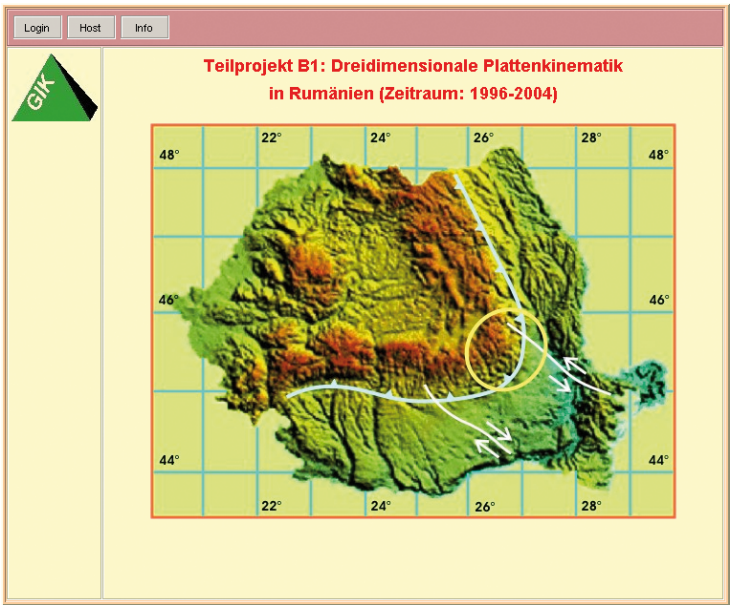
Michael Nutto

Internetbasiertes Fachinformationssystem zur Plattenkinematik



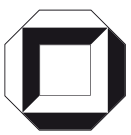
universitätsverlag karlsruhe

Michael Nutto

Internetbasiertes Fachinformationssystem zur Plattenkinematik

Internetbasiertes Fachinformationssystem zur Plattenkinematik

von
Michael Nutto



universitätsverlag karlsruhe

Dissertation, Universität Karlsruhe (TH)
Fakultät für Bauingenieur-, Geo- und Umweltwissenschaften, 2006
Referenten: Prof. Dr.-Ing. Dr.-Ing. E.h. G. Schmitt, Prof. Dr. rer. nat.W. Stucky

Impressum

Universitätsverlag Karlsruhe
c/o Universitätsbibliothek
Straße am Forum 2
D-76131 Karlsruhe
www.uvka.de

Universitätsverlag Karlsruhe 2006
Print on Demand

ISBN-13: 978-3-86644-063-0
ISBN-10: 3-86644-063-4

Internetbasiertes Fachinformationssystem zur Plattenkinematik

Zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS

von der Fakultät für

Bauingenieur-, Geo- und Umweltwissenschaften
der Universität Fridericiana zu Karlsruhe (TH)

genehmigte

DISSERTATION

vorgelegt von

Dipl.-Ing. Michael Nutto
aus Freiburg i. Br.

Tag der mündlichen Prüfung:	**13.01.2006**
Hauptreferent:	**Prof. Dr.-Ing. Dr.-Ing. E.h. G. Schmitt**
Korreferent:	**Prof. Dr. rer. nat. W. Stucky**

Karlsruhe (2006)

Kurzfassung

In der vorliegenden Arbeit wird ein internetbasiertes Fachinformationssystem (FIS) zur Detektion krustaler Deformationen in den Ostkarpaten vorgestellt, welches im Rahmen des Sonderforschungsbereichs 461 „Starkbeben: von wissenschaftlichen Grundlagen zu Ingenieurmaßnahmen“ entwickelt wurde.

Im Vordergrund stehen die Konzeption und Implementierung eines Fachinformationssystems für regionale beziehungsweise globale GPS-Projekte, die zur Aufdeckung von Plattengrenzen und Analyse von Plattenbewegungen eingerichtet wird. Dabei werden verschiedene Aspekte berücksichtigt. Der Zugang zu relevanten Projektinformationen via Internet soll einer breiten Öffentlichkeit unter besonderer Berücksichtigung der verschlüsselten Datenübertragung bereitgestellt werden. Eine sichere und zuverlässige Datenverwaltung wird mittels einer objektrelationalen Datenbank (DB) realisiert. Zur schnellen und sicheren Kommunikation zwischen Client und Server dient der Aufbau einer plattformunabhängigen, mehrschichtigen Architektur. Es wird ein Applikationsserver (AS) eingerichtet, der ausführungsunabhängige Projektkomponenten für die Entwicklung von Objekten und Methoden komplexer Geschäftsanwendungen zur Verfügung stellt. Ein einheitliches Austauschformat soll zur Datenübertragung mittels eines standardisierten Werkzeuges (XML) zur Verwendung von Auszeichnungssprachen entworfen werden. Wesentliche Ergebnisse aus den geodätischen Analysen und Auswertungen liegen zur graphischen Darstellung im FIS unter Verwendung der Java 2D Anwendungsprogrammierschnittstelle (API) vor.

Unter Berücksichtigung der oben genannten Aspekte wird ein Fachinformationssystem zur Dokumentation des 9-jährigen Projektverlaufs (1996-2004) im Teilprojekt B1 „Dreidimensionale Plattenkinematik in Rumänien“ aufgebaut, welches sich aus nicht kommerziellen Komponenten (Datenbank, Applikationsserver, Web-Server, Applet) zusammensetzt.

Abstract

An Internet based information system will be presented in this work to detect crustal movements in the Eastern Carpathians which is developed within the scope of the collaborative research center 461 „Strong Earthquakes: a Challenge for Geosciences and Civil Engineering“.

The focus is the concept and the implementation of an information system for regional respectively global GPS-projects, which are installed to detect plate borders and to analyse plate motions. In consideration of several aspects the following is realised: the access to relevant project information is granted via Internet for the general public taking into account the encrypted data transfer. A confident and reliable data management is implemented using an object-relational data base. A platform independent three-tier architecture is created to get a fast and protected communication between client and server. For that matter, an application server is used which provides implementation-independent project components to develop objects and methods for complex business applications. An uniform interchange format is integrated by using a standardised tool to design markup languages. Important results will be presented from geodetic analyses and evaluations using proper applications and visualisations.

For that reason, an information system is realised for documentation of the nine years old course of the project (1996-2004) „Three-dimensional plate kinematics in Romania“, which is modeled and build up by non commercial components (data base, application server, web server, applet).

Beschreibungen und Symbole

Alle Programmbeispiele in Kapitel 2 beziehen sich auf die Datenbank von *Oracle - Version 9i*. Daher können Abweichungen im Programmcode im Vergleich zu relationalen Open-Source-Datenbanken (MySQL, Firebird, MaxDB, PostgreSQL,...) auftreten. Der Open-Source-Applikationsserver *JBoss - Version 3.2* wird für die Programmbeispiele in Kapitel 3 und 4 herangezogen. Zum Aufbau der graphischen Bedieneroberfläche wurde die *Java Standard Edition 1.4*, zur Entwicklung der Enterprise JavaBeans die *EJB 2.0* API und die Plattform *Java 2 Enterprise Edition Version 1.3* verwendet.

Wichtige Begriffe und Fachausdrücke aus dem Bereich der Informatik werden bei ihrer ersten Verwendung kursiv geschrieben.

Symbole

$mod(n)$	:	modulo n
$diag(\lambda_i)$	:	Diagonalmatrix mit $i = 1,...,n$ Eigenwerten
$det(A)$	:	Determinante einer Matrix A
l	:	geodätische Beobachtung
υ	:	Verbesserung der geodätischen Beobachtung
μ	:	Erwartungswert $\mu = E(x)$ einer Zufallsvariable x
t_0	:	Bezugsepoche zum Zeitpunkt t = 0
σ	:	Standardabweichung
σ_0	:	A priori Varianzfaktor
$\sigma_B, \sigma_L, \sigma_h$	:	Standardabweichung der Stationskoordinaten in Breite, Länge und Höhe
σ_v	:	Standardabweichung der Geschwindigkeit v
α	:	Irrtumswahrscheinlichkeit
$c_{(1-\frac{\alpha}{2})}$	:	Quantile c
$N(\mu, \sigma_0^2)$	:	Normalverteilung, festgelegt durch den Erwartungswert und der Varianz
B, L	:	geographische Koordinaten
h	:	ellipsoidische Höhe
ω	:	Winkelgeschwindigkeit
v_h	:	vertikale Geschwindigkeit
ε	:	Exzentrum
o	:	Offset
H_0	:	Nullhypothese
H_A	:	Alternativhypothese
Θ	:	Signifikanzschwelle
$Q_{\hat{x}\hat{x}}$	:	Kofaktormatrix des unbekannten Koordinatenvektors $\hat{x}$
N	:	Normalgleichungsmatrix
A	:	Designmatrix
P	:	Gewichtsmatrix
$P(\%)$	:	Wahrscheinlichkeit
$C_{\hat{x}\hat{x}}$	:	Varianz-Kovarianzmatrix (VKM)
E	:	Einheitsmatrix
M	:	Modalmatrix
GF	:	Grober Fehler
Λ	:	Spektralmatrix der Eigenwerte
λ	:	Eigenwert
η_i	:	Normierter Eigenvektor des i-ten Eigenwertes
ϑ	:	Schwachform-Vektor
$\|\|\Delta v_i\|\|$	:	Betrag des Geschwindigkeitsdifferenzvektors

Abkürzungsverzeichnis

ADO	:	ActiveX Data Object
AES	:	Advanced Encryption Standard
API	:	Application Programming Interface
AS	:	Applikationsserver
ASP	:	Active Server Pages
AWT	:	Abstract Windowing Toolkit
BFILE	:	Binary File LOB
BLOB	:	Binary Large Object
BMP	:	Bean Managed Persistence
CMP	:	Container Managed Persistence
CORBA	:	Common Object Request Broker Architecture
DBA	:	Datenbankadministrator
DBMS	:	Datenbankmanagementsystem
DCL	:	Database Control Language
DCOM	:	Distributed Component Object Model
DDL	:	Data Definition Language
DML	:	Data Manipulation Language
DMT	:	Disaster Management Tool
DOM	:	Document Object Model
DSA	:	Digital Signature Algorithm
DSPL	:	Database Stored Procedure Language
EIS	:	Enterprise Information System
EJB	:	Enterprise JavaBeans
ENC	:	Environment Naming Context
ERP	:	Enterprise Resource Planning
FIS	:	Fachinformationssystem
GIS	:	Geographisches Informationssystem
GPS	:	Global Positioning System
GUI	:	Graphical User Interface
HTML	:	Hypertext Markup Language
HTTP	:	HyperText Transfer Protocol
IDAPI	:	Independent Database Application Programming Interface
IEF	:	Integrity Enhancement Features
IERS	:	International Earth Rotating and Reference System Service
IIOP	:	Internet InterOrb Protocol
IGS	:	International GPS Service
IPX	:	Internetwork Protocol Exchange
ISES	:	Netherlands Research Center for: Integrated Solid Earth Sciences
ISO	:	International Organisation for Standardization
ITRF2000	:	International Terrestrial Reference Frame 2000
J2EE	:	Java 2 Enterprise Edition
J2SE	:	Java 2 Standard Edition
JAAS	:	Java Authentication and Authorization Service
JAXP	:	Java API für XML-Prozesse
JCA	:	Java Connection Architecture
JDBC	:	Java Database Connectivity
JDOM	:	Java Document Object Model
JMS	:	Java Message Service
JNDI	:	Java Naming and Directory Interface
JRE	:	Java Runtime Environment
JSP	:	Java Server Pages
JTA	:	Java Transaction API
KB	:	Kilo Byte
KI	:	Künstliche Intelligenz
LAN	:	Local Area Network
LOC	:	Locator

MB	:	Mega Byte
MOM	:	Message Oriented Middleware
NDIS	:	Network Driver Interface Standard
NetBEUI	:	NetBios Extended User Interface
NetBIOS	:	Network Basic Input/Output System
ODBC	:	Open Database Connectivity
ODI	:	Open Datalink Interface
OEM	:	Original Equipment Manufacturer
OID	:	Objektidentifikator
ORDBMS	:	Objektrelationales Datenbankmanagementsystem
OWL	:	Web Ontology Language
P3P	:	Platform for Privacy Preferences Project
PGA	:	Process Global Area
PHP	:	Hypertext Preprocessor
PKI	:	Public Key Infrastructure
PPP	:	Point-to-Point Protocol
RDBMS	:	Relationales Datenbankmanagementsystem
RDF	:	Resource Description Framework
RMI	:	Remote Method Invocation
RSA	:	Rivest Shamir Adleman-Verfahren
SAX	:	Simple API for XML
SDO	:	Spatial Data Object
SFB 461	:	Sonderforschungsbereich 461
SGA	:	System Global Area
SMIL	:	Synchronized Multimedia Integration Language
SOAP	:	Simple Object Access Protocol
SPX	:	Sequenced Packed Protocol Exchange
SPI	:	Service Provider Interface
SQL	:	Structured Query Language
SSH	:	Secure Shell
SSI	:	Server Side Includes
SSL	:	Secure Sockets Layer
SVG	:	Scalable Vector Graphics
TCP/IP	:	Transmission Control Protocol/Internet Protocol
TrAX	:	Transformations-API für XML
UDT	:	User defined Data Types
URL	:	Uniform Resource Locator
VARRAY	:	Variables Array
VKM	:	Varianz-Kovarianzmatrix
W3C	:	World Wide Web Consortium
WAN	:	Wide Area Network
WAP	:	Wireless Application Protocol
WGS84	:	World Geodetic System 1984
WSDL	:	Web Service Definition Language
X.25	:	Standard-Protokoll
XML	:	Extensible Markup Language
XSL(T)	:	Extensible Stylesheet Language (Transformation)

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Der Sonderforschungsbereich 461 *„Starkbeben: von geowissenschaftlichen Grundlagen zu Ingenieurmaßnahmen“* wurde im Jahre 1996 gegründet, um drei Forschungsschwerpunkte im Bereich Geowissenschaften und Bauingenieurwesen zu etablieren:

1. Entwicklung eines geodynamischen Modells (Sperner et al., [74], S. 51-84) unter Berücksichtigung der tektonischen Ursachen der mitteltiefen Erdbeben in der Vrancea-Region (Rumänien), der Bewegungen und der Deformationen der vier aufeinander treffenden kontinentalen Einheiten in dieser Konvergenzzone.
2. Realisierung eines ShakeMap (Kienzle et al., [48]) unter Berücksichtigung der Bodenbedingungen sowie des Boden-Bauwerk-Verhaltens zur Erstellung realistischer Schadensprojektionen.
3. Aufbau eines Disastermanagement-Tools (DMT) für Starkbeben zur Unterstützung der Katastropheneinsatzleitung und ihrer Einsatzkräfte (Gehbauer et al., [24]).

Im Rahmen des SFB 461 wurde das Teilprojekt B1 „Dreidimensionale Plattenkinematik in Rumänien“ gegründet, welches sich mit der Bestimmung dreidimensionaler Bewegungen und Deformationen der Erdkruste in Rumänien durch wiederholte Messung eines GPS-Überwachungsnetzes beschäftigt (Dinter et al., [20]). Im Mittelpunkt der geowissenschaftlichen Forschung steht die Erfassung der Krusten- und Mantelstruktur der Vrancea-Region und unter dem durch Starkbeben besonders gefährdeten Vorland des Karpatenbogens. Ein Hauptziel des Teilprojektes ist deshalb, einen wesentlichen Beitrag zur Gefährdungsabschätzung der Vrancea-Region mittels Bestimmung krustaler Deformationen zu liefern (Abb. 1.1). Dabei ist es aufgrund des Einsatzes von GPS möglich, Aussagen über rezente Plattenkinematik zu treffen und somit ein besseres Verständnis der tektonischen Prozesse in Rumänien zu erhalten. Aufgrund der speziellen Situation, ein in nahezu vertikaler Stellung versteilter Slab im südöstlichen Bogen und der kontinentalen Kollision in den Karpaten, ergeben sich für das Projekt folgende Problemstellungen:

- Erfassung relativer Bewegungsabläufe,
- Nachweis von Plattendeformationen,
- Untersuchungen zur Genauigkeit und Zuverlässigkeit der GPS-Auswertungen,
- Analysen im Bereich Qualität, Deformation und Schwachform,
- Visualisierung der Ergebnisse zur Interpretation und Präsentation der rezenten Plattenkinematik.

Seit Bestehen des SFB 461 wurden insgesamt zwölf Kampagnen durchgeführt. Aufgrund dieser großen Datenbasis und den obengenannten Anforderungen ergaben sich für die Doktorarbeit folgende Ziele zur Entwicklung eines internetbasierten Fachinformationssystems zur dreidimensionalen Plattenkinematik in den Ostkarpaten:

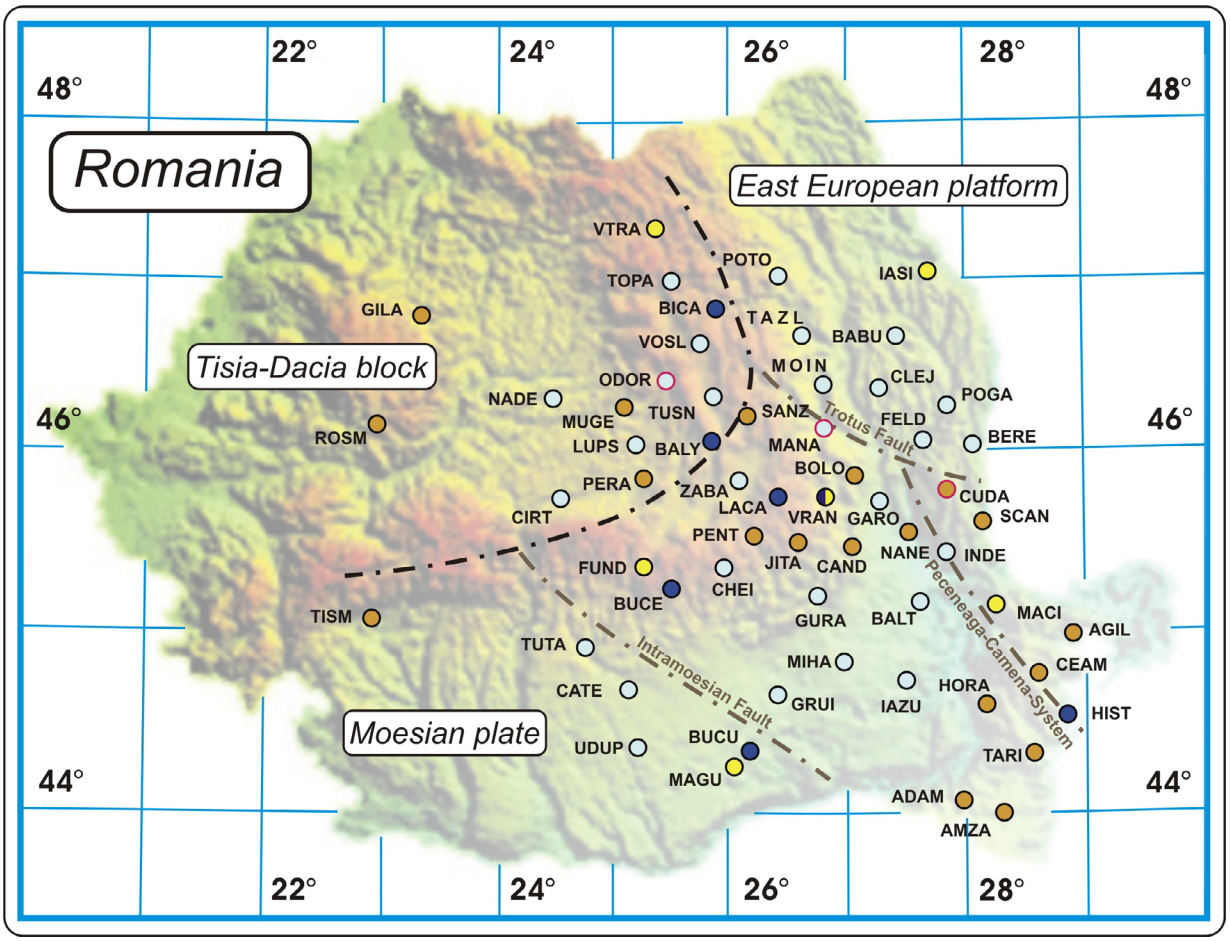


Abbildung 1.1: *GPS-Überwachungsnetz 2004 in Rumänien. Das geodätische Netz mit einer Ausdehnung von* 600×400 *(km*^{2}*) umfasst sieben Permanentstationen und 49 vermarkte Objektpunkte. Die vier tektonischen Einheiten bestehen aus der Ost-Europäischen Plattform, dem Tisia-Dacia-Block, der Moesischen Plattform und einer Mikroplatte, die durch ein System von Störungszonen (Trotus Verwerfung, Peceneaga-Camena-System und Intramoesische Störung) gebildet wird. Die mitteltiefen Erdbeben treten konzentriert im Karpaten-Bogen auf.*

- effiziente Datenverwaltung,
  - Vermeidung redundanter Datenhaltung,
  - Strukturierung und Standardisierung,
  - zuverlässige Projektdokumentation,
  - Berücksichtigung zeitlicher Veränderungen,
  - schnelle Bereitstellung relevanter Projektinformationen,
- sichere Erfassung und Validierung der erforderlichen Auswertungen und Analysen,
  - Aufbau eines projektbezogenen Qualitätsmanagements,
  - Darstellung der Ergebnisse mit modernen Visualisierungsmethoden,
  - langfristige Archivierung der Projektinformationen,
- Bereitstellung der Daten,
  - autorisierter Zugriff auf die Datenbank,
  - Realisierung eines sicheren Internetportals,
  - Aufbau einer modularen Software-Infrastruktur.

Dabei wurde der Schwerpunkt auf ein internetbasiertes Fachinformationssystem gelegt, welches sich aus nicht kommerziellen Komponenten zur Dokumentation des 9-jährigen Projektverlaufs zusammensetzt.

Allgemein läßt sich ein FIS als ein *Geographisches Informationssystem* (GIS) beschreiben, in dem fach- oder projektspezifische Komponenten die relevante Datenbasis neben den Geodaten bilden. Das Geographische Informationssystem wird folgendermaßen definiert (Bill/Fritsch, [7], S. 5):
„Ein Geo-Informationssystem ist ein rechnergestütztes System, das aus Hardware, Software, Daten und den Anwendungen besteht. Mit ihm können raumbezogene Daten digital erfasst und redigiert, gespeichert und reorganisiert, modelliert und analysiert sowie alphanumerisch und graphisch präsentiert werden."

Sinngemäß nach ISO 19104, ([34]): *„Das GIS ist ein digitales, wissensbasiertes System. Es nutzt raumbezogene Fakten und Regeln und liefert Informationen zur Unterstützung der Entscheidungsfindung."*

Als Datenbasis dienen raumbezogene Informationen, die Objekte und Eigenschaften der realen Welt sowie deren Beziehung zueinander repräsentieren. Das GIS besteht aus Software- und Hardwarekomponenten, wobei die Datenbank den zentralen Bestandteil der Datenverwaltung bildet. Die Methoden zur Analyse der Daten sowie Schnittstellen zum Datenaustausch und die Beziehungen untereinander bilden das Fundament eines Informationssystems. Mittels geeigneter Strukturierung und Präsentation dienen diese Systeme dem Menschen zur schnelleren Entscheidungsfindung. Durch die Abbildung der raumbezogenen und fachspezifischen Daten in Relationen und Knoten entsteht unter Einbeziehung expliziter Methoden zur Wissensrepräsentation ein wissensbasiertes System, welches sehr flexibel auf Änderungen des Modells reagieren kann. Dabei wird bei der Repräsentation zwischen dem Modell und den Methoden zur Wissensnutzung getrennt (Quint, [67], S. 36-39).

Ziel eines Fachinformationssystems ist es, Kompetenz- und Dienstleistungsnetze über ein Internetportal der Öffentlichkeit zur Verfügung zu stellen. Insbesondere sollen fachspezifische Daten zentralisiert, strukturiert, mittels Metadaten normiert, erfasst und langfristig archiviert werden. Die Methoden und Anwendungen umfassen folgende Bereiche: Metadaten, Wissensmanagement, Wissensrepräsentation und Visualisierung. Ihre Kombinationen stellen dem Benutzer bestmöglich und effizient Informationen zur Verfügung.

Stand der Forschung

Aufgrund der gestiegenen Anforderung, Informationen schnell, flexibel und zuverlässig einer großen Anzahl von Benutzern bereitzustellen, liegen die Forschungsschwerpunkte einerseits in der Entwicklung von Verfahren zum Aufbau von Expertensystemen zur wissensbasierten Informationsanalyse z.B. durch Nutzung von *Künstlicher Intelligenz* (KI) oder *neuronaler Netze* und andererseits in der Realisierung von Systemarchitekturen, um komplexe Geschäftsprozesse abzubilden, Kommunikationsprozesse zu unterstützen, Wissenskomponenten zu strukturieren, ein effektives Datenmanagement aufzubauen, einen Datentransfer in Echtzeit zu ermöglichen und dynamische Komponenten zu entwerfen. Besonders im Bereich der Transformation von Bild und Sprache (Bähr/Vögtle, [6]) sowie der Verknüpfung raumbezogener Daten mit der zeitlichen Dimension (Zipf/Krüger, [90]) werden theoretische Konzepte und Modelle zur Lösung von Entscheidungsproblemen herangezogen.

Expertensysteme sind *„Programme, mit denen das Spezialwissen und die Schlussfolgerungsfähigkeit qualifizierter Fachleute auf eng begrenzte Aufgabengebiete nachgebildet werden soll"* (Puppe, [66]). Der Vorteil dieser Systeme liegt darin, Regeln aufzustellen und Algorithmen zu implementieren, die als unabhängiger Bestandteil des Expertensystems gelten und flexibel auf Veränderungen des Analyseprozesses reagieren (Brezing/Benning, [11], S. 206-217). Während in konventionellen Regelungsprogrammen Wissen und Entscheidungsfindung unmittelbar miteinander verknüpft sind, trennt man beim Expertensystem die Bereiche des fachlichen Wissens (Wissensbasis) und des Wissens über das Vorgehen zum Lösen von Problemen (Inferenz). In Abbildung 1.2 wird schematisch ein Expertensystem vorgestellt.

Das Expertenwissen liefert Modelle und Methoden zur Datenanalyse, in denen unterschiedliche Problemlösungstypen und -strategien zur optimalen Entscheidungsfindung integriert werden können. Die Strategien wer-

den aus der Wissensbasis abgeleitet, deren Komponenten den Erwerb und die Verarbeitung von Wissen umfassen. Die Interpretation und Strategie zur Problemlösung werden dokumentiert und langfristig gesichert.

Die Wissensverarbeitung setzt sich aus den Bereichen *Wissenserwerb*, *Wissensinterpretation*, *Wissensformalisierung* und *Wissensrepräsentation* zusammen und wird in der Datenbank durch Fakten und Regeln definiert. Der Wissenserwerb erfolgt durch einen Experten, der sein Fachwissen über eine Wissenserwerbskomponente formalisiert. Eine erfolgreiche Interpretation von raum- oder sachbezogenen Informationen setzt voraus, dass dieses System nach Erfassung der Daten in der Lage ist, erstens den Informationsgehalt zu extrahieren, zweitens zu verstehen (Plausibilitätskontrollen, Vollständigkeit), drittens zu erkennen (Eigenschaften, Typ), viertens zu analysieren (Zuordnung der Eigenschaften), fünftens zu lernen (Anpassung an aktuelle Veränderungen) und letztendlich daraus Schlüsse zu ziehen (Modellwahl, Problemlösung, Interpretation). Die Darstellung komplexen Wissens erfolgt über spezielle Methoden, welche z.B. mit Frames, semantischen Netzen, regelbasierten Systemen oder KI-Programmiersprachen erzeugt werden.

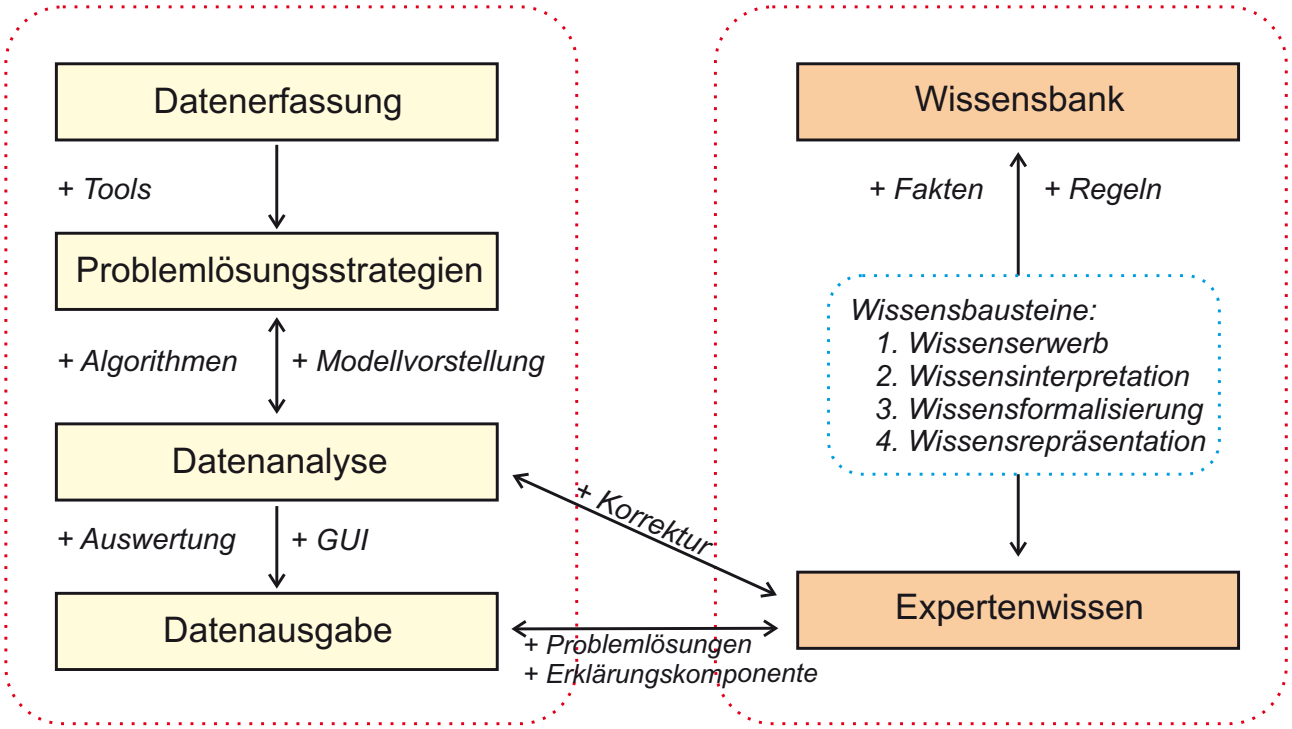


Abbildung 1.2: *Schematische Darstellung des Programmaufbaus eines Expertensystems zur wissensbasierten Informationsanalyse.*

Systemarchitekturen *beschreiben Komponenten des Systems in Bezug auf Datenmodelle, Datenmanagement, angewandte Funktionen und Regeln, Schnittstellen zu umgebenden Systemen und Anwendern sowie deren Beziehungen untereinander*. Die modulare Architektur eines Fachinformationssystems besteht aus Kernkomponenten, die sich folgendermaßen unterteilen lassen:

i. *Datenerfassung und Datenmodellierung:* Geeignete Strukturierung der heterogenen Daten. Aufbau funktionaler Beziehungen untereinander. Planung und Definition von Anforderungen und Argumenten unter Verwendung einer rechnergestützten Datenverwaltung. Beseitigung von Widersprüchen und Kontrolle der Datenqualität. Spezifizierung öffentlich zugänglicher, interoperabler Lösungen für raumbezogene (Bühler, [5]) und semantische Daten mittels einer normierten Auszeichnungssprache.

ii. *Aufbau einer standardisierten und strukturierten Wissensbasis:* Einsatz dynamischer Datenbanken zur Steuerung zeitlich-räumlicher Ereignisse und Prozesse (Peuquet, [64], S. 11-32). Entwicklung von Komponenten zur Wissensakquisition und Wissensrepräsentation.

iii. *Realisierung der Schnittstelle „Mensch⇔Maschine“:* Implementierung einer graphischen Benutzeroberfläche (GUI) als Kommunikationskomponente zwischen Benutzer und System. Entwicklung von Visualisierungstools zur idealen Darstellung der fachspezifischen Informationen durch widerspruchslose Integration von semantischen Daten mit Geometriedaten (Sester et al., [73], S. 51-62) unter Berücksichtigung der Abstraktion und der Skalierung.

iv. *Einsatz mobiler GIS-Dienste:* Bereitstellung und Verarbeitung von raumbezogenen Informationen (Wunderlich, [89]), Vernetzung der Anwender mit den Datenservern sowie Gewährleistung der gegenseitigen Kommunikation.

In den nachfolgenden Kapiteln werden die Konzepte und Modelle der Client-Server Architektur sowie die Realisierung und Implementierung des FIS zur Detektion dreidimensionaler Plattenbewegungen in Rumänien vorgestellt.

Kapitel 2

Objektrelationales Datenbanksystem

Die reale Welt bestmöglich abzubilden ist eine wesentliche Anforderung, die an ein raumbezogenes Informationssystem gestellt wird. Daher werden zwei Bedingungen bezüglich des Datenmodells und des funktionalen Modells definiert. Erstens soll das Datenmodell die Abstraktion der Wirklichkeit ermöglichen und zweitens soll das funktionale Modell die Methoden als Bestandteil der Objekte festlegen. Im allgemeinen versteht man unter einem Datenmodell

- die Darstellung von Informationen und deren Beziehungen in einem semantischen Kontext,
- das theoretische Konzept eines Datenbankmodells, welches als Grundlage eines Datenbanksystems dient.

Es werden Datenstrukturen aufgebaut und Operatoren zum Erfassen, Verändern, Abfragen und Archivieren der Daten entwickelt (Tab. 2.1). Mittels Regeln zur referentiellen Integrität werden die Möglichkeiten zur Manipulation der Daten festgelegt. Die universellen Aufgaben sind grundsätzlich in Gruppen von Anwendermethoden, Algorithmen und Funktionen beschrieben (Bartelme, [3], S. 23).

○ Erfassen	○ Präsentieren
○ Verändern	○ Analysieren
○ Strukturieren	○ Gestalten
○ Transformieren	○ Vergleichen
○ Konstruieren	○ Berechnen
○ Archivieren	○ Abfragen
○ Prüfen	○ Sichern

Tabelle 2.1: *Gruppen von GIS-Funktionen.*

Der Zugriff, das Erstellen, das Verändern und Kontrollieren sowie die Verwaltung der Daten erfolgt über ein Datenbankverwaltungssystem (DBMS), welches folgenden Anforderungen entspricht:

1. Konsistenz der Daten $\leftrightarrow$ Vermeidung von Integritätsverletzungen mittels Prozeduren,
2. Aufbau eines logischen Datenmodells,
3. Unabhängigkeit zwischen den Daten und deren Anwendungen,
4. Einrichtung von Sicherheitsmechanismen, die den Zugriff auf Objekte bzw. Tabellen steuern,
5. Wiederherstellung der Daten (Recovery-Strategie) mittels Transaktions-Protokolldateien,
6. Datensicherung im Online-Betrieb (Backup-Strategie),
7. dynamische Speicherverwaltung,

8. Mehrbenutzerbetrieb ↔ Synchronisation der Zugriffe.

Die Infrastruktur der abgebildeten Welt basiert auf einem objektrelationalen Datenmodell, welches die Objekte definiert und die Modellierung bestimmter Anwendungen festlegt. Zur Umsetzung des Datenmodells wird die *Datendefinitionssprache* (DDL) und zur Handhabung der Datenverwaltung die *Datenmanipulationssprache* (DML) eingesetzt.

2.1 Datenbankentwurf

Es wurde ein Datenbankentwurf zur Erfassung der Plattenkinematik in Rumänien aufgestellt, welcher eine *3-Schema-Architektur* verwendet. Diese Architektur setzt sich aus den Komponenten *konzeptuelles*, *logisches* und *physisches Schema* zusammen, die miteinander in Beziehung stehen (Vossen, [84]). In Abbildung 2.1 werden vom konzeptionellen Datenbankentwurf bis zur Ausführung des 3-Schema-Modelles die Komponenten der Architektur vorgestellt.

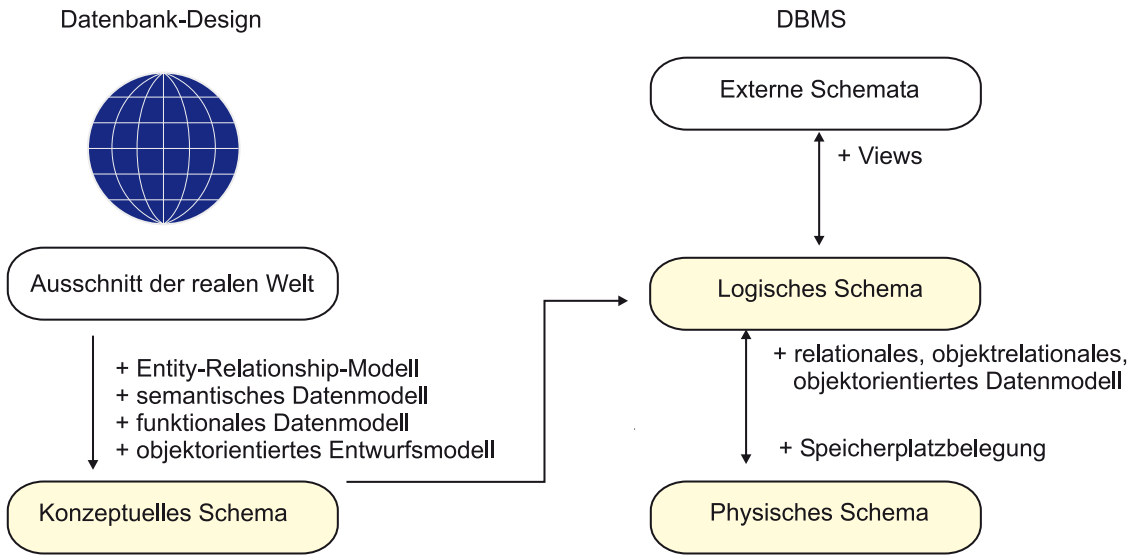


Abbildung 2.1: *Darstellung der 3-Schema-Architektur zur Modellierung der Daten. Im konzeptuellen Schema wird das Datenmodell mittels verschiedener Methoden entworfen (Datenmodellierer), im logischen Schema mittels eines bestimmten Datenmodells umgesetzt (Datenbankadministrator) und letztendlich im physischen Schema gespeichert (Datenbank). In den Datensichten (data views) werden die Informationen aus verschiedenen Tabellen dem Benutzer strukturiert bereitgestellt.*

Der Vorteil dieser Architektur liegt darin, dass eine gewisse Datenunabhängigkeit erreicht wird, die in die logische und physische Ebene aufgeteilt wird (Kemper/Eickler, [47], S. 20-21):

- *Physische Datenunabhängigkeit*: die Speicherstruktur wird verändert, dabei bleibt die logische Ebene unberührt. Ein nachträgliches Erstellen eines Indexes hat keinen Einfluss auf das logische Schemata.
- *Logische Datenunabhängigkeit*: die Eigenschaft eines Datenobjektes wird verändert, dabei bleibt die externe Ebene unberührt. Ein nachträgliches Ändern eines Attributes in einer Tabelle hat keinen Einfluss auf die Datensicht, die der Anwender aufruft.

Das Datenbankkonzept für das vorliegende Fachinformationssystem gliedert sich in die drei Kernbereiche: *Projektdaten*, *Geodaten* und *Analyse*. Diese Bereiche werden in mehrere Hauptmodule aufgeteilt (Abb. 2.2), die eine größere Anzahl von Klassen aufweisen, welche fachlich zu den Modulen zu zuordnen ist. Jede Klasse steht in Beziehung mit ihrer Vater- bzw. Kindobjektklasse, entweder über Relationen, welche über eindeutige

Klassen-Identifikatoren (Inst_ID, Station_ID, Koord_ID, Datum_ID und Ausw_ID) verfügen, oder über Referenzen, deren Werte eindeutig vom System generiert werden (Kap. 2.4.1).

Der vollständige Entwurf und die Beschreibung der Objekte sind Voraussetzungen zur Umsetzung des Datenbank-Konzeptes. Dabei wird der Klassenname eindeutig festgelegt, die Klasse zur Fachschale zugeordnet und die wesentlichen Eigenschaften beschrieben. Zusätzlich wird eine Klassen-Hierarchie aufgebaut, Relationen zu anderen Klassen bzw. Referenzen zu den Kindobjekten hervorgehoben und die Attribute spezifiziert (Anh. A).

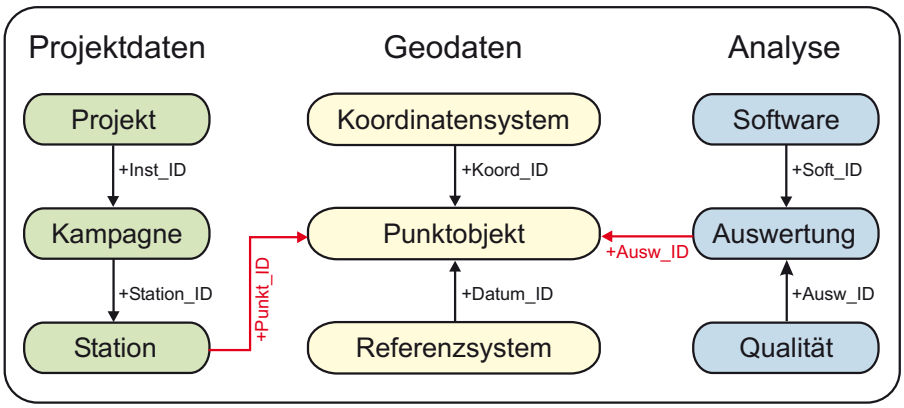


Abbildung 2.2: *Analyse der Projektinformationsanforderung - Übersicht der Hauptmodule eingeteilt in die Kernbereiche: Projektdaten, Geodaten und Analyse. Die Pfeile repräsentieren die logischen Beziehungen zwischen den Modulen mittels eindeutiger Schlüssel.*

Projektdaten umfassen die allgemeinen Informationen, die das Projekt betreffen. Unter anderem werden allgemeine Projektverwaltung, Mitwirkung der Personen, Beteiligung der Forschungseinrichtungen, Unterstützung durch Firmen, Ämter, Fachhochschulen und Universitäten sowie die Dokumentation des Projektverlaufes im Modul *„Projekt“* erfasst. Der Aufbau der Stationen, Definition der vermarkten Punkte und Zustandsberichte werden im Modul *„Station“* festgelegt. Im Modul *„Kampagne“* werden die eingesetzten GPS-Systeme, die Durchführung der Kampagnen und relevante Informationen über Messablauf (Besetzung, Beobachtungsdauer, Konfiguration) sowie über Untersuchungen bezüglich der GPS-Systeme (Mehrwegeeffekte, Antennenoffsets) beschrieben.

Geodaten werden durch die Module *„Referenzsystem“*, *„Koordinatensystem“* und *„Punktobjekt“* definiert. Mittels dieser Module werden die Ergebnisse der dreidimensionalen Positionsbestimmung einer Station, ihre Geschwindigkeit in Lage und Höhe, die Genauigkeit und Signifikanz, sowie das Koordinaten- und Bezugssystem erfasst.

Analysen werden durch die Module *„Software“*, *„Auswertung“* und *„Qualität“* repräsentiert, welche eine Qualitätssicherung der Daten und der Auswertungen sicherstellen soll. Dabei richtet sich der Schwerpunkt auf die Qualitätskontrolle und den -nachweis. Die Qualitätskontrolle zeigt dabei Unterschiede in den Ergebnissen der Deformationsanalyse bei Anwendung unterschiedlicher Auswertestrategien (z.B. Lösung der Phasenmehrdeutigkeiten in einer GPS-Auswertung) mit gleicher Softwareversion beziehungsweise mit unterschiedlichen Softwareversionen (z.B. Differenzen nach einem Upgrade der verwendeten GPS-Auswertesoftware) und gleicher Auswertestrategie auf. Der Qualitätsnachweis bezieht sich auf die Genauigkeit und Zuverlässigkeit der Deformationsanalysen. Insbesondere die zeitliche Komponente wurde während der Konzeption des Datenbankentwurfes berücksichtigt, da zukünftige GPS-Kampagnen zu verbesserten Ergebnissen in der Genauigkeit und Zuverlässigkeit der berechneten Koordinaten und Geschwindigkeiten führen werden. Die Attribute im Modul *Software* beschreiben die Version, die Urheber, die Modelle zur Deformationsanalyse und das entwickelte Programm *„DEFO3D“*. Die Klassenobjekte im Modul *Auswertung* legen die Methode und Stratgegie sowie eine ausführliche Beschreibung des Auswerteprozesses fest. Als Bezugsgrößen im Modul *Qualität* werden die Merkmale Vollständigkeit, Konsistenz und Aktualität sowie die Eigenschaften Genauigkeit und Zuverlässigkeit herangezogen (Born/Figura, [10], S. 25-27).

2.2 Datenbankarchitektur

Die Architektur einer Datenbank (Abb. 2.3) gliedert sich in die Benutzer-, Datenbankverwaltungs- und Datenbank/Instanz-Ebene (Pernul/Unland, [62], S. 183-185).

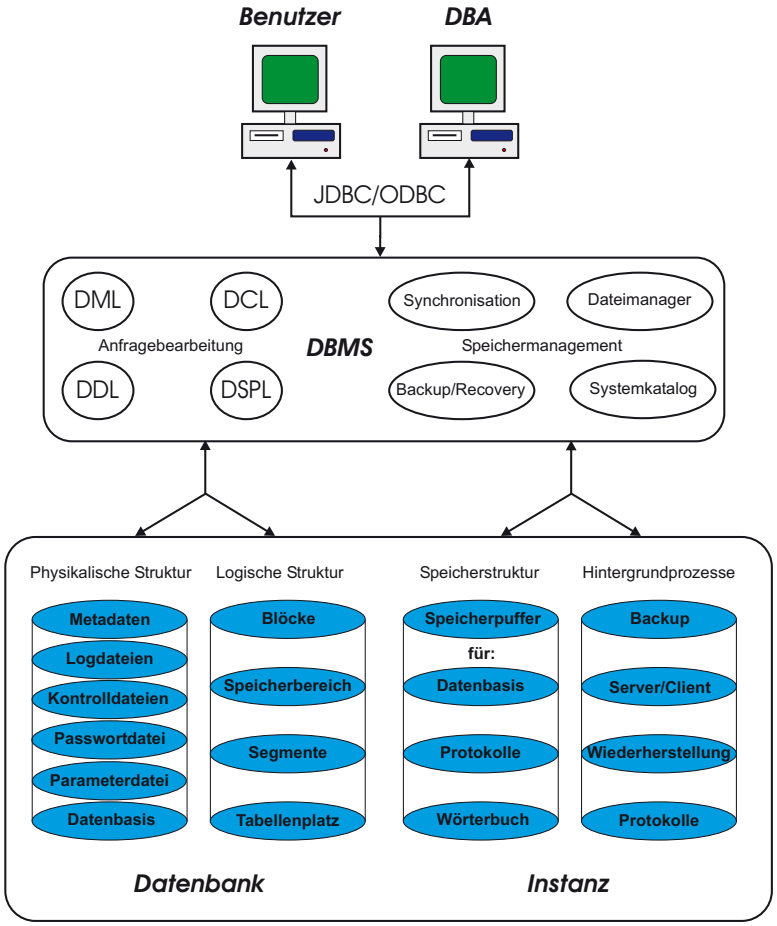


Abbildung 2.3: *Darstellung der Oracle-Datenbankarchitektur, die in folgende drei Hauptbereiche: Benutzer/DBA, DBMS und Datenbank/Instanz gegliedert wird (Adkoli/Velpuri, [1]).*

Benutzer-Ebene: Mittels definierter *Datenbankschnittstellen* werden die Anfragen des Benutzers vom DBMS bearbeitet. Dabei wird die Anfrage von der Syntax der Programmiersprache (z.B. Perl, PHP, Visual Basic oder Java) in die Syntax des Datenbanktreibers (z.B. JDBC, ODBC oder IDAPI) übersetzt.

Datenbankverwaltungssystem-Ebene: Das Datenbank-Managementsystem ist ein System, welches die Datenbank verwaltet. Es ist verantwortlich für die Speicherung, die Sicherheit, die Integrität, die Wiederherstellung und den Zugriff der Daten. Das DBMS steuert somit die Organisation und das Auffinden der Daten. Die Prozesse zur Synchronisation und Verwaltung der Systemkataloge, der Bibliotheken und der Wörterbücher werden ebenfalls im DBMS gesteuert. Es besteht die Möglichkeit, Anforderungen bzw. Anfragen an die Datenbank mittels diverser Datenbank-Sprachen geeignet zu übersetzen. Neben der Datenmanipulations- und der Datendefinitionssprache werden im DBMS die *Datenbankkontroll-* (DCL) und die *Prozedursprache* (DSPL) eingesetzt. Die DCL definiert, startet oder nimmt im DBMS *Transaktionen* (Kap. 4.4) zurück. Insbesondere werden die Konfiguration und die Steuerung der Datenbank sowie die Zugangsberechtigungen der Benutzer kontrolliert. Die DSPL definiert Trigger bzw. anwendungsspezifische Prozeduren, welche in der Datenbank abgelegt sind.

Datenbank/Instanz-Ebene: Oracle unterteilt die DB-Strukturebene in die Einheiten Datenbank und Instanz (Adkoli/Velpuri, [1], S. 121-138). Die Datenbank wird in die Komponenten *physikalische* und *logische Struktur* aufgeteilt. Dabei werden die Metadaten, die Datenbasis und verschiedene Protokolldateien in der physikali-

schen Struktur abgelegt. Die logische Datenbankstruktur besteht aus den Komponenten *Tablespaces*, *Segmente*, *Extents* und *Blöcke*. Tablespaces werden für das System, die Benutzer, die Indexe und für temporäre Objekte angelegt. Nach Anlegen des Tablespaces können z.B. Tabellen für Metadaten und die Datenbasis erstellt werden. Das System-Tablespace steuert das *Datenwörterbuch* (data dictionary), welches aus den *Katalogen* besteht, die sämtliche Systeminformationen verwalten. Die Segmente enthalten eine Menge von Extents, welche Indexe, temporäre Daten und Tabellen speichern. Ein Extent wird aus mehren benachbarten Blöcken gebildet, welche die kleinste Einheit bilden (Standardeinstellung: 8 Kilobyte). Die Instanz besteht aus Speicherstrukturen, die einerseits beim Starten der Datenbank Kontroll- und Systeminformationen für verschiedene Lese- und Schreiboperationen bereitstellt und andererseits Speicher z.B. für Datenbankblöcke, Bibliotheken oder Protokolle freigibt. Parallel laufen Dienste im Hintergrund, welche die Server-Client-Kommunikation, Backup- und Recovery-Strategie, Speicherkontrollen und verschiedene Transaktionen im DBMS steuern (Kap. 3.2.1).

2.3 SQL-Standard

Die Sprache „Structured Query Language“ (SQL) wurde von der internationalen Vereinigung der Standardisierungsgremien (ISO) als Standard zur Bearbeitung von Anfragen für relationale Datenbank-Managementsysteme (RDBMS) spezifiziert. Diese Sprache wurde zusammen mit dem universellen DBMS „DB2“ in den 80-er Jahren von IBM eingeführt. Die nachfolgende Tabelle stellt die zeitliche Entwicklung von SQL-Standards im Kontext des RDBMS/ORDBMS (Michels et al., [56], S. 30-38) vor.

Standard	***Entwicklung bis 1987***
SQL/87	Als Standard festgelegt (ISO, [35]). Entwicklung des 2-Level-Konzepts (Einführung + Voll). Unterstützt DDL und DML. Allgemeine Funktionen: → *connect, disconnect* [database] → *create, drop, alter* [table] → *create, drop* [index] → *insert, update, delete* [row] → *autocommit, rollback* [command] → *grant, revoke* [table, view]

Standard	***Entwicklung bis 1989***
SQL/89	Als Standard festgelegt (ISO, [36]). Zusätzlich zum 2-Level-Konzept: Erweiterung der *Integritätssicherungskonzepte* (IEF) ⇒ Spezifizierung des Primär- und des Fremdschlüssels. Standardwert für Spalte definieren und Überprüfung der Randbedingungen. Allgemeine Funktionen: → *not null, check, unique* [Spalte, Zeile] Neue/veränderte Datentypen: → *character* (feste Zeichenkette mit [n]-Länge) → *real, double precision* (4-, 8-byte Fließkomma)

Standard	***Entwicklung bis 1992, korrigiert bis 1996***
SQL/92	Als Standard festgelegt (ISO, [37]). Entwicklung des 3-Level-Konzepts (Einführung, Fortgeschritten, Final) + Erweiterungen. Eingebettetes, dynamisches SQL. Erstellen von Systemkatalogen (Ansammlung von Schemata): → *create, drop, grant* [schema] Festlegung des Schemas (Ansammlung von Deskriptoren für Datenobjekte): → *Domänen, Basistabellen, Sichten, Bedingungen, Rechte, Zeichenmengen, Zeichenordnungen, Zeichenübersetzungen* Aufbau eines Managements für die Verbindung zwischen Client und DBMS. Aufbau einer Session-Verwaltung. Zulassung temporärer Tabellen.

	Sprachmittel für Datenbankprozeduren (Stored Procedures). Ereignisgesteuerte Datenmanipulation (Trigger). Unterstützung verteilter Datenbanken. Erweiterung der referenziellen Integritätsregeln (domain, general and base table constraints): → *cascade, delete, set null, set constraints* [Tabelle] Neue/veränderte Datentypen: → *bit* → *character* (variable Zeichenkette mit [n]-Länge) → *integer* (4-byte Zahl) → *date* (Datum + Zeit) → *time, interval, timestamp* (Jahr, Monat, Tag, Stunde, Minute, Sekunde, Millisekunde)

Standard	***Entwicklung bis 1999, korrigiert bis 2003***
SQL/99	Als Standard festgelegt (ISO, [38]). Übergang von RDBMS nach ORDBMS. Persistent gespeicherte Module (PSM). Einfach-Vererbung. Abstrakte Datentypen, Methoden, Funktionen und Prozeduren. Sprachverknüpfungen: JDBC, SQLJ, C++ Call Interface (OCCI). Optimierung der Joins (Verknüpfung von Datenobjekten). Neue/veränderte Datentypen: → *distinct, structured types* (Benutzer definierte Typen und Funktionen) → *collection types* (variable Elemente oder Arrays) → *blob, clob* (binary large object, character large object) → *REF* (Selbstreferenzierung) → *date* (erweiterte Funktionen) → *numeric(i,j), boolean* (exakte Zahl mit genauer Formatsangabe, Rückgabe: wahr oder falsch) → *union, intersect, except* (Erweiterte Operationen zur Manipulation von Datenmengen)

Standard	***Entwicklung seit 2003***
SQL/03	Der Standard SQL/03 ist noch in der Entwicklung und ist in folgende Kommissionen unterteilt: 1. Framework 2. Foundation 3. Call Level Interface (SQL/CLI) 4. Persistent Stored Modules (SQL/PSM) 9. Management of External Data (SQL/MED) 10. Object Language Bindings (SQL/OLB) 11. Information and Definition Schemas (SQL/Schemata) 13. SQL Routines and Types Using the Java TM Programming Language (SQL/JRT) 14. XML-Related Specification (SQL/XML)

Tabelle 2.2: *Überblick über die SQL-Standards, die seit 1986 in regelmäßigen Zeitabschnitten von der ISO herausgegeben werden. Es wird die Entwicklung seit Einführung der RDBMS in kurzen Stichworten dargestellt, insbesondere der Übergang vom RDBMS zum ORDBMS.*

2.4 Objektrelationale Datenbank

Seit Festlegung der SQL/92-Normierung wird ein Schwerpunkt auf die Zusammenführung der relationalen und objektorientierten Modellierung gelegt. Vor- und Nachteil relationaler Datenbanken lassen sich wie folgt gegenüberstellen:

+ Der Vorteil liegt darin, dass sie weltweit verbreitet sind und die Daten dank der Standardisierung komfortabel in andere Datenbanksysteme migriert werden können.

- Der Nachteil liegt darin, dass die reale Welt nicht in Objekten abstrahiert wird, die untereinander in Beziehung stehen. Dieser Nachteil wird durch ein objektorientiertes Datenbankmodell eliminiert.

Daher wird seit der SQL/99-Normierung der Fokus auf eine Erweiterung der relationalen Eigenschaften und Objektfähigkeiten gelegt, indem abstrakte Datentypen entwickelt wurden, die reale Objekte besser abbilden können. Die bisherigen Beziehungen (Primär- / Fremdschlüssel) werden durch Referenzen auf die Datenobjekte abgelöst. Weitere Datentypen, wie z.B. große Objekte oder variable Felder, sind hinzugekommen, die eine flexiblere Definition von Objekten zulassen. Die Einfach-Vererbung wird teilweise zugelassen. Zusätzlich können benutzerdefinierte Routinen in das ORDBMS implementiert werden. Auf die Eigenschaften und Merkmale, die in SQL/92 entwickelt wurden, wird in dieser Arbeit nicht eingegangen. Es wird auf die einschlägige Literatur verwiesen (z.B. Date/Darwen, [17]).

2.4.1 Referenzierung - Objektidentifikator

Ein wesentliches Merkmal einer objektrelationalen Datenbank besteht darin, zwei Zeilenobjekte miteinander zu referenzieren. Dabei wird das Zeilenobjekt definiert *als Objekttyp, der eine vollständige Zeile enthält* (Tab. 2.3).

```
create or replace type SOFTWARE_TY as object
(
  Softwarename       varchar2(80),
  Soft_ID            varchar2(40),
  Soft_Version       varchar2(80),
  Soft_Quelle        varchar2(2000),
  Soft_Datum         date,
  Soft_Beschreibung  clob
);
```

Tabelle 2.3: *Anlegen eines Zeilenobjektes (SOFTWARE_TY) mittels der Anweisung: create or replace type [Name_TY] as object.*

Mittels eines eindeutigen Objektidentifikators (OID) wird auf das Zeilenobjekt ein Verweis gelegt. Die referenzielle Integrität der Zeilenobjekte, welche über die *REF-Funktion* miteinander referenziert werden können, wird durch das OID-Verwaltungssystems sichergestellt. Der Vorteil der Referenzierung gegenüber der relationalen Verknüpfung liegt darin, dass Anfragen nicht über herkömmliche Verbundoperationen, sondern direkt über die Referenz beantwortet werden. Dieses System der Anfragebearbeitung führt dazu, dass die Anfrage erheblich effizienter und schneller beantwortet wird.

Im FIS-Datenmodell werden REF-Anweisungen erzeugt, um Beziehungen zwischen über- und untergeordneten Klassen aufzubauen. Das Modul *„Auswertung“* setzt sich aus dem Aggregat *REFAUSWERTUNG_TAB* und den beiden Teilobjekten *AUSWDEF_TAB* und *SOFTWARE_TAB* zusammen (Abb. 2.4). Diese Form der Zusammensetzung wird im Klassendiagramm als *Aggregation* bezeichnet.

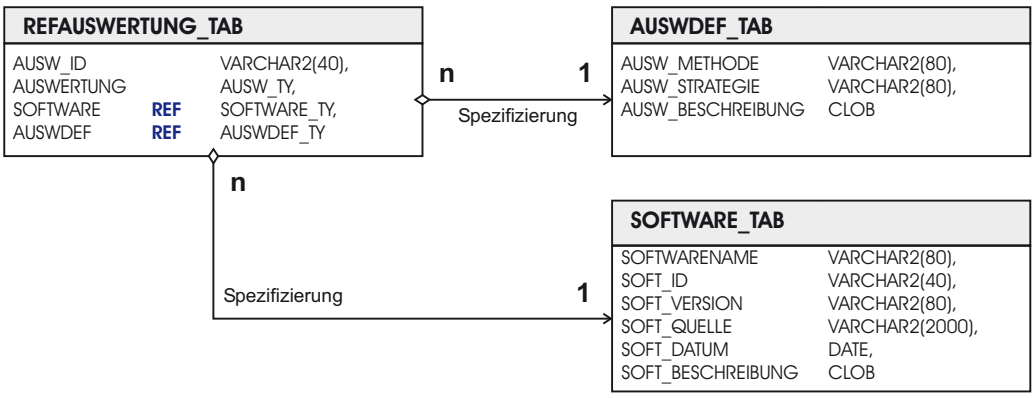


Abbildung 2.4: *Referenzierung von einfach strukturierten Zeilenobjekten, festgelegt durch* $(n:1)-$*Beziehung.*

Die Inhalte der Datentypen werden in Anhang A.2 detailliert dargestellt. Die Referenzierung erfolgt über die Zeilenobjekte *SOFTWARE_TY* und *AUSWDEF_TY*. Beim Erstellen der Objekttabelle wird gleichzeitig ein OID-Verwaltungssystem angelegt, welches jedem Zeilenobjekt einen eindeutigen Wert zuweist. Den strukturierten Objekttypen werden Referenzen mit $(1:1)-$ oder mit $(n:1)-$Beziehungen zugewiesen. Komplexere Strukturen mit $(n:m)-$Beziehungen sind prinzipiell möglich, aber im praktischen Einsatz noch nicht relevant (Stonebraker/Moore, [77]). Die Objekttabellen werden folgendermaßen implementiert:

- Zuerst werden die benutzerdefinierten Datentypen erzeugt: create or replace type [Name_TY] as object.
- Danach werden die Objekttabellen erstellt: create table [Name_TAB] as [Name_TY]. Die Objekttabellen *SOFTWARE_TAB* und *AUSWDEF_TAB* bestehen ausschließlich aus den definierten Datentypen *SOFTWARE_TY* bzw. *AUSWDEF_TY*.
- Die übergeordnete Tabelle *REFAUSWERTUNG_TAB* enthält die Attribute *SOFTWARE* und *AUSWDEF*, deren REF-Operatoren auf die jeweiligen Datentypen zeigen.

Durch Einfügen von Zeilenobjekten wird jeder Zeile ein OID-Wert zugewiesen, der in einigen Programmiersprachen einem Zeiger entspricht. Die Funktion *REF* übernimmt ein Zeilenobjekt als Argument und gibt den Rückgabewert für die Methode des Datentyps *REF* zurück. Um die Informationen der Zeilenobjekte zu erhalten, wird die DEREF-Funktion benötigt, die den Referenzwert einliest und das referenzierte Zeilenobjekt ausgibt (Tab.: 2.4). Da auf abstrakte Datentypen nicht direkt zugegriffen werden kann, wird im select-Befehl ein Tabellenalias („*ausw*“) den Spalten (*Spalte.Attribut*) direkt vorangestellt und der übergeordneten Tabelle *REFAUSWERTUNG_TAB* nachgestellt. Der Tabellenalias ist frei wählbar und besitzt seine Gültigkeit bis zur Beendigung der select-Anweisung.

```
SQL> select * from REFAUSWERTUNG_TAB;

AUSW_ID|AUSWERTUNG(AUSW_NAME, AUSW_TYP,AUSW_DATUM, AUSWERTER)|
SOFTWARE|AUSWDEF
-------------------------------------------------------------
Final 2002, AUSW_TY('3D-Defo-2002', 'Dreidimensionale Deforma-
tionsaanalyse', '15.03.03', 'Georg Dinter'),
0000220208DF740E7591804737A15A8E69592F96AD986998A947AGHIUM...,
0000220208KF789E7492355532B35C9L83692G32BT587453B128TZUROL...

SQL> select ausw.Ausw_ID, ausw.Auswertung, DEREF(ausw.Software)
  2> from REFAUSWERTUNG_TAB ausw;

AUSW_ID|AUSWERTUNG(AUSW_NAME, AUSW_TYP, AUSW_DATUM, AUSWERTER)|
DEREF(ausw.SOFTWARE)(SOFTWARENAME, SOFT_ID, SOFT_VERSION,
SOFT_QUELLE, SOFT_DATUM, SOFT_BESCHREIBUNG)
-------------------------------------------------------------
Final 2002, AUSW_TY('3D-Defo-2002', 'Dreidimensionale Deforma-
tionsanalyse', '15.03.03', 'Georg Dinter'), SOFTWARE_TY('3D-
DEFO1.1', '3D-Defo-1.1', 'Version 1.1', '...', '01.02.03',
'Die Software 3D-Defo ist entwickelt worden, um im Projekt
Dreidimensionale Plattenkinematik in Rumänien relative
Bewegungsabläufe der vier tektonischen Platten aufzudecken.
...')
```

Tabelle 2.4: *(1.) Ausführen der REF-Anweisung mittels des SQL-Befehls: select [Attribut] from [Tabellenname]. Eine 74-stellige OID wird als Rückgabewert zurückgegeben. (2.) Ausführen der DEREF-Anweisung mittels des SQL-Befehls: select [alias.attribut] from [Tabellenname alias]. Der Rückgabewert ist das Zeilenobjekt.*

2.4.2 Abstrakte Datentypen

Abstrakte Datentypen werden als *benutzerdefinierte Datentypen (UDT)* definiert, *die standardisierte Datentypen in einen Datentyp vereinigen* (Tab. 2.5).

```
create or replace type AUSW_TY as object
(
  Ausw_Name     varchar2(40),
  Ausw_Typ      varchar2(80),
  Ausw_Datum    date,
  Auswerter     varchar2(80)
);
/
create table REFAUSWERTUNG_TAB
(
  Auswertung    AUSW_TY,
  ...
)
```

Tabelle 2.5: *Erstellen eines abstrakten Datentyps mittels der Anweisung: create or replace type [Name_TY] as object. Der abstrakte Datentyp wird in der Tabelle REFAUSWERTUNG_TAB dem Attribut „Auswertung“ zugeordnet.*

Es lassen sich abstrakte Datentypen nach folgenden Typkategorien oder inhaltlichen Aspekten zusammenstellen:

1. Typkategorien

- skalare Datentypen (z.B. *Number*),
- zusammengesetzte Datentypen (z.B. *VARRAY*),
- Referenztypen (REF),
- Methoden (Member Function),
- abstrakte Datentypen (UDT),
- LOB-Typen (z.B. *BLOB*).

2. Inhaltliche Aspekte

- Identifikatoren,
- Texte und Matrizen,
- räumliche und zeitliche Daten,
- Videos und Bilder.

Diese Datentypen werden im FIS-Datenmodell als benutzerdefinierte Datentypen eingesetzt, um die inhaltlichen Aspekte der Objekte hervorzuheben.

2.4.3 Große Objekte

Datentypen werden als *große Objekte (LOB)* definiert (Tab. 2.6), *die binäre (BLOB) oder alphanumerische (CLOB) Dateninformationen bis zu einer Größe von 4 GB enthalten* (Loney/Koch, [53]).

Im Kontext der objektorientierten Modellvorstellung wird im FIS-Datenmodell vom atomaren Aufbau der Tabellen relationaler DBMS abgesehen. Bilder und Kartenwerke werden als BLOBs und allgemeine Objektbeschreibungen als CLOBs in der Datenbank einem reservierten Speicherbereich (*tablespace [LOB_Name]*) zugeordnet.

```
create or replace type IMAGE_TY as object
(
  Img_Name      varchar2(40),
  Image         blob
);
/
create table IMAGE_TAB of IMAGE_TY lob(Image) store as
( tablespace [LOB_Name] [optionale Parameter] );
```

Tabelle 2.6: *Erstellen großer Datenobjekte (BLOB oder CLOB) mittels der Anweisung: create or replace type [Name_TY] as object. Große Objekte werden in Tabellen angelegt, die einem gesonderten „Tablespace“ als Speicherbereich zugewiesen werden.*

Das Untermodul *„Image“* setzt sich aus dem Aggregat *IMGDEF_TAB* und den existenzabhängigen Teilobjekten *IMGMETADATA_TAB*, *IMAGE_TAB* und *KARTEDEF_TAB* zusammen (Abb. 2.5). Diese Form kennzeichnet die *Komposition*, welche im Klassendiagramm eine strengere Form der Aggregation darstellt. Die Abhängigkeit zwischen den Objektklassen wird durch $(1:1)-$Beziehung festgelegt. Die Objektklasse *IMGBFILE_TAB* wird temporär eingesetzt, um binäre Dateien mittels des Datentyps *BFILE* in die Objektklasse IMAGE_TAB einzufügen. Die Assoziation beschreibt temporär die gemeinsame Semantik und Struktur der $(1:1)-$Beziehung zwischen den beiden Klassen. Nach erfolgreichem Einfügen wird der Datenbanklink auf dem Verzeichnis der binären Datei gelöscht (Kap. 2.4.4).

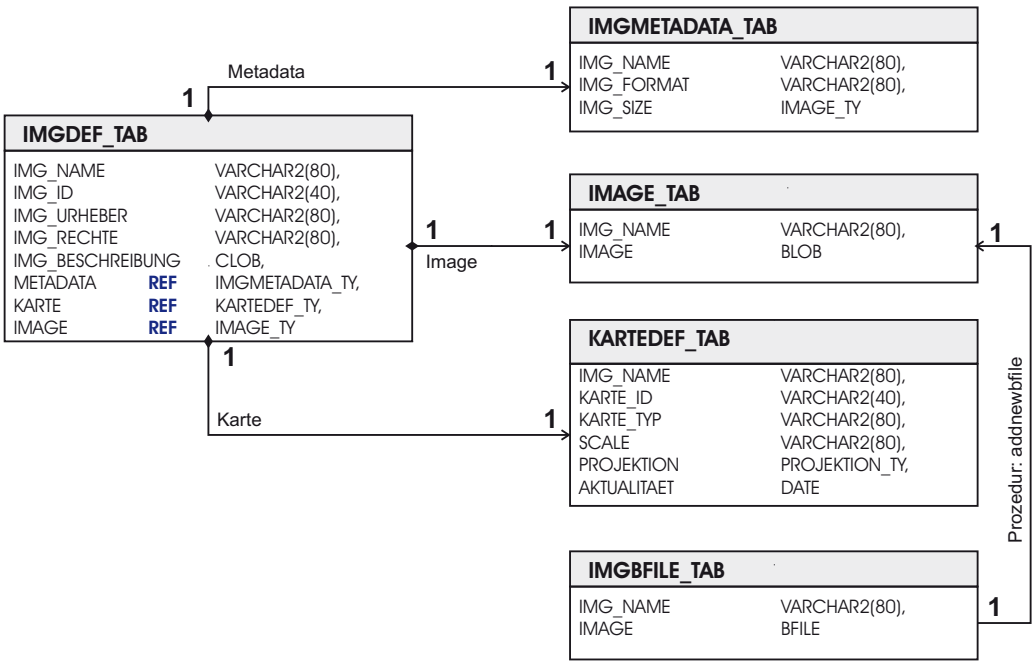


Abbildung 2.5: *Aufbau des Untermoduls „Image“ - Verwendung der Datentypen BLOB und CLOB.*

2.4.4 Prozeduren

Prozeduren werden definiert als *wiederverwendbare Programmcodes, die es erlauben konkrete Aufgaben durchzuführen* (Tab. 2.7). Prozeduren werden im Datenbanksystem gespeichert und über eine spezielle Umgebung (z.B. PL/SQL-Editor) aufgerufen. Der Kopfteil beginnt mit dem „PROZEDURNAME“. Der Rumpf wird mit „BEGIN“ eingeleitet und mit „EXCEPTION“ abgeschlossen. Falls Parameter definiert werden, so besteht die Parameterliste aus den Variablen, den Parameterarten (IN, OUT und IN OUT) und den Datentypen (VARCHAR2, DATE, INTEGER, usw.).

```
PROCEDURE name [ ( parameter [, parameter ] ) ] IS
    Deklarationsteil
BEGIN
    Programmteil
EXCEPTION
    Ausnahmebehandlung
END;
```

Tabelle 2.7: *Der prinzipielle Aufbau einer Prozedur entspricht dem eines PL/SQL - Blocks: Deklarationsteil, Programmteil und Ausnahmebehandlung.*

Die Prozeduren wurden im FIS entwickelt, um binäre und alphanumerische Daten in die jeweiligen Zeilenobjekte automatisch einzufügen. Die Prozedur *„ADDNEWBFILE“* speichert Bilder und Karten in die Tabelle *IMAGE_TAB* ab. Dabei wird zusätzlich der Datentyp *BFILE* eingeführt, um die binären Dateien mittels externer Datenbanklinks aufzurufen, da diese Dateien physikalisch außerhalb der Datenbank gespeichert und nur über den Verzeichnispfad verbunden sind. Zusätzlich wird in der temporären Tabelle *IMGBFILE_TAB* ein BFILE-Datentyp angelegt, damit die Lokalisierung der gespeicherten Bilddateien mittels dem *Locator* sichergestellt wird. Nachdem die Bilddatei auf „schon vorhanden / nicht vorhanden“ unter Verwendung eines *Cursors* überprüft worden ist, wird die Bilddatei mit Hilfe des *DBMS_LOB*-Packages (Urman, [83]) im vorgesehenen Tablespace abgelegt. Der vollständige Code wird in Anhang A.3 angegeben.

```
create or replace procedure ADDNEWBFILE
(
  p_Code      IN    Varchar2,
  p_Filename  IN    Varchar2
)
```

Mittels der Anweisung *create or replace procedure [Name]* wird die Prozedur angelegt. Dabei werden die Variablen *p_Code* und *p_Filename* im Kopfteil festgelegt. Der Datentyp Varchar2 der beiden Variablen muss dem Datentyp des Attributes *IMG_NAME* in der Objektklasse *IMAGE_TAB* und der temporären Klasse *IMGBFILE_TAB* entsprechen.

```
AS
  File_Loc       bfile;
  v_date         date;
  dest_lob       blob;
  var1           integer;
  img_check      varchar2(80);
```

Der Locator *File_Loc* und die Variablen (v_date $\Leftrightarrow$ Systemdatum, dest_lob $\Leftrightarrow$ Bilddatei, var1 $\Leftrightarrow$ Bilddatei existiert [0,1], img_check $\Leftrightarrow$ Bilddatei vorhanden [img_name oder null]) werden zusätzlich im Kopfteil der Prozedur definiert.

```
BFILE_NOTFOUND exception;
IMAGE_DOUBLEFOUND exception;
```

Es werden die Ausnahmen *„Datenbanklink wird nicht gefunden“* $\longrightarrow$ *BFILE_NOTFOUND* und *„Bilddatei mit dem Namen ist schon vorhanden“*$\longrightarrow$ *IMAGE_DOUBLEFOUND* definiert.

```
Cursor img_cursor IS select img_name from IMAGE_TAB
  where img_name = p_CODE;
```

Der Cursor ist ein Zeiger auf einen Kontextbereich. Der Kontextbereich ist ein reservierter Speicherbereich, der zur Ausführung einer Anweisung benötigt wird. In diesem Falle wird ein Speicherbereich festgelegt, um den Dateinamen eines Bildes aus der Tabelle zu selektieren.

```
Begin
  File_Loc      := BFilename('IMAGES',p_Filename);
  var1          := DBMS_LOB.FILEEXISTS(File_Loc);
  v_date        := sysdate;

  if var1 = 1 then
    img_check := null;
    OPEN img_cursor;
      Fetch img_cursor into img_check;
      if img_check is not null then
        Raise IMAGE_DOUBLEFOUND;
      end if;
    Close img_cursor;
```

Der Rumpf einer Prozedur wird durch den Befehlsaufruf [BEGIN] eingeleitet. Der Locator *File_Loc* zeigt durch Aufruf der Funktion *Bfilename(Pfad, Dateiname)* auf die Bilddatei, die in die Tabelle IMAGE_TAB eingefügt werden soll. Mittels der DBMS_LOB-Funktion *FILEEXISTS(loc)* wird überprüft, ob die Datei tatsächlich vorhanden ist $(var1 = 1)$ oder ob der Zeiger auf einen undefinierten Speicherbereich zeigt $(var1 = 0)$. Es wird die Systemzeit bestimmt und in der Variablen *v_date* festgelegt. Sofern die Bilddatei existiert, wird der Cursor geöffnet, um zu überprüfen ob die Bilddatei als BLOB schon abgelegt worden ist. Falls ja (Dateiname wird in die Variable *img_check* kopiert), wird die Exception „IMAGE_DOUBLEFOUND“ ausgelöst und somit die Datenintegrität sichergestellt.

```
insert into IMAGE_TAB(img_name,image) values (
  p_code,
  empty_BLOB());
```

Unter Verwendung der Funktion *empty_BLOB()* wird ein Datenobjekt vordefiniert, um eine binäre Datei zu einem späteren Zeitpunkt einzufügen.

```
insert into IMGBFILE_TAB(img_ref,img_file) values (
  p_code,
  BFilename('IMAGES',p_Filename));

select image into dest_lob from IMAGE_TAB where img_name = p_code
  for update;
```

Damit die DBMS_LOB-Packages verwendet werden können, wird eine Tabelle definiert, die eine Bild-ID und den Bfile-Datentyp für die Bilddatei festlegt. Temporär wird die Bildreferenz und das Datenobjekt eingefügt, solange bis die binäre Datei in der Datenbank als Blob-Datentyp gespeichert wurde. Die Tabelle IMAGE_TAB wird zur Aktualisierung durch die Anweisung: *select [Attribut] into [Variable] (where-Bedingung) for update* vorbereitet.

```
DBMS_LOB.FILEOPEN(File_Loc,DBMS_LOB.File_Readonly);
DBMS_LOB.LOADFROMFILE(dest_lob,File_Loc,DBMS_LOB.GETLENGTH(File_Loc));
```

Nachdem der Locator mittels der Methode FILEOPEN(loc, function) geöffnet wurde, kann die Datei eingelesen und die Dateilänge durch Anwendung der Funktion GETLENGTH(loc) bestimmt werden. Die Variable *dest_lob* enthält den vollständigen Datensatz.

```
update IMAGE_TAB set image = dest_lob where img_name = p_code;
. . .
delete IMGBFILE_TAB;
commit;
DBMS_LOB.FILECLOSE(File_Loc);
. . .
end addnewbfile;
```

Die Daten werden mittels der Anweisung *update [Tabelle] set [Spalte] = [Variable]* in das leere Datenobjekt Image kopiert. Danach wird das BFILE-Datenobjekt gelöscht. Durch *Commit* wird die Transaktion dauerhaft in der Datenbank gesichert. Anschließend wird der Locator geschlossen. Damit wurde die Prozedur erfolgreich beendet.

2.4.5 Collector

Ein Collector definiert sich dadurch, daß er *eine Menge geschachtelter Datenobjekte in einer Zeile einer Objekttabelle sammelt*. Im Collector sind daher Elemente bzw. Datensätze eingefügt, die Bestandteil einer einzigen Zeile sind. Der Collector enthält zwei Datentypen:

1. Verschachtelte Tabellen (Embedded Objects).
2. Variable Arrays (VARRAY).

2.4.5.1 Verschachtelte Tabellen

Tabellen, *die in einer Tabelle eingebettet sind*, werden als *verschachtelte Tabellen* definiert. Eine Menge von Datenobjekten wird in einer Spalte eingefügt und einem Zeilenobjekt zugeordnet (Tab. 2.8). Damit ist gleichzeitig die $(1:n)-$Beziehung zwischen den verschachtelten Tabellen hergestellt. Durch Kombination der verschachtelten Tabellen mit referenzierten Tabellen (Kap. 2.4.1) können $(n:m)-$Beziehungen aufgebaut werden.

```
create or replace type PHYS_TY as object (
ELL_ID            VARCHAR2(40),
Koeffizienten     KOEFF_NT,
Phys_Parameter    PHYS1_TY,
Phys_Konstante    PHYS2_TY);
/
create table PHYS_TAB of PHYS_TY
nested table Koeffizienten store as KOEFF_NT_TAB;
```

Tabelle 2.8: *Anlegen der verschachtelten Tabelle „KOEFF_NT_TAB“ in der Haupttabelle „PHYS_TAB“ mittels der Anweisung: create table [Tabellenname] of [Datentyp] nested table [Attribut] store as [Tabellenname der verschachtelten Tabelle]* .

Zuerst wird der Datentyp *PHYS_TY* als Zeilenobjekt definiert: Ellipsoid-Identität (*ELL_ID* als Referenzattribut), verschachtelte Tabelle (Attribut *Koeffizienten* als Datentyp „KOEFF_NT“) und benutzerdefinierte Datentypen (Attribute *Phys_Parameter* als Datentyp „PHYS1_TY“ und *Phys_Konstante* als Datentyp „PHYS2_TY“). Die Objektklasse *PHYS_TAB* wird als Tabelle des Datentyps *PHYS_TY* unter Berücksichtigung der verschachtelten Tabelle *KOEFF_NT_TAB* erzeugt, worin die Datenobjekte des Datentyps *KOEFF_NT* eingefügt werden. Die verschachtelte Tabelle ist Bestandteil der Tabelle *PHYS_TAB* und kann daher nicht direkt mittels einer $\{select...\}$-Anweisung aufgerufen werden. Ansonsten wird folgender Fehler angezeigt:

„Speichertabelle der Spalten einer verschachtelten Tabelle nicht referenzierbar!“

Im FIS-Datenmodell werden verschachtelte Tabellen eingesetzt, um $(1:n)-$Beziehungen herzustellen (Abb. 2.6). Das Untermodul *„Geodätisches Datum“* setzt sich aus dem Aggregat *REFDATUM_TAB* und den Teilobjekten *REFELLIPSOID_TAB*, *DATUM_TAB* und *PHYS_TAB* zusammen. In der Objektklasse *PHYS_TAB* wird zwischen der Ellipsoid-Identität (ELL_ID) und den zonalen Kugelfunktionskoeffizienten (KOEFFIZIENTEN) eine $(1:n)-$Beziehung aufgebaut.

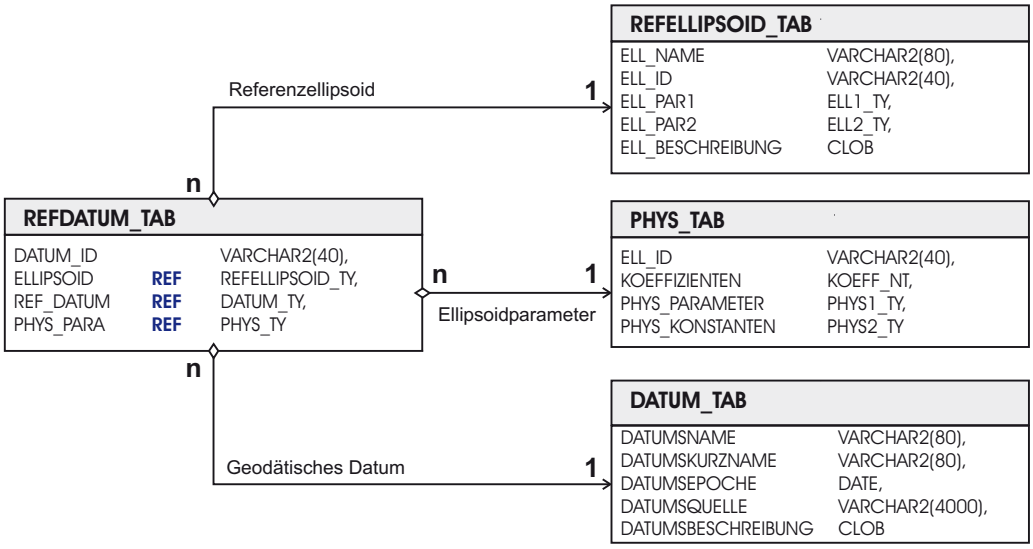


Abbildung 2.6: *Aufbau des Untermoduls „Geodätisches Datum“ - Verwendung von verschachtelten Tabellen - Realisierung einer* $(1:n)-$*Beziehung*

Die Ausgabe der Dateninformationen erfolgt mit einem Editor, der geschachtelte Datenobjekte ausliest (Tab. 2.9).

```
SQL> select Ell_ID, Koeffizienten, Phys_Parameter, Phys_Konstanten
  2> from PHYS_TAB where ELL_ID = 'GRS80';

Ausgabe> GRS80,
      2> KOEFF_TY('J2', 1.08263E-03), KOEFF_TY('J4', -2.37091222E-06),
      3> KOEFF_TY('J6', 6.08347E-09), KOEFF_TY('J8', -1.427E-11),
      4> KOEFF_TY('J10', 0.1E-13),
      5> PHYS1_TY(7.292115E-05, '[rad/s]', ...),
      6> PHYS2_TY(9.7803267715, 9.8321863685, '[m/s**2]', ...)
```

Tabelle 2.9: *Ausgabe der Datenobjekte der verschachtelten Tabelle PHYS_TAB (Referenzellipsoid: GRS80 | Kugelfunktionskoeffizienten:* J_n *mit* $n \in \mathbb{N}^+ = \{2,4,6,8,10\} \rightarrow$ *Bestimmung des Schwerepotentials mittels Kugelfunktionen als Basisfunktionen | physikalische Parameter: mittlere Winkelgeschwindigkeit der Erde, usw. | abgeleitete physikalische Konstanten: Normalschwere am Äquator und Pol, usw.)* .

2.4.5.2 Variable Arrays

Ein Spaltenobjekt wird definiert als ein Objekt, *das innerhalb einer Tabelle als Spalte repräsentiert wird.* Ein variables Array erfüllt diese Definition. Im Gegensatz zur verschachtelten Tabelle wird aber das *VARRAY* in derselben Tabelle integriert (Tab. 2.10).

Im FIS-Datenmodell werden VARRAY-Datentypen verwendet, um Genauigkeitmaße zur Beurteilung der Qualität des GPS-Deformationsnetzes in Rumänien darzustellen (Abb. 2.7). Die übergeordnete Objektklasse *REFEINZELPUNKT_TAB* besteht aus der Einzelpunkt-Identität, den notwendigen Verknüpfungen zu weiteren Objektklassen (insbesonders zum Bereich Qualitäts- und Deformationsanalyse) und der Einzelpunktbeschreibung.

```
create or replace type EP_VA as VARRAY(3) of number;
/
create or replace type ANORDNUNG_TY as object (
  Zeile         number,
  Station       varchar2(10));
/
create or replace type EP_VKM_TY as object (
  EP_ID      varchar2(20),
  Anordnung     Anordnung_TY,
  EP_VKM        EP_VA);
/
create table EP_VKM_TAB of EP_VKM_TY;
/
```

Tabelle 2.10: *Anlegen des Collectors EP_VA als VARRAY mit der maximalen Anzahl von drei Elementen pro Datensatz zur Beurteilung der Stationsgenauigkeiten.*

Zusätzlich fungiert die Objektklasse als Aggregat der Teilobjektklasse *EP_VKM_TAB*. Unter Nutzung des VARRAY-Datentyps wird in der Tabelle *EP_VKM_TAB* eine (3×3)-Varianzkovarianzmatrix (VKM) erzeugt.

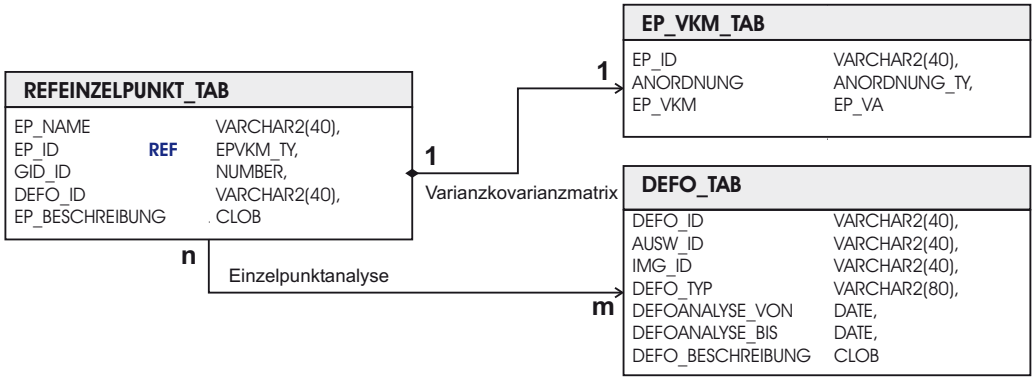


Abbildung 2.7: *Aufbau des Untermoduls „Einzelpunktanalyse“ unter Verwendung des Datentyps VARRAY zum Aufbau einer Matrix der Varianzen und Kovarianzen mit der Dimension* $[3 \times 3]$.

Die Struktur der VKM sieht folgendermaßen aus:

$$EP\_VKM = \begin{vmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{vmatrix} \tag{2.1}$$

Die Ausgabe der Tabelle *EP_VKM_TAB* im PL/SQL-Editor gibt die Genauigkeitsinformationen der Station Fundata zur Epoche 2003 aus der Einzelpunktanalyse (Kap. 6.2.4.2) wieder:

```
EP_ID           ANORDNUNG(Zeile, STATION)    EP_VKM
-------------------------------------------------------------------------
FUNDATA_2003    ANORDNUNG_TY(1, 'FUND_X')    EP_VA(0.5762, 0.1018, 0.2115)
FUNDATA_2003    ANORDNUNG_TY(2, 'FUND_Y')    EP_VA(0.1018, 0.4445, 0.1111)
FUNDATA_2003    ANORDNUNG_TY(3, 'FUND_Z')    EP_VA(0.2115, 0.1111, 0.6316)
```

2.4.6 Datum

Seit Einführung des Datentyps „*DATE*“ in SQL/92 und der umfangreichen Ausstattung von zusätzlichen Funktionen in SQL/99, können raumbezogene Daten zeitlich miteinander verknüpft werden. Damit besteht die Möglichkeit, Objekte mit vierdimensionalen Eigenschaften zu beschreiben. Die Merkmale des Datentyps „*DATE*“ sind folgende:

- Festlegung von Datum und Uhrzeit im Datentyp,
- Spezifizierung von Funktionen (Datumsberechnungen und Konvertierung von anderen Datentypen) und Formaten (Jahr, Quartal, Monat, Woche, Tag, Stunde, Minute, Sekunde, Millisekunde).

Mittels diesen Datentyps können Abläufe im FIS-Datenmodell historisch dokumentiert und Veränderungen zeitgerecht dargestellt werden. Dabei repräsentiert dieser Datentyp diskrete Zeitstempel zur Dokumentation des Projektverlaufes, der Einordnung der Qualitäts- und Deformationsanalysen, sowie der Positionen (Abb. 2.8) und Geschwindigkeiten der Stationspunkte auf der zeitlichen Skala von 1995 bis 2004 (Kap. 6.2.4.2). Kontinuierliche zeitliche Veränderungen zur Abstraktion der „realen Welt“ wurden im Datenmodell nicht berücksichtigt.

Die Objektklasse *REFPUNKT_TAB* wird mit der Objektklasse *SDO_MODELL_TAB* (Stationsobjekt) und weiteren Objektklassen (Geschwindigkeit, Genauigkeit und Zuverlässigkeit) über eine automatisch erzeugte Geometrie-Identität (GID) in eine $(1:1)-$Beziehung gesetzt. Die Eigenschaft von SDO_MODELL_TAB besteht aus der Beschreibung des Punktobjektes, der Geometrie, welche über ein *Spatial-Data-Objekt* (SDO) definiert wird, und dem Punktdatum, um eine zeitlich eindeutige Zuordnung zu erreichen. Damit kann das funktionale Modell der Deformationsanalyse (Kap. 6.2.4.3) in das Datenbankkonzept integriert werden.

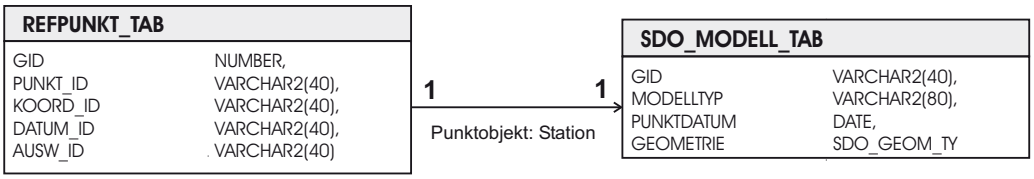


Abbildung 2.8: *Aufbau des Untermoduls Punktobjekt unter Verwendung des Spatial-Data-Objektes und des Datentyps DATE zur Festlegung der Stationskoordinaten.*

2.4.7 Methoden

Methoden werden als Funktionen (Member Function) definiert, *die Bestandteil einer Tabelle sind und Datenbank-Anweisungen oder einfache mathematische Operationen ausführen* (Tab. 2.11).

Folgender Aufbau beschreibt die Member Function:

- Kopfteil
  - Schlüsselwort angeben: Member,
  - Funktionsname definieren: Function [Name],
  - Parameterliste übergeben: Attribut, Parameterart und Datentyp,
  - Modus optional angeben: IN (nur Eingabe = default = call by reference), OUT (nur Ausgabe = call by result) oder IN OUT (Ein- und Ausgabe = call by value and result),
  - Wert der Member Function zurückgeben: return [Datentyp].
- Rumpfteil
  - Methode einleiten: begin,
  - Anweisung ausführen (z.B. mittlerer Punktfehler): $m_P = \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2}$,
  - Methode beenden: end.

Der Reinheitsgrad einer Methode wird mit der Anweisung *pragma restrict_references* in der Typ-Spezifikation festgelegt. Folgende Reinheitsgrade können angegeben werden:

```
create or replace type PKTFEHLER_TY as object (
  gid         number,
  sx          number,
  sy          number,
  sz          number,
  member function mp (sx IN number, sy IN number, sz IN number)
  return number,
  pragma restrict_references(mp, WNDS, RNPS, WNPS));
/
create or replace type body PKTFEHLER_TY
as
  member function mp (sx number, sy number, sz number)
  return number is
    begin
      return sqrt(power(sx,2) + power(sy,2)+ power(sz,2));
    end;
  end;
/
create table PKTFEHLER_TAB of PKTFEHLER_TY;
```

Tabelle 2.11: *Anlegen der Methode „mp“ unter Verwendung der Member Function zur Bestimmung des mittleren Punktfehlers.*

1. WNDS: Write No Database State (führt keine Datenbank-Anweisungen aus).
2. RNDS: Reads No Database State (führt keine select-Anweisung aus).
3. WNPS: Write No Package State (führt keine Änderungsanweisung der Package-Variablen aus).
4. RNPS: Reads No Package State (liest keine Package-Variablen).

Die Funktion *mp* führt eine mathematische Operation zur Berechnung des mittleren Punktfehlers von Stationspunkten im Kontext der Qualitätsanalyse aus (Kap. 6.2.4.1). Die Member Function wird im benutzerdefinierten Datentyp *PKTFEHLER_TY* integriert. Dabei wird die Member Function in einer zusätzlichen Anweisung *create or replace type body [UDT] as member function [Name]* dem Rumpf des Datentyps *PKTFEHLER_TY* zugewiesen.

Im FIS-Datenmodell werden Member Functions erzeugt, um Datenbank-Anweisungen und einfache mathematische Operationen innerhalb einer Objektklasse ausführen zu lassen (Tab. 2.12). Die Ausgabe des Punktfehlers der GPS-Station Fundata zur Epoche 2003 (Kap. 6.2.4.2) erfolgt unter Verwendung des Tabellenaliases *err* und dem Aufruf der Methode $mp(sx,sy,sz)$.

```
SQL> select sx, sy, sz, err.mp(sx,sy,sz) from PUNKTFEHLER_TAB err
  2> where gid = 1000025;

SX    SY    SZ    ERR.MP(SX,SY,SZ)
---------------------------------
1.8   1.5   3.2   4.0
```

Tabelle 2.12: *Ausgabe der Genauigkeiten sx, sy, sz und mp der GPS-Station Fundata 2003 mittels der select-Anweisung.*

2.5 Datenbankzugriff mittels JDBC

Das JDBC^{API} beschreibt die Spezifikation der *„Java Database Connectivity“* (JDBC), die entwickelt wurde, um eine einheitliche Datenbank-Schnittstelle für Java-Programme in mehrstufigen Server-Client-Architekturen zu realisieren. JDBC ist gleichzeitig Bestandteil der Java Plattform, welche zwei Programmpakete (Packages) bereitstellt: *java.sql* und *javax.sql* (Ellis et al., [23]). Damit wird ein Datenbankzugriff außerhalb des DBMS realisiert, um einerseits Daten zu manipulieren und andererseits einen vom Betriebssystem unabhängigen Datenfluss zu verwirklichen. Durch die Kompatibilität mit SQL/99 und die Interaktion mit XML (Kap. 5.1) wird JDBC als Schnittstelle zwischen der Datenbank und dem Applikationsserver zum Aufbau einer mehrschichtigen Architektur (Kap. 3) im FIS eingesetzt. Der Datenbankzugriff über das JDBC^{API} erfolgt aufgrund der Nutzung von vier aufeinander folgenden Klassen:

1. *Driver Manager*: Verwendung eines Datenbank-Treibers zum Verbindungsaufbau.
2. *Connection*: Herstellung der Datenbankverbindung.
3. *Statement*: Weiterleitung der SQL-Anweisungen an die Datenbank mittels der Statement-Klasse.
4. *Resultset*: Übergabe der Ergebnisse von Datenbankanweisungen mittels der Resultset-Klasse.

Die Applikationsentwicklung (Kap. 7) des Fachinformationssystems baut auf dem JDBC^{API} auf und besitzt folgende Eigenschaften:

- Gesicherter Zugriff auf die Datenbank über Eingabe eines Benutzers und eines Passwortes,
- Vorbereitung der SQL-Befehle über die *PrepareStatement*-Klasse und Ausführung über die *executeQuery*-Methode,
- Manipulation der Daten über von außen gesteuerte SQL-Anweisungen, sofern erfolgreiche Authentifizierung und Autorisierung (Kap. 3.2.7.3),
- Aufruf und Weiterverarbeitung der Datensichten in der FIS-Umgebung,
- Überwachung der Datenintegrität im DBMS.

Der Zugriff auf die Daten wird einerseits über den Sicherheitsmechanismus der Datenbank und andererseits über die Authentisierung am Applikationsserver (Kap. 3.2.7.3) gesteuert. Durch die Eingabe des Benutzers und des verschlüsselten Passwortes wird eine Datenbankverbindung aufgebaut (Tab. 2.13). Danach sind Datenbankabfragen erlaubt. Die Methode *getConnection(char[ ] password, String user)* enthält zwei Parameter: den Benutzer und das Passwort, die vom Hauptprogramm übergeben werden. Als Datentyp wurde eine Zeichenfolge (String) für den Benutzer und ein Zeichen-Feld (char[ ]) für das Passwort definiert. Die Variable *getaccess*, die als *boolean*-Datentyp deklariert wurde, legt den Rückgabewert fest, der durch die *return*-Anweisung den Wert *true* oder *false* zurückgibt. Der Zugang zum Fachinformationssystem wird bei *true* gewährt und bei *false* verweigert . Folgende Anweisungen ermöglichen den Datenbankzugriff mittels JDBC:

Class.forName(Treibername): der JDBC-Treiber (z.B. oracle.jdbc.driver) muss zuerst implementiert werden, um auf die Datenbank zugreifen zu können. Der Treiber wird durch den Treibermanager mittels des Konstruktors Class.forName(Treiber) geladen.

DriverManager.getConnection(URL, Benutzer, Passwort): der JDBC-Thin-Treiber, welcher eine reine Java-Lösung zum Verbinden einer Datenbank darstellt, wird verwendet, um die Datenbankverbindung zu ermöglichen. Durch die Angabe des Standortes der Datenbankquelle (Uniform Resource Locator = URL), des Benutzers und des Passwortes wird vom Treibermanager das Object *Connection* (con) zurückgegeben, sofern URL, User und Passwort korrekt eingegeben wurden.

Datenbankzugang: über Treiber.URL $\Longleftrightarrow$ jdbc:datenbank:thin:@Hostname:Port:Instanzname

```
public boolean getConnection(char[] password, String user)
throws SQLException, ClassNotFoundException
{
  try
  {
    Class.forName("oracle.jdbc.driver.OracleDriver");
    String url1 = ("jdbc:oracle:thin:@");
    String url2 = (":1521:Instanz");
    String net8 = (url1 + "SERVER" + url2);
    final String passwd1 = new String(password);
    con = DriverManager.getConnection(net8, user, passwd1);
    String dbcon = ("Connection to the database is established.");
    getaccess = true;
    oracle_message = "Datenbank ist hochgefahren";
    System.out.println(dbcon);
    return getaccess;
  }
  catch(Exception ne)
  {
    System.out.println("Datenbankverbindung nicht erfolgreich!");
    oracle_message = (ne.toString());
    getaccess = false;
    return getaccess;
  }
}
```

Tabelle 2.13: *Realisierung des Datenbankzugriffes mittels JDBC 3.0. Die Methode getaccess(arg1, arg2) gibt den Rückgabewert „Zugang gewährt = false oder true“ zurück.*

Ein Bestandteil des Fachinformationssystems ist die SQL-Editor-Benutzeroberfläche (Kap. 7.8), welche dem Benutzer ermöglicht, Datenbankabfragen auszuführen (Tab. 2.14). Folgende JDBC-Anweisungen sind erforderlich, um *Select*-Anfragen an die Datenbank zu senden und Antworten zu erhalten:

Methode getConnection( ): das Objekt vom Typ *Connection* (con), welches das Interface *java.sql.Connection* implementiert, wird bei erfolgreicher Datenbankverbindung zurückgegeben. Statement-Objekte werden unter Verwendung der Objekte vom Typ Connection erstellt.

Methode prepareStatement(SQL-Anweisung): es wird eine Methode definiert, die Anweisungen deklariert und zum Vorkompilieren an die Datenbank übergibt. Rückgabewert ist ein Objekt vom Typ *PreparedStatement* (ps), welches das Interface *java.sql.PreparedStatement* implementiert. Mittels der vorkompilierten SQL-Anweisungen können die formalen Parameter durch aktuelle Werte ersetzt und die Abfragen beliebig oft ausgeführt werden. Unter Verwendung der PreparedStatement-Objekte wird damit die Performance der Datenbankabfragen (z.B. in einer Schleife) gesteigert.

Methode executeQuery( ): die Methode führt die SQL-Anweisung mit den aktuellen Werten durch. Als Rückgabewert wird ein Objekt vom Typ ResultSet (result) geliefert.

Methode getMetaData( ): die Methode gibt ein Objekt vom Typ *ResultSetMetaData* (md) zurück. Das Objekt enthält Metadaten über die Tabelle, welche über das ResultSet-Objekt angefordert wurde (z.B. Spaltenname, Spaltenanzahl, Spaltenlänge, Spaltentyp, ...).

Methode getColumnCount( ): die Integer-Variable *colmax* enthält die Anzahl der Spalten der angeforderten Tabelle, die über die Methode bestimmt wurde.

Methode getWarnings( ): die Methode gibt ein Objekt vom Typ *SQLWarning* zurück, wenn eine Warnmeldung von der Datenbank ausgelöst wird.

Objekt result: das Objekt enthält alle Daten, die von der SQL-Anweisung angefordert wurden.

- result.next( ): die nächste Zeile wird ausgegeben, solange bis das Tabellenende erreicht wurde.
- result.getString( ): der Rückgabewert ist eine Zeichenkette vom Typ *String* (es wurden Methoden für verschiedene Datentypen get[Typname]( ) definiert, welche im Anhang A.4 vorgestellt werden).
- result.wasNull( ): der Rückgabewert ist ein Wert vom Typ *boolean*, welcher auf *true* gesetzt wird, wenn das Zeilenobjekt vom Typ *null* ist.

Stringbuffer [name] = new StringBuffer( ): es wird ein Speicherpuffer für die Datenobjekte angelegt, welche einem definierten Objekt vom Typ *Vector* hinzugefügt werden → *vector.addElement(buf.toString())*.

```
/* Import Klassen, Schnittstellen vom Paket java.sql */
import java.sql.Connection; import java.sql.PreparedStatement;
import java.sql.SQLWarning; import java.sql.ResultSetMetaData;
import java.sql.ResultSet;
. . .
/* JDBC-Objekte definieren */
Connection con = null;
PreparedStatement ps = null;
ResultSet result = null;
/* select-Anweisung vorbereiten */
String query = "select [user].[attribut1],... from [tablename]";
/* Datenbankverbindung aufbauen und Anfrage bearbeiten */
con = getConnection();
ps = con.prepareStatement (query);
result = ps.executeQuery();
ResultSetMetaData md = result.getMetaData();
int colmax = md.getColumnCount();
SQLWarning sqlw = con.getWarnings();
. . .
/* Tabelleninhalte in einem Vektor speichern */
Vector vector = new Vector();
. . .
while (result.next()) {
  StringBuffer buf = new StringBuffer();
  for (int j = 0; j < colmax; j++) {
    buf.append (result.getString (j+1));
    boolean nullobject = result.wasNull();
    . . .
  }
  vector.addElement(buf.toString());
}
. . .
```

Tabelle 2.14: *Realisierung einer Datenbankabfrage mittels JDBC 3.0, indem eine select-Anweisung erstellt und ausgeführt wird.*

Kapitel 3

Referenz-Architektur

Bestandteile vernetzter Referenzarchitekturen sind neben Datenbasis und DBMS, Dienstgeber und Dienstnehmer, die miteinander verknüpft sind (Lockemann/Dittrich, [52]). Da die Komponenten eng miteinander in Beziehung stehen, so dass sie nicht mehr als autonome Bausteine einer Architektur angesehen werden können, werden sie in Schichten linear aufeinander gelegt. Jeder dieser Dienste baut eine Schnittstelle zur über- bzw. untergeordneten Schicht auf. Damit ist die Datenübertragung zwischen den Ressourcen, welche durch das DBMS verwaltet werden, und den Benutzern sichergestellt (Abb. 3.1).

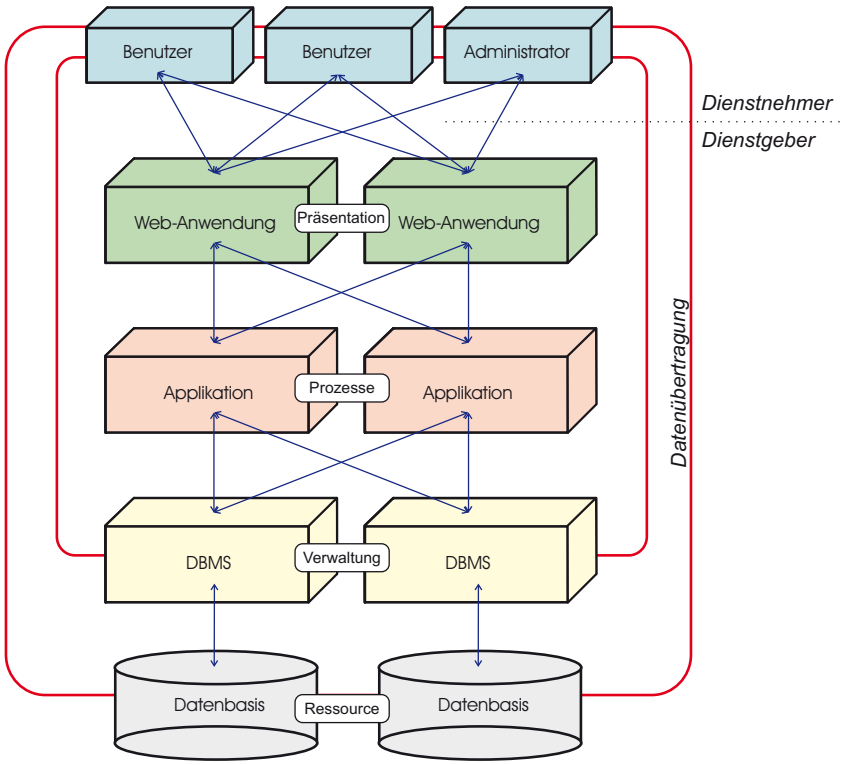


Abbildung 3.1: *Aufbau einer Mehr-Schichten-Architektur. Im Mittelpunkt steht die Middleware, die eine Interaktion zwischen Dienstgeber und Dienstnehmer aufbaut.*

Die Middleware ist ein Datenübertragungssystem, das sachlogische Ebenen berücksichtigt (Lockemann/Dittrich, [52], S. 70). Die Ebenen sind folgendermaßen aufgebaut:

1. Transportdienste: Aufbau sicherer und zuverlässiger Netzverbindungen.
2. Service-oriented Architecture: Aufbau der Dienste-Interoperabilität → Dienstnehmer und Dienstgeber bilden mit dem Verzeichnisdienst ein Verbindungsdreieck, wobei der Zugriff zwischen Dienstnehmer

und Dienstgeber stattfindet, die Registrierung zwischen Dienstgeber und Register erfolgt und sich die Bindung zwischen Register und Dienstnehmer einstellt. Dabei läuft der Austausch der Informationen zwischen den Diensten im Hintergrund.

3. Anwendungsdienste: Sicherstellung der technischen Interoperabilität, d.h. die Fähigkeit zur Kommunikation zwischen technischen Komponenten (z.B. Identifizierung und Autorisierung).

4. Datenübertragung unter Berücksichtigung der Sicherheit, Zuverlässigkeit und Robustheit (z.B. Registrierung der Transaktionsprotokolle).

5. Physikalische Speicherung der Datenquelle unter Berücksichtigung der Performance bei der Speicherung bzw. Datenübertragung.

3.1 Schichten-Architektur

Das Design von aktuellen Informationssystemen besteht aus Schichten, welche die Systemarchitektur in verschiedene Ebenen aufteilt. Dabei greifen höhere Schichten auf die niederen Schichten zu, die im Idealfall direkt darunter liegen. Allgemein wird die Systemarchitektur in folgende Komponenten dargestellt:

1. *Graphical User Interface*: grafische Benutzeroberfläche (GUI),

2. *Fachkonzept*: der Anwendungskern wird von Präsentation und Datenspeicherung gelöst. Änderung an der Datenbasis oder an der Präsentationsebene hat keinen Einfluss auf die Applikationsebene.

3. *Datenhaltung*: mittels einer Datenbank und eines DBMS wird die Manipulation der Daten von der Anwendung entkoppelt.

3.1.1 Ein-Schichten-Architektur

Die Ein-Schichten-Architektur ist ein monolithisches System. Der Zugriff, die Anwendungen und die Datenverwaltung finden auf einem Rechner statt.

3.1.2 Zwei-Schichten-Architektur

Die Zwei-Schichten-Architektur ist eine Client/Server-Architektur, welche die Datenhaltung und die Anwendungen physikalisch trennt (Abb. 3.2).

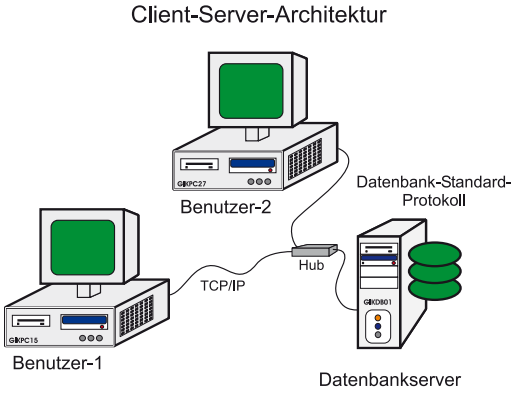


Abbildung 3.2: *Darstellung der Zwei-Schichten-Architektur. Die Applikationen und das Datenmanagement sind getrennt eingerichtet. Die Kommunikation erfolgt über ein Standardprotokoll. Mehrbenutzerbetrieb ist möglich.*

Der Zugriff auf das DBMS erfolgt durch mehrere Benutzer, die an verschiedenen Rechnern ihre Anwendungen ausführen. Die Clients können direkt über ein Standardprotokoll *z.B. Transmission Control Protocol/Internet Protocol* (TCP/IP) mit dem Server kommunizieren. Der Datenaustausch ist somit über verschiedene, miteinander verbundene Netzwerke möglich. Aufgrund des eindeutigen Hostnamens (z.B. in Abbildung 3.2: GIKPC15, GIKPC27 und GIKDB01) bzw. der eindeutigen IP-Adresse kann eine DB-Verbindung zum Server aufgebaut werden. Die IP-Adresse besteht aus folgenden beiden Teilen:

- Netzwerk-ID (172.*xx*.255.255),
- Rechner-ID (172.*xx*.*xxx*.*xxx*).

Zusätzlich wird eine Maske für das Teilnetz (Subnet-Mask: z.B. 255.255.0.0) definiert, in dem sich der Host befindet. Der im FIS eingesetzte Datenbankserver legt folgende Parameter zur eindeutigen Identifizierung im Netz fest:

- DB-Dienstname: Client kann sich direkt beim DB-Server anmelden,
- Protokoll: *TCP/IP*,
- Host-Name: *GIKDB01*,
- Port: *1521* (Standard-Port).

3.1.3 Drei-Schichten-Architektur

Die Drei-Schichten-Architektur besteht aus der Datenverwaltungs-, Applikations- und Präsentationsschicht (Abb. 3.3).

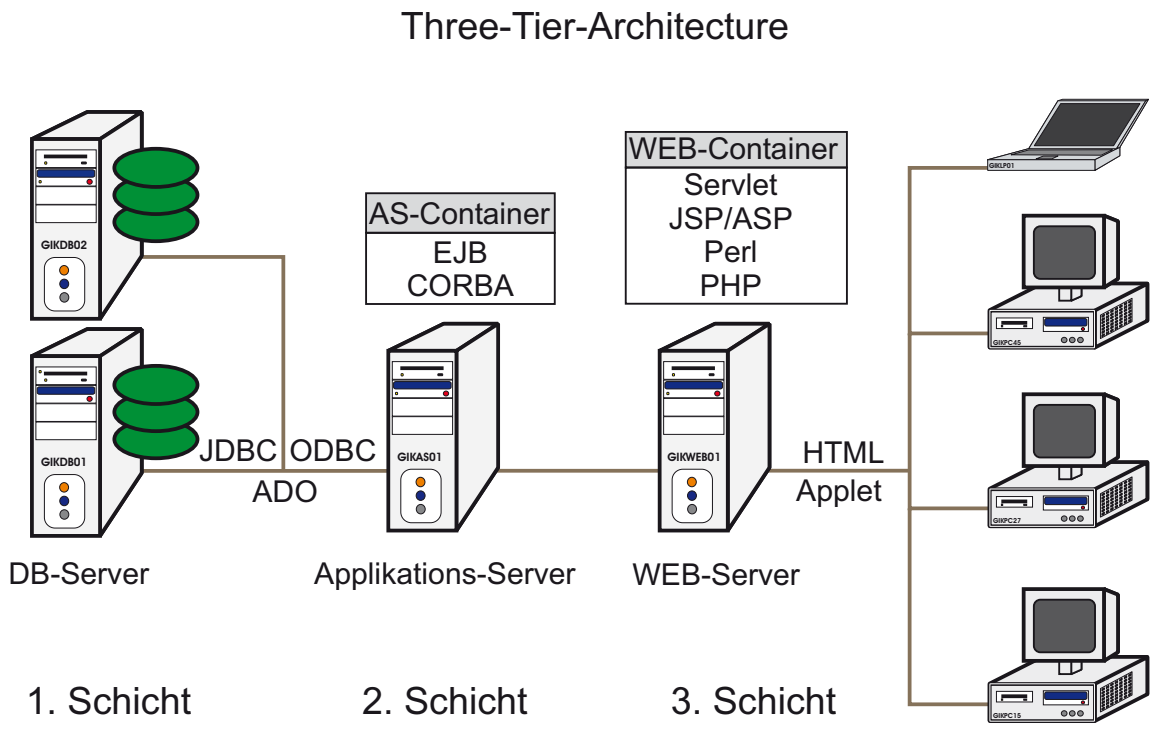


Abbildung 3.3: *Darstellung der Drei-Schichten-Architektur (1. Präsentations-, 2. Applikations- und 3. Datenbankschicht).*

Diese Architektur entkoppelt erstens die Präsentationsebene von der Anwendungsebene, zweitens die Anwendungsebene von der Datenbankverwaltungsebene und erfüllt drittens folgende Randbedingungen:

- Integration unterschiedlicher Systemkomponenten,
- Fusion unterschiedlicher Plattformen,
- Vereinigung verschiedener Protokolle,
- hohe Skalierbarkeit des Gesamtsystems (sehr große Anzahl von Clients, sehr hohe Datenübertragungsrate),
- technische und dienstliche Interoperabilität,
- hohe Flexibilität und Dynamik (Austausch bzw. Erweiterung der Schichten-Architektur),
- hohe Performance beim Anfrage/Antwort-Verhalten.

Der Applikationsserver (AS) kommuniziert mittels Schnittstellen (Tab. 3.1) mit dem Datenbankverwaltungssystem, welches die Daten nach erfolgreicher Verbindung überträgt.

	Beschreibung der Datenbank-Schnittstellen
JDBC	Die Schnittstelle *„Java Database Connectivity“* enthält Klassen und Methoden in JAVA (java.sql und javax.sql), die einen Datenbankzugriff und eine Übertragung der Daten ermöglichen (Treibermanager → Verbindung → Anweisung → Antwort → Kap. 2.5).
ODBC	Die Schnittstelle *„Open Database Connectivity“* wird als Standard-Schnittstelle zur Datenübertragung (DBMS ↔ Client) eingesetzt. Über eine *JDBC-ODBC-Bridge* können JDBC- und ODBC-Treiber miteinander kombiniert werden.
ADO	Die von Microsoft entwickelte Active-X Komponente *„ActiveX Data Object“* ist eine Schnittstelle, um auf Datenbanken zugreifen zu können.

Tabelle 3.1: *Beschreibung diverser Datenbank-Schnittstellen, die den Zugriff zwischen DBMS und AS steuern (Basler, [4], S. 177).*

Der WEB-Server übermittelt die Anfragen des Clients an den AS, der in Form von server- oder clientseitigen Anwendungen (Tab. 3.2) die Anfragen bearbeitet. Der Benutzer ruft mithilfe eines Browsers clientseitige Java-Applets bzw. HTML-Seiten auf, die Datenbankanweisungen senden können, oder greift unter Verwendung von implementierten Anwendungsprogrammen auf den WEB-Server zu.

	Beschreibung der serverseitigen Applikationen
Servlet	*Servlets* sind serverseitige Java-Programme, die vom WEB-Server geladen werden. Sie ermöglichen zur Datenübertragung, eine Verbindung zum Client und zur Datenbank aufzubauen.
EJB	*Enterprise Java Bean*-Architektur ist eine Komponentenarchitektur für die Entwicklung und Bereitstellung von komponentenbasierten und verteilten Geschäftsanwendungen (Kap. 3.2.2). Open-Source Applikationsserver verwenden diese Architektur, um z.B. Datenbankobjekte dem Client bereitzustellen.
CORBA	*Common Object Request Broker*-Architektur ist analog zur EJB-Architektur eine Komponentenarchitektur, die komplexe Netz-Anwendungen entwickelt. Die Architektur, welche Komponenten zur Spezifizierung und Entwicklung von Geschäftsmethoden bereitstellt, ist unabhängig von den eingesetzten Programmiersprachen (C++, Java, Smalltalk, COBOL, usw.).
JSP/ASP	*Java Server Pages* und *Active Server Pages* sind Technologien, die dynamische WEB-Anwendungen entwickeln. Der Code wird in HTML eingebettet und greift unter Verwendung einer geeigneten Schnittstelle (JDBC/ADO) auf Datenbankobjekte zu.

	Beschreibung der serverseitigen Applikationen
Perl	*Perl* (Akronym für Practical Extraction and Report Language) ist eine Open Source Programmiersprache, die in verschiedenen Bereichen (Netzwerk-Programmierung, Systemadministration, WEB/GUI-Entwicklung, DB-Anwendungen) eingesetzt wird.
PHP	*PHP* (Akronym für Hypertext Preprocessor) ist eine Open Source Skriptsprache, die speziell für WEB-Anwendungen entwickelt und in HTML eingebettet wird. Mittels PHP können serverseitige DB-Anwendungsprogramme implementiert werden.

Tabelle 3.2: *Beschreibung von serverseitigen Applikationen, die den Zugriff zwischen AS und WEB-Server steuern.*

3.1.4 Mehr-Schichten-Architektur

Grundlage der Mehr-Schichten-Architektur ist die in Kapitel 3.1.3 beschriebene Drei-Schichten-Architektur. Zwischen diesen Hauptschichten werden zusätzliche Schichten eingefügt (Grimm, [26]), die eine deutlichere Trennung zwischen Datenhaltung, Geschäftslogik und Präsentation erreichen (Abb. 3.4). Diese Schichten werden generell als *Abstraktions*- und *Zugriffsschichten* bezeichnet. Sie dienen dazu, heterogene Datenobjekte soweit aufzubereiten, dass sie über systemunabhängige und normierte Schnittstellen der Präsentationsschicht übergeben werden können. Gleichzeitig bieten die Zugriffsschichten ein Höchstmaß an Sicherheit bei der Datenübertragung.

Datenbankschicht: Verwendung von Datenbanken zur Datenhaltung und Datenbankverwaltungssystemen als Steuerungs- und Kontrollsysteme. Die DBMS lassen sich in folgende Gruppen einteilen:

1. relationale, objektrelationale und objektorientierte DBMS,
2. ERP-Systeme in Kombination mit SAP/R3-Systemen (Managementsysteme für Industrie- und Wirtschaftsunternehmen, wie z.B. Projektmanagement-, Vermögensmanagement- oder Lagerverwaltungssysteme) oder
3. XML-Datenbanken.

Zugriffschicht: Aufbau von Authentisierungs- und Autorisierungsdiensten zwischen Datenbank-Server und Applikations-Server (Kap. 3.2.7).

Applikationsschicht: Entwicklung von Anwendungsprogrammen, die Objektklassen aus verschiedenen Datenbankverwaltungssystemen aufbereiten (Kap. 4.2.1).

Persistenzschicht: Modellierung der Datenlogik und ihrer Methoden als persistente Objekte, deren Zustand in der Datenbank orts- und zeitunabhängig ist. Mittels Standard-Schnittstellen werden die Objektklassen zwischen Applikations-Server und Datenbank synchronisiert, so daß bei einem Systemabsturz die Objektklassen schnell wiederhergestellt werden können (Kap. 4.1.2).

Integrationsschicht: Herstellung von Verbindungen zu externen Datenverwaltungssystemen über normierte Schnittstellen, sowie deren Steuerung und Kontrolle über den Integrationsmanager.

Personalisierungsschicht: Personalisierung bezeichnet in der Informationstechnik, *die Anpassung von Programmen an die persönlichen Fähigkeiten eines Benutzers*. Dabei werden explizite (Veränderung durch den Nutzer) und implizite Personalisierungen (Anpassung aufgrund des Nutzerverhaltens) berücksichtigt.

Prozessmanagementschicht: Modellierung, Steuerung und Überwachung von Geschäftslogik-, Projekt- und Unternehmensprozessen.

Metadatenschicht: Aufbau von Metadaten für die im Informationssystem entwickelten Objektklassen. Unter Metadaten versteht man strukturierte Daten, *mit deren Hilfe eine Informationsressource beschrieben und dadurch besser auffindbar gemacht wird*. Als Standard für interoperable Metadaten wird ein generisches Framework spezifiziert, welches aus den vier Aspekten *Semantik, Datenmodell, Syntax* und *Identifikation* besteht.

Zugriffschicht: Aufbau von systemunabhängigen und normierten Authentisierungs- und Autorisierungsschnittstellen zwischen Web-Server und Applikations-Server (Kap. 3.2.7.3).

Präsentationsschicht: Darstellung der übertragenen Datenobjekte in einem spezifischen GUI oder WEB-Browser (Kap. 7).

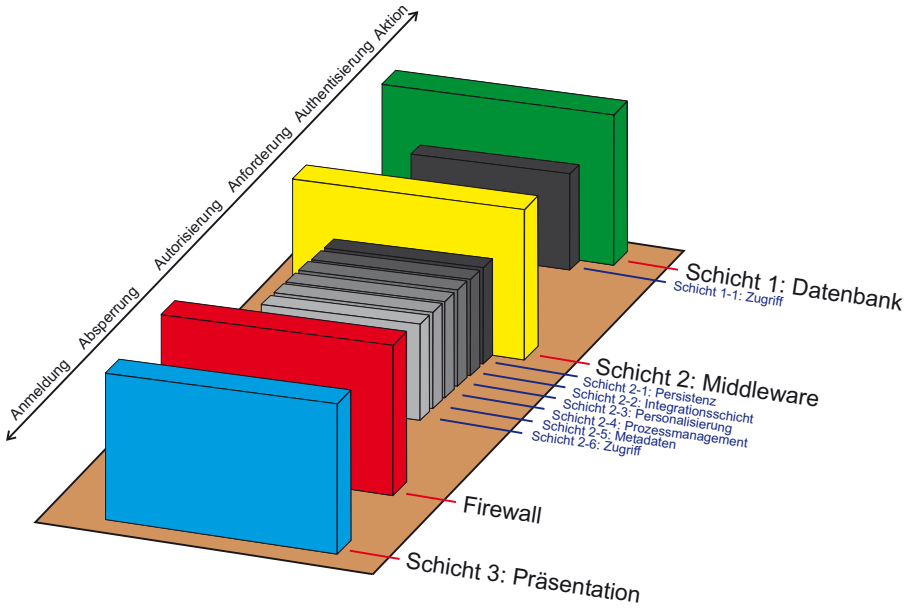


Abbildung 3.4: *Darstellung der Mehr-Schichten-Architektur. Die Architektur gliedert sich in mehrere Schichten: die Datenbankschicht, die Middlewareschicht, diverse Schichten zur Modellierung, Steuerung und Überwachung von Prozessen und Diensten sowie die Präsentationsschicht.*

Die Kommunikation zwischen der Präsentations- und Applikationsschicht verläuft über Netzwerkprotokolle (Tab. 3.3). Diese Protokolle definieren die Form, das Format und die Übernahmeart der Daten. Außerdem sind sie für die Überwachung der zu übertragenden Daten und die Zustellung an einen bestimmten Empfänger verantwortlich.

	Beschreibung einiger Netzwerkprotokolle
HTTP	Das *HyperText Transfer* Protokoll ist ein zustandsloses Protokoll und wird im weltweiten Web eingesetzt. Zustandslos bedeutet in diesem Zusammenhang, dass das Protokoll solange existent ist, bis die Anfrage am Server gestellt wurde. Danach sind die Informationen über den Client, der die Anfrage gestellt hat, dem Server nicht mehr verfügbar. Es wird kein Kontext zwischen verschiedenen HTTP-Requests gespeichert. Bei dieser Art der Übertragung kann keine sichere Datenverbindung gewährleistet werden. HTTP definiert das Format und die Übertragung der Daten und bestimmt die Reaktionen der WEB-Server und der Browser auf verschiedene Befehle.
LAN/WAN	Das *Local* und *Wide Area Network* erstreckt sich über ein begrenztes Gebiet, indem die Benutzer über eine Telefon- bzw. Funkverbindung miteinander kommunizieren können.
SOAP	Das *Simple Object Access* Protokoll ist eine Entwicklung des World Wide Web Konsortiums (W3C), um WEB-Dienste für XML-basierte Anwendungen anzubieten. Es besteht aus drei Teilen: 1. Rahmenbedingung der Meldung, 2. Instanziierung der Datentypen und 3. Konventionen von Methoden zur Datenübertragung.

Tabelle 3.3: *Erläuterung verschiedener Netzwerkprotokolle, die in Schichtenarchitekturen verwendet werden.*

3.2 FIS-Architektur

Das Drei-Schichten-Modell (Three Tier Architecture) ist Grundlage der FIS-Architektur. Dabei wurden Schwerpunkte in der Zugriffsberechtigung sowie der Steuerung und Kontrolle der Geschäftsprozesse gesetzt (Abb. 3.5).

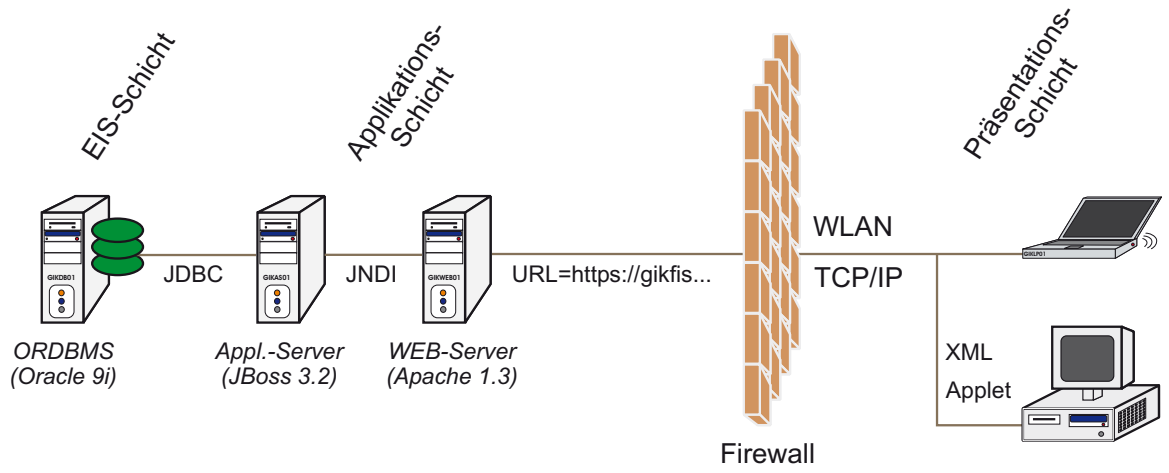


Abbildung 3.5: *Darstellung der Drei-Schichten-Architektur des Fachinformationssystems. Die Architektur gliedert sich in drei Schichten: die EIS-Schicht, die Applikationsschicht und die Präsentationsschicht.*

EIS-Schicht: die *Enterprise Information System*-Schicht besteht aus einer objektrelationalen Datenbank (Oracle Version 9i). Mittels der JDBC-Schnittstelle werden die Daten zum Applikationsserver übertragen.

Applikationsschicht: der Applikationsserver (JBoss Version 3.2) enthält einen EJB-Container (Kap. 4.1.4), der aus *zustandslosen Session-Beans* besteht. Session-Beans erledigen festgelegte Aufgaben und werden eingesetzt zur Erfüllung einer Anforderung (Kap. 4.2). Danach sind sie nicht mehr existent. Sie implementieren lokale und entfernte Interfaces, die Methoden zur Datenübertragung definieren. Mittels der Klassen *javax.naming.InitialContext* und *javax.rmi.PortableRemoteObject* werden eindeutige JNDI-Namen definiert und bei einer Daten-Anforderung kontrolliert (Kap. 3.2.3). Falls die Auflösung der Bean-Namen nicht erfolgreich war, wird eine Fehlermeldung *java.rmi.RemoteException* ausgelöst. Der WEB-Server (Apache Version 1.3) empfängt die vom AS ausgesandten Datenobjekte über die *RMI-IIOP*-Schnittstelle zur weiteren Datenübertragung.

Präsentationsschicht: über die URL *„https://gikfis.gik.uni-karlsruhe.de“* wird das FIS-Applet geladen. Mittels des gesicherten Hypertext Transfer Protokolls (HTTPS) werden die Informationen entweder im geladenen Applet verschlüsselt übergeben oder als XML-Objekte beziehungsweise in Form von HTML im Web-Browser dargestellt. Zusätzlich ist zwischen WEB-Server und Client eine Firewall geschaltet, die externe Netzangriffe verhindern soll. Die Datenübertragung zwischen Applikationsserver und dem Web-Server erfolgt über diverse normierte und systemunabhängige Schnittstellen (Tab. 3.4).

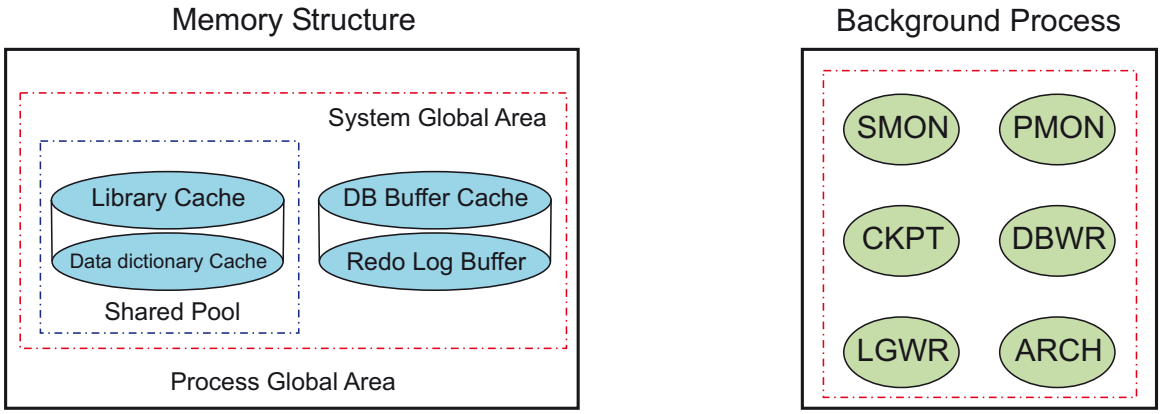
API	Beschreibung der Schnittstellen
JNDI	Die Schnittstelle *„Java Naming and Directory Interface“* bietet eine einheitliche Programmierschnittstelle an und ermöglicht für Java-Applikationen einen transparenten Zugriff auf Namens- und Verzeichnisdienste (Backschat/Gardon, [2]). JNDI ist im J2SE integriert und wird zur Lokalisierung von Enterprise JavaBeans (EJB) eingesetzt.
JMS	Der Dienst *„Java Message Service“* wird in Java-Programmen eingesetzt, die einen Zugriff auf eine nachrichtenorientierte Middleware (MOM) benötigen (Chappel/Monson-Haefel, [14]).
JTA	Das *„Java Transaction API“* steuert diverse Transaktionen in Anwendungsprogrammen (Perrone et al., [63]).

RMI-IIOP	Der Methodenaufruf *„Remote Method Invocation“* (RMI) erlaubt einen Zugriff auf Objekte, die nicht lokal angelegt sind (Remote-Objekte). Das Interface java.rmi.Remote wird implementiert, um Remote-Objekte aufzurufen. In Kombination mit dem Protokoll *„Internet InterOrb Protocol“* (IIOP) werden EJB- bzw. CORBA-Objekte vom AS zum WEB-Server übergeben.

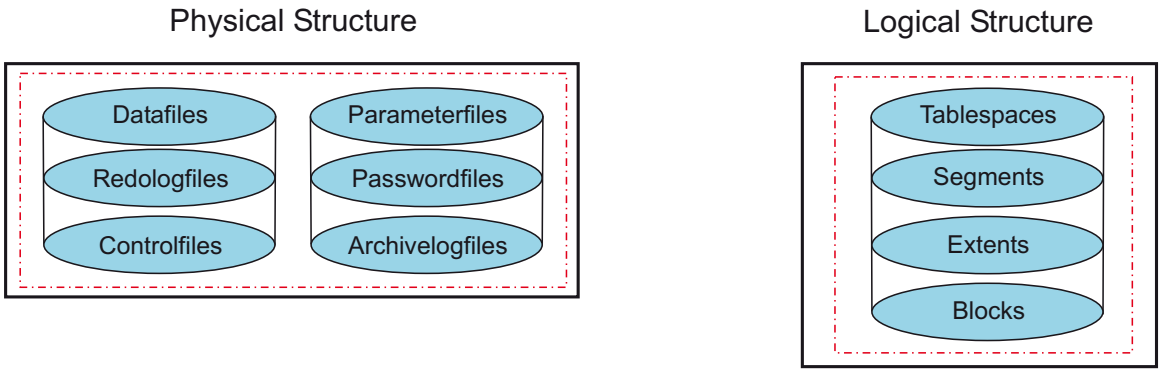
Tabelle 3.4: *Beschreibung verschiedener APIs, die den Zugriff zwischen AS und WEB-Server steuern.*

3.2.1 Architektur des Oracle-Datenbankservers

Die Hauptkomponenten der DB-Architektur bestehen aus der *Datenbank* und der *Instanz*. Die *Speicherstruktur* und die *Hintergrundprozesse* bilden die Instanz von Oracle (Abb. 3.6.1). Die Datenbank unterteilt sich in die Bestandteile *physikalische* und *logische Struktur* (Abb. 3.6.2).



3.6.1: Die Instanz untergliedert sich in die Komponenten „Speicherstrukturen“ und „Hintergrund-Prozesse“.



3.6.2: Die Datenbank besteht aus der „physikalischen und logischen Struktur“.

Abbildung 3.6: *Darstellung der Architektur des Datenbankservers von Oracle (Version 9i). Die Architektur setzt sich aus der Instanz und der Datenbank (Adkoli/Velpuri, [1], S. 121).*

3.2.1.1 Memory Structure

Die *Speicherstruktur* (Memory Structure) besteht aus dem globalen Prozessbereich (Process Global Area) und dem globalen Systembereich (System Global Area). Verschiedene Pufferspeicher (Buffer Cache) werden für die Bibliothek (Library), das Datenbeschreibungsverzeichnis (Data Dictionary), die Oracle-Datenbank und die Wiederherstellungsprotokolldateien (Redo Log) bereitgestellt.

Process Global Area: der PGA enthält einen größeren Speicherbereich, der vom Server bereitgestellt wird, um spezifische Benutzerprozesse ausführen zu können. Es werden z.B. die Login-Informationen im PGA gespeichert.

System Global Area: der SGA reserviert einen Speicherbereich für Daten und Kontrollinformationen der Instanz. Beim Start der Datenbank wird dem SGA ein maximaler physikalischer Speicherbereich *SGA_MAX_SIZE* für den *Shared Pool, Buffer Cache, Large Pool und Java Pool* zugeordnet. Der SGA lässt sich in gesamten Speicherbereich, in festen und variablen Größen sowie in benötigten Pufferspeicher für die Datenbank und Wiederherstellungsdateien darstellen. Der Speicherbereich für das Fachinformationssystem wurde folgendermaßen festgelegt:

```
Total System Global Area  135338868 bytes
Fixed Size                   453492 bytes
Variable Size             109051904 bytes
Database Buffers           25165824 bytes
Redo Buffers                 667648 bytes
```

Shared Pool: die Speicherstruktur besteht aus dem Pufferspeicher für die Bibliothek und dem Datenbeschreibungsverzeichnis. Der Shared Pool wird zum Parsen und Kompilieren von SQL-Statements für *Library* und *Data Dictionary* verwendet. Über den Parameter SHARED_POOL_SIZE kann die Größe des Shared Pools dynamisch verändert werden.

Library: die Bibliothek enthält Informationen über SQL-Anweisungen und PL/SQL-Programme, die von der Datenbank ausgeführt werden. *PL/SQL*-Blöcke, ausführbare *Cursor*-Routinen, *Java*-Klassen und *SQL*-Anweisungen werden im *Library Cache* gehalten, welcher im Shared Pool angelegt wird. Dazu wird der LRU-Algorithmus (Last Recently Used) eingesetzt, um die zuletzt verwendeten PL/SQL-Statements zu speichern. Der Cursor wird als Zeiger auf einen Kontextbereich definiert, *der zur Verarbeitung einer SQL-Anweisung im Hauptspeicher reserviert wird*. PL/SQL-Programme steuern mittels des Cursors den Kontextbereich, der Informationen zu SQL-Anweisungen bereithält, welche zur Ausführung der Datenbankanfragen benötigt werden.

Data Dictionary: das Datenbeschreibungsverzeichnis legt Datenbankstrukturen fest, welche der Server zur Verwaltung der Datenobjekte verwendet. Der Inhalt setzt sich aus folgenden Eigenschaften zusammen:

- Informationen über den Benutzer, seine Privilegien und den zugeordneten Rollen,
- Namen und Beschreibung der Schemata (Tabellen, Indizes, Datensichten, Synonyme, Sequenzen, Cluster, Quelltypen und Benutzertypen),
- Integritätsbedingungen,
- Informationen über Datenbankstrukturen,
- Funktionen, Prozeduren und Programmgruppen.

Im Data Dictionary Cache werden die zuletzt verwendeten DB-Objekte gespeichert.

Database Buffer Cache: die Größe des Speicherbereiches der Oracle Datenbank wird über den Parameter DB_CACHE_SIZE eingestellt. Im Cache werden die Kopien der zuletzt verwendeten Datenblöcke unter Verwendung des LRU-Algorithmus gespeichert. Der Parameter DB_BLOCK_SIZE bestimmt die Größe der einzelnen Datenblöcke.

Redo Log: es wird ein Speicherbereich (Redo Log Buffer) reserviert, um Wiederherstellungsdateien zu protokollieren. Im Falle eines Systemabsturzes werden diese Dateien verwendet, um die Datenbank mit dem letzten gesicherten Zustand hochzufahren. Für jede Änderungsoperation, die durch eine Transaktion ausgelöst wird, werden zwei Protokollinformationen erzeugt: eine *Redo*- und eine *Undo*-Nachricht. Das Recovery-Verfahren benötigt zusätzlich die *Log*-Einträge mit folgenden Parametern:

1. *Log Sequenznummer*: eindeutige Kennung der Log-Einträge,
2. *Transaktionskennung*: eindeutige Kennung der Transaktionen, die Änderungen durchführen,

3. *Seiten-Identität*: eindeutige Kennung der Seiten, die Änderungen festhalten,

4. *Zeiger*: eindeutige Markierung auf den vorhergehenden Log-Eintrag der auszuführenden Transaktion.

Im Redo Log Buffer werden die Änderungen von Datenblöcken gespeichert. Mittels der *commit*-Anweisung wird eine Transaktion abgeschlossen und die Informationen aus dem Buffer in eine Redo-Log-Datei gespeichert.

3.2.1.2 Background Process

Der Datenbankserver benötigt zur Verwaltung der Datenbank diverse *Server-Prozesse*, die spezifische Funktionen besitzen. Diese lassen sich in drei Komponenten klassifizieren: 1. Serverprozesse (Master), 2. untergeordnete Dienstprozesse (Slave) und 3. Hintergrundprozesse (Background). Die *Hintergrundprozesse* sind Bestandteile der Oracle-Instanz. Sie sind verantwortlich, anstehende Aufgaben zu koordinieren und zu bewältigen, die durch gleichzeitige Benutzung der Datenbank entstehen.

SMON: der *System-Monitor* ist zuständig für die automatische Wiederherstellung der Instanz. Der Prozess verwaltet Speicherplatz, führt Datenbank-Jobs aus und optimiert die Größe der Rollback-Segmente.

CKPT: der *Checkpoint*-Prozess wird aktiviert, um Änderungen am Datenbestand zeitlich festzuhalten. In der Regel benutzen mehrere Clients gleichzeitig die Datenbank. Dabei werden bei Suchanfragen, die Daten in einem Zwischenspeicher abgelegt. Simultan können Änderungen an den Daten vorgenommen werden. Nach der Änderung müssen die Daten aktualisiert den anderen Benutzern zur Verfügung stehen. Der alte Zustand wird im Zwischenspeicher gelöscht. Das Intervall zur Setzung der Checkpoints wird über den Parameter LOG_CHECKPOINT_INTERVAL in der Initialisierungsdatei eingestellt:

$$\text{Log\_Checkpoint\_Interval} = \frac{\text{Größe der Redo-Log-Datei}}{\text{Interval} * \text{Diskblock Size}} \tag{3.1}$$

LGWR: der *Log-Schreiber* übergibt Daten an die Redo-Log-Bücher, die vorher im Redo-Log-Speicherpuffer zwischen gespeichert waren. Dieser Prozess wird bei einer *COMMIT*-Anweisung aktiv.

PMON: der *Process-Monitor* detektiert nicht benutzte Server-Prozesse und entfernt diese. Außerdem werden andere Hintergrundprozesse überwacht, indem sie neu gestartet werden, falls sie unzulässig abgebrochen wurden.

DBWR: der *Datenbank-Schreiber* überträgt Daten vom Block-Zwischenspeicher in die Daten-Dateien.

ARCH: der *Archiver*-Prozess erstellt Kopien der Transaktions-Logbücher sofern die Datenbank im *ARCHIVE LOG*-Modus betrieben wird. Damit ist eine Wiederherstellung eines alten Zustands der Datenbank sichergestellt. Außerdem wird ein Protokollwechsel eingeleitet, wenn der Speicherumfang einer Redo-Log-Datei erreicht wurde.

3.2.1.3 Physical Structure

Die *physikalische Struktur* besteht aus einer Menge von Dateien, die zur Verwaltung von Daten in einer Datenbank notwendig sind.

Datafiles: Daten-Dateien enthalten die Datenobjekte und deren Daten. Sie sind in der Regel einem Tablespace (System, Users, Temp, ...) zugeordnet, wobei das Tablespace mit mehreren Daten-Dateien verbunden sein kann. Mittels folgender SQL-Anweisung wird eine Daten-Datei mit maximaler Speichergröße erzeugt:
⇒ *alter database datafile Dateiname.dbf autoextend on maxsize _M;*

Redologfiles: Wiederherstellungsdateien zeichnen die Änderungen des Datenbankzustandes z.B. während einer Transaktion auf. Falls ein unerwartetes Ereignis (Stromausfall, Platten-Crash) auftritt, kann der vorherige

Zustand wiederhergestellt werden. Es werden mehrere Kopien dieser Dateien (bestmöglich auf verschiedenen Festplatten) angelegt, um eine Wiederherstellung zu garantieren. Dazu werden Checkpoints mit folgender SQL-Anweisung gesetzt und die Redo-Log-Dateien hinzugefügt:

⇒ *alter system checkpoint;*

⇒ *alter database add logfile Log-Name, ... size _M;*

Controlfiles: Kontrollparameter werden gespeichert, die notwendig sind, um die Datenbank zu starten. Dabei werden die Daten- und Redo-Log-Dateien identifiziert sowie der aktuelle Zustand der Datenbank mittels eines Zeitstempels markiert. Die Kontroll-Datei sollte als Kopie auf verschiedenen Festplatten gespeichert werden, um den Start der Datenbank nach einem Platten-Crash sicherzustellen. Folgende SQL-Anweisung wird zum Erzeugen einer neuen Kontrolldatei benötigt:

⇒ *create controlfile database Datenbankname;*

Parameterfiles: zur Konfiguration der Datenbank werden Initialisierungsparameter definiert, welche verschiedene Ressourcen definieren und die maximale Größe festlegen. Beim Start der Datenbank werden die Parameter für die Instanz wirksam, die durch einen eindeutigen System Identifier (SID) gekennzeichnet ist. Folgende SQL-Anweisung ist zur Realisierung einer Initialisierungsparameter-Datei notwendig:

⇒ *create PFILE = '/oracle/ora92/database/dateiname.ora' from spfile '/oracle/ora92/database/spfile';*

Passwordfiles: die Datei enthält den Benutzernamen und das Passwort, um sich bei der Datenbank anzumelden. Der Benutzer erhält Privilegien und Rollen, um bestimmte DB-Aktionen (z.B. Backup, Wiederherstellung, Ausführung von Java-Applikationen, XML-Datenbankadministration, usw.) ausführen zu können. Mittels der folgenden SQL-Anweisung wird ein Nutzer mit den Privilegien als Systemadministrator verbunden:

⇒ *conn[ect] Benutzername [/Passwort][@Dienstbenennung][as sysdba];*

Archivelogfiles: sobald die Datenbank im Archive-Modus gestartet wird, werden sämtliche Log-Dateien archiviert. Damit werden Online-Backups und die Wiederherstellung der Datenbank zum letzten Sicherungszeitpunkt möglich. Folgende SQL-Anweisungen sind notwendig, um zuerst den Archive-Modus zu aktivieren und anschließend die Datenbank zu öffnen:

⇒ *alter database archivelog;*

⇒ *alter database open;*

3.2.1.4 Logical Structure

Die *logische Struktur* besteht aus Speicherelementen, die hierarchisch aufgebaut sind (Kap. 2.2).

Tablespaces: Tablespaces sind Datendateien, welche für das System, Benutzer, Indexe und temporäre Objekte angelegt werden.

Segments: Segmente enthalten eine Menge von Extents, welche Tabellen, Indexe und temporäre Daten speichern. Segmente werden logisch in einem Tablespace und physikalisch in Dateien gespeichert.

Extents: Extents werden aus einer bestimmten Anzahl von benachbarten Blöcken gebildet.

Blocks: Blöcke bilden die kleinste Speichereinheit.

3.2.2 Architektur des JBoss-Applikationsservers

Der Applikationsserver stellt Werkzeuge zur Entwicklung von Komponenten zur Verfügung, um Geschäftsmethoden im Rahmen von J2EE im EJB-Container-Konzept zu implementieren und die Infrastruktur *Java Management Extensions* (JMX^{API} v2.0) als Hauptbestandteil der Server-Architektur zu unterstützen (Abb. 3.7). Die JBoss-Architektur basiert auf einen Micro-Kernel Ansatz, welcher dem Administrator ermöglicht, Komponenten zur Laufzeit einzufügen. Die Server-Plattform bietet Dienste für *Persistenz*, *Transaktionen*, *Sicherheit*, *Namensverwaltung*, *Nachrichten*, *Login* und *Geschäftskomponenten* an. Diese Systemdienste werden über einen Web-Server gesteuert.

JMX-Umsetzung: JMX steht für Java Management Extensions. Diese Technologie stellt Werkzeuge zur Verwaltung, Anpassung und Überwachung für folgende Aufgaben zur Verfügung:

- Entwicklung von web-basierten und dynamischen Anwendungen,
- Realisierung von verteilten und modularen Anwendungen,
- Unterstützung von Netzwerk-Diensten.

JMX spezifiert vier Typen: standard, dynamische, offene und modellierte *MBeans*, die von JBoss direkt verwaltet werden (Managed Beans). Sämtliche MBeans werden von JMX-Agenten gesteuert, welche eine Anzahl von Diensten im MBean-Server bereitstellen. Eine MBean wird durch die Implementierung der Schnittstelle und ihrer Methoden nach der JMX-Spezifikation definiert.

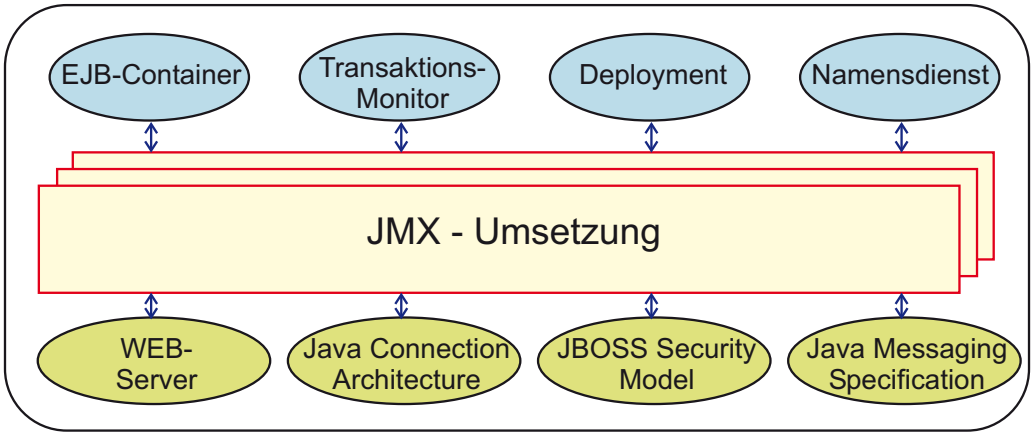


Abbildung 3.7: *Übersicht über die Hauptkomponenten der JBoss-Architektur (Version 3.2.6). Die Architektur setzt sich aus der Infrastruktur „Java Management Extensions“ zusammen, die Standard-JBoss-Komponenten bedient (Stark/JBoss Group, [76], S. 42).*

EJB-Container: der Container enthält Home- und Remote-Objekte sowie deren Bean-Instanzen (Kap. 3.2.3). Mittels definierter Schnittstellen werden Dateninformationen vom Datenbankserver zum Client und umgekehrt übertragen. Die EJB-Spezifikation legt das Verhalten und Zusammenspiel der Geschäftskomponenten fest und regelt somit den Ablauf des Datentransfers.

Transaktionsmonitor: Transaktionsmonitore bestehen aus Softwarekomponenten, welche den Ablauf einer großen Anzahl von gleichzeitig gestarteten Transaktionen steuern. Ihre Komponenten bestehen aus Message-Queueing-Systemen (nachrichtenorientierte Kommunikationssysteme), Transaktions-, Log-, Ressource- und Recovery-Manager, Rollback-Funktionen, Scheduler (Verteilung eingehender Anweisungen auf die einzelnen Serverprozesse), Lock-Manager (Blockierung von Datenbanktabellen) und weiteren Steuerungsfunktionen. Die Transaktionsmonitore zeichnen sich durch hohe Verfügbarkeit bei schnellem Antwortverhalten und Integrität der gleichzeitig genutzten Ressourcen aus. Allgemein bestehen Transaktionsverarbeitungssysteme aus folgenden Komponenten:

- Anwendungen,
- Netzwerksteuerung,
- Entwicklungswerkzeuge,
- Transaktionsmonitor,
- Datenbank.

Deployment: die Inbetriebnahme der Beans erfolgt durch die *Deployment-Deskriptoren*, die das Verhalten der Enterprise Beans zur Laufzeit anpassen können. Sobald eine Bean-Klasse und deren Schnittstellen definiert sind, werden die Deskriptoren zur Bean-Klasse angelegt, die sämtliche Laufzeitattribute enthalten, um die Entity-, Session- oder Message-Driven-Beans (nachrichtengesteuerte Beans) zur Laufzeit zu verwalten oder zu kontrollieren.

Namensdienst: die Bean-Container-Interaktion beinhaltet auch den Java Namens- und Verzeichnisdienst (JNDI), der für jede Bean einen spezifischen Namensraum bereitstellt. Mittels des Umgebungs-Namens-Kontextes (ENC) wird sichergestellt, dass der eindeutige JNDI-Name aufgerufen wird. Gleichzeitig kann im EJB-Container auf andere Beans und Container-Ressourcen zugegriffen werden (Kap. 3.2.3). Zur FIS-Authentisierung wird von der Datenbankzugangs-Bean auf die Ressourcen der OracleDS-Datei zugegriffen, um eine Verbindung über den Applikationsserver zur Oracle-Datenbank herzustellen (vollständiger Programmcode in Anh. B.2, C.1, C.2 und C.3).

```
## Verzeichnis Meta-Inf
## XML-Datei: JBoss
## JNDI-Name: DBAccessBean
<jboss>
    <ejb-name>DBAccessBean</ejb-name>
    <resource-ref>
        <res-ref-name>jdbc/FISDB</res-ref-name>
        <jndi-name>java:/OracleDS</jndi-name>
    </resource-ref>
</jboss>

## JBoss Server Konfiguration
## XML-Datei: Oracle-DS
## JNDI-Name: OracleDS
<datasources>
    <jndi-name>OracleDS</jndi-name>
    <connection-url>
        jdbc:oracle:thin:@servername:1521:DB-Aliasname
    </connection-url>
</datasources>
```

Web-Server: die serverseitigen Dienste werden im Apache-Server verwendet, um einerseits eine verschlüsselte Datenübertragung zu ermöglichen und andererseits das Applet dem Web-Client zum Aufbau des Fachinformationssystem in einem Java-fähigen Browser bereitzustellen (Kap. 3.2.4). Der Web-Server fungiert als Verbindungseinheit zwischen dem Benutzer und dem Applikationsserver. Die Anfrage des FIS-Anwenders wird nach Durchlaufen der Firewall per HTTPS dem Web-Server übertragen, welcher mit Hilfe der RMI-IIOP-Schnittstelle die Anforderung weiterleitet. Falls die Anfrage erfolgreich war, wird der Web-Server aktiv, um die Antwort dem Nutzer zurückzusenden. In der JBoss-Umgebung dient der Web-Server in Kombination mit der JMX-Konsole zusätzlich als Steuerungs- und Kontrollinstrument der serverseitigen Bean-Komponenten. Über http://localhost:8080/jmx-console/ werden alle im Deployment-Verzeichnis enthaltenen EJB-Beans als MBeans verwaltet.

Java Connection Architecture: JCA definiert eine Reihe von Interfaces, um Verbindungen zwischen Applikationsservern und Anwendern herzustellen. Damit ist die Interoperabilität und Portabilität zwischen AS und Client gewährleistet.

JBoss Security Model: das Sicherheitsmodell basiert auf einer Server-Container-Architektur, welche folgende Klassen, Methoden und Interfaces zum Abfangen von unerlaubten Zugriffen bereitstellt:

- Überprüfung der Identität und Validierung der Passwörter, digitale Signaturen, Sessionschlüssel, usw. .
  $\Rightarrow$ package org.jboss.security.EJBSecurityManager

- die *getPrincipal*-Methode übergibt die Identität des Benutzers dem betriebsbereiten System und bekommt die Identität der Anwendungsdomäne zurück. Mittels der *doesUserHaveRole*-Methode ist die Identität validiert und die Berechtigungsfunktionen der Anwendungsdomäne zugeordnet (Abgleich mit vordefinierten Benutzerrollen, welche im Deployment-Deskriptor festgelegt werden).
  ⇒ org.jboss.security.RealmMapping

- Die Klasse SecurityProxy erlaubt allgemeine Sicherheitskontrollen für home- und remote-Interfaces auf Grundlage des SecurityInterceptors.
  ⇒ org.jboss.security.SecurityProxy

Java Messaging Specification: die Spezifikation der Nachrichten ist Bestandteil der Java Message Service API. Damit werden diversen Anwendungen Komponenten zum Erstellen, Senden, Empfangen und Lesen von Nachrichten zur Verfügung gestellt.

3.2.3 Architektur des EJB-Containers

Die Enterprise Java Beans werden als komprimierte JAR-Dateien im Deployment-Verzeichnis des Applikationsservers abgelegt. Aufgrund der Eigenschaften der Deployment-Komponente werden die Beans in Betrieb genommen bzw. zur Laufzeit des Servers angepasst. Die Aufstellung der Bean-Komponenten erfolgt im EJB-Container. Innerhalb des Containers stehen die Entity-, Session- oder Message-Driven Beans mit ihren Instanzen in Beziehung (Abb. 3.8).

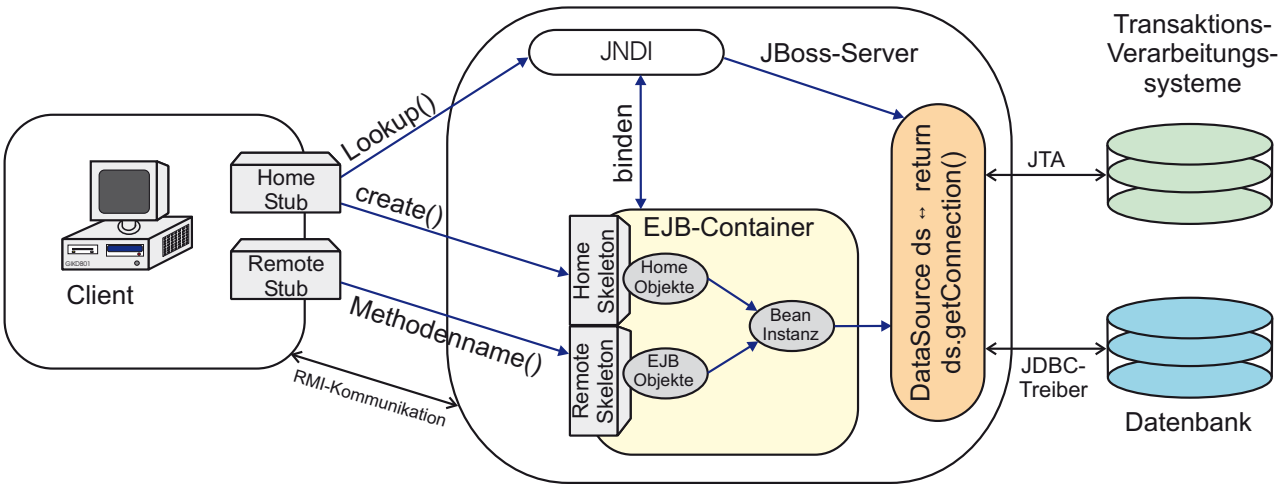


Abbildung 3.8: *Übersicht über das Konzept zur Entwicklung der JBoss-EJB-Container-Architektur. Im Container sind die EJB-, Home-Objekte und die Bean-Instanzen enthalten. (Liu, [51]).*

Der Client kann mittels geeigneter Schnittstellen (i.d.R. Einsatz des *RMI-IIOP*-Kommunikationsprotokolles) innerhalb des Containers auf die EJB-Objekte zugreifen. Zusätzlich besitzt der Container die Fähigkeit, Bean-Instanzen anzulegen oder freizugeben, falls diese Instanz nicht mehr benötigt wird. Der Vorteil liegt darin, dass die Speicherauslastung vom Container gesteuert und kontrolliert wird. Mittels der definierten Schnittstellen wird der Zugriff von außen gesteuert, so dass sich die Aufgaben des EJB-Containers auf die Verwaltung der Komponenten beschränken. Der Container verwaltet neben den Beans auch Remote-Objekte, die dort instanziiert wurden. Die Besonderheit der Remote-Objekte liegt darin, dass auf diese Objekte von Benutzern und diversen Prozessen außerhalb des Containers zugegriffen werden können. Daher unterstützen die Container rechner- und prozessübergreifende Aufrufe der Remote-Objekte (Backschat/Gordon, [2], S. 28-59).

Der Container besitzt außerdem die Eigenschaft, *Entity Beans* persistent zu halten. Die Entity Bean ist ein Komponententyp, welcher in der Lage ist, sich mit der EIS-Schicht zu verbinden. Dabei wird dieser Typ während der Laufzeit im Arbeitsspeicher gehalten und die Daten nach Vorgabe des Containers geladen bzw. gespeichert.

Der Abgleich der Daten wird in einem spezifizierten Zeitintervall vom Container vollzogen (*Container Managed Persistence*). Wenn die EJB-Container mit Transaktionsmonitoren über JTA-Interfaces verknüpft werden, werden die notwendigen Protokolle zur Ausführung der Transaktionen im Container gespeichert, da zahlreiche Clients gleichzeitig auf dieselben Daten zugreifen, um diese eventuell zu verändern. Somit wird der Transaktionsdienst von der AS-Seite aus organisiert und ausgeführt. Andererseits können Verbindungen zu relationalen Datenbanken mittels JDBC verwirklicht werden und Zugriffe direkt auf die Objektklassen erfolgen.

Um die instanziierten Beans korrekt auszuwählen, werden Namens- bzw. Verzeichnisdienste über das JNDI^{API} aufgebaut, die einen eindeutigen Bezug zu den Enterprise Java Beans herstellen. Dabei wird die Methode *Lookup( )* eingesetzt, um ein referenziertes Objekt jederzeit wiederzufinden, welches durch den Container in einem Verzeichnisbaum gebunden wird. Die Zuordnung der Objekte wird zwischen Client und EJB-Container über die Stub- und Skeleton-Objekte hergestellt. Diese Objekte sind Bestandteil der RMI-IIOP-API und stellen mit ihren entsprechenden Schnittstellen Referenzobjekte für Remote- und Home-Objekte dar. Folgender Ablauf wird zur Kommunikationsverbindung ausgeführt:

1. Schritt: Der Remote-Client sendet über definierte Geschäftsmethoden-Aufrufe eine Anweisung an das generierte Stub-Objekt.
2. Schritt: Die Anweisung wird an das Skeleton-Objekt weitergeleitet.
3. Schritt: Das Skeleton-Objekt übergibt die Client-Anfrage an das Remote-Objekt, das mit dem Skeleton-Objekt verbunden ist.

Mittels des Home-Interfaces (Kap. 4.1) werden die Methoden *create( ), find( )* und *remove( )* zum Erzeugen, Auffinden und Entfernen der EJB-Objekte bereitgestellt. Beim Installieren der Enterprise Java Beans werden die Home-Objekte automatisch generiert. Beim Aufrufen der Methode *create( )* wird ein EJB-Home-Objekt automatisch erzeugt und eine Instanz für das EJB-Objekt angelegt.

3.2.4 Architektur des Apache Web-Servers

Der Web-Server dient als Verbindungskomponente zwischen der Präsentations- und der Applikationsschicht. Folgende Java-Logik wird vom Web-Server unterstützt:

- *Client-seitige Web-Anwendung*: Java-Applets werden vom Web-Server in einen Java-fähigen Browser heruntergeladen.
- *Server-seitige Web-Anwendung*: Servlets und Java Server Pages werden vom Web-Server als selbständige Applikationen gesteuert.
- *DB-Integration*: Java gespeicherte Prozeduren werden in der Datenbank integriert. Der Zugriff auf die Objektklassen erfolgt über SQL-Anweisungen und unter Verwendung der JDBC^{API}.

Die FIS-Architektur setzt die Client-seitige Web-Anwendung (Fad Client/Thin Server) um. Der modulare Aufbau des Apache-Servers ermöglicht einen flexiblen Einsatz der Client-Server-Kommunikation. Die Architektur besteht aus vier Komponenten, die einen Kreislauf bilden (Abb. 3.9).

HTTP_Protokoll: das HyperText Transfer Protocol ist ein zustandsloses Datenaustausch-Protokoll zur Übertragung von Daten und stellt auf dem öffentlichen Port 80 die Verbindung zur Kommunikation zwischen Anwendungsschicht (Client) und Präsentationsschicht (Web-Server) her. Somit lässt sich dieses Protokoll der Ebene 7 des OSI-Referenzmodells (Kap. 3.2.5) zuordnen. Der Web-Browser eines Clients verwendet dieses Protokoll, um Daten vom Web-Server übertragen bzw. empfangen zu können. HTTP wurde zuerst für das World Wide Web (WWW) entwickelt, wird aber aufgrund seiner einfachen Struktur mittlerweile für den Austausch von beliebigen Daten (Anfragemethode → Headerinformation → Fehlercode) verwendet.

HTTP_Main: das Starten und Beenden des Server wird in der Hauptkomponente Main gesteuert und enthält die Schleife „main server loop“ zum Monitoring des Verbindungsaufbaues. Dabei werden die Ressourcen extrahiert

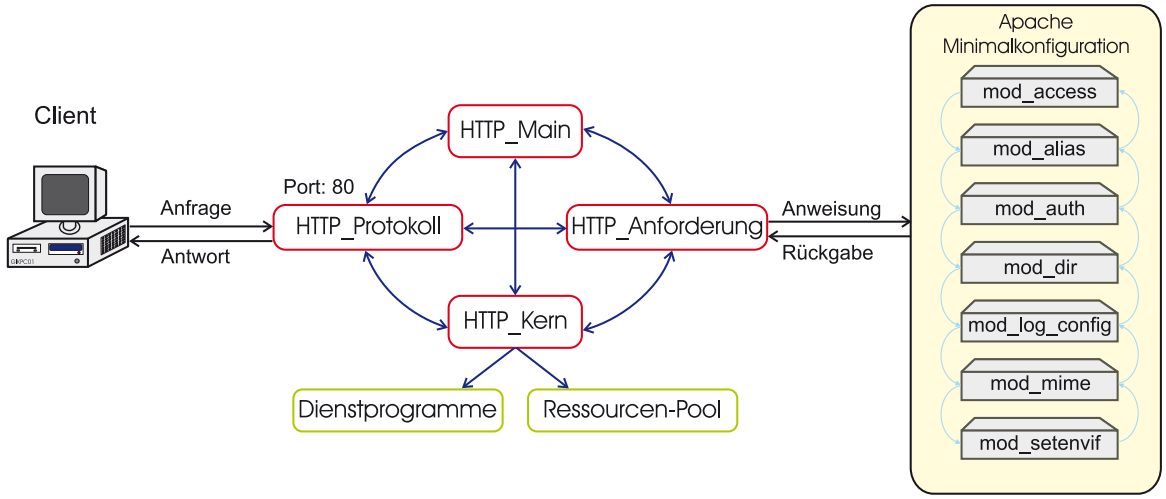


Abbildung 3.9: *Übersicht über die Architektur des Web-Servers (Dragoi/Preston, [22]). Darstellung der Minimalkonfiguration des Apache Web-Servers (Version 1.3).*

(Dateiname erkennen), angesprochen (Datei öffnen) und übermittelt (Datei senden). Nach Beantwortung der Anfrage wird die Verbindung geschlossen.

HTTP_Anforderung: die Anforderungskomponente bearbeitet den Datenfluss der Anfrage, steuert das Zusammenspiel der Module und behandelt die Fehlermeldungen.

HTTP_Kern: die Kernkomponente setzt allgemeine Funktionen um. Darunter fallen die Verwaltung und Kontrolle der Dienstprogramme, Unterstützung der virtuellen Hosts und das Reservieren von Speicher für die Ressourcen.

Die Module legen die Spezifikation des Apache Web-Servers individuell fest. Bei der Installation können verschiedene Konfigurationen (Minimal-, Standard-, Maximalkonfiguration oder eine individuelle bzw. empfohlene Konfiguration) ausgewählt werden. Aufgrund dieses offenen Konzeptes können eigene Module entwickelt und in die bestehende Architektur integriert werden. Die Module der Minimalkonfiguration besitzen Eigenschaften zur Konfiguration, Steuerung, Kontrolle und Authentisierung der HTTP-Requests und sind in Abbildung 3.9 als folgende Modulblöcke dargestellt (Thau, [82]):

mod_access: das Modul steuert durch Kontrolle des Hostnamens und der IP-Adresse den Zugang zum Web-Server.

mod_alias: infolge des Zugriffs auf den Server wird die URL durch das Modul manipuliert. Die *Alias*- und *ScriptAlias*-Anweisung bilden die URL-Pfade in Dateiverzeichnisse ab. Somit können Inhalte abgerufen werden, die nicht Bestandteil des Dokument-Hauptverzeichnisses sind:

```
## URL und Hauptverzeichnis-Pfad des Web-Servers.
ServerName   name@domäne.uni-karlsruhe.de
ServerRoot             "/usr/local/httpd"
## URL: http://www.domäne.uni-karlsruhe.de/index.html
DocumentRoot           "/usr/local/httpd/htdocs/"
## URL: http://www.domäne.uni-karlsruhe.de/icons/...
Alias        /icons/   "/usr/local/httpd/icons/"
## URL: http://www.domäne.uni-karlsruhe.de/cgi-bin/...
ScriptAlias  /cgi-bin/ "/usr/local/httpd/cgi-bin/"
```

mod_auth: das Modul stellt Direktiven zur Verfügung, die den Zugriff auf passwortgeschützte Dokumente oder Verzeichnisse konfigurieren. In der Konfigurationsdatei kann unter Verwendung der Directory-Anweisung ein Verzeichnis folgendermaßen passwortgeschützt werden:

```
## Passwortschutz für Verzeichnis Intern.
<Directory /"Hauptverzeichnis"/sfb461/Intern>
    AllowOverride AuthConfig
</Directory>
```

Im Verzeichnis, welches geschützt werden soll, wird die *.htaccess*-Datei definiert, welche folgende Parameter zur Authentifizierung des Benutzers festlegt:

```
## Argument zur eindeutigen Kennzeichnung festlegen.
AuthName "SFB 461"
## Festlegung des Authentifizierungstyps:
##   Digest --> verschlüsselte Passwortübertragung.
##   Basic  --> offene Passwortübertragung.
AuthType Basic|Digest
## Gruppen- und Passwort-Datei:
AuthUserfile /usr/local/httpd/conf/passwd
AuthGroupFile /usr/local/httpd/conf/sfbgroup
## Zugriff nur für gültigen Nutzer festlegen.
require valid-user
```

mod_dir: das Modul behandelt allgemeine Eigenschaften der Verzeichnisse. Die *DirectoryIndex*-Anweisung ermittelt automatisch die „index.html“-Datei und gibt diese zurück, sofern sie existiert. Ansonsten wird das Verzeichnis aufgelistet.

mod_log_config: das Modul unterstützt das Aufzeichnen der Client-Anweisungen. Die „access log“-Datei, welche das Herunterladen der HTML-Seiten vom Web-Server aufzeichnet, wird durch das Log-Format spezifiziert.

mod_mime: das Modul kennzeichnet basierend auf der Datei-Endung den Typ (text/html) der Datei, welche gesendet wird.

mod_setenvif: das Modul setzt Umgebungsvariablen basierend auf den Client-Informationen. Es kontrolliert die Umgebungsvariablen der CGI-Skripte sowie der SSI-Seiten (Server Side Includes), welche Instruktionen an den Server senden, um diverse Aufgaben zu erledigen (z.B. ein Skript auszuführen, wenn eine bestimmte HTML-Seite aufgerufen wird).

3.2.5 OSI-Referenzmodell

Protokolle dienen dazu, Daten zwischen zwei oder mehreren Schichten auszutauschen. Die dafür notwendigen Regeln werden standardisiert, um einen sicheren Verbindungs- und Datenaustausch zu gewährleisten. Das Architekturmodell *„OSI-Referenzmodell“* (ISO, [41]) wird in folgenden sieben Schichten unterteilt:

1. *Physikalische Schicht*: in der Physikalischen Schicht werden die Bitsequenzen in übertragbare Formate gewandelt. Außerdem sind die Eigenschaften der Übertragungsmedien (Kabel, Funk, Lichtwellenleiter) definiert sowie Steckverbindungen, Wellenlängen und Signalpegel.

2. *Verbindungsschicht*: die Verbindungsschicht organisiert und überwacht den Zugriff auf das Übertragungsmedium. Der Bitstrom wird auf dieser Ebene segmentiert und in Paketen zusammengefasst.

3. *Netzwerkschicht*: die Vermittlung und Zustellung von Datenpaketen übernimmt die Netzwerkschicht. Die Kopplung verschiedener Netzwerktopologien zur Weiterleitung der Pakete ist in dieser Ebene möglich.

4. *Transportschicht*: mittels der Transportschicht werden Verbindungen aufgebaut und beendet diese nach korrekter Datenübertragung. Dabei sind die Synchronisation der Verbindungen und die Verteilung auf mehrere Verbindungen wesentliche Bestandteile der vierten Ebene.

5. *Sitzungsschicht*: über die Sitzungsschicht laufen Dienste, die zur Organisation der Datenübertragung dienen. Somit können Verbindungen trotz zwischenzeitlicher Unterbrechung wieder aufgenommen werden.

6. *Präsentationsschicht*: in der Präsentationsschicht werden Daten für die Anwendungsschicht vorbereitet. Dabei werden Protokolle verwendet, die auf der Anwendungsebene in eine festgelegte Darstellung umgewandelt werden. Die Daten können einer weiteren Kontrolle unterzogen, verschlüsselt oder dekodiert werden.

7. *Anwendungsschicht*: per Anwendungsschicht werden Verbindungen zwischen Anwendungsprogrammen (z.B. E-Mail, HTML, Applet, usw.) hergestellt.

3.2.6 Architektur des FIS-Applets

JApplets sind Anwendungen, *die im Web-Browser geladen und in einer HTML-Seite eingebettet werden* (Abb. 3.10). Der Hierarchiebaum des FIS-Applets wird in Abbildung 3.10 vorgestellt (Krüger, [49]):

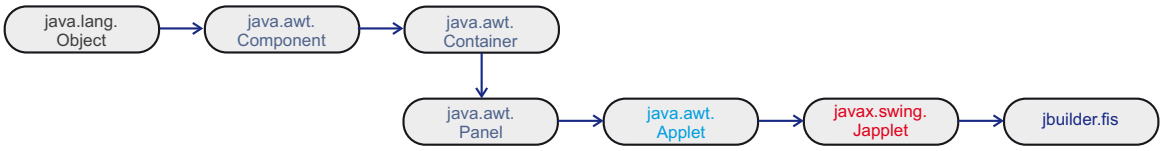


Abbildung 3.10: *Übersicht über die Architektur eines JApplets.*

Im Sinne der objektorientierten Programmiersprachen ist die Klasse *Object* die Superklasse im Vererbungsbaum des FIS-Applets. Die nachfolgenden Klassen erben die Eigenschaften der Vaterklasse beziehungsweise werden aus der nächst höheren Klasse abgeleitet (Tab. 3.5).

	Beschreibung der Klasse
Object	Die Klasse Object ist der Ursprung der Klassenhierarchie und wird jeder weiteren Klasse als Superklasse übergeben.
Component	Die Klasse Component ist ein Fensterobjekt, welches Komponenten (Schaltflächen, Ankreuzfelder, Bildlaufleisten, Textkomponenten, Listen, Kennsätze, Auswahl-Schaltflächen oder Kanevas) graphisch darstellt und in der Lage ist, mit dem Benutzer zu interagieren.
Container	Die abstrakte Klasse Container ist Bestandteil der Fensterklassen zur Erstellung graphischer Oberflächen (Abstract Windowing Toolkit - Akronym: AWT). Sie stellt Methoden bereit, um Komponenten aufzunehmen oder zu entfernen.
Panel	Die Klasse Panel ist eine einfache Container-Klasse, mit den Eigenschaften der Container- und Component-Klasse. Der Konstruktor *public void Panel(LayoutManager layout)* erstellt ein neues Feld und spezifiziert einen Layout-Manager.
Applet	Die Klasse Applet enthält die Rahmenbedingungen, um eine Anwendung mittels eines Web-Browsers auszuführen. Dabei ist das Applet im HTML oder im Applet-Viewer eingebettet.
JApplet	JApplet ist eine Unterklasse von Applet. JApplet wird initialisiert, wenn Swing-Komponenten verwendet werden.
fis	Die Objektklasse *fis* ist die Hauptklasse des Fachinformationssystems. Sämtliche FIS-Klassen, die am Aufbau des GUI beteiligt sind, werden im Package jbuilder zusammengefasst.

Tabelle 3.5: *Beschreibung des Hierarchiebaumes: Object ⇔ Component ⇔ Container ⇔ Panel ⇔ Applet ⇔ JApplet ⇔ fis (Sun Microsystems, [81]).*

3.2.6.1 Merkmale und Eigenschaften des FIS-Applets

Das Applet des Fachinformationssystems besteht aus folgenden Merkmalen und Eigenschaften:

- Angaben seiner Größe und Position,
- Aktionen, die ausgelöst werden,
- Ereignissen, die empfangen werden,
- Grafische Darstellung der Anwendungen.

Im Package *javax.swing.** sind die notwendigen Klassen enthalten, um eine Java-Anwendung als Applet zu entwickeln. Die Hauptklasse *„fis“* erbt sämtliche Eigenschaften der JApplet-Klasse und bildet den Kern des Fachinformationssystems. Über die *import*-Anweisung werden die benötigten Java-Klassen, die in unterschiedlichen Packages liegen, in die Hauptklasse geladen. Durch Verwendung des *new*-Operators wird eine neue Instanz der ursprünglichen Klasse erzeugt und auf die Objektvariable referenziert. Damit besitzt die neue Objektklasse sämtliche Eigenschaften der Originalklasse (Programmcode: Anh. B.1). Das JApplet wird zuerst instanziiert, indem der parameterlose Konstruktor mittels der *extends*-Anweisung der FIS-Klasse vererbt wird. Der allgemeine Aufbau der FIS-Klasse sieht folgendermaßen aus:

```
public class fis extends JApplet {
    public void init() {
        try {fachinformationssystem();}
        catch (Exception ex) {ex.printStackTrace();}
    }

    private void fachinformationssystem() throws Exception {
    /*** Aufbau des FIS-Applets ***/
        ...
    }
}
```

Einige Methoden werden explizit zum Verwalten des FIS-Applets vorgegeben (Tab. 3.6).

	Beschreibung der FIS-Methoden
public void init() {...}	Es wird die Methode aufgerufen, um das JApplet zu initialisieren.
private void fachinformationssystem() throws Exception {...}	Die Methode enthält alle notwendigen Eigenschaften zur grafischen Darstellung der fachrelevanten Informationen des Projektes „Dreidimensionale Plattenkinematik in Rumänien“.
public void start() {...}	Die Methode startet das JApplet. Im Gegensatz zur Methode init() kann diese Methode mehrfach aufgerufen werden.
public void stop() {...}	Die Methode beendet das JApplet. Diese Methode kann ebenso mehrfach aufgerufen werden.
public void destroy() {...}	Diese Methode vervollständigt die Funktionen zum Starten und Beenden eines JApplets.

Tabelle 3.6: *Instanziierung, Initialisierung, Starten, Stoppen und Zerstören des FIS-Applets.*

3.2.6.2 Aufruf des FIS-Applets im Web-Browser

Im Kontext der Internet-Präsentation wird das FIS-Applet über einen Java-fähigen Web-Browser geladen und somit einer breiten Öffentlichkeit zur Verfügung gestellt. Dabei wird das Applet unter Verwendung von einem *Applet-TAG* (TAG = Auszeichnungssymbol) in eine HTML-Seite eingebettet, welche der Browser in eine graphische Darstellung umsetzen kann.

Die Angabe der URL *https://gikfis.gik.uni-karlsruhe.de* entspricht folgendem Verzeichnispfad im Server: */mypath/htdocs/index.html*. Wobei ein Link (Name der Default-Startseite: *index.html*) auf die HTML-Seite *fis.html* gesetzt wird.

```
user@server:/mypath/htdocs> ls -la
-rw-r--r--   1 user   group      2848 Apr 14 10:10 fis.html
lrwxrwxrwx   1 user   group         8 Apr 14  2005 index.html -> fis.html
```

Die Applet-Klasse *fis.class* ist im Package *jbuilder* abgelegt, welches dem Verzeichnis */mypath/htdocs/lib* mit den restlichen Packages als komprimierte *Jar*-Datei hinzugefügt wurde (Abb. 3.11). Alle komprimierten Jar-Dateien, die zum Aufruf des Fachinformationssystems notwendig sind, werden im *lib*-Verzeichnis abgelegt.

```
user@server:/mypath/htdocs/lib> ls -la
-rw-r--r--   1 user   group    466184 Apr 14 12:10 gikfis.jar
-rw-r--r--   1 user   group   1522487 Apr 14 12:15 jbossall-client.jar
-rw-r--r--   1 user   group    140804 Apr 14 12:20 jdom.jar
```

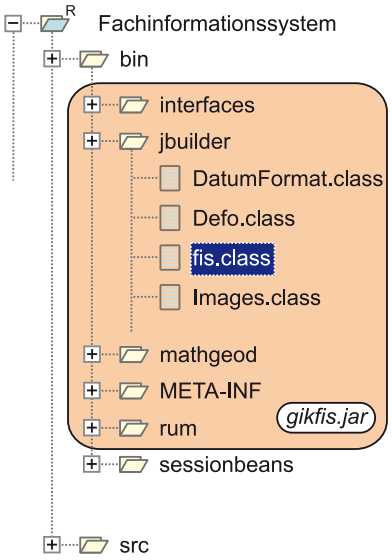


Abbildung 3.11: *Übersicht über den Verzeichnisbaum der Java-Klassen*

Die Applet-Anweisung *<APPLET>...</APPLET>* enthält mehrere Parameter, die für das Laden des FIS-Applets im Web-Browser eingesetzt werden:

```
<APPLET  CODE = "jbuilder.fis" CODEBASE ="."
    <!-- Laden der FIS-Bibliothek -->
    ARCHIVE = "lib/gikfis.jar, lib/jbossall-client.jar, lib/jdom.jar"

    <!-- Parameter zur Darstellung des Applets im Browser -->
    ALT = "FIS wird nicht gestartet!" WIDTH = 750 HEIGHT = 620 NAME = "fisclient">
</APPLET>
```

CODE: Angabe des Namens der Applet-Klasse (fis.class) → Bestandteil des Package jbuilder → Angabe des Package-Namens + Applet-Klasse (jbuilder.fis) ohne Dateierweiterung.

CODEBASE: das Dokumentenverzeichnis /mypath/htdocs/ - gekennzeichnet durch "." - wird zum Laden der Klassen verwendet.

ARCHIVE: sämtliche angegebenen JAR-Archive (/mypath/htdocs/lib/*.jar) werden geladen.

ALT: falls ein Browser Java nicht unterstützt, erscheint der Alternativ-Text *„FIS wird nicht gestartet!“*.

WIDTH/HEIGHT: Angabe der Größe des Applets.

NAME: Festlegung des Applets mit einem eindeutigen Namen. Somit können mehrere Applets unterschieden werden.

3.2.6.3 Zertifizierung des FIS-Applets

Applets werden in zwei Kategorien unterteilt: *vertrauenswürdig* und *nicht vertrauenswürdig*. Der wichtigste Unterschied besteht darin, dass vertrauenswürdige Applets keinen Sicherheitsbeschränkungen unterliegen (JBoss Group, [43]). Folgende Restriktionen sind bei Applets zu beachten, die als „nicht vertrauenswürdig“ eingestuft werden:

- Zugriff auf Dateien des Clients ist nicht erlaubt,
- Aufruf externer Programme auf dem Computer des Clients ist nicht erlaubt,
- Netzwerkverbindungen werden nur zwischen Benutzer und dem Host aufgebaut, von welchem das Applet geladen wurde.

Das Sicherheitskonzept von Java Development Kit (JDK 1.2 oder höher) sieht vor, dass die Berechtigungen über „Policies“ geregelt werden. Dabei besteht eine Policy aus Regeln, die Einschränkungen erstellt oder die standardisierten Sicherheitsrichtlinien modifiziert. Eine Regel wiederum enthält den Herkunftsort des Applets *(grant codeBase[URL, Pfadname])* und die Berechtigung *(grant[AllPermission, RuntimePermission, SocketPermission, PropertyPermission, SerializablePermission, ...])* als festen Bestandteil. Die Signatur bzw. Identität des FIS-Applets wird optional durch den Urheber „Geodätisches Institut Karlsruhe“ (GIK) gesetzt. Beim Start des Applets wird die Datei *java.policy* von der Java Runtime Environment (JRE) ausgelesen. Die Integration des JAVA^{TM}-Plugins von Sun als Ergänzungsmodul im Web-Browser ist Voraussetzung für das Laden von Java-Applets.

→ Java(TM) Plug-in: Version 1.4.2_xx
→ Verwendung der JRE-Version 1.4.2_xx Java HotSpot^{TM}Client VM
→ Home-Verzeichnis des Benutzers = /< Home-Pfad >/name

Wird die Berechtigung *„java.security.AllPermission“* gesetzt, werden alle Sicherheitsbeschränkungen aufgehoben. Dadurch haben nicht vertrauenswürdige Applets denselben Status wie vertrauenswürdige Applets. Die Berechtigungen zum Zugriff auf das FIS-Applet werden individuell gesetzt (Tab. 3.7).

Berechtigung	Package	Beschreibung
SocketPermission	java.net.*	Zugriff auf TCP/IP-Verbindungen.
RuntimePermission	java.lang.*	Zugriff auf Runtime-Klassen.
PropertyPermission	java.util.*	Zugriff auf Systemeigenschaften.
SerializablePermission	java.io.*	Zugriff auf Output/Input-Datenströme zur Überschreibung der serialisierten bzw. deserialisierten Objekte.

Tabelle 3.7: *Beschreibung der Sicherheitsrichtlinien im FIS-Applet. Vergabe der Berechtigungen in der Datei „java.policy“, welche von JRE ausgelesen wird (Middendorf et al., [57]).*

Die unabhängige Darstellung des Applets auf allen Java-fähigen Browsern ist Voraussetzung für das internetbasierte Fachinformationssystem, dessen Applikationsentwicklung ausführlich in Kapitel 7 beschrieben wird. Dabei müssen sämtliche Jar-Dateien, die über den Applet-Tag aufgerufen werden, signiert werden. Folgende Schritte sind auszuführen:

1. *Generierung*: die Generierung eines Schlüsselpaares erfolgt mit der *keytool*-Anweisung. Dabei wird eine eigene passwortgeschützte Schlüsseldatenbank (Dateiname = .keystore · Schlüsselpaare = private/public · Alias = <Name>) und ein dazugehöriges Zertifikat vom Administrator angelegt. Diese DB-Datei darf nicht öffentlich zugänglich sein, da Applets zur Erhaltung der Datenintegrität und Authentifizierung unter Verwendung dieser Schlüsselpaare signiert werden.
   ⇒*keytool -keystore <.keystore> −genkey −alias <Aliasname>*

2. *Zertifizierung*: das Erstellen eines vertrauenswürdigen Zertifikates ist Aufgabe der Zertifikatsstelle. Die signierten Zertifikate können exportiert, angezeigt und importiert werden.
   ⇒*keytool −export −alias <Aliasname> −file <Zertifikatsname.crt>*
   ⇒*keytool −print −alias <Aliasname −file <Zertifikatsname.crt>*
   ⇒*keytool −import −alias <Aliasname> −file <Zertifikatsname.crt>*

3. *Signierung*: die Jar-Bibliotheken werden mit der *jarsigner*-Anweisung signiert.
   ⇒*jarsigner −keystore /<Pfad>/.keystore −storepass <Passwort> <gikfis.jar> <Aliasname>*

Bevor das FIS-Applet gestartet wird, werden dem Benutzer diverse Sicherheitshinweise angezeigt, die zu akzeptieren oder abzulehnen sind (Abb. 3.12). Wird das Applet endgültig als vertrauenswürdig eingestuft, wird das Applet im Browser geladen.

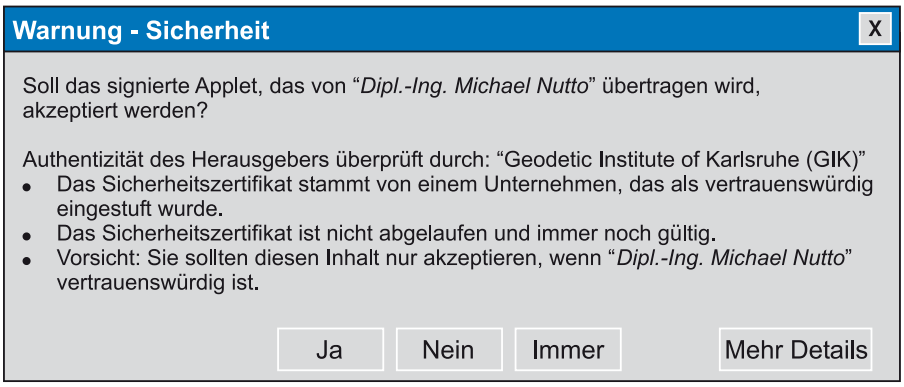


Abbildung 3.12: *Sicherheitshinweis beim Laden des signierten FIS-Applets. Der Client muss zustimmen beziehungsweise ablehnen, ob das Applet auf seinem Rechner gestartet werden darf.*

3.2.7 Sicherheit im Fachinformationssystem

Die Sicherheit ist einer der Hauptfaktoren, welche bei der Implementierung des Fachinformationssystems berücksichtigt wurde. Die Einhaltung der Sicherheitsrichtlinien ist Voraussetzung, um die sensiblen Daten des Sonderforschungsprojektes im Internet präsentieren zu können. Dabei wird eine Firewall zwischen Benutzer und Web-Server geschaltet, um Angriffe aus dem Internet bzw. unerlaubte Zugriffe auf den Applikationsserver zu verhindern (Kap. 3.2.7.1).
Die Daten werden unter Verwendung zertifizierter Protokolle per HTTPS verschlüsselt übertragen (Kap. 3.2.7.2). Die Feststellung der Identität eines Benutzers und seine Zugriffsberechtigung vom Applikationsserver wird mittels der von Java bereitgestellten Authentisierungs- und Autorisierungsdiensten gesteuert (Kap. 3.2.7.3). Die Datenbank-Anweisungen werden zusätzlich über Sicherheitsrollen überprüft, die im Zuge des Datenbankdesigns für diverse Benutzergruppen angelegt wurden (Kap. 3.2.7.4). Somit werden Datenbankmanipulationen von unberechtigten Personen verhindert. In der Abbildung 3.13 werden Sicherheitsmechanismen zum Aufbau einer mehrschichtigen Architektur unter Berücksichtigung der Benutzerrechte dargestellt:

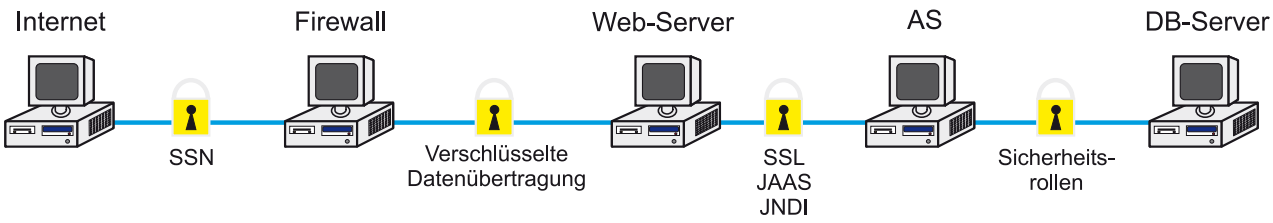


Abbildung 3.13: *Integration der Sicherheitskomponenten im Kontext internetbasierter Informationssysteme.*

3.2.7.1 Firewall

Es wird ein Firewall-System aufgebaut, um ein sicheres Servernetz (SSN) einzurichten, wobei dessen Zugriffe auf das GIK-Netz oder externen Netzen klar geregelt werden. Das Servernetz besteht aus WWW-, Mail- bzw. FTP-Servern, die mit dem Universitätsnetz verbunden sind. Dabei gelten folgende Regeln:

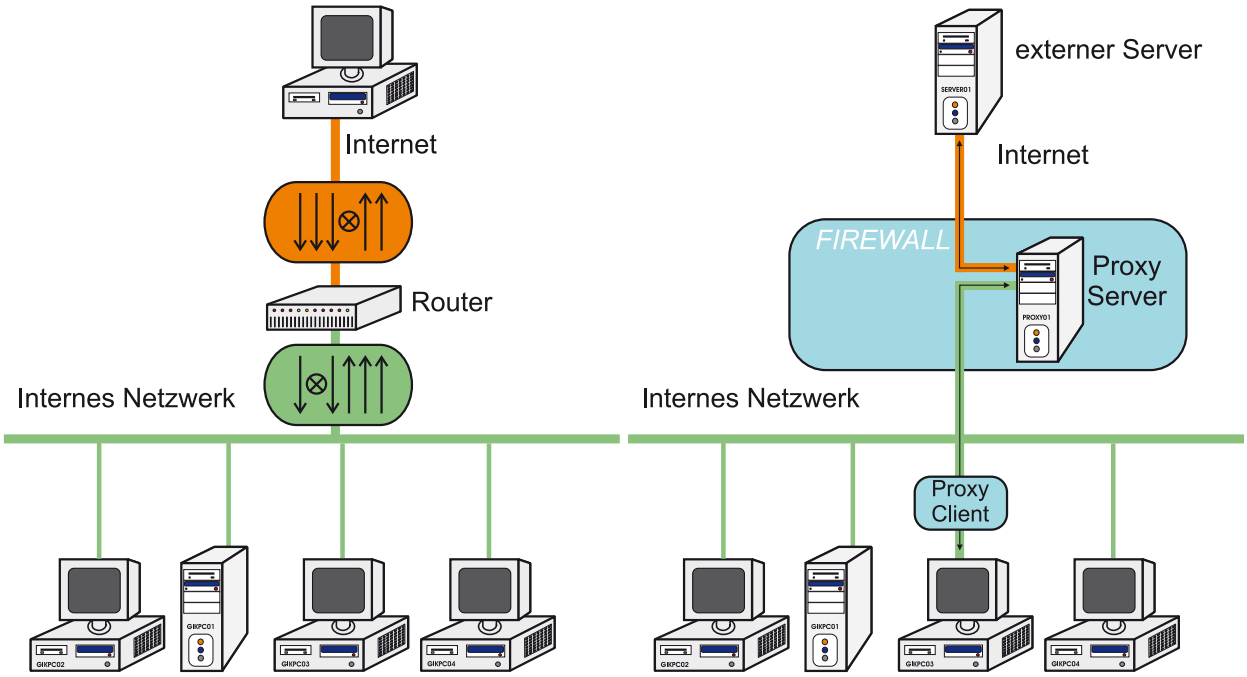
1. Das Servernetz hat in der Regel keinen Zugriff auf das Mitarbeiternetz. Mittels Secure Shell (SSH) ist Zugriff bei Bedarf möglich.
2. Das Servernetz hat nur beschränkten Zugriff auf das externe Netz (Verminderung von Netzangriffen).

Für den Zugriff vom externen Netz auf das Servernetz ist eine beschränkte Anzahl von Protokollen zugelassen (Tab. 3.8).

Protokoll	**Port**
Ident	TCP 113
SSH	TCP 22
SSH alternativer Port für SSH2	TCP 24
WWW	TCP 80

Tabelle 3.8: *Zugelassene Protokolle für den Zugriff vom externen Netz auf das Servernetz. Eine beschränkte Anzahl von Rechnern ist zugelassen (RZ, [46]).*

Mittlerweile werden verschiedenste Architekturen von Firewall-Systemen angeboten, die ein Gleichgewicht zwischen hoher Sicherheit und vernünftiger Performance der Datenübertragung herstellen (Abb. 3.14). Zwei Grundtypen werden vorgestellt, die durch unterschiedliche Kombinationen zu komplexen Firewall-Systemen aufgebaut werden können (Chapman/Zwicky, [13]).



3.14.1: Grundtyp 1 - Zwischen Internet und einem internen Netzwerk wird ein Router zwischengeschaltet.

3.14.2: Grundtyp 2 - Zwischen einem externen Server und einem Proxy-Client wird ein Proxy-Server eingebunden.

Abbildung 3.14: *Aufbau von simplen Firewall-Systemen.*

Beim *Grundtyp 1* werden Regeln für den Router zur Filterung von IP-Paketen definiert. Die IP-Pakete bestehen aus *IP-Adressen (Quelle+Ziel)*, *Portnummern* und *Status-Bits (0/1)*. Aufgrund der eingestellten Regeln

wird vom Router entschieden, ob die Pakete durchgelassen werden oder nicht (das Kennzeichen $\otimes$ entspricht Ablehnung). Es werden dabei zwei Typen von Filtern unterschieden:

1. Adressen-Filterung (nur bestimmte Adressen werden zugelassen oder unbekannte Adressen werden herausgefiltert). Es werden die Quelle, das Ziel, die Richtung (ein/aus) und die Aktion (ja/nein) durch eine Tabelle beschrieben.
2. Port-Filterung (nur bestimmte Ports sind zugelassen).

Beim *Grundtyp 2* wird ein Proxy-Server aufgebaut, der die externen Daten aufnimmt und an den Proxy-Client weiterleitet. Der Proxy-Servers besitzt die Eigenschaft, den direkten Zugriff auf das interne Netz vorzutäuschen. In diesem System besteht die Möglichkeit, einzelne Funktionen von Internetdiensten zu verbieten und die Kommunikation auf bestimmte Rechner einzuschränken.

Das Sicherheitskonzept der Universität Karlsruhe (TH) ist in vier Stufen festgelegt:

- Filter am Uplink zum Provider (BelWü),
- private IP-Adressen für das Universitätsgelände,
- private IP-Adressen mit Institutsgrenzen,
- dedizierte (für eine bestimmte Aufgabe festgelegte) Firewalls auf der Applikationsebene.

Auf Institutsebene werden private IP-Adressen vergeben, welche nur in der Einrichtung Gültigkeit haben. Daher ist ein Verbindungsaufbau von außen nicht möglich. Es werden *Application Gateways* für Web-Proxies und Mail-Server oder *Adressen-Umsetzer* (Port/Network Address Translation) für Anwendungen oder Protokolle eingesetzt, um eine Kommunikation mit dem Internet zu ermöglichen. Dabei analysieren und kontrollieren Application Gateways die übertragenen Pakete auf der Anwendungsebene.

3.2.7.2 Secure Sockets Layer

Secure Sockets Layer (SSL) wurde als Standard vom Internet Engineering Task Force (IETF) definiert und ist zurzeit in der Version 3.0 erhältlich. Seit 1999 wird parallel die Transport Layer Security (TLS) in der Version 1.0 bereitgestellt. Das SSL/TLS-Protokoll wurde als Übertragungsprotokoll zur verschlüsselten Kommunikation entwickelt. Das Protokoll wird zwischen Anwendungsebene (z.B. HTTP) und Transportebene (z.B. TCP/IP) eingefügt und stellt die notwendigen Sicherheitsfunktionen zur Server/Client-Authentisierung und zur Unversehrtheit der Nachrichten zur Verfügung. Im Internet wird SSL/TLS von Browsern unterstützt, die HTTPS-URLs aufrufen können. Dabei werden die empfangenen Nachrichten nicht über Port 80 (definierter Port für HTTP) sondern über Port 443 geleitet. Wenn eine Server/Client-Verbindung über SSL/TLS aufgebaut wird, werden folgende Sicherheitsfunktionen ausgeführt:

- Authentisierung des Servers (optional Client) $\rightarrow$ eindeutiger Nachweis des Ursprungs der digitalen Nachricht. Laut Signaturgesetz (SigG) gilt: *„eine digitale Signatur ist ein mit einem privaten Signierschlüssel erzeugtes Siegel zu digitalen Daten, das mit Hilfe eines zugehörigen öffentlichen Schlüssels, der mit einem Signierschlüssel-Zertifikat versehen ist, den Inhaber des Signierschlüssels und die Unverfälschtheit der Daten erkennen läßt“*.
- Anwendung von Verschlüsselungsalgorithmen.
- Authentifizierung über X.509-Zertifikat.
- Austausch der Nachrichten mittels symmetrischer oder asymmetrischer Schlüsselverfahren.

Zur Authentisierung von verschlüsselten Datenübertragungen per HTTPS werden X.509-Zertifikate mit Schlüssellängen bis zu 2048 Bits für die Web-Server verwendet. Um eine klare Abtrennung zur offenen Datenübertragung per HTTP herbeizuführen, wird eine sichere Kommunikation über HTTPS aufgebaut. Der Sicherheitsmechanismus zur Verschlüsselung der Daten und zur Herstellung der digitalen Signatur wird durch das *Public-Key-Verfahren* beschrieben. Dieses asymmetrische Verfahren arbeitet im Gegensatz zu symmetrischen Verfahren (Private-Key-Verfahren) mit zwei verschiedenen Schlüsseln (Abb. 3.15).

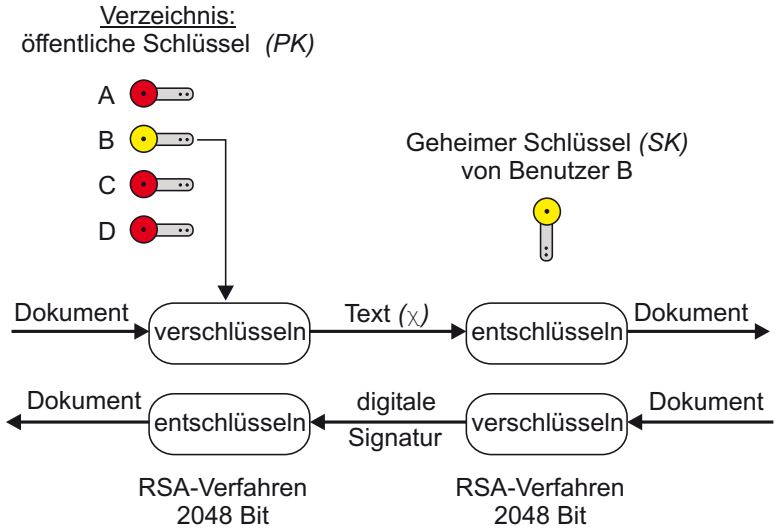


Abbildung 3.15: *Entwicklung des Public-Key-Verfahrens als asymmetrisches Verfahren zur verschlüsselten Datenübertragung (Pohlmann, [65]).*

Das RSA-Verfahren, welches nach seinen Urhebern *Rivest, Shamir und Adleman* benannt wurde, ist als asymmetrisches Verfahren zur verschlüsselten Datenübertragung entwickelt worden (Rivest et al., [70]). Dabei wird der Klartext als Binärzahl m mit $m \in \mathbb{N}$ und $m < n$ dargestellt. Der Klartext wird mit einem öffentlichen Schlüssel verschlüsselt und mit einem geheimen Schlüssel entschlüsselt (und umgekehrt). Es gelten folgende Formeln zum chiffrieren

$$\chi = m^{PK} mod(n) \tag{3.2}$$

und dechiffrieren der Text-Datei:

$$m = \chi^{SK} mod(n) \tag{3.3}$$

m	:	Binärzahl
n	:	Produkt zweier geheimer Primzahlen p und q
PK	:	öffentlicher Schlüssel
SK	:	geheimer Schlüssel
mod	:	Rest aus der Division ganzer Zahlen

X.509-Zertifikate werden bei der autorisierten Zertifikatsstelle (Wurzelinstanz: UNIKA-CA 2005-2008) beantragt und für zwei Jahre ausgestellt. Das Ziel dieser Zertifikate ist nach RFC 3280 (Housley et al., [30]), eine Infrastruktur für öffentliche Schlüssel (PKI) zu entwickeln, welche die Notwendigkeiten der automatischen Identifikation, der Authentifizierung, der Zugangskontrolle und der dazugehörigen Funktionen berücksichtigt. Zur Beantragung eines X.509-Zertifikates werden folgende Felder, welche den Herkunftsort und Urheber eindeutig identifizieren, ausgefüllt und der Zertifikatsstelle (CA) ausgehändigt:

```
Country Name (2 letter code) [ ]        : 'DE'
Organization Name (company) [ ]         : 'Universitaet Karlsruhe'
Organizational Unit Name (Fakultaet) [ ]: 'Fakultaet fuer Bauingenieur-, Geo- und
```

```
                                        Umweltwissenschaften'
Organizational Unit Name (Institut) [ ] : 'Geodaetisches Institut Karlsruhe'
Application or Service Name [ ]         : 'Fachinformationssystem'
Common Name (name) [ ]                  : 'Servername'
Email Address [ ]                       : 'name@gik.uni-karlsruhe.de'
```

Die Zertifikatsinformationen (*servername.crt*) bestehen aus Feldern, die Werte zur Authentisierung und zum Verschlüsselungsverfahren besitzen (Tab. 3.9)

Feld	**Wert**
Version	v3
Signaturalgorithmus	md5RSA
Aussteller	Wurzelinstanz (Universitaet Karlsruhe)
Gültigkeit	von - bis
Antragsteller	name@gik.uni-karlsruhe.de (Geodaetisches Institut)
Öffentlicher Schlüssel	RSA - 1024 Bits
Schlüsselverwendung	digitale Signatur
Schlüsselkennung	Verzeichnisadresse, . . .
. . .	
Fingerabdruckalgorithmus	sha1
Fingerabdruck	87B3 0084 0XD3 ...

Tabelle 3.9: *Das X.509-Zertifikates ist in drei Bereiche eingeteilt: 1. Inhalt des Zertifikates, 2. Verschlüsselungsalgorithmus und 3. digitale Signatur*

3.2.7.3 Sicherheit beim Applikationsserver

Das Sicherheitskonzept beruht auf einem hierarchischen Aufbau der Schichten, die mit der Betriebssystemebene beginnt und mit der Zugriffskontrolle des Web-Servers endet (Abb. 3.16).

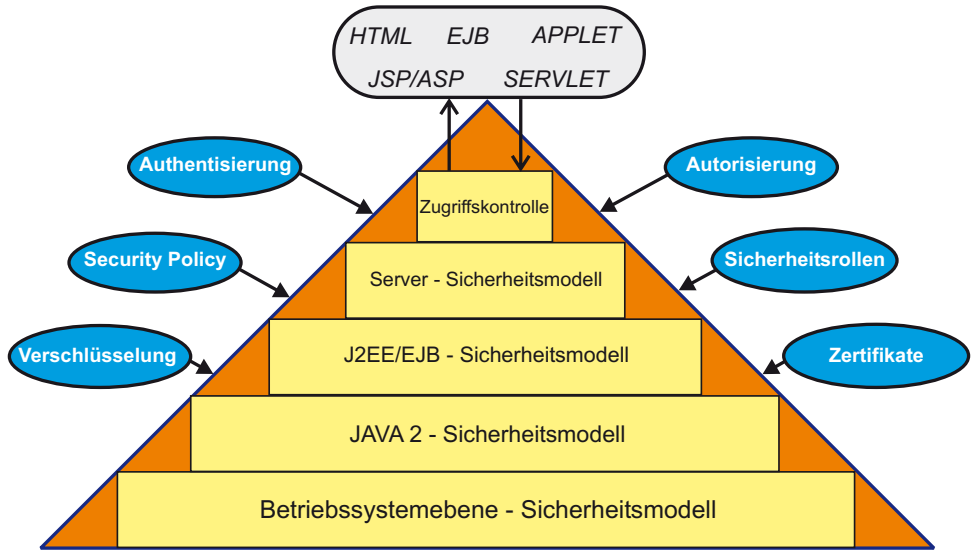


Abbildung 3.16: *Hierarchischer Aufbau der Sicherheitsmodelle zur Beschreibung der Benutzerrechte und der Zugriffsrechte auf Ressourcen und Dienste sowie zur sicheren Datenübertragung. (Backschat/Gardon, [2]).*

Im Zentrum des Sicherheitskonzeptes von Applikationsservern stehen folgende Komponenten:

- *Authentisierung*: Feststellung der Identität eines Benutzers → Verwendung von Zertifikaten, JAAS oder JNDI.
- *Autorisierung*: Zugriff auf den AS festlegen. Die Sicherheitsrichtlinien (security policy) definieren die Berechtigungen eines Benutzers, welche Dienste oder Anwendungen im System angewandt werden dürfen. Die Zugriffskontrolle mittels J2EE wird über programmatische Sicherheit (javax.ejb.EJBContext), deklarative Methoden (method permission) oder über rollenbasierte Modelle im Deployment-Deskriptor (ejb-jar.xml) realisiert:

```
<ejb-jar>
    <security-role>
        <description>
            Diese Rolle darf alle Methoden der Bean ausführen
            und vergibt Schreiblese-Zugriff.
        </description>
        <role-name>Administrator</role-name>
    </security-role>
</ejb-jar>
```

- *Sichere Kommunikation*: Verschlüsselung der Datenübertragung (Zertifikate) → SSL wird unterstützt.

Zur Authentifizierung werden verschiedene Mechanismen eingesetzt. Das Ziel bleibt der Identitätsnachweis des Clients, wenn dieser sich beim Server anmeldet. Für den Client werden verschiedene Methoden zur Verfügung gestellt, deren Gültigkeit während der laufenden Sitzung vom Server beibehalten wird.

Einfache Authentisierung: Angabe des Users und des Passwortes. Es besteht die Möglichkeit, ein Login-Formular per HTML, JSP, PHP, Perl, usw. zu erstellen, wobei ein Fehlerformular bei nicht erfolgreicher Anmeldung angezeigt wird.

```
void start_db_anbindung_actionPerformed(ActionEvent e) {
    String user = pwd.user_input.getText();
    char[] password = pwd.passwd_input.getPassword();
    boolean connected = false;

    DBAccessBean();

    if (connected == true) {
        getRahmen_Startseite();
    }
}
```

Gegenseitige kryptographische Authentisierung: Identifizierung über Zertifikate (z.B. X.509-Zertifikat) unter Verwendung von SSL. Authentifizierung wird somit beim Server und Client gewährleistet (Weimerskirch, [87]).

J2EE-Authentisierung: Die J2EE-Spezifikation garantiert über den Deployment-Deskriptor (ejb-jar.xml, server-web.xml) eine vertrauenswürdige Datenübertragung (z.B. in einer EJB-Anwendung oder Web-Applikation).

```
<web-app>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</web-app>
```

JNDI-Authentisierung: Ein Benutzer eines Applets wird über die JNDI-Verbindung mit den Sicherheitskontext-Anweisungen authentifiziert (PRINCIPAL = Benutzer + Benutzerhaupteigenschaften; CREDENTIALS = Berechtigungsnachweis).

```
Properties p = new Properties();
    p.put(Context.SECURITY_PRINCIPAL, user);
    p.put(Context.SECURITY_CREDENTIALS, password);
Context ctx = new InitialContext(p);
Object ref = ctx.lookup("gikfis");
```

Java Authentication and Authorization Service: JAAS wurde als Sicherheitsmodell entwickelt, um Dienste zur Verfügung zu stellen, die Authentisierung ausführen und Zugangskontrollen (Überprüfung der Benutzerrechte) sicherstellen. JAAS ist somit ein Bestandteil von J2EE. Es werden Module implementiert, die einerseits zur Authentifizierung und andererseits zur benutzerorientierten Autorisierung herangezogen werden. Die Kernkomponenten von JAAS werden in drei Bereiche untergliedert:

1. *allgemeine Klassen*: die Klasse *javax.security.auth.Subject* enthält die Information für ein einzelnes Merkmal, wie z.B. Person oder Ressource. Die Schnittstelle *java.security.Principal* umfasst die Haupteigenschaften einer Person oder eines Dienstes mit öffentlichen und privaten Berechtigungsnachweisen.
2. *Authentisierungsklassen oder -schnittstellen*: Eine Reihe von Klassen und Schnittstellen (Callback, Configuration, LoginContext, ...) werden benötigt, um die Authentizität des Benutzers nachzuweisen.
3. *Autorisations-Schnittstellen*: es werden Sicherheits-Policen verwendet, welche die Berechtigungen der Personen oder Dienste enthalten.

FIS-Authentisierung: Die Container-verwaltete Zugriffskontrolle wird über das Element <resource-ref> gesteuert. Der Zugriff auf die Datenbank *„FISDB“* erfolgt mittels JDBC und die Überprüfung des Identitätsnachweises im Deployment-Deskriptor *„jboss.xml“* über den JNDI-Namen „OracleDS“. Die Datei *OracleDS.xml* enthält die Informationen zur Anmeldung bei der FIS-Datenbank (Programmcode s. Anhang B.2). Bei falscher Angabe des Benutzers oder des Passwortes wird eine Zugriffsverweigerung ausgelöst und eine entsprechende Fehlermeldung ausgegeben.

```
<jboss>
    <enterprise-beans>
        <session>
            <ejb-name>StationszustandBean</ejb-name>
            <jndi-name>StationszustandBean</jndi-name>
            <resource-ref>
                <res-ref-name>jdbc/FISDB</res-ref-name>
                <jndi-name>java:/OracleDS</jndi-name>
            </resource-ref>
            <method-attributes> </method-attributes>
        </session>
        <--! weitere zustandslose Sessionbeans -->
    </enterprise-beans>
</jboss>
```

3.2.7.4 Datenbankzugriff

Der Datenbankzugriff wird über den *Benutzernamen* und das *Passwort* geregelt. Das Benutzerkonto wurde zuvor vom Datenbankadministrator eingerichtet. Diverse Sicherheitsrollen sowie System- und Objektberechtigungen werden dem Benutzer zugewiesen. Die *connect-Rolle* definiert den öffentlichen Zugriff auf die FIS-Objektklassen und Datensichten, die über das Applet dem Client zur Verfügung stehen. Um eine hohe Sicherheit zu garantieren, werden dem Anwender keine Schreibrechte zugewiesen.

BENUTZER	*EXECUTE*-Rechte auf Tabellen und Datensichten. *SELECT*-Rechte auf Tabellen und Datensichten.

Kapitel 4

Enterprise JavaBeans

Die Enterprise JavaBeans Architektur stellt Komponenten für die Entwicklung und Bereitstellung von komponentenbasierten und verteilten Geschäftsanwendungen zur Verfügung. Die Anwendungen sind persistent, skalierbar, übertragbar und multi-userfähig. Außerdem sind die EJB-Module plattform- und implementierungsunabhängig, sofern die Server die J2EE- und EJB-Spezifikation unterstützen. Somit definieren Enterprise JavaBeans ein serverseitiges Komponentenmodell (DeMichiel, [18]).

Entsprechend der EJB-Spezifikation werden Komponenten in Programmgruppen $javax.ejb.*$ definiert, die den Ansprüchen einer mehrschichtigen Client/Server-Architektur gerecht werden. Daher dienen EJBs als Komponenten zur Kommunikation zwischen Prozessen und sind speziell für verteilte Geschäftsobjekte entworfen worden. Weiterhin werden asynchrone Botschaften mittels EJBs erstellt, die einen Nachrichtenaustausch zwischen mehreren Benutzern ermöglichen.

Enterprise JavaBeans werden in drei Formen zur Erstellung von serverseitigen Komponenten angeboten:

1. *Entity Objekt:* Modellieren von Geschäftskonzepten.
2. *Session Objekt:* Bereitstellen von Methoden für diverse Geschäftsanwendungen.
3. *Message Driven Objekt:* Verteilen von Nachrichten-Paketen.

Die Aufgaben werden auf verschiedene Entwicklungsverantwortlichkeiten aufgeteilt, um das Implementierungsverfahren auf folgende Kernbereiche abzugrenzen:

Entwicklungsebene A: Entwicklung von Anwendungen, wie z.B. Applet, HTML, Servlet oder JSP.

Entwicklungsebene B: Aufbau des EJB-Komponentenmodells, Realisierung des Enterprise Bean-Konzeptes, Aufbau der Metadaten mittels des Deployment-Deskriptors (XML-Datei).

Assemblerebene: Integration der beiden Entwicklungsebenen. Erstellen von weiteren Beans, welche die Interaktion der Bean-Objekte sicherstellen. Steuerung der Zugriffsrollen und -rechte sowie Kontrolle der Transaktionsverwaltung. Aufbau der Archive für den Deployer.

Deployment Verwaltungsebene: Konfiguration des EJB-Containers und Integration der EJB-Archive. Der EJB-Container, dessen Architektur in Kapitel 3.2.2 beschrieben wird, verwaltet die EJB-Klassen und Interfaces zur Steuerung der Methodenaufrufe und zur Überwachung der Interaktionen zwischen den Beans. Anpassung herstellerspezifischer Module, welche im AS zu integrieren sind. Kenntnisse über die Architektur des AS sind somit Voraussetzung.

Administrationsebene: Bereitstellung der Netzwerkinfrastruktur, der Sicherheitsrichtlinien und der Schnittstellen zu Datenbankservern oder ERP-Systemen.

Die Ziele der EJB-Komponentenarchitektur lassen sich folgendermaßen zusammenfassen:

- Entwicklung von komponentenbasierten, transaktionsorientierten, mehrschichtigen Anwendungen,
- automatische Transaktions- und Zustandsverwaltung. Integration von Persistenz, Sicherheit und Ressourcenverwaltung im AS,
- Herstellung von plattform- und implementierungsunabhängigen Komponenten,
- Integration neuer Objekte während der Laufzeit des AS (EJB-Komponenten werden als JAR-Dateien im Deployer eingefügt),
- Kompatibilität mit Systemen, die andere serverseitige Komponentenmodelle (z.B. CORBA) einsetzen. Bereitstellung interoperabler Protokolle - WSDL 1.1, Beschreibungssprache für Web-Dienste (Christensen et al., [15]), oder SOAP 1.1, einfaches Zugangsprotokoll für den Austausch von strukturierten Informationen in dezentralisierten und verteilten Umgebungen (Gudgin et al., [27]).

4.1 Entity-Beans

Entity-Beans beschreiben den Zustand und das Verhalten von Objekten der realen Welt. Sie modellieren Objekte, deren Objekteigenschaften festgelegt und deren Informationen in einem Datenbankverwaltungssystem gespeichert werden. Voraussetzung für die Erstellung von Entity-Beans ist die Fähigkeit, den Zustand der Daten persistent zu halten. Persistenz bedeutet im Kontext der EJBs, *dass ein Objekt die Eigenschaft besitzt, seinen Zustand permanent im EJB-Container aufrechtzuerhalten.* Falls die Anwendungen des Clients oder Dienste des Servers unterbrochen oder beendet werden, können die Entity-Beans zu einem späteren Zeitpunkt wiederhergestellt werden. In der Regel spiegelt das Konzept der Entity-Beans das Datenbankmodell auf Seiten des Applikationsservers wieder. Die Entity-Bean ist deshalb eine zentrale Ressource, dass heißt ihre Instanz wird von mehreren Clients gleichzeitig benutzt und die Daten werden allen Benutzern bereitgestellt. Ihre Kennzeichen lassen sich wie folgt zusammenfassen:

- liefert eine Übersicht der Datenobjekte,
- erlaubt getrennten Zugriff auf die Objekte von mehreren Benutzern,
- lange Lebenszeit der Entity-Objekte,
- die Eigenschaften und die Primärschlüssel eines Objektes werden bei einem Absturz des Servers wiederhergestellt,
- unterstützt eine skalierbare Laufzeitumgebung.

Die Angabe, wie eine Entity-Bean vom EJB-Container verwaltet wird, lässt sich aus den folgenden Bestandteilen ableiten:

Remote Interface: diese Schnittstelle definiert die Geschäftsmethoden einer Entity-Bean auf die wiederum außerhalb des EJB-Containers zugegriffen wird.
⇒*public interface <Klassenname_Remote> extends EJBObject{ Geschäftsmethoden( ) }*

Remote Home-Interface: diese Schnittstelle stellt Methoden bereit, welche den Lebenszyklus eines Entity-Objektes definieren, wie z.B. das Anlegen, Löschen und Finden von Bean-Objekten. Anwendungen können auf das entfernte Home-Interface außerhalb des EJB-Containers zugreifen.
⇒*public interface <Klassenname_RemoteHome> extends EJBHome{ Lebenszyklus-Methoden( ) }*

Lokales Interface: diese Schnittstelle definiert die Geschäftsmethoden einer Bean, die von anderen Beans verwendet werden können, welche im selben Container liegen.
⇒*public interface <Klassenname_Local> extends EJBLocalObject{ Geschäftsmethoden( ) };*

Lokales Home-Interface: diese Schnittstelle definiert die Lebenszyklus-Methoden, die von den Beans verwendet werden können, welche im selben Container liegen. Der Vorteil liegt darin, dass die Beans Nachrichten untereinander im Container austauschen können. Damit wird die Performance erheblich gesteigert.
⇒*public interface <Klassenname_LocalHome> extends EJBLocalHome{ Lebenszyklus-Methoden( ) };*

Abstrakte Entity-Bean Klasse: die Geschäfts- und Lebenszyklus-Methoden einer Bean werden in der Bean-Klasse entwickelt und implementiert.
⇒*public abstract class <Klassenname_Bean> implements EntityBean{ Methoden( ) };*

Folgende Klassen und Methoden sind Bestandteil des Lebenzykluses einer Entity-Instanz:

- setEntityContext( ), unsetEntityContext( ),
- ejbHome<Methode>( ), ejbSelect<Suffix>( ), ejbFind<Suffix>( ),
- ejbActivate( ), ejbPassivate( ), ejbCreate( ), ejbPostCreate( ), ejbRemove( ),
- ejbLoad( ), ejbStore( ),
- <Geschäftsmethoden>( ).

4.1.1 Deployment-Deskriptor

Der Deployment-Deskriptor *„ejb-jar.xml“* enthält sämtliche Informationen zum Steuern der Beans während der Laufzeit unter Berücksichtigung der Sicherheit-, Transaktion- und Namensdienste, die vom EJB-Container automatisch angeboten werden. Die Fähigkeit, das Verhalten der Enterprise Beans zur Laufzeit zu verändern, ist deshalb ein wesentliches Merkmal des Deployment-Deskriptors.

```
<!-- Wurzelelement: Metadaten der Jar-Dateien, die Enterprise Beans
     enthalten. -->
<ejb-jar>
    <!-- Festlegung des Bean-Typen -->
    <enterprise-beans>
    <!-- Beschreibung des Entity-Objektes -->
        <entity>
            <!-- eindeutiger, deskriptiver Name -->
            <ejb-name>KlassennameEJB</ejb-name>
            <!-- Bezug: RemoteHome-Interface -->
            <home>KlassennameRemoteHome</home>
            <!-- Bezug: Remote-Interface -->
            <remote>KlassennameRemote</remote>
            <!-- Bezug: LocalHome-Interface -->
            <local-home>KlassennameLocalHome</local-home>
            <!-- Bezug: Local-Interface -->
            <local>KlassennameLocal</local>
            <!-- Bezug: Entity-Bean Klasse -->
            <ejb-class>KlassennameBean</ejb-class>
            <!-- Persistenz automatisch über Container -->
            <persistence-type>Container</persistence-type>
            <!-- Primärschlüssel besitzt den Datentyp Integer -->
            <prim-key-class>java.lang.Integer</prim-key-class>
        </entity>
    </enterprise-beans>
</ejb-jar>
```

4.1.2 Container-verwaltete Persistenz

Die Persistenz beschreibt den Zustand eines Datenobjektes, welcher trotz Unterbrechung oder Abschalten des Servers erhalten bleibt. Ein auf der Festplatte oder in der Datenbank gespeichertes Objekt besitzt die Eigenschaft der Persistenz. Der Applikationsserver greift im EJB-Container auf die Beans zu, deren Datenobjekte i.d.R. in einer Datenbank abgelegt sind. Mittels der Container-verwalteten Persistenz (CMP) wird der Zustand der Bean-Objekte bei Unterbrechung (z.B. einem Stromausfall) des Servers automatisch wiederhergestellt. Die Merkmale einer Bean-Instanz werden auf die Datenbank abgebildet und die Transaktionen, wie z.B. einfügen, aktualisieren und löschen der Daten, automatisch verwaltet. Die Merkmale bestehen aus folgende Elementen:

1. Container-verwaltete Persistenzfelder, welche im Deployment-Deskriptor als Elemente definiert sind:

```
<!-- Eigenschaften von Entity-Beans -->
<persistence-type>Container</persistence-type>
<cmp-version>2.x</cmp-version>
<abstract-schema-name>Tabellenname</abstract-schema-name>
<!-- Container-verwaltete Persistenzfelder -->
<cmp-field><field-name>Spaltenname-1</field-name>
<!-- Primärschlüsselfeld definieren -->
<primkey-field>Primärschlüssel-ID</primkey-field>
```

2. Container-verwaltete Beziehungsfelder, deren Elemente im Deployment-Deskriptor aufgelistet sind:

```
<!-- Eigenschaften für relationale Beziehungen -->
<ejb-relationship-role>
    <!-- Container-verwaltete Beziehungsfelder -->
    <cmr-field>
        <cmr-field-name>Beziehungsfeldname</cmr-field-name>
    </cmr-field>
</ejb-relationship-role>
```

4.1.3 Bean-verwaltete Persistenz

Bean-verwaltete Persistenz (BMP) bedeutet, dass die Persistenz-Logik der Entity-Objekte manuell durch den Bean-Entwickler erzeugt wird. Das heißt, dass der Zugang zu den Datenobjekten über eine Schnittstelle (z.B. über JDBC) erfolgt, welche von den EJB-Diensten unterstützt wird. Die Datenbank-Anweisungen werden explizit in die Entity-Bean Klassen implementiert. Die Anweisungen (z.B. PreparedStatement, etc.) werden in den ejbCreate( )-, ejbLoad( )-, ejbStore( )-, ejbFind( )- , ejbRemove( )-Methoden oder in den Geschäftsmethoden eingefügt. Die Authentifizierung kann automatisch über den Container oder direkt über die Applikation erfolgen (Element: <res-auth>). Der Deployment-Deskriptor enthält folgende Elemente zur Festlegung der BMP:

```
<persistence-type>Bean</persistence-type>
<resource-ref>
    <description>Datenquelle: Zugang mittels JDBC</description>
    <res-ref-name>jdbc/Servername-DB</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container/Application</res-auth>
</resource-ref>
```

4.1.4 Laufzeitumgebung eines EJB-Containers

Die Laufzeitumgebung besteht aus verschiedenen Diensten: Sicherheitsrichtlinie, Transaktionsverwaltung, Container-verwaltete Persistenz und Ressourcen-Manager. Diese Dienste werden durch den Deployment-Deskriptor festgelegt (Abb. 4.1). Danach übernimmt der Container automatisch das Management aller Laufzeitaspekte. Somit ist der Container für das Erzeugen, Freigeben oder Wiederverwenden von Bean-Instanzen

verantwortlich und steuert den Lebenszyklus der einzelnen deployten Enterprise Bean-Komponenten. Der Container ordnet jeder Bean automatisch eine Instanz zu. Damit wird ausreichender Speicher für die EJB-Objekte beim Methodenaufruf eines Benutzers instanziiert. Falls die Instanz nicht mehr benötigt wird, kann diese für andere Anwendungen wieder freigegeben werden. Mittels Pooling wird gewährleistet, dass einerseits ausreichender Speicher zur Verfügung steht und andererseits kein überflüssiger Speicher durch unbelegte Bean-Instanzen bestehen bleibt. Der Vorteil liegt offensichtlich darin, dass weniger Instanzen als vorhandene EJB-Objekte benötigt werden → effiziente Ressourcenverwaltung.

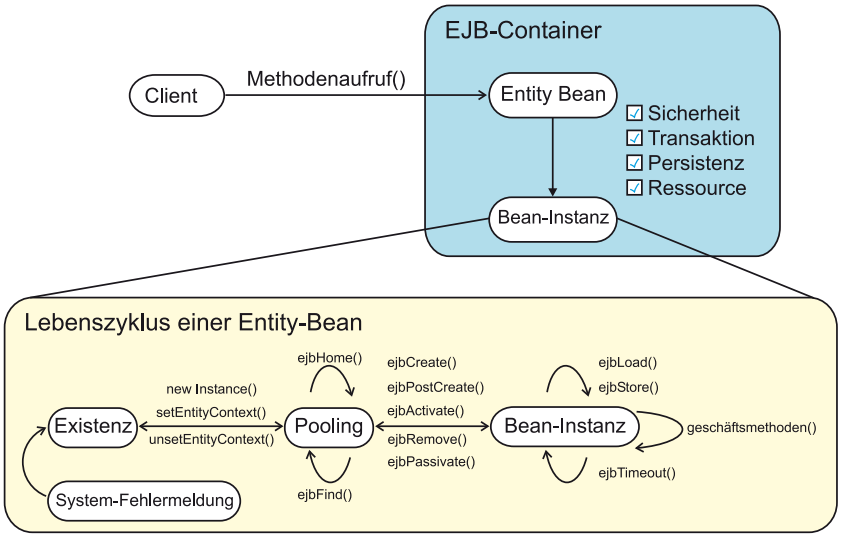


Abbildung 4.1: *Darstellung eines EJB-Containers mit impliziten Diensten (Container-verwaltet) und der Bean-Instanz sowie deren Lebenszyklus.*

Der Container kommuniziert mit der Bean-Instanz über folgende Mechanismen:

1. Callback-Methoden: Bei der Implementierung einer Enterprise Bean-Klasse (z.B. public abstract class <Name_Bean> implements javax.ejb.EntityBean) wird ein Enterprise Bean-Interface erstellt. Diese Schnittstelle wird als Superinterface der Schnittstellen *EntityBean*, *SessionBean* und *MessageDrivenBean* bezeichnet. Damit sind diese Schnittstellen aus dem javax.ejb.EnterpriseBean-Interface abgeleitet (z.B. public interface EntityBean extends EnterpriseBean). Die Callback-Methoden werden von den drei Schnittstellen zur Verfügung gestellt. Die Methoden setEntityContext( ), ejbCreate( ), ejbActivate( ), ejbRemove( ), usw. werden vom Container verwendet, um sämtliche Ereignisse betreffend des Lebenszykluses einer Bean-Instanz zu steuern (Sun Microsystems, [80]).

2. EJBContext: Das EJBContext-Interface ist aus den obengenannten EJB-Schnittstellen abgeleitet. Das EJBContext-Objekt stellt die Schnittstelle zwischen Bean-Instanz und Container dar. Der Container ruft nach Erstellen der Instanz das Objekt auf, welches eine Referenz zur Bean-Instanz besitzt. Durch Verwendung der Kontext-Objekte, z.B. getEJBHome( ), werden der Instanz Informationen über die Laufzeitumgebung übermittelt. Das Kontext-Objekt existiert solange die Bean-Instanz nicht gelöscht wird.

3. JNDI: Das Akronym JNDI steht für ein Namens- und Verzeichnis-Interface, das speziell für JAVA Anwendungen entwickelt wurde. Somit wurde eine Architektur realisiert, die einerseits einen Zugang zu verschiedenen Namens- und Verzeichnisdiensten (JNDI^{API}) und andererseits zu beliebigen Dienstleistungsanbietern (JNDI^{SPI}) bereitstellt (Abb. 4.2).

Abbildung 4.2: *Darstellung der JNDI-Architektur, die verschiedene Schnittstellen anbietet.*

Allgemein unterstützt ein Verzeichnisdienst den Zugang zu Informationen über einen Benutzer und die benötigten Ressourcen in einer Netzwerkumgebung. Derartige Informationen zur Identifizierung und Verwaltung von sogenannten *„Verzeichnis-Objekten“* werden in der Regel über Namensdienste gesteuert. Eine Verzeichnis-Objekt besteht dabei aus Attributen, einem Bezeichner und einer Reihe von Werten. Der Kern eines Namensdienstes besteht aus zwei Komponenten: erstens eine eindeutige *Namenskonvention* und zweitens dem *Kontext*. Der Kontext stellt in diesem Zusammenhang ein Objekt dar, welches einen Satz von Verbindungen bereitstellt und einen eindeutigen Bezug zu einem Objekt durch einen atomaren Namen herstellt. Dabei werden Methoden zum Binden, Lösen, Auflisten und Nachschlagen von eindeutigen Namen als Bestandteil der Schnittstelle *Context* implementiert (Sun Microsystems, [78]).

```
public interface Context
{
    public Object lookup(Name name) throws NamingException;
    public void bind(Name name, Object object) throws NamingException;
    public NamingEnumeration listBindings(Name name) throws NamingException;
}
```

Ein EJB-Server unterstützt die Eigenschaften und Funktionen von JNDI, welche in der J2EE-Spezifikation eingebunden sind. Es wird eine Umgebung über den Environment Naming Context (ENC) aufgebaut, welche einen Informationsaustausch zwischen Container und EJB ermöglicht. Damit wird ein JNDI-Namensraum realisiert, der Container-, Bean-Kontexte und Subkontexte enthält. Unter Verwendung der Lookup-Methode greift das Bean auf den ENC zu und liest die darin enthaltenen EJB-Objekte aus. Dabei übergibt der Container der Enterprise Bean den Kontext, um die Informationen auszulesen. Art und Umfang der Informationen werden wiederum über den Deployment Deskriptor festgelegt (Kap. 4.1.1).

```
\** Aufbau der Netzwerk-Verbindung zum EJB-Server *\
    Context jndiContext = getInitialContext();
\** Erstellen eines Referenz-Objektes zum entfernten Home-Interface *\
    Object ref = jndiContext.lookup("KlassennameRemoteHome");
```

4.2 Session-Beans

Session-Beans stellen Geschäftsmethoden zur Verfügung, um Java-Anwendungen auszuführen. Dieser Typ von Enterprise JavaBean unterteilt sich in die Gruppe der *Stateless Session* oder *Stateful Session Bean*:

- **Stateless Session-Bean:** Die Bean ist zustandslos. Dies bedeutet, dass nur eine Interaktion zwischen Client und Bean möglich ist. Der Server benötigt einen Zustand, der solange bestehen bleibt wie der Methodenaufruf wirksam ist.
- **Stateful Session-Bean:** Die Bean besitzt einen Zustand, der aufrechterhalten bleibt, um eine Kommunikation mit dem Client zu ermöglichen. Dabei interagiert der Client mit dem EJB-Objekt über mehrere Methodenaufrufe. Deshalb wird diese Art von Bean-Objekten bei Geschäftsprozessen implementiert, die mehrere Interaktionsschritte benötigen.

Session-Beans werden deshalb als kurzlebige Objekte bezeichnet und besitzen nicht die Eigenschaft der Persistenz.

4.2.1 Kommunikation zwischen Client und Stateless Session-Bean

Zustandslose Session-Beans besitzen einen kurzen Lebenszyklus. Mittels der Klasse *Class.newInstance( )* wird beim Hochfahren des Servers eine Bean instanziiert. Danach wird die Methode *setSessionContext(SessionContext ctx)* aufgerufen, um die Instanz mit dem EJB-Kontext zu referenzieren. Durch die Methode *ejbCreate( )*

wird eine Instanz erzeugt und dem *Instanzen-Pool* zugewiesen. Schließlich kann die *Geschäftsmethode( )* einmal im Lebenszyklus einer Session-Bean ausgeführt werden. Die Bean-Instanz bleibt solange im Pool, bis die *ejbRemove( )*-Methode aufgerufen oder der Server heruntergefahren wird. Im Deployer des Applikationsservers wird die Jar-Datei *„gikfisbean.jar“* eingelesen, die sämtliche Bean-Objekte enthält (Tab. 4.1).

Der clientseitige Zugriff auf die Informationen, die vom AS geliefert werden, erfordert eine Reihe von Schnittstellen und Methoden, die im Applet implementiert werden müssen. Mittels der *import*-Anweisung werden die Remote- und Remote Home-Interfaces im Programmkopf *„Fachinformationssystem - SFB 461“* eingefügt. Zusätzlich werden die *InitialContext*- und *PortableRemoteObject*-Klassen benötigt:

```
import interfaces.<NAME>BeanRemoteHome;
import interfaces.<NAME>BeanRemote;
. . .
import javax.naming.InitialContext;
import javax.rmi.PortableRemoteObject;
```

Namensdienst:	
[org.jboss.naming.NamingService]	Context.PROVIDER_URL in server jndi.properties, url = localhost
[org.jboss.naming.NamingService]	Listening on port 1099
[org.jboss.naming.NamingService]	Started jboss:service = Naming
[org.jboss.naming.JNDIView]	Started jboss:service = JNDIView
Deployer:	
[org.jboss.deployment.MainDeployer]	Starting deployment of package: /serverpath/deploy /gikfisbean.jar
[org.jboss.ejb.EjbModule]	Deploying Stateless<Name>Bean
. . .	
Instanzenpool:	
[org.jboss.ejb.plugins.StatelessSessionInstancePool]	Started jboss.j2ee: jndiName = Stateless<Name>Bean, plugin = pool, service = EJB
. . .	
Container:	
[org.jboss.ejb.StatelessSessionContainer]	Started jboss.j2ee: jndiName = Stateless<Name>Bean, service = EJB
. . .	
Modul gikfisbean:	
[org.jboss.ejb.EjbModule]	Started jboss.j2ee: module = gikfisbean.jar, service = EjbModule
[org.jboss.ejb.EJBDeployer]	Deployed: /serverpath/deploy/gikfisbean.jar
[org.jboss.deployment.MainDeployer]	Deployed package: /serverpath/deploy/gikfisbean.jar

Tabelle 4.1: *Start des JBoss-AS → der JNDI-Namensdienst, der JBoss-Deployer, der Instanzen-Pool, der EJB-Container und das EJB-Modul werden geladen.*

Um Anweisungen senden bzw. Daten empfangen zu können, wird eine Verbindung zum AS per JNDI hergestellt. Unter Verwendung der *try/catch*-Anweisung wird eine Methode implementiert, die eine Referenz zum Container aufbaut und eine Datenbankanweisung ausführt. Die Exception in der catch-Anweisung wird ausgelöst, wenn die try-Anweisung nicht erfolgreich ausgeführt wurde. Die Klasse *Properties* legt eine leere Eigenschaftsliste an. Mittels der *setProperty*-Methode wird eine *Factory* für die Namens-Kontexte initialisiert. Dabei wird die Factory-Klasse als Hilfsmittel zum Erzeugen komplexer Objekte (Spezifizierung der JNDI-Eigenschaften) verwendet, die über eine Netzwerkverbindung geladen werden. Dem URL des Providers wird der Hostname des AS zugeordnet (default: localhost). Zur Kommunikation zwischen Benutzer und JBoss-

Server wird die Schnittstelle *org.jnp.interfaces* über einen RMI-Port aufgebaut. Als Standard-Port für Namensdienste wird 1099 festgelegt. Der Aufbau der Datei *„jndi.properties“* sieht folgendermaßen aus:

1. java.naming.factory.initial ↔ org.jnp.interfaces.NamingContextFactory.
2. java.naming.provider.url ↔ Hostname.
3. java.naming.factory.url.pkgs ↔ org.jboss.naming:org.jnp.interfaces.

Es wird ein Remote Objekt unter Verwendung der *PortableRemoteObject*-Klasse initialisiert. Mittels der *lookup*-Methode wird der Name der Session-Bean in der Namensliste nachgeschlagen. Nach erfolgreicher Suche wird ein Zusammenhang zum Home-Interface hergestellt und eine Referenz mit dem Methodenaufruf *home.create( )* erzeugt. Danach wird die Geschäftsmethode aufgerufen und die Daten einem String-Feld übergeben (Burke/Labourey, [12]).

```
try
{
    Properties jndiProps = new Properties();

    jndiProps.setProperty("java.naming.factory.initial",
        "org.jnp.interfaces.NamingContextFactory");
    jndiProps.setProperty("java.naming.provider.url", par1.myServer);
    jndiProps.setProperty("java.naming.factory.url.pkgs",
        "org.jboss.naming:org.jnp.interfaces");

    <Klassenname>BeanRemoteHome home = (<Klassenname>BeanRemoteHome)
        PortableRemoteObject.narrow(
            new InitialContext(jndiProps).lookup("<Klassenname>Bean"),
                <Klassenname>BeanRemoteHome.class);
    <Klassenname>BeanRemote remote = home.create();

    String[] stringname = remote.<Methodenname>(String DB-Anweisung);
}
catch (Exception ex)
{
    ...
}
```

4.2.2 Kommunikation zwischen Stateless Session-Bean und Datenbank

Die Verbindung zwischen dem Applikationsserver und dem DB-Server wird über Klassen realisiert, welche Session-Beans implementieren sowie RMI-, JDBC- und JNDI-APIs zur Datenbankverbindung einsetzen. Aufgrund der Festlegung der Attribute und der Eigenschaften der Session-Beans im Deployment-Deskriptor, wird eine eindeutige Referenz zwischen AS und DB hergestellt, um Anweisungen auszuführen und die Ergebnisse an den Client zurückzusenden. Um die in Kapitel 3.2.7.3 beschriebenen Sicherheitsrichtlinien einzuhalten, wird die zustandslose Session-Bean *„DBAccessBean“* implementiert, die einen verschlüsselten User/Passwort-Eintrag überprüft, bevor Datenbankanfragen gestellt werden können. Die DBAccessBean (vollständiger Programmcode im Anhang C.1) enthält folgende Methoden:

- Callback-Methoden: ejbCreate( ), ejbActivate( ), ejbPassivate( ) und ejbRemove( ),
- Callback-Methode: setContext(Sessioncontext context),
- Geschäftsmethoden: dbaccess(char[ ] passwd, String user), getConnection(char[ ] password, String user) und getoraclemessage( ).

Der *getConnection*-Methodenaufruf ermöglicht eine Datenbankverbindung und legt anwendungsbasierte Zugriffe auf die DB im Deployment-Deskriptor fest (Anh. C.2). Dabei werden explizit Benutzer und Passwort abgefragt und der JDBC-Treiber der Datenbank geladen.

Mittels der *dbaccess*-Methodenanweisung *public boolean dbaccess(char[ ] password, String user) throws java.rmi.RemoteException* wird der Benutzername und das verschlüsselte Passwort der Main-Klasse übergeben. Es erfolgt ein containerbasierter Zugriff auf die Datenbank, welcher im Deployment-Deskriptor festgelegt wurde ⇒ Anhang C.2). Der automatische Zugang wird für alle nachfolgenden Anweisungen definiert. Die Main-Klasse des Fachinformationssystems gibt erst bei erfolgreicher Anmeldung die Java-Menüleiste *„Dreidimensionalen Plattenkinematik in den Ostkarpaten“* für den Benutzer frei (Abb. 4.3).

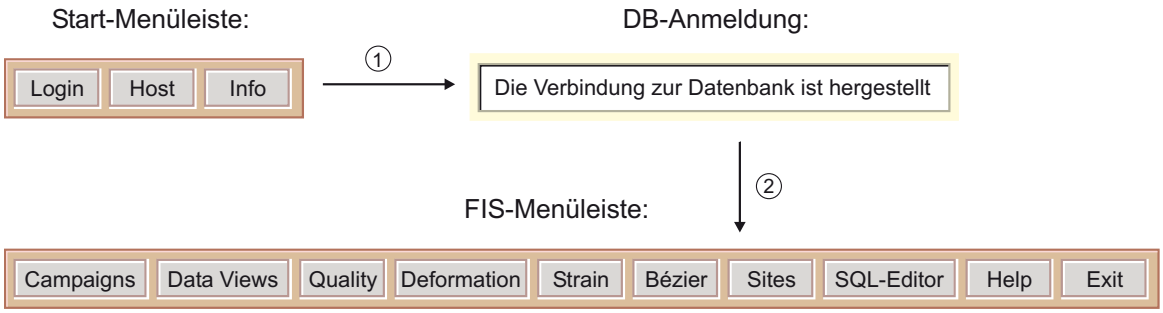


Abbildung 4.3: *Änderung der Menüleiste bei erfolgreicher DB-Anmeldung.*

Der Deployment-Deskriptor legt sämtliche Eigenschaften der Session-Bean fest:

1. <*enterprise-beans*>Attribute der Session-Bean</*enterprise-beans*>: Festlegung des EJB-Namens; Zuordnung der Remote-, Remote Home-Schnittstelle und der Session-Bean-Klasse; Spezifizierung des Session-Typs (stateless or stateful) und des Transaktions-Typs (Container-verwaltete versus Bean-verwaltete Transaktion ⇔ Kapitel 4.4).

   - <*res-ref-name*>jdbc/FISDB</*res-ref-name*>: Festlegung eines eindeutigen Referenznamens, um einen Bezug zu einer Ressource (Datenbank) herzustellen. Der Name wird mit dem *Ressource Factory*-Objekt im JNDI-Namensraum verbunden.
   - <*jndi-name*>java:< Pfad >/< Dateiname ></*jndi-name*>: Festlegung des JNDI-Namens (java:/OracleDS), um auf die Ressource zuzugreifen (Anh. B.2).

2. <*assembly-descriptor*>weitere Assembler Eigenschaften </*assembly-descriptor*>: der Assembler berücksichtigt die Sicherheitsrollen, Methoden-Zugriffe und die Transaktionsverwaltung ⇒ d.h. voller Zugriff auf die Session-Beans für jeden Anwender. Die Passwortabfrage erfolgt über die *DBAccessBean* → <*trans-attribute*>required</*trans-attribute*>, um sich bei der Datenbank anzumelden. Deshalb ist der Assembler für die Dienste *Zugriffskontrolle* und *Transaktion* zuständig, die vom Container zur Verfügung gestellt werden.

4.3 Nachrichten-gesteuerte Beans

Message Driven Beans bauen eine nachrichtenbasierte Kommunikation auf, indem die Nachrichten asynchron versendet werden. Im Gegensatz zu Entity-Beans bzw. Session-Beans wird ein Nachrichtensystem eingesetzt, welches sich vom herkömmlichen Verbindungssystem (RMI-IIOP) durch folgende Punkte unterscheidet (Sun Microsystems, [79]):

- **Asynchroner Aufruf:** der Zugriff auf ein Nachrichten-gesteuertes Objekt wird während der Verarbeitung eines Methoden-Aufrufs nicht blockiert, wenn mehrere Clients auf das Objekt zugreifen.

- **Garantierte Zustellung:** die Nachrichten werden dem Empfänger zugestellt, auch wenn dieser nicht erreichbar ist. Eine Unterbrechung durch eine Netzstörung oder einem Serverausfall führt nicht automatisch zum Ausfall der Nachrichtenzustellung.

- **Message Oriented Middleware (MOM):** die Architektur unterstützt Funktionalitäten, die Nachrichten von verschiedenen Servern zu mehreren Clients verteilen. Dazu wird in der aktuellen J2EE-Spezifikation *1.3* das Java Message Service *API* als Standard-Schnittstelle für Messaging-Systeme verwendet.

Beim Aufbau des Verbindungssystems unterstützt der JAVA-Nachrichtendienst (JMS) zwei Kommunikationsmodelle: das *Point to Point*- und das *Publish/Subscribe*-Modell (Abb. 4.4):

1. **Point to Point-Modell:** Dieses Modell wird eingesetzt, um eine Nachricht direkt an einen bestimmten Empfänger zu senden. Dabei verwaltet der *Message-Broker* eine Warteschlange (Queue), in der die Nachrichten solange gehalten werden, bis sie nach und nach zugestellt werden.

2. **Publish/Subscribe-Modell:** Das Konzept wurde entwickelt, um eine Nachricht an mehrere Empfänger zu versenden, die unter einem bestimmten Fachgebiet (Topic) eingetragen sind. Dabei verwaltet der *Message-Broker* die Topics.

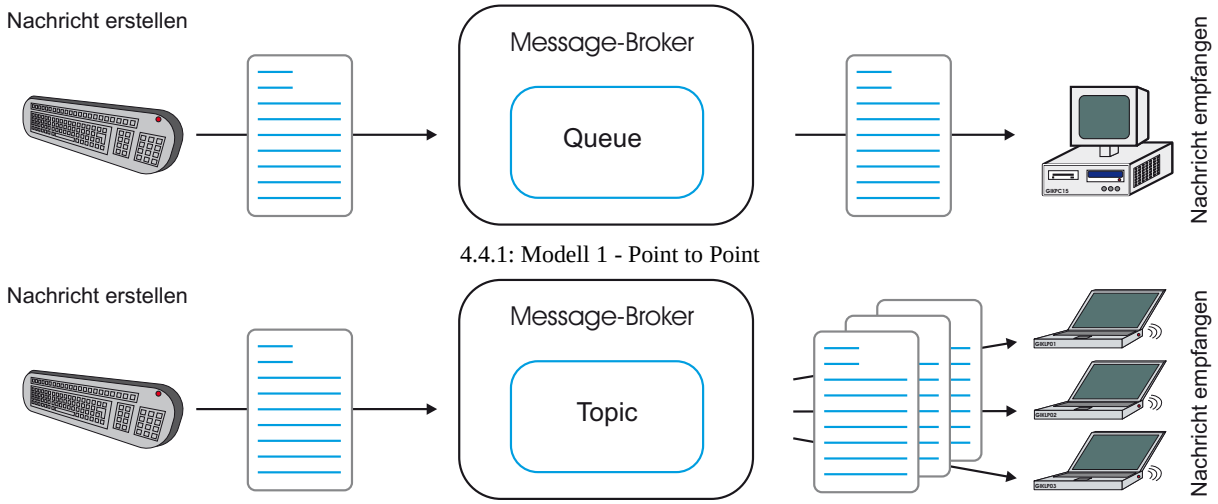


Abbildung 4.4: *Gegenüberstellung der beiden Kommunikationsmodelle des Java Message Services.*

4.4 Transaktionsverwaltung

„Eine Transaktion ist die Anzahl einzelner Arbeitsschritte, die den physikalischen und logischen Zustand einer Anwendung beeinflussen“ (Gray/Reuter, [25]). Bezüglich der Ausführung von Transaktionen in einem DBMS werden folgende vier grundlegende Eigenschaften berücksichtigt (Lang/Lockemann, [50]):

- **Atomarität:** die Transaktion läuft komplett durch oder die Datenbank wird bei Abbruch der Transaktion automatisch in den vorherigen Zustand gesetzt.

- **Konsistenz:** während einer Transaktion werden die Integritätsbedingungen eingehalten, ansonsten wird die Transaktion abgebrochen. Somit werden die Daten in einem konsistenten Zustand gehalten. Die Einhaltung der Regeln, die durch das logische Datenbankmodell aufgestellt werden, führt zur Folgerichtigkeit der Datenspeicherung.

- **Isolation:** Transaktionen können ohne gegenseitige Störung oder Verletzung der Konsistenz nebeneinander ausgeführt werden. Ansonsten stellt der Ressourcen-Manager Sperrmechanismen zur Verfügung, die weitere Transaktionen blockieren, welche auf dieselben Daten zugreifen. Somit wird eine Daten-Transaktion vor dem Zugriff anderer Transaktionsprozesse geschützt.
- **Dauerhaftigkeit:** wird eine Transaktion mit der *commit*-Anweisung abgeschlossen, werden alle Daten in der Datenbank dauerhaft gespeichert.

Der Container-gesteuerte Transaktionsdienst des EJB-Servers garantiert die Einhaltung der *ACID*-Eigenschaften. Die Attribut-Werte werden im Deployment-Deskriptor gleichzeitig mit den Bean-Methoden festgelegt.

⇒<*transaction-type*> Container </*transaction-type*>
⇒<*trans-attribute*> Wert </*trans-attribute*>
⇒<*method-name*> Methodenname </*method-name*>

Die Attributwerte steuern den Transaktionsverlauf einer Datenbank. Zusätzlich stellen sie Regeln dar, wie Methoden sich im Transaktionsprozess zu verhalten haben (Tab. 4.2).

Required	Eine Methode (z.B. getConnection() → DBAccessBean) muss immer im Kontext der Transaktion (Abfrage des Benutzers und des Passwortes) ablaufen. Das heißt, es wird eine neue Transaktion gestartet, wenn keine andere offen ist, ansonsten wird die offene Transaktion verwendet. Die Transaktion wird nach Ausführung der Methode beendet.
RequiresNew	Eine Methode muss immer im Kontext einer neuen Transaktion ablaufen. Falls eine Transaktion vom aufrufenden Client offen sein sollte wird diese suspendiert und eine neue gestartet.
Mandatory	Eine Transaktion muss vorher gestartet sein, bevor eine Methode mit diesem Attribut ausgeführt wird. Ansonsten folgt eine *TransactionRequired*-Exception.
Never	Eine Methode darf nicht während einer offenen Transaktion ausgeführt werden. Ansonsten folgt eine *RemoteException* oder *EJBException*.
NotSupported	Beim Aufruf der Bean-Methode, wird eine laufende Transaktion während der Ausführung der Methode ausgesetzt und danach an der unterbrochenen Stelle weitergeführt → der Methodenaufruf läuft ohne Transaktionskontext und gehört daher nicht zum Gültigkeitsbereich einer Transaktion.
Supports	Die Methode kann während einer offenen Transaktion oder ohne vorhandene Transaktion ausgeführt werden. Sie gehört zum Gültigkeitsbereich einer Transaktion, sofern eine Transaktion gestartet wurde. Ansonsten gilt der Status „NotSupported“.

Tabelle 4.2: *Beschreibung der Attribut-Werte von Container-gesteuerten Transaktionsdiensten, die im Deployment-Deskriptor des EJB-Servers definiert werden.*

4.5 Das Container-System

Die nächsten beiden Diagramme (Abb. 4.5 und Abb. 4.6) stellen Container-Systeme dar, welche den Funktionsumfang zum Erzeugen und Löschen der EJB-Komponenten in einer Container-Systemumgebung auflisten und das Zusammenspiel der Callback- und Geschäfts-Methoden aufzeigen (Monson-Haefel, [60], S. 587-603).

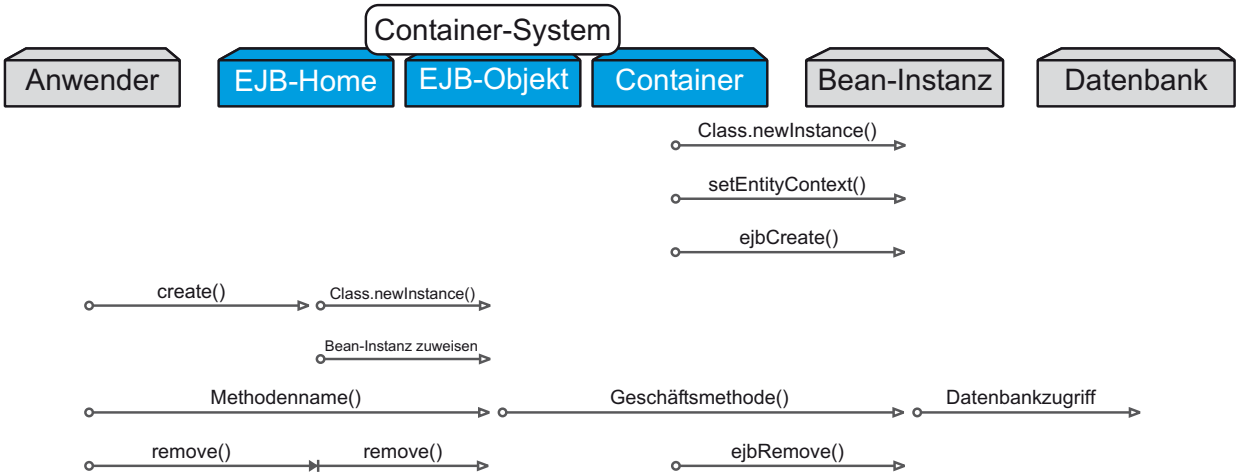


Abbildung 4.5: *Ein Container-System-Diagramm für Stateless Session-Beans.*

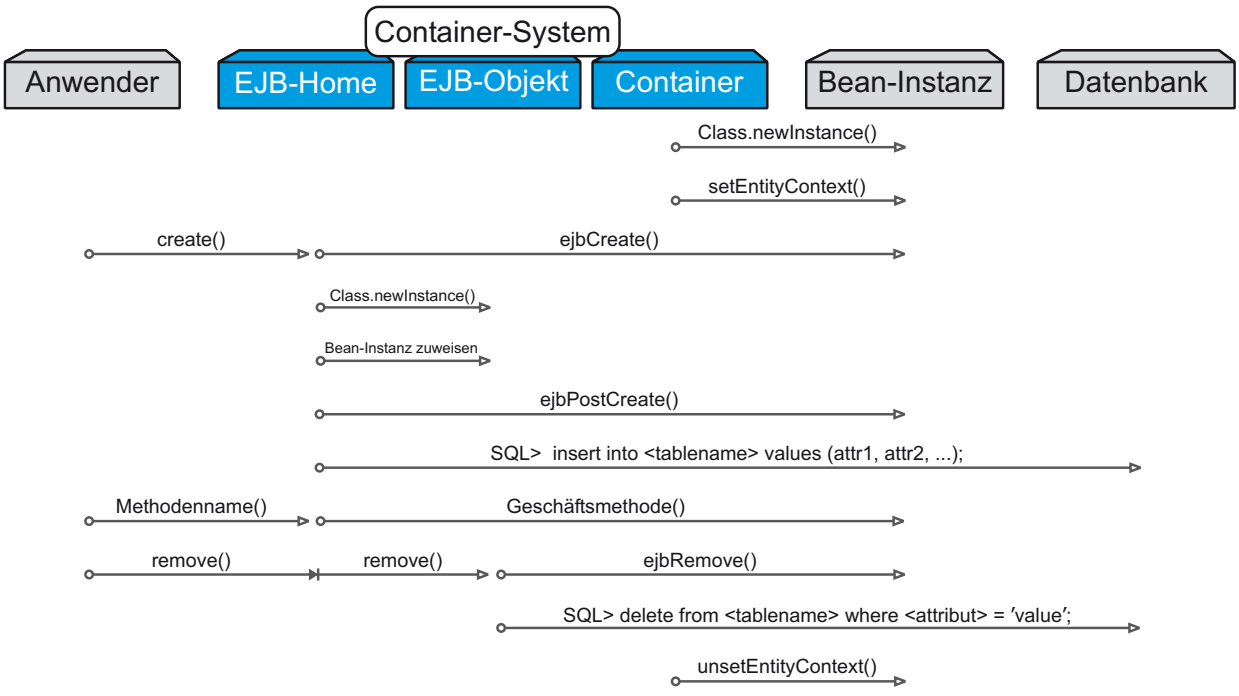


Abbildung 4.6: *Ein Container-System-Diagramm für Entity-Beans mit CMP.*

Kapitel 5

Datenübertragung im XML-Format

Extensible Markup Language (XML) ist ein Werkzeug zur Erzeugung, Gestaltung und Verwendung von Auszeichnungssprachen. XML wurde in den 90er Jahren entwickelt und gilt als Tool zur Realisierung eines unabhängigen standardisierten Datenaustausch-Formates. Nachdem sich HTML als Auszeichnungssprache im Internet durchgesetzt hatte, wurde eine flexible und einheitliche Grundlage zur Schaffung von Auszeichnungssprachen gebildet. Damit stehen Regeln zur Verfügung, die es erlauben einen eigenständigen Dokumenttyp zu erzeugen oder HTML korrekt zu definieren. Eine Forderung ist dann erfüllt, wenn HTML und XML sich zu XHTML vereinigen. Ein weiterer Schwerpunkt der XML-Spezifikation ist es zum Beispiel, eine Datenübertragung zu XML-Datenbanken herzustellen. Somit bildet XML die Plattform zum Austausch von Daten im Internet. Folgende Eigenschaften besitzt XML (Ray, [69]):

- Entwicklung von Stylesheets zur Darstellung und Transformation von Daten,
- Methoden zur Verknüpfung von Ressourcen,
- Werkzeuge zum Manipulieren und Abfragen von Daten,
- Festlegung von Regeln zur Strukturierung und Überprüfung der Daten in einer Entwicklungsumgebung.

5.1 Der Aufbau von XML

XML definiert Regeln und Elemente zum Aufbau eines XML-Dokumentes.

Regel 1: der Inhalt eines XML-Dokuments besteht aus strukturierten Elementen, die hierarchisch geschachtelt sind. Dabei werden die Elemente in Form von Auszeichnungssymbolen (<*Tag*>...</*Tag*>) dargestellt. Die Reihenfolge der Ebenen (Anfang + Ende) muss eingehalten werden.

```
<Wurzel>
    <1-Knoten>
        <2-Knoten>
            . . .
        </2-Knoten>
    </1-Knoten>
</Wurzel>
```

Regel 2: zu einem Anfangs-XML-Fragment gehört immer ein End-XML-Fragment.

```
<XML-Fragment>
 Dateninhalt
</XML-Fragment>
```

Regel 3: der Dateninhalt, welcher im Dokument abzubilden ist, wird zwischen den Tags eingefügt. Die Informationen werden bezüglich ihrer Merkmale den Elementen zugeordnet.

```
<Station>
    <Name>Bukarest</Name>
    <ID>BUCU</ID>
    <Typ>Permanentstation</Typ>
    . . .
</Station>
```

Regel 4: die Elemente besitzen zusätzlich Attribute, die mit Hilfe von bestimmten Such-Methoden oder Abfrage-Funktionen Beziehungen untereinander herstellen können. Das Attribut besteht aus einem kleingeschriebenen Namen (source) und einem in Anführungszeichen gesetzten Wert (Deformationsanalyse).

```
<web-page source="Deformationsanalyse">
    . . .
<web-page>
```

Regel 5: Tags, die keine Inhalte einschließen sondern ausschließlich aus Attributen bestehen, ersetzen das End-XML-Fragment mit einem Schrägstrich am Ende.

```
<Geschwindigkeit signifikant="true" />
```

Regel 6: XML besitzt Literale, welche den Inhalt der internen Eigenschaften, die Attributwerte und externe Bezeichner spezifizieren (W3C, [86]).

```
Entität-Wert   : [ ^ % & " ' ] Referenz *
Attribut-Wert  : [ ^ < & " ' > ] Referenz *
System-Literal : [ ^ " ' ] *
PubidChar      : #x20 | #xD | #xA | [a-zA-Z0-9] | [-'()+,./:=?;!*#@$_%]
```

Regel 7: die Kommentare zur Gliederung des XML-Dokumentes sind in einem speziellen Element enthalten und werden beim Auswerten der Daten übergangen.

```
<!-- Kommentar: Sonderforschungsbereich 461 - Teilprojekt B1 -->
```

Regel 8: eine XML-Datei ist wohlgeformt, wenn sie die oben genannten Regeln einhält.

Regel 9: eine XML-Datei enthält eine Kopfdefinition. Durch die *encoding*-Anweisung wird ein 8-Bit-Zeichensatz implementiert, der mit $X = 1$ den westeuropäischen Zeichensatz nach der ISO-Norm (entspricht *Latin-1*) festlegt.

```
<?xml version="1.0" encoding="ISO-8859-X"?>
```

Regel 10: eine XML-Datei kann Dokumenttyp-Definitionen, Deklarationen von speziellen Textteilen, Textkodierungen und Anweisungen enthalten. Zum Beispiel sind die Beschreibungen zum Dokument mit dem spezifizierten Dokumenttyp (!DOCTYPE Name) in einer Datei mit der Erweiterung *.dtd* dargestellt.

```
<!DOCTYPE html PUBLIC
  "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd";>
```

5.2 Der XML-Verzeichnisbaum im Fachinformationssystem

Die Produkte des Teilprojektes B1 *„Dreidimensionale Plattenkinematik in Rumänien* bestehen aus folgenden Komponenten:

1. Metadaten (Kennzeichen und Merkmale der Dateninhalte).
2. Qualitätsanalyse (Positionsbestimmung der GPS-Stationen, Genauigkeitsbeurteilung).
3. Deformationsanalyse (Geschwindigkeit + Genauigkeit + Signifikanz).
4. Strainanalyse (elastische Verformung des Netzes: Dehnung + Stauchung + Winkelverformung).
5. Schwachformanalyse (Beurteilung der Netzsteifigkeit).
6. Allgemeine Qualitätsbeurteilung der Produkte (Produktkontrolle, Zustand des GPS-Netzes, Softwarekontrolle, ...).

Der Verzeichnisbaum zum Produkt *Deformationsanalyse* setzt sich aus den Komponenten Metadaten, Koordinaten, Geschwindigkeit und Genauigkeit zusammen. Dabei wird die Komponente *Koordinate* in die Elemente *<geozentrisch>, <geographisch>* und *<UTM>* aufgeteilt. Der Verzeichnisbaum in Abbildung 5.1 besitzt als Wurzelelement <web-page>. Die Struktur der folgenden Elemente orientiert sich an dem Aufbau der Tabellen. Der Tag <Tabelle_*Name*> bezieht sich entweder auf eine eindeutige Tabelle in der Datenbank oder auf eine Tabelle, die im Dokument dargestellt werden soll. Die Inhalte einer Zeile werden vom <read>-Tag umgeben. Jede Zeile kann unter Verwendung des Attributes *id=″Stationsname″* direkt ausgewählt werden (Anh. D.1).

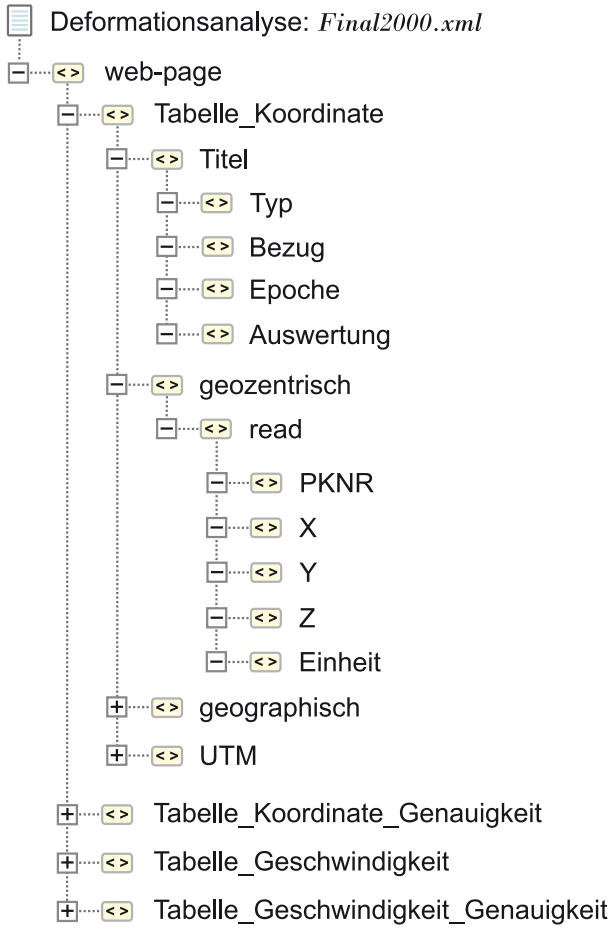


Abbildung 5.1: *Überblick über die Elemente des XML-Verzeichnisbaums zur Ausgabe der Ergebnisse aus der Deformationsanalyse „Final2000.xml“.*

5.3 XML-Architektur

Die Architektur verknüpft Komponenten, die im Kontext von XML als technische Basis für die Integration von Dokumenten, Medien, Datenverarbeitung und Kommunikationsnetzen eingesetzt werden (Mintert, [58]). Die Plattform XML bildet einen Unterbau für verschiedene Anwendungsbereiche und einem Überbau bestehend aus vier Säulen (Abb. 5.2).

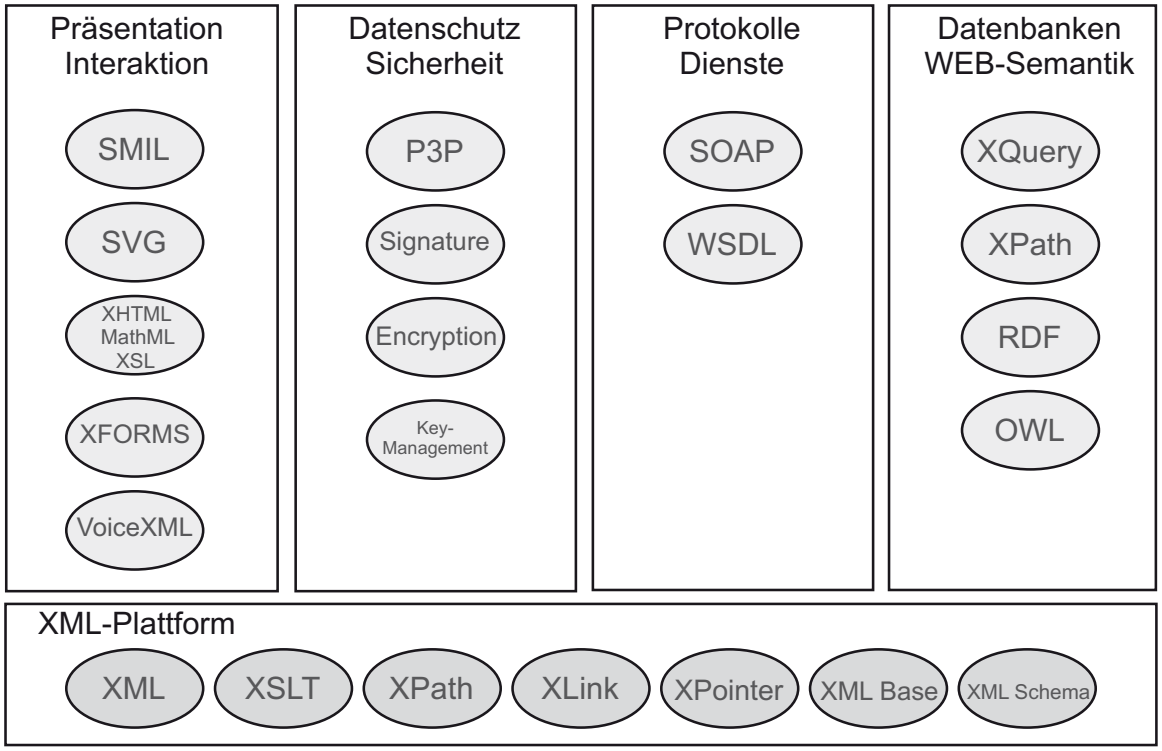


Abbildung 5.2: *Aufbau der XML-Architektur, bestehend aus der XML-Plattform und den vier Blöcken Präsentation/Interaktion, Datenschutz/Sicherheit, Protokolle/Dienste und Datenbanken/Web-Semantik (Birkenbihl, [8]).*

Die XML-Plattform besteht aus mehreren Modulen (XML, XSLT, XML Schema, XML Base, XPath, XLink, XPointer, usw.), welche das Fundament der XML-Architektur bilden. Diese Module sind zentrale Bausteine, um XML-Dokumenten zu erstellen, verschiedene Anwendungen miteinander zu verknüpfen, Methoden für XML-Datenbankabfragen zu entwickeln und Web-Dienste bzw. mobile Kommunikationssysteme aufzubauen. Der Überbau setzt sich auf folgenden Bereichen zusammen:

1. Präsentation und Interaktion: darunter fallen *Multimedia-Anwendungen* (Synchronized Multimedia Integration Language = SMIL), *Grafik-Anwendungen* (Scalable Vector Graphics = SVG), *Präsentationen* (XHTML, MathML, XSL) und *Interaktionen* (XFORMS). XFORMS bezeichnet eine Anwendung, die XML und Formulare kombiniert. Dabei interagieren die Benutzer mit den WEB-Applikationen. Die *Mobilität* wird durch Werkzeuge wie z.B. VoiceXML erreicht.

2. Datenschutz- und Sicherheit: Ein wesentlicher Aspekt von Web-Anwendungen ist der Schutz von persönlichen Informationen beim Zugriff auf Internet-Seiten. Daher wurde die Spezifikation P3P (Plattform für Datenschutz-Präferenz-Projekte) zur Standardisierung von *Datenschutz-Richtlinien* entwickelt. Dabei kann individuell die Sicherheitsstufe zur Bereitstellung persönlicher Informationen im Internet eingestellt werden (W3C, [85]). Es wird zurzeit eine XML-Syntax entwickelt, die *Signaturen* für Web-Ressourcen und Nachrichten bereitstellen soll und eine *Verschlüsselung* bzw. *Zertifizierung* von XML-Dokumenten ermöglicht.

3. Protokolle und Dienste: XML-Nachrichten werden über SOAP abgewickelt und die Web-Dienste durch WSDL beschrieben. Dabei wurde SOAP als ein Zugangsprotokoll für Objekte entwickelt, welches Strukturen und Eigenschaften für Nachrichten im XML-Format spezifiziert. WSDL ist die Bezeichnung für „*Web Service*

Definition Language“ und beschreibt Netzwerkdienste im XML-Format als eine Reihe von Nachrichtenvorgängen, welche Dokument- bzw. Prozedur-Informationen enthalten. WSDL nutzt außerdem eine Reihe von Netzwerkprotokollen und Nachrichtenformate (SOAP, HTTP, usw.), um Web-Dienste auszuführen.

4. Datenbanken und Web-Semantik: Um einen Zugriff auf XML-fähige Datenbanken zu erhalten, werden die Module *XQuery* und *XPath* eingesetzt. XQuery ist das Synonym für Anweisungen und Abfragen im Zusammenhang mit Datenbank-Verwaltungssystemen mittels XML. XPath wird zur Navigation und zum Auffinden der Knoten in einem XML-Dokument verwendet. Resource Description Framework (RDF) und Web Ontology Language (OWL) sind Semantik-Standards im Web. Deren Spezifikationen legen einen Rahmen für eine weltweit verknüpfte Datenverwaltung im Internet auf eine Weise fest, dass die Informationen miteinander verlinkt werden, um einen schnellen und einfachen Zugriff mittels der Suchmaschinen zu erreichen.

5.4 Transformation: XML + XSLT → (X)HTML

XSLT ist ein Teilbereich der erweiterten Stilvorlagen-Sprache *XSL*, welche XML-Dateien in eine Druckvorlage transformiert. Aufgrund der eindeutigen Struktur der XML-Dokumente können HTML-Seiten unter Verwendung spezieller Template-Regeln und über den Vorgang des Umwandelns der XML-Vorlagen mittels XSLT erstellt werden. Die Struktur der XML-Datei (Kap. 5.2) wird in XSLT als Knoten-Verzeichnisbaum abgebildet. Beginnend am Wurzel-Knoten, welcher einen abstrakten Knoten oberhalb des Dokument-Elementes darstellt, werden die Element-, Attribut-, Text-, Kommentar-Knoten, Steueranweisungen und Namensraum-Deklarationen hinzugefügt.

```
<?xml version="1.0"?>                          # Steueranweisung
<web-page xmlns="https://gikfis.gik.uka.de">   # Element-Knoten (Wurzelelement)
                                               # + Namensraum-Deklaration
  <!--Element: Tabelle_Koordinate-->           # Kommentar-Knoten (Tabellenname)
  <Tabelle_Koordinate>                         # Element-Knoten (1. Ebene)
    <Titel id=" 1: ">                          # Element-Knoten (2. Ebene)
                                               # + Attribut-Knoten (id)
      <Bezug>ITRF 97</Bezug>                   # Element-Knoten (3. Ebene)
                                               # + Text-Knoten
      <Epoche>15.06.99</Epoche>                # Element-Knoten (3. Ebene)
                                               # + Text-Knoten
    </Titel>
  </Tabelle_Koordinate>
</web-page>
```

Folgende Regeln können eingesetzt werden:

- Verwendung des Wurzel-Verzeichnisbaums als Indexverzeichnis für Such-Algorithmen,
- strukturierter Aufbau der Lokalisierungspfad-Anweisungen (z.B. Wurzel/Ebene1/Ebene2/...),
- Ausführung der Knotentests (z.B. Ausdruck: node() → findet jeden Knoten),
- Aufstellung von Ergebnislisten.

Die XSL-Datei *„final2000.xsl“* (Anh. D.2) enthält folgende Reihe von Anweisungen zur Transformation der XML-Datei *„final2000.xml“* (Anh. D.1) in eine korrekt formatierte HTML-Seite, welche über die FIS-Applikation (Kap. 7) heruntergeladen wird:

Steueranweisung: derzeit gültige Version ″1.0″ und Standard-Zeichensatz. ″UTF-8″

```
<?xml version="1.0" encoding="UTF-8"?>
```

Namensraum-Deklaration: die URL ″http://www.w3.org/1999/XSL/Transform″ wird in die Namensraum-Deklaration eingefügt.

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns="http://www.w3.org/TR/xhtml1/strict">
<xsl:output method="html"/>
```

Template-Anweisung: Aufbau der HTML-Seite mit der Kopfdeklaration (Titel: Fachinformationssystem) und der Rumpfdeklaration (Darstellung der SFB-Seite + Ausführung der nachfolgenden Template-Anweisungen).

```
<xsl:template match="/">
    <html>
        <head><title>Fachinformationssystem</title></head>
        <body text="#000000" bgcolor="#FFFFCC" link="#0000EE" vlink="#551A8B">
            <center>
                <h2>Deformationsanalyse: Final 2000</h2>
            </center>
            <xsl:apply-templates></xsl:apply-templates>
        </body>
    </html>
</xsl:template>
```

Darstellung der Tabelle *geozentrische Koordinate*. Aufbau von 5 Spalten, die als erste Zeile die Überschriften PKNR, X-Achse, Y-Achse, Z-Achse und Einheit enthalten. Die nachfolgenden Zeilen werden mittels der Anweisung *<xsl:value-of select = ″Elementname″/>* automatisch eingefügt, bis das Ende der Schleife *<xsl:for-each select = ″read″/>* im Verzeichnisbaum *Tabelle_Koordinate/geozentrisch* erreicht wird. Zusätzlich wird der Attribut-Knoten [*@id = ´Attributname´*] in die Verknüpfungsanweisung *<xsl:template match = .../>* miteinbezogen.

```
<xsl:template match="Tabelle_Koordinate/geozentrisch[@id='Koordinate']">
    <center>
        <table border="1" cellspacing="1" cols="5" bgcolor="#FFCC99">
              <tr>
                <td align="center" width="20%"><b>PKNR</b></td>
                <td align="center" width="20%"><b>X-Achse</b></td>
                <td align="center" width="20%"><b>Y-Achse</b></td>
                <td align="center" width="20%"><b>Z-Achse</b></td>
                <td align="center" width="20%"><b>Einheit</b></td>
              </tr>
            <xsl:for-each select="read">
              <tr>
                <td align="center" width="20%"><xsl:value-of select="PKNR"/></td>
                <td align="center" width="20%"><xsl:value-of select="X"/></td>
                <td align="center" width="20%"><xsl:value-of select="Y"/></td>
                <td align="center" width="20%"><xsl:value-of select="Z"/></td>
                <td align="center" width="20%"><xsl:value-of select="Einheit"/></td>
              </tr>
            </xsl:for-each>
        </table>
    </center>
</xsl:template>
```

5.5 JDOM

Das *Java Document Object Model* (JDOM) wurde entwickelt, um Java auf einfache Weise mit XML zu verbinden (Hunter/McLaughlin, [32]). Deshalb gilt: *„JDOM ist eine API zum Lesen, Schreiben und Manipulieren von*

XML-Dokumenten unter Verwendung von Java-Programmcodes“. Die Architektur der JDOM-Spezifikation setzt sich aus Klassen und Schnittstellen zusammen, die JDOM-Objekte aus diversen Quellen aufbauen und zu verschiedenen Bestimmungsorten transferieren. Die Ergebnisse werden aus Analysen, welche im Forschungsprojekt erzeugt und im Fachinformationssystem abgerufen werden, als XML-Dokumente unter Verwendung der JDOM^{API} ausgegeben. Dabei wird die Struktur der Dateninhalte mit Hilfe der Klasse *FinalResultToXML* (Anh. D.3), welche diverse Methoden zur Transformation (Datenobjekte $\Leftrightarrow$ XML-Objekte) bereitstellt, wohlgeformt aufgebaut. Zum Erstellen der in Kapitel 5.4 beschriebenen XML-Datei werden JDOM-Elemente benötigt (Tab. 5.1). Nach erfolgreicher Transformation ist ein wohlgeformtes XML-Dokument erzeugt worden.

String		**Wert**
jroot	=	″root″;
web	=	″web-page″;
jfile	=	″Deformationsanalyse; entwickelt im Sonderforschungsbereich 461″;
jtitle	=	″Dreidimensionale Plattenkinematik in den Ostkarpaten (Vrancea)″;
titel	=	″Titel″;
daten	=	″geozentrische Koordinate (X, Y, Z)″;

Objekt		**Anweisung**
Document doc	=	new Document(new Element(jroot).setName(web).setAttribute(″source″, jfile) .setAttribute(″title″, jtitle));
Comment ebene1	=	new Comment(″Element: Tabelle_Koordinate″);
Element root	=	doc.getRootElement( ).addContent(ebene1);
root	=	doc.getRootElement( ).addContent(new Element(koord_tab));
Comment ebene2	=	new Comment(″Element: Titel Attribut:table = Tabelle Attribut:id = $tabid″); root.getChild(koord_tab).addContent(ebene2); root.getChild(koord_tab).addContent(new Element(titel).setAttribute(″table″, ″Tabelle″).setAttribute(″id″, tabid));
Comment ebene3	=	new Comment(″Elementliste: <Typ />″); root.getChild(koord_tab).getChild(titel).addContent(ebene3); root.getChild(koord_tab).getChild(titel).addContent(new Element(typ).setText(daten));

Tabelle 5.1: *Erzeugen von XML-Dokumenten mittels JDOM.*

Zuerst werden die Parameter jroot, web, usw. definiert. Anschließend wird das Dokument mit dem Wurzelelement (root = web) und den Attributen (source = jfile, title = jtitle) erstellt. Mit dem Comment-Objekt werden Kommentarzeilen festgelegt. Danach wird ein neues Element (Tabelle_Koordinate) mit folgender Methode erzeugt:

⇒*getRootElement.addContent(new Element(String element));*

Weitere Element-Objekte folgen mit der getChild-Methode.

⇒*getChild(String element).addContent(new Element(String name).setAttribute(″attr″, ″Wert″));*

Zwischen den Kind-Element-Objekten werden weitere Kommentarzeilen eingefügt. Zuletzt werden die Dateninformationen, welche z.B. aus der Datenbank über JDBC abgerufen werden, mit folgender Methode integriert:

⇒*getChild(String element).addContent(new Element(String name).setText(String daten));*

Aufgrund der bereitgestellten Ausgabe-Klasse kann das Ausgabeformat individuell gestaltet werden.

⇒*org.jdom.output.XMLOutputter*

Zusätzlich können Attribute in die Steueranweisung eingefügt werden.

⇒*setEncoding(″ISO-8859-1″))*

Ausgabe der Ergebnisse aus der geodätischen Deformationsanalyse (Kap. 6.2.4.3) als wohlgeformtes XML-Dokument (vollständige Darstellung in Anhang D.1):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<web-page source="Deformationsanalyse; entwickelt im Sonderforschungsbereich
 461" title="Dreidimensionale Plattenkinematik in den Ostkarpaten (Vrancea)">
    <!-- Element: Tabelle_Koordinate -->
    <Tabelle_Koordinate>
         <!-- Element: Titel 1. Attribut:table=Tabelle 2. Attribut:id=$tabid -->
         <Titel table="Tabelle" id="1:">
             <!-- Elementliste: Typ -->
             <Typ>geozentrische Koordinate (X, Y, Z)</Typ>
             <!-- Einfügen weiterer Titel-Elemente -->
         </Titel>
         <!-- Einfügen der geozentrischen Koordinaten-Elemente -->
     </Tabelle_Koordinate>
     <!-- Einfügen weiterer Tabellen-Elemente -->
</web-page>
```

5.6 XML-Einsatz im Fachinformationssystem

Das XML-Konzept zeigt die Verbindungen zwischen den Komponenten auf, die im FIS zum Einsatz kommen (Abb. 5.3). Unter Verwendung der JDBC^{API} und JDOM^{API} werden die Informationen aus der Datenbank in XML-Dokumente umgewandelt (Hunter/McLaughlin, [33]). Dabei wird auf Wohlgeformtheit getestet. Die Daten werden anschließend in den Analyse-Prozess der geodätischen Programme (z.B. Deformationsanalyse) integriert oder mittels XSLT in HTML-Seiten eingebettet und anschließend im Web präsentiert. Im Bereich Middleware wird XML für die Meta-Informationen (Kap. 4.1.1) im AS eingesetzt.

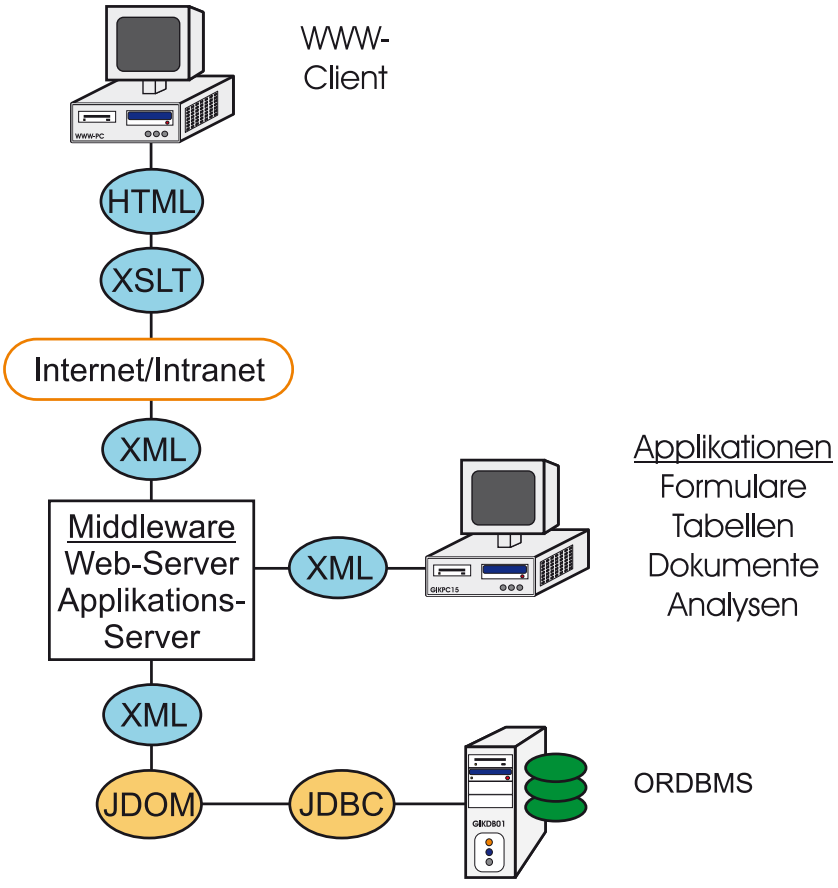


Abbildung 5.3: *Darstellung des XML-Konzeptes im Fachinformationssystem*

Kapitel 6

Geodätische Aspekte

Bei der Entwicklung des Fachinformationssystems wurden die geodätischen Aspekte zur Darstellung der Gefährdungsabschätzung von Starkbeben berücksichtigt. Die Lokalisierung der Starkbeben und ihre Auswirkung auf die Hauptstadt Bukarest stellt neben der technischen Realisierung eines Katastrophenmanagementsystems eines der Hauptziele des Sonderforschungsbereich 461 *„Starkbeben: von geowissenschaftlichen Grundlagen zu Ingenieurmaßnahmen“* dar. Deshalb wurden folgende Projektziele im geodätischen Bereich formuliert:

- Detektion von Plattengrenzen oder Verifizierung der Hypothese, dass aufgrund der fortgeschrittenen Kollision keine Plattengrenzen mehr durch geodätische Messmethoden nachweisbar sind (Cloetingh et al., [16]),
- Lokalisierung des Systems von Störungszonen Bistrita-, Trotus-Verwerfung, Peceneaga-Camena-System, Capidava-Ovidiu Verwerfung und die Intramoesische Störung (Abb. 6.1) durch Bestimmung der Deformationsparameter auf dem Kontinuum und Vergleich mit den geologischen Untersuchungen der tektonischen Evolution der Ost- und Südkarpaten (Matenco/Bertotti, [54]),
- Quantifizierung der Bewegungen der Einheiten *Tisia-Dacia-Block*, *Moesische Platte* und *Osteuropäische Plattform* mittels GPS (Globales Positionierungssystem),
- Bestimmung der aktiven tektonischen Deformationsstrukturen an der Oberfläche,
- Nachweis der rezenten Hebung der Ostkarpaten und der Vrancea-Region sowie der Setzung im Foscani-Becken aufgrund der Ablagerung von Sediment-Gesteinen,
- Analysen der rezenten Plattenkinematik bezüglich Genauigkeit und Signifikanz,
- Bestimmung von Qualitätsmerkmalen zur dreidimensionalen Qualitäts- und Deformationsanalyse.

Im Kontext der interdisziplinären Zusammenarbeit mit den Instituten der Fachrichtung Geologie, Geophysik und Bauingenieurwesen wurden folgende Anforderungen gestellt, welche zur optimalen Unterstützung des Projektmanagements im Fachinformationssystem umgesetzt wurden:

1. Umfassende Projektinformationen bezüglich der Durchführung der Messkampagnen, der Entwicklung des GPS-Netzes und der verwendeten GPS-Systeme.
2. Ausführliche Informationen über Position und Zustand der GPS-Stationen.
3. Dokumentation des Messablaufes, Auswertung der GPS-Daten, Bestimmung der dreidimensionalen Plattenkinematik.
4. Aufbau eines Qualitätsmanagements zur optimalen Beurteilung der erzielten Ergebnisse.

5. Grafische Darstellung der projektrelevanten Dateninhalte und Auswertungen (Mess-Konfiguration des GPS-Netzes, Schwachformen, Geschwindigkeiten, Deformationen, Genauigkeiten, statistische Tests, Bezugsrahmen, usw.).

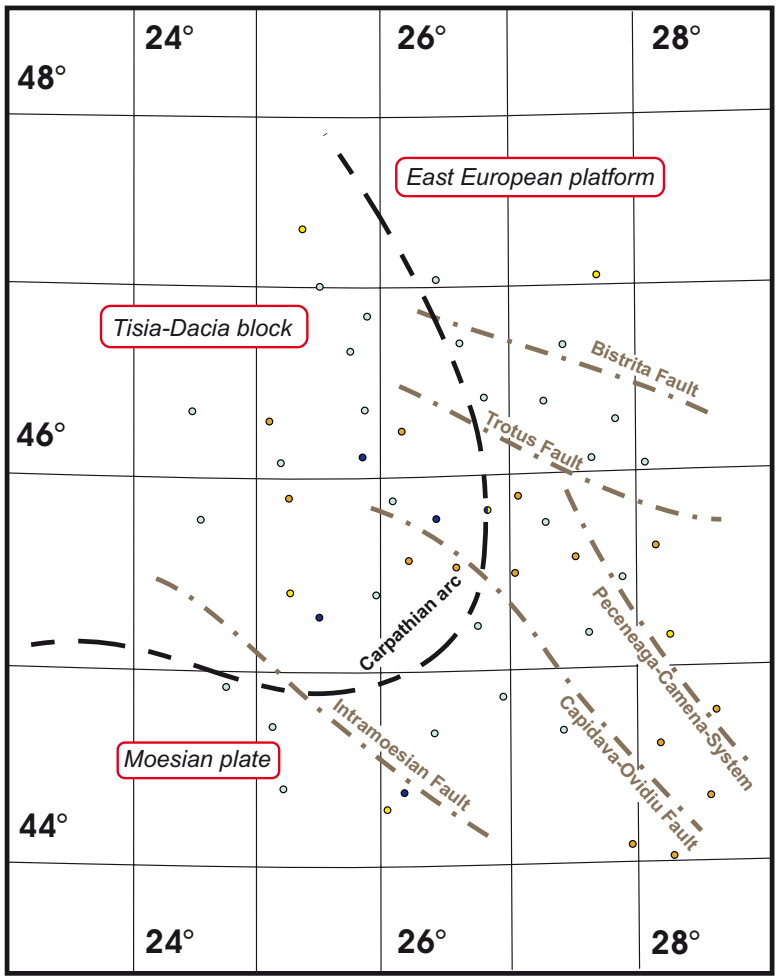


Abbildung 6.1: *Lokalisierung der Störungszonen im Bereich der Karpaten*

6.1 Dreidimensionale Plattenkinematik in den Ostkarpaten

Die Kinematik ist die Lehre von der Bewegung eines Körpers, ohne dabei auf die Kräfte als Ursachen für die Bewegungsarten einzugehen. Es erfordert daher die Kenntnis der rezenten Plattenkinematik, um krustale Spannungen und Erdbeben auslösende Entkopplungsmechanismen zu verstehen. Die Interpretation geodynamischer Modelle und die Analyse von Erdbeben, welche hauptsächlich als mitteltiefe Beben in der Vrancea-Region auftreten, sind wesentliche Beiträge zur Grundlagenforschung in Rumänien. Daher wurde ein GPS-Überwachungsnetz mit einer Ausdehnung von 600 km (West-Ost-Achse) $\times$ 400 km (Nord-Süd-Achse) eingerichtet, welches in Abständen von ein bis zwei Jahren mittels GPS-Verfahren gemessen wird, um rezente Bewegungen der vier Einheiten in Lage und Höhe zu bestimmen. Dabei besteht das tektonische Szenario aus der Osteuropäischen Plattform im Nordosten, dem Tisia-Dacia-Block im Zentrum und der Moesischen Plattform im Süden. Im Südosten von Rumänien wird eine vierte Einheit gebildet, die zwischen der Intramoesischen Störung und einem System von Störungszonen, der Trotus-Verwerfung und dem Peceneaga-Camena-System, liegt. Die diskreten Punkte, die auf den vier Einheiten dauerhaft vermarkt wurden, repräsentieren das Deformationsnetz. Aufgrund der räumlichen Positionsbestimmung der GPS-Stationen, die in einem Zeitraum von neun Jahren regelmäßig beobachtet wurden, werden die Geschwindigkeiten relativ zueinander in Lage und Höhe abgeleitet.

Es werden letztendlich die physikalischen Größen *Ort* und *Zeit* bestimmt. Ausgesuchte Permanentstationen, die in Zentral- und Osteuropa seit mehreren Jahren im Betrieb und Bestandteil des ITRF-Netzes sind, legen den Referenzrahmen fest. Als Bezugsepoche wurde ITRF2000.0 ausgewählt. Um relative Geschwindigkeiten zu erhalten, wird die Osteuropäische Platte, welche wiederum von einigen IGS-Permanentstationen repräsentiert wird, als ein Körper im Ruhezustand betrachtet. Dabei werden die tatsächlichen absoluten Geschwindigkeiten (z.B. Lage-Bewegung in Nordost-Richtung: $25 \pm 2.0\frac{mm}{a}$) vernachlässigt. Somit ist ein eindeutiger Referenzrahmen zur Bestimmung der Position (geozentrische und geographische Koordinaten im ITRF2000) und eine eindeutige Bezugsepoche zur Bestimmung der Geschwindigkeit (bezogen auf die Osteuropäische Plattform) definiert. Aufgrund der zu erwartenden geringen Geschwindigkeiten ($1 - 3\frac{mm}{a}$) und des kurzen Zeitintervalls von 1995 bis 2004 wird in erster Linie die *Kinematik der geradlinigen Bewegungen* betrachtet, insbesondere wird eine gleichförmige Bewegung geschätzt. Mögliche Beschleunigungen, Drehbewegungen oder Überlagerungen von Bewegungen einzelner Platteneinheiten sind in dem bisherigen Messzeitraum als *„nicht signifikant“* analysiert worden.

6.1.1 ITRF 2000

Der *internationale terrestrische Referenzrahmen* wurde als weltweites geometrisches Bezugssystem eingerichtet, welches die geodynamischen Deformationsprozesse der Erde berücksichtigt, die z.B. durch Erdgezeiten und Plattenkinematik hervorgerufen werden. Der Bezugsrahmen wird mit Hilfe von Referenzpunkten realisiert, deren Positionen durch verschiedene Messverfahren bestimmt werden. Die gemessenen Daten werden unter Berücksichtigung unterschiedlicher Verarbeitungsstrategien (getrennte bzw. kombinierte Berechnungsmethoden) ausgewertet. Die folgenden geodätischen Raumverfahren werden zur Realisierung des Referenzsystems eingesetzt (Schlüter, [71]):

- Satellite Laser Ranging (SLR) und Lunar Laser Ranging (LLR),
- Very Long Baseline Interferometry (VLBI),
- Global Positioning System (GPS) und Global Navigation Satellite System (GLONASS),
- Doppler Orbitography by Radiopositioning Integrated on Satellite (DORIS).

Zusätzlich vervollständigen folgende Beobachtungsmethoden die Festlegung des Bezugsrahmens:

- Synthetic Aperture Radar (SAR),
- Satellitenaltimetrie,
- Schweremessungen.

Das Referenzsystem stimmt weitgehend mit der Vorstellung eines geeigneten *„idealen terrestrischen Referenzsystems“* überein. Dabei wird das geozentrische Koordinatensystem im Massenmittelpunkt der Erde gelagert und die Z-Achse orientiert sich an der Rotationsachse der Erde. Die Ebene, aufgespannt durch die X- und Y-Achse, stellt die *Äquatorebene* dar. Die X-Achse liegt rechtwinklig zur Z-Achse und entspricht der Schnittgeraden, welche durch den Schnitt der Greenwich Meridian- und der Äquatorebene gebildet wird (Heck, [28]).

Sechs ausgewählte Stationen *GRAZ, PENC, SOFI, ZECK, JOZE* und *GLSV* definieren das SFB-Bezugssystem in der Epoche *„2000.0“* (Abb. 6.2). Die Stationskoordinaten und Geschwindigkeiten sind aus den Auswertungen vom International GPS Service (IGS) abgeleitet (ITRF, [42]) und bilden den Referenzrahmen zur Bestimmung der Plattenkinematik in Rumänien. Weitere IGS-Stationen werden in die Deformationsanalyse miteinbezogen, um einerseits das überregionale Bewegungsverhalten zu ermitteln und andererseits einen stabileren Referenzrahmen bezüglich des rumänischen Subnetzes zu erhalten. Dabei werden die Permanentstationen in den Bezugsrahmen miteinbezogen, deren Analysen keine relativen Bewegungen untereinander anzeigen. Die IGS-Station Bukarest ist im GPS-Subnetz enthalten und bildet zusammen mit weiteren regionalen Permanentstationen das Gerüst des SFB-Netzes (Hoeven et al., [29]).

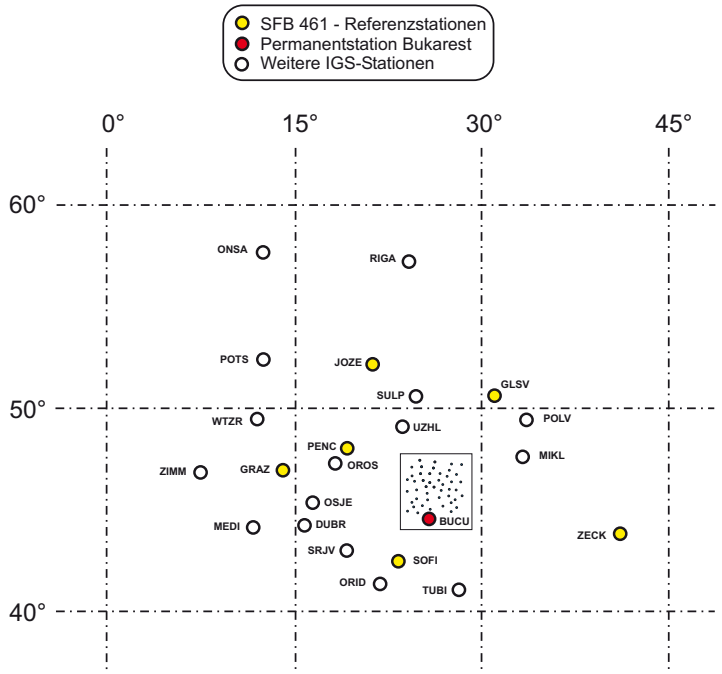


Abbildung 6.2: *Der Referenzrahmen des SFB 461 - eingebettet im ITRF2000 (Epoche 2000.0).*

6.1.2 Tektonisches Szenario im Karpatenbogen

Nach aktueller Modellvorstellung liegt eine abgeschlossene Delamination des lithosphärischen Mantels bis auf einen *Slab* im südöstlichen Bogen unterhalb der Vrancea Region vor (Wortel/Spakman, [88]). Dieser Slab stellt ein Reststück der subduzierten ozeanischen Lithosphäre dar und ist im nordöstlichen Teil der seismisch aktiven Vrancea-Zone mit der Lithosphäre verbunden. Nach der Kollision versteilte sich der Slab in nahezu vertikaler Stellung (Sperner et al., [75]) und befindet sich heute als Hochgeschwindigkeitskörper unterhalb der Südost-Karpaten und sinkt in die Asthenosphäre ab (Abb. 6.3).

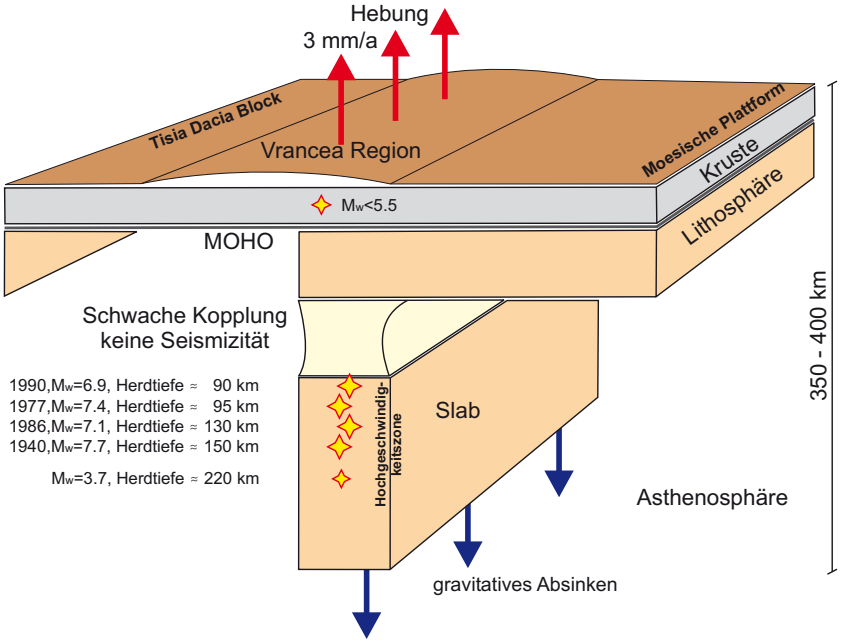


Abbildung 6.3: *Interpretation des geodynamischen Modells im Kontext der neotektonischen Entwicklung. Die Abbildung ist nach (Müller et al., [59], S. 147) erstellt und erweitert worden.*

Verschiedene Hypothesen zur Kopplung des Slabs wurden aufgestellt (Sperner et al., [74]):

I Kopplung: aufgrund des gravitativen Absinkens des Slabs wird die kontinentale Kruste in die umliegende Region miteinbezogen → die Vrancea-Region sinkt ebenfalls ab, starke Erdbeben sind zu erwarten.

II Schwache Kopplung: aufgrund der schwachen Verbindung zwischen Slab und Kruste wird die überlagernde Kruste entlastet → die Region hebt sich leicht, Erdbeben unterhalb der Kruste in 40-70 km Tiefe sind nicht zu erwarten (Bonjer et al., [9]), Nachweis einer Niedriggeschwindigkeitszone unterhalb der Moho.

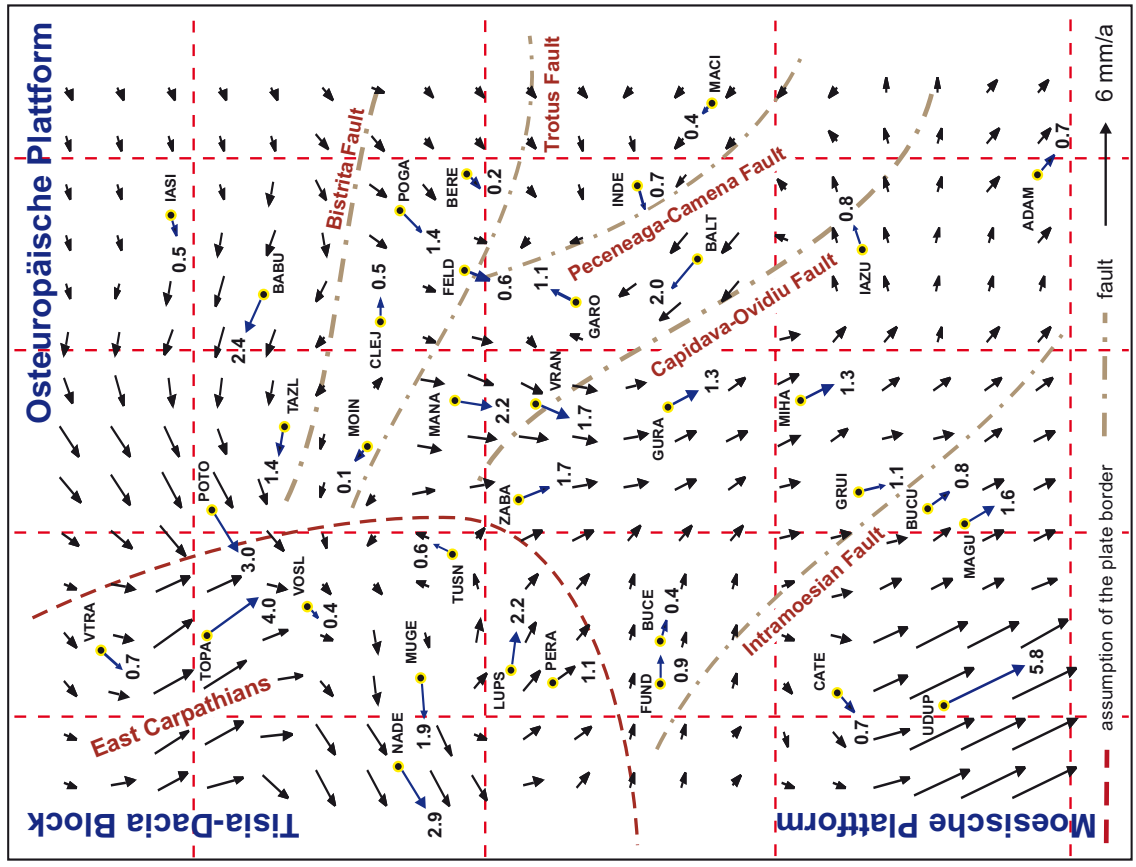
III Keine Kopplung mehr vorhanden: der Slab sinkt in die Asthenosphäre ab, die Kruste ist entkoppelt → aufgrund des lokalen isostatischen Ausgleichs hebt sich die Vrancea-Region, Erdbebengefährdung verringert sich.

IV Postseismische Spannungsumlagerungen der Starkbeben (insbesondere des 77-er Bebens) überlagern die vertikalen Deformationsmuster. Weitere Prozesse können daran beteiligt sein.

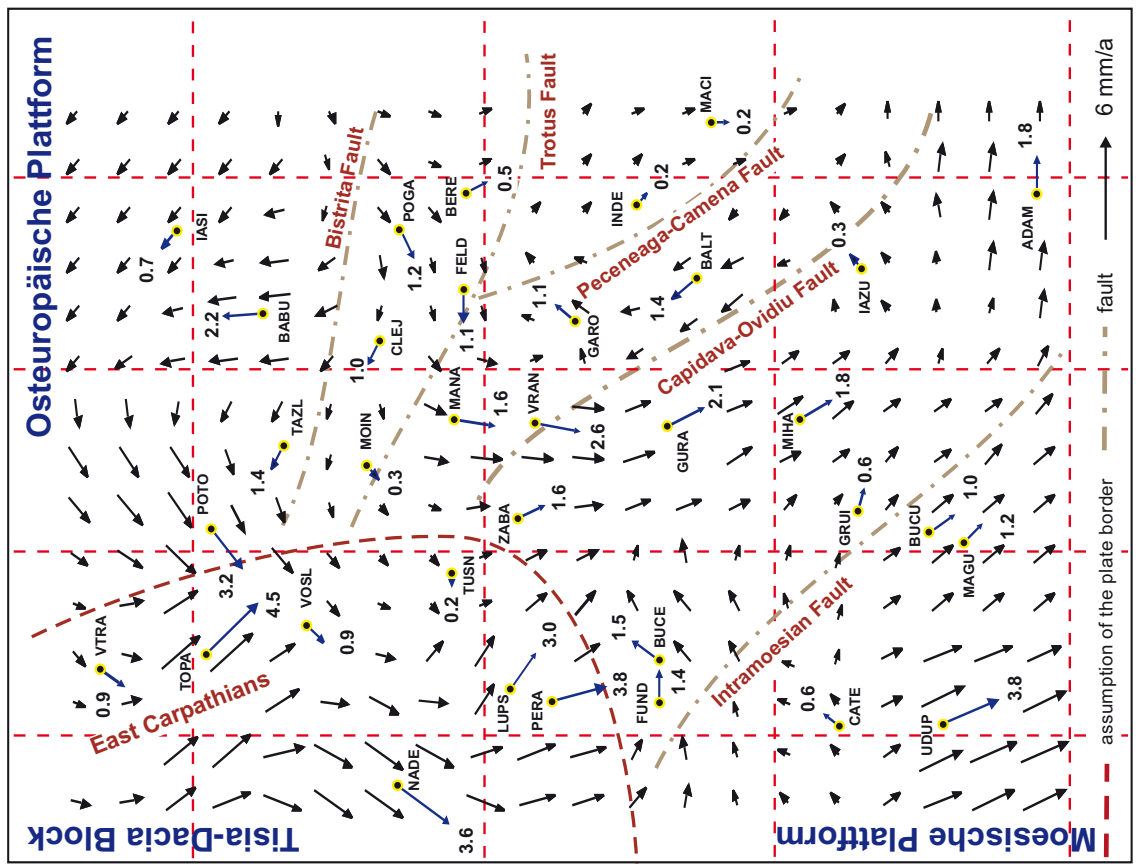
6.1.3 Horizontale Plattenbewegung

Die Deformationsanalyse 2003 und 2004 berechnet die horizontalen Geschwindigkeiten der diskreten Punkte, welche im Zeitraum von 1995 bis 2003 bzw. bis 2004 besetzt wurden. Die Gegenüberstellung der beiden Ergebnisse verdeutlicht die Zuverlässigkeit und tatsächliche Genauigkeit der Lösungen (Abb. 6.4). Insbesondere die starke Streuung einzelner Stationen lässt sich auf starke lokale Einflüsse oder auf einen relativ kurzen Messzeitraum zurückführen → mehrere Stationen wurden zur Erweiterung des GPS-Netzes auf dem Tisia-Dacia-Block und der Moesischen Plattform erst im Jahre 1999 vermarkt. Der Vergleich der beiden berechneten Vektorfelder zeigt für die Lagekomponente folgendes auf:

- Zwischen der Intramoesischen Störung und der Capidava-Ovidiu Verwerfung bewegt sich die tektonische Einheit in der Größenordnung von $\approx 2.0 \pm 0.5 \frac{mm}{a}$ in Richtung Südosten.
- Die Moesische Plattform bewegt sich in Richtung Südosten, wobei keine klare Aussage über die Geschwindigkeit getroffen werden kann, da lokale Einflüsse die Bewegung der Station UDUP überlagern und die Geschwindigkeitsvektoren der Station CATE im Vergleich 2003 und 2004 zu stark streuen. Realistisch scheint die Bewegung der Stationen BUCU und MAGU mit ca. $1.0 \frac{mm}{a}$ zu sein.
- Zwischen der Capidava-Ovidiu und der Peceneaga-Camena Verwerfung scheint eine aktive Zone vorhanden zu sein, deren tektonische Einheit sich in gegensätzlicher Richtung (Nordwest-Orientierung) bewegt ($\approx 1.5 \frac{mm}{a}$).
- Die Lage der Region zwischen der Peceneaga-Camena und Trotus Verwerfung ist bezüglich des Bezugsrahmens relativ stabil.
- Es zeichnet sich ein Trend der Plattenbewegung in Richtung Westen oberhalb der Bistrita Verwerfung ab ($\approx 1.5 \frac{mm}{a}$).
- Die Bewegungen oberhalb des Karpatenbogens in nordwestlicher Richtung sind uneinheitlich und zurzeit nicht aussagekräftig.
- Der Grenzbereich der maximalen Geschwindigkeit liegt bei ca. $5.0 \frac{mm}{a}$.



6.4.2: Horizontales Geschwindigkeitsfeld 2004



6.4.1: Horizontales Geschwindigkeitsfeld 2003

Abbildung 6.4: *Gegenüberstellung der geodätischen Deformationsanalysen 2003/2004 zur Abschätzung der Zuverlässigkeit von horizontalen Geschwindigkeiten.*

6.1.4 Vertikale Plattenbewegung

Die Ergebnisse aus den GPS-Messungen zeigen folgende neotektonische Entwicklung der vertikalen Krustenbewegung in Rumänien auf (Abb. 6.5):

- **Karpatenbogen:** Der Karpatenbogen und die Vrancea Region heben sich in der Größenordnung von $3.5 \pm 1.7\,\frac{mm}{a}$. Durch abgelagerte Sedimente wird der allgemeine Trend im Brasov Becken unterbrochen. Dort ist eine Setzung von $\approx 4.0 \pm 1.7\,\frac{mm}{a}$ zu beobachten. Unter Berücksichtigung der signifikanten Hebungsrate oberhalb des Slabs und der „seismischen Lücke“ unterhalb der Moho entspricht Hypothese II in Kombination mit Hypothese IV (bezogen auf Kapitel 6.1.2) am ehesten der Realität.
- **Ostkarpaten:** Die Bewegungen zeigen ein unheitliches Bild. Regionen mit leichten Hebungen ($\leq 1.0\,\frac{mm}{a}$) wechseln sich mit abgesenkten Bereichen ($\leq 2\frac{mm}{a}$) und stärker gehobenen Zonen ($2.0 \leq v \leq 3.0\,\frac{mm}{a}$) ab. Aufgrund der wenigen signifikanten Geschwindigkeiten ist zurzeit keine Aussage zur Geodynamik in den Ostkarpaten zu treffen. Weitere GPS-Beobachtungen sind durchzuführen.
- **Foscani Becken:** Es ist eine Setzung in der Größenordnung von $2.0 \pm 1.5\frac{mm}{a}$ zu beobachten. Diese wird auf abgelagerte pliozäne Sedimentabfolgen zurückgeführt und koinzidiert mit geologischen Untersuchungen (Matenco et al., [55]).
- **Vorland der Ostkarpaten:** Das Vorland senkt sich aufgrund der Sedimentablagerungen um ca. $2.0 \pm 1.5\,\frac{mm}{a}$ ab.
- **Ostbereich zwischen Bistrita und Trotus Verwerfung:** Es zeichnet sich der Trend einer Hebung ab, wobei die Größenordung zwischen 0.5 und $2.5\frac{mm}{a}$ streut.
- **Nord Dobrogea:** Der Bereich hebt sich um $\approx 2.0 \pm 1.5\frac{mm}{a}$. Diese Ergebnisse sind jedoch für abschließende Analysen nicht zuverlässig genug.
- **Moesische Plattform:** Die Anzahl der Stationen ist nicht repräsentativ zur Bestimmung der Plattenbewegung. Weitere Stationen sind zur Steigerung der Genauigkeit und Zuverlässigkeit der Plattenbewegung vermarkt worden. Die bis zu diesem Zeitpunkt ermittelten Geschwindigkeiten sind aufgrund der Zeitspanne von 4 Jahren zu ungenau ($\pm 3.5\frac{mm}{a}$).

6.2 Qualitätsmanagement

Ein Qualitätsmanagement (QM) umfasst alle Tätigkeiten, welche die Qualitätspolitik eines Unternehmens vorgibt, um die Ziele und Verantwortungen im Rahmen des QM-Systems festzulegen. Die Realisierung des QM-Systems erfolgt durch den Einsatz der Qualitätsplanung, -kontrolle, -sicherung und -verbesserung (ISO, [40]).

Im Normalfall unterziehen sich die Unternehmen regelmäßigen Kontrollen oder lassen sich nach der *ISO-Normreihe* von unabhängigen und autorisierten Stellen zertifizieren. Diese Verfahren werden deshalb angewiesen, um eine Tätigkeit (Qualitätskontrolle bzw. -sicherung) auf eine festgelegte Art und Weise auszuführen. Die Komponenten zur Realisierung eines QM-Systems lassen sich folgendermaßen beschreiben (ISO, [39]):

- **Qualitätsplanung:** Festlegung der Ziele unter Berücksichtigung der Qualitätsanforderungen. Zusätzlich werden die Elemente eines QM-Systems spezifiziert. Qualitätsplanung umfasst dabei folgende Planung bezüglich des Produktes:
  - Identifizierung, Klassifizierung und Gewichtung der Qualitätsmerkmale,
  - Spezifizierung der Ziele, der Qualitätsanforderungen und der einschränkenden Bedingungen,
  - Planung bezüglich der Führungs- und Ausführungstätigkeiten,

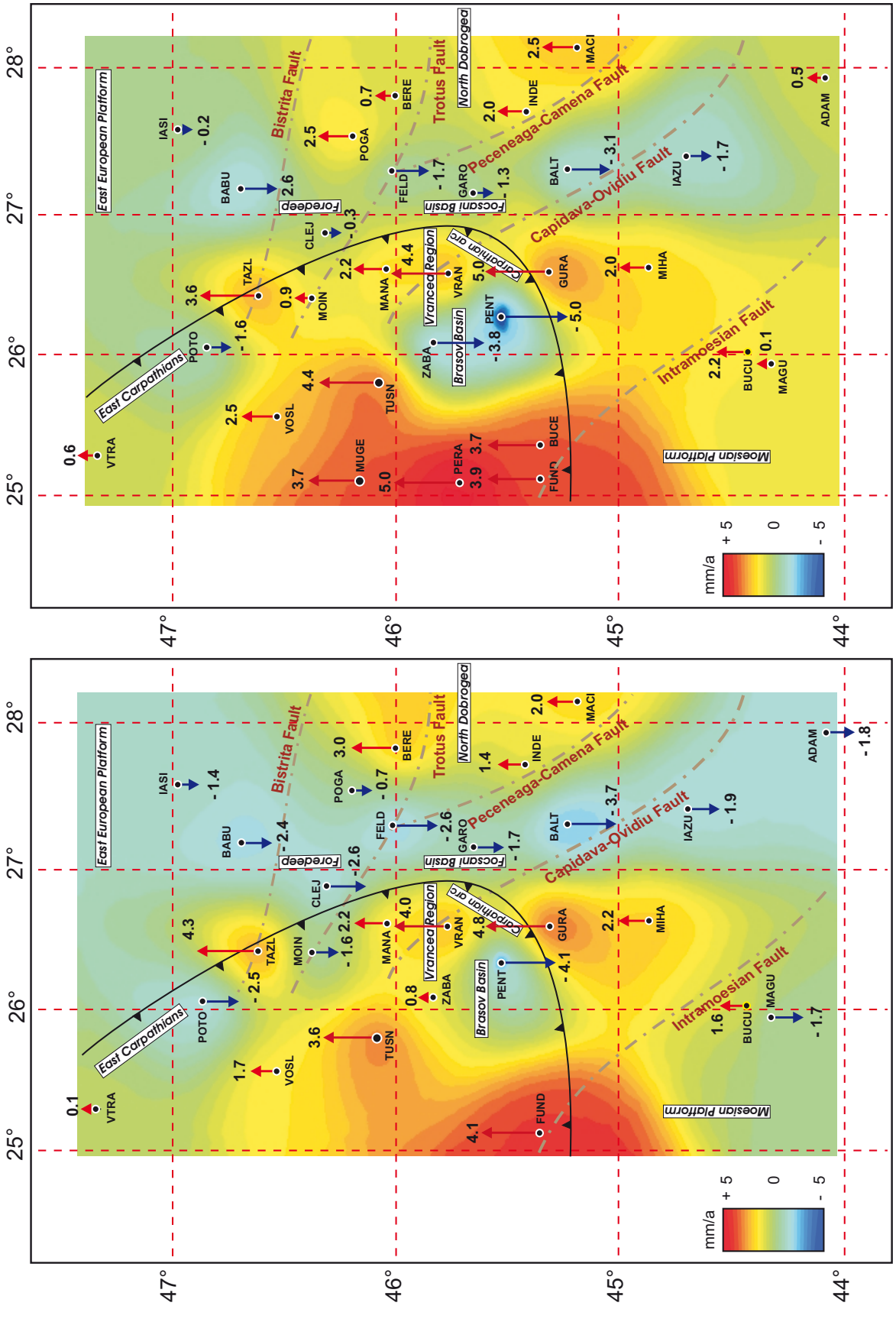


Abbildung 6.5: *Gegenüberstellung der geodätischen Deformationsanalysen 2003/2004 zur Abschätzung der Zuverlässigkeit von vertikalen Geschwindigkeiten.*

  - Vorbereitung des QM-Systems mit Ablauf- und Zeitplänen.

- **Qualitätskontrolle:** Ausführung der Kontrollmechanismen und Tätigkeiten, die Qualitätsforderungen an ein Produkt steuern und kontrollieren.
- **Qualitätssicherung:** Durchführung sämtlicher Maßnahmen und Aufgaben, welche die Qualitätsanforderungen erfüllen und damit Vertrauen schaffen. Die Qualitätssicherung geht aus der Qualitätskontrolle hervor, welche Maßstäbe zur Einhaltung der Qualität spezifiziert.
- **Qualitätsverbesserung:** Erfassung des Maßnahmenkataloges, um einerseits die Effektivität von Arbeitsabläufen und Prozessen zu erhöhen und andererseits das Produkt soweit zu verbessern, dass maximale Wirtschaftlichkeit für das Unternehmen und maximaler Nutzen für die Kunden erzielt wird.
- **Qualitätsmanagement:** Übertragung der Verantwortung auf allen Entscheidungsebenen unter Einschaltung des Managements eines Unternehmens.

6.2.1 Qualitätsmanagement bei Geodaten

Es werden Prüfverfahren zur Aufdeckung eventueller Abweichungen der Qualitätsziele zur dreidimensionalen Plattenkinematik entwickelt, die entweder automatisch mit Hilfe von definierten Testkriterien (Aufdeckung grober Fehler, Festlegung von Genauigkeitsmaßen) die digitalen Daten kontrollieren oder durch visuelle Vergleiche des Prüfers (Verifizierung und Validierung der Analysen) die Qualitätsgüte ermitteln. Die Prüfungen erfolgen in jährlichen Abständen, wobei vorausgesetzt wird, dass eine neue Kampagne in Rumänien durchgeführt wurde. Es wird die Konsistenz, Richtigkeit und Vollständigkeit des gesamten Datenbestandes durch Stichproben der digital erfassten Daten geschätzt, die in eine Datenbank zur permanenten Datenhaltung übertragen werden. Die Qualitätsverbesserung erfolgt über eine kritische Bewertung der durchgeführten Kampagnen, Verbesserung der Analyseverfahren und Entwicklung neuer Modelle zur Auswertung der GPS-Daten. Durch Einführung geeigneter Werte für Qualitätsmaße werden zur Erfüllung der Qualitätsziele folgende Methoden des QM angewendet (Joos, [45], S. 60):

1. Aufbau eines Qualitätsmanagements: Planung der Projektleitung, Festlegung der Projektziele, Ausbildung der wissenschaftlichen Mitarbeiter, Dokumentation der Projektentwicklung, laufende Überprüfung des Projektstandes, Erstellen von Berichten und Veröffentlichungen sowie eventuellen fortführenden Projektanträgen.

2. Prüfung auf Richtigkeit, Vollständigkeit und Konsistenz: Entwicklung von softwaretechnischen Maßnahmen zur automatischen Kontrolle des digitalen Datenbestandes. Einsatz eines leistungsfähigen DBMS.

3. Statistische Verfahren zur Qualitätskontrolle: Analysen im Bereich Qualität, Deformation, Strain und Schwachform werden mit statistischen Tests untermauert. Aufgrund der berechneten Qualitätsmaße der Koordinaten, Geschwindigkeiten, Spannungen und Schwachformen werden Kriterien über Signifikanz und Zuverlässigkeit der Produkte abgeleitet.

4. Verifizierung und Validierung: Die Ergebnisse werden mit unabhängigen Methoden (z.B. Untersuchung aktiver Deformationsstrukturen durch Flussterrassen- und Grabenstrukturanalysen im Ostkarpatenbogen) auf Plausibilität überprüft und durch unterschiedliche GPS-Auswertestrategien und Analyse-Tools des niederländischen Kooperationspartners validiert.

6.2.2 Qualitätskreislauf im Projekt - Umsetzung im FIS

Es ergeben sich zwei Schwerpunkte zur Erfüllung der Projektanforderungen. Auf der einen Seite erfolgt die Planung und Durchführung der Kampagne, deren Qualitätsziele eine optimale Besetzung der Stationen und Ausbildung der Teilnehmer enthalten. Auf der anderen Seite müssen Auswertestrategien und Analyseprozesse den aktuellen Modellen bzw. Verfahren angepasst werden. Der Kreislauf ergibt sich aus den Projektzielen (Abb. 6.6): Herstellung von Dateninformationen, die auf Genauigkeit und Zuverlässigkeit überprüft wurden,

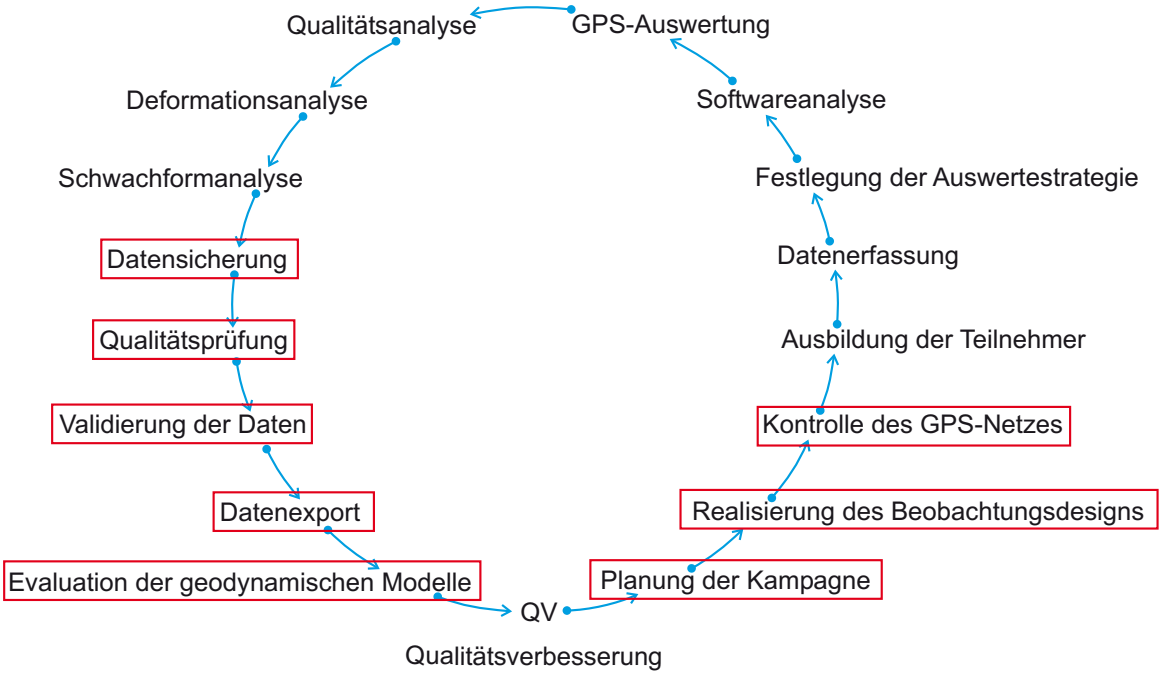


Abbildung 6.6: *Darstellung des Qualitätsmanagement-Kreislaufes im Teilprojekt B1 unter Einbeziehung des FIS als Hilfsmittel.*

zur Bestimmung der dreidimensionalen Plattenkinematik und Anwendung der Ergebnisse zur Modellierung rezenter geotektonischer Prozesse.

Im Rahmen der Applikationsentwicklung (Kap. 7) wurden Module zur Qualitätsverbesserung der im Projekt festgesetzten Anforderungen erstellt (in Abbildung 6.6 rot eingerahmt). Es werden zur *Planung der Kampagnen* und *Realisierung des Beobachtungsdesigns* sämtliche notwendigen Informationen (Kap. 7.4) im FIS bereitgestellt, um eine GPS-Kampagne in Rumänien durchzuführen. Qualitätsverbesserungen bezüglich des Beobachtungsdesigns werden aus den Schwachformanalysen (Kap. 6.2.4.4) in das FIS übernommen. Aufgrund der besonderen Umstände in Rumänien ist es notwendig geworden, das GPS-Netz jährlich auf Beschädigungen oder Zerstörungen zu kontrollieren. Die Feldbücher zum Auffinden der Stationen und Protokollieren des Stationszustandes (Status: *ok*, *beschädigt* - Reparatur notwendig, *mangelhaft* - Wiederherstellung möglich oder Ausweichen auf Exzentrum, *zerstört* - Aufgabe der Station) sind über das FIS abrufbar (Kap. 7.6).

Die Voraussetzungen zur *Qualitätsprüfung* und *-verbesserung* sind hauptsächlich in der statistischen Analyse der Projektschwerpunkte (1. Auswertestrategien, 2. Software, 3. Qualität des GPS-Netzes, 4. Deformationen und 5. Schwachformen) zu finden. Das FIS unterstützt dabei die Entscheidungsfindung des Qualitätsprüfers bei der *Validierung der Daten* durch tabellarische und grafische Darstellungen der Ergebnisse aus den betreffenden Analysen (Kap. 7.7). Dabei werden Vergleiche der Analysen bezüglich verschiedener Epochen durch den Zeitbezug im Datenmodell ermöglicht (Kap. 2.4.6). Die Sicherstellung der Datenintegrität wird über den Einsatz eines objektrelationalen Datenbank-Managementsystemes erreicht (Kap. 2). Der Datenexport zu verschiedenen Auswerteprogrammen oder zur Präsentation der erzielten Ergebnisse wird über wohlgeformte XML-Dokumente (Kap. 5.1) geregelt. Zusätzlich können über den FIS-SQL-Editor spezielle DB-Anfragen zu den einzelnen Objektklassen gestellt werden (Kap. 7.8).

6.2.3 Qualitätsanforderungen und -ziele

Die Qualitätsziele teilen sich in zwei Hauptgruppen auf: erstens soll eine bestmögliche Durchführung der Kampagne ermöglicht und zweitens eine optimale Auswertestrategie entwickelt werden. Daher werden Kontrollmechanismen eingesetzt, um die Machbarkeit der definierten Qualitätsanforderungen zu überprüfen. Folgende Qualitätsanforderungen bzw. -ziele wurden im Projekt formuliert:

Ausbildung der Teilnehmer: ein dreitägiges Programm zur „Einführung in GPS“ bildet die Grundlage für die Messungen in jeder Kampagne.

GPS-Netz: das Überwachungsnetz wird vor jeder Kampagne kontrolliert. Der Zustandsbericht ist Grundlage zur Planung des Beobachtungsdesigns. Eventuelle Wiederherstellungen oder komplette Zerstörungen der Stationen müssen berücksichtigt werden. Die Einordnung der Qualitätsmaße (ok, beschädigt, mangelhaft, zerstört) der GPS-Stationen bestehend aus einem Hauptpunkt und zwei Exzentren erfolgt vor Ort.

Stationskontrolle: die Messteams werden von mindestens drei Organisationsteams während einer Kampagne betreut. Die Dokumentation der Stationsbesetzungen wird von den Aufsichtsteams kontrolliert.

Datenerfassung: jede Station wird mindestens 72 Stunden lang durchgehend besetzt.

GPS-System: es wird ein System der gleichen Baureihe und desselben Unternehmens in einer Kampagne eingesetzt.

Auswertestrategie: es wird die aktuellste wissenschaftliche GPS-Auswertesoftware unter Berücksichtigung neuester Erkenntnisse in der Modellierung der Troposphäre, der Antennenkalibrierung und der Einflüsse der Mehrwegeeffekte verwendet. Realisierung einer optimalen GPS-Auswertestrategie zur Positionsbestimmung der diskreten Punkte.

Genauigkeitsmaße Position: die empirische Streuung der Stationskoordinaten einer freien Netzausgleichung beträgt in der Lage $-4\,mm \leq \sigma_L \leq +4\,mm$ und in der Höhe $-12\,mm \leq \sigma_H \leq +12\,mm$ bei einer Irrtumswahrscheinlichkeit $\alpha = 1\%$. Es gilt für die Sicherheitswahrscheinlichkeit P des Konfidenzintervalls der geodätischen Beobachtung l, dem Erwartungswert λ mit bekannter Standardabweichung σ unter der Annahme einer Normalverteilung $l \sim N(\lambda, \sigma^2)$ und dem festgelegten kritischen Wert (Quantile) c = 2.576 folgendes:

$$P(l - c_{(1-\frac{\alpha}{2})}\sigma < \lambda < l + c_{(1-\frac{\alpha}{2})}\sigma) = 1 - \alpha = 99\% \tag{6.1}$$

Genauigkeitsmaße Geschwindigkeit: die Genauigkeit der Stationsgeschwindigkeiten ist abhängig von dem gewählten Zeitintervall I. Bei $I = [1995, 2004]$ beträgt die empirische Streuung in der Lage $-1.0\frac{mm}{a} \leq \sigma_L \leq +1.0\,\frac{mm}{a}$ und in der Höhe $-2.5\,\frac{mm}{a} \leq \sigma_h \leq +2.5\,\frac{mm}{a}$ bei einer Irrtumswahrscheinlichkeit $\alpha = 1\%$.

6.2.4 Qualitätsprüfung

Der Schwerpunkt der Qualitätsprüfung liegt in der statistischen Erfassung der Aufdeckung grober Fehler (Kap. 6.2.4.2), Bestimmung des mittleren Fehlers, Beurteilung der Genauigkeit und Zuverlässigkeit der Lösungen. Zuerst werden die einzelnen Kampagnen untersucht, um die innere Genauigkeit der Geometrie eines Netzes zu beurteilen. Dabei werden insbesondere die Einflüsse der Fehler mittels Spektralzerlegung der Normalgleichungsmatrix einer vermittelnden Ausgleichung im Gauss-Markov-Modell auf die Netzcharakteristik des Überwachungsnetzes untersucht. Anschließend erfolgt eine Einzelpunktanalyse über den gesamten Messzeitraum, zur Bestimmung der Stationsgeschwindigkeiten, der Exzentren in der Lagekomponente und der Offsets in der Höhenkomponente. Nachdem die groben Fehler eliminiert wurden, erfolgt eine koordinatenbezogene Deformationsanalyse im einheitlichen stabilen Datum, das durch ausgewählte IGS-Stationen auf der Osteuropäischen Plattform definiert wird. Die Abbildung 6.7 stellt das Deformationskonzept 2004 vor, welches eine optimale Auswertestrategie garantieren soll.

6.2.4.1 Qualitätsanalyse

Die dreidimensionale Qualitätsanalyse wurde in Form einer Gesamtnormminimierung eines freien Netzes durchgeführt. Um die Konsistenz des Gleichungssystems (Kap. 6.2.4.3) zu erreichen, wurde eine Varianzkomponentenschätzung für die drei Beobachtungsgruppen geographische Breite (B), geographische Länge (L) und ellipsoidische Höhe (h) durchgeführt. Der globale Faktor, welcher die zu optimistischen Genauigkeitsangaben aus der GPS-Auswertung korrigiert, beträgt für die Lagekomponente 10 und für die Höhenkomponente

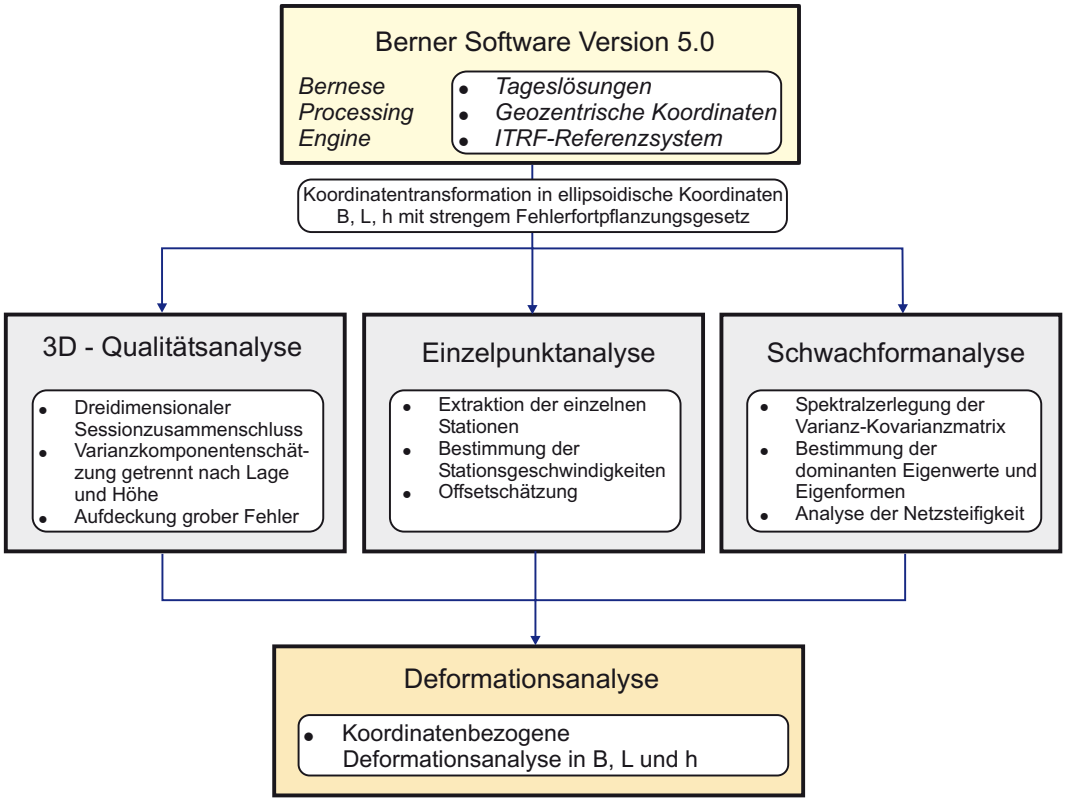


Abbildung 6.7: *Darstellung des Deformationskonzeptes 2004 zur optimalen Auswertung der GPS-Daten und Analyse des geodynamischen Szenarios in den Ostkarpaten. Das Konzept besteht aus den Komponenten a) wissenschaftliche GPS-Auswertesoftware, b) Kampagnen-Qualitätsanalysen, c) Einzelpunktanalysen über den Zeitraum 1995 bis 2004 und d) Schwachformanalysen der GPS-Netze. Nach Bereinigung der groben Fehler bzw. Schätzung weiterer Parameter (Offsets und Exzentren) wird eine Deformationsanalyse über den gesamten Projektzeitraum durchgeführt.*

5. Lokale Faktoren wurden nicht eingeführt, da die einzelnen Sessions (Tageslösungen) überwiegend homogene Genauigkeiten für die einzelnen Komponenten aufweisen. Die in Tabelle 6.1 angegebenen maximalen, minimalen und durchschnittlichen Punktfehler ($\alpha = 1\%$) entsprechen den ermittelten Genauigkeitsangaben der einzelnen Qualitätsanalysen. Im Vergleich zu den Qualitätszielen wird offensichtlich, dass die Anforderungen mit den tatsächlich zu erreichenden Genauigkeiten mit wenigen Ausnahmen übereinstimmen. Jedoch ist anzumerken, dass diese Genauigkeiten erst nach Bereinigung der Daten (Entfernen grober Fehler und Schätzung von Exzentren und Offsets mittels Einzelpunktanalyse) erreicht werden.

Minimaler Punktfehler	**mm**
Lage	1.5
Höhe	4.0
Durchschnittlicher Punktfehler	**mm**
Lage	3.0
Höhe	8.0
Maximaler Punktfehler	**mm**
Lage	6.0
Höhe	15.0

Tabelle 6.1: *Minimaler, durchschnittlicher und maximaler Punktfehler.*

6.2.4.2 Einzelpunktanalyse

Um die systematischen Abweichungen einzelner Stationen herauszufiltern, werden *Offsets* für die Höhenkomponente und *Exzentren* für die Lagekomponente geschätzt. Dabei werden alle Ergebnisse aus der GPS-Auswertung einer Einzelpunktanalyse unterzogen, indem die Stationskoordinaten und ihre Varianz-Kovarianzmatrix (VKM) aus den GPS-Tageslösungen extrahiert und der Deformationsanalyse zugeführt werden. Im Rahmen der Analyse werden Exzentren und Offsets geschätzt, wenn folgende Kriterien erfüllt sind:

- die Residuen der Lagekomponente sind größer als 6.0 mm (maximaler Punktfehler) und können eindeutig einem Beobachtungsblock zugeordnet werden,
- die Residuen der Höhenkomponente sind größer als 15 mm (maximaler Höhenfehler) und können eindeutig einem Beobachtungsblock und somit einer Antenne zugeordnet werden.

Die Abbildung 6.8 stellt exemplarisch die Einzelpunktanalyse für die Höhenkomponente der Station Fundata dar, die im Zeitraum 1995-2004 beobachtet wurde. Auf der x-Achse werden die Tageslösungen TL_i mit $i = [1, n]$ aufgetragen, die aus den Auswertungen von acht Kampagnen stammen. Auf der y-Achse sind die Residuen in der Einheit *mm* wiedergegeben. Die erste Zeitreihe wurde ohne Offset-Schätzung und ohne Elimination der groben Fehler (GF) aufgestellt. Nach Bereinigen der Daten - es werden drei grobe Ausreißer entfernt und drei Offsets berechnet - wird eine Höhenbewegung von $3.9 \pm 1.5 \frac{mm}{a}$ ermittelt. Allgemein zeigen die Zeitreihen, nachdem die Exzentren und Offsets geschätzt sowie die groben Fehler eliminiert wurden, für mindestens 80% der Tageslösungen folgendes auf:

- *Höhenkomponente*: die Residuen streuen im Intervall I=[-15 mm, 15 mm],
- *Lagekomponente*: die Residuen streuen im Intervall I=[-5 mm, 5 mm].

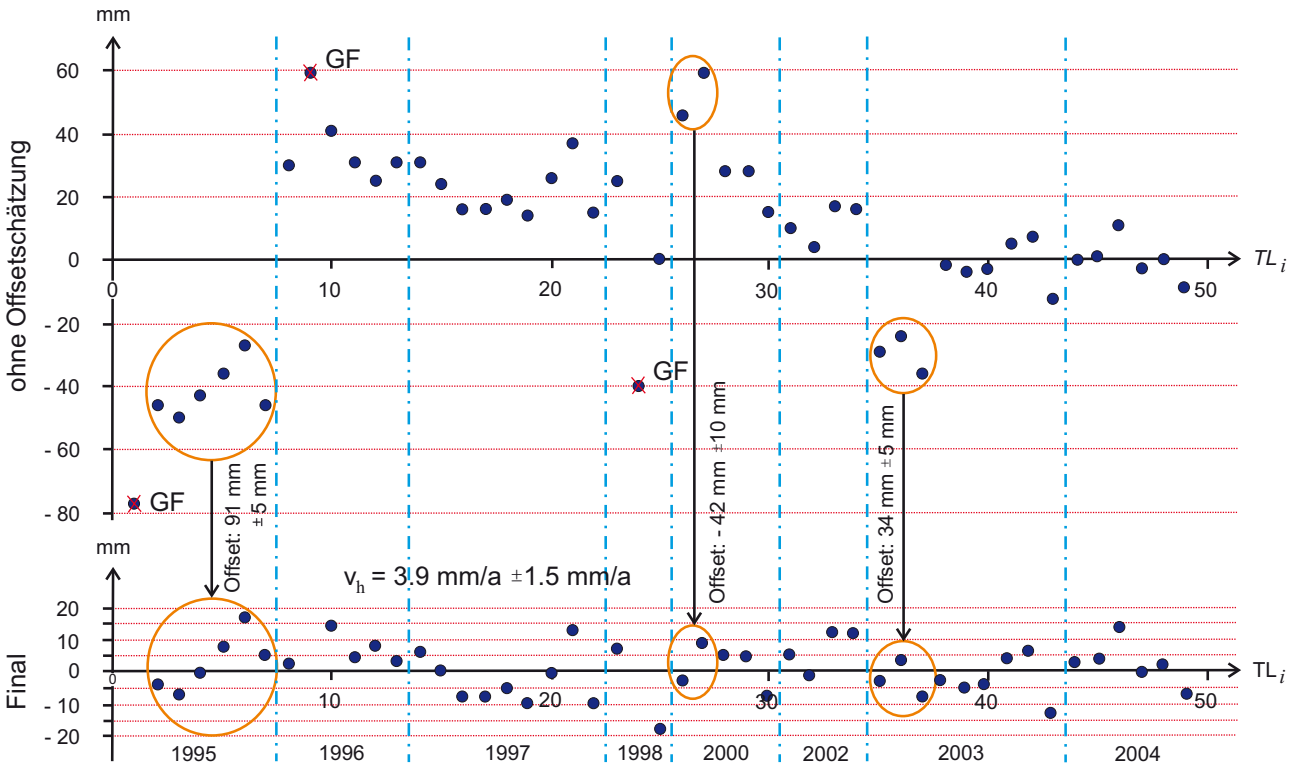


Abbildung 6.8: *Darstellung der Residuen der Höhenkomponente der Station Fundata. Die erste Zeitreihe wurde ohne Offsets und ohne Elimination der groben Fehler (GF) aufgestellt.*

Die Ursache für eine größere Streuung der Residuen eines Beobachtungsblockes in der horizontalen Komponente liegt in der Regel im exzentrischen Aufstellen des GPS-Systems. Im Bereich der vertikalen Komponente müssen hingegen die Modelle verfeinert werden, um die Ursachen des Offset-Phänomens erstens eindeutig zu ermitteln, zweitens die allgemeine Streuung der Residuen ($\pm$ 2.0 cm) zu minimieren und drittens extreme Ausreißer ($\approx$ 5.0 cm) zu vermeiden. Folgende Annahmen zur Entstehung der Offsets sind derzeit im Projekt spezifiziert:

- das Offset ist auf stationsabhängige Einflüsse zurückzuführen,
- die Empfangseigenschaften der Antenne und ihre Empfindlichkeit bezüglich der Mehrwegesignale stehen zur Diskussion,
- die Auswirkungen der troposphärischen Refraktion sind durch ein geeignetes Modell zu verringern.

6.2.4.3 Deformationsanalyse

Es wird ein Lösungsverfahren zur koordinatenbezogenen Deformationsanalyse angewendet, welches dreidimensionale Bewegungen im Deformationsnetz des SFB 461 bestimmt (Dinter, [19]). Es wird ein funktionales Modell ellipsoidischer Koordinatenbeobachtungen auf einem i-ten Stationspunkt aufgestellt:

$$\begin{pmatrix} B \\ L \\ h \end{pmatrix}_{i,t} + \begin{pmatrix} \upsilon_B \\ \upsilon_L \\ \upsilon_h \end{pmatrix}_i = \begin{pmatrix} B \\ L \\ h \end{pmatrix}_{i,t_0} + \begin{pmatrix} \omega_B \\ \omega_L \\ \mathrm{v}_h \end{pmatrix}_i (t - t_0) + \begin{pmatrix} \varepsilon_B \\ \varepsilon_L \\ o_h \end{pmatrix}_i + \begin{pmatrix} \delta B \\ \delta L \\ \delta h \end{pmatrix}_i \tag{6.2}$$

B, L, h	:	geographische Breite und Länge; ellipsoidische Höhe.
i	:	i-ter Stationspunkt mit i = 1, 2, 3, ...
t_0, t	:	Bezugsepoche, Zeitpunkt t $\neq t_0$.
$\upsilon_B, \upsilon_L, \upsilon_h$	:	Verbesserungen der Beobachtungen B, L und h.
ω_B, ω_L	:	Winkelgeschwindigkeiten, welche infinitesimale Änderungen der Breite und Länge in Abhängigkeit der Zeit beschreiben.
v_h	:	Höhenänderung in Abhängigkeit der Zeit.
$\varepsilon_B, \varepsilon_L$	:	Exzentrum eines Stationspunktes.
o_h	:	Höhen-Offset.
δB, δL, δh	:	Differenzen der Breite, Länge und Höhe beim Datumsübergang.

Das im Projekt festgelegte konventionelle geodätische Koordinatensystem (ITRF2000) und Referenzellipsoid (GRS80) legen die Parameter B, L, h bezüglich eines Raumpunktes P fest. Aufgrund der Abhängigkeit der ellipsoidischen Koordinaten sowohl von den Ellipsoiddimensionen (a: große Halbachse und f: Erdabplattung) als auch von der Lage des Ursprungs und den Achsenrichtungen des Koordinatensystems, wird der Zusammenhang beim Datums- bzw. Ellipsoidübergang zwischen den kleinen Differenzen δB, δL, δh und den Einflussgrößen $\delta a, \delta f$ als Differenzen der Ellipsoiddimensionen, δx, δy, δz als Differenzen des Verschiebungsvektors vom Ursprung des Ausgangssystems zum Zielsystem (Translation), $\delta\varepsilon_x$, $\delta\varepsilon_y$, $\delta\varepsilon_z$ als Differenzwinkel für Drehungen um die Koordinatenachsen (Rotation) und δm als Maßstabsunterschied hergestellt. Die allgemeinen Formeln sind aus der geodätischen Literatur (z.B. Heck, ([28], S. 80)) zu entnehmen.

GPS-Messungen aus Kampagnen, die in einem festgelegten Zeitabstand durchgeführt werden, bilden die Datengrundlage zur Bestimmung der relativen Plattenbewegungen in Rumänien. Die Verformungen (relative Lageänderungen) eines Körpers, die Bestandteile der Deformationsmaße sind, werden hierbei nicht berücksichtigt. Im Zuge der *„Strainanalyse“* werden die Verzerrungsmaße erfasst und quantifiziert, die als infinitesimale Deformationen (relative Längen- und Winkeländerungen) betrachtet werden. Derzeit wird ein Verfahren im Sinne der Kontinuumsmechanik entwickelt, welches das Überwachungsnetz als elastomechanischen Körper beschreibt. Neben den Verschiebungen und Rotationen von materiellen Punkten werden auch Dehnungen und Spannungen berechnet (Rawiel, [68]). Aufgrund des derzeitigen Entwicklungsstadiums stehen dem Fachinformationssystem keine Ergebnisse zur Verfügung.

Die Genauigkeitsmaße der horizontalen und vertikalen Geschwindigkeiten der diskreten Punkte, welche in Permanentstationen (IGS und ISES) und Netzpunkte (SFB, CERGOP und ISES) aufgeteilt sind, werden in Tabelle 6.2 zusammenfasst. Die aktuellen Werte erfüllen im Kontext der Qualitätsziele nicht alle Anforderungen. Die

Permanentstationen	σ_{V_B}	σ_{V_L}	σ_{V_h}
Min (mm/a)	0.2	0.3	1.4
Mittel (mm/a)	0.3	0.4	1.6
Max (mm/a)	0.4	0.6	2.7
Netzpunkte	σ_{V_B}	σ_{V_L}	σ_{V_h}
Min (mm/a)	0.5	0.5	1.5
Mittel (mm/a)	1.0	1.0	3.0
Max (mm/a)	2.0	2.0	7.5

Tabelle 6.2: *Darstellung der Genauigkeitsmaße der dreidimensionalen Plattenkinematik aufgeteilt in Permanentstationen und Netzpunkte.*

kleinsten Fehler treten bei den Permanentstationen auf, die durchgehend Messungen registrieren und sich für die Festlegung eines Bezugsrahmens aufgrund ihrer guten Stabilitätseigenschaften optimal eignen. Die maximalen Streuungen liegen in der Horizontalkomponente unter einem Millimeter pro Jahr und in der Vertikalkomponente um die geforderten 2.5 mm/a. Damit sind die Qualitätsziele für die Permanentstationen erreicht. Im Bezug auf die Netzpunkte entspricht die durchschnittliche Streuung den Qualitätszielen. Jedoch sind die Maximalwerte deutlich über dem spezifizierten Bereich. Ein wesentlicher Grund dafür liegt in dem kurzen Zeitabstand der Beobachtungen von $\leq$4 Jahren. Durch die Ausdehnung des Netzes nach Westen und Südosten und der Netzverdichtung auf der Moesischen Plattform und in der Vrancea-Zone ist die Anzahl der GPS-Stationen in den letzten 4 Jahren verdoppelt worden. Weitere Kampagnen und neue Modelle zur Höhenbestimmung werden signifikante Verbesserungen in den Genauigkeitsangaben bringen.

Die Signifikanz ist ein statistischer Begriff für das Maß an Wahrscheinlichkeit, mit der eine Aussage die Wirklichkeit trifft. Im Kontext der Bestimmung von Plattenbewegungen ist es von entscheidender Bedeutung folgende Verteilungsaussagen unter Angabe der Testgröße $\Theta_{(1-\frac{\alpha}{2})} = \frac{\mathrm{V}}{\sigma_\mathrm{v}} \sim N(0,1)$ und der Wahrscheinlichkeit $P(\%) = 1-\alpha = 99\%$ bei einem beidseitigen Test nachzuweisen bzw. zu verwerfen (Niemeier, [61]):

1. Nullhypothese H_0: $\mathrm{v} \sim N_0(\mu_0 = 0, \sigma_0^2) \Leftrightarrow$ es liegt keine signifikante Geschwindigkeit vor.
2. Alternativhypothese H_A: $\mathrm{v} \sim N_A(\mu_A \neq 0, \sigma_A^2) \Leftrightarrow$ es liegt eine signifikante Geschwindigkeit vor.
3. Signifikanzschwelle $\Theta_{0.995}$: keine eindeutige Entscheidung möglich, falls $\mathrm{v} \approx \Theta_{0.995} \cdot \sigma_\mathrm{v}$.

In der nachfolgenden Tabelle sind die Geschwindigkeitskomponenten getrennt nach Lage und Höhe aufgelistet. Dabei wird die Anzahl der diskreten Punkte in Prozent angegeben, welche die oben genannten Annahmen erfüllen. Es ist festzustellen, dass einige Lösungen im kritischen Bereich liegen, das heißt der Abstand des Geschwindigkeitsvektors entspricht etwa dem Abstand zur Signifikanzschwelle, und es kann keine eindeutige Aussage bezüglich der Stationsgeschwindigkeiten getroffen werden. Deshalb werden weitere Kampagnen zur Klärung der dritten Aussage *„keine eindeutige Entscheidung möglich“* beitragen, da die Genauigkeit zur Bestimmung von linearen Bewegungen für Stationen neueren Datums durch Hinzunahme von aktuellen Beobachtungsdaten und Ausdehnung des Zeitrahmens gesteigert wird.

Geschwindigkeitskomponente	H_0	H_A	$\Theta_{(1-\frac{\alpha}{2})}$
horizontal	20 %	60 %	20 %
vertikal	15 %	42.5 %	42.5 %

6.2.4.4 Schwachformanalyse

Die Schwachformanalyse eines Deformationsnetzes basiert auf der Spektralzerlegung der Varianzkovarianzmatrix (VKM). Analog zu elastomechanischen Systemen bestimmen dabei die größten Eigenwerte die dominanten Eigenformen des Systems und zeigen den Trend eines Netzes auf, seine Geometrie zu verlassen. Dies führt zu dem Problem, die netzeigenen Charakteristika, die sich als *Schwachformen* beziehungsweise als *scheinbare Deformationen* präsentieren, von den tatsächlichen Deformationen des Überwachungsnetzes zu unterscheiden. Bereits in der Entwurfsphase des Beobachtungsdesigns müssen die Effekte der dominanten Eigenformen zur Erhöhung der Netzsteifigkeit und zur besseren Trennbarkeit von Schwachformen und Deformationen berücksichtigt werden. Unter Verwendung der spektralen Optimierung von Deformationsnetzen kann das Verhalten des Netzes bei bestimmten Beobachtungskonstellationen analysiert werden (Schmitt, [72]). Das Lösungsverfahren einer freien Netzausgleichung mittels des Gauss-Markov-Modells liefert folgende Kofaktormatrix $Q_{\hat{x}\hat{x}}$ des unbekannten Koordinatenvektors ($\hat{\mathbf{x}}$) als Inverse der Normalgleichungsmatrix N:

$$Q_{\hat{x}\hat{x}} = N^{-} = (A^T P A)^{-} \tag{6.3}$$

A	:	Designmatrix $A \in \mathbb{R}^{(m \times n)}$ mit $m > n$.
n	:	Koordinatenunbekannten.
m	:	GPS-Beobachtungen (B, L, h).
P	:	Gewichtsmatrix $P \in \mathbb{R}^{(m \times m)}$ der Beobachtungen.
N	:	Normalgleichungsmatrix $N \in \mathbb{R}^{(n \times n)}$ der Unbekannten.
$C_{\hat{x}\hat{x}}$	:	VKM $C_{\hat{x}\hat{x}} \in \mathbb{R}^{(n \times n)} = \sigma_0^2 Q_{\hat{x}\hat{x}}$ ist symmetrisch.
σ_0	:	A priori Varianzfaktor.

Bevor die Eigenwertzerlegung durchgeführt werden kann, müssen im Zuge der freien Netzausgleichung alle Parameter (z.B. Orientierungsunbekannten, Maßstabsfaktor, ...) abgesehen von den Koordinatenunbekannten eliminiert werden. Die VKM $C_{\hat{x}\hat{x}}$ spiegelt die Genauigkeitssituation im Überwachungsnetz wieder. Zur Betrachtung weiterer Qualitätsmerkmale wird zur Lösung des Eigenwertproblems ein Verfahren zur *„speziellen Eigenwertzerlegung“* angewandt, welches die Eigenwerte λ_i und -vektoren η_i bestimmt. Es wird vorausgesetzt, dass die Kofaktormatrix $Q_{\hat{x}\hat{x}}$ quadratisch ist. Dann gilt (Höpcke, [31], S. 33-35):

$$(Q_{\hat{x}\hat{x}} - \lambda_i E) \cdot \eta_i = \mathbf{o} \tag{6.4}$$

Es hat genau n-Lösungen ($\eta_i \neq o$), wenn gilt: $det(Q_{\hat{x}\hat{x}} - \lambda_i E) = 0$ mit $i = 1, 2, ..., n$. Die Spektralmatrix Λ besitzt keinen Rangdefekt, wenn $Q_{\hat{x}\hat{x}}$ regulär ist. Es gilt folgendes:

$$M^T Q_{\hat{x}\hat{x}} M = \Lambda = diag(\lambda_i) \tag{6.5}$$

E	:	Einheitsmatrix $E \in \mathbb{R}^{n \times n}$.
λ_i	:	Die nach ihrem Betrage geordneten Eigenwerte mit $i = 1, 2, ..., n$.
η_i	:	Dazugehörige normierte Eigenvektoren mit $i = 1, 2, ..., n$.
M	:	Modalmatrix $M = [\eta_1, ..., \eta_n] \in \mathbb{R}^{n \times n}$ mit $M^T M = E$ und $M^{-} = M^T$ (M ist orthonormal).
Λ	:	Spektralmatrix $\Lambda \in \mathbb{R}^{n \times n}$; es gilt: $[\lambda_1, \lambda_2, ..., \lambda_n] \cdot E = \Lambda$

Es gilt für die Darstellung des Schwachform-Vektors ϑ_i (Produkt aus Skalar des Eigenwertes und dazugehörigem Eigenvektor) folgendes:

$$\vartheta_i = (\sqrt{\lambda_i}\,\sigma_0) \cdot \eta_i \tag{6.6}$$

Der Entwurf eines Beobachtungsplans hängt von verschiedenen Faktoren ab. Die optimale Messanordung von GPS-Netzen, deren Netzpunkte homogen über das Kontrollgebiet verteilt sind, besteht aus der gleichzeitigen Besetzung mit GPS-Systemen. Dem gegenüber steht die Wirtschaftlichkeit. Es müssen ausreichend GPS-Systeme, Messteams, Aufsichtsteams und finanzielle Ressourcen zur Durchführung einer Kampagne in Rumänien zur Verfügung stehen. Daher ist das Ziel, ein Konzept zur optimalen Besetzung unter Reduzierung von Schwachform-Einflüssen zu entwickeln. Aus vorherigen Untersuchungen ist bekannt, dass die Hauptschwachform der vertikalen Komponente in verketteten Netzen einer Kippung der allgemein dominierenden

Schwachform freier Höhennetze entspricht (Jäger/Kaltenbach, [44]). Die Abbildung 6.9 stellt die Auswirkungen der dominanten Eigenformen verketteter Teilnetze schematisch dar, die aufgrund logistischer Engpässe in den Kampagnen 1997, 1998 und 2000 in drei Blöcken (A, B und C) hintereinander zu besetzen waren.

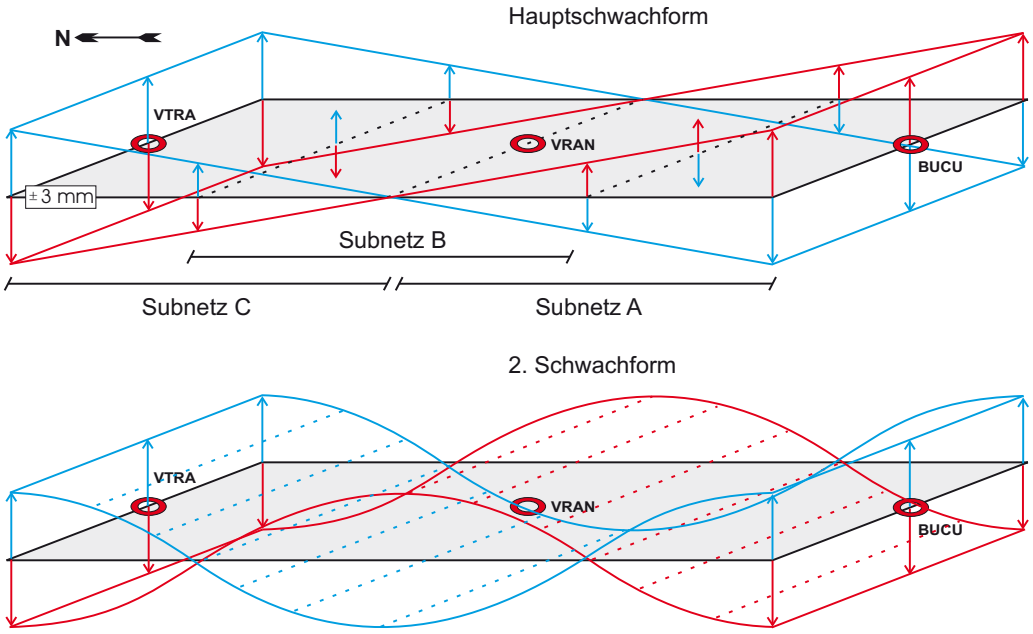


Abbildung 6.9: *Schematische Darstellung der Hauptschwachform und der zweiten Schwachform im Deformationsnetz 1997, 1998 und 2000 - Höhenkomponente*

Die Schwachformanalyse 2003 repräsentiert exemplarisch die Ergebnisse für die Kampagnen nach 2000. Es werden wiederum drei Beobachtungsblöcke gebildet. Diesmal aber sind die zu besetzenden Stationen über das gesamte Überwachungsgebiet zufällig verteilt. Damit wird eine bestmögliche Streuung und Netzkonfiguration unter den Gesichtspunkten Netzsteifigkeit und Wirtschaftlichkeit erreicht. Die Abbildung 6.10 zeigt das Spektrum der Eigenwerte λ_i, sortiert nach ihren Beträgen und getrennt nach Lage und Höhe, wobei das Eigenwertspektrum der Lagekomponente nicht vollständig dargestellt wird. Die maximalen Beträge der Eigenwerte liegen für die horizontale Komponente um 4.5 mm und für die vertikale Komponente zwischen 3.5 und 4 mm. Gleichzeitig entsprechen diese Skalare in Kombination mit den Eigenvektoren den dominanten Schwachformen des GPS-Netzes.

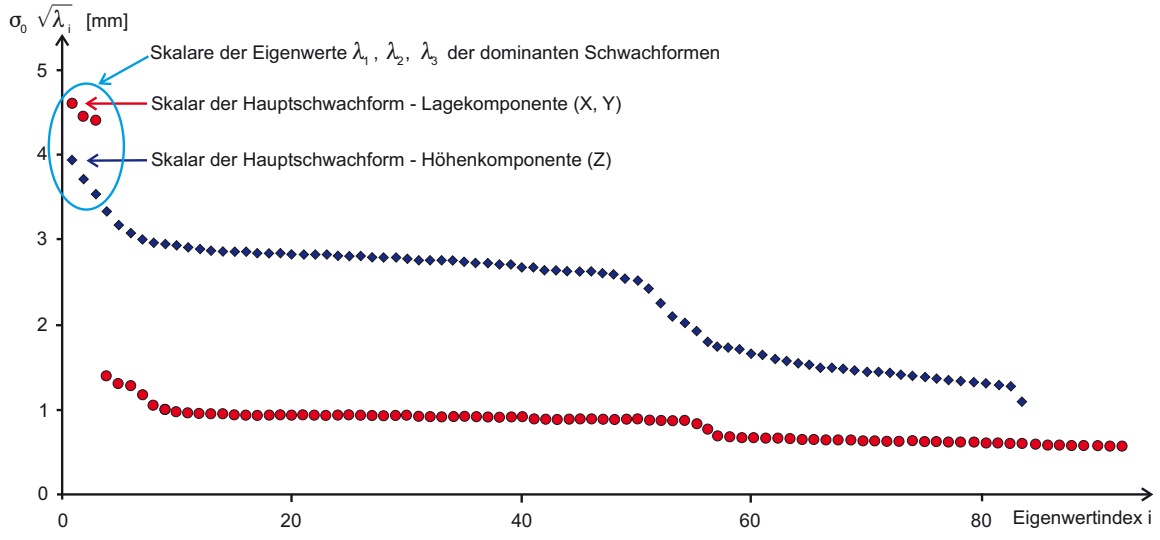


Abbildung 6.10: *Darstellung des Eigenwertspektrums 2003 der Höhen- und Lagekomponente.*

Die Gruppen ungefähr gleichgroß auftretender dominanter Schwachformen in Abbildung 6.11 entsprechen dem Beobachtungsdesign. Jedoch sind die Einflüsse nicht mehr so stark ausgeprägt ($\leq 1.0\,mm$) wie in den ersten drei Analysen 97, 98 und 2000 ($\leq 3.0\,mm$). Besonders augenfällig ist die Tatsache, dass keine netzeigenen Charakteristika (Kippung oder wellenförmige Eigenform) mehr nachzuweisen sind. Daraus kann gefolgert werden, dass die Einflüsse dominanter Schwachformen die Ergebnisse aus den Deformationsanalysen nicht signifikant verfälschen. Das Beobachtungsdesign wird für die weiteren SFB-Kampagnen beibehalten.

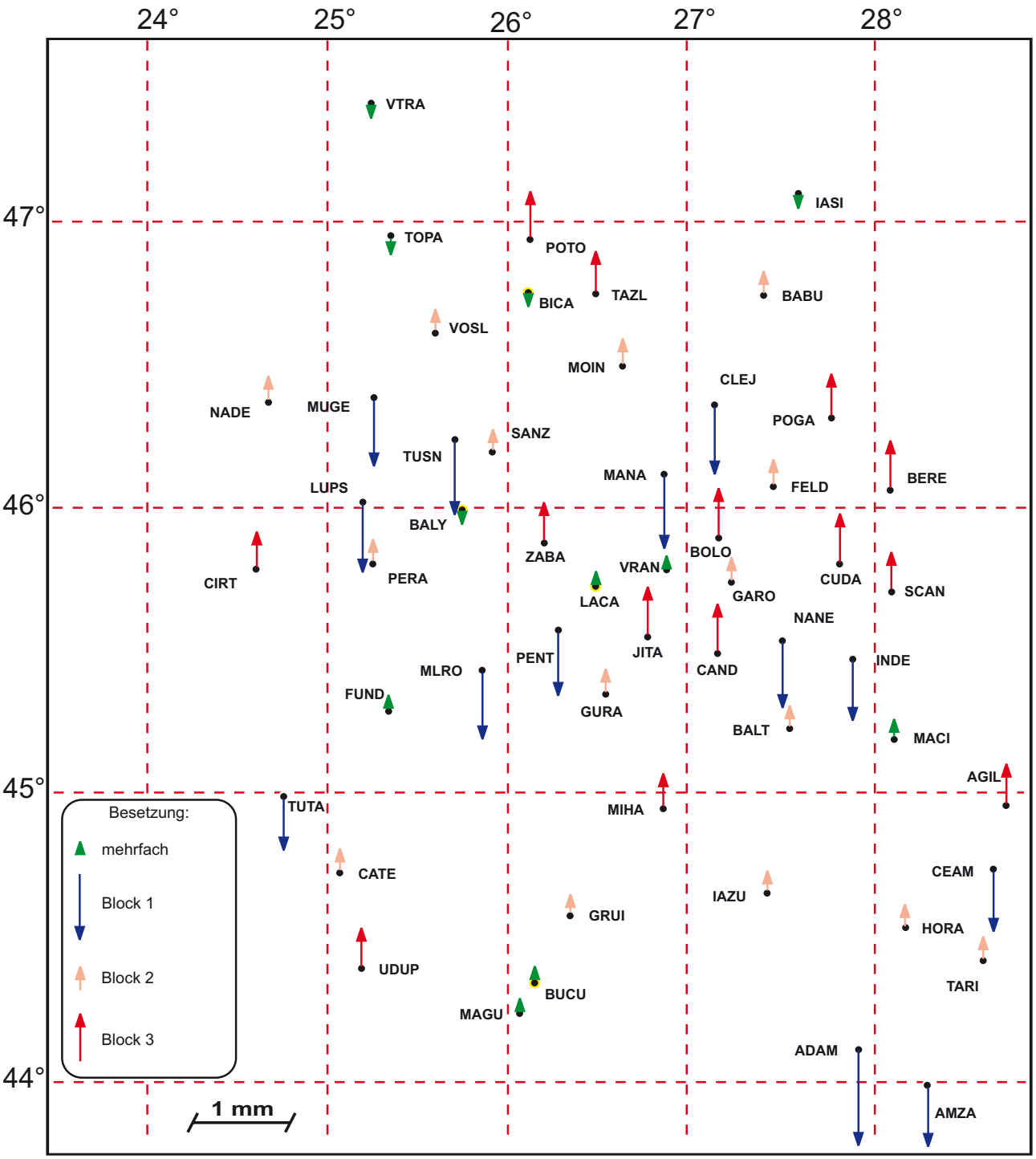


Abbildung 6.11: *Hauptschwachform 2003 - Höhenkomponente*

Im Bezug auf die Lagekomponente stellt die Hauptschwachform in Abbildung 6.12 ein Pulsieren des Netzes als Eigenform dar, welches einerseits als Maßstabseinfluss interpretiert (Dinter/Schmitt, [21]) und andererseits die Auswirkung unter Einführung eines Maßstabsfaktors in der freien Netzausgleichung verringert werden kann. Dann würde die zweite dominante Schwachform die Position der Hauptschwachform einnehmen. Die nachfolgenden dominanten Schwachformen verhalten sich ähnlich wie die vorangegangenen Schwachformen der Höhenkomponente. Sie spiegeln das Beobachtungsdesign wieder und es können keine weiteren typischen Eigenformen abgeleitet werden. Der maximale Einfluss der Hauptschwachformen beträgt 1.5 mm und ist mit den Ergebnissen aus den ersten drei Analysen 97, 98 und 2000 vergleichbar.

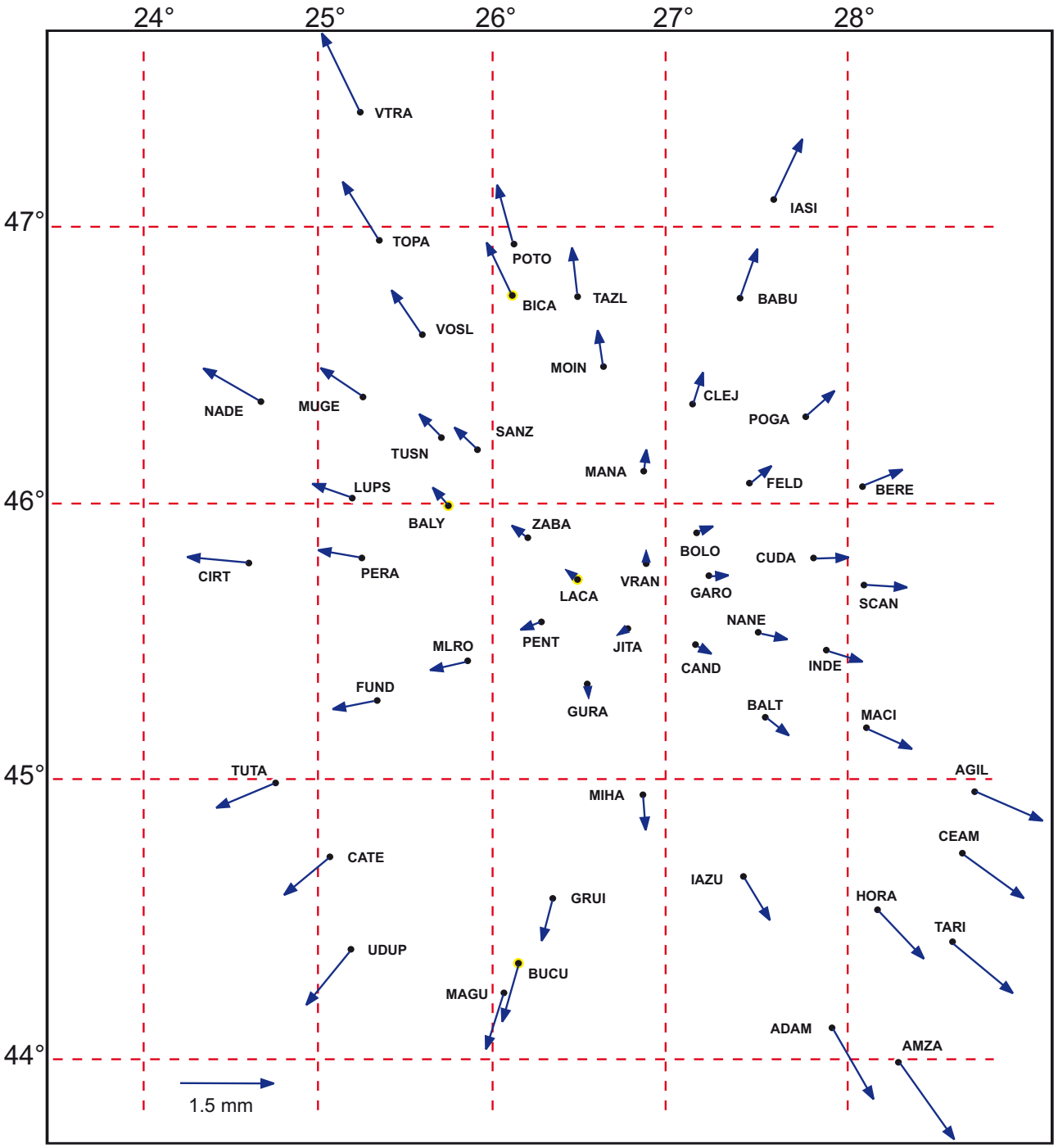


Abbildung 6.12: *Hauptschwachform 2003 - Lagekomponente*

Kapitel 7

FIS-Applikationsentwicklung

Das Fachinformationssystem wurde entwickelt, um sowohl den Mitarbeitern ein Hilfswerkzeug zur Projektplanung, -durchführung und -analyse zur Verfügung zu stellen als auch den Projektpartnern ein Informationssystem zu geodätischen Aspekten (Kap. 6) zur Weiterverarbeitung bzw. Integration der Ergebnisse in ihren Projektanforderungen und zur Validierung parallel verlaufender Analysen zur dreidimensionalen Plattenkinematik in Rumänien bereitzustellen. Dabei sollten folgende Gesichtspunkte berücksichtigt werden:

- Daten bzw. Informationen schnell, verständlich und zur Weiterverarbeitung geeignet aufbereiten. Lösung: internetbasiert, standardisierte Ausgabe mittels XML (Kap. 5.1), grafische Darstellung und ausführliche Dokumentation.
- Sicherheitsrelevante Aspekte beachten. Lösung: Drei-Schichten-Architektur mit Sicherheitsrichtlinien (Kap. 3.1.3, 3.2.7 und 7.2).
- Standardisierte und sichere Datenverwaltung zur Aufbereitung, Übergabe und Steuerung der Daten für projektbezogene Prozesse (z.B. Qualitäts- und Deformationsanalysen). Lösung: Einsatz eines objektrelationalen Datenbank-Managementsystems (Kap. 2).

Zur Umsetzung der applikationsabhängigen Komponenten wurde eine grafische Benutzeroberfläche (GUI) mittels Java Swing Klassen entwickelt (Abb. 7.1). Der Aufbau des GUI sieht drei Bereiche vor: Norden (Menüleiste), Westen (Aktionssteuerung) und Zentrum (Präsentation der Anwendung).

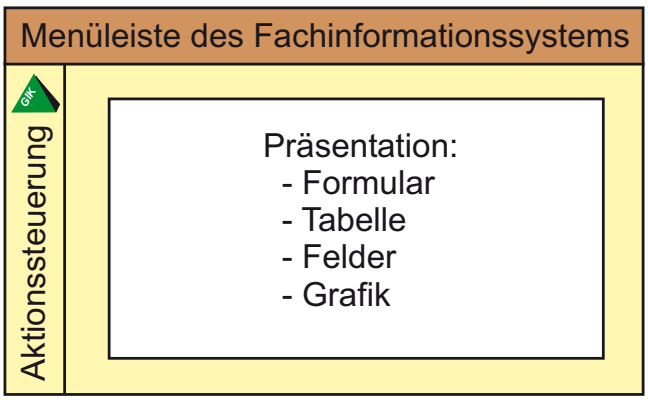


Abbildung 7.1: *Aufbau der grafischen Benutzeroberfläche im FIS*

7.1 FIS-Anforderungskatalog

Im Rahmen der Erstellung des FIS-Konzeptes wurde ein Anforderungskatalog formuliert, der in zwei Hauptbereiche gegliedert ist:

1. Start-Menüleiste: allgemeine Anwendungen (FIS-Anmeldung, Server aktiv/deaktiv, FIS-Information).
2. FIS-Menüleiste: nach erfolgreicher Anmeldung werden fachspezifische Anwendungen sichtbar (Projektdokumentation, Kampagnenverlauf, Analysen, Feldbücher, Datenbankzugriff, etc.).

Die Start-Menüleiste (Abb. 7.2) erfüllt die Bedingung, dass nur autorisierte Benutzer Zugang zum Fachinformationssystem haben. Über den User *„public“* wird ein allgemeiner Benutzer mit eingeschränkten Berechtigungen (Darstellung der Datensichten, aber kein direkter DB-Zugriff auf Tabellen) festgelegt, der sich über den *„Login“*-Knopf mit dem FIS verbindet. Zusätzlich kann über den *„Host“*-Knopf überprüft werden, ob der Datenbank- und Web-Server aktiv sind. Allgemeine Informationen über das FIS erhält man über den *„Info“*-Knopf.

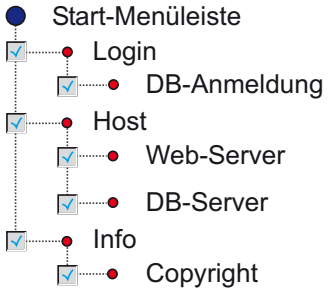


Abbildung 7.2: *Darstellung der Anforderungsliste zur Gestaltung der Start-Menüleiste und deren Anwendungen.*

Die FIS-Menüleiste besitzt Knöpfe als Schaltflächen, die durch Anklicken mittels des Maus-Cursors eine Aktion ausführen. Die in Abbildung 7.3 dargestellte Anforderungsliste wurde übernommen, um die geodätischen Aspekte des Forschungsprojektes umzusetzen. Weitere Ereignisse werden durch Bedienen der aktionsgesteuerten Komponenten im Westbereich des GUI ausgelöst.

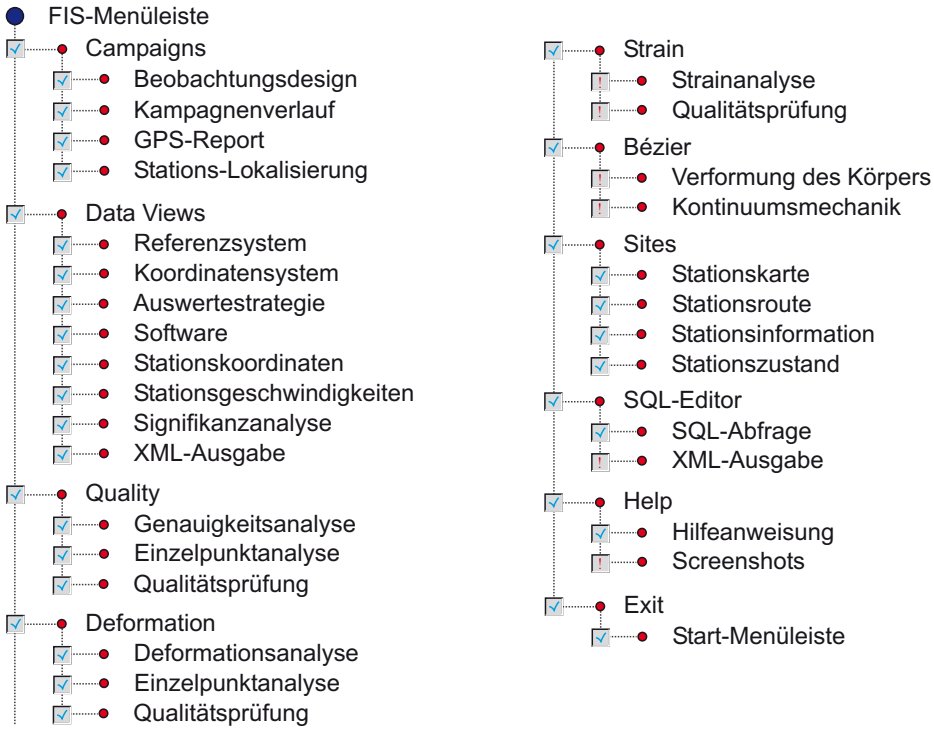


Abbildung 7.3: *Darstellung der Anforderungsliste zur Gestaltung der FIS-Menüleiste und deren Anwendungen. Die Check-Liste zeigt an, welche Applikationen implementiert wurden (entweder* $\surd$ *= ja oder ! = nein).*

7.2 FIS-Zugriffskontrolle

Zentraler Bestandteil der FIS-Architektur ist die Sicherstellung der verschlüsselten Datenübertragung, der richtigen Lokalisierung des JApplets, der Authentisierung und Autorisierung der FIS-Nutzer und der korrekten Datenbankverbindung (Abb. 7.4).

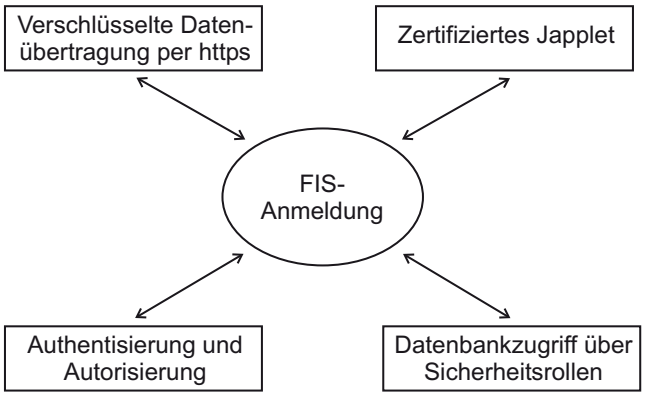


Abbildung 7.4: *Darstellung des FIS-Anmeldungskonzeptes.*

Das FIS-Sicherheitskonzept schließt dabei ein, dass Administratoren und Hauptbenutzer, die über einen vollen Zugriff verfügen, von den allgemeinen Benutzern unterschieden werden. Deshalb werden Sicherheitsrollen im Datenbankverwaltungssystem festgelegt, die dieser Anforderung gerecht werden. Der Applikationsserver übergibt durch Aktivieren der DBAccessBean (Anh. C.1) den Benutzernamen und das Passwort dem Datenbank-Managementsystem, welches die Sicherheitsdienste steuert und für die Datenfreigabe verantwortlich ist. Nach erfolgreicher Anmeldung des Benutzers werden Container-gesteuerte Transaktionen (Anh. C.2) für anschließende Datenbankanfragen ausgeführt. Die FIS-Applikation unterscheidet ebenfalls allgemeine Benutzer von den übergeordneten Nutzern, indem der Zugriff auf den SQL-Editor verweigert wird. Somit wird sichergestellt, dass keine unerlaubten DB-Anweisungen ausgeführt werden. Die in der FIS-Applikation implementierte Methode *„DBAccessBean()“* (Anh. C.3), welche über ein Interface mit der DBAccessBean des JBoss-Servers kommuniziert, übergibt den Benutzernamen und das verschlüsselte Passwort (Abb. 7.5).

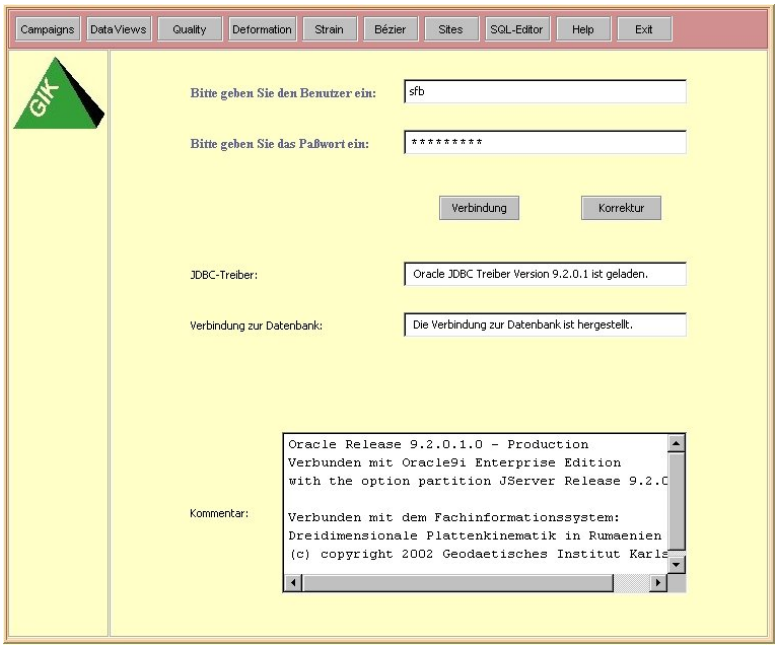


Abbildung 7.5: *Erfolgreiche FIS-Datenbankanmeldung beim Oracle 9i-Datenbankserver.*

Nach erfolgter DB-Anmeldung stehen die fachspezifischen Anwendungen zur Verfügung. Durch das zertifizierte JApplet wird dem Nutzer die Entscheidung überlassen, die FIS-Applikation zu laden und einen Datentransfer zu zulassen (Kap. 3.2.6.3). Die verschlüsselte Datenübertragung im Netz wird durch das vom Rechenzentrum ausgestellte X.509-Zertifikat und dem Web-Server garantiert, welcher das SSL-Protokoll als Übertragungsprotokoll kontrolliert (Kap. 3.2.7.2).

7.3 FIS-Monitoring

Der Applikationsserver stellt ein Werkzeug zur Überwachung der FIS-Transaktionen zur Verfügung, um das Erzeugen, Finden, Aktivieren und Beenden der FIS-Session-Beans zu protokollieren, (Abb. 7.6). Nach erfolgter Datenbankanbindung werden die Transaktionen (in der Regel Datenübertragungen vom DB-Server) am Monitor angezeigt. Bei Überlastung des AS kann der Administrator eingreifen, indem entweder der Server neu gestartet wird oder die in der JMX-Konsole als MBeans erscheinenden SessionBeans zerstört werden (Kap. 3.2.2). Somit wird der vollständige Lebenszyklus einer Enterprise Java Bean vom JBoss-Server gesteuert (Kap. 4.1.4).

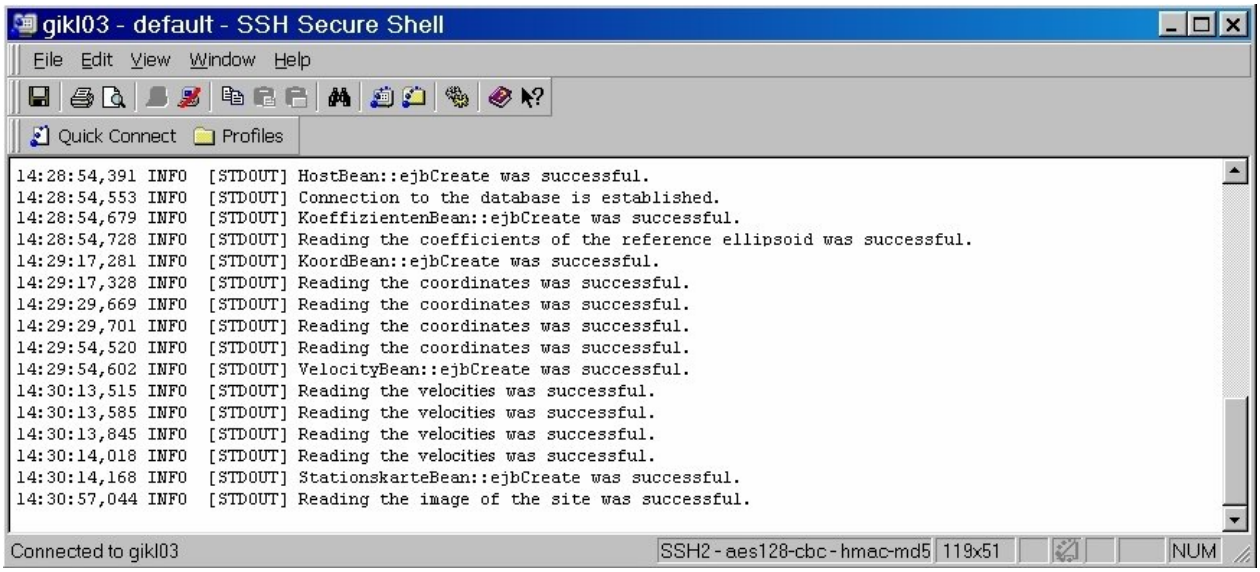


Abbildung 7.6: *Überwachung der FIS-Zugriffe mittels des JBoss-Monitors.*

7.4 FIS-Kampagne

Neben der Übersicht der GPS-Kampagnen (Abb. 7.7), deren Entwicklung seit Bestehen des Sonderforschungsbereiches 461 *„Starkbeben: von geowissenschaftlichen Grundlagen zu Ingeniermaßnahmen“* in zeitlicher Reihenfolge dokumentiert werden, sind folgende Funktionen entwickelt worden:

- Ausdehnung und Verdichtung des Deformationsnetzes,
- Dokumentation der GPS-Beobachtungstage,
- Berücksichtigung der verwendeten Beobachtungsdesigns,
- Beschreibung der GPS-System-Eigenschaften,
- Darlegung der Konfigurationsmerkmale,

- Definition der Offsets,
- Anzeige des Kartenmaßstabes,
- Protokollierung der Kampagne.

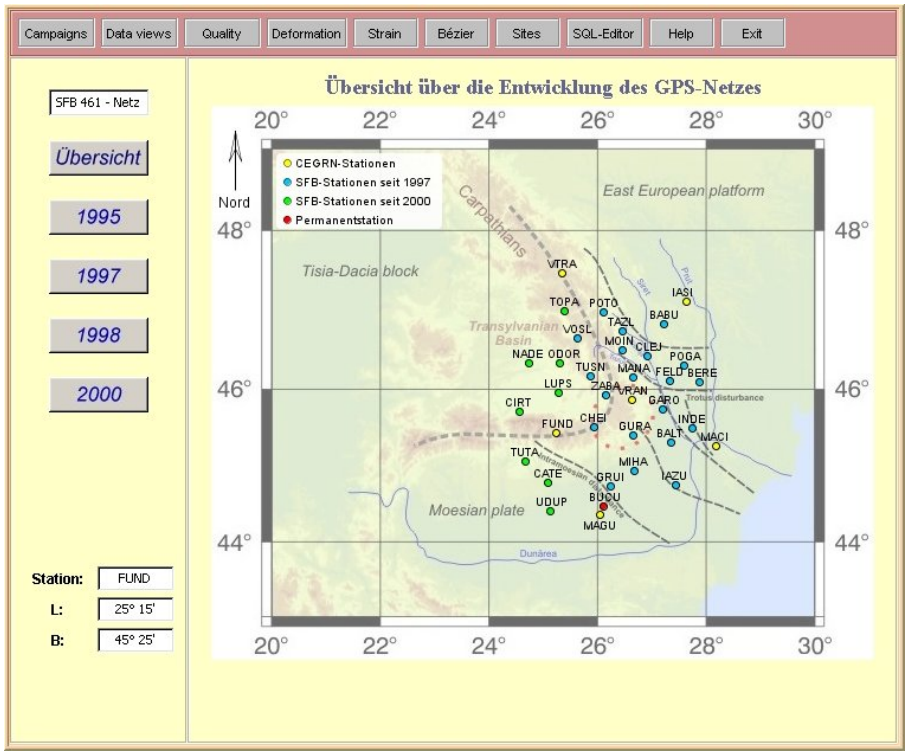


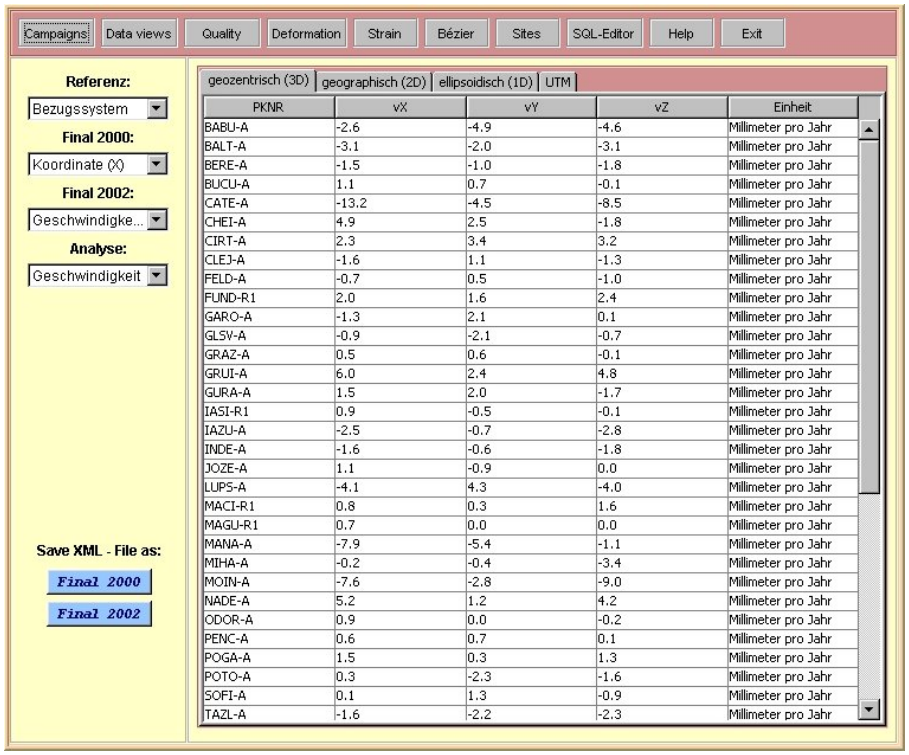
Abbildung 7.7: *Auszug aus der FIS-Applikation „Campaigns“.*

Im Rahmen der Qualitätsprüfung werden Qualitätsmerkmale aus den angezeigten Informationen abgeleitet. Insbesondere sind folgende Faktoren zu berücksichtigen, um die Qualität der Ergebnisse aus den Deformationsanalysen zuverlässig zu bestimmen:

1. die Dauer einer GPS-Beobachtung → optimale Beobachtung: 24 Stunden,
2. die Besetzung der Station → optimale Besetzung: permanent,
3. der Einsatz des GPS-Systems → optimaler Einsatz: neueste Generation und einheitliches Model in einer Kampagne verwenden,
4. präziser Aufbau des GPS-Systems → fachliche Betreuung der Messteams durch Organisationsteams (Nachweis über Photos),
5. ausführliche Dokumentation → Angabe der Offsets, Konfiguration des Systems, usw. .

7.5 FIS-Datensichten

Die Datensichten werden im Datenmodell festgelegt, um ausführliche Informationen tabellarisch den Nutzern zur Verfügung zu stellen (Abb. 7.8). Dabei wurden nicht nur die Auswertungen (Koordinaten, Geschwindigkeiten, Deformationen, Genauigkeiten und Maße zur inneren und äußerlichen Zuverlässigkeit) zur Beurteilung der Qualität des Überwachungsnetzes präsentiert, sondern auch ausführliche Informationen zum Gebrauch des Koordinaten- und Referenzsystems sowie der Auswertestrategie und der Software vermittelt. Damit soll die



PKNR	vX	vY	vZ	Einheit
BABU-A	-2.6	-4.9	-4.6	Millimeter pro Jahr
BALT-A	-3.1	-2.0	-3.1	Millimeter pro Jahr
BERE-A	-1.5	-1.0	-1.8	Millimeter pro Jahr
BUCU-A	1.1	0.7	-0.1	Millimeter pro Jahr
CATE-A	-13.2	-4.5	-8.5	Millimeter pro Jahr
CHEI-A	4.9	2.5	-1.8	Millimeter pro Jahr
CIRT-A	2.3	3.4	3.2	Millimeter pro Jahr
CLEJ-A	-1.6	1.1	-1.3	Millimeter pro Jahr
FELD-A	-0.7	0.5	-1.0	Millimeter pro Jahr
FUND-R1	2.0	1.6	2.4	Millimeter pro Jahr
GARO-A	-1.3	2.1	0.1	Millimeter pro Jahr
GLSV-A	-0.9	-2.1	-0.7	Millimeter pro Jahr
GRAZ-A	0.5	0.6	-0.1	Millimeter pro Jahr
GRUI-A	6.0	2.4	4.8	Millimeter pro Jahr
GURA-A	1.5	2.0	-1.7	Millimeter pro Jahr
IASI-R1	0.9	-0.5	-0.1	Millimeter pro Jahr
IAZU-A	-2.5	-0.7	-2.8	Millimeter pro Jahr
INDE-A	-1.6	-0.6	-1.8	Millimeter pro Jahr
JOZE-A	1.1	-0.9	0.0	Millimeter pro Jahr
LUPS-A	-4.1	4.3	-4.0	Millimeter pro Jahr
MACI-R1	0.8	0.3	1.6	Millimeter pro Jahr
MAGU-R1	0.7	0.0	0.0	Millimeter pro Jahr
MANA-A	-7.9	-5.4	-1.1	Millimeter pro Jahr
MIHA-A	-0.2	-0.4	-3.4	Millimeter pro Jahr
MOIN-A	-7.6	-2.8	-9.0	Millimeter pro Jahr
NADE-A	5.2	1.2	4.2	Millimeter pro Jahr
ODOR-A	0.9	0.0	-0.2	Millimeter pro Jahr
PENC-A	0.6	0.7	0.1	Millimeter pro Jahr
POGA-A	1.5	0.3	1.3	Millimeter pro Jahr
POTO-A	0.3	-2.3	-1.6	Millimeter pro Jahr
SOFI-A	0.1	1.3	-0.9	Millimeter pro Jahr
TAZL-A	-1.6	-2.2	-2.3	Millimeter pro Jahr

Abbildung 7.8: *Auszug aus der FIS-Applikation „Data Views“.*

Wiederverwendung der Daten in weiterführenden Projekten gewährleistet werden. Die Ausgabe der Fachinformationen erfolgt über XML-Dateien.

Die zeitlich geordnete Darstellung der Geschwindigkeiten der einzelnen Stationen ermöglicht dem Auswerter mittels der FIS-Analyse folgenden Vergleich und folgende Hypothesentests (H_0, H_A) durchzuführen:

- Gegenüberstellung der Stationsgeschwindigkeiten aus Deformationsanalysen verschiedener Zeitepochen E_i mit $i \in \mathbb{N}^* = \{1,2,3,...\}$ und die Bestimmung des Betrages der Differenz zweier Geschwindigkeitsvektoren $[\mathbf{v}_{E_i}, \mathbf{v}_{E_{i+1}}]$, um einen ersten Überblick über die Stationsbewegungen zu erhalten.
- Unter der Annahme, dass die Stationsgeschwindigkeiten gleichförmige Plattenbewegungen wiederspiegeln, gilt für den Betrag $||\Delta\mathbf{v}_i||$ aus der Differenzbildung der Geschwindigkeitsvektoren zweier benachbarter Epochen $[E_i, E_{i+1}]$ unter Berücksichtigung der Fehlertoleranz $\sigma_{\Delta\mathbf{V}_i}$ und nach ausreichender Beobachtungsdauer T_i mit $i \geq 3$ folgendes (Abb. 7.9.1):

$$||\Delta\mathbf{v}_i|| = ||\mathbf{v}_{E_{i+1}} - \mathbf{v}_{E_i}|| = 0 \tag{7.1}$$

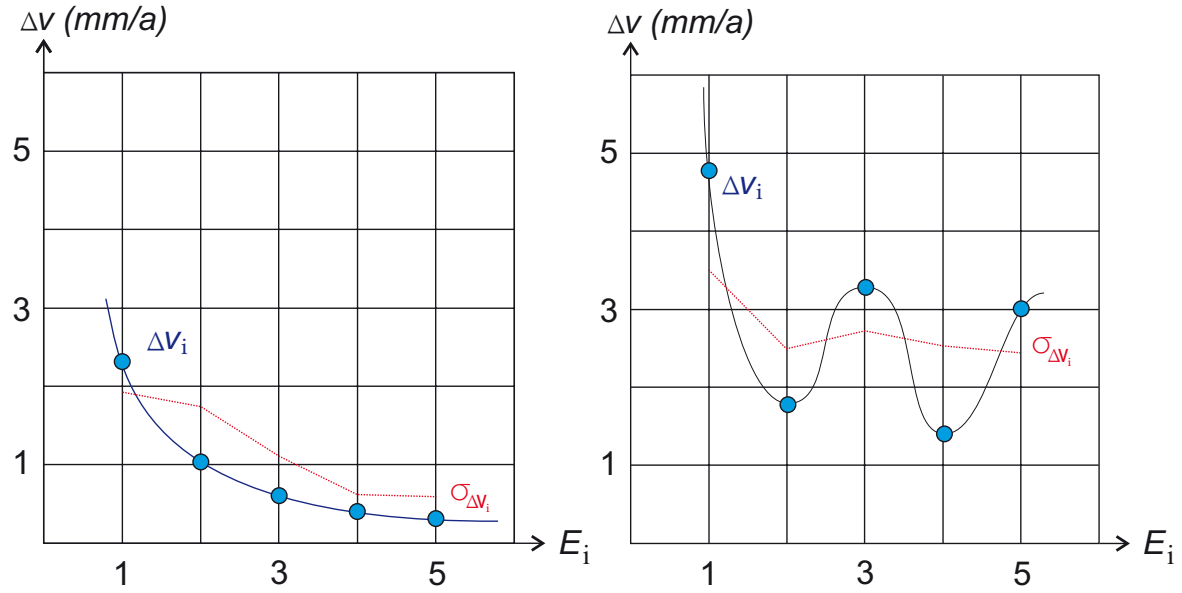
$$\sigma^2_{\Delta\mathbf{V}_i} = \sigma^2_{\mathbf{V}_{E_{i+1}}} + \sigma^2_{\mathbf{V}_{E_i}} \tag{7.2}$$

$$H_0 : ||\Delta\mathbf{v}_i|| \leq \sigma_{\Delta\mathbf{V}_i} \tag{7.3}$$

- Falls die Alternativhypothese $H_A : ||\Delta\mathbf{v}_i|| > \sigma_{\Delta\mathbf{V}_i}$ gilt, werden die Stationsbewegungen entweder durch lokale Einflüsse stark gestreut oder aufgrund atmosphärischer Störungen, grober Fehler im Beobachtungsmaterial, signifikanter Schwachformen sowie elektrischer Eigenschaften des GPS-Systems verfälscht (Abb. 7.9.2). Weitere Maßnahmen (z.B. lokale Gewichtungsfaktoren im stochastischen Modell einführen), Modellerweiterungen (z.B. Antennenoffsets, meteorologische Modelle, usw. einsetzen) oder Elimination der betroffenen Stationen sind vorzunehmen.

Die in Klammern gesetzten Jahre gelten für Stationen, die im Rahmen des *Central Europe Regional Geodynamic* Projektes 1995 aufgebaut und besetzt wurden.

Differenzvektor	:	$\Delta\mathbf{v}_1 = \mathbf{v}_{E_2} - \mathbf{v}_{E_1}$	$\Delta\mathbf{v}_2 = \mathbf{v}_{E_3} - \mathbf{v}_{E_2}$
		$\Delta\mathbf{v}_3 = \mathbf{v}_{E_4} - \mathbf{v}_{E_3}$	$\Delta\mathbf{v}_i = \mathbf{v}_{E_{i+1}} - \mathbf{v}_{E_i}$
Beobachtungsdauer	:	$T_1 = 3\,(5)$ Jahre	$T_2 = 5\,(7)$ Jahre
		$T_3 = 6\,(8)$ Jahre	$T_4 = 7\,(9)$ Jahre
Zeitepoche	:	$E_1 = 1997\,(95) - 2000$	$E_2 = 1997\,(95) - 2002$
		$E_3 = 1997\,(95) - 2003$	$E_4 = 1997\,(95) - 2004$



7.9.1: H_0: gleichförmige Stationsbewegung

7.9.2: H_A: ungleichförmige Stationsbewegung

Abbildung 7.9: *Grafische Darstellung der Geschwindigkeitsdifferenzen mittels Splinefunktion - nach aufeinander folgenden Epochen sortiert.*

7.6 FIS-Stationskontrolle

Die Eigenschaften der Applikation *Sites* sind gekennzeichnet durch folgende Punkte:

- Elektronisches Feldbuch zum Auffinden der Stationen,
- Dokumentation der Stationszustände mit folgender Zustandsbeschreibung:
  - guter oder schlechter Zustand,
  - beschädigt → Reparatur ist möglich,
  - Hauptpunkt zerstört → Exzentrum verwenden,
  - Station zerstört → Aufgabe oder neu vermarken.
- Festlegung von Qualitätskriterien zur Überprüfung der Stabilität der Stationen,
- Fortführung und Erweiterung der elektronischen Feldskizzen.

Die Stabilität und ein langfristig gesicherter guter Zustand der Station sind Voraussetzung zur Detektion von Plattenbewegungen. Daher wurden bei Vermarkung der GPS-Punkte geophysikalische Gutachten über die Bodenbeschaffenheit der Stationen erstellt. Regelmäßige Kontrollen und Beschreibungen der Stationszustände stellen die geeignete Nutzung der GPS-Messungen zur Bestimmung rezenter Plattenkinematik in Rumänien sicher. Die Stationskarte „Vrancioaia“ beschreibt den Zugang zur Station, die durch einen Hauptpunkt R1, einem Exzentrum R2 und deren Versicherungen gekennzeichnet ist (Abb. 7.10). Stationsinformationen, Wegbeschreibungen, Stationsaufnahmen, Stationszustand, Dokumentation und Darstellung der Vermarkung sowie das geophysikalische Gutachten vervollständigen die Funktionalität „FIS-Stationskontrolle“.

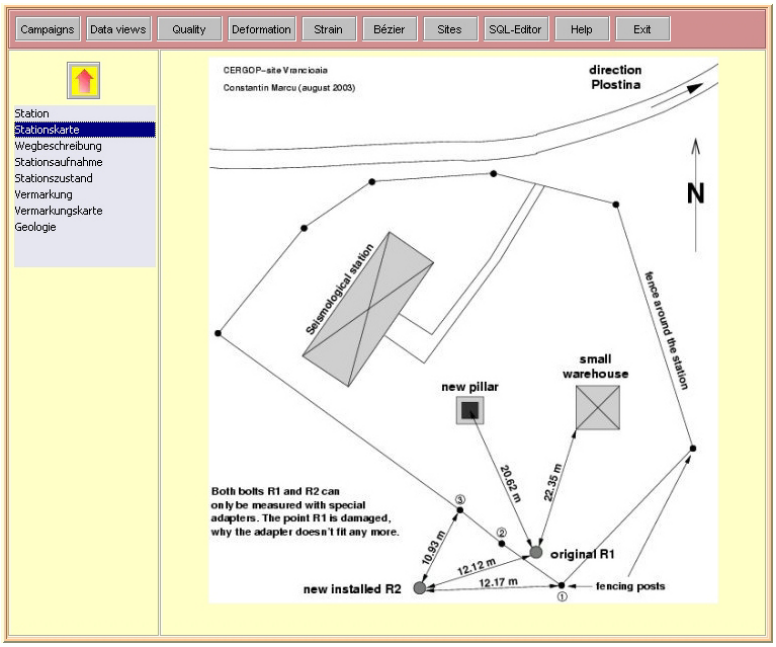


Abbildung 7.10: *Auszug aus der FIS-Applikation „Sites“.*

7.7 Validierung der Analysen

Wesentliche Aufgabe der Projektmitarbeiter ist die Erzeugung genauer und zuverlässiger Deformationsparameter. Dafür wurde eine Reihe von Analysen mit unterschiedlichen Auswertestrategien, Modellen und Softwareprogrammen erstellt, um *grobe* und *systematische* Fehler auszuschließen sowie *zufällige* Fehler statistisch korrekt zu erfassen. Die Gegenüberstellung und Validierung der Ergebnisse wurde in der Applikation „Deformation“ realisiert (Abb. 7.11).

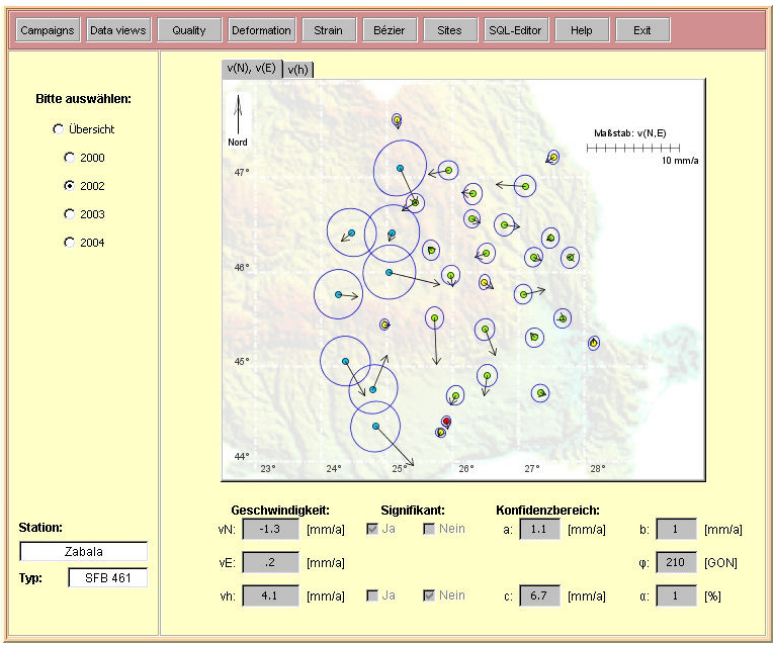


Abbildung 7.11: *Auszug aus der FIS-Applikation „Deformation“.*

7.8 SQL-Editor

Den Administratoren des FIS steht die Funktionalität „SQL-Editor“ zur Verfügung. Mittels einer select-Anweisungen wird direkt auf die Tabellen der Datenbank zugegriffen. Durch Drücken des Start-Knopfes wird die SQL-Anweisung ausgeführt und die Informationen erscheinen tabellarisch im SQL-Editor. Da die verwendete JDBC^{API} den SQL/92-Standard voll unterstützt, werden objektrelationale Datenobjekte (referenzierte und verschachtelte Tabellen) mit ihren Aliasnamen aufgelistet (Abb. 7.12). Die im Datenmodell implementierten Objektklassen (in Anhang A.1 und Abbildung A.1 vollständig dargestellt) können selektiert werden. Funktionen zur Manipulation der Daten stehen ebenfalls zur Verfügung.

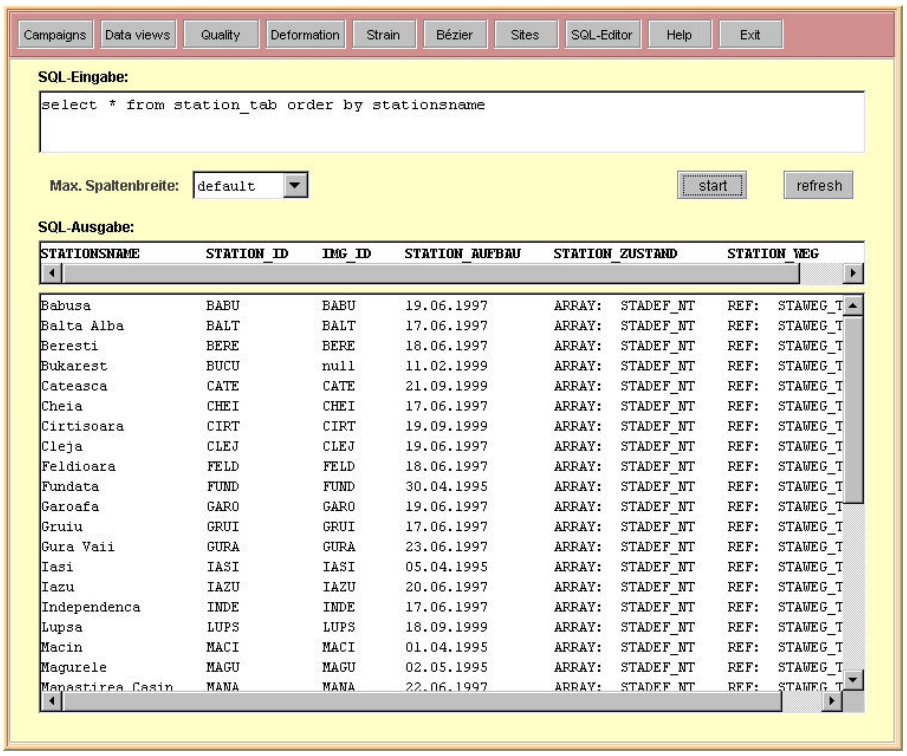


Abbildung 7.12: *Auszug aus der FIS-Applikation „SQL Editor“.*

Kapitel 8

Zusammenfassung

In den vorhergehenden Kapiteln wurde die Systemarchitektur des Fachinformationssystems „Dreidimensionale Plattenkinematik in Rumänien“ ausführlich dargestellt. Dabei wurde eine Drei-Schichten-Architektur aufgebaut, deren wesentliche Komponenten aus einer objektrelationalen Datenbank, einem Applikationsserver und als Präsentationsschicht aus einem Web-Server und aus einer graphischen Benutzeroberfläche bestehen. Es wurde eine Systemarchitektur speziell für geodätische Problemstellungen entwickelt, was eine Erweiterung der bisher geläufigen Architekturen darstellt. Im Vordergrund standen die Aspekte eines modernen Datenbankentwurfes, der Systemarchitektur, des Sicherheitskonzeptes und der geodätischen Anwendungen zum Aufbau eines internetbasierten Fachinformationssystems. Insbesondere ist die Verwendung der Enterprise JavaBeans-Architektur von komponentenbasierten und verteilten Geschäftsanwendungen und die Entwicklung eines unabhängigen und standardisierten Datenaustausch-Formates Gegenstand dieser Doktorarbeit. Der Zusammenhang zu den geodätischen Fachanwendungen ist hergestellt und die Dokumentation der drei Projektphasen (1996-2004) dieses Teilprojektes B1 im Sonderforschungsbereich 461 *„Starkbeben: von geowissenschaftlichen Grundlagen zu Ingenieurmaßnahmen* ist realisiert worden.

Kapitel 1 zeigt als Einleitung aktuelle Entwicklungen im Bereich der Informationssysteme auf. Im Kontext wissensbasierter Systeme wird das FIS eingeordnet und vorgestellt.

Kapitel 2 beschreibt das objektrelationale Datenbanksystem in seinen relevanten Bestandteilen. Insbesondere die Ansätze der objektorientierten Datenmodellierung zur Abstraktion der realen Welt stehen mehr und mehr im Vordergrund moderner Datenbanken. Die Eigenschaften und Merkmale der ORDBMS, wie z.B. Objekt-Referenzierung, Definition abstrakter Datentypen, Verwendung von LOBs und verschachtelten Tabellen sind beim DB-Konzept und bei der Implementierung der Datenbank entsprechend berücksichtigt worden.

Kapitel 3 präsentiert als Kern dieser Doktorarbeit Referenzarchitekturen, welche im Bezug auf Geographische Informationssysteme eingesetzt werden. Besonders auf die Drei-Schichten-Architektur wird näher eingegangen, die im Kontext internetbasierter Systemarchitekturen mit den drei Bausteinen - EIS-Schicht, Applikations- und Präsentationsschicht - ihre große Bedeutung habt und dementsprechend in dieser Arbeit für geodätische Problemstellungen entwickelt wurde. Der Fokus wurde dabei auf das Sicherheitskonzept gelegt, um ein leistungsfähiges und sicheres Informationssystem im Internet einzusetzen. Signierte Applets und verschlüsselte Datenübertragung mittels zertifizierten Authentifizierungsmechanismen und Secure Socket Layer-Protokollen sind Lösungsverfahren zum sicheren Datentransfer. Letztendlich wird der Datenbankzugriff über Sicherheitsrollen gesteuert.

Kapitel 4 stellt die Enterprise JavaBeans Architektur vor, die auf der Applikationsebene aufgebaut wurde, um die Dateninformationen dem Benutzer im Internet zur Verfügung zu stellen. Der Vorteil der komponentenbasierten Applikationen liegt darin, dass sie skalierbar, leicht übertragbar und multi-userfähig sind und äußerst flexibel im geodätischen Umfeld eingesetzt werden können. Aufgrund der persistenten Datenhaltung kann automatisch ein Datenabgleich zur Laufzeitumgebung erfolgen. Es werden die verschiedenen Varianten Entity-, Session- und Message Driven-Beans beschrieben, wobei besonders auf die Session-Beans eingegangen wird, da dieser Typ zur Kommunikation *„Mensch ⇔ Maschine“* verwendet wurde.

Kapitel 5 enthält Abschnitte zum Erzeugen, Gestalten und Verwenden von Auszeichnungssprachen mittels XML. Dieses Werkzeug wird eingesetzt, um die Datenübertragung zu standardisieren. Es stehen Regeln und Elemente zur Verfügung, um Dokumente zur Darstellung der Ergebnisse aus dem Projekt und zur automatischen Datenspeicherung im DBMS zu erstellen. Die Architektur stellt die vielseitigen Möglichkeiten dar, um XML im Bereich Datenverarbeitung und Kommunikation sowie im Bereich der Datenbanken und Web-Anwendungen einzusetzen. Es werden die notwendigen Schnittstellen zur Transformation von XML-Dokumenten ins HTML-Format spezifiziert bzw. das Java Document Object Model zur Manipulation von XML-Dokumenten erläutert, welches als Standard zum Erzeugen und Transformieren von Datenobjekten im FIS festgelegt wurde. Abschließend wird das XML-Konzept im FIS präsentiert.

Kapitel 6 befasst sich mit den geodätischen Aspekten. Ein wesentliches Kennzeichen geodätischer Anwendungen ist die Positionsbestimmung diskreter Punkte mittels GPS-Messverfahren. Durch den Aufbau eines Überwachungsnetzes können durch Wiederholungsmessungen Deformationsparameter abgeleitet werden, die wichtige Faktoren zur Modellierung geodynamischer Prozesse darstellen. Die Analysen zur rezenten Plattenkinematik werden den Projektpartnern über das FIS bereitgestellt. Damit werden hohe Qualitätsanforderungen gestellt, deren technische Realisierung das Qualitätsmanagementsystem darstellt. In diesem Kapitel werden deshalb die Qualitätsanforderungen und -merkmale vorgestellt, Lösungsverfahren entwickelt und die Ergebnisse der Analysen auf vielfältige Weise im Internet präsentiert.

Kapitel 7 bezieht sich auf die Applikationsentwicklung des Fachinformationssystems *„Dreidimensionale Plattenkinematik in den Ostkarpaten“*. Es werden die Anforderungen, die an das FIS gestellt wurden, sowie die Ziele ausführlich beschrieben. Die Unterkapitel geben die Umsetzung der Funktionalitäten zur Dokumentation der fachlichen Informationen wieder, welche in Form von Formularen, Tabellen und Grafiken im Browser dargestellt werden. Schwerpunkte im Bereich Sicherheit, Qualitätssicherung und Dokumentation geodätischer Analysen werden festgehalten. Mittels des FIS wird ein schneller und sicherer Zugang zu Daten, Informationen und Ergebnisse aus diversen Auswertungen ermöglicht, um Validierungen durch internationale Projektpartner zu zulassen. Außerdem gehen die Analysen in weitere Teilprojekte des Sonderforschungsbereiches 461 ein, um z.B. postseismische Deformationen von der rezenten Tektonik zu trennen.

Dank

Mein Dank gilt Herrn Professor Günter Schmitt für die Übernahme des Hauptreferats, sein Vertrauen und die freie Arbeitsmöglichkeit, die mir während meiner Zeit am Geodätischen Institut gewährt wurde. Herrn Professor Wolffried Stucky danke ich für die Übernahme des Korreferats.

Besonders bedanken möchte ich mich bei Frau Heike Birkel und Herrn Konrad Ringle für ihre Geduld und Mühe beim Korrekturlesen der Arbeit.

Allen Kolleginnen und Kollegen, die mir bei der Erstellung der Arbeit und der Bewältigung der Aufgaben im Sonderforschungsbereich 461 geholfen haben, danke ich für die Unterstützung.

Anhang A

Datenbankentwurf

A.1 Konzeptuelles Schema

Q1	***Klassenname:***	***PRODUKT_TAB***
	Fachschale:	Qualitätsmanagement
	Beschreibung:	Definition des Produktes (Koordinaten, Geschwindigkeiten und Deformationsparameter), Beschreibung der Eigenschaften und Merkmale sowie Erfassung des Herstellungsdatums.
	Hierarchie:	null
	Relation:	AUSW_ID
	Referenzierung:	URHEBER, QUALITAET, PRUEFER, BEURTEILUNG
	Attribut:	PROD_NAME | PROD_TYP | PROD_MERKMAL | PROD_VERSION | PROD_BESCHREIBUNG | URHEBER | QUALITAET | PRUEFER | BEURTEILUNG | AUSW_ID
Q2	***Klassenname:***	***URHEBER_TAB***
	Fachschale:	Qualitätsmanagement
	Beschreibung:	Erfassung der allgemeinen Geschäftsbedingungen, Festlegung der Produkthaftung und der Nutzungsrechte am Produkt. Benennung des Urhebers und des Vertriebes.
	Hierarchie:	PRODUKT_TAB
	Relation:	null
	Referenzierung:	URHEBERNAME
	Attribut:	URHEBERNAME | NUTZUNGSRECHT | HAFTUNG | VERTRIEB | AGB
Q3	***Klassenname:***	***QUALITAETSMERKMAL_TAB***
	Fachschale:	Qualitätsmanagement
	Beschreibung:	Erfassung der allgemeinen Qualitätsmerkmale: *Vollständigkeit*, *Richtigkeit*, *Konsistenz* und *Aktualität*.
	Hierarchie:	PRODUKT_TAB
	Relation:	null
	Referenzierung:	MERKMAL
	Attribut:	MERKMAL | VOLLSTAENDIGKEIT | RICHTIGKEIT | KONSISTENZ | Q_EINHEIT | AKTUALITAET
Q4	***Klassenname:***	***PRUEFER_TAB***
	Fachschale:	Qualitätsmanagement
	Beschreibung:	Benennung des Prüfers, Verknüpfung zur Tabelle Institution, Erfassung des Prüfungsdatums sowie die Freigabe des Produktes an Dritte. Allgemeine Qualitätsbeurteilung des Produktes.
	Hierarchie:	PRODUKT_TAB / INSTITUTION_TAB
	Relation:	INST_KURZNAME
	Referenzierung:	PRUEFERNAME
	Attribut:	PRUEFERNAME | INST_KURZNAME | PRODUKTPRUEFUNG | FREIGABE | BEURTEILUNG

Q5	***Klassenname:***	***QUALITAETSBEURTEILUNG_TAB***
	Fachschale:	Qualitätsmanagement
	Beschreibung:	Gegenüberstellung der Genauigkeit (min, mittel, max), Festlegung der Zuverlässigkeit, Beurteilung der Stationszustände und der Auflistung der Dauer von GPS-Beobachtungen (min, mittel, max) zum Zeitpunkt der Produktherstellung. Beurteilung der Netzdeformation hervorgerufen durch Schwachformen. Erfassung der Offsets und der GPS-Systeme. Zusammenfassung der GPS-Deformationsanalysestrategie.
	Hierarchie:	PRODUKT_TAB
	Relation:	null
	Referenzierung:	QUALITAETSNAME
	Attribut:	QUALITAETSNAME | GENAUIGKEIT | ZUVERLAESSIGKEIT | SCHWACHFORM | STATIONSZUSTAND | BESETZUNG | EQUIPMENT | OFFSET | DEFOSTRATEGIE
K1	***Klassenname:***	***KAMPAGNE_TAB***
	Fachschale:	Kampagne
	Beschreibung:	Auflistung der Kampagne; Dokumentation (Zeitraum, Besetzung) und allgemeine Beschreibung der durchgeführten Kampagnen.
	Hierarchie:	null
	Relation:	KAMP_NAME / INST_ID
	Referenzierung:	null
	Attribut:	KAMP_NAME | KAMP_VON | KAMP_BIS | ANZ_MESSTAGE | ANZ_STATION | INST_ID | KAMP_BESCHREIBUNG
K2	***Klassenname:***	***GPS_BEOBACHTUNG_TAB***
	Fachschale:	Kampagne
	Beschreibung:	Erfassung der Punkte, die in einer Kampagne gemessen wurden.
	Hierarchie:	KAMPAGNE_TAB
	Relation:	KAMP_NAME
	Referenzierung:	KALENDER, BEOBACHTUNG, GPS_SYSTEM, KONFIGURATION, OFFSET
	Attribut:	KAMP_NAME | PUNKT_ID | KALENDER | BEOBACHTUNG | GPS_SYSTEM | KONFIGURATION | OFFSET
K3	***Klassenname:***	***KALENDER_TAB***
	Fachschale:	Kampagne
	Beschreibung:	Erfassung des Datums, des GPS-Tages und der GPS-Woche.
	Hierarchie:	GPS_BEOBACHTUNG_TAB
	Relation:	null
	Referenzierung:	GPS_DATUM
	Attribut:	GPS_DATUM | GPS_WOCHE | GPS_ID
K4	***Klassenname:***	***MESSUNG_TAB***
	Fachschale:	Kampagne
	Beschreibung:	Erfassung der Messdauer und des Messtyps.
	Hierarchie:	GPS_BEOBACHTUNG_TAB
	Relation:	null
	Referenzierung:	MESS_ID
	Attribut:	MESS_ID | MESS_VON | MESS_BIS | MESS_DAUER | MESS_EINHEIT | MESS_TYP
K5	***Klassenname:***	***SYSTEM_TAB***
	Fachschale:	Kampagne
	Beschreibung:	Erfassung der eingesetzten GPS-Systeme mit Seriennummer der Antenne, Sensor und Terminal.
	Hierarchie:	GPS_BEOBACHTUNG_TAB
	Relation:	null
	Referenzierung:	SYSTEM_ID
	Attribut:	SYSTEM_ID | SYSTEM_NAME | SENSOR | SRNR | ANTENNE | ATNR | TERMINAL | TMNR
K6	***Klassenname:***	***KONFIGURATION_TAB***
	Fachschale:	Kampagne

	Beschreibung:	Definition der verwendeten Konfiguration (Stativ, SFB 461-Aufbau, Dreiecksplatte), die beim Aufbau des GPS-Systems verwendet wurde.
	Hierarchie:	GPS_BEOBACHTUNG_TAB
	Relation:	null
	Referenzierung:	KONF_NAME
	Attribut:	KONF_NAME | KONF_OFFSET | KONF_EINHEIT | KONF_BESCHREIBUNG
K7	***Klassenname:***	***OFFSET_TAB***
	Fachschale:	Kampagne
	Beschreibung:	Definition der Offsets, die zur Bestimmung der Position einer Station benötigt werden. Dies können interne Antennenoffsets der GPS-Systeme oder externe Offsets sein, die z.B. durch eine Einzelpunktanalyse geschätzt wurden. Offsets, die durch den Systemaufbau festgelegt wurden, werden in der Tabelle Konfiguration erfasst.
	Hierarchie:	GPS_BEOBACHTUNG_TAB
	Relation:	null
	Referenzierung:	OFFSET_ID
	Attribut:	OFFSET_ID | OFFSET_X | OFFSET_Y | EINHEIT_1 | OFFSET_Z | EINHEIT_2 | OFFSET_BESCHREIBUNG
K8	***Klassenname:***	***BEOBACHTUNGSPLAN_TAB***
	Fachschale:	Kampagne
	Beschreibung:	Übersicht über die Planung der GPS-Kampagne, Verknüpfung zur Abbildung.
	Hierarchie:	KAMPAGNE_TAB
	Relation:	KAMP_NAME / IMG_ID
	Referenzierung:	null
	Attribut:	KAMP_NAME | IMG_ID | BEOBACHTUNGSDESIGN
I1	***Klassenname:***	***INSTITUTION_TAB***
	Fachschale:	Projektverwaltung
	Beschreibung:	Erfassung der Institute, Ämter und Firmen, die an der Durchführung der GPS-Kampagnen beteiligt sind.
	Hierarchie:	null
	Relation:	INST_ID / INST_KURZNAME
	Referenzierung:	null
	Attribut:	INST_ID | INST_NAME | INST_KURZNAME
I2	***Klassenname:***	***ADRESSE_TAB***
	Fachschale:	Projektverwaltung
	Beschreibung:	Erfassung der Adressen.
	Hierarchie:	INSTITUTION_TAB
	Relation:	INST_KURZNAME
	Referenzierung:	null
	Attribut:	INST_KURZNAME | LAND | PLZ | STADT | STRASSE | HAUSNUMMER
I3	***Klassenname:***	***PROJEKT_TAB***
	Fachschale:	Projektverwaltung
	Beschreibung:	Erfassung der Projekte.
	Hierarchie:	INSTITUTION_TAB
	Relation:	INST_KURZNAME
	Referenzierung:	PROJ_BESCHREIBUNG
	Attribut:	PROJ_NAME | PROJ_KURZNAME | INST_KURZNAME | PROJ_BESCHREIBUNG
I4	***Klassenname:***	***PROJEKTDEF_TAB***
	Fachschale:	Projektverwaltung
	Beschreibung:	Beschreibung der Projekte.
	Hierarchie:	PROJEKT_TAB
	Relation:	null
	Referenzierung:	PROJ_KURZNAME
	Attribut:	PROJ_KURZNAME | PROJ_BESCHREIBUNG
I5	***Klassenname:***	***MITARBEITER_TAB***
	Fachschale:	Projektverwaltung
	Beschreibung:	Erfassung der Mitarbeiter.
	Hierarchie:	INSTITUTION_TAB

	Relation:	INST_KURZNAME
	Referenzierung:	KONTAKT
	Attribut:	MITARB_ID | MITARB_TITEL | VORNAME | NACHNAME | BERUFSBEZEICHNUNG | KONTAKT | INST_KURZNAME
I6	***Klassenname:***	***KONTAKT_TAB***
	Fachschale:	Projektverwaltung
	Beschreibung:	Erfassung der Mitarbeiterkontakte.
	Hierarchie:	MITARBEITER_TAB
	Relation:	null
	Referenzierung:	EMAIL
	Attribut:	TELEFON | HANDY | FAX | EMAIL
P1	***Klassenname:***	***PUNKTDEF_TAB***
	Fachschale:	Station
	Beschreibung:	Definition der Vermessungspunkte.
	Hierarchie:	null
	Relation:	PUNKT_ID / STATION_ID
	Referenzierung:	null
	Attribut:	PUNKT_NAME | PUNKT_ID | STATION_ID | PUNKT_STATUS | PUNKT_TYP | PUNKT_ART
P2	***Klassenname:***	***VERMARKUNG_TAB***
	Fachschale:	Station
	Beschreibung:	Erfassung der Vermarkung und Beschreibung der geologischen Verhältnisse.
	Hierarchie:	PUNKTDEF_TAB
	Relation:	PUNKT_ID
	Referenzierung:	null
	Attribut:	VERM_NAME | PUNKT_ID | VERM_TYP | OBJEKT | VERM_HERSTELLUNG | GEOLOGIE | VERMARKUNG
P3	***Klassenname:***	***EXZENTRUM_TAB***
	Fachschale:	Station
	Beschreibung:	Erfassung der gemessenen Exzentren mittels GPS oder klassischer Messmethoden (z.B. Nivellement, Streckenmessung).
	Hierarchie:	PUNKTDEF_TAB
	Relation:	STATION_ID
	Referenzierung:	null
	Attribut:	STATION_ID | MESS_TYP | PUNKT_VON | PUNKT_NACH | EXZ_MESSUNG
P4	***Klassenname:***	***STATION_TAB***
	Fachschale:	Station
	Beschreibung:	Definition und Erfassung der Stationen.
	Hierarchie:	PUNKTDEF_TAB
	Relation:	STATION_ID
	Referenzierung:	STATION_WEG
	Attribut:	STATIONSNAME | STATION_ID | IMG_ID | STATION_AUFBAU | STATION_ZUSTAND | STATION_WEG
P5	***Klassenname:***	***STAWEG_TAB***
	Fachschale:	Station
	Beschreibung:	Beschreibung des Stationsweges.
	Hierarchie:	STATION_TAB
	Relation:	null
	Referenzierung:	STATION_ID
	Attribut:	STATION_ID | WEGBESCHREIBUNG
F1	***Klassenname:***	***IMGDEF_TAB***
	Fachschale:	Karte
	Beschreibung:	Definition und Beschreibung der Karte, Festlegung des Urhebers und der Bildrechte.
	Hierarchie:	null
	Relation:	IMG_ID
	Referenzierung:	METADATA / KARTE / IMAGE

	Attribut:	IMG_NAME | IMG_ID | IMG_URHEBER | IMG_RECHTE | IMG_BESCHREIBUNG | METADATA | KARTE | IMAGE
F2	***Klassenname:***	***IMGMETADATA_TAB***
	Fachschale:	Karte
	Beschreibung:	Erfassung der Metadaten.
	Hierarchie:	IMGDEF_TAB
	Relation:	null
	Referenzierung:	IMG_NAME
	Attribut:	IMG_NAME | IMG_FORMAT | IMG_SIZE
F3	***Klassenname:***	***IMAGE_TAB***
	Fachschale:	Karte
	Beschreibung:	Speicherung der Karte, der Abbildung oder des Photos als *binary large object*.
	Hierarchie:	IMGDEF_TAB
	Relation:	null
	Referenzierung:	IMG_NAME
	Attribut:	IMG_NAME | IMAGE
F4	***Klassenname:***	***KARTE_TAB***
	Fachschale:	Karte
	Beschreibung:	Erfassung der Kartendefinition, Typ, Skalierung und Aktualität.
	Hierarchie:	IMGDEF_TAB
	Relation:	null
	Referenzierung:	IMG_NAME
	Attribut:	IMG_NAME | KARTE_ID | KARTE_TYP | SCALE | PROJEKTION | AKTUALITAET
A1	***Klassenname:***	***REFAUSWERTUNG_TAB***
	Fachschale:	Analyse
	Beschreibung:	Erfassung der Auswertung (Datum und Name).
	Hierarchie:	null
	Relation:	AUSW_ID
	Referenzierung:	SOFTWARE / AUSWDEF
	Attribut:	AUSW_ID | AUSWERTUNG | SOFTWARE | AUSWDEF
A2	***Klassenname:***	***AUSWDEF_TAB***
	Fachschale:	Analyse
	Beschreibung:	Erfassung der Methode und Strategie. Beschreibung der Auswertung.
	Hierarchie:	REFAUSWERTUNG_TAB
	Relation:	null
	Referenzierung:	AUSW_METHODE
	Attribut:	AUSW_METHODE | AUSW_STRATEGIE | AUSW_BESCHREIBUNG
A3	***Klassenname:***	***SOFTWARE_TAB***
	Fachschale:	Analyse
	Beschreibung:	Erfassung des Softwarenamens, der Quelle, der Version sowie die Aktualität. Beschreibung der Software.
	Hierarchie:	REFAUSWERTUNG_TAB
	Relation:	null
	Referenzierung:	SOFT_ID
	Attribut:	SOFTWARENAME | SOFT_ID | SOFT_VERSION | SOFT_QUELLE | SOFT_DATUM | SOFT_BESCHREIBUNG
A4	***Klassenname:***	***DEFODEF_TAB***
	Fachschale:	Analyse
	Beschreibung:	Erfassung der Deformationsanalyse, Typ und Zeitraum. Beschreibung der Analyse.
	Hierarchie:	REFAUSWERTUNG_TAB
	Relation:	DEFO_ID / AUSW_ID / IMG_ID
	Referenzierung:	null
	Attribut:	DEFO_ID | AUSW_ID | IMG_ID | DEFO_TYP | DEFOANALYSE_VON | DEFOANALYSE_BIS | DEFO_BESCHREIBUNG
S1	***Klassenname:***	***KOORD_SYSTEM_TAB***
	Fachschale:	Koordinatensystem

	Beschreibung:	Definition des Namens, Erfassung des Typs und der Dimension. Beschreibung des Koordinatensystems.
	Hierarchie:	null
	Relation:	KOORD_ID
	Referenzierung:	KOORD_ACHSE
	Attribut:	KOORD_SYSTEMNAME | KOORD_ID | KOORD_TYP | KOORD_DIMENSION | KOORD_ACHSE | KOORD_BESCHREIBUNG
S2	***Klassenname:***	***KOORD_ACHSEN_TAB***
	Fachschale:	Koordinatensystem
	Beschreibung:	Definition der Achsen, Erfassung der Einheit. Beschreibung der Koordinatenachsen.
	Hierarchie:	KOORD_SYSTEM_TAB
	Relation:	null
	Referenzierung:	ACHSEN_ID
	Attribut:	ACHSEN_ID | ACHSENNAMEN | ACHSEN_SI | ACHSENBESCHREIBUNG
D1	***Klassenname:***	***REFDATUM_TAB***
	Fachschale:	Geodätisches Datum
	Beschreibung:	Definition der ID.
	Hierarchie:	null
	Relation:	DATUM_ID
	Referenzierung:	NETZ_DATUM / REF_DATUM / ELLIPSOID / PHYS_PARA / REF_HOEHE
	Attribut:	DATUM_ID | NETZ_DATUM | REF_DATUM | ELLIPSOID | PHYS_PARA | REF_HOEHE
D2	***Klassenname:***	***NETZ_TAB***
	Fachschale:	Geodätisches Datum
	Beschreibung:	Definition des Netznamens, Erfassung des Bezugsrahmens für die Koordinaten und Geschwindigkeiten. Beschreibung des GPS-Netzes.
	Hierarchie:	REFDATUM_TAB
	Relation:	null
	Referenzierung:	NETZNAME
	Attribut:	NETZNAME | KOORD_DATUM | GESCHW_DATUM | NETZBESCHREIBUNG
D3	***Klassenname:***	***DATUM_TAB***
	Fachschale:	Geodätisches Datum
	Beschreibung:	Definition des Datums, Erfassung der Epoche und der Quelle. Beschreibung des festgelegten Datums.
	Hierarchie:	REFDATUM_TAB
	Relation:	null
	Referenzierung:	DATUMSNAME
	Attribut:	DATUMSNAME | DATUMSKURZNAME | DATUMSEPOCHE | DATUMSQUELLE | DATUMSBESCHREIBUNG
D4	***Klassenname:***	***REFELLIPSOID_TAB***
	Fachschale:	Geodätisches Datum
	Beschreibung:	Definition und Beschreibung des Bezugsellipsoids.
	Hierarchie:	REFDATUM_TAB
	Relation:	null
	Referenzierung:	ELL_NAME
	Attribut:	ELL_NAME | ELL_ID | ELL_PAR1 | ELL_PAR2 | ELL_BESCHREIBUNG
D5	***Klassenname:***	***PHYS_TAB***
	Fachschale:	Geodätisches Datum
	Beschreibung:	Definition der physikalischen Parameter.
	Hierarchie:	REFDATUM_TAB
	Relation:	null
	Referenzierung:	ELL_ID
	Attribut:	ELL_ID | PHYS_KONSTANTE | KOEFFIZIENTEN | PHYS_PARAMETER
D6	***Klassenname:***	***REFHOEHE_TAB***
	Fachschale:	Geodätisches Datum
	Beschreibung:	Definition der physikalischen Höhenparameter.
	Hierarchie:	REFDATUM_TAB

	Relation:	null
	Referenzierung:	SYSTEMNAME
	Attribut:	SYSTEMNAME | AEQUIPOTENTIALFLAECHE | SYSTEMBESCHREIBUNG | ELLIPSOID_BESCHREIBUNG
G1	***Klassenname:***	***REFPUNKT_TAB***
	Fachschale:	Geometrische Objekte
	Beschreibung:	Definition der geometrischen Objekte (Topologische Punkte, GPS-Netzpunkte).
	Hierarchie:	null
	Relation:	GID / PUNKT_ID / KOORD_ID / DATUM_ID / AUSW_ID
	Referenzierung:	null
	Attribut:	GID | PUNKT_ID | KOORD_ID | DATUM_ID | AUSW_ID
G2	***Klassenname:***	***SDO_MODELL***
	Fachschale:	Geometrische Objekte
	Beschreibung:	Erfassung und Beschreibung der Punktobjekte unter Verwendung von Spatial Data Objects.
	Hierarchie:	REFPUNKT_TAB
	Relation:	GID (Automatische Generierung mittels des Datenbankobjektes Sequenz)
	Referenzierung:	null
	Attribut:	GID | MODELLTYP | PUNKTDATUM | GEOMETRIE | AUSW_ID
G3	***Klassenname:***	***SDO_SIGMA_MODELL***
	Fachschale:	Geometrische Objekte
	Beschreibung:	Erfassung und Beschreibung der Genauigkeit der Punktobjekte.
	Hierarchie:	REFPUNKT_TAB
	Relation:	GID (Automatische Generierung mittels des Datenbankobjektes Sequenz)
	Referenzierung:	null
	Attribut:	GID | SIGMA | PKT_ERROR
G4	***Klassenname:***	***SDO_FEHLER_MODELL***
	Fachschale:	Geometrische Objekte
	Beschreibung:	Erfassung und Beschreibung der Fehler der Punktobjekte.
	Hierarchie:	REFPUNKT_TAB
	Relation:	GID (Automatische Generierung mittels des Datenbankobjektes Sequenz)
	Referenzierung:	null
	Attribut:	GID | FEHLER | ERR_EINHEIT
G5	***Klassenname:***	***SDO_GESCHWINDIGKEIT***
	Fachschale:	Geometrische Objekte
	Beschreibung:	Erfassung und Beschreibung der Geschwindigkeit.
	Hierarchie:	REFPUNKT_TAB
	Relation:	GID (Automatische Generierung mittels des Datenbankobjektes Sequenz)
	Referenzierung:	null
	Attribut:	GID | GESCHWINDIGKEIT | EINHEIT_VP
G6	***Klassenname:***	***SDO_SIGMA_GESCHWINDIGKEIT***
	Fachschale:	Geometrische Objekte
	Beschreibung:	Erfassung und Beschreibung der Genauigkeit der Geschwindigkeit.
	Hierarchie:	REFPUNKT_TAB
	Relation:	GID (Automatische Generierung mittels des Datenbankobjektes Sequenz)
	Referenzierung:	null
	Attribut:	GID | SIGMA_VP | EINHEIT_MVP
G7	***Klassenname:***	***SDO_FEHLER_GESCHWINDIGKEIT***
	Fachschale:	Geometrische Objekte
	Beschreibung:	Erfassung und Beschreibung der Fehler der Geschwindigkeit.
	Hierarchie:	REFPUNKT_TAB
	Relation:	GID (Automatische Generierung mittels des Datenbankobjektes Sequenz)
	Referenzierung:	null
	Attribut:	GID | ERR_GESCHW | ERR_EINHEIT

Tabelle A.1: *Überblick über die Objektklassen und Beziehungen zueinander, die nach der Erstellung des Datenbankdesigns im FIS entwickelt wurden.*

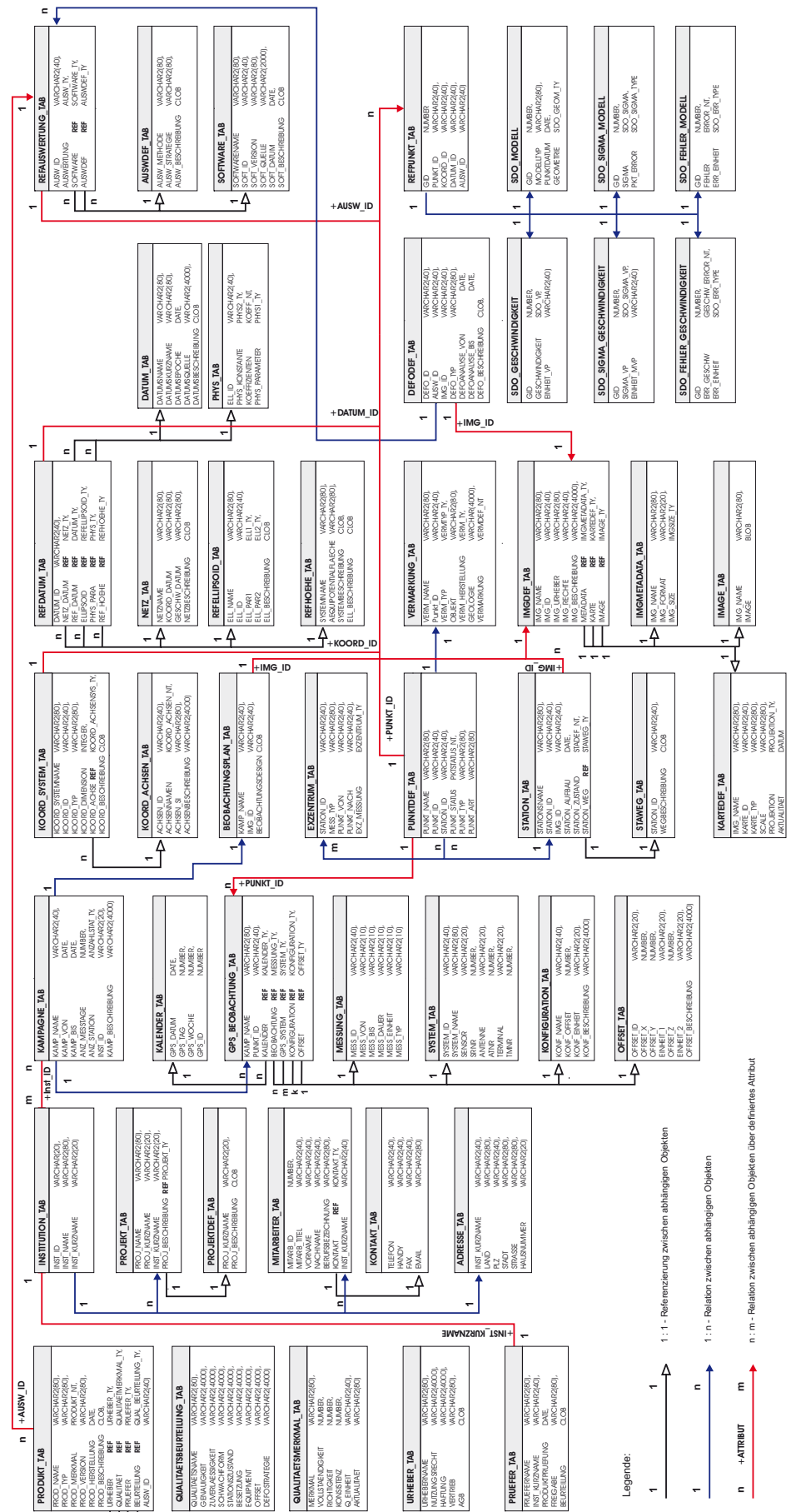


Abbildung A.1: *Kompletter Datenbankentwurf des Fachinformationssystems „Dreidimensionale Plattenkinematik in Rumänien“.*

A.2 Programmcode: Referenzierung

```
create or replace type Software_TY as object
(
  Softwarename       varchar2(80),
  Soft_ID            varchar2(40),
  Soft_Version       varchar2(80),
  Soft_Quelle        varchar2(2000),
  Soft_Datum         date,
  Soft_Beschreibung  CLOB
);
/
create table Software_TAB of Software_TY
  lob(Soft_Beschreibung) store as
  (
    tablespace SFB461
    storage (initial 50K next 50K pctincrease 0)
    chunk 16K pctversion 10 nocache logging
  );
create or replace type Auswdef_TY as object
(
  Ausw_Methode       varchar2(80),
  Ausw_Strategie     varchar2(80),
  Ausw_Beschreibung  CLOB
);
/
create table Auswdef_TAB of Auswdef_TY
  lob(Ausw_Beschreibung) store as
  (
    tablespace SFB461
    storage (initial 50K next 50K pctincrease 0)
    chunk 16K pctversion 10 nocache logging
  );
create or replace type Ausw_TY as object
(
  Ausw_Name          varchar2(80),
  Ausw_Typ           varchar2(80),
  Ausw_Datum         date,
  Auswerter          varchar2(80)
);
/
create table RefAuswertung_TAB
(
  Ausw_ID            varchar2(80),
  Auswertung         Ausw_TY,
  Software     Ref   Software_TY,
  Auswdef      Ref   Auswdef_TY
);
```

A.3 Programmcode: BFILE → BLOB

```
create or replace procedure addnewbfile
/* Head */
(
  /* Definition der Attributnamen/Datentypen */
  p_Code      IN Varchar2,
  p_Filename  IN Varchar2
) AS

  /* Parameter */
  File_Loc       BFILE;
  v_date         date;
  dest_lob       Blob;
  var1           integer;
  img_check      varchar2(80);
  /* Exception */
  BFILE_NOTFOUND exception;
  IMAGE_DOUBLEFOUND exception;

  /* Cursor */
  Cursor img_cursor IS
    select img_name from image_tab where img_name = p_CODE;

/* Body */
Begin

  /* Pfadangabe + Dateiname */
  File_Loc      := BFilename('IMAGES',p_Filename);
  var1          := DBMS_LOB.FILEEXISTS(File_Loc);
  v_date        := sysdate;

  /* Begin Exception: IMAGE_DOUBLEFOUND - RAISE EXCEPTION */

  /* File exists is true */
  if var1 = 1 then
    img_check := null;
    OPEN img_cursor;
      Fetch img_cursor into img_check;
      if img_check is not null then Raise IMAGE_DOUBLEFOUND;
      end if;
    Close img_cursor;

    /* Bildnamen in Image_Tab einlesen */
    insert into image_tab(img_name,image)
    values
    (
      p_code,
      empty_BLOB()
    );


    /* Dateneingabe im Bfile */
    insert into ImgBfile_tab(img_ref,img_file)
      values(p_code,Bfilename('IMAGES',p_Filename));

    /* Update für BLOB vorbereiten */
    select image into dest_lob from image_tab where img_name = p_code
      for update;
```

```
    /* DBMS_LOB.Packages */
    DBMS_OUTPUT.PUT_LINE('BFILE_Locator: . . . ok');
    DBMS_LOB.FILEOPEN(File_Loc,DBMS_LOB.File_Readonly);
    DBMS_LOB.LOADFROMFILE(dest_lob,File_Loc,DBMS_LOB.GETLENGTH(File_Loc));

    /* Update BLOB in die Bildtabelle */
    update image_tab set image = dest_lob where img_name = p_code;
    select image into dest_lob from image_tab where img_name = p_code;

    /* Ausgabe am Bildschirm */
    DBMS_OUTPUT.PUT_LINE('Check Image_BLOB: '||
    DBMS_LOB.GETLENGTH(dest_lob)||'(KB)');
    DBMS_OUTPUT.PUT_LINE('Programm ausgeführt am: '||v_date);

    /* Bilddatei in der BFILE_Tabelle löschen */
    delete imgbfile_tab;
    commit;

    /* Datei schließen */
    DBMS_LOB.FILECLOSE(File_Loc);

  end if; /* wenn BFile-Pfad und Datei existiert */

  /* File doesn't exist */
  if var1 = 0 then
    Raise BFILE_NOTFOUND;
  end if;

  /* Definition of the exception */
  Exception
    when BFILE_NOTFOUND then
    Raise_Application_Error(-20001,'Datei ist nicht vorhanden!');
    when Image_DOUBLEFOUND then
    DBMS_OUTPUT.PUT_LINE(SQLERRM(-20002)||'Img_REF ist in Tabelle
                    <IMAGE_TAB> vorhanden.');
end addnewbfile;
```

A.4 Programmcode: JDBC ⇔ SQL-Anweisung

```
public String [] listsqlorder(String query, String spaltenbreite) {
try {
  con = getConnection();
  ps = con.prepareStatement (query);
  result = ps.executeQuery();
  ResultSetMetaData md = result.getMetaData();
  SQLWarning sqlw = con.getWarnings();
  . . .
  for (int i = 0; i < colmax; i++) {
    collength[i] = md.getColumnDisplaySize(i + 1);
    coltype[i] = md.getColumnTypeName(i + 1);
    colname = md.getColumnName(i+1);
    spmax[i] = colname.length();
    . . .
  }
  . . .
  while (result.next()) {
    StringBuffer buf = new StringBuffer();
    int spanz = 0;
    for (int i=0; i < colmax; i++) {
      if (coltype[i] == "VARCHAR2") {
       String zelle = result.getString (i+1);
       nullobject = result.wasNull();
        . . .
      }
      else if (coltype[i] == "NUMBER") {
        double zelle = result.getDouble (i+1);
          // int zelle = result.getInt (i+1);
          // float zelle = result.getFloat (i+1);
          // long zelle = result.getLong (i+1);
        . . .
      }
      else if (coltype[i] == "BOOLEAN") {
        boolean zelle = result.getBoolean (i+1);
        . . .
      }
      else if (coltype[i] == "DATE") {
        Date zelle = result.getDate(i+1);
          // Time zelle = result.getTime(i+1);
          // Timestamp zelle = result.getTimestamp(i+1);
        . . .
      }
      else if (coltype[i] == "CLOB") {
        Clob clob = result.getClob (i+1);
        long anzchar = clob.length();
        int charende = (int)anzchar;
        String zelle = clob.getSubString(1, charende);
        . . .
      }
      else if (coltype[i] == "BLOB") {
        Blob blob = result.getBlob(i + 1);
        long anzbyte = blob.length();
        int byteende = (int) anzbyte;
        byteende = byteende / 1000;
        String  zelle =("BLOB: " + byteende + " KB");
        . . .
      }
```

```
      else {
        Object object = result.getObject(i + 1);
          //Ref ref = result.getRef(i+1);
          //Array A = result.getArray(i+1);
          // Struct struct = (Struct)result.getObject(i+1);
        String zelle = object.toString();
          //String zelle = ref.toString(); String zelle = A.toString();
          //String zelle = struct.toString();)
        /* Ausgabe: oracle.sql.REF@.... */
        if (zelle.substring(11, 14).equals("REF")) {
          buf.append("REF:  " + subzelle);
          . . .
        }
        /* Ausgabe: oracle.sql.ARRAY@.... */
        else if (zelle.substring(11, 16).equals("ARRAY")) {
          buf.append("ARRAY:  " + subzelle);
          . . .
        }
        /* Ausgabe: oracle.sql.STRUCT@.... */
        else if (zelle.substring(11, 17).equals("STRUCT")) {
          buf.append("STRUCT:  " + subzelle);
          . . .
        }
        . . .
      }
    }
  }
  . . .
  System.out.println("Reading the table of the SQL-statement
                      was successful.");
  return returnArray;
}
catch (Exception e) {
  String[] sqlexception = new String[1];
  sqlexception[0] = e.toString()+"\n\n";
  System.out.println("Reading the data was not successful!\n");
  rowcounter = -1;
  return sqlexception;
}
finally {
  try {result.close ();} catch (Exception ignored){}
  try {ps.close ();    } catch (Exception ignored){}
  try {con.close ();   } catch (Exception ignored){}
  }
}
```

Anhang B

Details zur Referenzarchitektur

B.1 Programmcode: Rahmen des FIS-Applets

```
package jbuilder;

import interfaces.*;
import mathgeod.*;
import rum.*;
import java.awt.*;
import java.text.*;
import java.util.*;
import javax.naming.InitialContext;
import javax.rmi.PortableRemoteObject;
import javax.swing.*;
import javax.swing.event.*;

/**
 * <B>
 *   Fachinformationssystem "Dreidimensionale Plattenkinematik
 *   in den Ostkarpaten"
 * </B>
 *
 * @author Dipl.-Ing. Michael Nutto
 * @version 1.0
 * @since 01.01.2002
 *
 * <B>
 *   Copyright (c) 2002-2005
 *   Geodätisches Institut - Universität Karlsruhe (TH)
 * </B>
 */

public class fis extends JApplet
{
  // Datenbankobjekte:
  Statement                         stmt             = null;
  Connection                        con              = null;
  ResultSet                         rs;
  ResultSetMetaData                 md;
  DatabaseMetaData                  dbmd;
  byte[]                            image_station    = null;

  //DB-Anbindung: boolean connected char[] password String user
```

```
  boolean                          connected        = false;
  char[]                           password;
  String                           user             = null
  ...

  /** ***************************************************************** */
  /** ******************** Klassen: JBUILDER ************************** */
  /** ***************************************************************** */

  Ausnahme                         ex1              = new Ausnahme();
  ...
  XML                              xml              = new XML();

  public void init() {
    try {
      fachinformationssystem();
    }
    catch (Exception ex) {
      ex.printStackTrace();
    }
  }

  private void fachinformationssystem() throws Exception {
    this.contentPane.setVisible(false);
    . . .
  }

  public void processEvent(AWTEvent e) {
    if (e.getID() == Event.WINDOW_DESTROY) {
      System.exit(0);
    }
  }

  public static void main(String args[]) {
    try {
      UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch (Exception e) {
      e.printStackTrace();
    }
    JFrame gikfis = new JFrame("Fachinformationssystem Version 1.0");
    fis fiswin = new fis();
    fiswin.init();
    fiswin.start();
    . . .
  }
  static {
    try {
      UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  }

  public void start()   {this.setEnabled(true); }
  public void stop()    {this.setEnabled(false);}
  public void destroy() {this.removeAll();      }
}
```

B.2 FIS-Authentifizierung

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ===================================================================== -->
<!--  JBoss Server Configuration                                           -->
<!-- ===================================================================== -->
<!-- $Id: oracle-ds.xml,v 1.1.2.4 2003/09/17 03:46:01 ejort Exp $         -->
<!-- ===================================================================== -->

<datasources>
  <local-tx-datasource>

    <jndi-name>OracleDS</jndi-name>
    <connection-url>
      jdbc:oracle:thin:@servername.gik.uni-karlsruhe.de:1521:DB-Aliasname
    </connection-url>

    <!-- Mögliche Verbindungsformen
    <connection-url>
      jdbc:oracle:oci:@youroracle-tns-name
    </connection-url>
    <connection-url>
      jdbc:oracle:oci:@(description=(address=(host=youroraclehost)
      (protocol=tcp)(port=1521))(connect_data=
      (SERVICE_NAME=yourservicename)))
    </connection-url>
    -->

    <!-- Anmeldung bei der Datenbank -->
    <driver-class>oracle.jdbc.driver.OracleDriver</driver-class>
    <user-name>username</user-name>
    <password>********</password>

    <!-- Uses the pingDatabase method to check a connection is still
    valid before handing it out from the pool -->
    <!--valid-connection-checker-class-name>
    org.jboss.resource.adapter.jdbc.vendor.OracleValidConnectionChecker
    </valid-connection-checker-class-name -->
    <!-- Checks the Oracle error codes and messages for errors -->
    <exception-sorter-class-name>
    org.jboss.resource.adapter.jdbc.vendor.OracleExceptionSorter
    </exception-sorter-class-name>

  </local-tx-datasource>
</datasources>

</xml>
```

Anhang C

Datenübertragung: von der DB zum Client

C.1 Zustandslose Session-Bean

```
package sessionbeans;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.ejb.CreateException;
import javax.ejb.EJBException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;

public class DBAccessBean implements SessionBean
{
  private SessionContext sessionContext;
  boolean access;
  boolean getaccess;
  String oracle_message = "null";

  public void setSessionContext(SessionContext context)
  {
    sessionContext = context;
  }

  Connection con = null;
  PreparedStatement ps = null;
  ResultSet result = null;

  public void ejbCreate() throws CreateException
  {
    System.out.println ("DBAccessBean::ejbCreate was successful.");
  }

  public boolean dbaccess(char[] password, String user)
  {
   try
    {
      access = getConnection(password, user);
      return access;
```

```
    }
    catch (Exception e)
    {
      access = false;
      oracle_message = "Die Datenbank ist nicht hochgefahren!";
      return access;
    }
    finally
    {
      try { con.close (); } catch (Exception ignored){}
    }
  }

  public String getoraclemessage()
  {
   return oracle_message;
  }

  public boolean getConnection(char[] password, String user)
  throws SQLException, ClassNotFoundException
  {
    try
    {
      /* Treiber laden */
      Class.forName("oracle.jdbc.driver.OracleDriver");

      /* Verbindung herstellen */
      String url1 = ("jdbc:oracle:thin:@");
      String url2 = (":1521:dbname");
      String net8 = (url1 + "hostname" + url2);
      final String passwd = new String(password);
      con = DriverManager.getConnection(net8, user, passwd);
      String dbcon = ("Connection to the database is established.");
      getaccess = true;
      oracle_message = "Datenbank ist hochgefahren";
      System.out.println(dbcon);
      return getaccess;
    }
    catch(Exception ne)
    {
      System.out.println("Datenbankverbindung nicht erfolgreich!");
      oracle_message = (ne.toString());
      getaccess = false;
      return getaccess;
    }
  }

  public void ejbActivate() throws EJBException {
    System.out.println ("DBAccessBean::ejbActivate is done.");
  }

  public void ejbPassivate() throws EJBException {
    System.out.println ("DBAccessBean::ejbPassivate is done.");
  }

  public void ejbRemove() throws EJBException {
    System.out.println ("DBAccessBean::ejbRemove is done.");
  }
}
```

C.2 Deployment-Deskriptor

META-INF\ejb-jar.xml:

```
<ejb-jar >
  <enterprise-beans>
    <session>
      <description>Datenbankzugriff</description>
      <display-name>DBAccessBean</display-name>
      <ejb-name>DBAccessBean</ejb-name>
      <home>interfaces.DBAccessBeanHome</home>
      <remote>interfaces.DBAccessBeanRemote</remote>
      <ejb-class>sessionbeans.DBAccessBean</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
      <resource-ref >
        <res-ref-name>jdbc/FISDB</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <res-auth>Application</res-auth>
      </resource-ref>
    </session>
    <session >
      <description>DB-Anweisung</description>
      <display-name><NAME>Bean</display-name>
      <ejb-name><NAME>Bean</ejb-name>
      <home>interfaces.<NAME>Bean</home>
      <remote>interfaces.<NAME>Bean</remote>
      <ejb-class>sessionbeans.<NAME>Bean</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
      <resource-ref >
        <res-ref-name>jdbc/FISDB</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <res-auth>Container</res-auth>
      </resource-ref>
    </session>
  </enterprise-beans>

  <assembly-descriptor>
    <security-role>
      <description>
        <![CDATA[Diese Rolle erlaubt vollen
          Zugriff von allen Anwendern.]]>
      </description>
      <role-name>everyone</role-name>
    </security-role>

    <method-permission>
      <role-name>everyone</role-name>
      <method>
        <ejb-name>DBAccessBean</ejb-name>
        <method-name>*</method-name>
      </method>
    </method-permission>

    <container-transaction>
      <method>
        <ejb-name>DBAccessBean</ejb-name>
        <method-name>*</method-name>
```

```
      </method>
      <trans-attribute>Required</trans-attribute>
    </container-transaction>
  </assembly-descriptor>
</ejb-jar>
```

META-INF\jboss.xml:

```
<jboss>
  <enterprise-beans>
    <session>
      <ejb-name>DBAccessBean</ejb-name>
      <jndi-name>DBAccessBean</jndi-name>
      <resource-ref>
        <res-ref-name>jdbc/FISDB</res-ref-name>
        <jndi-name>java:/OracleDS</jndi-name>
      </resource-ref>
      <method-attributes> </method-attributes>
    </session>
    <!-- weitere Sessionbeans -->
  </enterprise-beans>
</jboss>
```

C.3 Methode: DBAccessBean()

```
/** ********************************************************************** */
/** ****************** Methode: SessionBean() ************************** */
/** ********************************************************************** */
  private void DBAccessBean()
  {
    try
    {

      Properties jndiProps = new Properties();
      jndiProps.setProperty("java.naming.factory.initial",
          "org.jnp.interfaces.NamingContextFactory");
      jndiProps.setProperty("java.naming.provider.url", par1.myServer);
      jndiProps.setProperty("java.naming.factory.url.pkgs",
          "org.jboss.naming:org.jnp.interfaces");

      DBAccessBeanHome home = (DBAccessBeanHome) PortableRemoteObject.
        narrow(new InitialContext(jndiProps).lookup("DBAccessBean"),
        DBAccessBeanHome.class);

      // Start the stateless session bean
      //System.out.println("Starting GeoKoord Sessionbean.");
      DBAccessBeanRemote db = home.create();

      connected = db.dbaccess(password, user);
      String ora_mess = db.getoraclemessage();

      //System.out.println(connected+"\n"+user+"\n"+ora_mess);

      if (connected == true)
      {
        String dbjdbc = ("Oracle JDBC Treiber Version 9i ist geladen.");
        pwd.jdbc_output.setText(dbjdbc);
```

```
        String dbcon = ("Die Verbindung zur Datenbank ist hergestellt.");
        pwd.connection_output.setText(dbcon);

        /* Kommentare */
        pwd.commentArea.setText("");
        String com1 = "Oracle Release 9i - Production \n";
        String com2 = "Verbunden mit Oracle9i Enterprise Edition \n";
        String com3 = "with the option partition JServer Release 9i \n\n";
        String com4 = "Verbunden mit dem Fachinformationssystem:\n";
        String com5 = "Dreidimensionale Plattenkinematik in Rumaenien\n";
        String com6 = "(c) copyright 2005 Geodaetisches Institut Karlsruhe \n";
        pwd.commentArea.append(com1);
        pwd.commentArea.append(com2);
        pwd.commentArea.append(com3);
        pwd.commentArea.append(com4);
        pwd.commentArea.append(com5);
        pwd.commentArea.append(com6);
      }
      else
      {
        pwd.connection_output.setText("Keine Verbindung zur Datenbank!\n");
        pwd.commentArea.append("\n");
        pwd.commentArea.append("Oracle Fehlermeldung: \n");
        pwd.commentArea.append("--------------------- \n\n");
        pwd.commentArea.append(ora_mess);
      }

      db.remove();
    }
    catch (SecurityException se)
    {
      pwd.commentArea.setText(se.getMessage());
    }
    catch (Exception ex)
    {
      connected = false;
      pwd.connection_output.setText("Keine Verbindung zur Datenbank!\n");
      pwd.commentArea.setText("\n");
      pwd.commentArea.append("Keine Verbindung zum Server!\n");
      pwd.commentArea.append("JBOSS Version 3.2\n\n");
      pwd.commentArea.append("Bitte benachrichtigen Sie den Administrator.");
    }
  }
```

Anhang D

Programmcode: XML

D.1 XML-Datei: Final2000.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-page source="Deformationsanalyse - Sonderforschungsbereich 461"
  title="Dreidimensionale Plattenkinematik in den Ostkarpaten (Vrancea)">
  <!--Element: Tabelle_Koordinate-->
  <Tabelle_Koordinate>
    <!--Element: Titel Attribut:table=Tabelle Attribut:id=$tabid-->
    <Titel table="Tabelle" id=" 1: ">
      <!--Elementenliste: <Typ /><Bezug /><Epoche /><Auswertung />-->
      <Typ>geozentrische Koordinate (X, Y, Z)</Typ>
      <Bezug>ITRF 97</Bezug>
      <Epoche>15.06.99</Epoche>
      <Auswertung>Final 2000</Auswertung>
    </Titel>
    <!--Element: geozentrisch Attribut:id=Koordinate-->
    <geozentrisch id="Koordinate">
      <!--Element: read Attribut:id=$Stationsname-->
      <read id="BABU-A">
        <PKNR>BABU-A</PKNR>
        <X>3888849.881</X>
        <Y>2001096.604</Y>
        <Z>4627242.761</Z>
        <Einheit>Meter</Einheit>
      </read>
      . . .
    </geozentrisch>
    <!--Element: Titel Attribut:table=Tabelle Attribut:id=$tabid-->
    <Titel table="Tabelle" id=" 2: ">
      <!--Elementenliste: <Typ /><Bezug /><Epoche /><Auswertung />-->
      <Typ>geographische Koordinate (B, L, h)</Typ>
      <Bezug>ITRF 97</Bezug>
      <Epoche>15.06.99</Epoche>
      <Auswertung>Final 2000</Auswertung>
    </Titel>
    <!--Element: geographisch Attribut:id=Koordinate-->
    <geographisch id="Koordinate">
      <!--Element: read Attribut:id=$Stationsname-->
      <read id="BABU-A">
        <PKNR>BABU-A</PKNR>
        <Breite>46.806861767</Breite>
        <Länge>27.229120986</Länge>
```

```
        <Einheit1>Grad</Einheit1>
        <Höhe>202.429</Höhe>
        <Einheit2>Meter</Einheit2>
      </read>
      . . .
    </geographisch>
    <!--Element: Titel Attribut:table=Tabelle Attribut:id=$tabid-->
    <Titel table="Tabelle" id=" 3: ">
      <!--Elementenliste: <Typ /><Bezug /><Epoche /><Auswertung />-->
      <Typ>UTM - Koordinate (N, E)</Typ>
      <Bezug>ITRF 97</Bezug>
      <Epoche>15.06.99</Epoche>
      <Auswertung>Final 2000</Auswertung>
    </Titel>
    <!--Element: UTM Attribut:id=Koordinate-->
    <UTM id="Koordinate">
      <!--Element: read Attribut:id=$Stationsname-->
      <read id="BABU-A">
        <PKNR>BABU-A</PKNR>
        <North>5183727.206</North>
        <East>517481.723</East>
        <Einheit>Meter</Einheit>
      </read>
      . . .
    </UTM>
  </Tabelle_Koordinate>

  <!--Element: Tabelle_Koordinate_Genauigkeit-->
  <Tabelle_Koordinate_Genauigkeit>
    <!--Element: Titel Attribut:table=Tabelle Attribut:id=$tabid-->
    <Titel table="Tabelle" id=" 4: ">
      <!--Elementenliste: <Typ /><Bezug /><Epoche /><Auswertung />-->
      <Typ>Genauigkeit der geozentrischen Koordinaten s(X, Y, Z)</Typ>
      <Bezug>ITRF 97</Bezug>
      <Epoche>15.06.99</Epoche>
      <Auswertung>Final 2000</Auswertung>
    </Titel>
    <!--Element: geozentrisch Attribut:id=Genauigkeit-->
    <geozentrisch id="Genauigkeit">
      <!--Element: read Attribut:id=$Stationsname-->
      <read id="BABU-A">
        <PKNR>BABU-A</PKNR>
        <sigmaX>4.6</sigmaX>
        <sigmaY>2.8</sigmaY>
        <sigmaZ>5.3</sigmaZ>
        <Einheit>Millimeter</Einheit>
      </read>
      . . .
    </geozentrisch>
. . .
  </Tabelle_Koordinate_Genauigkeit>
  . . .
</web-page>
```

D.2 XSLT-Datei: Final2000.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns="http://www.w3.org/TR/xhtml1/strict">
<xsl:output method="html"/>

<xsl:template match="/">
<html>
 <head><title>Fachinformationssystem</title></head>
 <body text="#000000" bgcolor="#FFFFCC" link="#0000EE" vlink="#551A8B">
  <center>
   <table border="5" cellspacing="2" cols="2" bgcolor="#FFCC99">
    <tr>
     <td align="center" width="50%" height="100%">
      <b>Sonderforschungsbereich 461</b>
      <br/>Starkbeben:
      <br/>von geowissenschaftlichen Grundlagen
      <br/>zu Ingenieurmaßnahmen
      <br/><b>- Universität Karlsruhe (TH) -</b>
      <br/>Geodätisches Institut
     </td>
     . . .
    </tr>
   </table>
   <h2>Deformationsanalyse: Final 2000</h2>
  </center>

 <xsl:apply-templates></xsl:apply-templates>
 </body>
</html>
</xsl:template>

<xsl:template match="Titel[@id='1:']">
 <center>
  <blockquote>
   <b>Tabelle 1:</b>
   <i>
    <xsl:apply-templates select="Typ"></xsl:apply-templates> -
    <xsl:apply-templates select="Bezug"></xsl:apply-templates> -
    <xsl:apply-templates select="Epoche"></xsl:apply-templates> -
    <xsl:apply-templates select="Auswertung"></xsl:apply-templates>
   </i>
  </blockquote>
 </center>
</xsl:template>

<xsl:template match="Tabelle_Koordinate/geozentrisch[@id='Koordinate']">
 <center>
  <table border="1" cellspacing="1" cols="5" bgcolor="#FFCC99">
   <tr>
    <td align="center" width="20%"><b>PKNR</b></td>
    <td align="center" width="20%"><b>X-Achse</b></td>
    <td align="center" width="20%"><b>Y-Achse</b></td>
    <td align="center" width="20%"><b>Z-Achse</b></td>
    <td align="center" width="20%"><b>Einheit</b></td>
   </tr>
  <xsl:for-each select="read">
```

```
   <tr>
    <td align="center" width="20%"><xsl:value-of select="PKNR"/></td>
    <td align="center" width="20%"><xsl:value-of select="X"/></td>
    <td align="center" width="20%"><xsl:value-of select="Y"/></td>
    <td align="center" width="20%"><xsl:value-of select="Z"/></td>
    <td align="center" width="20%"><xsl:value-of select="Einheit"/></td>
   </tr>
  </xsl:for-each>
  </table>
 </center>
</xsl:template>

. . .

<xsl:template match="Titel[@id='6:']">
 <blockquote>
  <center>
   <b>Tabelle 6:</b>
   <i>
    <xsl:apply-templates select="Typ"></xsl:apply-templates> -
    <xsl:apply-templates select="Bezug"></xsl:apply-templates> -
    <xsl:apply-templates select="Epoche"></xsl:apply-templates> -
    <xsl:apply-templates select="Auswertung"></xsl:apply-templates>
   </i>
  </center>
 </blockquote>
</xsl:template>

<xsl:template match="Tabelle_Koordinate_Genauigkeit/UTM[@id='Genauigkeit']">
 <center>
  <table border="1" cellspacing="1" cols="4" bgcolor="#FFCC99">
   <tr>
    <td align="center" width="20%"><b>PKNR</b></td>
    <td align="center" width="30%"><b>Sigma(N)</b></td>
    <td align="center" width="30%"><b>Sigma(E)</b></td>
    <td align="center" width="20%"><b>Einheit</b></td>
   </tr>
  <xsl:for-each select="read">
   <tr>
    <td align="center" width="20%"><xsl:value-of select="PKNR"/></td>
    <td align="center" width="30%"><xsl:value-of select="sigmaN"/></td>
    <td align="center" width="30%"><xsl:value-of select="sigmaE"/></td>
    <td align="center" width="20%"><xsl:value-of select="Einheit"/></td>
   </tr>
  </xsl:for-each>
  </table>
 </center>
</xsl:template>

. . .

</xsl:stylesheet>
```

D.3 Klasse: FinalResultToXML

```
public void setXMLFinalResult(Vector v1, . . .) {
try {
  /** Festlegung der Variablen **/
  Comment ebene1, ebene2, ebene3;
  Element root;
  String koord_tab, geozentrisch, stationsname;
  String read, pknr, col_X, col_Y, col_Z, col_SI, tabid;
  int maxtab1, maxtab2;

  /** *** Tabelle 1: geozentrische Koordinaten *** **/
  tabid = " 1: ";
  setXMLKoordinateGeozentrisch(tabid);
  maxtab1 = numcol[0];
  for (i = 0; i < maxtab1; i++) {
    stationsname = (String) e1.nextElement();
    s2 = (String) e2.nextElement();
    . . .
    root.getChild(koord_tab).getChild(geozentrisch).addContent(
      new Element(read).setAttribute("id", stationsname).addContent(
      new Element(pknr).setText(stationsname)).addContent(
      new Element(col_X).setText(s2)).addContent(
      new Element(col_Y).setText(s3)).addContent(
      new Element(col_Z).setText(s4)).addContent(
      new Element(col_SI).setText(s5)));
  }

  /** *** Tabelle 2: geozentrische Genauigkeiten der Koordinaten *** **/
  tabid = " 4: ";
  setXMLGenauigkeitKoordinateGeozentrisch(tabid);
  maxtab2 = numcol[0] + numcol[1];
  for (i = maxtab1; i < maxtab2; i++) {
    stationsname = (String) e1.nextElement();
    s2 = (String) e2.nextElement();
    . . .
    root.getChild(sigma_tab).getChild(geozentrisch).addContent(
      new Element(read).setAttribute("id", stationsname).addContent(
      new Element(pknr).setText(stationsname)).addContent(
      new Element(col_sX).setText(s2)).addContent(
      new Element(col_sY).setText(s3)).addContent(
      new Element(col_sZ).setText(s4)).addContent(
      new Element(col_SI).setText(s6)));
  }
  . . .

  } catch (Exception e) {
    String except1 = e.getMessage();
  }
}

public void setXMLKoordinateGeozentrisch(String tabid) {
  ebene1 = new Comment("Element: Tabelle_Koordinate");
  root = doc.getRootElement().addContent(ebene1);
  root = doc.getRootElement().addContent(new Element(koord_tab));
    ebene2 = new Comment(
      "Element: Titel Attribut:table=Tabelle Attribut:id=$tabid");
    root.getChild(koord_tab).addContent(ebene2);
    root.getChild(koord_tab).addContent(
```

```
      new Element(titel)
        .setAttribute("table", "Tabelle").setAttribute("id", tabid));
      ebene3 = new Comment(
        "Elementenliste: <Typ /><Bezug /><Epoche /><Auswertung />");
      root.getChild(koord_tab).getChild(titel).addContent(ebene3);
      root.getChild(koord_tab).getChild(titel).addContent(
        new Element(typ).setText("geozentrische Koordinate (X, Y, Z)"));
      root.getChild(koord_tab).getChild(titel).addContent(
        new Element(bezug).setText(bezug1));
      root.getChild(koord_tab).getChild(titel).addContent(
        new Element(epoche).setText(epoche1));
      root.getChild(koord_tab).getChild(titel).addContent(
        new Element(auswertung).setText(auswertung1));
    . . .
  }

  public void setXMLGenauigkeitGeschwindigkeitGeozentrisch(String tabid) {
    ebene1 = new Comment("Element: Tabelle_Geschwindigkeit_Genauigkeit");
    root = doc.getRootElement().addContent(ebene1);
    root = doc.getRootElement().addContent(new Element(sigmav_tab));
      ebene2 = new Comment(
        "Element: Titel Attribut:table=Tabelle Attribut:id=$tabid");
      root.getChild(sigmav_tab).addContent(ebene2);
      root.getChild(sigmav_tab).addContent(
        new Element(titel)
          .setAttribute("table", "Tabelle").setAttribute("id", tabid));
        ebene3 = new Comment(
          "Elementenliste: <Typ /><Bezug /><Epoche /><Auswertung />");
        root.getChild(sigmav_tab).getChild(titel).addContent(ebene3);
        root.getChild(sigmav_tab).getChild(titel).addContent(
          new Element(typ)
            .setText("Genauigkeit der Geschwindigkeiten s(vX, vY, vZ)"));
        root.getChild(sigmav_tab).getChild(titel).addContent(
          new Element(bezug).setText(bezug1));
        root.getChild(sigmav_tab).getChild(titel).addContent(
        new Element(epoche).setText(epoche1));
        root.getChild(sigmav_tab).getChild(titel).addContent(
        new Element(auswertung).setText(auswertung1));
    . . .
  }
  . . .
}
```

Literaturverzeichnis

[1] ADKOLI, A. und R. VELPURI: *ORACLE 9i for Windows Handbook*. Oracle Press - exclusively from McGraw-Hill/Osborne, 2002.

[2] BACKSCHAT, M. und O. GARDON: *Enterprise JavaBeans. Grundlagen - Konzepte - Praxis. EJB 2.0/2.1*. Spektrum Akademischer Verlag, Heidelberg, Berlin, 2002.

[3] BARTELME, N.: *Geoinformatik Modelle Strukturen Funktionen*. Springer Verlag, Berlin, Heidelberg, 1995.

[4] BASLER, D.: *Anwendungsentwicklung mit Java*. C&L Computer- und Literaturverlag, Böblingen, 2001.

[5] BÜHLER, K.: *OPENGIS Reference Model*. Open GIS Consortium Inc., http://www.opengeospatial.org/ specs/?page =orm, März 2003.

[6] BÄHR, H.-P. und TH. VÖGTLE: *Digitale Bildverarbeitung - Anwendung in Photogrammetrie, Fernerkundung und GIS*. Herbert Wichmann Verlag, 4., völlig neubearbeitete und erweiterte Auflage, S. 217-234, Karlsruhe, 2005.

[7] BILL, R. und D. FRITSCH: *Grundlagen der Geo-Informationssysteme. Band 1: Hardware, Software und Daten*. Wichmann Verlag GmbH, Karlsruhe, 1991.

[8] BIRKENBIHL, K.: *XML-Architektur. Tutorial*. Logon Webday, Frankfurt, September 2003.

[9] BONJER, K.-P., M. RIZESCU, V. SOKOLOV, M. RADULIAN und B. GRECU: *Peak Ground Motion and Intensity Pattern of large and moderate intermediate depth Vrancea Earthquakes*. IAGA-IASPEI Abstract, Hanoi, 2001.

[10] BORN, J. und J. FIGURA: *Neue Dimension der Datenqualität. Auswirkungen der Zertifizierung von Geodaten für Anbieter, Kunden und Volkswirtschaft*. GeoBit, Seiten 25–27, 11/ 2002.

[11] BREZING, A. H. und W. BENNING: *Entwicklung eines Expertensystems zur wissensbasierten Deformationsanalyse*. AVN, Seiten 206–217, 6/ 2001.

[12] BURKE, B. und S. LABOUREY: *JBoss™ 3.2 Workbook for Enterprise JavaBeans™, 3rd Edition*. O'Reilly & Associates, Inc., 2003.

[13] CHAPMAN, D.B. und E.D. ZWICKY: *Building Internet Firewalls*. O'Reilly & Associates, Inc., Köln, 1995.

[14] CHAPPEL, D. und R. MONSON-HAEFEL: *Java Message Service*. O'Reilly & Associates, Inc., Köln, 2001.

[15] CHRISTENSEN, E., F. CURBERA, G. MEREDITH und S. WEERAWARANA: *Web Services Description Language (WSDL) 1.1*. W3C® Note, http://www.w3.org/TR/wsdl, March 2001.

[16] CLOETHINGH, S. A. P. L., E. BUROV, L. MATENCO, G. TOUSSAINT, G. BERTOTTI, P. A. M. ANDRIESSEN, W. J. R. WORTEL und W. SPAKMAN: *Thermo-mechanical controls on the model of continental collision in the SE Carpathians (Romania)*. Earth Planet, Sci. Lett. 218:57–76, 2004.

[17] DATE, C. J. und H. DARWEN: *SQL - Der Standard: SQL/92 mit den Erweiterungen CLI und PSM*. ADDISON-WESLEY, 1998.

[18] DEMICHIEL, L. G.: *Enterprise JavaBeans™ Specification, Version 2.1 Final Release*. Sun Microsystems, http://java.sun.com/products/ejb/docs.html, November 2003.

[19] DINTER, G.: *Generalisierte Orthogonalzerlegungen in der Ausgleichungsrechnung*. Doktorarbeit, Universität Karlsruhe (TH), DGK, Reihe C, Heft Nr. 559, München 2002.

[20] DINTER, G., M. NUTTO, G. SCHMITT, U. SCHMIDT, D. GHITAU und C. MARCU: *Three Dimensional Deformation Analysis with Respect to Plate Kinematics in Romania*. Reports on Geodesy, 2(57):29–42, Warschau, 2001.

[21] DINTER, G. und G. SCHMITT: *Three Dimensional Plate Kinematics in Romania*. Natural Hazards, 23:389–406, 2001.

[22] DRAGOI, O. A. und J. PRESTON: *The Conceptual Architecture of the Apache Web Server*. http://www. grad.math. uwaterloo.ca/∼oadragoi/CS746G/a1/apache_conceptual_arch.html, Januar 1999.

[23] ELLIS, J., L. HO und M. FISHER: *JDBC™ 3.0 Specification. Final Release*. Sun Microsystems, Inc., http://java. sun.com/products/jdbc/overview.html, Oktober 2001.

[24] GEHBAUER, G., M. MARKUS, F. FIEDRICH und C. SCHWEIER: *The Disaster Management Tool – A Multidisciplinary Approach to Earthquake Disaster Response*. In: *Proceedings of the Eleventh Annual Conference of the International Emergency Management Society*, Melbourne, Mai 2004.

[25] GRAY, J. und A. REUTER: *Transaction Processing: Concepts and Techniques*. Morgan Kaufman Verlag, 1993.

[26] GRIMM, S.: *Der Portalmarkt im Wandel. Portale als Instrument zur Kostensenkung - Funktion, Architektur, Vorgehen*. http://portal.wiso.uni-erlangen.de/wps/portal, Mai 2002.

[27] GUDGIN, M., M. HARDLEY, N. MENDELSOHN, J.-J. MOREAU und H. F. NIELSEN: *SOAP Version 1.2 Part 1: Messaging Framework*. W3C® Recommendation, http://www.w3.org/TR/2003/REC-soap12-part1-20030624, Juni 2003.

[28] HECK, B.: *Rechenverfahren und Auswertemodelle der Landesvermessung. Klassische und moderne Methoden*. Herbert Wichmann Verlag, 3., neu bearbeitete und erweiterte Auflage, Heidelberg, 2003.

[29] HOEVEN, A. G. A. VAN DER, B. A. C. AMBROSIUS, G. SCHMITT, G. DINTER, V. MOCANU und W. SPAKMAN: *GPS Probes the Kinematics of the Vrancea Seismogenic Zone*. EOS, 85(19):185–196, Mai 2004.

[30] HOUSLEY, R., W. POLK, W. FORD und S. SOLO: *Internet X.509 Public Key Infrastructure. Certificate and Certificate Revocation List (CRL) Profile*. http://www.ietf.org/rfc/rfc3280.txt, April 2002.

[31] HÖPCKE, W.: *Fehlerlehre und Ausgleichsrechnung*. Walter de Gruyter & Co., Berlin, New York, 1980.

[32] HUNTER, J. und B. MCLAUGHLIN: *Easy Java/XML Integration with JDOM. Part I and Part II*. Java World Magazine, http://www.javaworld.com/javaworld/jw-07-2000/jw-0728-jdom2.html, Juli 2000.

[33] HUNTER, J. und B. MCLAUGHLIN: *JDOM v 1.0 API Specification*. http://www.jdom.org/docs/apidocs/index.html, 2004.

[34] ISO, 19104: *Geographic information - Terminology*. Beuth Verlag GmbH, Oktober 2002.

[35] ISO, 9075: *Database Language SQL*. Document ISO/IEC 9075:1987, Genf, 1987.

[36] ISO, 9075: *Database Language SQL*. Document ISO/IEC 9075:1989, Genf, 1989.

[37] ISO, 9075: *Information Technology - Database Languages - SQL - Technial Corrigendum 2*. Document ISO/IEC 9075:1992/Cor.2:1996(E), Genf, 1992.

[38] ISO, 9075: *Information Technology – Database languages – SQL – Part 1:Framework*. Document ISO/IEC 9075-1:1999, Genf, 1999.

[39] ISO, DIN EN ISO 8402: *Quality Management and Quality Assurance - Vocabulary*. Beuth Verlag GmbH, August 1995.

[40] ISO, DIN EN ISO 9000: *Qualitätsmanagementsysteme - Grundlagen und Begriffe (ISO 9000:2000)*. Beuth Verlag GmbH, Dezember 2000.

[41] ISO, DIN ISO 7498: *Informationsverarbeitung Offener Systeme, Basis-Referenzmodell*. Beuth Verlag GmbH, 1982.

[42] ITRF: *ITRF Site Information and Selection*. http://www.iers.org/iers/products/itrf/itrf.html, Mai 2005.

[43] JBOSS: *JBoss 3.0 Documentation*. http://www.huihoo.com/jboss/online_manual/3.0/index.html, Mai 2002.

[44] JÄGER, R. und H. KALTENBACH: *Spectral analysis and optimization of geodetic networks based on eigenvalues and eigenfunctions*. Manuscripta Geodaetica, 15:302–311, 1990.

[45] JOOS, G.: *Zur Qualität von objektstrukturierten Geodaten*. Schriftenreihe des Studiengangs Geodäsie und Geoninformation der Universität der Bundeswehr München, Heft 66, Neubiberg, 2000.

[46] RZ, KARLSRUHE: *Grundeinstellung für die Firewallsysteme der Universitätsinstitute*. Rechenzentrum der Universität Karlsruhe (TH), http://www.rz.uni-karlsruhe.de/dienste/, Juni 2005.

[47] KEMPER, A. und A. EICKLER: *Datenbanksysteme*. Oldenbourg Verlag, 4. Auflage, München, 2001.

[48] KIENZLE, A., D. HANNICH, W. WIRTH, V. CIUGUDEAN, J. ROHN und K. CZURDA: *Seismic Microzoning of Bucharest. A GIS – Supported Analysis of Earthquake Hazard*. In: *Proceedings Volume, Int. Conference on Earthquake Loss Estimation and Risk Reduction*, Bucharest, Oktober 2002.

[49] KRÜGER, G.: *GoTo Java™ 2 - Handbuch der Java-Programmierung*. Addison Wesley Verlag, 2. Auflage, München, 2000.

[50] LANG, S. M. und P. C. LOCKEMANN: *Datenbankeinsatz*. Springer-Verlag, Berlin, Heidelberg, 1995.

[51] LIU, J.: *Research Project: An Analysis of JBoss Architecture*. http://www.cs.usyd.edu.au/∼jennyliu/ jboss.html, April 2002.

[52] LOCKEMANN, P. C. und K.R. DITTRICH: *Architektur von Datenbanksystemen*. Dpunkt-Verlag, 2004.

[53] LONEY, K. und G. KOCH: *Oracle 8i. Die umfassende Referenz*. Carl Hanser Verlag, München, Wien, 2001.

[54] MATENCO, L. und G. BERTOTTI: *Tertiary tectonic evolution of the external East Carpathians (Romania)*. Tectonophysics, 316:255–286, 2000.

[55] MATENCO, L., G. BERTOTTI, S. CLOETINGH und C. DINU: *Subsidence analysis and tectonic evolution of the external Carpathian - Moesian Platform region during Neogene times*. Sedimentary Geology, 156:71–94, 2003.

[56] MICHELS, J.-E., K. KULKARNI, C. M. FARRAR, A. EISENBERG, N. MATTOS und H. DARWENDOI: *SQL-Standard*. IT - Information Technology, 45(1):30–38, Februar 2003.

[57] MIDDENDORF, S., R. SINGER und J.HEID: *Java™ Programmierhandbuch und Referenz für die Java™ 2 Plattform*. Dpunkt-Verlag, 3. Auflage, Heidelberg, 2002.

[58] MINTERT, S.: *XML & Co. Die W3C-Spezifikationen für Dokumenten- und Datenarchitektur*. Addison Wesley Verlag, München, 2002.

[59] MÜLLER, B., B. SPERNER, J. DIRKZWAGER, O. HEIDBACH, A. ISMAIL-ZADEH, A. WÜSTEFELD, S. HETTEL, M. NEGUT und F. POLLITZ: *Teilprojekt A6 - Rezentes Spannungsfeld und Geodynamik*. Sonderforschungsbereich 461 - Starkbeben: von geowissenschaftlichen Grundlagen zu Ingenieurmassnahmen - Universität Karlsruhe (TH), S.137-178, Berichtsband 2002-2004.

[60] MONSON-HAEFEL, R.: *Enterprise JavaBeans*. O'Reilly Verlag, 3. Auflage, Köln, 2002.

[61] NIEMEIER, W.: *Ausgleichungsrechnung*. Walter de Gruyter, Berlin, New York, 2001.

[62] PERNUL, G. und R. UNLAND: *Datenbanken im Unternehmen. Analyse, Modellbildung und Einsatz*. Oldenbourg Verlag, München, Wien, 2001.

[63] PERRONE, P. J., S. R. VENKATA, R. CHAGANTI und T. SCHWENK: *J2EE Developer's Handbook*. Sams Verlag, 1. Auflage, Indiana, Juni 2003.

[64] PEUQUET, D. J.: *Making Space for Time: Issues in Space-Time Data Representation*. Geoinformatica, 5(1):11–32, 2001.

[65] POHLMANN, N.: *Firewall-Systeme. Sicherheit für Internet und Intranet*. International Thomson Verlag, 1. Auflage, Bonn, 1997.

[66] PUPPE, F.: *Einführung in Expertensysteme*. Springer Verlag, 2. Auflage, Berlin, 1991.

[67] QUINT, F.: *Kartengestützte Interpretation monokularer Luftbilder*. Doktorarbeit, Universität Karlsruhe (TH), DGK, Reihe C, Heft Nr. 477, München 1997.

[68] RAWIEL, P.: *Dreidimensionale kinematische Modelle zur Analyse von Deformationen an Hängen*. Doktorarbeit, Universität Karlsruhe (TH), DGK, Reihe C, Heft Nr. 533, München 2001.

[69] RAY, E. T.: *Learning XML*. O'Reilly & Associates, Inc., 1. Auflage, Köln, 2001.

[70] RIVEST, R. L., A. SHAMIR und L. M. ADLEMAN: *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. Communications of the ACM, 21(2):120–126, Februar 1978. http://citeseer.ist.psu.edu/rivest78 method.html.

[71] SCHLÜTER, W.: *Ziele des Programms 2001-2005*. Workshop 2004 der Forschungsgruppe Satellitengeodäsie, http://www.wettzell.ifag.de/veranstaltungen/fgs/workshop2004b/praesentationen.html, Oktober 2004. Session 5: Entwicklungen in Wettzell, Concepcion und O'Higgins (Instrumente, Datenmanagement, Beobachtungen).

[72] SCHMITT, G.: *Spectral analysis and optimization of two dimensional networks*. Geomatics Research Australasia, 1(67):47–64, Dezember 1997.

[73] SESTER, M., M. BUTENUTH, G. V. GÖSSELN, C. HEIPKE, S. KLOPP, U. LIPECK und D. MANTEL: *New Methods for Semantic and Geometric Integration of Geoscientific Data Sets with ATKIS - Applied to Geo-Objects from Geology and Soil Science*. Geotechnologien Science Report, Part 2, Seiten 51–62, Potsdam, 2003.

[74] SPERNER, B., D. IOANE und R. J. LILLIE: *Slab behaviour and its surface expression: New insights from gravity modelling in the SE-Carpathians*. Tectonophysics, 382(1-2):51–84, April 2004.

[75] SPERNER, B., F. LORENZ, K. BONJER, S. HETTEL, B. MÜLLER und F. WENZEL: *Slab break-off - abrupt cut or gradual detachment? New insights from the Vrancea Region (SE Carpathians, Romania)*. Terra Nova, 13:172–179, 2001.

[76] STARK, S. und THE JBOSS GROUP: *JBOSS Administration and Development. Third Edition (3.2.x Series)*. JBoss Group, sales@jbossgroup.com, June 2003.

[77] STONEBRAKER, M. und D. MOORE: *Object-Relational DBMSs: The Next Great Wave*. Morgan Kaufman Verlag, 1996.

[78] SUN MICROSYSTEMS, INC.: *Java Naming and Directory Interface™ Application Programming Interface (JNDI API)*. http://java.sun.com/products/jndi/reference/api/index.html, July 1999.

[79] SUN MICROSYSTEMS, INC.: *Java™ Message Service Specification Version 1.1*. http://java.sun.com/products/jms/docs.html, April 2002.

[80] SUN MICROSYSTEMS, INC.: *Java™ 2 Platform Enterprise Edition v 1.4 API Specification*. http://java.sun.com/j2ee/1.4/docs/api/index.html, 2003.

[81] SUN MICROSYSTEMS, INC.: *Java™ 2 Platform Std. Ed. v 1.4.2*. http://java.sun.com/j2se/1.4.2/docs/api/overview-summary.html, Release June 25, 2003.

[82] THAU, R. S.: *Apache API notes*. http://modules.apache.org/doc/API.html, Juni 2005.

[83] URMAN, S.: *Oracle9i. PL/SQL-Programmierung*. Carl Hanser Verlag, München, Wien, 2002.

[84] VOSSEN, G.: *Datenbankmodelle, Datenbanksprachen und Datenbankmanagementsysteme.* Oldenbourg Verlag, 4. Auflage, München, 2000.

[85] W3C: *The Platform for Privacy Preferences 1.0 (P3P1.0) Specification*. http://www.w3.org/TR/P3P, April 2002.

[86] W3C: *Extensible Markup Language (XML) 1.0 (Third Edition)*. http://www.w3.org/TR/REC-xml, Februar 2004.

[87] WEIMERSKIRCH, A.: *Authentikation und Identifikation in Ad-hoc und Sensor Netzen*. In: *Forschungsverbund Datensicherheit NRW*, Bochum, Dezember 2003. EUROBITS.

[88] WORTEL, M. J. R. und W. SPAKMAN: *Subduction and Slab Detachment in the Mediterranean-Carpathian Region*. Science, 290(5498):1910–1917, Dezember 2000.

[89] WUNDERLICH, TH.: *Einsatz mobiler GIS in der Ingenieurpraxis - Trends und Perspektiven*. In: *Proceedings zum Fortbildungsseminar Geoinformationssysteme*, TU München, März 2000.

[90] ZIPF, A. und S. KRÜGER: *Flexible Modellierung und Verwaltung temporaler 3D-Geodaten*. GeoBIT, 6:20–27, 12/2001.

In der vorliegenden Arbeit wird ein internetbasiertes Fachinformationssystem zur Detektion krustaler Deformationen in den Ostkarpaten vorgestellt, welches im Rahmen des Sonderforschungsbereichs 461 „Starkbeben: von wissenschaftlichen Grundlagen zu Ingenieurmaßnahmen“ entwickelt wurde.

Im Vordergrund stehen die Konzeption und Implementierung eines Fachinformationssystems für regionale beziehungsweise globale GPS-Projekte, die zur Aufdeckung von Plattengrenzen und Analyse von Plattenbewegungen eingerichtet wird. Dabei werden verschiedene Aspekte berücksichtigt. Der Zugang zu relevanten Projektinformationen via Internet soll einer breiten Öffentlichkeit unter besonderer Berücksichtigung der verschlüsselten Datenübertragung bereitgestellt werden. Eine sichere und zuverlässige Datenverwaltung wird mittels einer objektrelationalen Datenbank realisiert. Zur schnellen und sicheren Kommunikation zwischen Client und Server dient der Aufbau einer plattformunabhängigen, mehrschichtigen Architektur. Es wird ein Applikationsserver eingerichtet, der ausführungsunabhängige Projektkomponenten für die Entwicklung von Objekten und Methoden komplexer Geschäftsanwendungen zur Verfügung stellt. Ein einheitliches Austauschformat soll zur Datenübertragung mittels eines standardisierten Werkzeuges – XML – zur Verwendung von Auszeichnungssprachen entworfen werden. Wesentliche Ergebnisse aus den geodätischen Analysen und Auswertungen liegen zur graphischen Darstellung im FIS unter Verwendung der Java 2D API vor.

Unter Berücksichtigung der oben genannten Aspekte wird ein Fachinformationssystem zur Dokumentation des 9-jährigen Projektverlaufs (1996-2004) im Teilprojekt B1 „Dreidimensionale Plattenkinematik in Rumänien“ aufgebaut, welches sich aus nicht kommerziellen Komponenten zusammensetzt.

ISBN-13: 978-3-86644-063-0
ISBN-10: 3-86644-063-4

www.uvka.de