

Tobias Vollmer

25.1.2005

ex|Xcellent
solutions

Mit MDA vom GUI bis zur Datenbank



- x| Ziel: Modellgetriebene Entwicklung mit pleXX
- x| Überblick über pleXX
- x| Erstellen eines Objektmodells mit pleXX Business
- x| Erstellen von GUIs mit pleXX Presentation
- x| Tools und Utilities rund um pleXX
- x| pleXX im Projektalltag

x| Modellierung der fachlichen Aspekte

- | Fokussierung auf die Geschäftsobjekte und –prozesse
- | NICHT das technische Umfeld

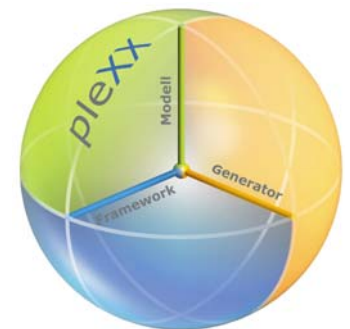
x| Herausforderung:

Überführung bzw. Verwendung des fachlichen Modells in der Programmierung

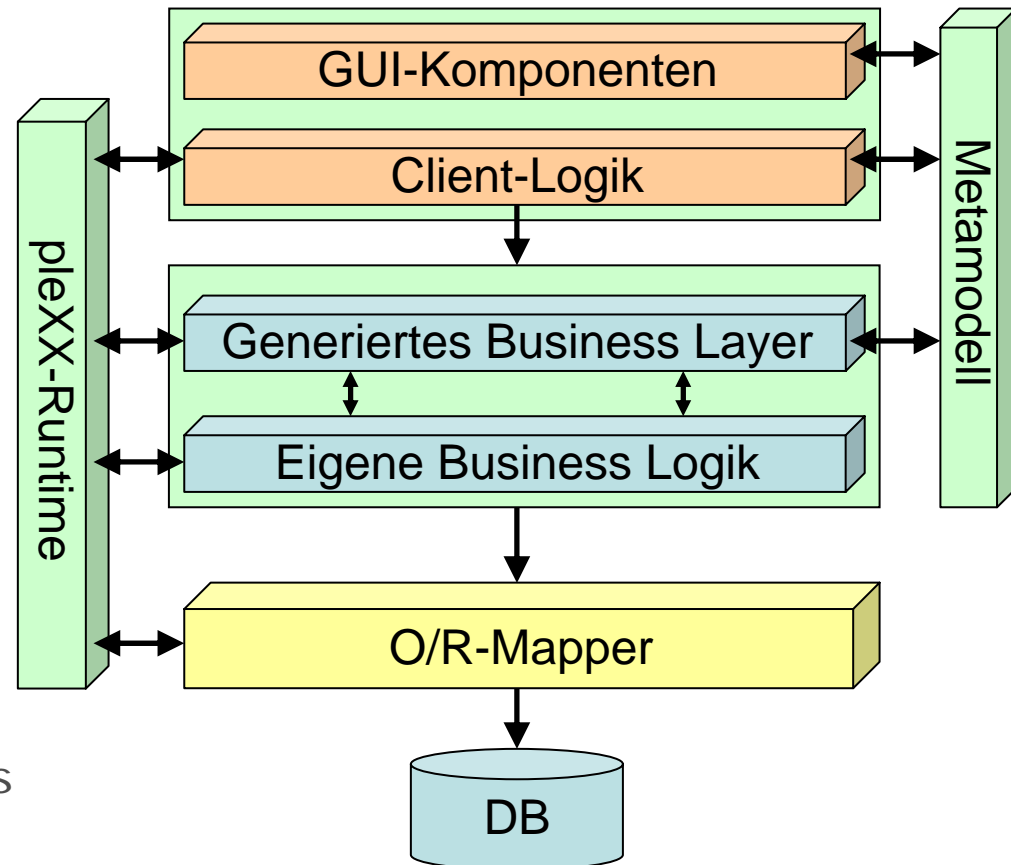
x| Lösung: **Generativer Ansatz** in Kombination mit **Frameworks**. **pleXX** ist ein wichtiger, pragmatischer Schritt auf diesem Weg

x| Vorteile:

- | Bessere Beherrschbarkeit der Komplexität
- | Implementierung näher an Kundensicht
- | Effizienz und Wirtschaftlichkeit



- x| pleXX Business:** MDA-Ansatz zur Modellierung und Abbildung persistenter Business-Objekte
- x| pleXX Presentation:** Effiziente Erstellung von GUIs auf Basis des Objektmodells mit mächtigen GUI-Komponenten und Metamodell
- x| 2/3-Schicht-Architekturen** für Thin- und Rich-Clients
- x| Zahlreiche Tools und Utilities** rund um pleXX



x| pleXX Business

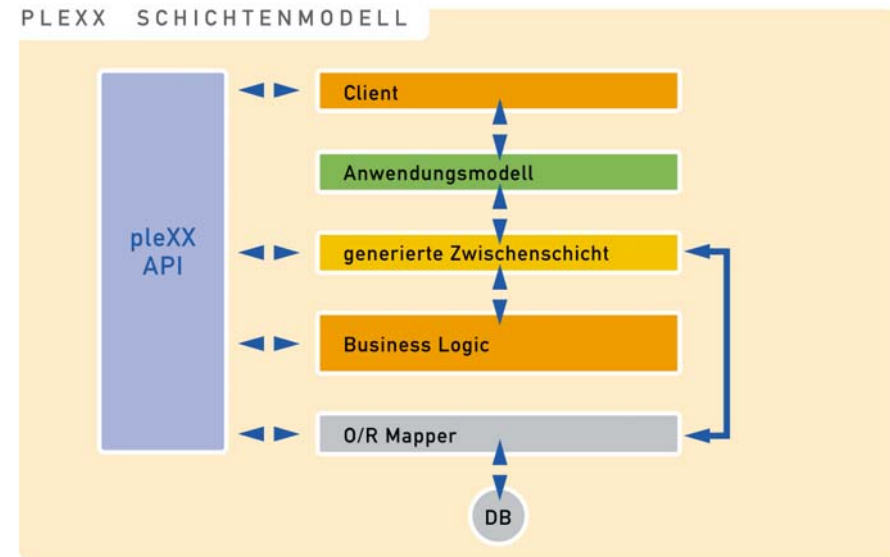
- | entstand aufgrund konkreter Projekterfahrungen:
 - mit EJB: zu hoher technischer Overhead
 - mit Historisierungs-Frameworks
- | Entwicklung parallel zu konkreten Projekten
- | seit April 2002
- | Einsatz in vielen Projekten (30-200 DB-Tabellen)

x| pleXX Presentation

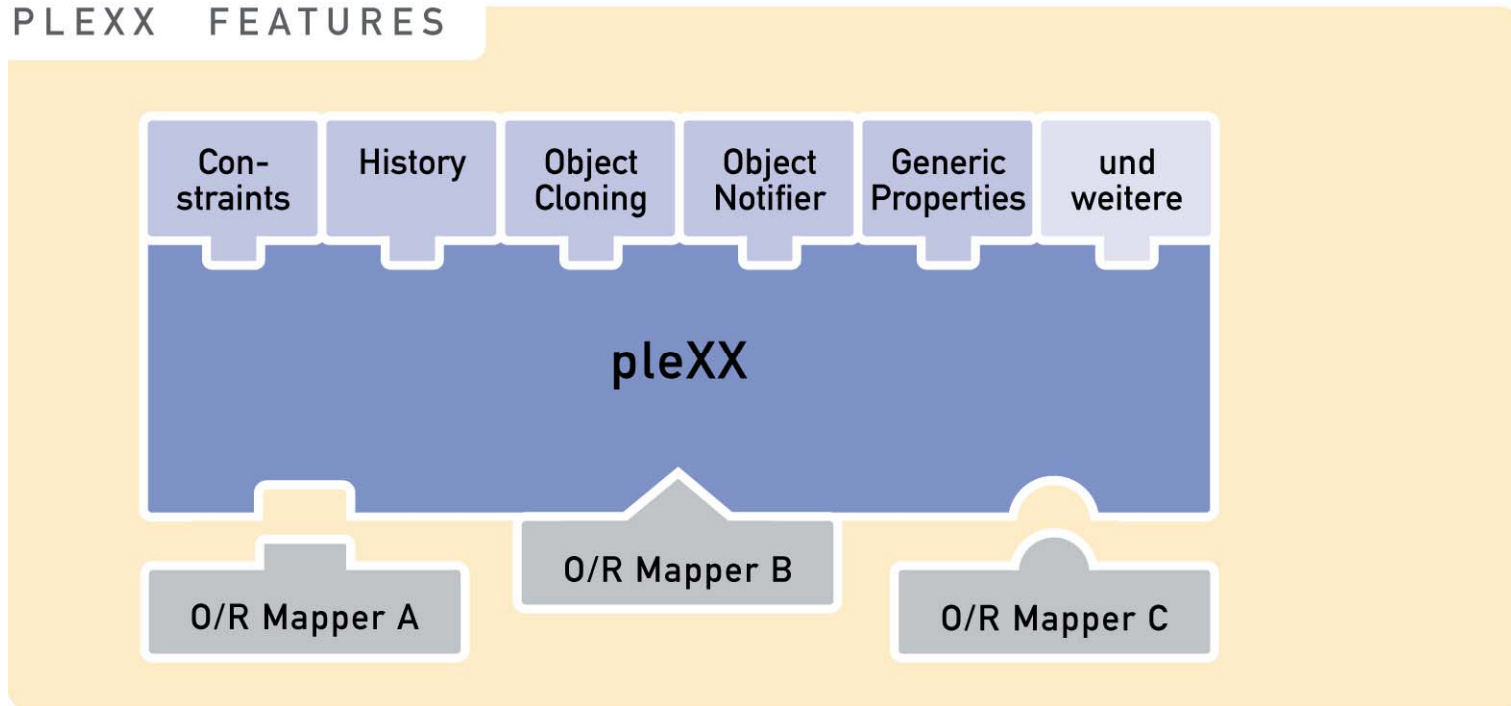
- | Ansatz durch Erfahrungen mit pleXX business
- | sowie mehreren (auch generativen) GUI-Frameworks.
- | Entwicklung seit November 2003
- | Bisher eingesetzt in 3 Projekten (30-100 DB-Tabellen)

Allgemeines Framework für die Modellierung und Abbildung persistenter Business Objekte und deren Integritätsregeln

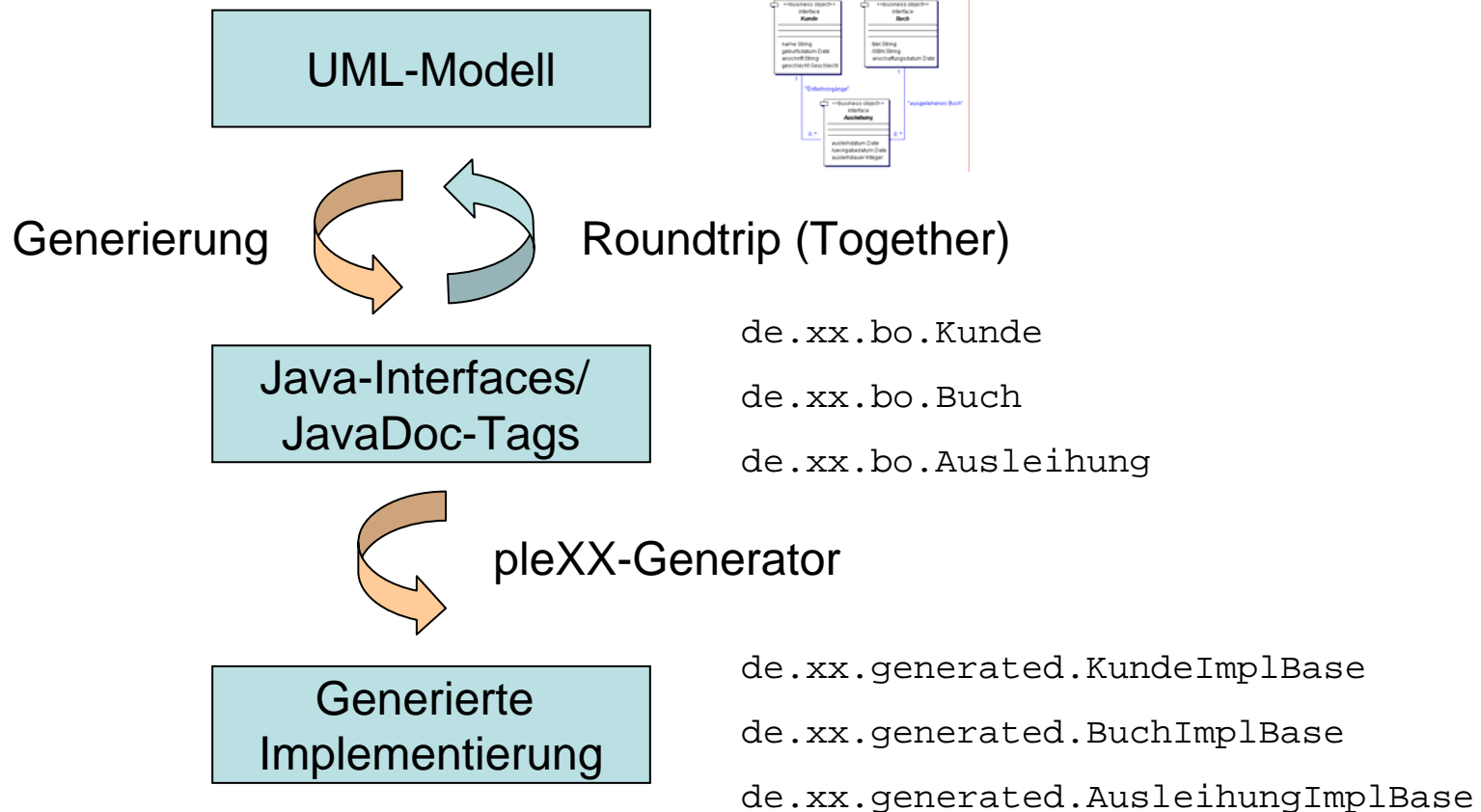
- xl Modellierung in UML Klassendiagrammen
- xl Generierung basierend auf Java Interfaces
- xl Abstraktionsschicht zwischen der Java-Anwendung und dem eigentlichen Persistenzmechanismus
- xl Mehrwertdienste im Framework



PLEXX FEATURES



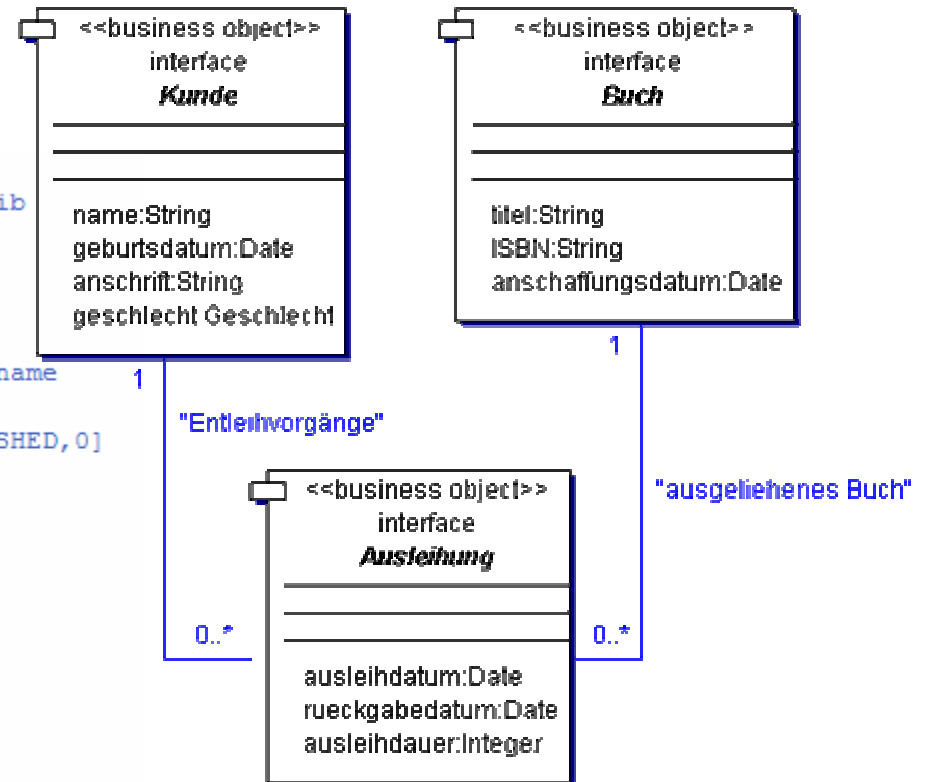
- x| (OCL-)Constraints
- x| Historisierte Objekte
- x| Clonen von Objektgraphen
- x| Change Notifier
- x| Benutzerdefinierte Metadaten
- x| Undo innerhalb der Transaktion
- x| Bidirektionale Beziehungen
- x| Austauschbarer O/R Mapper



x| Ein einfaches Objektmodell, modelliert mit pleXX ...

```
/**
 * Ein Kunde ist eine Person, die Bücher aus der Bib
 * @stereotype business object
 */
public interface Kunde {
    /**
     * Kompletter Name inklusive Vor- und Nachname
     * @mandatory [ON_ADJUSTMENT_FINISHED,0]
     * @maxLength(3,50) [ON_ADJUSTMENT_FINISHED,0]
     * @label "Name"
     *
     * @columnName "Name"
     * @hoverText "Name des Kunden"
     * @mnemonic N
     */
    public String getName();

    /**
     * Geburtsdatum des Kunden.
     * @mandatory [ON_ADJUSTMENT_FINISHED,0]
     * @label "Geburtsdatum"
     *
     * @columnName "Geburtsdatum"
     */
}
```



- x|** Implementierungsklasse für modellierte Interfaces
 - | Mit Pflege von Rückwärtsbeziehungen
 - | Constraintprüfungen (bei historisierten Objekten zu allen notwendigen Zeitpunkten)
 - | Implementierung der eigenen Business-Methoden erfolgt gemäß ‚generation-gap‘ Ansatz in einer abgeleiteten Klasse

- x|** Meta-Modell
 - | Zum Zugriff auf im Modell hinterlegte Informationen wie Geschäftsobjekte, Constraints, Attribute, Beziehungen
 - | Enthält generierte Konstanten (→ Fehlererkennung zur Compilezeit)

x| Strategische Vorteile

- | Vereinheitlichung der Projekte
- | Forcierung von Modellierungsrichtlinien

x| Operative Vorteile

- | Arbeitersparnis und Reduktion von Fehlerquellen durch Codegenerierung
- | Bereitstellung von Mehrwertdiensten

- x|** Im Modell vorhandene Information wird häufig im GUI dupliziert
 - | Schlechtere Wartbarkeit
 - | Inkonsistenzen
- x|** Prüfungen aus dem Objektmodell werden im GUI dupliziert
- x|** Häufig viele ähnliche Dialoge zur Stammdatenverwaltung
- x|** Hoher Aufwand für Positionierung, Mnemonics, Label-Verknüpfungen, angemessene Fehlerbehandlung
 - | Hoher Aufwand bei Bugs in nachprogrammierter Querschnittsfunktionalität
 - | GUI-Guidelines nur schwer sicherzustellen
 - | Hoher Testaufwand
 - | Keine Abhilfe durch GUI-Builder

x| Ziel

- | Effiziente Erstellung grafischer Benutzeroberflächen
- | Erstellung von Web- und Desktopanwendungen basierend auf **einem** Modell
- | Automatische Synchronisation von Model (Business-/Persistenzlayer) und View (Presentationlayer)

x| Idee

- | Hinterlegen der wesentlichen GUI-Informationen zu Attributen
- | Mächtige Komponenten reduzieren Programmieraufwand
- | Konfigurierbarkeit

x| Technologien

- | Swing
- | wingS

x| Verknüpfung von Objektmodell mit der GUI und Programmierung

The image shows a software application window titled 'Kunden bearbeiten' (Edit Customer) overlaid on a main data entry form. The dialog box contains the following fields:

- Name: Ma
- Anschrift: Schützenstraße 12
- Geburtsdatum: 30.04.1977
- Geschlecht: Weiblich

The main form in the background is titled 'A 000 545 38 26 - Kontakt bearbeiten' and contains various technical specifications for a contact component, including:

- Teilenummer (1): A 000 545 38 26
- Benennung [de] (1): Buchsenkontakt
- Benennung [en] (1): socket contact
- Teilenummer Einzelware: [empty]
- Pos. Nummer (1): P750
- Kontaktform (2): Flachsteckhülse
- Kontaktfamilie (2): FF6,3
- Min. Leitungsquerschnitt (2): 2,5 mm²
- Max. Leitungsquerschnitt (2): 2,5 mm²
- Min. Isolationsdurchmesser: [empty] mm
- Max. Isolationsdurchmesser: [empty] mm
- Oberflächenmaterial (2): [empty]
- Gewicht (2): [empty] g
- Primäre Verriegelung (2): [empty]
- Bemerkung Mehrfachanschlag: [empty]
- Mehrfachanschlag (2):

25 Codezeilen

- x| Automatisches Layout von Eingabemasken, Tabellen, Suchmasken
- x| Synchronisation von Model und View zu konfigurierbaren Zeitpunkten
- x| Frühzeitige Gültigkeitsprüfung von Benutzereingaben
→ Immediate-Constraints aus dem Modell werden geprüft
- x| Berechtigungssteuerung
 - | Aktivierung/Deaktivierung von UI Komponenten
- x| Konfigurierbarkeit
 - | Mehrstufig
 - | Callbacks
 - | Factories

- x|** Internationalisierung
 - | Komfortabler Editor von ResourceBundles
 - | Generierung von ResourceBundles aus dem Modell
 - | Generierung von Zugriffsklassen auf ResourceBundles
 - | Automatische Prüfung von Konsistenz und Vollständigkeit

- x|** Fertige Applikationsrahmen
 - | Rich-Client
 - | Thin-Client

- x|** Wiederverwendbare Applikationskonzepte
 - | Benutzer, Rechte, Rollen Verwaltung

- x|** Hat sich sehr bewährt
 - | auch bei größeren Projekten (ca. 150 persistente Klassen, 200 Datenbanktabellen)
 - | bei internationalisierten Anwendungen
 - | im Web-Bereich auch mit Layoutvorgaben
 - | im Prototyping-Bereich

- x|** Enormer Produktivitäts- und Qualitätssprung
 - | im Objektmodell besonders durch Constraints sowie Historisierung
 - | im GUI-Bereich durch Standardisierung, einheitliche Präsentation, geringere Komplexität bei größerer Usability
 - | deutlich weniger Quelltext

- x| Für Massendatenverarbeitung ist Objektnavigation zu langsam
 - | Hier OQL/OR-Mapper-QL verwenden
 - | Sinnvolle Policy:
 - Massenabfragen an pleXX vorbei
 - Datenmanipulation ausschließlich über pleXX

- x| Projektspezifische Highlights:
 - | Automatische Erzeugung von Datenbank-Dokumentation
 - | Änderungsjournal
 - | Berechtigungssteuerung auf Basis von Zuständigkeiten
 - | Together-Audits über das Objektmodell

- x| Bei weitergehenden Fragen:
 - | Tobias Vollmer
 - | t.vollmer@exxcellent.de
 - | +49 (0)731 55026 32
 - | <http://www.exxcellent.de/plexx>