Repository-Technologie am Beispiel des Microsoft Repositories

Seminarvortrag von

Rasmus Hofmann

4. Februar 2000

Übersicht

- Motivation
- Allgemeine Definition
  - Bestandteile
  - Dienste
- Microsoft Repository
  - Typmodell
  - Objektorientierte Schnittstellen
  - Dienste
  - Open Information Model

Motivation

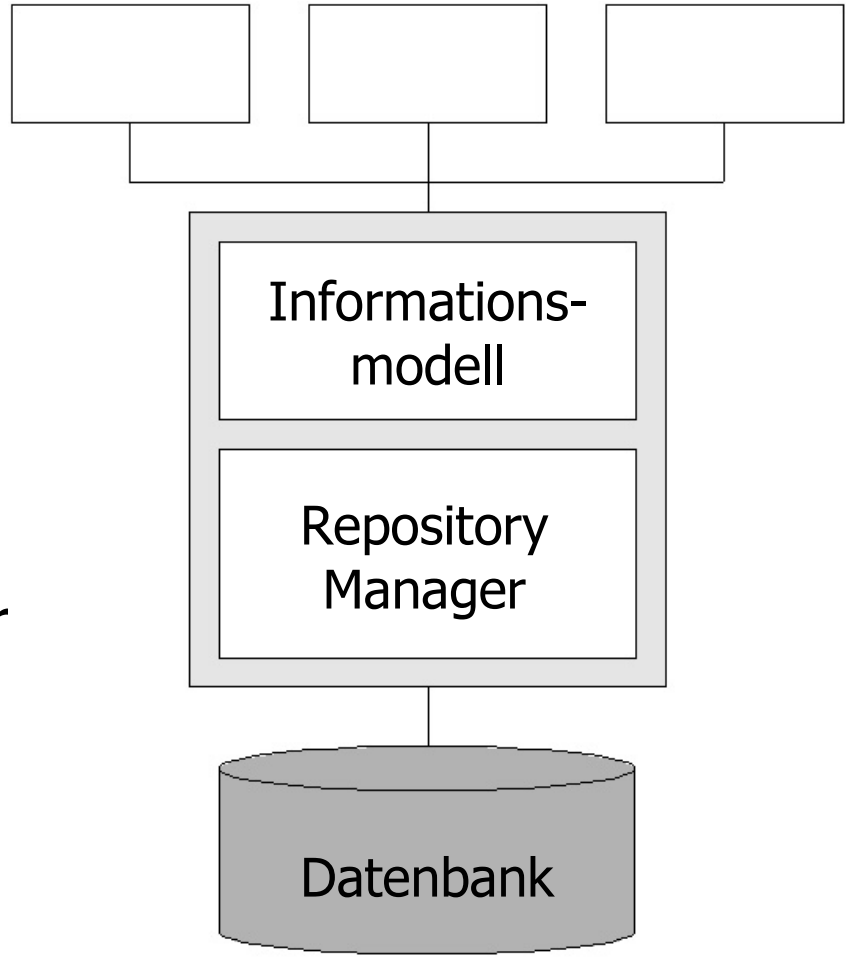
- Wozu braucht man Repositories?

- Relevanz der Metadaten-Verwaltung
  - Software-Entwicklung, Wiederverwendung
  - Datenintegration, Data Warehouse
  - BPR, CASE usw.

- Eigenschaften
  - Dynamisch
  - Team-Orientiert

Definition

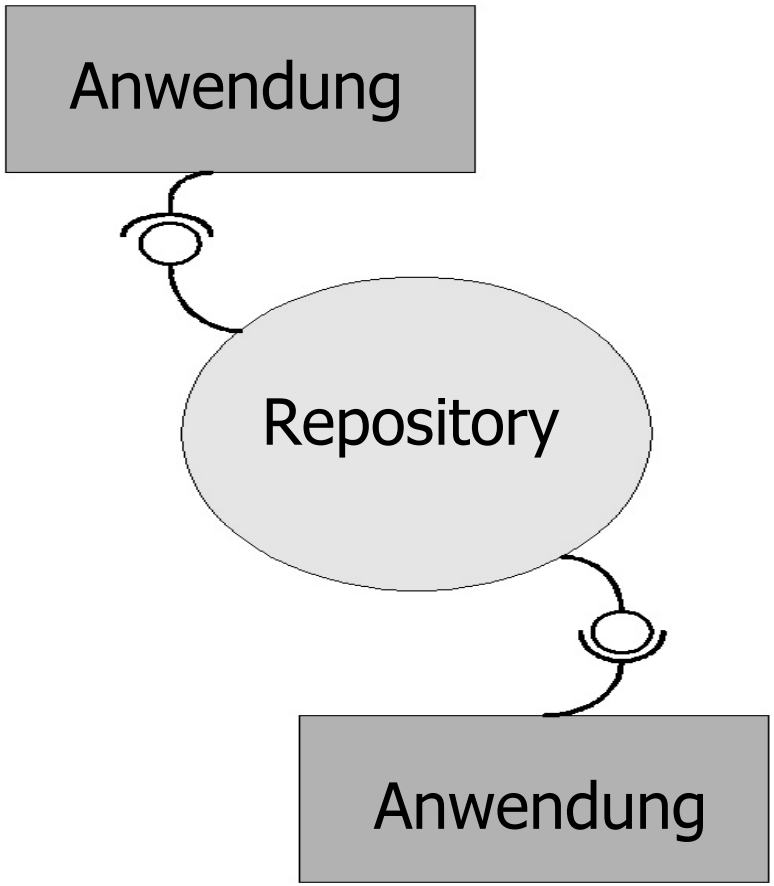
- Aufgabe
  - Speicherung und
  - Verwaltung von Metadaten
- Bestandteile
  - Repository Manager
  - Informationsmodell



Dienste (nach Bernstein)

- Schnittstellen
- Checkin/Checkout
- Versionsverwaltung
- Konfigurationsverwaltung
- Notifikation
- Kontextverwaltung
- Workflow Control

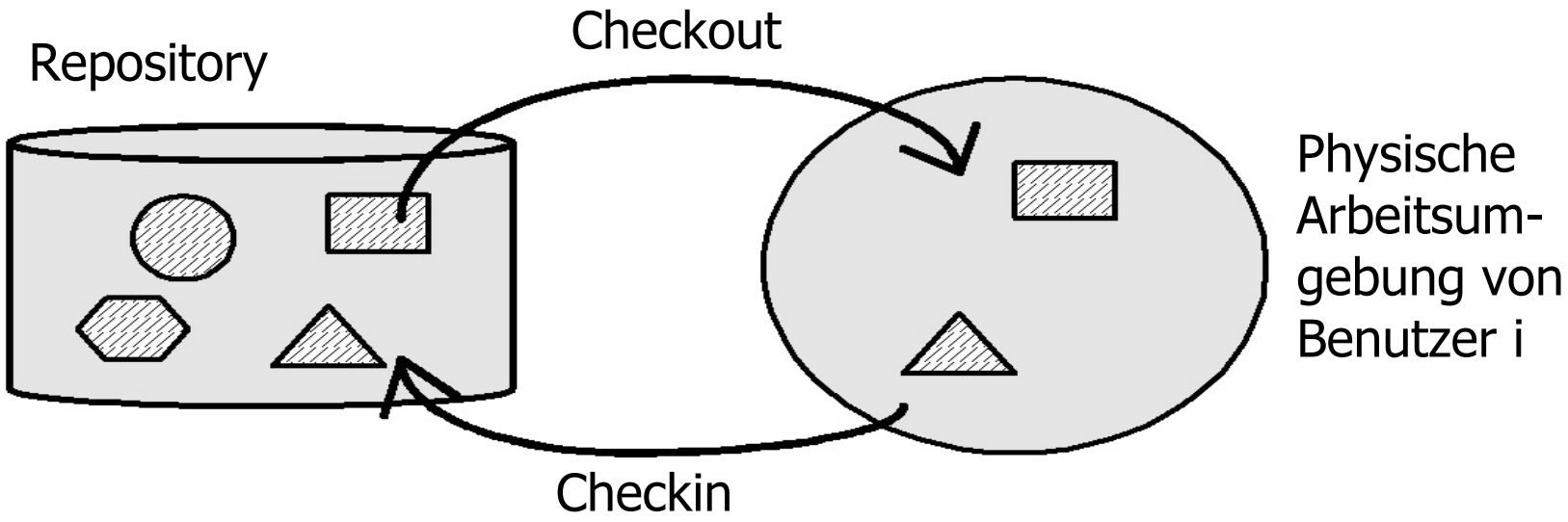
Schnittstellen



Anforderungen:

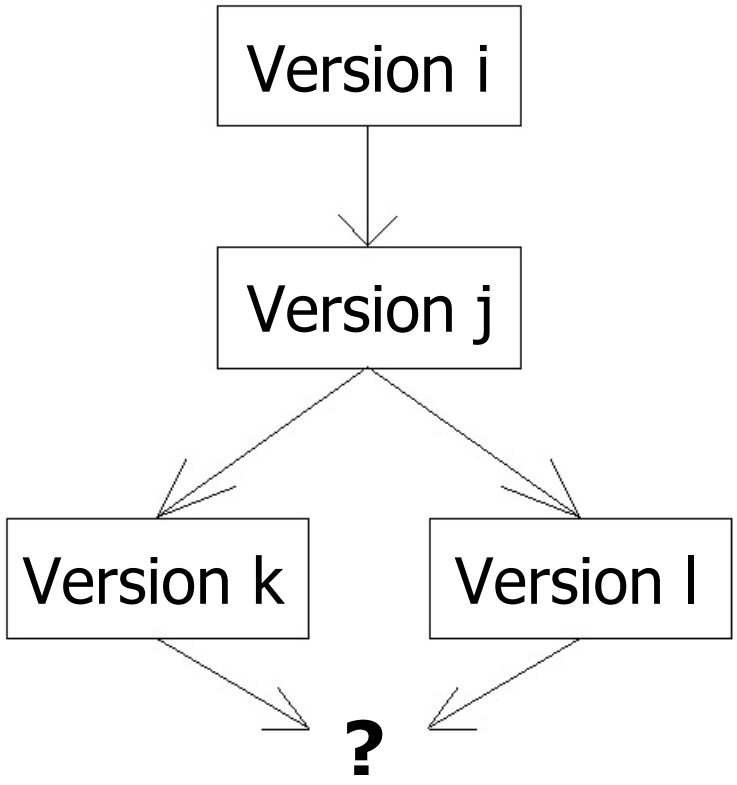
- Navigation in den Objekten (object-at-a-time)
  auch: Anfragen
- Klassen, die Eigenschaften modellieren, z.B. Version, Konfiguration, Regel
- Unabhängig von der Implementierungssprache

Checkin/Checkout



- Aufgaben / Zweck / Wunsch:
  - Kurzer Zugriffsweg
- Probleme:
  - inkonsistente Kopien

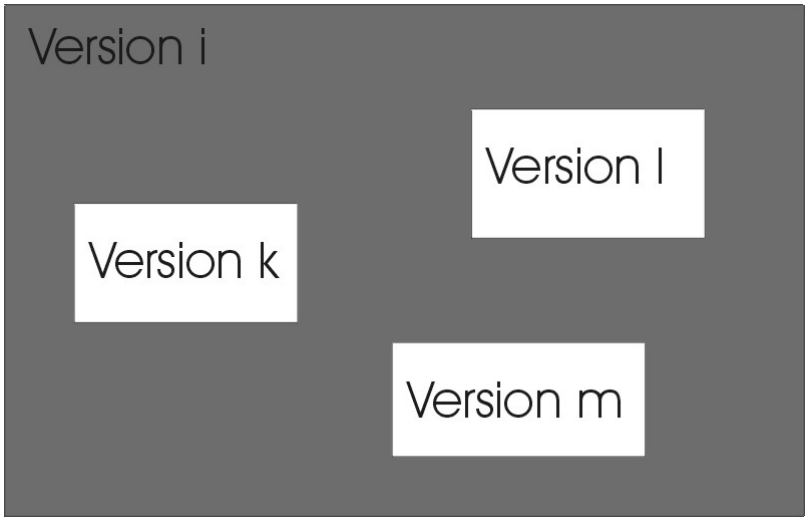
Versionsverwaltung



Anforderungen:

- Identifikation von Versionen möglich
- Mechanismus zur Benennung
- Vereinigungs-mechanismus

Konfigurationsverwaltung

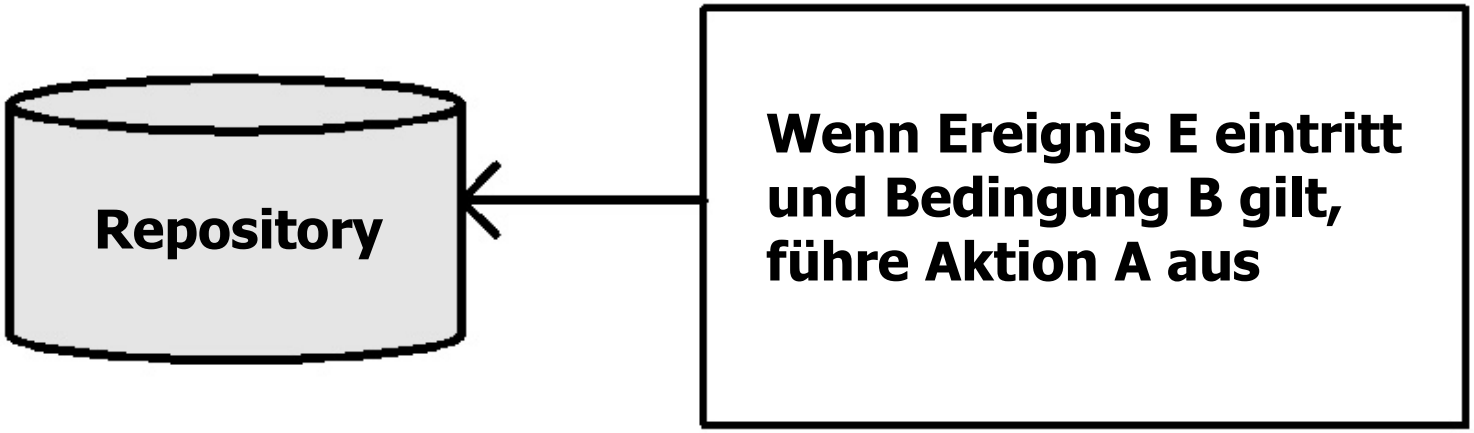


Anforderungen:

- Bindung bestimmter Versionen an eine Konfiguration explizit möglich
- Regeln festlegbar (implizite Bindung)
- Auswirkung von Änderungen auf die Konfiguration

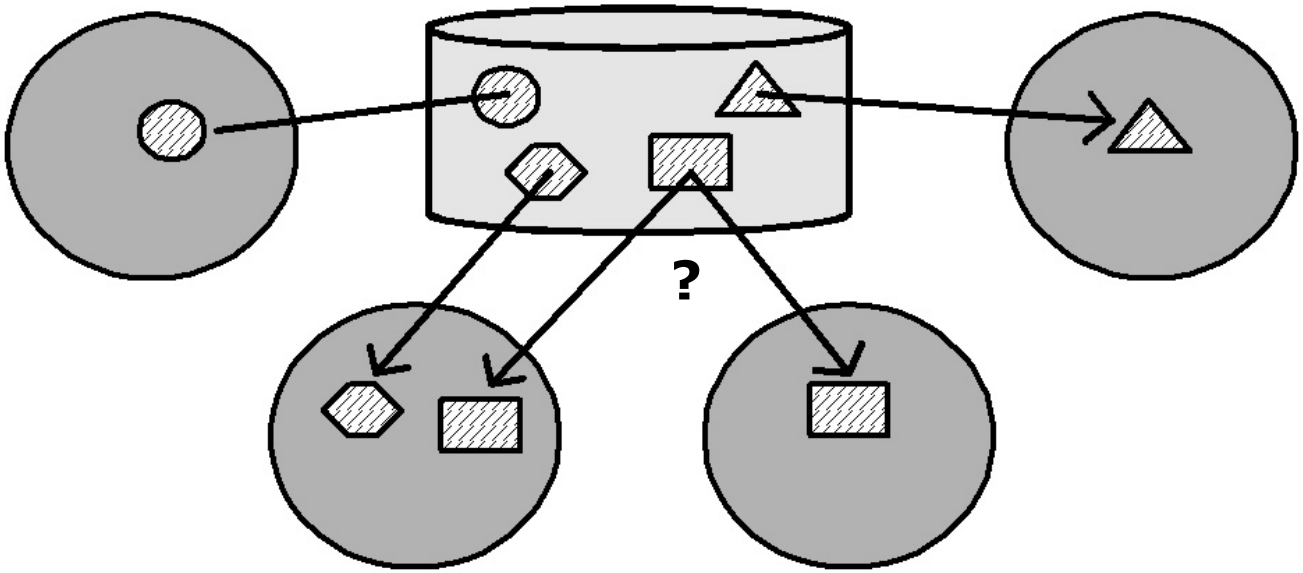
Notifikation

- Operationen automatisieren
- Nachrichten versenden



Beispiele:
Wenn ein bestimmtes Objekt geändert wurde, benachrichtige Benutzer i
Wenn ein hierarchisches Objekt gelöscht wurde, lösche die Komponenten

Kontextverwaltung



Operationen:

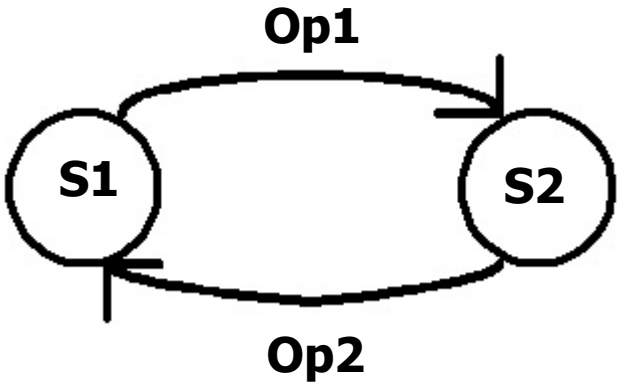
- Erzeugen, Laden, Speichern

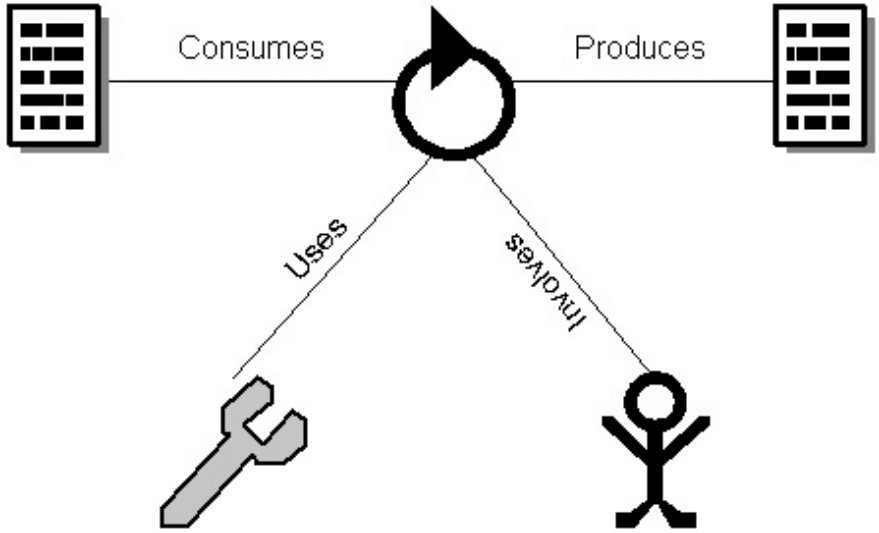
Zweck:

- Komplexität verringern
- Schutzmechanismus

Workflow control

- Status von Objekten
- Vorgegebener Prozeß





Microsoft Repository

- Bestandteile:
  - Repository Engine
  - Open Information Model
  - Diverse Werkzeuge

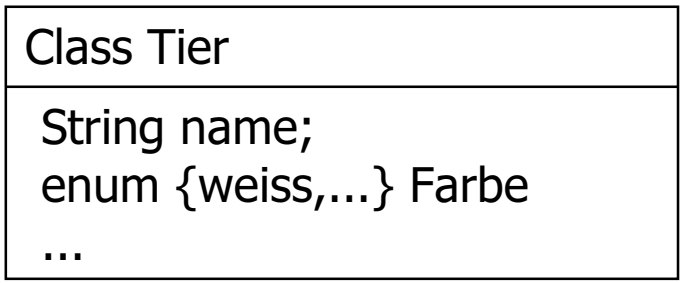
- Anforderungen
  - Kompatibilität zu COM
  - Erweiterbarkeit
  - Versions- & Konfigurationsverwaltung

Typmodell

Typmodell

außerdem: RelationshipDef, PropertyDef, CollectionDef

Informationsmodell



Daten

ID	Name	Farbe	...
1	Frosch	grün	...

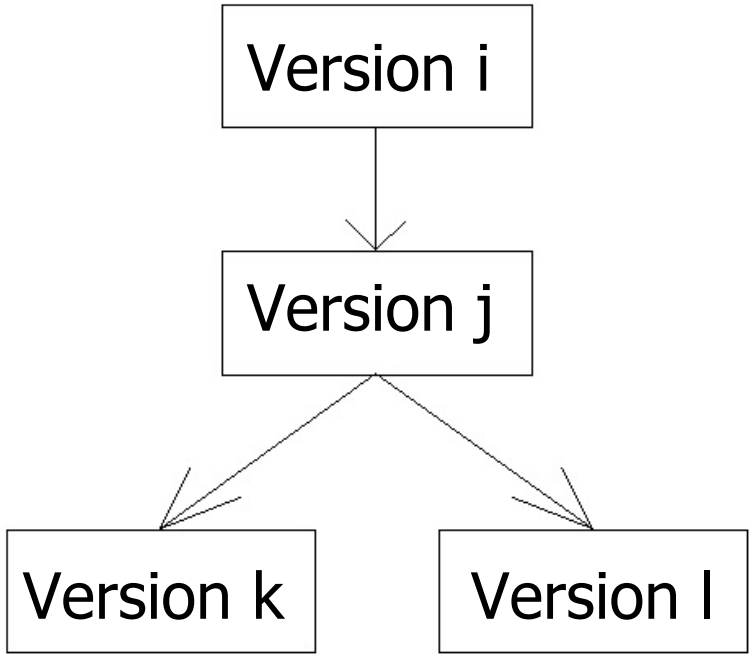
Schnittstellen

- Repository Session
  Methoden: Open, New
- Repository Object
  modelliert ein Objekt im Repository
- Relationship Object
  modelliert eine Beziehung (mit Feldern)
- Collection Object
  Methoden: Add, Count, Remove, Item, _NewEnum

Navigation

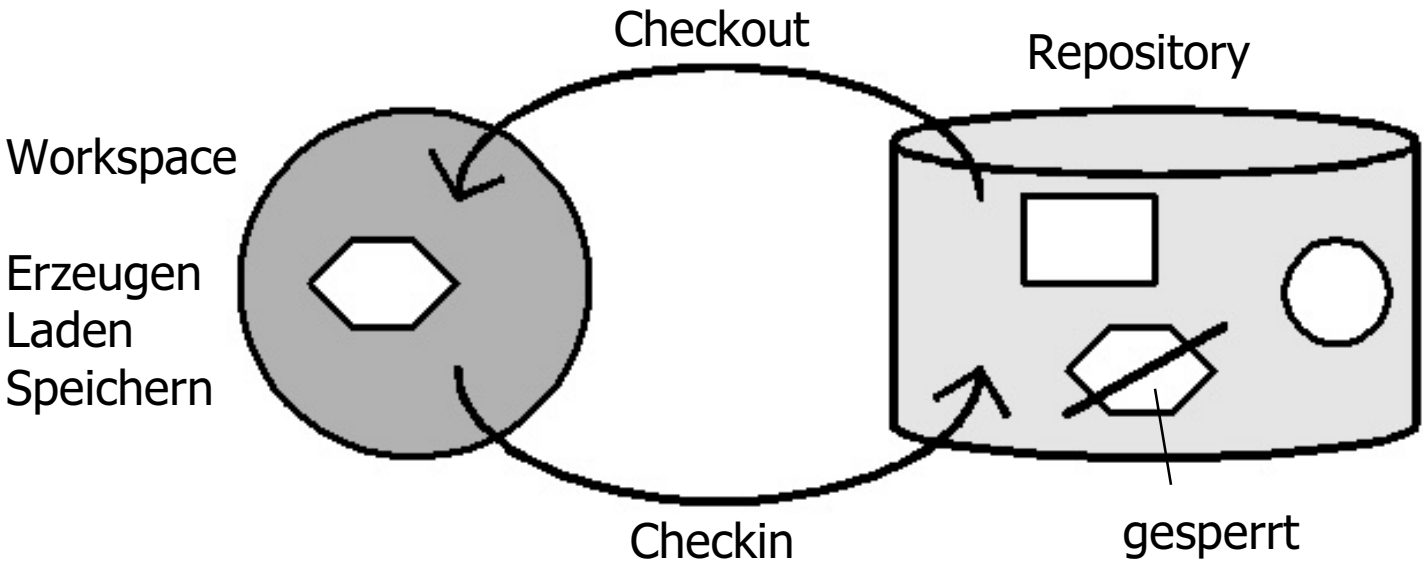
- RepositorySession RS = new RepositorySession();
  RepositoryObject root = RS.open(Dateiname);
- Collection col = root.relationshipX();
- Collection col = RS.ObjectInstances(Klassenname);
- Collection col = RS.query(SQL-Anfrage);
- RepositoryObject o = RS.getId(Identifikation);

Versionsverwaltung



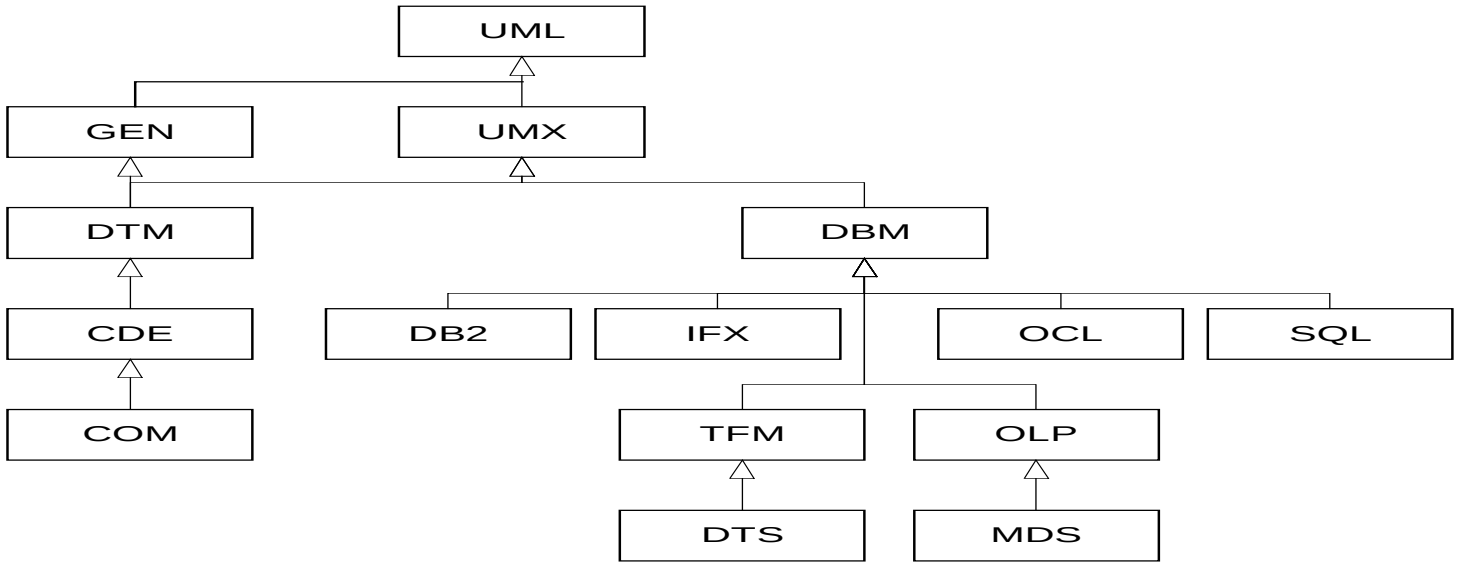
- Eindeutige Identifikation
- frozen und unfrozen -> Delta Storage
- Vereinigungsoperation

Kontextverwaltung



- Workspaces sind „virtuelle" Repositories
- Können nicht verschachtelt werden
- Enthalten max. eine Version eines Objektes

Open Information Model



- 250 Schnittstellen und 100 Beziehungen
- Unterteilt in Submodelle
- Kennzeichnung der Submodelle:
  - z.B.: IDbmSchema

Unified Modelling Language

- Industriestandard
- grafische Notation für Analyse, Spezifikation und Design von Softwareartefakten
- andere Modelle davon abgeleitet, z.B.:

  IUmlPackage - owns - IUmlElement

  IDbmCatalog - contains - IDbmSchema

Zusammenfassung

- Repositories dienen der Verwaltung von Metadaten
- Bestandteile:
  - Repository Manager
  - Informationsmodell
- Vergleich zwischen allgemeinen Anforderungen und der Microsoft Repository