiums, daher steigen mit der Zeit auch die Anforderungen auf das Lower-CASE-Tool. Tools mit geringerem Funktionsaufwand werden von Studierenden niedrigerer Semester komplexeren bevorzugt. Hingegen werden in höheren Semestern oft Tools mit einem erheblichen Funktionsumfang verwendet. Das Lower-CASE-Tool muss in der Lehre bei beiden Zielgruppen einsetzbar sein, was die Auswahl beträchtlich erschwert.

Die Verwendbarkeit der Evaluierungsobjekte sollte einerseits anhand der Gestaltung der Benutzeroberfläche (Interaktionsproblem) und andererseits bezüglich der Funktionalität beurteilt werden. Kirchner und Richter definieren derartige Kriterien zur Evaluierung der Benutzerfreundlichkeit. Die für die Evaluierung der Datenbank-Lower-CASE-Tools relevanten (zusammengefassten) Kriterien werden in den folgenden Abschnitten vorgestellt (vgl. [Richter 1997], S.4 und [Kirchner 2001], S.38f.). Dieses Hauptkriterium besitzt ein Gewicht von 40 Punkten. Diese Anzahl wird durch das Projektteam im Rahmen der praktischen Evaluierung auf die untergeordneten Evaluierungskriterien verteilt.

5.2.1. Individualisierbarkeit

Individualisierbarkeit definiert sich durch die Möglichkeit Dialoge, die Menüstruktur und Oberfläche des Tools an die jeweiligen Vorstellungen bzw. Arbeitsweisen anzupassen. Dies beinhaltet beispielsweise die freie Positionierung von Fenstern, Menü- und Symbolleisten und die Erweiterbarkeit der Funktionalität der Software durch selbstdefinierte Makros (vgl. [Kirchner 2001], S.40). In Bezug auf die Lehre kann durch einen hohen Grad an Individualisierbarkeit die Einarbeitungszeit der Studierenden verringert werden, da beim Einsatz in früheren Semestern erweiterte Funktionalitäten ausgeblendet werden können.

Beurteilung:
Für die bestmögliche Beurteilung „+“ muss die grafische Benutzeroberfläche anpassbar sein, erweiterte Funktionalität ausblendbar sein und die Erstellung von Makros bzw. die Integration von „Add-Ins“ möglich sein. Fehlt dem

Evaluierungsobjekt eine dieser Fähigkeiten resultiert der Kriterienertrag mit „o“, ansonsten mit „-“.

5.2.2. Erwartungskonformität

Ein erwartungskonformes CASE-Tool definiert sich durch die durchgehende Konsistenz aller Dialoge und Darstellungselemente (vgl. [ISO/IEC 14102:2008], S.28). Die Bedienung und Präsentation der Inhalte muss den Kenntnissen des Benutzers entgegenkommen. Ein Beispiel sind hierbei die Metaphern der Schaltflächen, die gewisse Funktionen der Software repräsentieren. Sind diese erwartungskonform, können die Benutzer anhand der grafischen Repräsentation, auf die Funktionalität dieser Schaltfläche schließen (vgl. [Kirchner 2001], S.39). Dieses Kriterium ist für den Einsatz in der Lehre von großer Bedeutung, da sich ein nicht erwartungskonformes Verhalten negativ auf den Lernerfolg auswirken kann.

Beurteilung:
Der Kriterienertrag resultiert in „+“, wenn alle Elemente der Benutzeroberfläche des Datenbank-Lower-CASE-Tools durchgehend konsistent sind bzw. die Funktionen keine unerwarteten Ergebnisse liefern. Außerdem müssen De-facto-Standards der Benutzerführung eingehalten werden. Die Beurteilung „o“ wird bis zu maximal zwei ermittelten Verstößen vergeben. Ansonsten wird mit „-“ beurteilt.

5.2.3. Selbstbeschreibungsfähigkeit / Lernförderlichkeit

Ein Tool, das einen hohen Grad an Selbstbeschreibungsfähigkeit aufweist, ermöglicht Benutzern alle – für die Arbeit relevanten – Informationen direkt aus dem Programm zu entnehmen. Diese Informationen müssen in der Benutzeroberfläche kommuniziert werden und unmittelbar verständlich sein. Die Präsentation dieser Informationen kann beispielsweise auf Verlangen des Benutzers erfolgen oder immer eingeblendet sein. Wenn die Funktionen eines Tools nur durch Ausprobieren oder Recherche in der Dokumentation verstanden werden können, hat dies negative Auswirkungen auf die Benutzerführung (vgl. [Kirchner 2001], S.38).

Da jeder Benutzer ein neues System erst einmal erlernen muss, wird die Lernförderlichkeit als einfach messbar angesehen (vgl. [Richter 1997], S.10). Es wird die Zeit gemessen, die aufgebracht werden muss, um das Tool produktiv zu nutzen (vgl. [ISO/IEC 14102:2008], S.28).

Im Bezug zur Lehre ist dieses Evaluierungskriterium besonders interessant, da nur Evaluierungsobjekte mit einem hohen Grad an Selbstbeschreibungsfähigkeit lernförderlich sind.

Beurteilung:
Für die bestmögliche Beurteilung „+“ müssen alle angezeigten Programmfunktionen in einer logischen Art und Weise gruppiert sein und „Mouse-Over“-Informationen eingesetzt werden. Zudem sollten immer nur die, für den jeweiligen Kontext wichtigen Schaltflächen angezeigt werden. Überfrachtete Benutzeroberflächen werden daher automatisch auf „o“ abgestuft. Fehlt eine der aufgezeigten Fähigkeiten resultiert der Kriterienertrag mit „-“.

5.2.4. Informationsgestaltung

Es existieren zwar Richtlinien, die konkrete Anweisungen für Bild- und Dialoggestaltung beinhalten, diese sind jedoch, aufgrund der unterschiedlichen und ständig wechselnden Präferenzen der Benutzer umstritten. Gute Standards sind nicht absolut gut, sondern stellen maximal nur immer die optimale Alternative im Vergleich zu den bekannten Gestaltungsalternativen dar (vgl. [Richter 1997], S.7). Aspekte wie Übersichtlichkeit, der richtige Einsatz von Farbe zur Gruppierung und Darstellung von wichtigen Werten können an den Evaluierungsobjekten jedoch gemessen werden (vgl. [Kirchner 2001], S.40).

In Bezug auf die Lehre fördert beispielsweise eine neutral gestaltete Benutzeroberfläche das Wohlbefinden, während für den jeweiligen Kontext relevante Informationen grafisch so aufgebreitet werden können, dass sie vom Benutzer schnell erfasst und interpretiert werden können.

Beurteilung:
Der Kriterienertrag resultiert in „+“, wenn die Übersichtlichkeit und neutrale Farbgestaltung gegeben ist, besondere Informationen farblich

hervorgehoben werden und die Benutzeroberfläche insgesamt modern wirkt. Solle das Tool veraltet erscheinen oder die Übersichtlichkeit bzw. eine gute Strukturierung nicht gegeben sein, resultiert dies in „o“. Bei gröberen Mängeln wird das Kriterium mit „-“ beurteilt.

5.2.5. Steuerbarkeit

Laut dem Grundsatz der Steuerbarkeit, sollte die Interaktionsgeschwindigkeit nicht von der Benutzeroberfläche vorgeschrieben werden, sondern sich den individuellen Fähigkeiten und Vorlieben der Benutzer einpassen. Die Benutzer müssen schnell zu den jeweilig benötigten Funktionen gelangen können und nicht durch den Programmaufbau gebremst werden (vgl. [Glier 2005], S.36). Als Indikator für die Steuerbarkeit kann folglich die Anzahl der benötigten Mausklicks herangezogen werden. Durch die Verfügbarkeit von mehreren unterschiedlichen Wegen, die schlussendlich zur benötigten Programmfunktion führen, kann der Grad der Steuerbarkeit erhöht werden.

Beurteilung:
Wenn wichtige Programmfunktionen bzw. Dialoge über mehrere Wege erreichbar sind bzw. ein einfacher Zugriff auf alle Programmkomponenten möglich ist, wird mit „+“ beurteilt. Zwingt das Programm den Benutzern eine spezielle Vorgehensweise auf oder stellt nur sehr wenige „Short-Cuts“ zur Verfügung resultiert dies automatisch in „o“. Bei mehreren oder gröberen Verstößen wird das erhält das Kriterium die Beurteilung „-“.

5.2.6. Verhalten bei Benutzerfehlern

Dieses Kriterium behandelt einerseits, in wie weit die Anwendung den Benutzer bei der Identifizierung von Fehlern und deren Behebung unterstützt, andererseits ob die Anwendung ausreichende Funktionalitäten zur Vermeidung von falschen Benutzereingaben bietet (vgl. [ISO/IEC 14102:2008], S.28). Sowohl bei Benutzer-, als auch bei Programmfehlern sollte eine umfassende Fehlermeldung die Art des Fehlers erklären und Problemlösungen präsentieren (vgl. [Kirchner 2001], S.40). Für den Einsatz in der Lehre sind klar formulierte Fehlermeldungen und Problemlösungs-

vorschläge äußerst wünschenswert, da sie das selbstständige Beheben des Probleme unterstützen.

Beurteilung:
Der Kriterienertrag resultiert in „+“ wenn die Fehlermeldung aussagekräftig ist, aus dem Dialogfenster kopiert werden kann und Fehlercodes in den Meldungen enthalten sind. Falls eines dieser Merkmale nicht erfüllt werden kann, wird mit „o“ beurteilt. Bei gröberen Mängeln wird mit „-“ beurteilt.

5.2.7. Projektfähigkeit

Ein hoher Grad an Projektfähigkeit ergibt sich durch die Bereitstellung von unterstützenden Funktionen, die eine parallele Entwicklung durch mehrere Projektmitglieder ermöglichen. Dies kann beispielsweise durch zentrale „Repositories“, „ToDo“-Listen und Versioning-Funktionalitäten erfolgen.

Beurteilung:
Der Kriterienertrag resultiert in „+“ wenn alle zuvor angeführten Fähigkeiten unterstützt werden. Für eine Beurteilung mit „o“ dürfen nur „ToDo“-Listen fehlen. Das Kriterium erhält den Wert „-“ wenn grundlegende Fähigkeiten für die Projektfähigkeit vermisst werden.

5.3. Kriterien zur Funktionalität

Dieses Hauptkriterium behandelt die funktionalen Eigenschaften, die Datenbank-Lower-CASE-Tools für den Einsatz in der Lehre besitzen müssen. Die Grundlage für die enthaltenen Evaluierungskriterien wurde im Rahmen des Kapitels 3 („Entwicklung von Datenbank-anwendungen“) erarbeitet. Als Basis für die Strukturierung wurden die Evaluierungskriterien von Jenkins herangezogen, dessen spezifizierter Kriterienkatalog zur Evaluierung von CASE-Tools innerhalb einer Lehrveranstaltung von den Studierenden angewandt wird (vgl. [Jenkins 2008], S.66).

Auf die Evaluierungskriterien werden in Folge der praktischen Evaluierung 35 Gewichtungs-Punkte verteilt. Der funktionale Bereich von Lower-CASE-Tools ist somit das zweitwichtigste Hauptkriterium.

5.3.1. Formulargenerierung

Laut Mannino sind Formulare, neben den Berichten, die Eckpfeiler der meisten Datenbankanwendungen. Formulare ermöglichen Endbenutzern eine einfachere Eingabe und Änderung von Daten ([Mannino 2007], S.9, S.353, S.359). Dieses Kriterium beschäftigt sich daher mit den Rapid-Prototyping-Fähigkeiten zur Formulargenerierung, dem Grad der Unterstützung durch „Wizards“ und der Anzahl an vordefinierten Formulararten. Eine angemessene rechnergestützte Erstellung von Formularen wirkt sich positiv auf den Lernerfolg aus, da Ergebnisse schneller erzielbar sind.

Beurteilung:
Der Kriterienertrag resultiert in „+“ wenn Methoden zur vereinfachten Generierung von unterschiedlichen Formulartypen existieren und die unfertigen Prototypen einfach erweitert bzw. verändert werden können. Fehlt nur eine dieser Fähigkeiten ist mit „o“ zu beurteilen, ansonsten mit „-“.

5.3.2. Berichtgenerierung

Für die Berichtgenerierung gelten prinzipiell die gleichen Vorgaben wie für die Formulargenerierung. Ein zusätzlicher Fokus wird auf die angebotenen Druck- bzw. Exportformate wie beispielsweise PDF (Portable Document Format) gelegt.

Beurteilung:
Das Kriterium wird mit „+“ beurteilt, wenn unterschiedliche Berichttypen durch den Schritt-für-Schritt-Wizard generiert werden können und unterschiedliche Export-Formate unterstützt werden. Der Wert „o“ wird vergeben wenn der Druck bzw. Export nur schwer möglich ist. Ansonsten auf „-“ abgestuft, da dann elementare Rapid-Prototyping-Fähigkeiten fehlen würden.

5.3.3. Design

Das Kriterium Design behandelt die Möglichkeiten der grafischen Gestaltung der End-Anwendung und behandelt Fähigkeiten wie die Definition des Grundaufbaus („Master-Layout“) für die Formulare und Berichte. Bereits

vordefinierte Vorlagen und eine erleichterte Navigationserstellung zwischen den Formularen fließen positiv in die Beurteilung ein.

Beurteilung:
Der Kriterienertrag resultiert in „+“, wenn die grafische Gesamtgestaltung der Ziel-Anwendung mit Vorlagen und „Master-Layouts“ gestaltet werden kann und die Navigationserstellung durch wenige Arbeitsschritte ermöglicht wird. Können wenige bestimmte Designelemente nur umständlich angepasst werden, können wird mit „o“, ansonsten mit „-“ beurteilt.

5.3.4. Dokumentation Entwickler/Kunde

Lower-CASE-Tools erleichtern die Erstellung der Entwicklungs-Dokumentation und der kontextsensitiven Benutzerhilfe für die Datenbank-anwendung durch vordefinierte Komponenten.

Beurteilung:
Wenn das Datenbank-Lower-CASE-Tool sowohl Dokumentierungs-möglichkeiten für den Entwicklungsprozess beinhaltet, als auch die Erstellung einer kontextsensitiven Hilfe erleichtert, wird mit „+“ beurteilt. Sollte ein Merkmal nicht enthalten sein, wird der Wert „o“ eingesetzt, ansonsten ist mit „-“ zu urteilen.

5.3.5. Unterstützte DBMS

Innerhalb dieses Kriteriums werden die standardmäßig unterstützten DBMSe der Software evaluiert. Die physische Unterstützung von verschiedenen DBMSen ist von Vorteil und fließt positiv in die Beurteilung ein. In Bezug auf die Lehre ist eine größere Flexibilität von Vorteil.

Beurteilung:
Wenn das Datenbank-Lower-CASE-Tool verschiedene DBMSe unterstützt und die Veränderung von Einstellungen direkt im Datenbank-Lower-CASE-Tool durchgeführt werden kann, wird mit „+“ beurteilt. Bei teilweiser Erfüllung dieser Merkmale wird mit „o“, ansonsten mit „-“ beurteilt.

5.3.6. Code-Erzeugung/Überprüfung

Durch „Syntax-Highlighting“ werden Schlüsselwörter einer Programmiersprache farblich hervorgehoben. „Code Completion“ stellt eine Liste von, an der aktuellen Cursorposition, möglichen Ausdrücken zur Verfügung. (vgl. [Wihsböck 2007], S.6-8, S.18).

Beurteilung:
Wird sowohl „Syntax-Highlighting“ als auch „Code Completion“ unterstützt, resultiert der Kriterienertrag in „+“. Sollten Funktionalitäten fehlen, oder beide nur teilweise unterstützt werden, wird mit „o“, ansonsten mit „-“ gewertet.

5.3.7. Erweiterte SQL Unterstützung

Dieses Kriterium umfasst die Unterstützung von PL-SQL und erweiterten Elementen wie z.B. Triggern und Procedures durch die Software. Wenn die Möglichkeit besteht die Objekte der Datenbank intuitiv in einer grafischen Oberfläche zu verändern, fließt dies positiv in die Beurteilung ein.

Beurteilung:
Werden erweiterte Elemente von PL/SQL, wie Trigger und Procedures unterstützt und ist ein Objektbrowser für alle Objekte des Datenbanksystems verfügbar, wird mit „+“ gewertet. Bei nur teilweiser Unterstützung ist mit „o“ zu urteilen, ansonsten mit „-“.

5.3.8. Debugging

Debugging-Fähigkeiten erlauben die Überprüfung und Manipulation des Codes zur Laufzeit der Anwendung (vgl. [Wihsböck 2007], S.6-8, S.18).

Beurteilung:
Der Kriterienertrag ist mit „+“ festzulegen wenn Debugging-Funktionalität sowohl für die Backend- als auch für die Frontend-Sprachen zur Verfügung steht und der Code während der Ausführung manipuliert werden kann. Wird dies nur teilweise unterstützt, wird das Kriterium mit „o“ beurteilt, bei keiner Unterstützung von Debugging-Funktionalitäten wird der Wert „-“ angewandt.

5.3.9. Zielplattformen

Da die plattformunabhängige Programmierung, vor allem bei Datenbankanwendungen, ein wichtiger Lehrbereich an der FH-Joanneum ist, beleuchtet dieses Kriterium die unterstützten Zielplattformen. Die gleichzeitige Entwicklung von „Thick“- und „Thin“-Clients ist ebenfalls Thema dieses Evaluierungskriteriums.

Beurteilung:
Wenn die gleichzeitige Entwicklung für unterschiedliche Zielplattformen sowohl als „Thick“- als auch als „Thin“-Client möglich ist, oder die Ziel-Applikation plattformunabhängig ist, wird mit „+“ gewertet. Trifft dies nur teilweise zu, wird mit „o“ beurteilt. Wenn nur ein bestimmtes Zielsystem möglich ist, wird der Ertrag mit „-“ festgelegt.

6. Resümee

Im Rahmen dieser Arbeit wurde der Software- bzw. Datenbank-Entwicklungsprozess erläutert und mit CASE in Verbindung gebracht. Zur Identifizierung der Merkmale eines Datenbank-Lower-CASE-Tools wurde die Architektur von DBSen beleuchtet und die Benutzeroberfläche einer Datenbankanwendung, mit ihren grafischen Komponenten beschrieben. Um eine zielgerichtete und methodische Vorgehensweise zu erreichen, wurden die Aspekte der allgemeinen und der technischen Evaluierung miteinander vereint und zusätzlich die Nutzwertanalyse zur Bestimmung des optimalen Evaluierungsobjekts definiert.

Durch Anwendung der konzeptionellen Phasen des Vorgehensmodells konnte der Kriterienkatalog für Datenbank-Lower-CASE-Tools, unter Einbeziehung der zuvor identifizierten Merkmale, spezifiziert werden. In den nächsten Abschnitten werden die ermittelten Evaluierungskriterien dargestellt und die Ergebnisse diskutiert.

6.1. Übersicht des Kriterienkatalogs

Abbildung 10 zeigt alle Kriterien der drei Hauptbereiche in einer schematischen Übersicht.

Allgemeine Kriterien	***Kriterien zur Benutzerfreundlichkeit***	***Kriterien zur Funktionalität***
Kosten	Individualisierbarkeit	Formulargenerierung
Lizenzmodell und Softwarevarianten	Erwartungskonformität	Berichtgenerierung
Systemvoraussetzungen	Selbstbeschreibungsfähigkeit /Lernförderlichkeit	Design
Installation	Informationsgestaltung	Dokumentation Entwickler/Kunde
Programmdokumentation	Steuerbarkeit	Unterstützte DBMS
Support	Verhalten bei Benutzerfehlern	Code-Erzeugung/Überprüfung
	Projektfähigkeit	Debugging
		Zielplattformen

Abbildung 10: Kriterienübersicht

Das Ergebnis dieser Arbeit konnte somit geliefert werden, der Einleitung der praktischen Evaluierung steht nichts mehr im Wege.

6.2. Diskussion

Insgesamt kann die Definition des Vorgehensmodells und die methodische Durchführung der einzelnen Schritte, als eine äußerst komplexe Aufgabe betrachtet werden. Diese wurde durch den bestehenden Mangel an verfügbarer Literatur zu Datenbank-Lower-CASE-Tools weiter verschärft. Durch die systematische Beleuchtung der Themengebiete und der Darstellung der relevanten Aspekte, konnte diese Problematik jedoch überwunden werden.

Das Aufstellen der Evaluierungskriterien ist mit einem nicht zu unterschätzenden Zeitaufwand verbunden. Zwar gibt es zu allgemeinen Kriterien und zu Kriterien für die Benutzerfreundlichkeit eine große Anzahl an verfügbarer Literatur, die Kriterien mussten jedoch erst mit dem sehr speziellen Bereich Datenbank-Lower-CASE in Verbindung gebracht und auf den Lehrfokus angepasst werden. Besonders zur Bestimmung der Kriterien zur Funktionalität von Datenbank-Lower-CASE-Tools war die detaillierte Spezifizierung des Datenbankanwendungsentwicklungsprozesses unabdingbar.

Die Methodik der Evaluierung wird mit dem erarbeiteten Vorgehensmodell sehr genau beschrieben. Dieses Modell könnte daher auch für andere Evaluierungen verwendet werden. Durch die Einführung der 100-Punkte-Gewichtungsskala in Verbindung mit der Methode des Paarvergleichs, wurde eine flexible Lösung für die Gewichtung der einzelnen Kriterien gefunden, die den Grad der Subjektivität verringert. Das Ergebnis der Evaluierung kann durch Neuverteilung der Gewichte für eine andere Präferenzordnung angepasst werden. Dies erhöht den Wert für die FH-Joanneum, falls die Evaluierungsergebnisse für Auswahlentscheidungen herangezogen werden müssen.

Literaturverzeichnis

Bücher:

[Hansen 2009]
Hansen, Hans Robert; Neumann, Gustav: Wirtschaftsinformatik 1 – Grundlagen und Anwendungen, 10. Aufl., UTB, 2009

[Lutz 2009]
Lutz, Heinrich; Stelzer, Dirk: Informationsmanagement. Grundlagen, Aufgaben, Methoden, 9. Aufl., Oldenbourg Verlag München 2009

[Mannino 2007]
Mannino, Michael V.: Database Design, Application Development, and Administration, 3. Aufl., McGraw-Hill/Irwin, 2007

[Sommerville 2007]
Sommerville, Ian: Software Engineering, 8. Aufl., Addison-Wesley, 2007

[Störrle 2005]
Störrle, Harald: UML 2 für Studenten, 1. Aufl., Pearson Studium Verlag, 2005

[Vossen 2000]
Vossen, Gottfried: Datenmodelle, Datenbanksprachen und Datenbank-managementsysteme, 4. Aufl., Oldenbourg Wissenschaftsverlag, 2000

[Weber 2005]
Weber, J.: Die Nutzwertanalyse zur Beurteilung von Entscheidungs-alternativen im öffentlichen Sektor, 1. Aufl., GRIN Verlag, 2005

Conference Papers und Zeitschriftenartikel:

[Chatterjee 2004]
Chatterjee, Ramkrishna; Arun, Gopalan; Agarwal, Sanjay; Speckhard, Ben; Vasudevan, Ramesh: Using Data Versioning in Database Application Development, in: Proceedings of the 26th International Conference on Software Engineering, S.315-325, 2004

[Chen 1976]
Chen, Peter Pin-Shan: The Entity-Relationship Model – Toward a Unified View of Data, in: ACM Transactions on Database Systems, 1(1), S.9-36, 1976

[Chen 1989]
Chen, Minder; Nunamaker, Jay F.; Weber, E. Sue: Computer-aided software engineering: present status and future directions, in: SIGMIS Database, 20(1), S.7-13, 1989

[Forte 1992]
Forte, Gene; Norman, Ronald J.: A self-assessment by the software engineering community, in: Communications of the ACM, Volume 35, Issue 4, S.28-32, 1992

[Frank 2006]
Frank, Ulrich: Evaluation von Forschung und Lehre an Universitäten, in: ICB-Research Report, No.6, Universität Duisburg-Essen, 2006

[Fuggetta 1993]
Fuggetta, Alfonso: A Classification of CASE Technology, in: Journal Computer, Volume 26, Issue 12, S.25-38, 1993

[Jenkins 2008]
Jenkins, Marcelo: Teaching Computer Aided Software Engineering at a Graduate Level, in: Proceedings of the 13th annual conference on Innovation and technology in computer science education, Madrid, S.63-67, 2008

[Lending 1998]
Lending, Diane: The use of CASE tools, in: Proceedings of the 1998 ACM SIGCPR conference on Computer personnel research, New York, S.49-58, 1998

[Nüttgens 2002]
Nüttgens, M.: Rahmenkonzept zur Evaluierung von Modellierungs-werkzeugen zum Geschäftsprozessmanagement, in: Informationssystem Architekturen, Wirtschaftsinformatik Rundbrief der GI Fachgruppe WI-MobiS, 9(2), S. 101-111, 2002

[Post 1998]
Post, Gerald; Kagan, Albert; Keim, Robert T.: A comparative evaluation of CASE tools, in: Joural of Systems and Software, Volume 44, Issue 2, S.87-96, 1998

[Saiedian 1997]
Saiedian, Hossein: An evaluation of extended entity-relationship model, in: Information and Software Technology, Volume 39, Number 7, S.449-462, 1997

[Song 1995]
Song, Il-Yeol; Evans, Mary; Park, E.K.: A Comparative Analysis of Enitity-Relationship Diagrams, in: Journal of Computer and Software Engineering, 3(4), S.427-459, 1995

Papers:

[Atlas 1999]
Atlas, Nermin; Wolf, Steven; Bühr, Ralf: CASE-Tools zur Entity-Relationship-Modellierung - Eine Bibliographie und Sammlung von Werkzeugen, Semesterarbeit, FH-Darmstadt, 1999

[Beckworth 1993]
Beckworth, Geoff: Selection Criteria for CASE Tools, Deakin University, Australia, 1993

[Glier 2005]
Glier, M.: Systemantwortzeiten als Aspekt der Software-Ergonomie und der Wirtschaftsinformatik, Universität Hamburg, Deutschland, 2005

[ISO/IEC 14102:2008]
ISO/IEC 14102:2008: Information technology - Guideline for the evaluation and selection of CASE tools, 2. Aufl., 2008

[Kirchner 2001]
Kirchner, Lutz: Entwicklung und Anwendung eines Bezugsrahmens zur Evaluierung von UML-Modellierungswerkzeugen, Diplomarbeit im Studiengang Informatik, Universität Koblenz-Landau, 2001

[Pleumann 2007]
Pleumann, Jörg: Ein Ansatz zur Entwicklung von Modellierungswerkzeugen für die softwaretechnische Lehre, in Lehrstuhl für Software-Technologie, Dortmund, Lehrstuhl 10, Memo 166, 2007

[Richter 1997]
Richter, M.: Kriterien der Benutzerfreundlichkeit, Universität Zürich, Schweiz, 1997

[Stamelos 2000]
Stamelos, Ioannis; Vlahavas, Ioannis; Refanidis, Ioannis; Tsoukiàs, Alexis: Knowledge Based Evaluation of Software Systems. A Case Study, Aristotle University of Thessaloniki 2000

[Wihsböck 2007]
Wihsböck, Michael: Moderne Softwareentwicklungsumgebungen. Evaluierung Java-basierter Ansätze. Diplomarbeit Technische Universität Wien 2007

Internetquellen:

[https://www.bsi.bund.de]
[1] BSI Bund Beschreibung Datenbanksysteme; https://www.bsi.bund.de/ContentBSI/grundschutz/kataloge/baust/b05/b05007.html; 02.04.2011

[http://msdn.microsoft.com]
[1] Entwickeln einer Datenbankanwendung (MFC); http://msdn.microsoft.com/de-de/library/cc438658(v=vs.71).aspx; 03.04.2011

[http://www.npd-solutions.com]
[1] Computer Aided Software Engineering; http://www.npd-solutions.com/case.html; 02.04.2011

[http://www.oracle.com]
[1] Oracle APEX Overview; http://www.oracle.com/technetwork/developer-tools/apex/overview/index.html; 07.05.2011

[2] Oracle Query Builder;
http://download.oracle.com/docs/cd/E10513_01/doc/appdev.310/e12857/qry_bldr.htm#CHDJFGBE; 09.05.2011

[3] Oracle APEX Features;

http://apex.oracle.com/pls/otn/f?p=4600:6:0; 10.05.2011

[4] Oracle APEX Building an Application;
http://download.oracle.com/docs/cd/B25329_01/doc/appdev.102/b25309/bldapp.htm; 10.05.2011