ORDIX AG

Best Practice PL/SQL

DOAG Webinar
Markus Fiegler
14.11.2014, Paderborn

info@ordix.de
www.ordix.de

Agenda

ORDIX AG

- Überprüfung der fachlichen Korrektheit in PL/SQL
  - PL/SQL-Unittest mit utPLSQL
- Automatische Code-Analyse
  - PL/SQL-Warnungen
  - PL/Scope
  - Auswertung der DD-Views
- Fazit

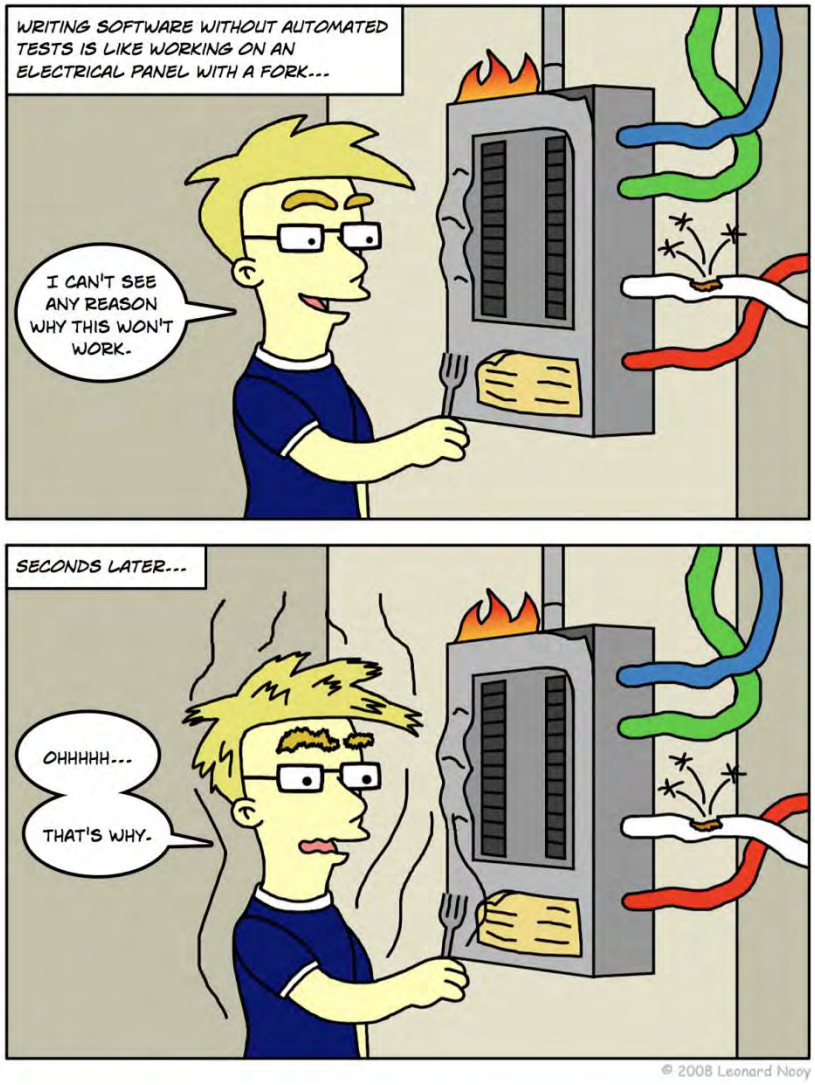
Agenda

ORDIX AG

- Überprüfung der fachlichen Korrektheit in PL/SQL
  - PL/SQL-Unittest mit utPLSQL
- Automatische Code-Analyse
  - PL/SQL-Warnungen
  - PL/Scope
  - Auswertung der DD-Views
- Fazit

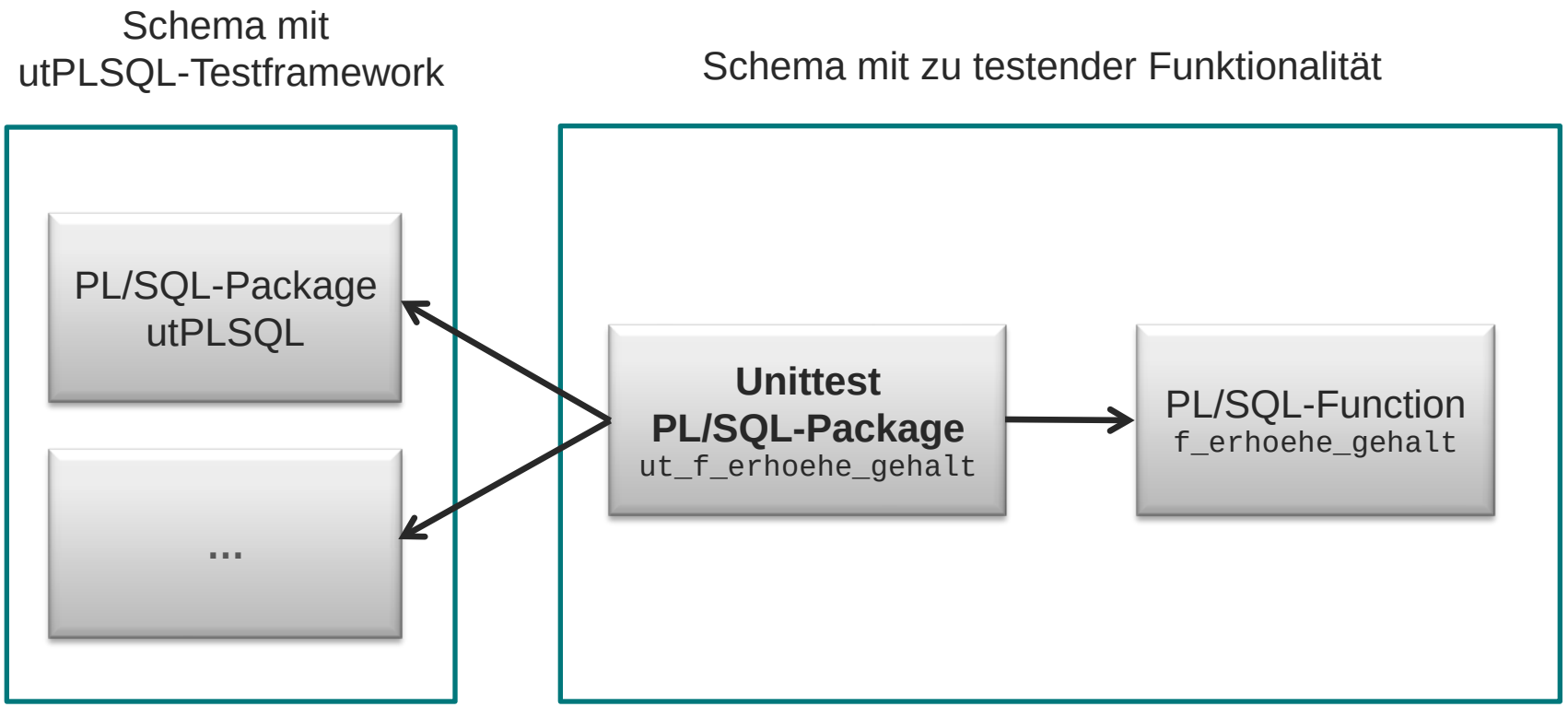
Motivation für Test-Automatisierung

ORDIX AG



Quelle:
http://www.leonardscomic.com

- OpenSource Test-Framework für PL/SQL (http://utplsql.sourceforge.net)
- Entwickelt von Steven Feuerstein
- Automatisierung der Komponententests (Unittest)
- Aufruf der zu testenden Funktionalität und Vergleich des Ergebnisses mit dem erwarteten Testergebnis
- Voraussetzung für Refactoring


Schema mit
utPLSQL-Testframework
Schema mit zu testender Funktionalität
PL/SQL-Package
utPLSQL
...
Unittest
PL/SQL-Package
ut_f_erhoehe_gehalt
PL/SQL-Function
f_erhoehe_gehalt

ORDIX AG

- Download und Entpacken von utplsql-2-2-X.zip
- utPLSQL DB-User anlegen und anmelden
- utPLSQL installieren:

```
SQL> @ut_i_do install
```

- utPLSQL deinstallieren:

```
SQL> @ut_i_do uninstall
```

Aufbau eines Unittest

ORDIX AG

- Ein Unittest besteht aus einem PL/SQL-Package mit mehreren Testprozeduren.
- Der Name einer Test-Prozedur muss folgenden Aufbau haben:
  - `<prefix><program>` z.B. `ut_test01` ("ut_" ist der Default-Präfixname)
- Ein Unittest muss zwei spezielle Prozeduren beinhalten:
  - `setup`
  - `teardown`
- Prüfung der gelieferten Ergebnisse mit den zu erwartenden Ergebnissen über Asserts (utAssert PL/SQL-Package)

PL/SQL-Packages des utPLSQL-Testframework

ORDIX AG

Das utPLSQL-Testframework stellt folgende PL/SQL-Packages zur Verfügung:

- utPLSQL
- utConfig
- utAssert
- utGen
- …

PL/SQL-Package

utPLSQL

ORDIX AG

Prozedur	Beschreibung
`utPLSQL.run`	Unittest ausführen
`utPLSQL.test`	Unittest testen, bedeutet Dateien (.pks und .pkb) mit einem Unittest-PL/SQL-Package in die Datenbank einspielen und ausführen
`utPLSQL.runsuite`	Test-Suite ausführen
`utPLSQL.testsuite`	Test-Suite testen, bedeutet Dateien (.pks und .pkb) mit Unittest-PL/SQL-Packages einer Test-Suite in die Datenbank einspielen und ausführen
`utPLSQL.version`	Version des utPLSQL Testframework anzeigen
...	...

PL/SQL-Package

utConfig

ORDIX AG

Prozedur	Beschreibung
`utConfig.showFailuresOnly`	Ausgabe der erfolgreichen Tests wird deaktiviert
`utConfig.showConfig`	Konfiguration anzeigen
`utConfig.setdir`	Setzt das Verzeichnis in dem sich die Unittest-PL/SQL-Packages befinden
`utConfig.dir`	Liefert das Verzeichnis in dem sich die Unittest-PL/SQL-Packages befinden zurück
`utConfig.setprefix`	Setzt das Standard Unittest-Präfix
`utConfig.prefix`	Liefert das Standard Unittest-Präfix zurück
…	…

PL/SQL-Package

utAssert

ORDIX AG

Prozedur	Beschreibung
`utAssert.eq`	Überprüfung auf Gleichheit von skalaren Werten
`utAssert.this`	Überprüfung von boolschen Ausdrücken
`utAssert.isNull`	Überprüfung auf NULL
`utAssert.eqTable`	Überprüfung auf Gleichheit von Tabellen
`utAssert.eqTabCount`	Überprüfung auf Gleichheit von Anzahl der Datensätze
`utAssert.eqQuery`	Überprüfung auf Gleichheit von Abfrageergebnissen
`utAssert.eqQueryValue`	Überprüfung auf Gleichheit von einer Abfrage mit einem vorgegebenen Wert
`utAssert.eqFile`	Überprüfung auf Gleichheit von Dateien
`utAssert.throws`	Überprüfung ob eine Exception geworfen wird
…	…

PL/SQL-Package

utGen

ORDIX AG

Prozedur	Beschreibung
`utGen.testpkg`	Generiert ein Unittest-Package für ein zu testendes PL/SQL-Objekt mit Aufrufparametern und Rückgabewerten
…	…

Beispiel mit utPLSQL (I)

ORDIX AG

```
-- Funktion f_erhoehe_gehalt anlegen...
CREATE OR REPLACE FUNCTION f_erhoehe_gehalt(p_geh NUMBER, p_proz NUMBER)
  RETURN NUMBER IS
BEGIN
  IF   p_proz >= 0 THEN RETURN p_geh + (p_geh*p_proz/100);
  ELSE RAISE_APPLICATION_ERROR(-20001, 'FEHLER: p_proz muss >= 0 sein');
  END IF;
END f_erhoehe_gehalt;
/
-- Unittest PL/SQL-Package ut_f_erhoehe_gehalt anlegen...
CREATE OR REPLACE PACKAGE ut_f_erhoehe_gehalt IS
  PROCEDURE ut_setup;
  PROCEDURE ut_teardown;
  PROCEDURE ut_test1;
END ut_f_erhoehe_gehalt;
/
CREATE OR REPLACE PACKAGE BODY ut_f_erhoehe_gehalt IS
  PROCEDURE ut_setup IS BEGIN DBMS_OUTPUT.PUT_LINE('Start ut_setup'); END ut_setup;
  PROCEDURE ut_teardown IS BEGIN DBMS_OUTPUT.PUT_LINE('End ut_teardown'); END
    ut_teardown;

  PROCEDURE ut_test1 IS
  BEGIN
    utAssert.eq( msg_in          => 'TF1: ',
                 check_this_in   => f_erhoehe_gehalt(1000,5),
                 against_this_in => 1050 );
  END ut_test1;
END ut_f_erhoehe_gehalt;
```

Beispiel mit utPLSQL (II)

ORDIX AG

```
set serveroutput on
-- Unittest ut_f_erhoehe_gehalt starten...
BEGIN
  utplsql.run(
    testpackage_in  => 'ut_f_erhoehe_gehalt',
    owner_in        => USER );
END;
/
Start ut_setup
.
>    SSSS   U     U   CCC     CCC   EEEEEEE   SSSS     SSSS
>   S    S  U     U  C   C   C   C  E        S    S   S    S
>  S        U     U C     C C     C E       S        S
>   S       U     U C       C       E        S        S
>    SSSS   U     U C       C       EEEE      SSSS     SSSS
>        S  U     U C       C       E             S        S
>         S U     U C     C C     C E              S        S
>   S    S   U   U   C   C   C   C  E        S    S   S    S
>    SSSS     UUU     CCC     CCC   EEEEEEE   SSSS     SSSS
.
SUCCESS: "ut_f_erhoehe_gehalt"
.
> Individual Test Case Results:
>
SUCCESS - ut_f_erhoehe_gehalt.UT_TEST1: EQ "TF1: " Expected "1050" and got
"1050"
>
> Errors recorded in utPLSQL Error Log:
>
> NONE FOUND
End ut_teardown
```

Beispiel mit utPLSQL (III)

ORDIX AG

```
-- Test Suite definieren und ausfuehren...
BEGIN
  -- Test Suite definieren...
  utsuite.add(name_in => 'SUITE_F_ERHOEHE_GEHALT');

  -- Unittest Packages der Test Suite zuordnen...
  utPackage.add(
    suite_in   => 'SUITE_F_ERHOEHE_GEHALT',
    package_in => 'UT_F_ERHOEHE_GEHALT_1'
  );
  utPackage.add(
    suite_in   => 'SUITE_F_ERHOEHE_GEHALT',
    package_in => 'UT_F_ERHOEHE_GEHALT_2'
  );

  -- Test Suite ausfuehren...
  utPLSQL.runsuite(
    suite_in            => 'SUITE_F_ERHOEHE_GEHALT',
    reset_results_in    => TRUE,
    per_method_setup_in => FALSE
  );
END;
/
```

Besonderheiten Unittest

ORDIX AG

- Keine Abhängigkeiten unter den Unittests:
  - Die Unittests müssen in beliebiger Reihenfolge aufgerufen werden können!

- Werden z.B. Datensätze in bestimmten Tabellen oder externe Dateien für die Durchführung eines Tests benötigt, müssen diese Datensätze oder Dateien durch den Unittest selbst erzeugt werden.

- Condition Compiler Option (compiler directive) für Fehlersimulationen verwenden

WICHTIG!

Die einzelnen Tests sollen jederzeit ohne manuelle vorbereitende Maßnahmen durchgeführt werden können.

Agenda

ORDIX AG

- Überprüfung der fachlichen Korrektheit in PL/SQL
  - PL/SQL-Unittest mit utPLSQL
- Automatische Code-Analyse
  - PL/SQL-Warnungen
  - PL/Scope
  - Auswertung der DD-Views
- Fazit

Agenda

ORDIX AG

- Überprüfung der fachlichen Korrektheit in PL/SQL
  - PL/SQL-Unittest mit utPLSQL
- Automatische Code-Analyse
  - PL/SQL-Warnungen
  - PL/Scope
  - Auswertung der DD-Views
- Fazit

PL/SQL-Compilerwarnungen
Übersicht

ORDIX AG

- PL/SQL-Compilerwarnungen können genutzt werden um:
  - Laufzeitfehler zu vermeiden
  - potenzielle Performance-Probleme zu identifizieren
  - Faktoren zu kennzeichnen, die zu nicht definierten Ergebnissen führen
- PL/SQL-Compilerwarnungen können folgendermaßen aktiviert werden:
  - `PLSQL_WARNINGS` Initialisierungsparameter
  - `DBMS_WARNING` PL/SQL-Package

PLSQL_WARNINGS
Syntax

ORDIX AG

- Syntax von PLSQL_WARNINGS:

```
PLSQL_WARNINGS='value_clause' [, 'value_clause' ] ...

value_clause :=
{ ENABLE | DISABLE | ERROR }:
{ ALL | SEVERE | INFORMATIONAL | PERFORMANCE
  | { integer | (integer [, integer ] ...) }
}
```

- Beispiel:

```
-- Alle Warnungen aktivieren...
ALTER SESSION SET PLSQL_WARNINGS='ENABLE:ALL';

-- Warnungen als Fehler (ERROR) behandeln...
ALTER SESSION SET PLSQL_WARNINGS='ERROR:(5005,5018)';
```

PL/SQL-Compilerwarnungen

Beispiel

ORDIX AG

```
CREATE OR REPLACE PACKAGE pkg1 IS
  FUNCTION f1(p1 IN OUT VARCHAR2) RETURN NUMBER;
END pkg1;
CREATE OR REPLACE PACKAGE BODY pkg1 IS
  PROCEDURE proc1 AS
    CURSOR c1 IS SELECT * FROM ma;
    v1 VARCHAR2(1);
  BEGIN
    SELECT DUMMY INTO v1 FROM dual;
  EXCEPTION
    WHEN OTHERS THEN
      NULL;
  END proc1;

  FUNCTION f1(p1 IN OUT NOCOPY VARCHAR2)
  RETURN NUMBER IS
    v_prov  VARCHAR2(3) := '2';
  BEGIN
    IF p1=1 THEN
      INSERT INTO ma(nr, name, gehalt, prov)
      VALUES(1,'Mueller', 40000, v_prov);
    ELSE
      RETURN 1000;
      proc1;
    END IF;
  END f1;
END pkg1;
```

PLW-07203: Parameter "P1" kann von der Benutzung des Compiler-Hints NOCOPY profitieren

PLW-06006: nicht aufgerufene Prozedur "C1" wird entfernt.

PLW-06009: Prozedur "PROC1" OTHERS Handler endet nicht in RAISE oder RAISE_APPLICATION_ERROR

PLW-05000: Nichtübereinstimmung in NOCOPY-Qualifikation zwischen Spezifikation und Body

PLW-07202: Bind-Typ würde zu einer Konvertierung weg vom Spaltentyp führen

PLW-06002: Auf Code kann nicht zugegriffen werden

PLW-05005: Funktion F1 ergibt keinen Rückgabewert in Zeile 22

(**INFORMATIONAL** **PERFORMANCE** **SEVERE**)

Agenda

ORDIX AG

- Überprüfung der fachlichen Korrektheit in PL/SQL
  - PL/SQL-Unittest mit utPLSQL
- Automatische Code-Analyse
  - PL/SQL-Warnungen
  - PL/Scope
  - Auswertung der DD-Views
- Fazit

PL/Scope

Code-Analyse

ORDIX AG

- Analyse der Identifier in Bezug auf:
  - Deklaration
  - Definition
  - Referenz
  - Aufruf
  - Zuweisung
- Aktivierung über `PLSCOPE_SETTINGS`, z.B.

```
-- Code-Analyse aktivieren...
ALTER SESSION SET PLSCOPE_SETTINGS='IDENTIFIERS:ALL';
```

- Speicherung in SYSAUX-Tablespace
- Auswertungsmöglichkeiten über `user|all|dba_identifiers`

- Suche nach einer Zuweisung:

```
a := 1;   fetch c1 into a;   proc1( a );
```

- Überprüfung der Namenskonventionen von z.B. Variablen oder Prozeduren

- Antworten auf Fragen wie:
  - Welche Variablen wurden deklariert aber nicht verwendet?
  - Welche Variablen werden referenziert aber nicht initialisiert?
  - Welche Variablen werden nach einer Zuweisung nicht verwendet?

PL/Scope

Beispiel

ORDIX AG

```
alter session set PLSCOPE_SETTINGS='identifiers:all';
SQL> create or replace procedure p_test is
  2    a  varchar2(1);
  3  begin
  4    a := 'A';
  5    dbms_output.put_line(a);
  6  end p_test;
  7  /
Prozedur wurde erstellt.

select line, col, name, type, usage
from user_identifiers where object_name='P_TEST' order by line;

 LINE   COL NAME       TYPE                USAGE
----- ----- ---------- ------------------- -----------
    1    11 P_TEST     PROCEDURE           DEFINITION
    1    11 P_TEST     PROCEDURE           DECLARATION
    2     3 A          VARIABLE            DECLARATION
    2     6 VARCHAR2   CHARACTER DATATYPE  REFERENCE
    4     3 A          VARIABLE            ASSIGNMENT
    5    24 A          VARIABLE            REFERENCE
```

- Object Type-Variablen:
  - Zuweisung beim Konstruktoraufruf
  - Keine Zuweisung der Attribute beim Konstruktoraufruf
- Record-Variablen:
  - Keine Zuweisung
  - Deklarationen der Attribute bei `TYPE … IS RECORD`
  - Keine Deklarationen der Attribute bei `%ROWTYPE`
  - Keine Zuweisung der Attribute bei `BULK COLLECT`
- …

Agenda

ORDIX AG

- Überprüfung der fachlichen Korrektheit in PL/SQL
  - PL/SQL-Unittest mit utPLSQL
- Automatische Code-Analyse
  - PL/SQL-Warnungen
  - PL/Scope
  - Auswertung der DD-Views
- Fazit

Auswertung der DD-Views

ORDIX AG

- Anzahl der Code-Zeilen pro PL/SQL-Objekt (`ALL_SOURCE`)
- Anzahl der Parameter pro PL/SQL-Objekt (`ALL_ARGUMENTS`)
- Funktionen mit OUT-Parametern (`ALL_ARGUMENTS`)
- Gegenseitige Abhängigkeiten (`ALL_DEPENDENCIES`)
- Tabelle ohne PK (`ALL_CONSTRAINTS`)
- Fremdschlüsselspalten ohne Index (`ALL_CONS_COLUMNS, …`)
- …

Agenda

ORDIX AG

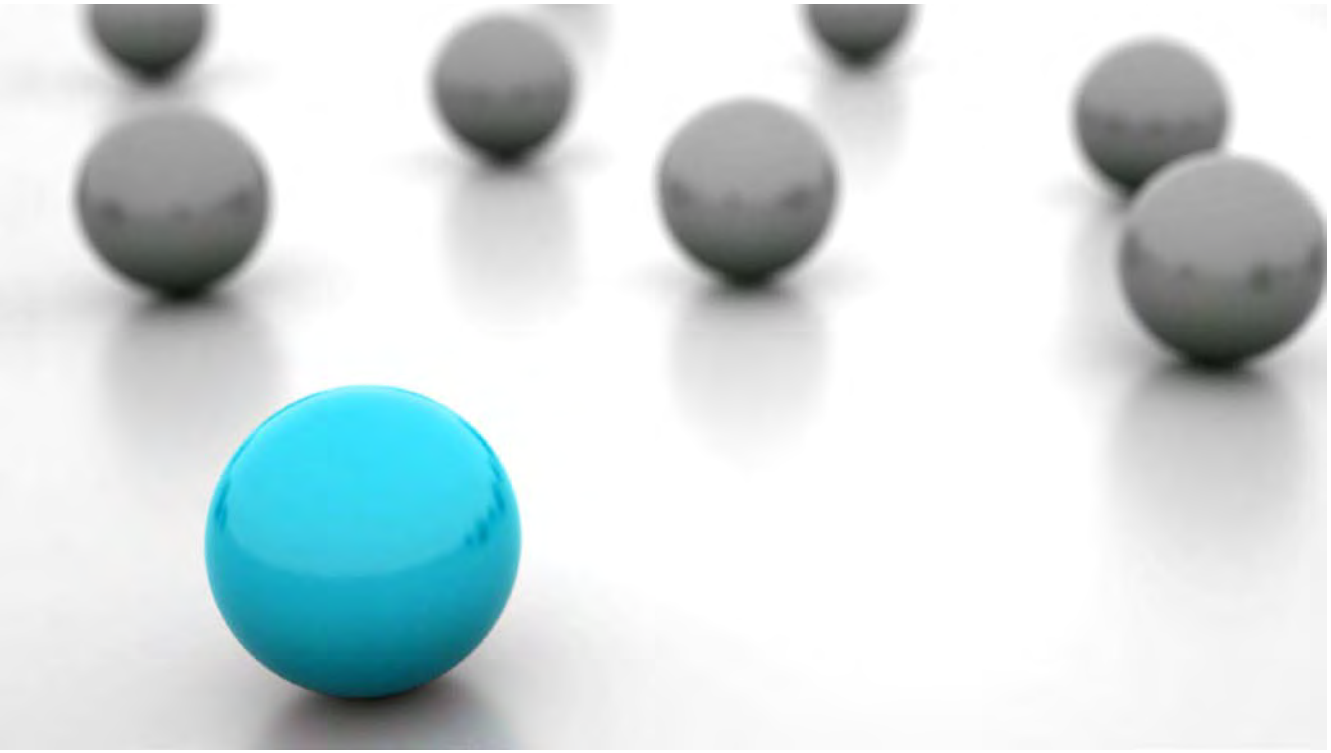
- Überprüfung der fachlichen Korrektheit in PL/SQL
  - PL/SQL-Unittest mit utPLSQL
- Automatische Code-Analyse
  - PL/SQL-Warnungen
  - PL/Scope
  - Auswertung der DD-Views
- Fazit

Fazit

ORDIX AG

- Überprüfung der fachlichen Korrektheit von PL/SQL-Programmen kann automatisiert werden.
- Mit einer Code-Analyse kann die Qualität der Software deutlich verbessert werden.

ORDIX AG



Zentrale Paderborn
Westernmauer 12 - 16
33098 Paderborn
Tel.: 05251 1063-0

Seminarzentrum Wiesbaden
Kreuzberger Ring 13
65205 Wiesbaden
Tel.: 0611 77840-00

Zentrales Fax:
0180 1 67349 0
0180 1 ORDIX 0

Weitere Geschäftsstellen
in Köln, Münster und Neu-Ulm

E-Mail: info@ordix.de
Internet: http://www.ordix.de

Vielen Dank für Ihre Aufmerksamkeit!