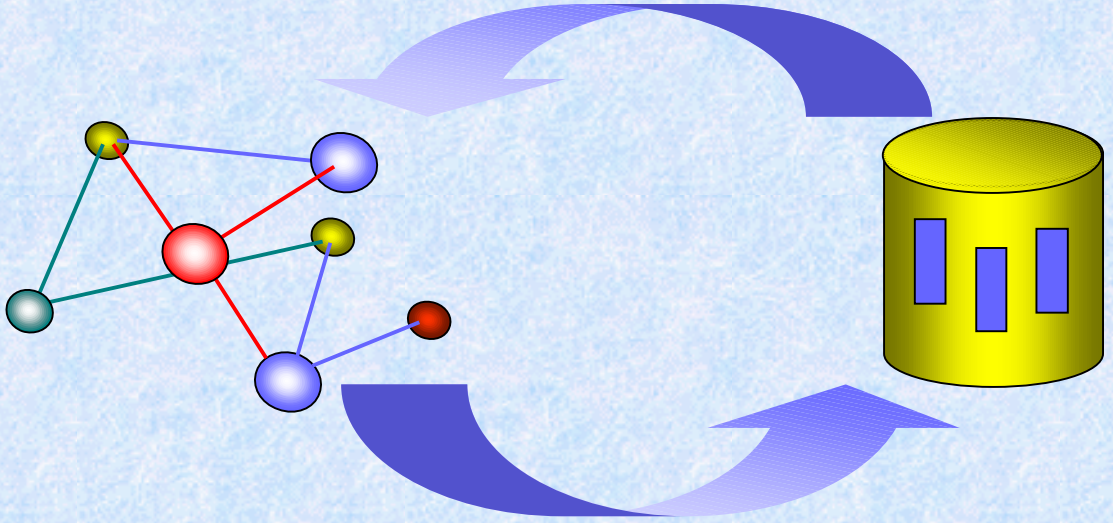
Effizienzsteigerung durch Mapping-Tools bei der Integration von RDBMS in Java-Anwendungen



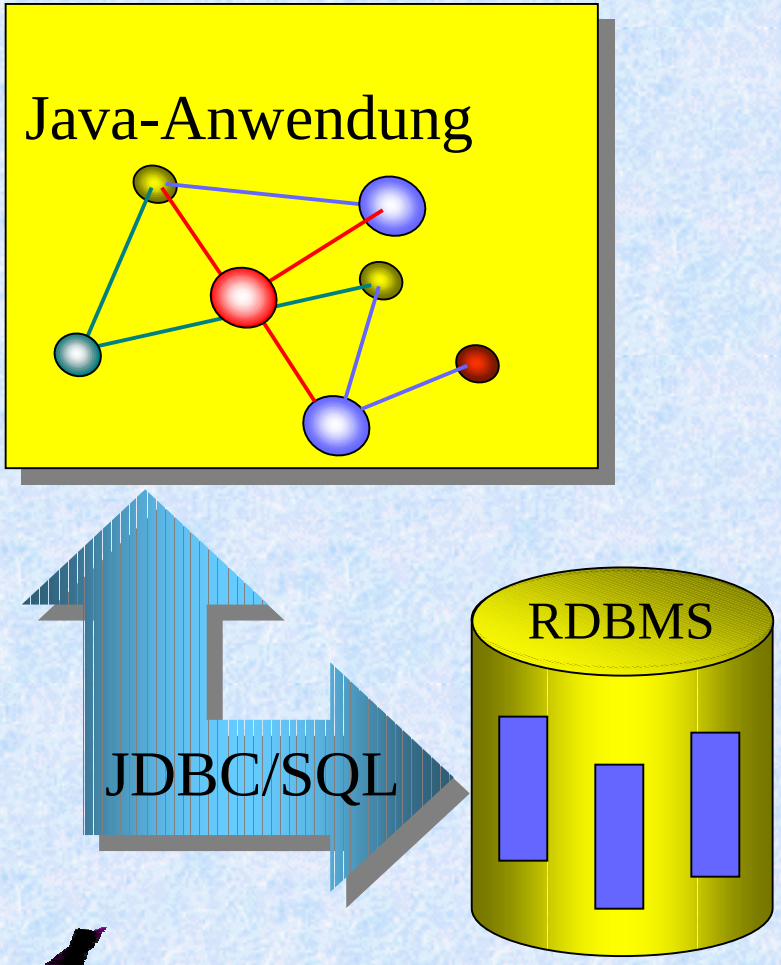
ARS NOVA Software GmbH
Klaus Kiehne

Java Forum, 28. Juni, 9:45 - 10:30

Inhalt

- *Aufgaben und Fähigkeiten*
- *Einsatz- und Auswahlkriterien*
- *Auswirkungen auf die Architektur*
- *Auswirkungen auf die Entwicklung*
- *Performanzoptimierung mit Mapping-Tools*
- *Hinweise für das Projektmanagement*
- *Zusammenfassung*

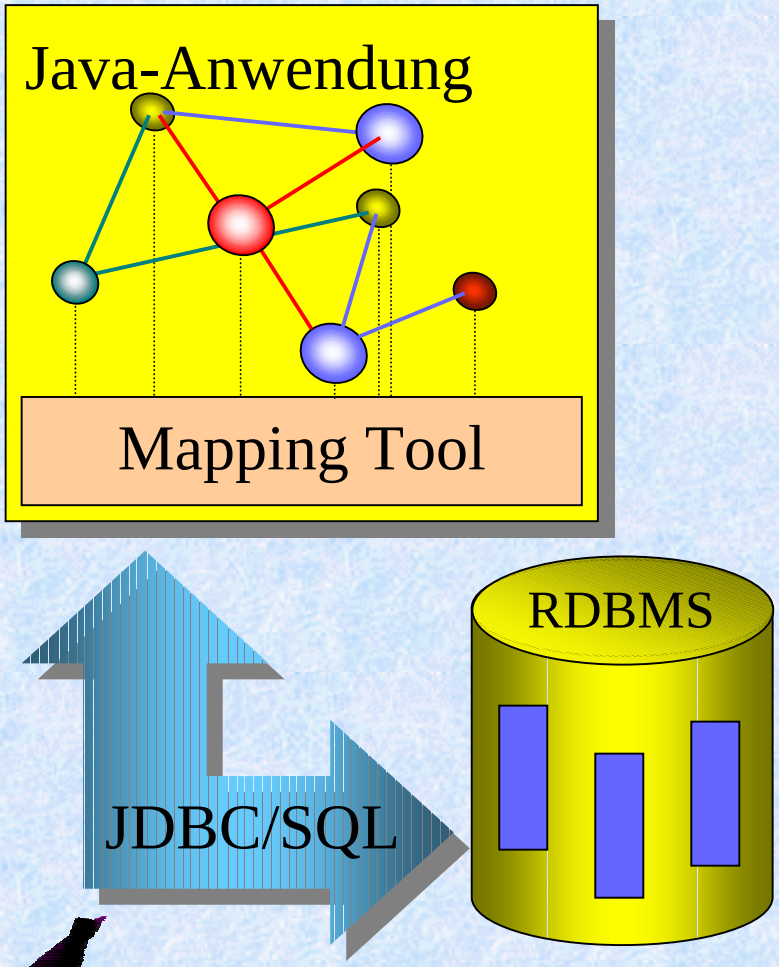
Aufgabe eines Mapping-Tools ...



Ohne Mapping-Tool:

```
rs = c.createStatement().executeQuery(
     “select n, p, a from P“);
Person p = new Person(rs.getString(1),
     rs.getString(2), rs.getInt(3));
rs.close();
rs = c.createStatement().executeQuery(
     “select nr,str from address where “
     + “id = “ + p.a);
Address a = new Address(rs.getInt(1),
     rs.getString(2));
rs.close();
```

Effiziente Implementierung!



Mit Mapping-Tool:

```
Person p = tool.getObject(Person.class);
Address a = p.getAddress();
```

Charakterisierende Eigenschaften von Mapping-Tools

- Abbildung von Geschäftsobjekten auf Entitäten
- Unterstützung von Transaktionen und Sperrstrategien
- automatische Generierung von SQL-Statements
- Wahrung der Objektidentität
- Optimierung von Objektzugriffen durch Caching
- verzögertes Lesen von Objekten (read on demand)
- konsistentes Löschen verknüpfter Objekte
- Unterstützung mehrschichtiger, verteilter Anwendungen

Generelle Kriterien für den Einsatz eines Mapping-Tools

- Die Persistenz soll mittels RDBMS gelöst werden
- Die Anwendung besitzt ein Geschäftsobjektmodell
  - Geschäftsobjekte bilden Hierarchien von Klassen
  - Vielfältige Beziehungen zwischen Geschäftsobjekten
- Die Anwendung bearbeitet Geschäftsobjekte
- Eine ‚nicht triviale' Auswertungslogik
- Die Änderbarkeit oder Erweiterbarkeit des Geschäftsobjektmodells wird benötigt

„Konzentration auf Anwendungslogik vs. Infrastruktur"

Auswahlkriterien für ein Produkt (1)

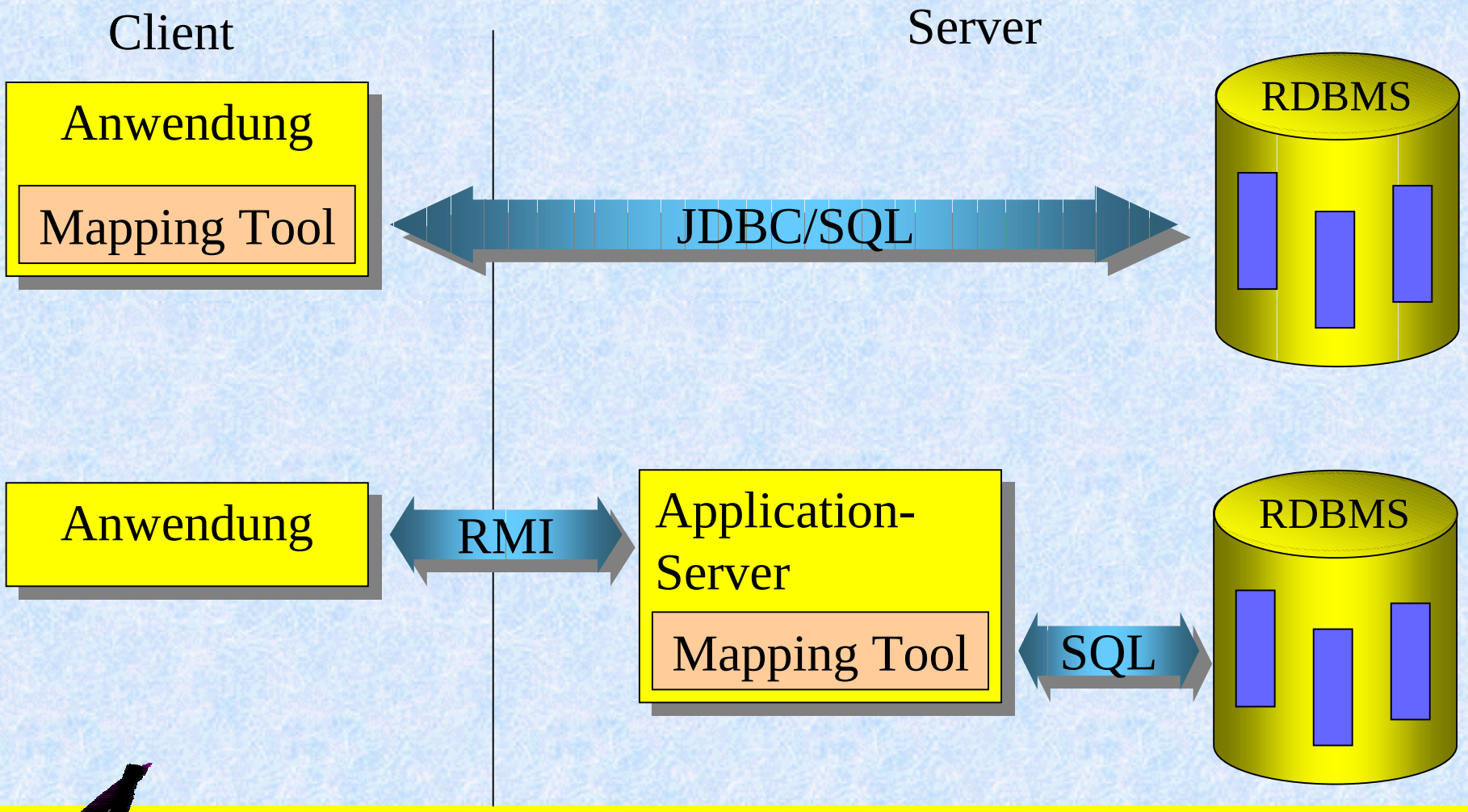
- Ist der Mapping-Mechanismus mächtig genug?
  - Unterstützung von 1:1-, 1:n-, n:m-Beziehungen
  - Unterscheidung von Aggregation und Assoziation
  - Unterstützung selbst definierter Basistypen
  - Alternativen für die Abbildung von Klassenhierarchien
  - z.B. Java: Unterstützung von Interfaces als Attributtyp

- Wie sieht das Transaktionskonzept aus?
- Gibt es Eingriffsmöglichkeiten zur Optimierung?
- Welche Locking-Strategien werden unterstützt?
- Werden die benötigten Architekturtypen unterstützt?

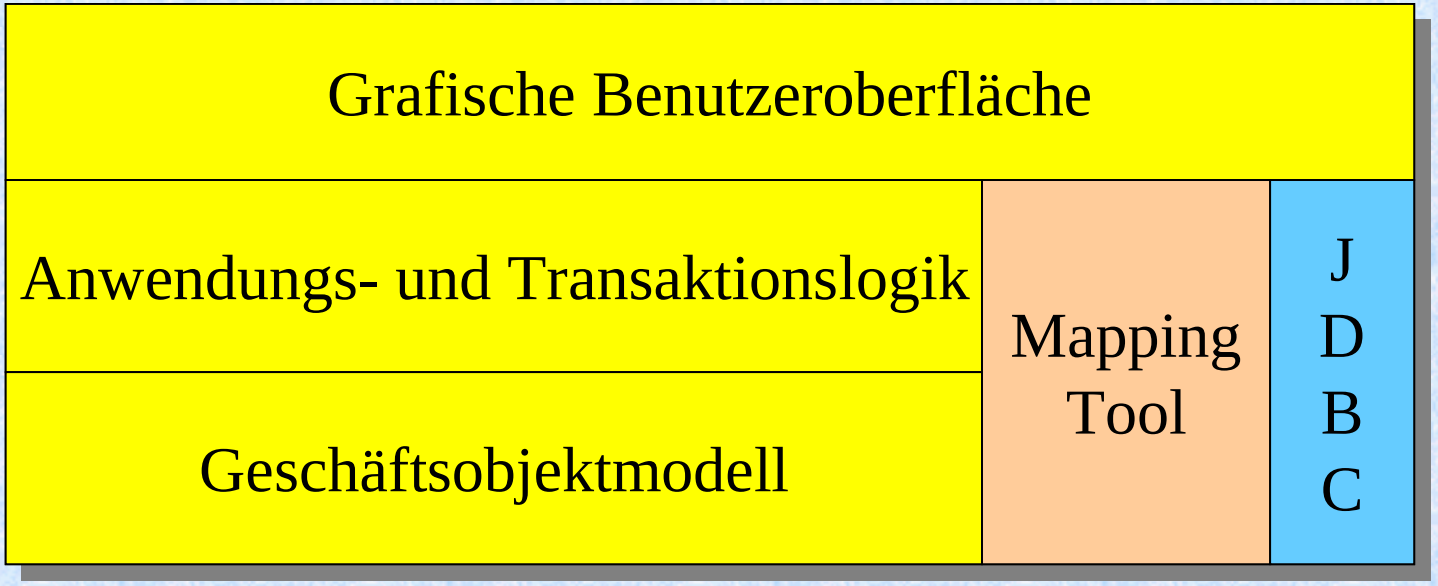
Auswahlkriterien für ein Produkt (2)

- Wie gut sind die Entwicklungswerkzeuge?
- Welchen Einfluß nimmt das Tool auf ...
  - die Implementierung der Geschäftsobjekte?
  - die Anwendungsarchitektur?
  - den Entwicklungsprozeß?
- Werden die betriebswirtschaftlichen Ziele unterstützt?
  - Kosten (Entwicklungs- / Runtime-Lizenzen) vs. Nutzen
  - Marktposition / Referenzprojekte
  - Verfügbarkeit von Support
  - Einarbeitungs- und Nutzungsaufwand

Architektur - Szenarien

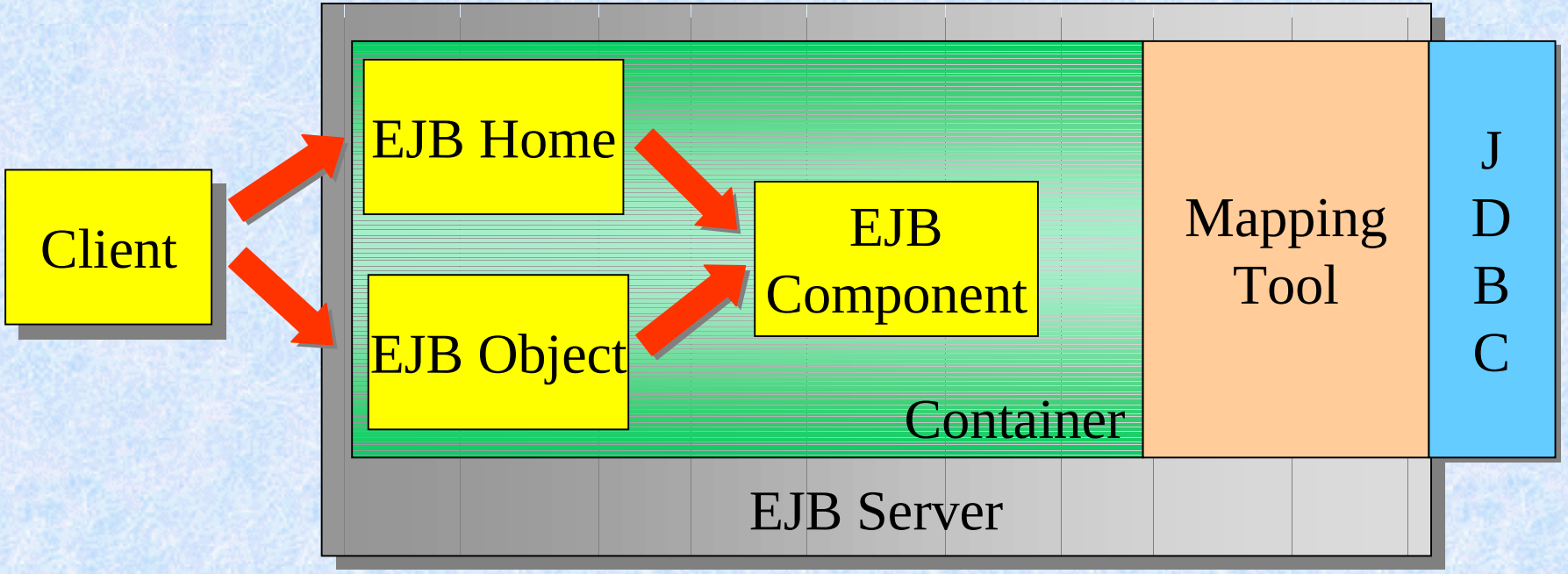


Architektur - Schichtenmodell



- Komponentenbasiertes Schichtenmodell
- Kapselung des Mapping-Tools von der Anwendung

Architekturmodell mit EJB



- Speicherung des Zustands eines Geschäftsobjektes (Entity Bean) mittels Mapping-Tool
- Granularität der EJB Komponenten ist u.U. abhängig vom Datenbankmapping

Mapping-Tools und mögliche Auswirkungen auf die Geschäftsobjekte

- Zusätzliche Attribute für technische Felder (Schlüssel, Sortierkennzeichen)
- Anpassung der Objekthierarchien und Attributtypen an die Abbildungsmöglichkeiten des Mapping-Tools
- Spezielle Logik für effiziente Datenzugriffe, wo die Automatismen des Tools nicht ausreichen
- Persistente Attribute müssen sichtbar sein
- Alternativ: Instrumentierung mittels Preprocessor

Mapping-Tools und mögliche Auswirkungen auf das DB-Schema

- Zusätzliche Datenbankspalten (z.B. Klassen-kennzeichen, Interfacekennzeichnung)
- Der Einsatz von DB-Constraints ist evtl. Abhängig von der Ausführungslogik (des Mapping-Tools)
- Eine zusätzliche Spalte für Optimistic Locking (bei OLC-Erkennung via Versionen)

Einige der genannten Punkte sind auch ohne Mapping-Tool relevant

Performance-Optimierung mit Mapping-Tools (1)

- Anpassen der Anfragelogik:
  - Gezieltes/verzögertes Lesen von Objekten (Einsatz von Wrapper/Proxies)
  - Partielles/inkrementelles Lesen (z.B. für Listendarstellung)
  - Virtuelle Listen für große Objektmengen
  - Minimierung der Anzahl von DB-Anfragen durch Joins
  - Minimierung der Häufigkeit von DB-Zugriffen durch Objekt-Caching oder gezieltes Vorauslesen
  - Entlastung der Datenbank durch Vermeidung von zu komplexen Joins
  - Eigene SQL-Anfragen für kritische Abfragen

Performance-Optimierung mit Mapping-Tools (2)

- Anpassen der Zuordnung:
  - Objekte mit 1:1-Beziehung in eine Tabelle zuordnen
  - Klassenhierachien auf eine Tabelle abbilden
  - Nutzung geeigneter Datenstrukturen (z.B. Vector statt LinkedList)
- Anpassen der Datenbank:
  - Definition geeigneter Indizes
  - Trigger, Constraints, Stored Procedures
  - RDBMS-spezifisches Tuning

Hinweise für das Projektmanagement (1)

- Frühzeitig Evaluation und Auswahl von Tools unter Berücksichtigung der angestrebten Architektur planen
- Ausbildung und Einarbeitung der Mitarbeiter planen
- Kosten für Lizenzen, Schulung und Support beachten
- Unterstützung durch den Hersteller sicherstellen

Hinweise für das Projektmanagement (2)

- Release-Wechsel nur bei konkretem Bedarf durchführen
- Mapping und DB-Schema in Konfigurations-management aufnehmen
- SQL und RDBMS-Know-How bereitstellen

Zusammenfassung (1)

- Die effiziente Entwicklung der Objektpersistenz wird unterstützt. ➠ Die Entwicklungskosten werden reduziert**.**
- ‚Plug and Play' durch automatisches Mapping und Datenbankgenerierung. Die Optimierung kann später erfolgen. ➠ Ein lauffähiges Programm liegt schneller vor**.**
- Verfügbare Produkte sind sehr jung. Nicht alle Funktionen sind ausgereift. ➠ Evaluation empfohlen.
- Gute Mapping-Tools lassen sich nahtlos in den Entwicklungsprozeß einbinden.
- Ein Mapping-Tool kann Einfluß nehmen auf: Objektmodell, Datenbank, Architektur der Anwendung. ➠ Ein späterer Wechsel ist in der Regel aufwendig und teuer.

Zusammenfassung (2)

- Mit einem Mapping-Tool ist man länger unabhängig von einem konkreten Datenbankprodukt.
- Mit einem Mapping-Tool erzielt man Vorteile bei Änderbarkeit und Skalierbarkeit gegenüber einem ad-hoc-Ansatz.
- Qualitätsgewinn durch
  - systematische Zuordnung zwischen Objektwelt und Datenwelt
  - die Nutzung von ausgereiften und flexiblen Mechanismen

Effizienzsteigerung durch den Einsatz von Mapping-Tools bei der Integration von RDBMS in Java-Anwendungen

ARS NOVA Software GmbH
Klaus Kiehne

Java Forum, 28. Juni, 9:45 - 10:30