Löwenfels partner

The power of metadata

Ressourcen-Einsparung durch
Wiederverwendung von Metadaten aus
einem Analyse- & Design-Tool

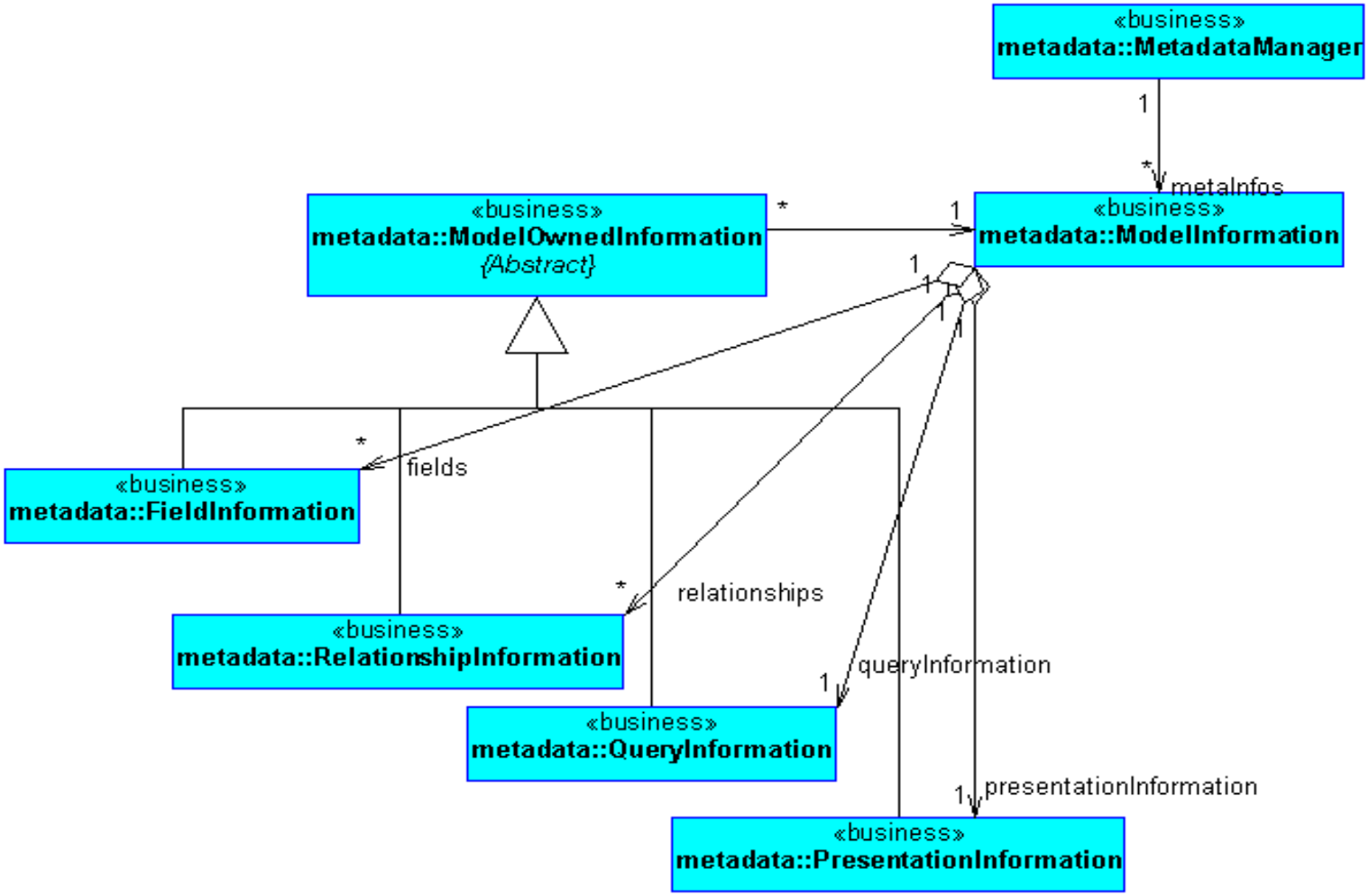
Definition Metadaten

- Metadaten dienen der Beschreibung von Daten
- Können meist aus Analyse- & Design-Tools exportiert werden (mittels XMI [XML Metadata Interchange])
- Mit der Ergänzung von Metadaten durch eigene Definitionen kann eine 100%ig generische Software-Lösung erstellt werden.

Funktion von Metadaten

- Grundlage zur Implementation
  - Java-Code
  - DB-Definition
  - MLS-Grundlage (properties-Dateien)
  - Dokumentation des OO-Modells
- Validierung zur Runtime
- Aufbau von generischen GUIs (sehr wirksam in Publishing-Frameworks, z.B. Cocoon)

Möglicher Aufbau von Metadaten



Auswirkungen von Metadaten

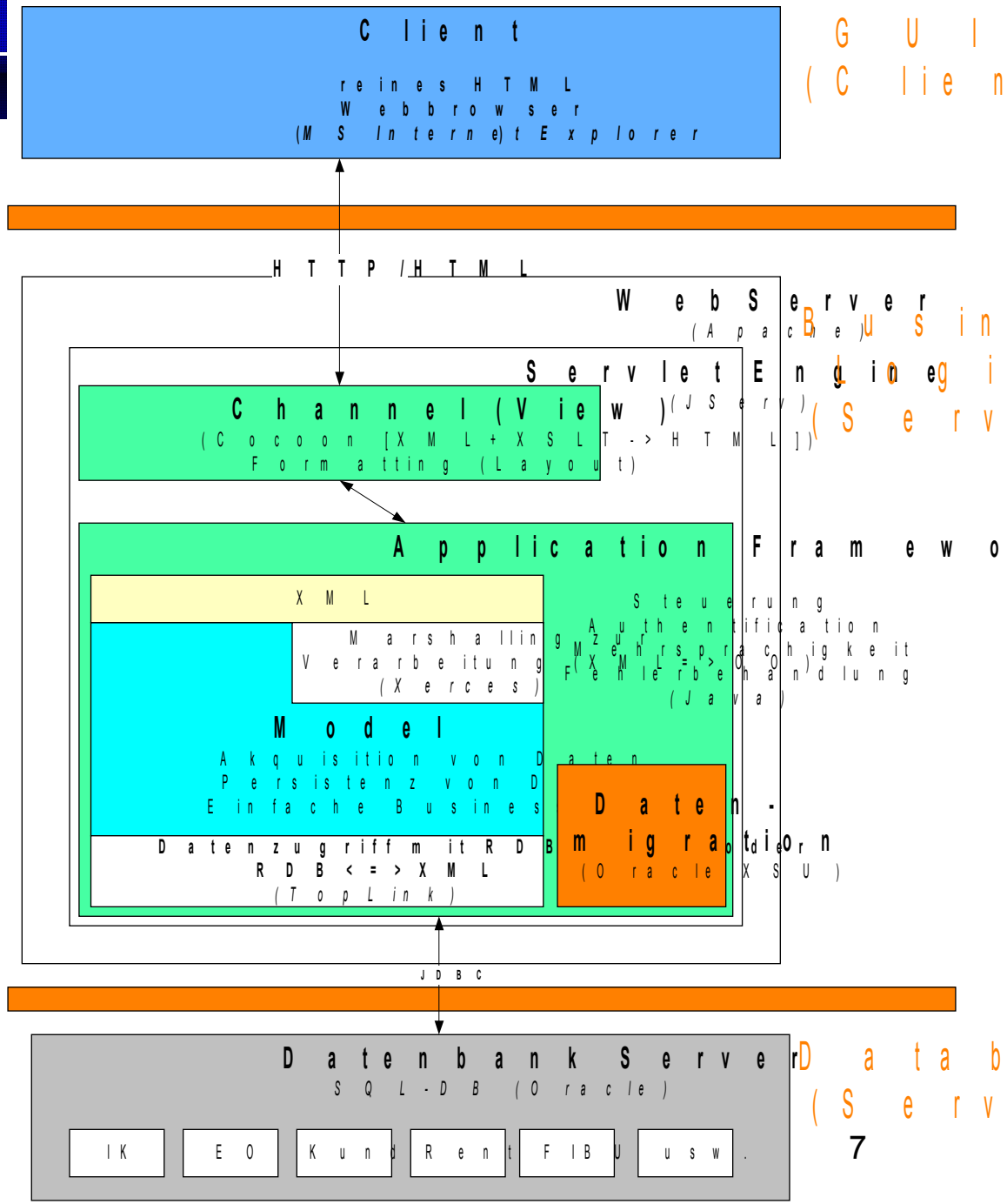
- Massive Reduktion der Implementationszeit durch Generierung
- Modelländerungen im Design-Tool können mit den Metadaten der Runtime verglichen werden
  => Unterstützung von iterativem Vorgehen
- Generalisierung in diversen Bereichen:
  - Validierung
  - generische GUIs
- Grössere Komplexität des Frameworks, dafür Black-Box-Verhalten und Vereinfachung bei der Applikationsentwicklung

Probleme der Metadaten

- Aus Performance-Gründen in Business-Object-Klassen eingebunden
  => Änderungen bedeuten Regenerierung der Java-Klassen
  => Teil-Auslagerung in XML (Performance-Frage)
- Metadaten müssen jeweils für zusätzliche Informationen erweitert werden
  => Lösungsansatz Meta-Metadaten
  (Daten, welche die Metadaten beschreiben)

System-Architektur Infosystem

3 mögliche Views im GUI
- Suchmaske
- Resultatliste
- Detailansicht



Benutzersicht (GUI)

Client

- Basis = Browser (IE5.5)
- Menüleiste / Pushbuttons / Help
- Views: Query-View, List-View, Detail-View, Detail-Header-View, Link-View
- Navigationsmöglichkeiten
- History (Verlauf), Multi-Language-Support / Sprachumschaltung

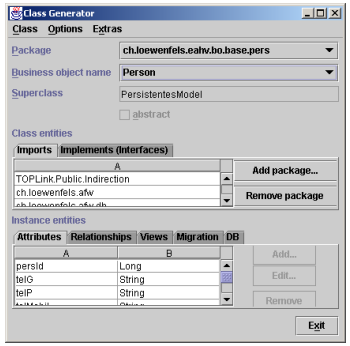
Modellanpassung (Ziel):

- neue Navigation von Person -> Hauseigentum -> Lokation

Verwendung von Metadaten bei Löwenfels Partner AG

Export als XMI

SELECT Enterprise
OOA/OOD
nach RUP (stark vereinfacht)
mittels UML



Code Generator
Erweiterung der Metadaten

Java Business Objects
(inkl. Metadaten)

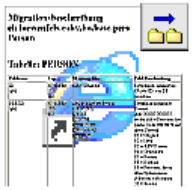
TOPLink Runtime Information
OO-RDB Mapping

ORACLE

DDL Skripts
(z.B.CREATE TABLE ...)

Deutsch
Français
Italiano

MLS Vorlagen
(Java properties)



HTML Migrations-beschreibung

Hinterlegte Metadaten

- Klassen (Import-Statements für Regenerierung)
- Felder (Längen, Typen, Formatierung)
- Beziehungen (Beziehungsklasse, Kardinalitäten)
- Ansichtsarten (Suchkriterien / Resultatspalten / Sortierungsreihenfolge / Detailansicht inkl. Beziehungen),
  inkl. Möglichkeit, dynamische Methodenaufrufe zu definieren.
- DB-Indices
- Migrations-Informationen

Einsparungen / Vorteile

- Ca 10 Manntage Implementation der Business Objekte pro Modell-Schnitt (monatlich)
- Ca 5 Manntage GUI-Definitionen / Monat
- Ca 3 Manntage Test / Monat
- Wesentlich geringere Fehleranfälligkeit
- Referenzierung von Attributen/Beziehungen in den Metadaten als Konstanten, wodurch bei Wegfall von solchen bereits zur Entwicklungszeit Compiler-Fehler auftreten
- Unterstützung der inkrementellen Entwicklung durch Vergleich des SELECT-XMI mit den vorhandenen Java-Metadaten

➢ Schnellere, kostengünstigere Umsetzung von Kundenwünschen

Einarbeitungsaufwand

Grundvoraussetzung: Kenntnisse in Java und VisualAge for Java

- Maximal 2 Tage Schulungsaufwand für die Beherrschung des Code-Generators und Vorgehensweise

Zukunft unserer Metadaten

- Weitere Ansichtsarten (z.B. Ansicht für‘s Editieren)
- Metadaten für erweiterte Geschäftslogik
- Metadaten für Assistenten (Abbildung der Geschäftsfälle)

Konkretes Beispiel

Aufgabenstellung (Modell-Erweiterung)

- Zusätzlich zu den vorhandenen Daten über Personen und Adressen sollen Informationen über Hauseigentum erfasst werden

Zustand vor der Änderung / Ausgangslage

- Klasse Person (mit Subklassen), Lokation
- Assoziation Person -> AdrBeziehung (zeitabh.) -> Lokation

Metadaten im Modell (Select Enterprise)

- Vorhandene Metadaten pro Klasse / Attribut
- Template Editor für zusätzliche Daten

Erweiterung / Vorgehen

Änderungen

- neue Klasse „Hauseigentum“ definieren
- Attribute zur Klasse Hauseigentum definieren
- Assoziation zu Person und Lokation definieren
- Modell-Informationen exportieren -> XMI
- Code-Generator starten mit XMI-File als Input
- Views definieren / anpassen
- Betroffene Java-Klassen neu generieren
- DDL-Script generieren und auf DB ausführen
- Testdaten erfassen
- Deployment der geänderten Java-Klassen
- Test im Browser