professional it software & services pitss®

Roadshow

Oracle Forms von A - Z

Das technologie-unabhängige PITSS.CON Vorgehensmodell

Andreas Gaede

The Oracle Modernization Experts

ORACLE Gold Partner

Das technologie-unabhängige PITSS.CON Vorgehensmodell

- Die Modernisierung beginnt bei Forms
  - Vor jedem Vorgehen steht die Entscheidung
  - PITSS Vorgehensmodell
- Die Applikations-Analyse – der Schlüsselstellung
  - Analysieren nach welchen Gesichtspunkten
    - Code-Beschaffenheit und Code-Qualität
    - Dokumentation und Werterhalt
    - Als Ausgangsbasis für …
- Reduzieren Sie Ihre Applikations- Komplexität um bis zu 40%
  - Unused Code finden und entfernen
  - Redundanter Code finden und zusammenführen
- Die Koexistenz von Oracle Forms, Oracle ADF und APEX

professional it software & services

pitss®

Die Modernisierung

The Oracle Modernization Experts

ORACLE Gold Partner

was tun?

upgrade

develop

integrate

evolve

Modernization

Forms 11g | JEE Open Source | Fusion

ADF | .Net | ESB

APEX | GWT Google Toolkit | SAP

SOA WebService | others | others

Der Forms-Upgrade

upgrade

ORACLE FUSION MIDDLEWARE 11g FORMS

- Pros
  - bewährte Technologie
  - erfahrene Mitarbeiter
  - geringste Kosten
  - Integration Möglichkeiten
  - leistungsfähige Umgebung WLS
  - Ideale Ausgangsbasis für eine samfte Migration
  - Kein Druck
- Cons
  - Legacy / die alte Applikation
  - bekannte Probleme bleiben
  - Know-How geht verlohren
  - It’s not a Hype

Die Neuentwicklung – Re-Writing

develop

Microsoft .net

- Pros
  - Motivation (alles Neu)
  - neues Applikationsdesign
  - neue funktionale Spezifikationen Geschäftsprozesse
  - klare Nutzung der Technologie
  - das modernste Nutzen

eclipse

ORACLE 11g FUSION MIDDLEWARE APPLICATION DEVELOPMENT FRAMEWORK

- Cons
  - hoher Trainingsaufwand
  - oft mit eigenem Framework
  - Ressourcen-intensiv
  - Zeit-aufwendig
  - Risiko-behaftet
  - Fehler-anfällig
  - Kosten-intensiv

How to Survive a Rewrite

The following is only funny if you're not living it. I present: the five stages of the software rewrite.

Phase 1 Euphorie. "Yay! We've finally thrown off the yoke of that crappy, buggy product! We get to start over, and by God, this time we're going to do it **right!**“

Phase 2 “sich reinknien”. "There's a lot of work to do here. We'd better get to it."

Phase 3 Ernüchternung. "There's a metric (censored)-load of (censored) work here. We're not going to get done on time. All this stuff we've been doing (like testing, designing, integrating) is nice in theory... but we've really got to start cranking code or we're never going to make it!"

Phase 4 Erschöpfung. "We've missed our deadline again. (What was that... number three? Or number thirty?) Better call the family and tell them we'll be working late again. Six months of overtime... it could be worse... right?"

Phase 5 Befreiung. "We finally released! Okay, it's buggy and the design is kind of crappy, but at least we're done! It didn't turn out too well so we'll need to rewrite in a few years. And next time, by God, we're going to do it **right!**"

http://jamesshore.com/Blog/How-to-Survive-a-Rewrite.html

In Standardsoftware integrieren

integrate

- Pros
  - reichhaltige Funktionalität
  - moderne Prozesse
  - hohe Integration
  - geringer Wartungsaufwand
  - neue Releases
  - erprobt
- Cons
  - großer Beratungsaufwand
  - Ressourcen-intensiv
  - Kosten-intensiv
  - Zeit-intensiv
  - Anpassungen verringern die Vorteile
  - Anbieter abhängig

Siebel SAP Oracle APPS Microsoft Dynamics

Aus dem bestehenden entwickeln

evolve

- Pros
  - biete viele Vorteile einer Neuentwicklung
  - bewahrt das Gute
  - schrittweises Vorgehen
  - nimmt das Team mit
  - Reinwachsen in die Technologie
  - schnell zu starten
  - akzeptiert unterschiedliche Technologien
  - Risiko-minimiert
  - Kosten-optimiert
- Cons
  - nicht alles Neu
  - kann lange dauern

professional it software & services

pitss®

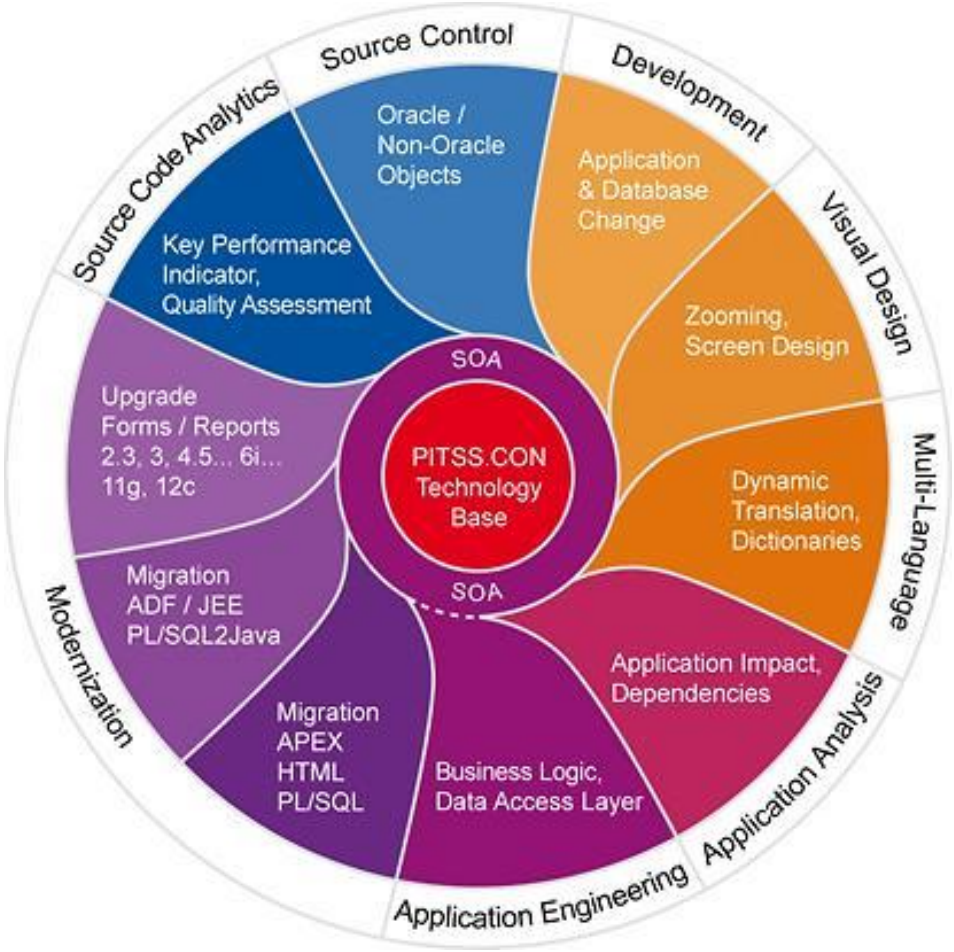
PITSS.CON

The Oracle Modernization Experts

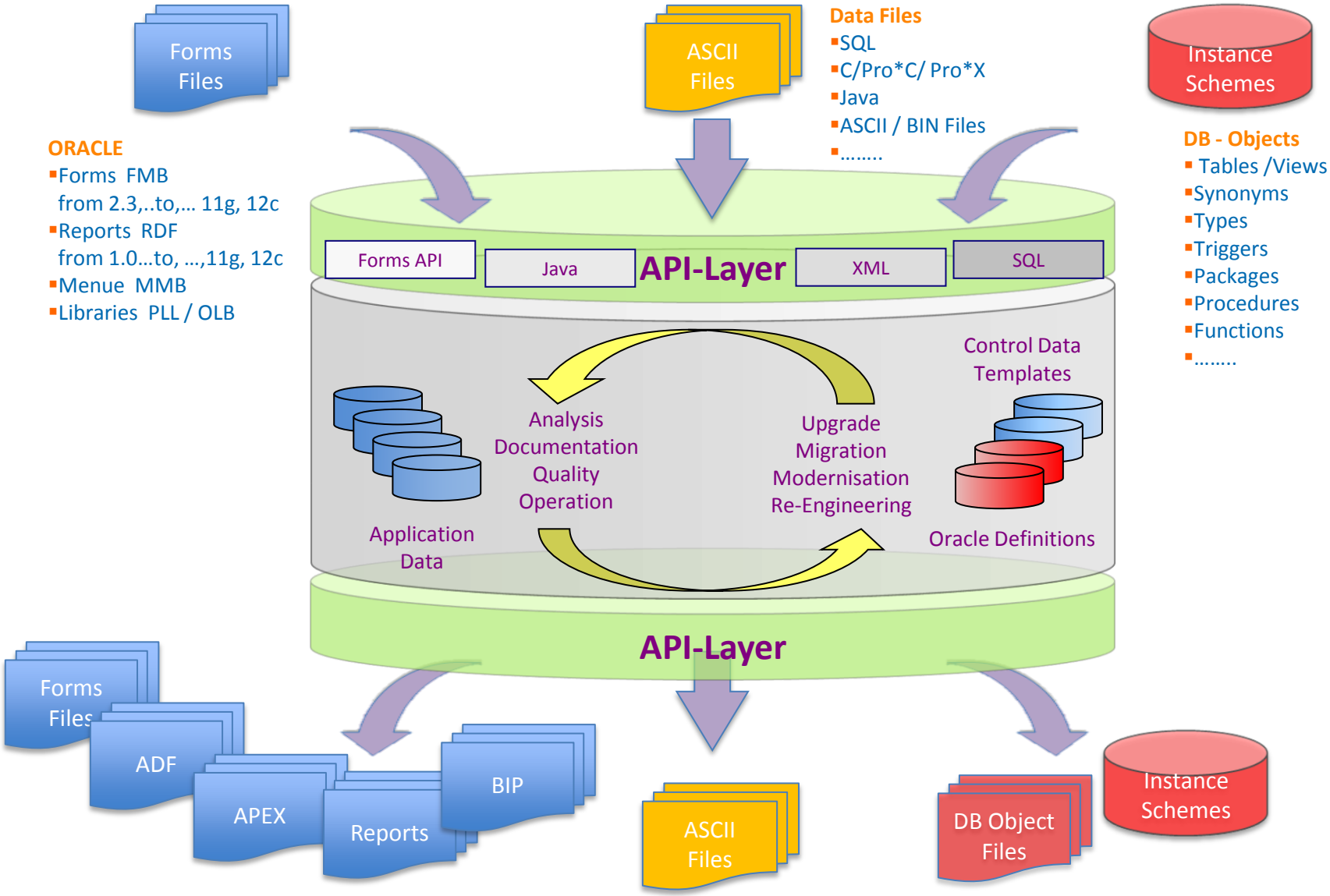
ORACLE Gold Partner

PITSS.CON - Source Code Analytics

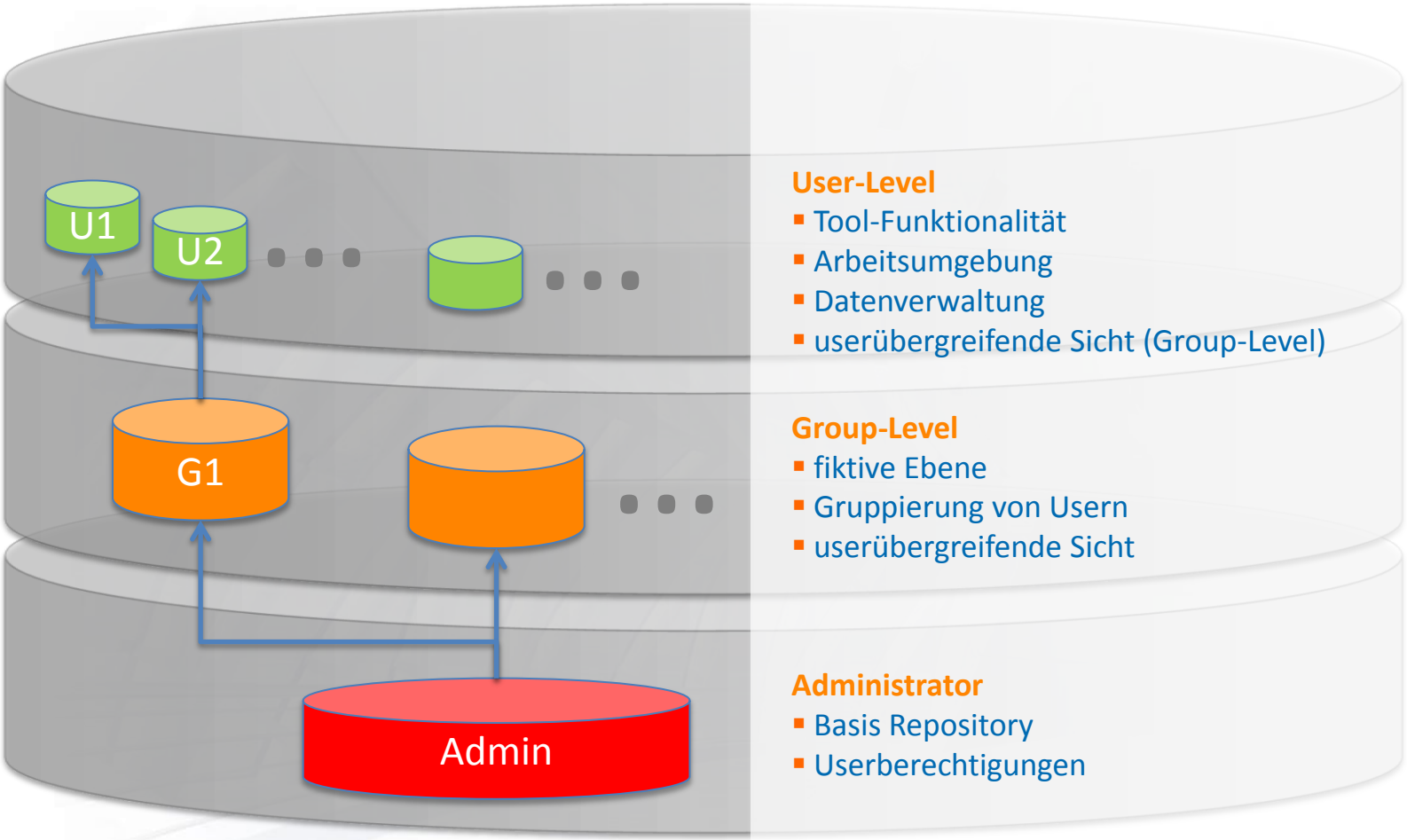
- Technology Base
- Maintenance / Development
- Graphical Visual Design
- Dynamic Multi-Language
- Application Analysis
- Application Engineering for SOA
- ADF- / APEX- Assistant
- Automatic Forms upgrading
- Source Code Analytics
- Source Control



PITSS.CON – Repository



PITSS.CON Administration

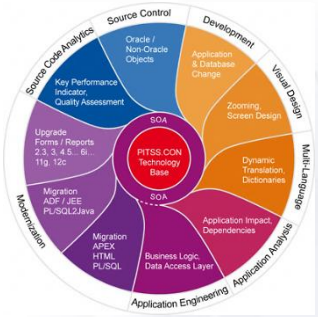


User-Level
- Tool-Funktionalität
- Arbeitsumgebung
- Datenverwaltung
- userübergreifende Sicht (Group-Level)

Group-Level
- fiktive Ebene
- Gruppierung von Usern
- userübergreifende Sicht

Administrator
- Basis Repository
- Userberechtigungen

PITSS.CON live



PITSS.CON Live

- Look & Feel
- Ladeprozesse
  - Module
  - Datenbank

professional it software & services

pitss®

Vorgehensmodell

The Oracle Modernization Experts

ORACLE Gold Partner

Interessante mögliche Prozesse – Viele gute Ideen

BL

Auditing

Co-Existence of Technologies

Any-GUI

Agile Development

Documentation

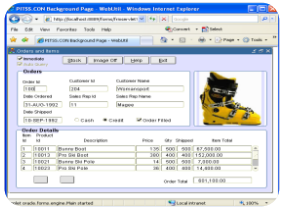
ADF

Analysis

Iterations

Redundancy

Forms 11g



Dead Code

SOA

APEX

Partial Projects

Definition

- organisiert einen Gesamtprozesses
- besteht aus mehreren Abschnitten (Phasen) und Meilensteinen die parallel oder nacheinander abgearbeitet werden können
- sinnhafte, logische Ordnung
- Iterationen meist notwendig und sinnvoll
- Indivituell anpassbar (tailoring)
- Meilensteine reduzieren das Risiko des Scheiterns oder Zeit und Kosten explosition

http://de.wikipedia.org/wiki/Vorgehensmodell

PITSS.CON Modernization Process

Software Quality / Documentation / Auditing

Agile Development

Analysis

Dead Code

Redundancy

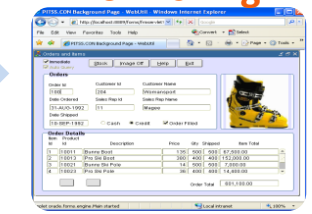
SOA

BL2DB

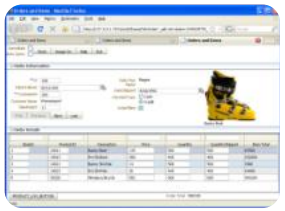
Iterations

Co-Existence of Technologies/ Partial Projects

Forms 11g

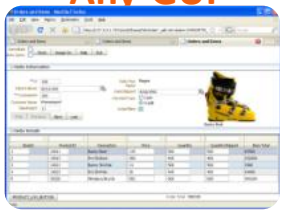


ADF



APEX

Any-GUI



PITSS.CON Modernisierungs-Pfad

Analysis

Dokumentation
-Programmsicht
-Applikationssicht

Informationen
-Abhängigkeiten
-Beschaffenheit
-Verwendung

Code-Zerlegung
- Code Parsing

Code –Abstraktion
- Metadaten

Code – Prüfung

Assessments
-Aufwand
-Migrationen
-Upgrade

Dead Code

auf Basis der Analyse

Fragmente
ohne Verwendung

Code Ketten ohne Verwendung

Null Code

Nicht aktivierter Code

Technologie-Relevanz von Obj.

Laufzeitbetrachtung
Mit RUEI

Redundancy

Vergleich von
-Objekten
-Code
-Programmen (4GL)
-Applikationen

Identifikation von
-gleichen Obj.
-ähnlichen Obj.

Prüfung der Referenzen
-Vererbung
-gebrochene Ref.
-Inhalte

Code Qualität

Verringerung der Komplexität, kleinere Funktionen

Lesbarkeit,
Wartbarkeit und
Werthaltigkeit
➔ Ressourcen-unabhängig
➔ modernisierbar

fehlervermeidend

Audit-fähig
zertifizierbar
Marktchancen

BL2DB

Identifizierung von Logik

Prüfung und Einschätzung
-einzelner Objekte
-ganzer Programme
-Forms
-Libraries

Code Generatoren für DB
-DAL
-BL
für externe Zugriffe
-DBLA
-WS

Table Based → Procedure Based

SOA enabling

Agile Development

Analyse
Code Zerlegung

Hohe Granularität
im schnelle Zugriff

Teamfähigkeit
Aufgabenteilung

Schnelle Umsetzung
- Objektfindung
- Automaten
- Generatoren

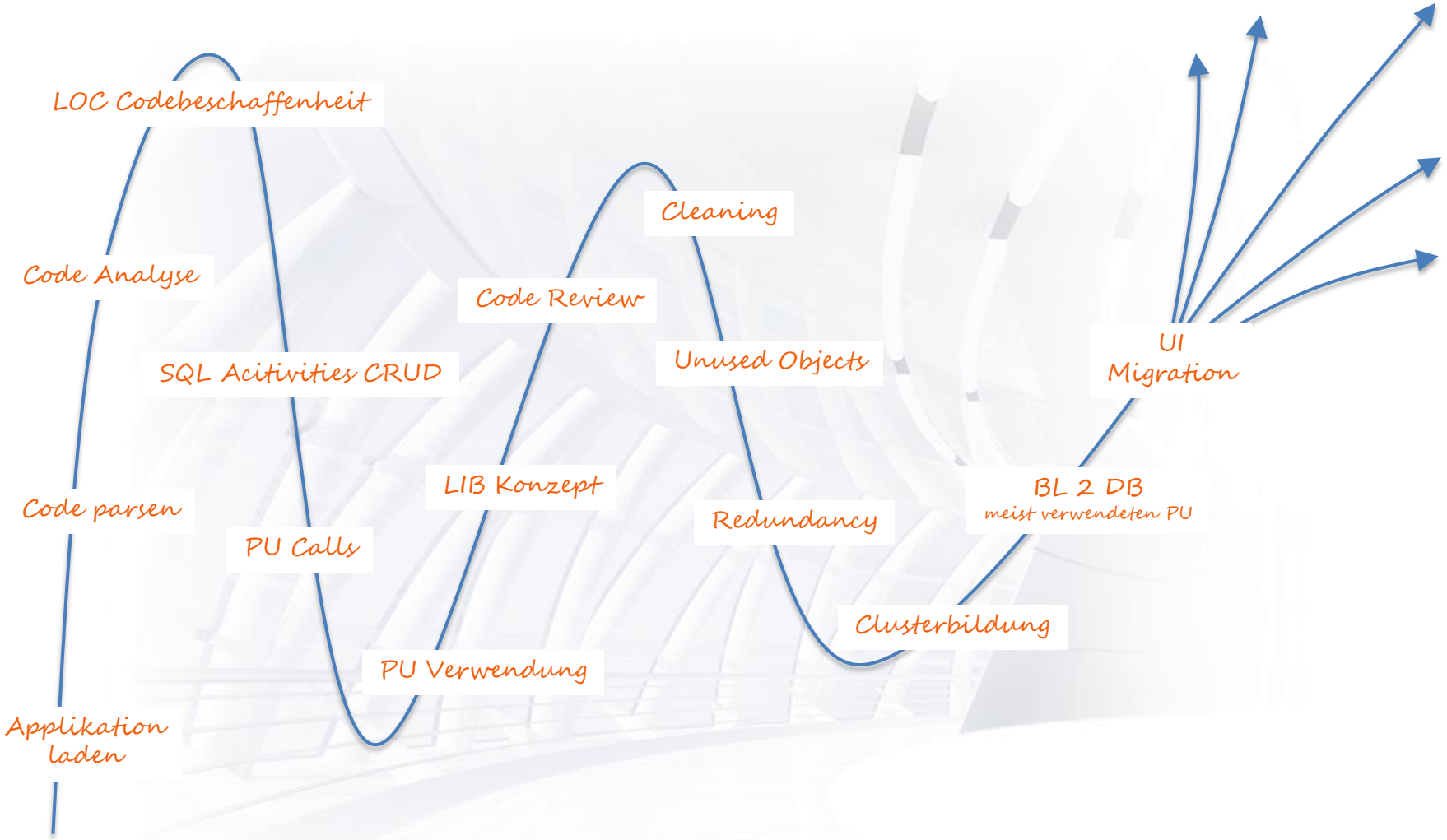
Beschleunigung der Entwicklungsphasen

Optimierbarer Deploymentprozess

Vorgehensweise im Analyse-Workshop

- Applikationsanalyse
  - Mengengerüste
  - Objektverwendungen
- allgemeine Betrachtung der Code-Qualität
  - Verständlichkeit
  - Komplexität
  - ➔ Wartbarkeit / Wertigkeit
- Library-Konzept / Refenzen
  - Objekt und Code Referenzen
- Unused Code / Objekte
- Code Strukturierung
  - Redundanter Code
- Code Restrukturierung
  - Zusammenziehen von Programm-Code
  - Re-Strukturierung der Libraries
  - Code -> DB
- Cluster / Domain / logische Gruppen
  - Code-Zerlegung

Das Vorgehensmodell im Detail



professional it software & services

pitss®

Code Bereiningung - Unused Code

The Oracle Modernization Experts

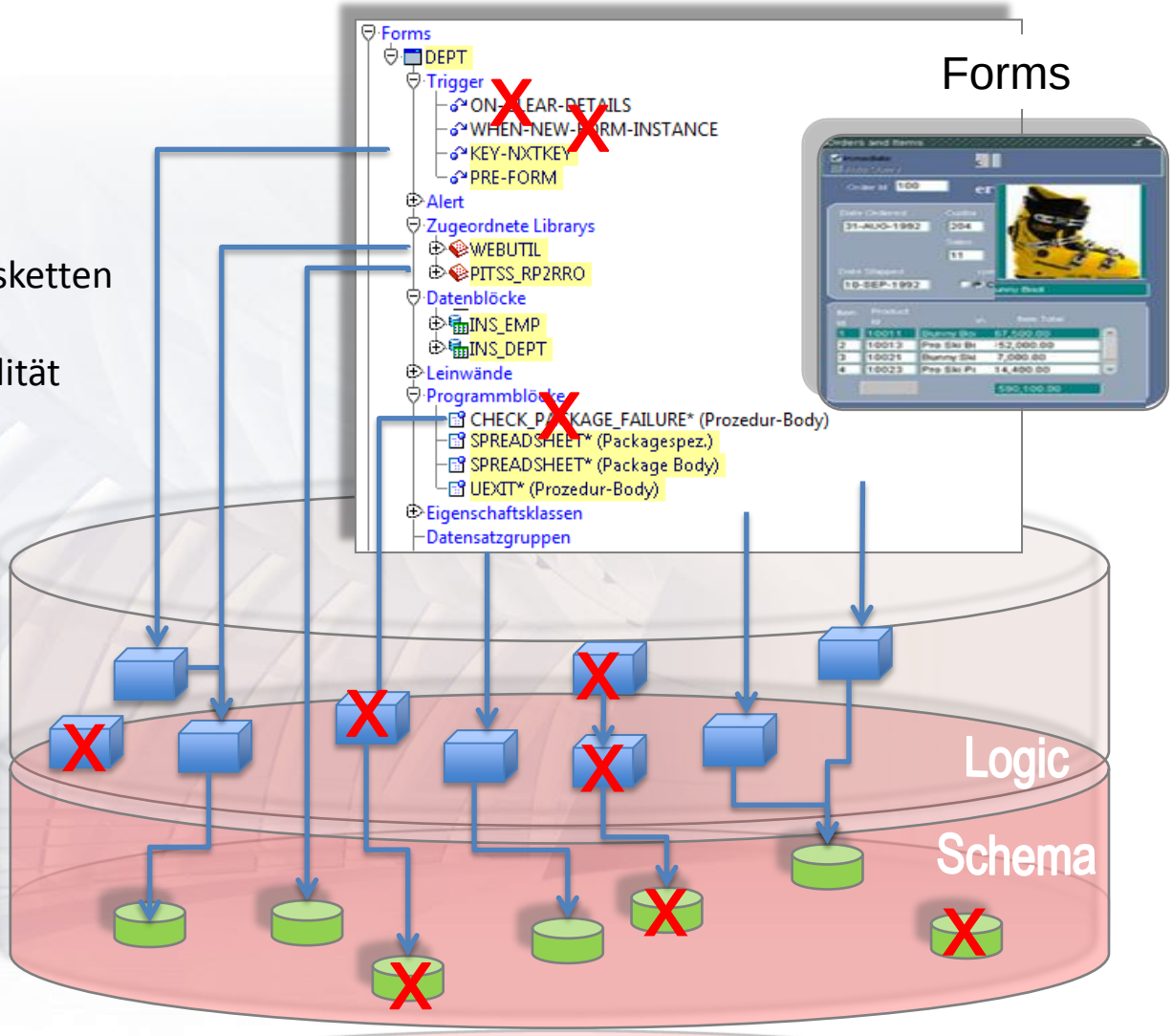
ORACLE Gold Partner

Unused Code / Dead Code

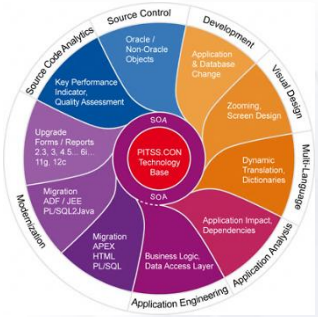
Unused Code

- nicht verwendete Objekte
- nicht verwendeter Code
- nicht verwendete Funktionsketten
- Null Code
- nicht getriggerte Funktionalität

- nicht Runtime-relevant ?!



PITSS.CON live



PITSS.CON Live

- Unused Code
- Kriterien
- Arbeitsweise

professional it software & services

pitss®

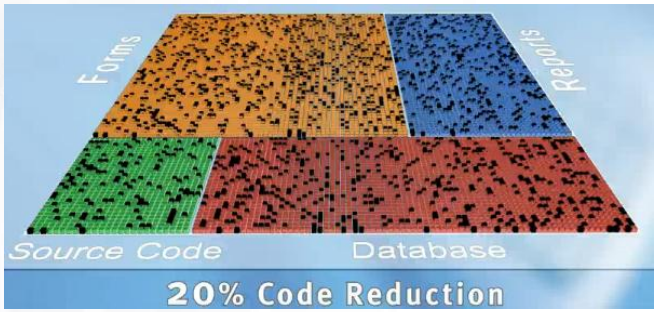
Redundanter Code

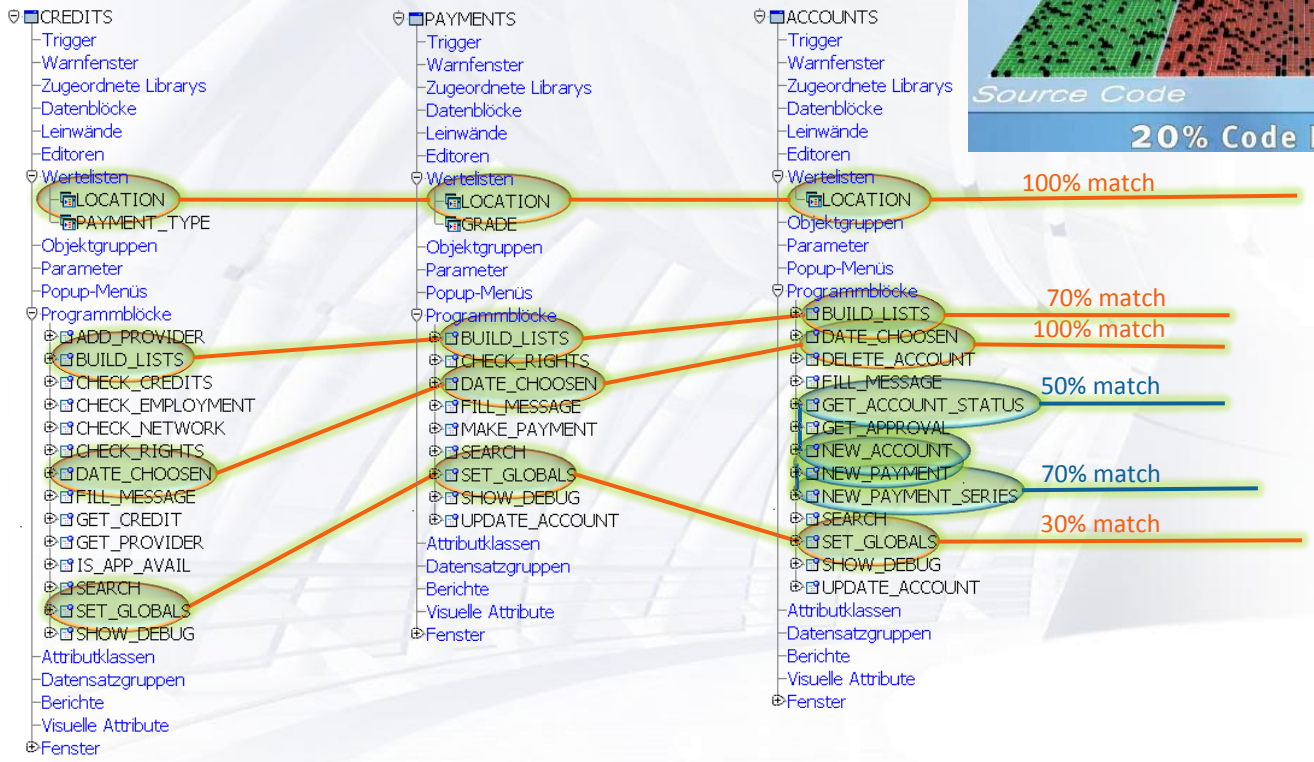
The Oracle Modernization Experts

ORACLE Gold Partner

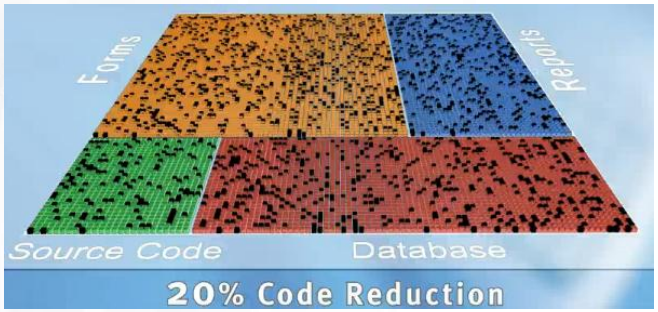
Forms Objekte / Business Logik

- wiederverwendbar
- zentralisierbar





Welche Kriterien werden zur Beurteilung von redundanten Code herangezogen



Objektparameter

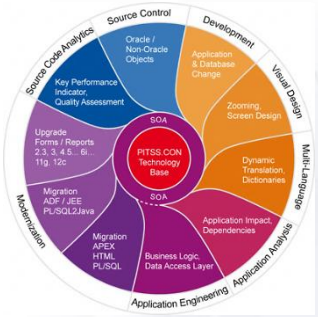
- Name
- Typ
- Objekt-Ebene

Objektinhalt

- SQL Statements
  - Statements Details
- aufgerufene Objekte
- direkte Zuweisungen
- logische Zuweisungen (Join-Conditions)
- Return-Werte

Erkennung

- erlaubt Unschärfe
- sehr flexible
- Modul-übergreifend



PITSS.CON Live

- Redundante Objekte

professional it software & services

pitss®

Code Qualität

The Oracle Modernization Experts

ORACLE Gold Partner

Warum Software-Qualität

Nach Untersuchungen von Standish Group, Gartner Group, Cutter Consortium und Center for Project Management:

- *23 % aller Softwareprojekte erfolgreich,*
- *53 % über Budget und/oder über Zeit und*
- *24 % abgebrochen*

International Data Corporation (IDC) - Umfrage

Umfrage zur Sicherung der Softwarequalität versus durch Softwaredefekte auftretenden internen und externen Kosten

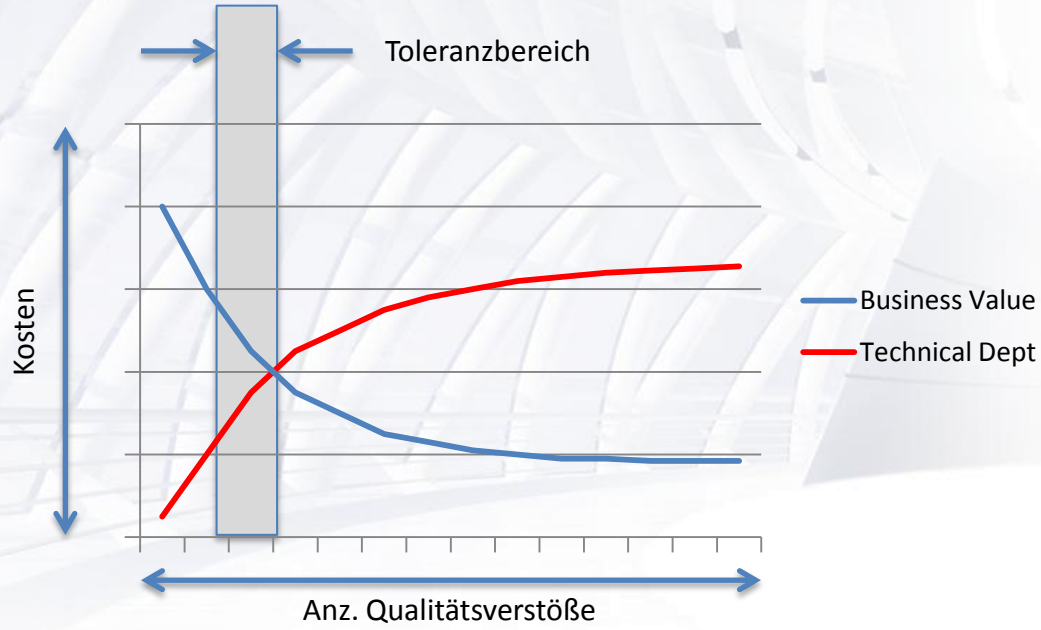
Etwa 14 Millionen Euro (22 Millionen Dollar) Kosten bringen Unternehmen je nach Größe pro Jahr für die Fehlerbeseitigung auf

IDC-Umfrage sieht die Probleme mit der Softwarequalität in mehreren Faktoren begründet:

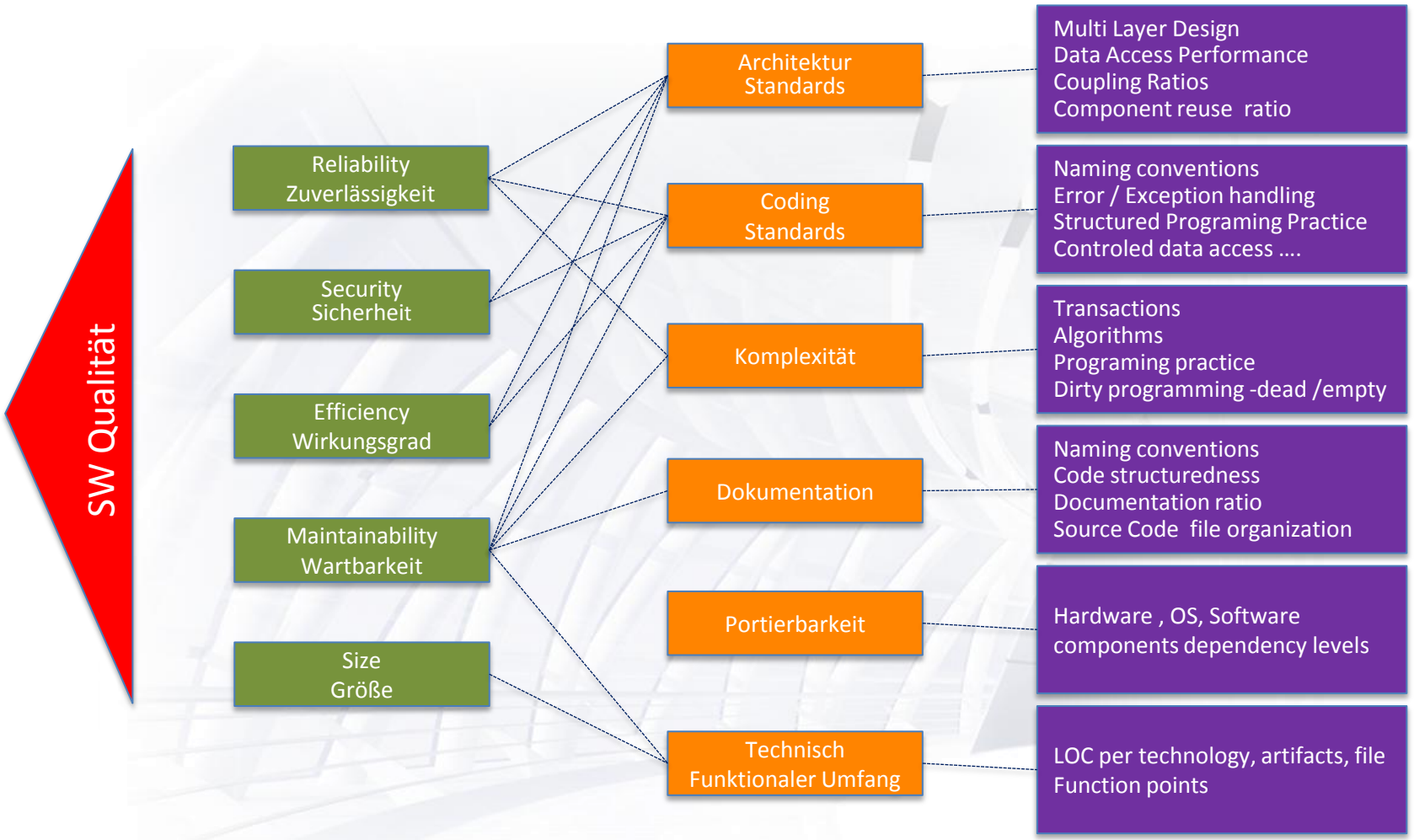
- wachsende Komplexität von Code,
- auf verschiedene Standorte aufgeteilte Teammitglieder,
- Outsourcing,
- veralteter Code,
- **Rückgriff auf Open-Source-Code** und
- das vermehrte Aufkommen von Multi-Thread-Anwendungen.

Gartner Group, Studie

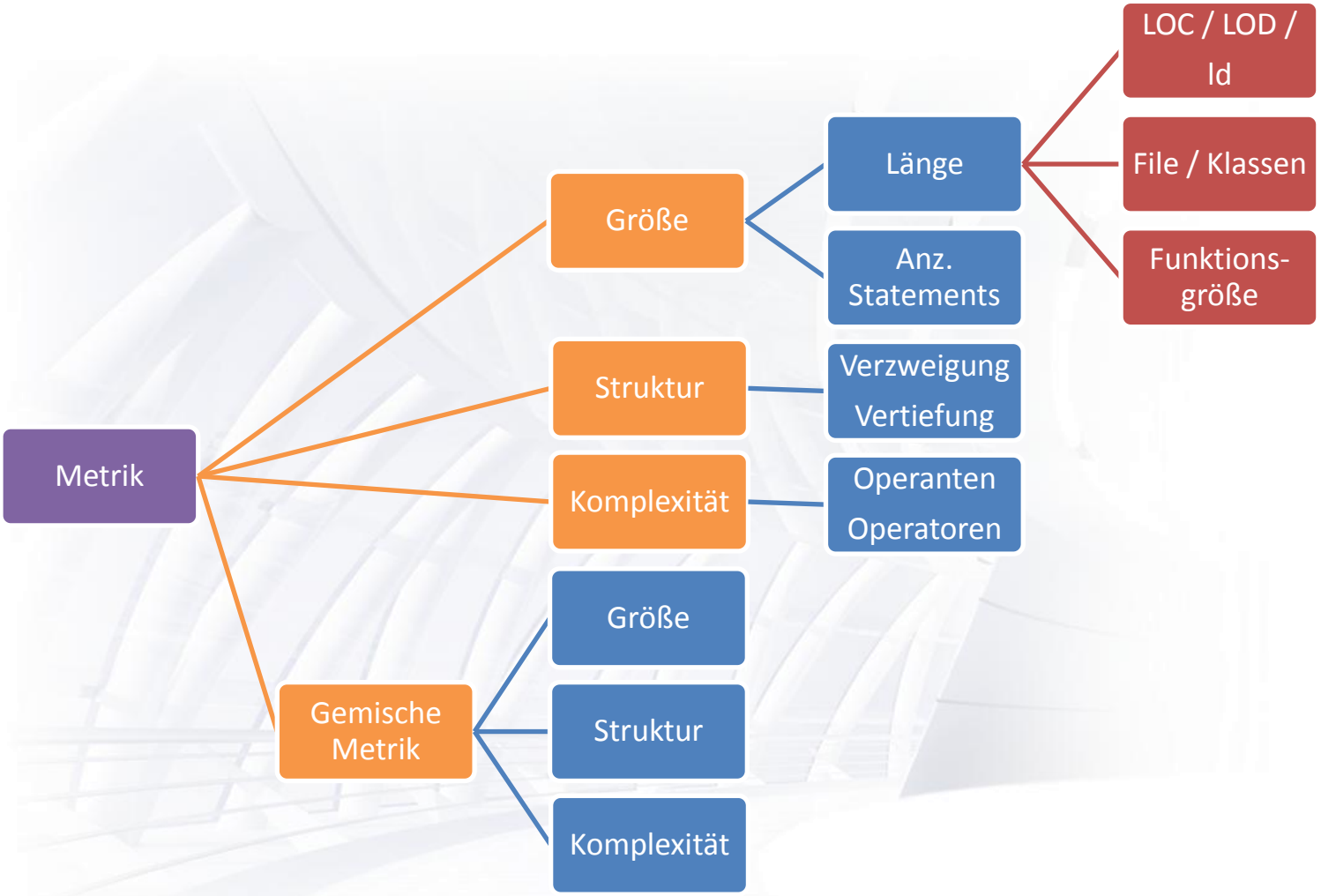
- *374.000 Zeilen Programmcode enthalten ein mittleres finanzielles Risiko von kalkuliert 1Millionen $*
- *500 Milliarden $ beträgt diese „Technical Debt“ technische Hypothek weltweit*
- *1 Billionen $ das Doppelte im Jahre 2015*



wünschenswerte Charakteristik und messbare Eigenschaften



Metriken zur Messung der Software Qualität



Code Beispiel

```
PROCEDURE err_msg(msg   IN VARCHAR2, type  IN INTEGER,loc   IN VARCHAR2 DEFAULT '') IS con_name VARCHAR2(240);
BEGIN
con_name := cg$errors.parse_constraint(msg, type);
IF (con_name = 'BELPOS_PK') THEN
cg$errors.push(nvl(BELPOS_PK
,cg$errors.MsgGetText(cg$errors.API_PK_CON_VIOLATED
,cg$errors.APIMSG_PK_VIOLAT,'BELPOS_PK','BELPOS')),'E','API',cg$errors.API_PK_CON_VIOLATED,loc);
ELSIF (con_name = 'BELPOS_BELWER_FK') THEN
cg$errors.push(nvl(BELPOS_BELWER_FK,cg$errors.MsgGetText(cg$errors.API_FK_CON_VIOLATED
,cg$errors.APIMSG_FK_VIOLAT,'BELPOS_BELWER_FK','BELPOS')),'E','API',cg$errors.API_FK_CON_VIOLATED,loc);
ELSE cg
END er

------
PROCED
```

```
--              loc      Place where this procedure was called for error
--                       trapping
--------------------------------------------------------------------------------
PROCEDURE err_msg(msg    IN VARCHAR2,
                  type   IN INTEGER,
                  loc    IN VARCHAR2 DEFAULT '') IS
con_name VARCHAR2(240);
BEGIN
    con_name := cg$errors.parse_constraint(msg, type);
    -- Decision about creation
    IF (con_name = 'BELPOS_PK') THEN
        cg$errors.push(nvl( BELPOS_PK,
                            cg$errors.MsgGetText( cg$errors.API_PK_CON_VIOLATED,
                                                  cg$errors.APIMSG_PK_VIOLAT,
                                                  'BELPOS_PK',
                                                  'BELPOS')
                                                ),
                            'E',
                            'API',
                            cg$errors.API_PK_CON_VIOLATED,
                            loc);

    ELSIF (con_name = 'BELPOS_BELWER_FK') THEN
```

Warum Software Qualität

- saubere Architektur (Templates)
  - Klare Zuordnungen z.B. DAL
- Komplexität kontrollierbar
  - Programm- / Funktionsgröße
  - Struktur und Strukturtiefe
- macht eine Applikation beherrschbar
  - lesbar
  - modular
  - portabel
  - …
- Entwicklungsrichtlinien (Coding Standards)
- Dokumentation
  - Funktionale Dokumentation
  - Code Verwendungen
- reduziert Fehler
- spart Kosten
- erhält und steigert den Wert einer Applikation

Warum Software Qualität - ein Parameter der Modernisierung

Einsatz von Werkzeugen zur Absicherung der Software Qualität

- klare Ansätze / Informationen überall greifbar
- Prüfungen wiederholbar
- Audit-fähig -> Zertifizierung
- Qualitätsverbesserung oder Qualitätserhalt ist messbar

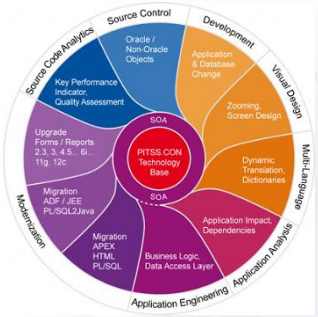
- Eingangskontrolle für Fremdsoftware bei größeren Unternehmen z.B. BMW

PITSS.CON im Besonderen

- auf Forms Applikationen ausgelegt
- Vermessung der ganzen Anwendung über alle Module FMF /PLL / MMB / DB / …
- mehrere Verfahren LOC / LOD / CRUD / McCabe / Halstead / MI-Index
- Top-Listen

- operative Analytik

- Fazit: Automatismen ein muss für Applikationen



PITSS.CON Live

- Source Code Analytics
- Denkensweise
- Beispiele

professional it software & services

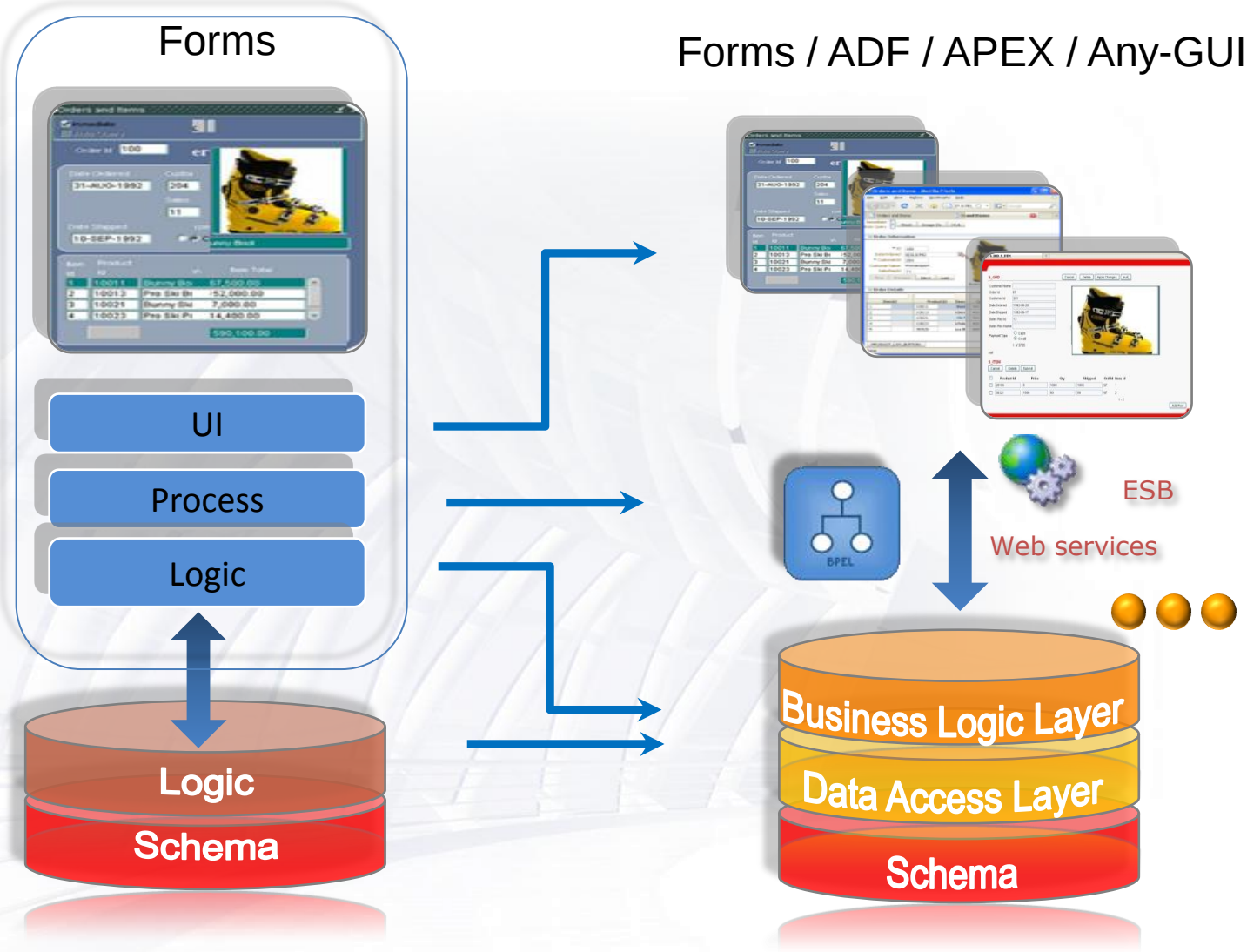
pitss®

Business Logik 2 Datenbank BL2DB

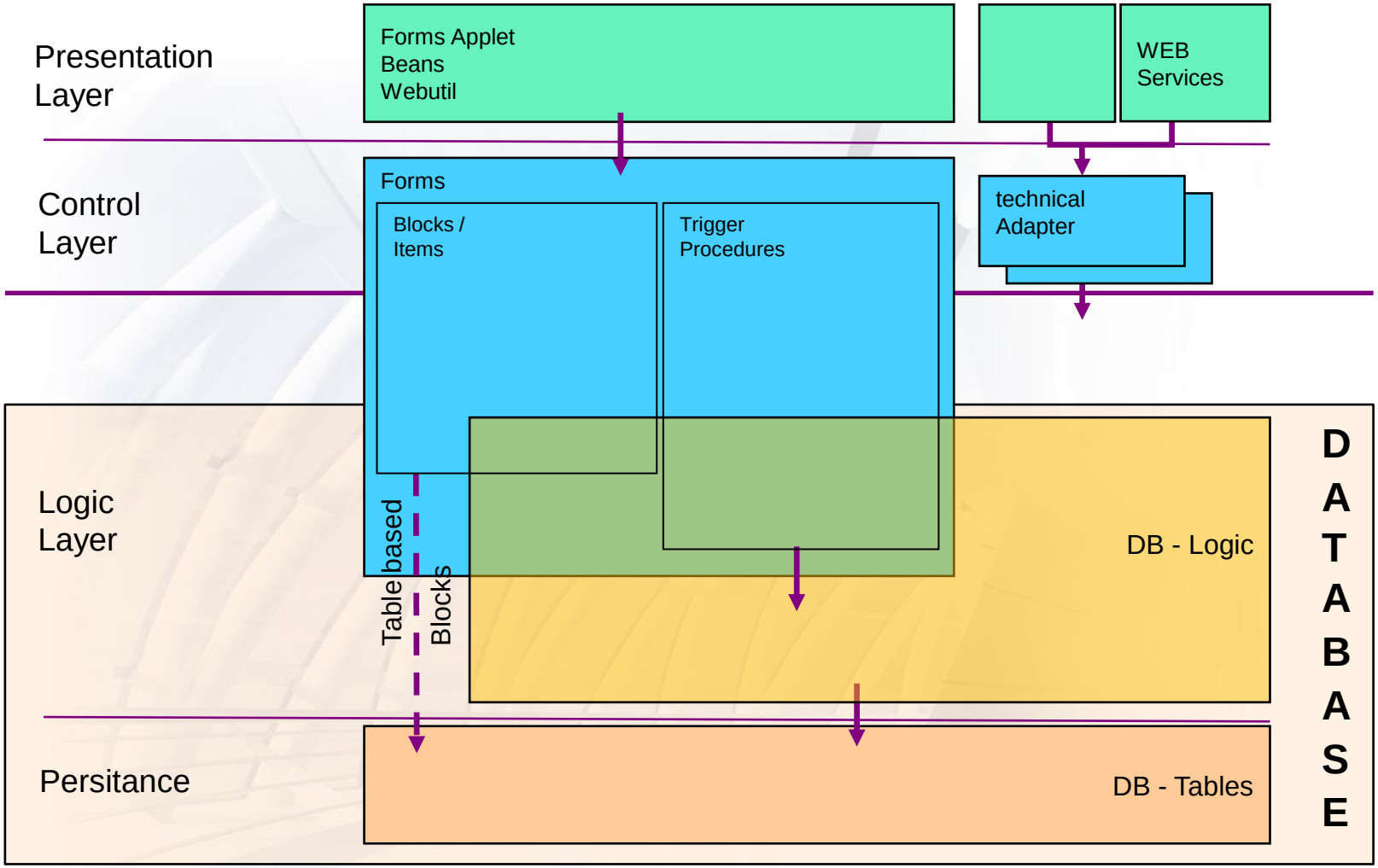
The Oracle Modernization Experts

ORACLE Gold Partner

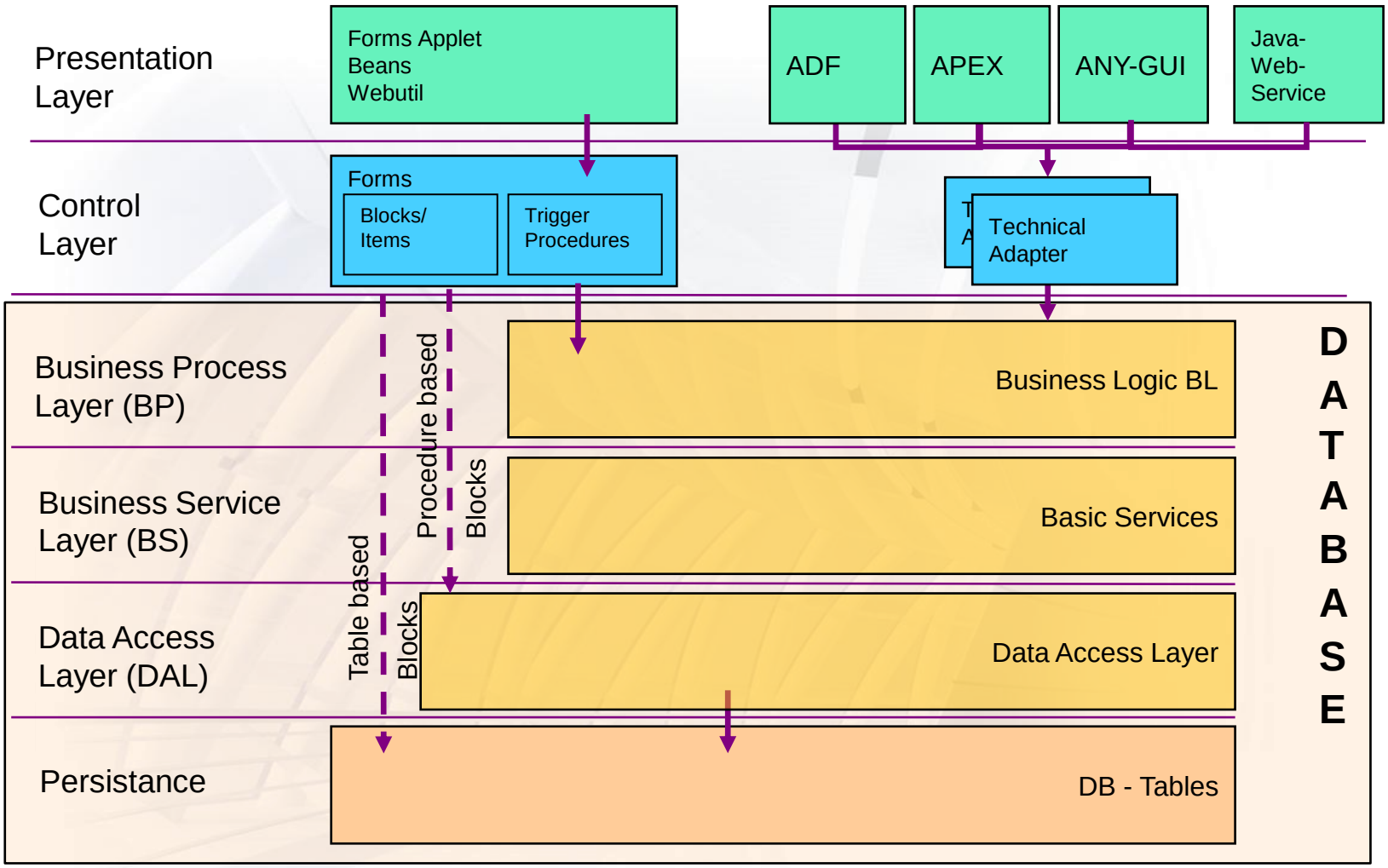
Re-Factoring of Business Logic BL



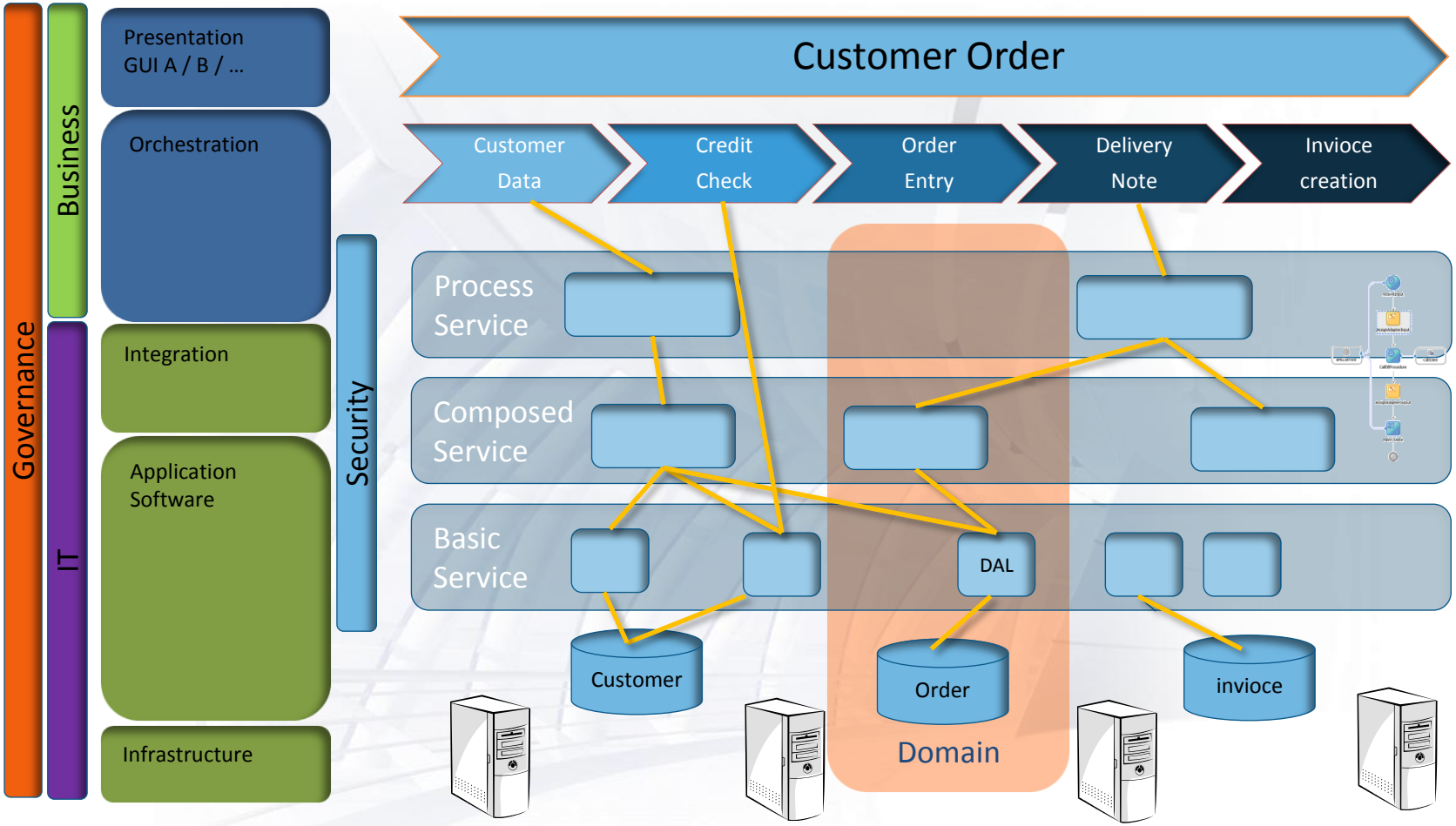
Forms Architektur



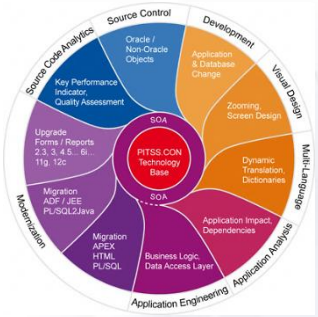
Zielarchitektur



Service Oriented Architectur



PITSS.CON live



PITSS.CON Live

- DAL
- BL2DB
- Logikerkennung
- Arbeitsweise
- Pool – Unused Code
- Beispiel

professional it software & services

pitss®

Agile Entwicklung mit PITSS.CON

The Oracle Modernization Experts

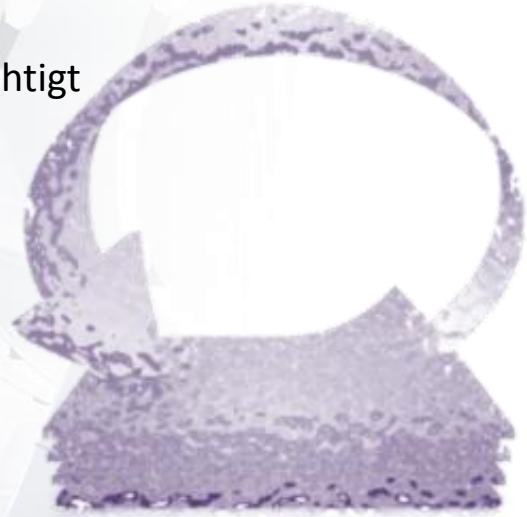
ORACLE Gold Partner

Elemente der agilen Entwicklung

Agile Entwicklung - Scrum

Klassische Entwicklungen

- Entwicklung / -Phase laufen zu lange ca. 1 Jahr
- Klassische Phasen 30% Spez, 40% Entw, 30 Test
- Missverständnisse, Fehler und Korrekturen verzögern erneut
- kurzfristig Marktanforderungen nicht berücksichtigt
- einstige Anforderungen nicht mehr relevant
- Kunde ist unzufrieden



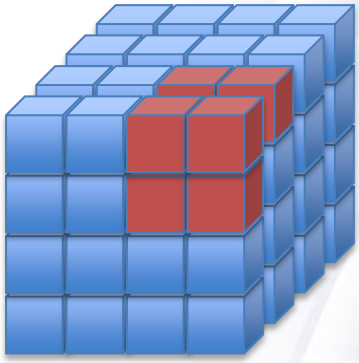
Agile Entwicklung

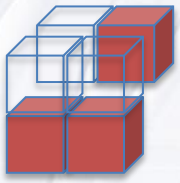
- kurze Entwicklungszeiten ca. 2-3 Wochen
- weniger Dokumentation mehr Funktionalität
- On Demand
- kleine Entwicklungseinheiten
- schnelle Verbesserung beim Kunden
- größere Entw-Vorhaben sind sehr gut getestet (in vielen Sprints)
- Eigenverantwortung beim Entw-Team

- automatische Tests notwendig

Agiler Entwicklungsprozess mit PITSS.CON

Product Backlog
Gesamt Anforderungen
Zerlegt durch PITSS.CON

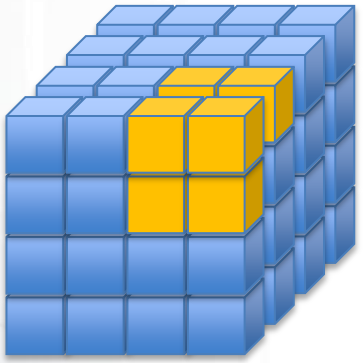




Sprint Backlog
ToDo
Teilanforderungen
Herausgelöst durch PITSS.CON
Gruppen zugeordnet (Pools)

Sprint Backlog
Done

Releasebildung und Delivery
mittels
PITSS.CON VC

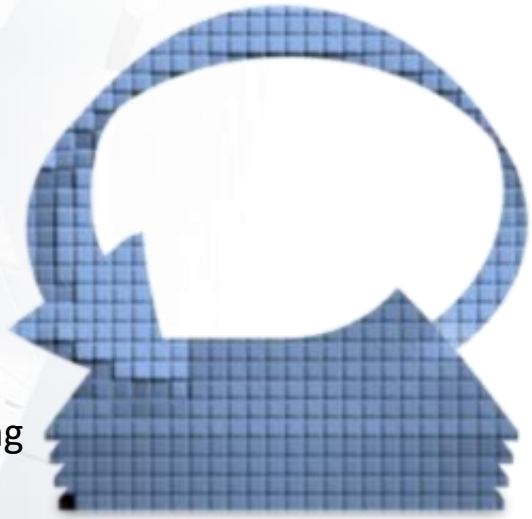


Test
mittels
RUEI
Oracle ATS

Agile Entwicklung mit PITSS.CON

PITSS.CON Eigenschaften der agilen Entwicklung

- schnelle Entscheidungsfindung durch detaillierte Analyse
- genaue Aufwandsbetrachtungen
- geführte Prozesse
  - ToDo – In Progress - Done
- Step by Step Prozess
  - Verbesserung pro Step
- Verfeinerung der Arbeitseinheiten
- Teamfähigkeit, Arbeitsteilung (Pools)
- Clusterbildung
  - Reduzierung des Arbeitsvolumens
- leistungsfähige, fehlerfreie Generatoren
- einheitliche Objektstruktur
- Deployment-Optimierung mit Versionsverwaltung



- implementierbares Vorgehensmodell

Unterstützung durch

- automatische Tests
  - Recording mittels RUEI
  - Testläufe mittels ATS

professional it software & services

pitss®

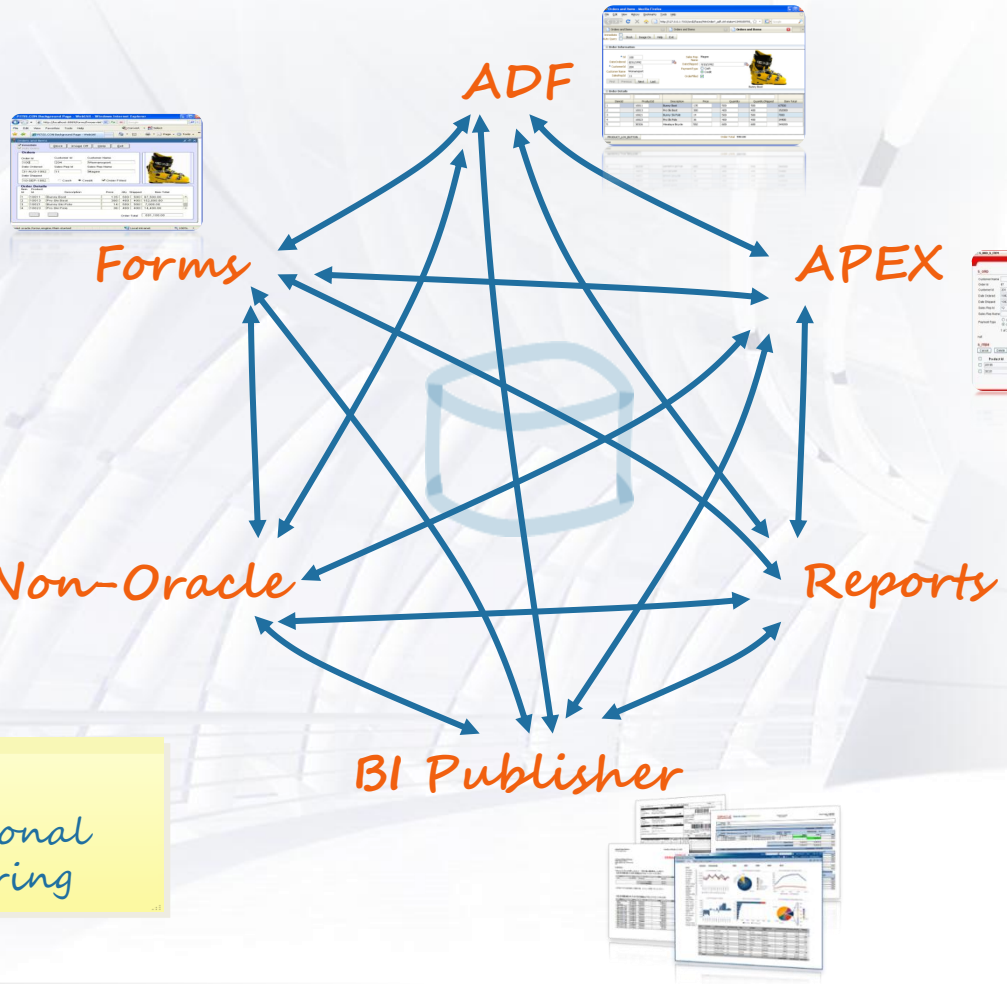
Migrationsphase

The Oracle Modernization Experts

ORACLE Gold Partner

Spiel mit den Technologien

Wie können Forms, ADF und APEX (peacefully) ko-existieren?



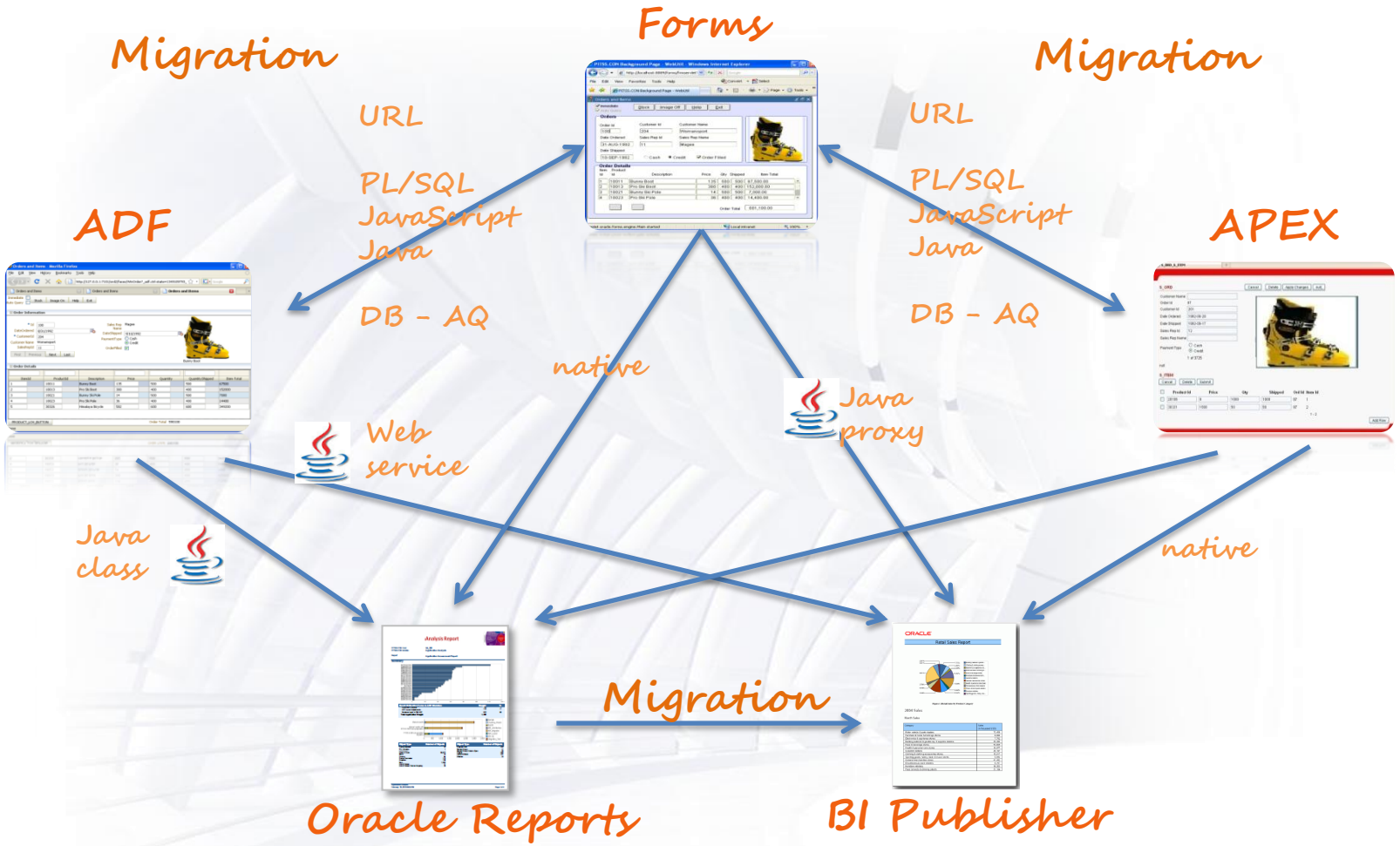
!!!
Passing
Parameters

!!!
Transaction
Management

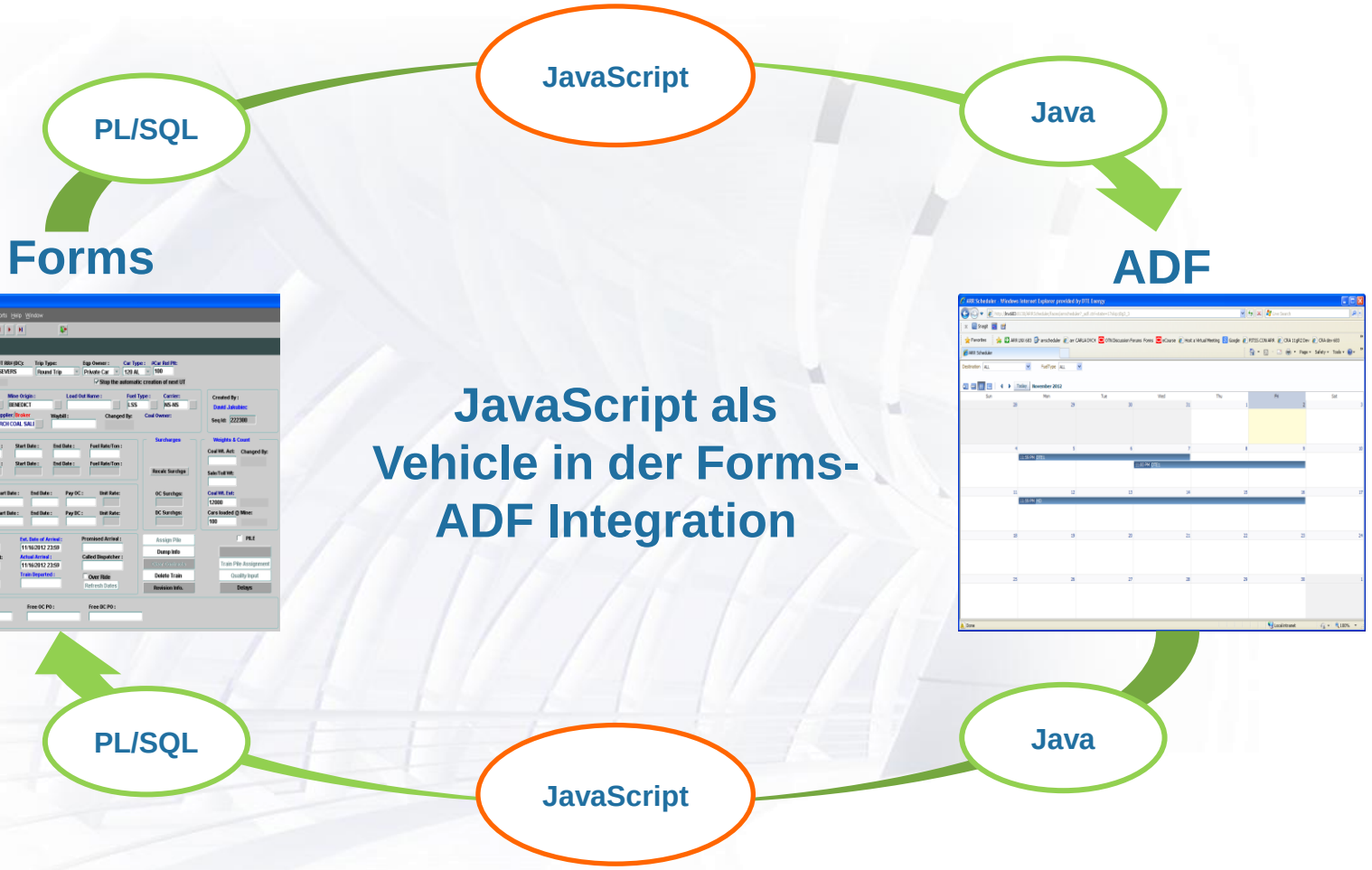
!!!
Functional
Clustering

!!!
Migration
Process

Integration der Oracle Technologien



Ko-Existenz



professional it software & services

pitss®

Generatoren

The Oracle Modernization Experts

ORACLE Gold Partner

Automaten generieren gleichen Code über die ganze Anwendung

WHO really wants

to build

his application

manually?

wenn

- es ein Dokument gibt, dass besagt wie ein Anwendung idealerweise aufgebaut werden soll

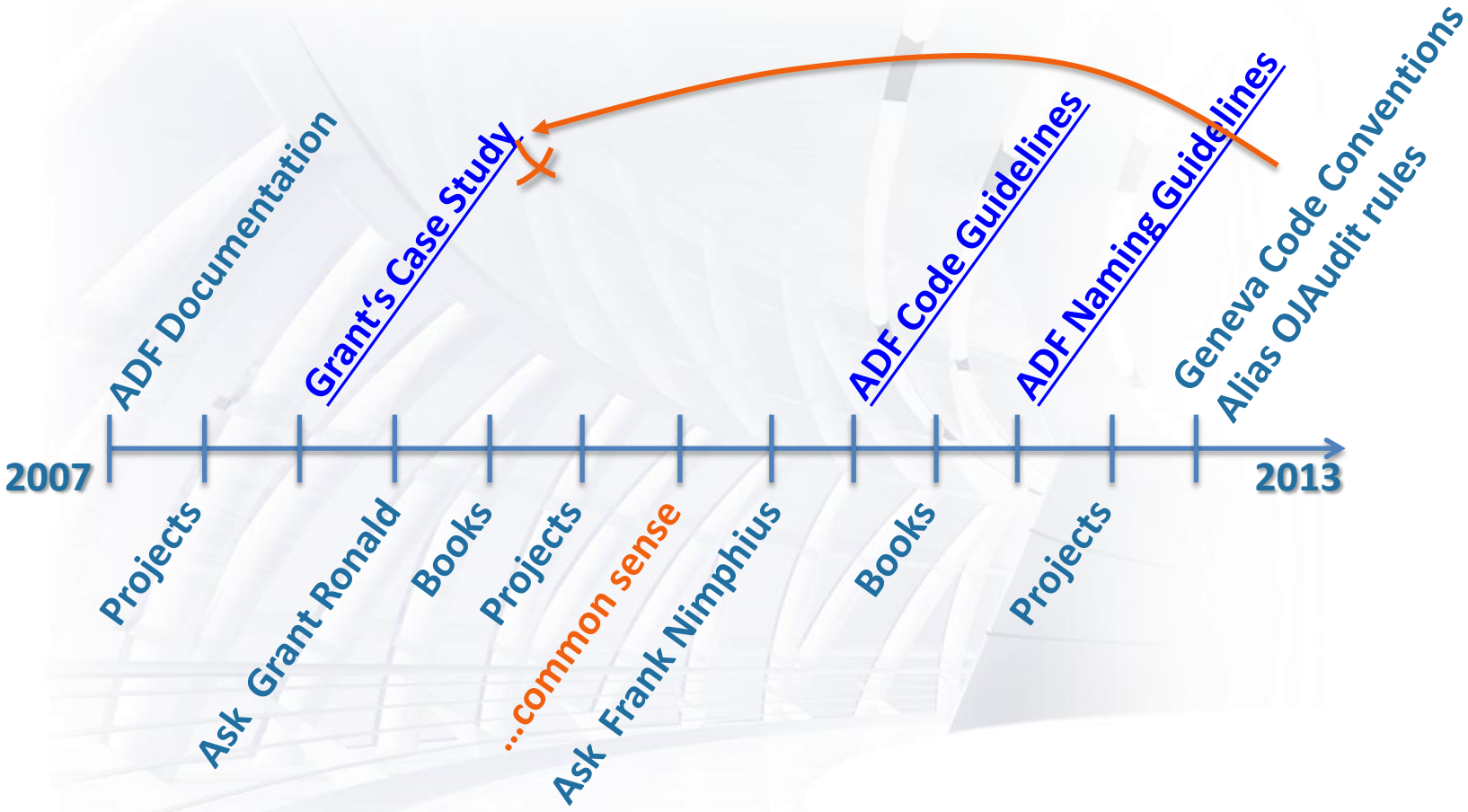
  http://www.oracle.com/technetwork/developer-tools/adf/learnmore/adfarchitect-1639592.html

  - *ADF Code Guidelines* and *ADF Naming Guidelines* from **Chris Muir** Oracle Product Manager

und

- es einen automatischen Ansatz gibt, der in der Lage ist 90 bis 95% abzudecken

Regel der Präzedenz? Wenn Zweifel, dann anwenden…



Software und Generatoren

- verständlicher Code
- gleiche Qualität über alle Module, Anwendungen …..
- leicht zu warten, Wiedererkennung der Code Muster (Software Pattern)
- einfache Anpassbarkeit
- saubere Struktur über mehrere Ebene

- Trennung Basis-Technologie / Geschäftslogik

- hohe Effizienz im Entwicklungsprozess
- Konzentration auf die Technik, damit steile Lernkurve

- Ansätze reproduzierbar
- Kosten optimiert
- Risiko minimiert

- Re-Engineering Ansätze einfacher

- Erfolgsfaktor in der Applikationsentwicklung
- Framework-Ersatz

- Bekannte Generatoren: Wizards für Forms, Reports usw., CASE Tools, deklarative Umgebungen, ……
- Kerngedanke des Modelgetriebenen Ansatzes DSL

professional it software & services pitss®

Cluster-Bildung

The Oracle Modernization Experts

ORACLE Gold Partner

Warum Cluster ?

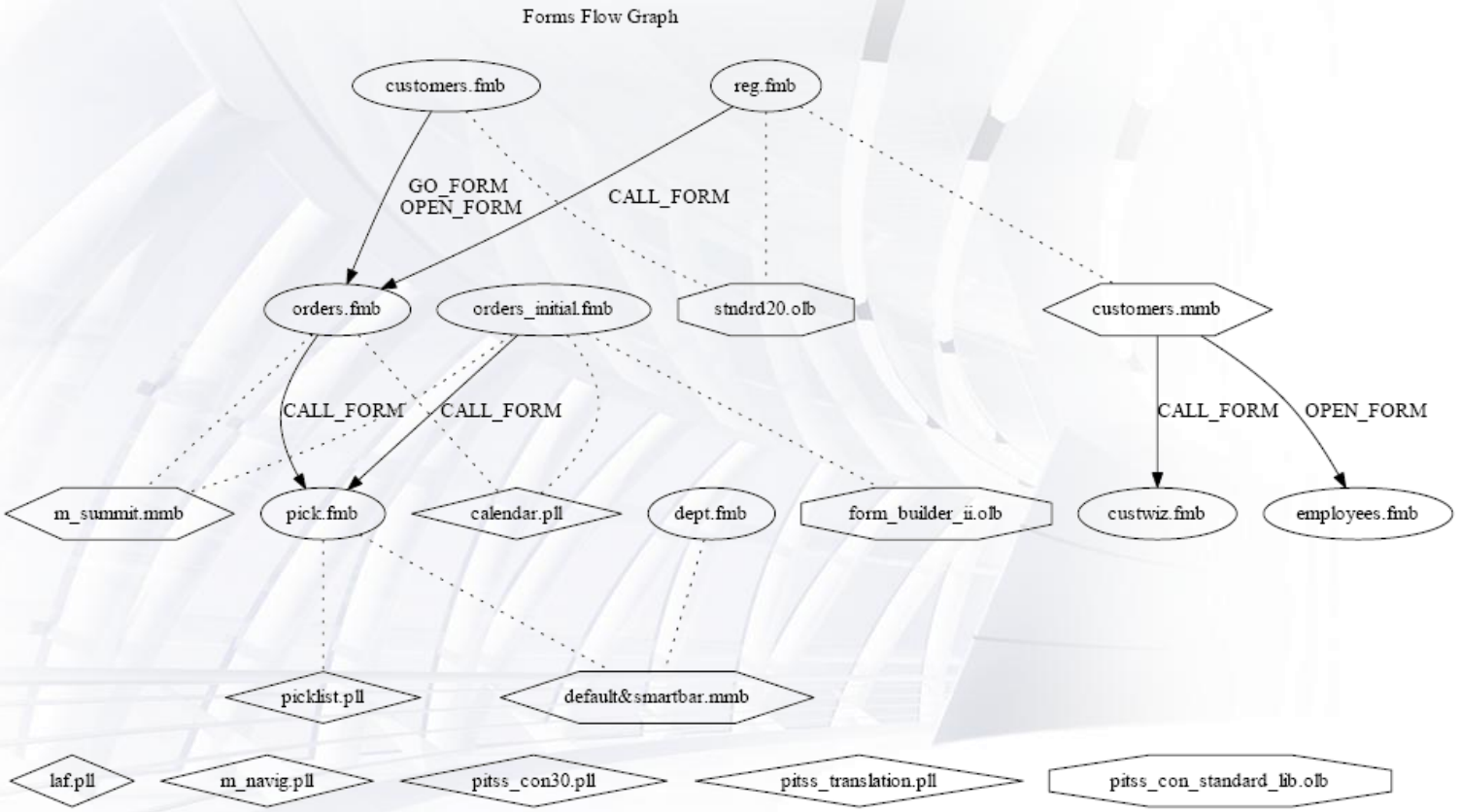
- aufbrechen monolithischer Systeme

- klare Ordnungskriterien
- klare Strukturierung
  - z.B. nach Geschäftsprozessen
- kleinere Einheiten – Modularität
  - einfacher zu warten
  - modular austauschbar
- einheitliche Kommunikation zwischen Cluster (Basisfunktionalität)
  - einheitliche Schnittstellen
  - Standards z.B.: WEBService

- SOA
- Shared objects in Java Technologien

- unterstützt Ko- Existenz mit anderen Technologien
- Schlüssel zur erfolgreichen Modernisierung

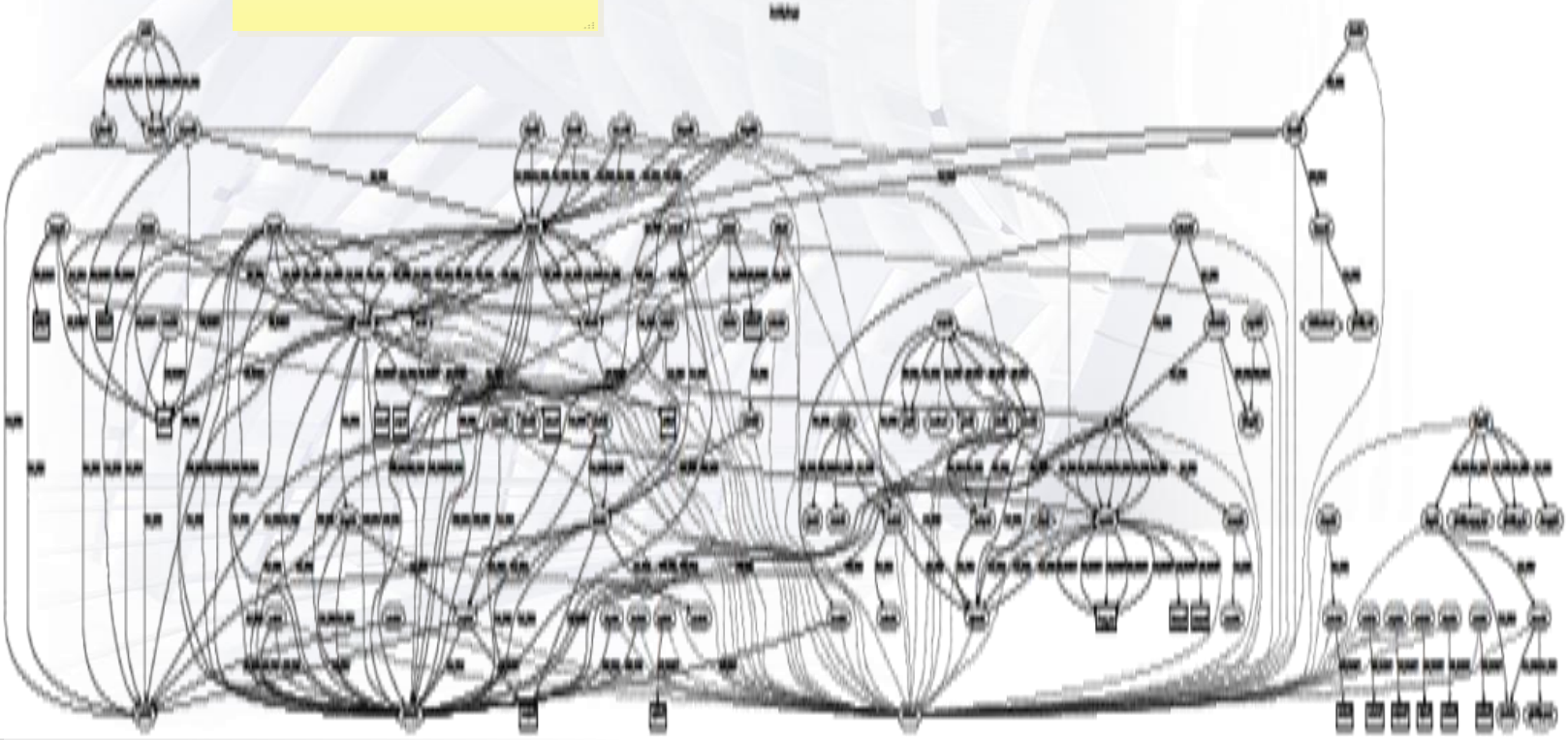
Demo-Applikation



Beispiel: 100 FMB-application

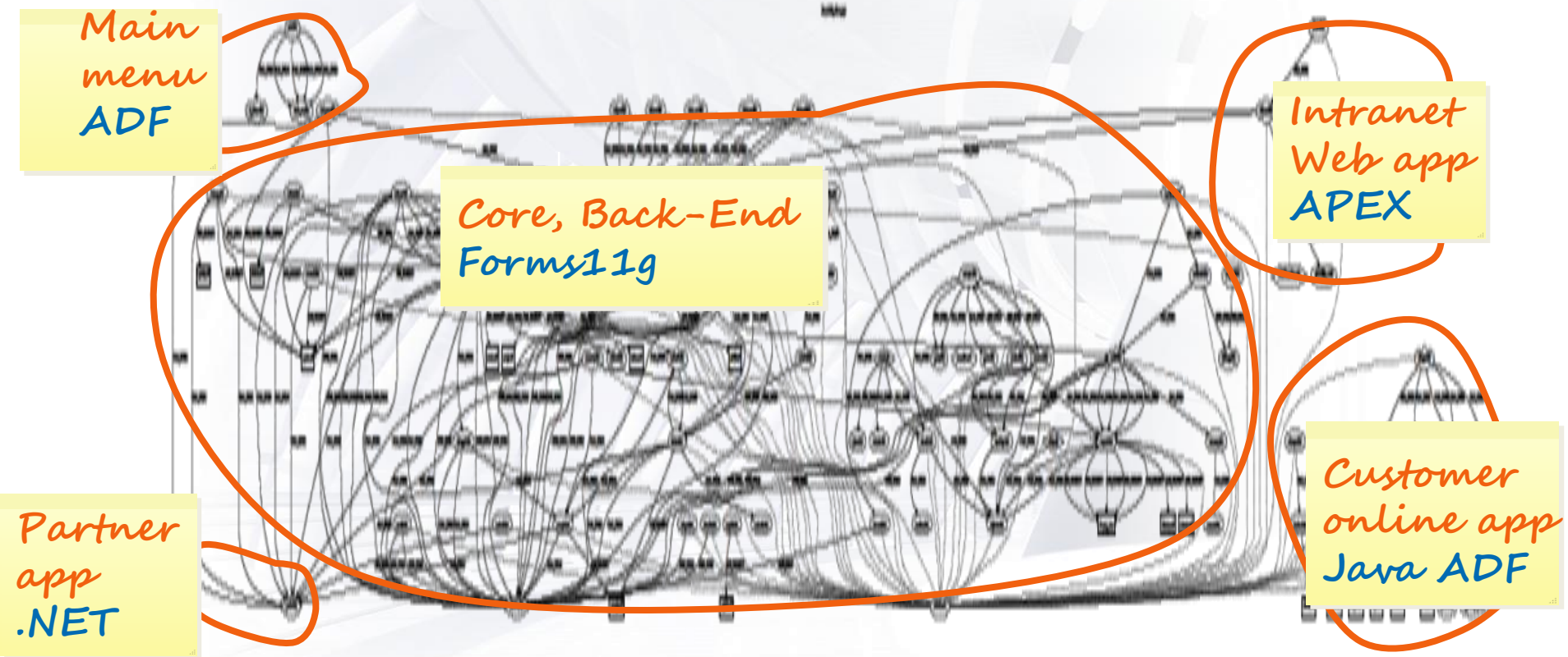
=>
Clustering

=>
Multiple Technologies



Unsere Kundenvorliebe:

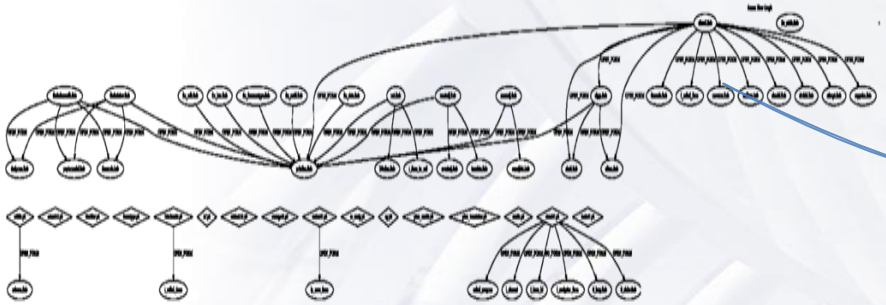
Technologie Mix zwischen der funktionalen Cluster

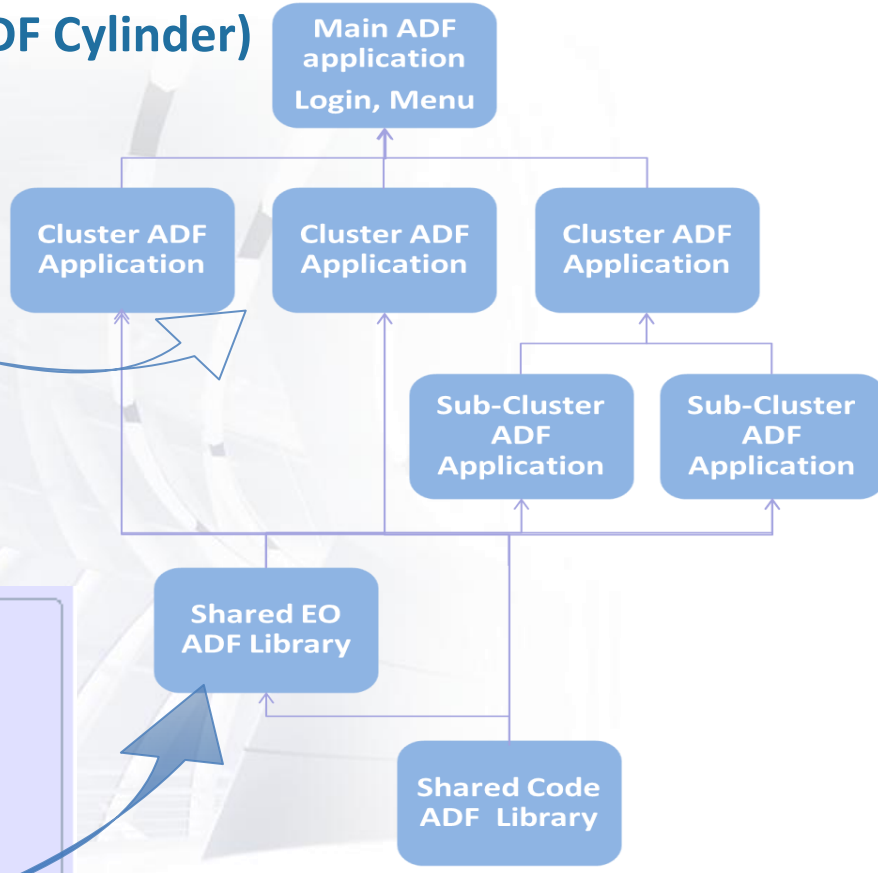


ADF Architektur Beispiel

PITSS.CON Forms Flow => Clustering

Clustering = Trennung nach Aufgaben (= ADF Cylinder)



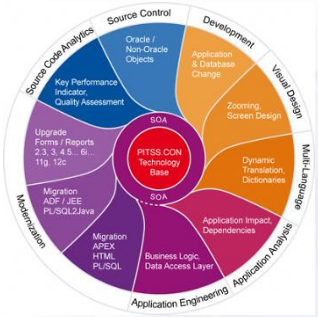


PITSS.CON ADF Settings

Model Project

Application Module Name	RetekService
Project Name	Model
Package Name	retek.recutadj.model
Application Module Package	retek.recutadj.model.services
Entity Object Package	retek.recutadj.model.entities
Association Package	retek.recutadj.model.entities.associations
View Object Package	model.queries
View Link Package	model.queries.viewlinks
Shared E.O. App. Workspace	

PITSS.CON live



PITSS.CON Live

- Application Flow
- ADF Generierung

professional it software & services

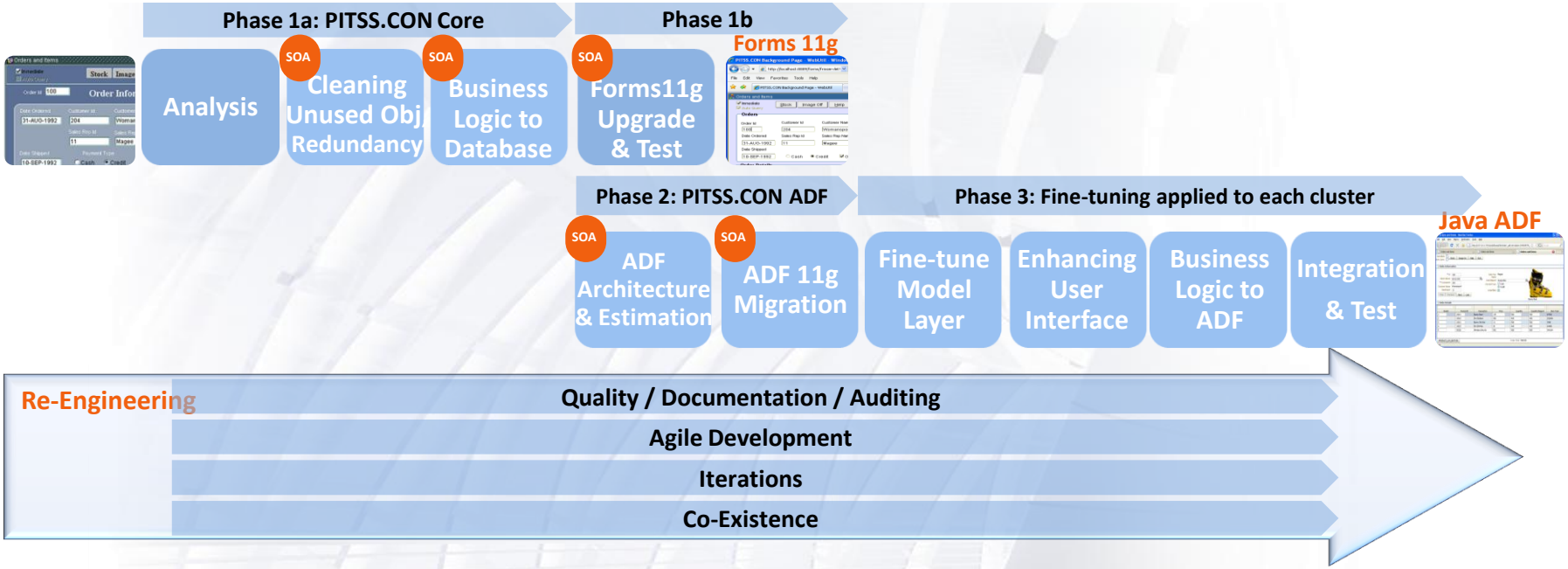
pitss®

Vorgehensmodell - ADF

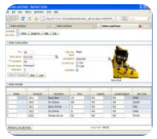
The Oracle Modernization Experts

ORACLE Gold Partner

PITSS.CON Modernization Process in ADF Projects

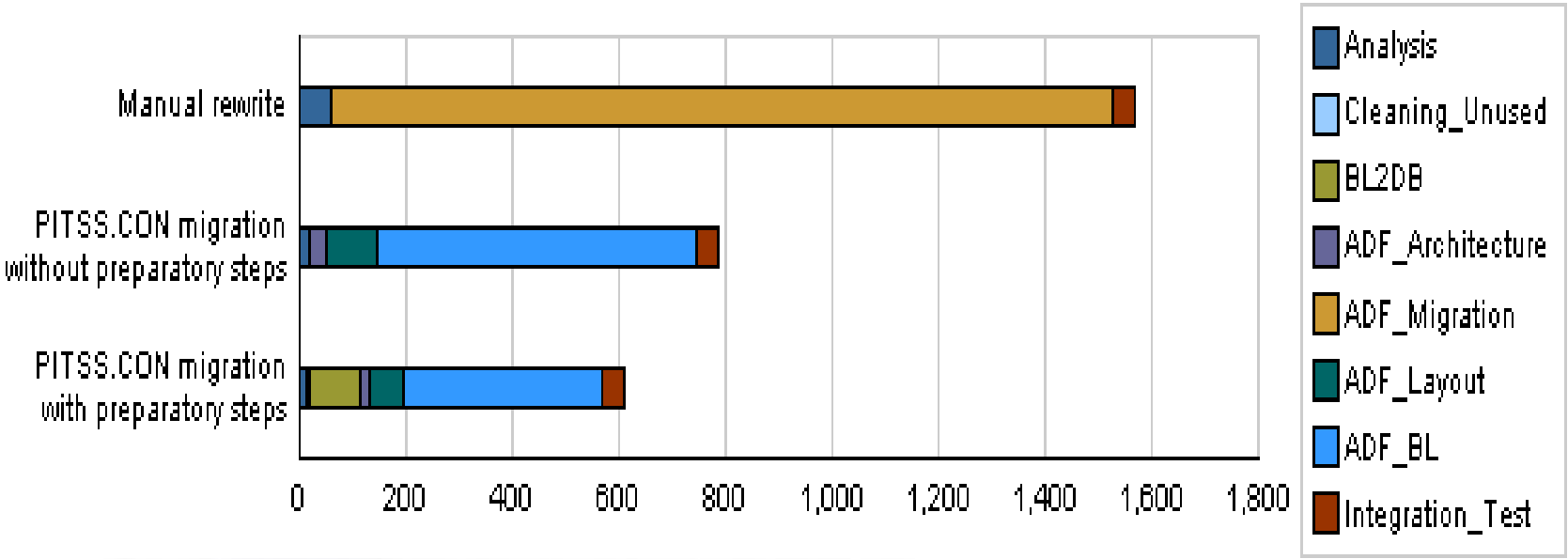


ADF Architecture & Estimation | ADF 11g Migration | Fine-tune Model Layer | User Interface | Business Logic to ADF | Integrate & Test



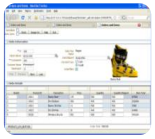
Genaue Aufwandsabschätzung: Pro Cluster und Phase

Zahlen für die Projektphase => Ressourcen-Planung



Forms to ADF Migration

ADF Architecture & Estimation | **ADF 11g Migration** | Fine-tune Model Layer | User Interface | Business Logic to ADF | Integrate & Test



Object	Converted
Blocks Master-Detail Relations Other Forms Objects: Alerts, Canvases, LOVs, Windows	**Yes**, alle Datenquellen **Yes**, in jeder Tiefe **Yes**, die häufig verwendeten Objekte werden übersetzt
Items: Text, Display, Button, Check Box, List, Radio Group, Tree, Image	**Yes**, außer Java Beans (die meisten sind obsolete in ADF)

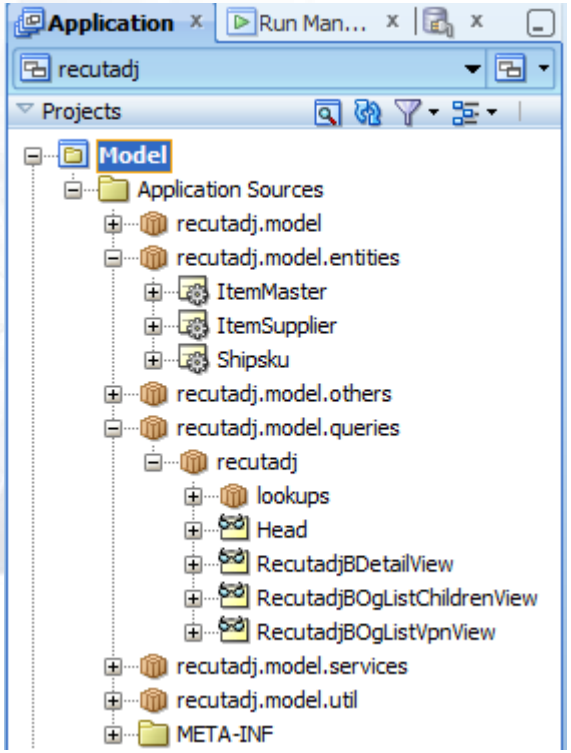
Conversion Degree		
Data Model	95%	'Hand-made' Architektur Basis für die gesamte Anwendung.
User Interface	75%	Relevante Objekte werden konvertiert.
Business Logic	55%	30-50% PL/SQL Code werden während der Cleaning-Phasen bereinigt. Optionale Migration Von PL/SQL 2 DB, Java oder manuellem Redesign.

Fine-tuning des Model Layer

Der Model Layer sollte nach der Migration 95% operativ sein

wann braucht der Model-Layer Fine-tuning:

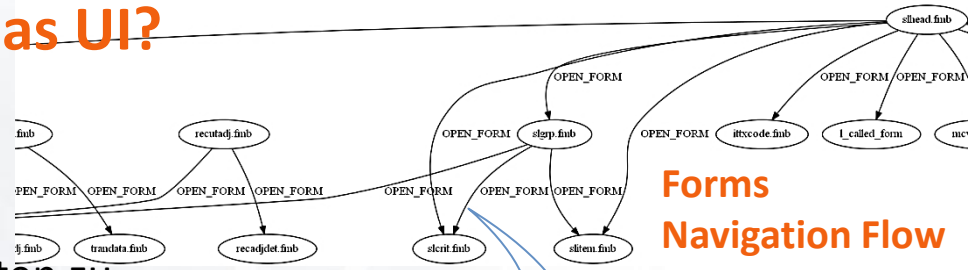
- wenn die Datenbank Definitionen nicht richtig oder unvollständig in PITSS.CON geladen wurden
- wenn es dynamische Konstrukte enthält

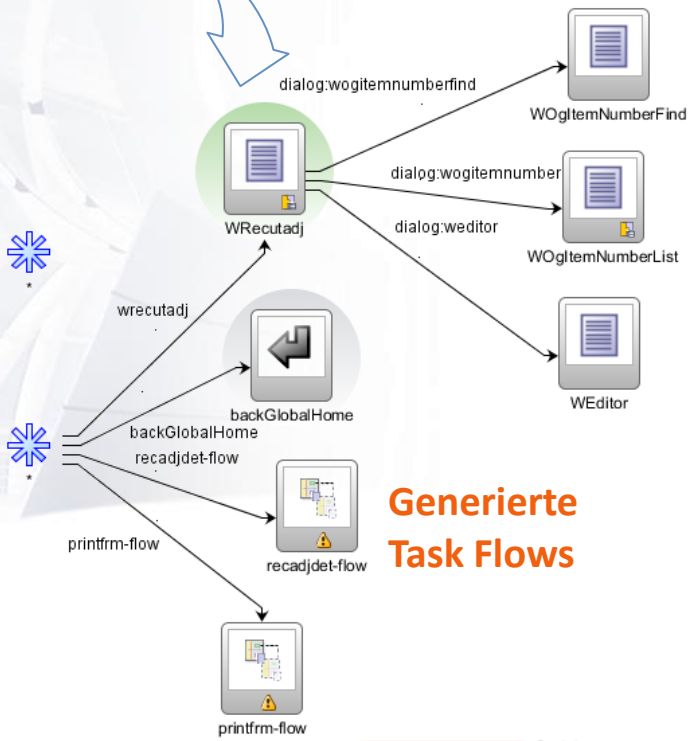


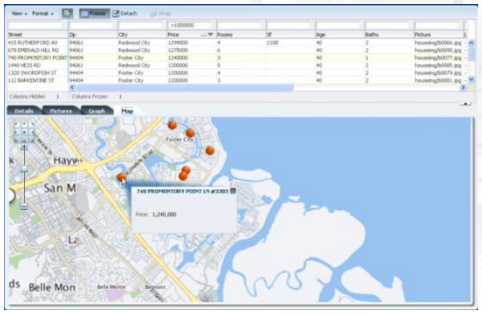
Erweiterung des User Interfaces UI

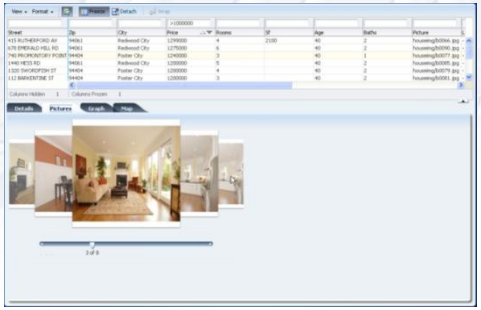
Welche Änderungen braucht das UI?

- Fine-tuning des Navigation-Flow
- Drag-and-Drop, um die UI Komponenten zu positionieren
- Um moderne WEB Komponenten hinzuzufügen









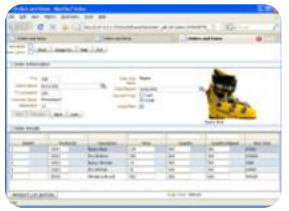
PITSS Modernisierungs-Ansatz

Software Quality / Documentation / Auditing

Agile Development

Analysis | Dead Code | Redundancy | SOA BL2DB

ADF



APEX

Iterations

Co-Existence of Technologies/ Partial Projects

professional it software & services

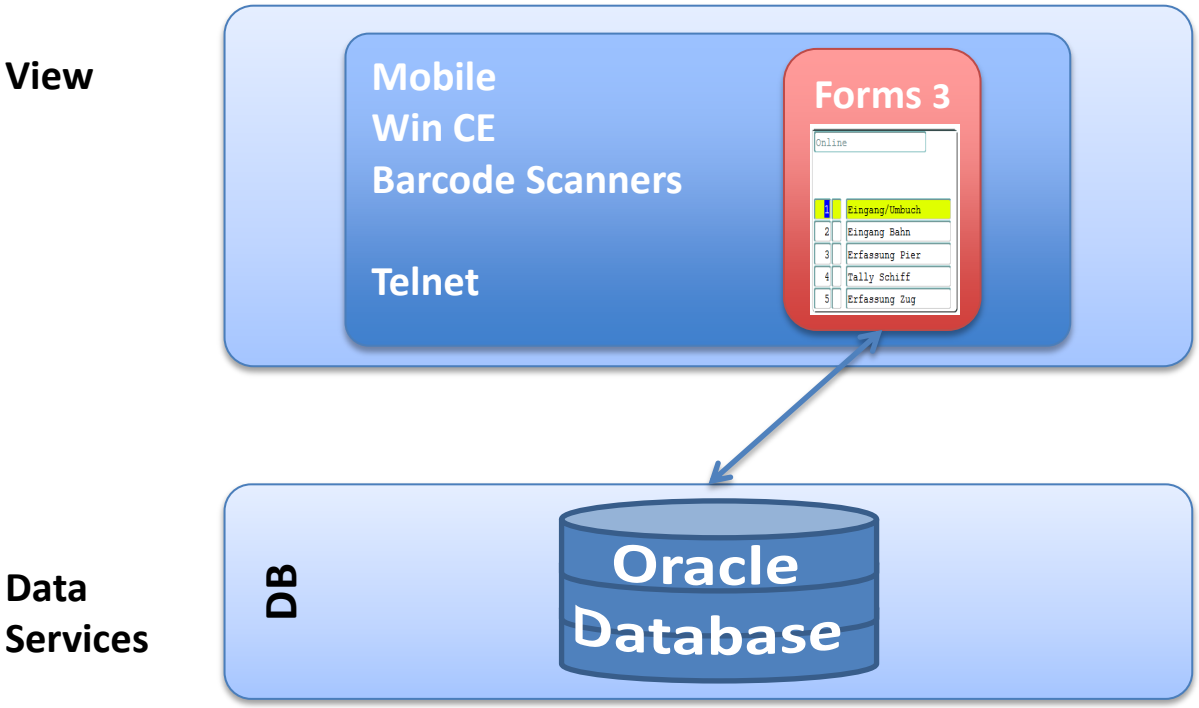
pitss®

aktuelles Migrationsprojekt

The Oracle Modernization Experts

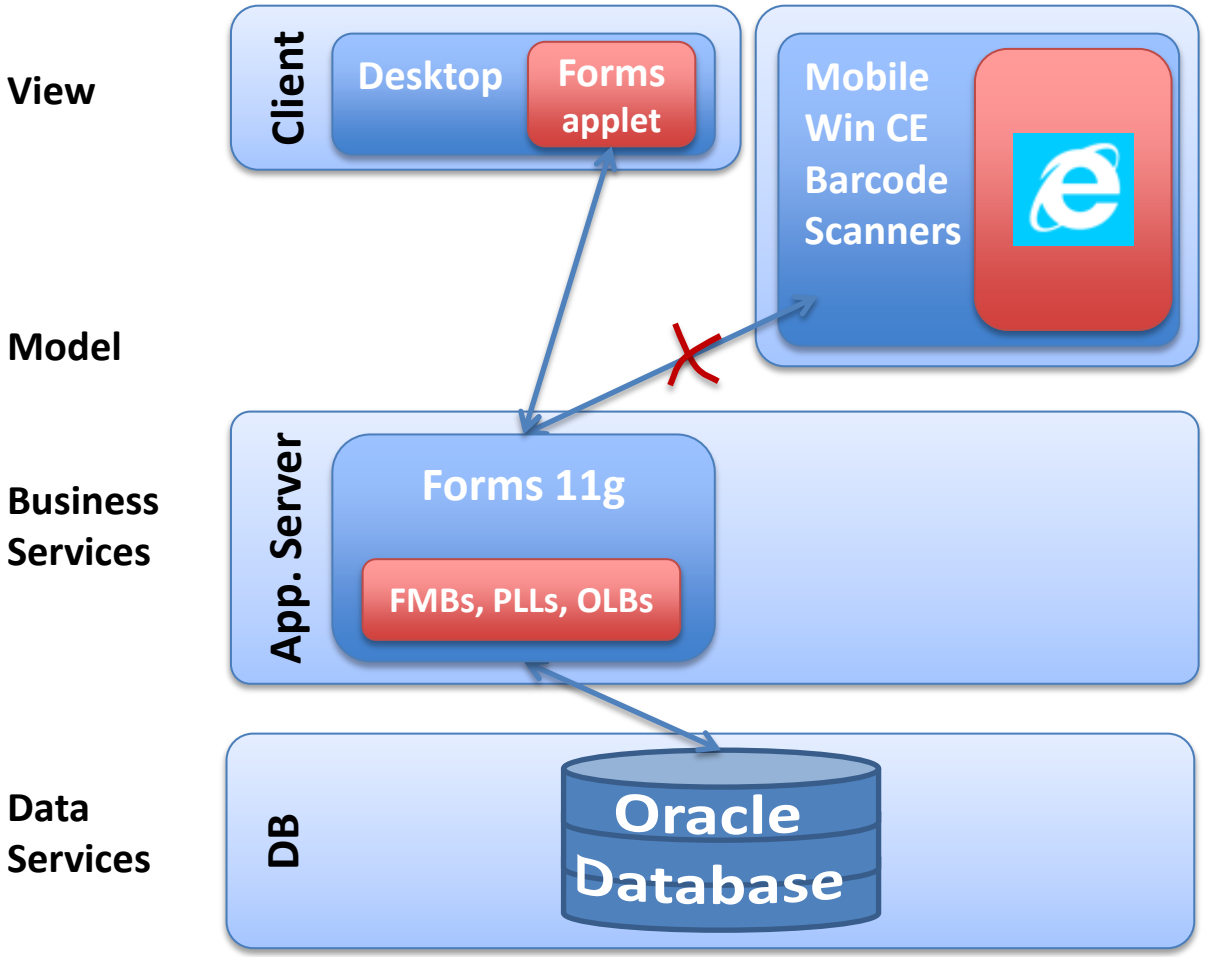
ORACLE Gold Partner

Die Herausforderung: Scanner mit Forms 3.0



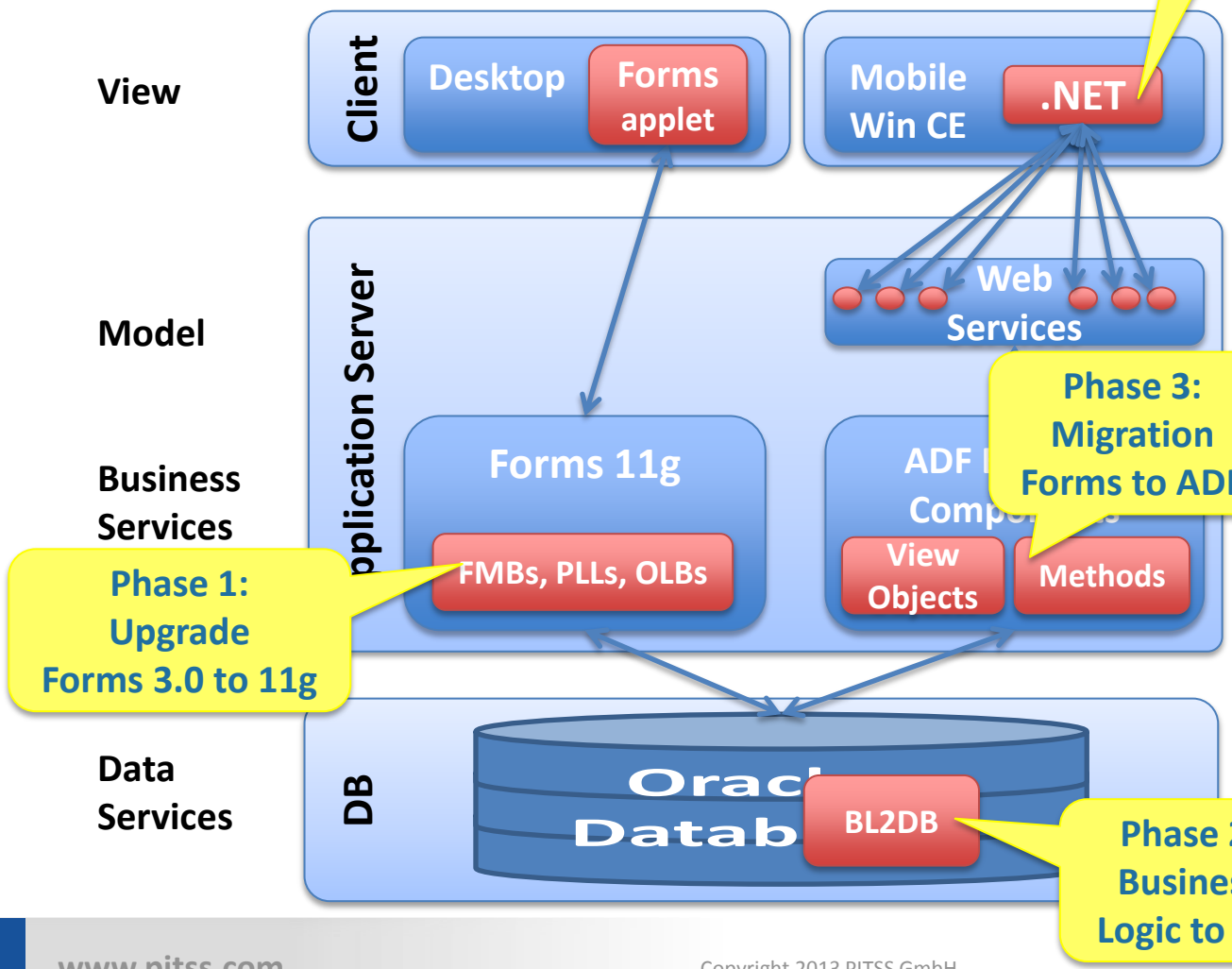
- Logistik Anwendung geschrieben in Forms3.0 mit Scannerprogrammen in Ko-Existenz
- Upgrade auf 11g der Logistik Anwendung
- ??????
  Was passiert mit den Scannern

Problemstellung: Barcode-Scanner



- Brauch einheitlich Security für WebLogic und mobile Applikation
- Legacy Hardware und Software: Windows CE 5.0 Barcode Scanners
- ADF Mobile Browser braucht JavaScript für Function Keys. JavaScript Browser nicht verfügbar Windows CE 5.0
- ADF Mobile nicht supported für Windows CE 5.0

Lösung: Forms, ADF Server + .NET Client



Thin Client

- Key-Functions
- Full-Screen
- modernes UI
- kompatibel mit aktuellen Geräten

Thick (90%) ADF + DB

- aktuelle DB
- Ko-Existenz mit der
- restlichen Forms Anwendung
- Model Layer enthält View Objekte und Methoden (Business Logik) angeboten als Web Services

Look and Feel

Before: Forms INP

After: ADF + .NET

Fragen und Antworten

professional it software & services

pitss®

Thank You

For Your Attention!

Andreas Gaede

The Oracle Modernization Experts

ORACLE Gold Partner