Migration of Oracle Forms applications

Kelly Garcés
Rubby Casallas
Edgar Sandoval
Camilo Alvarez

Cristo Rodríguez
Alejandro Salamanca
Fabian Melo
Sandra Pinto
Juan Soto

Jordi Cabot

CSw Research Group
Universidad de los Andes
Bogotá D.C., Colombia

Asesoftware SAS.
Bogotá D.C., Colombia

Universitat Oberta de Cataluyna
Barcelona, Spain

Oracle Forms Modernization Project

Case Study: Asesoftware (est. 1991) [1]

Business: develop & maintain Oracle Forms systems

Challenge: moving from Oracle Forms to modern technologies

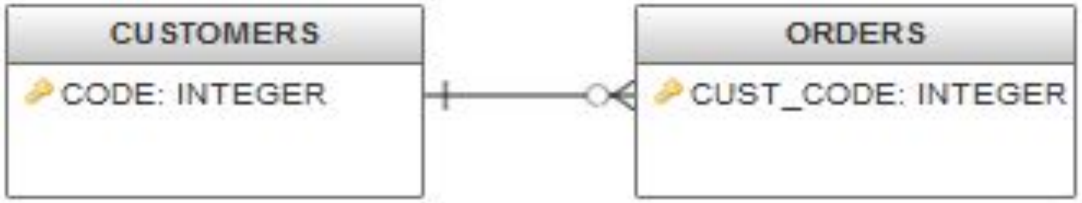
- Lack of design information
- Little visibility of what is expected from the modernization that results on (over)underestimation of time and budget
- It's a time consuming and error prone task

*[1] www.**asesoftware**.com/*

What is Oracle Forms?

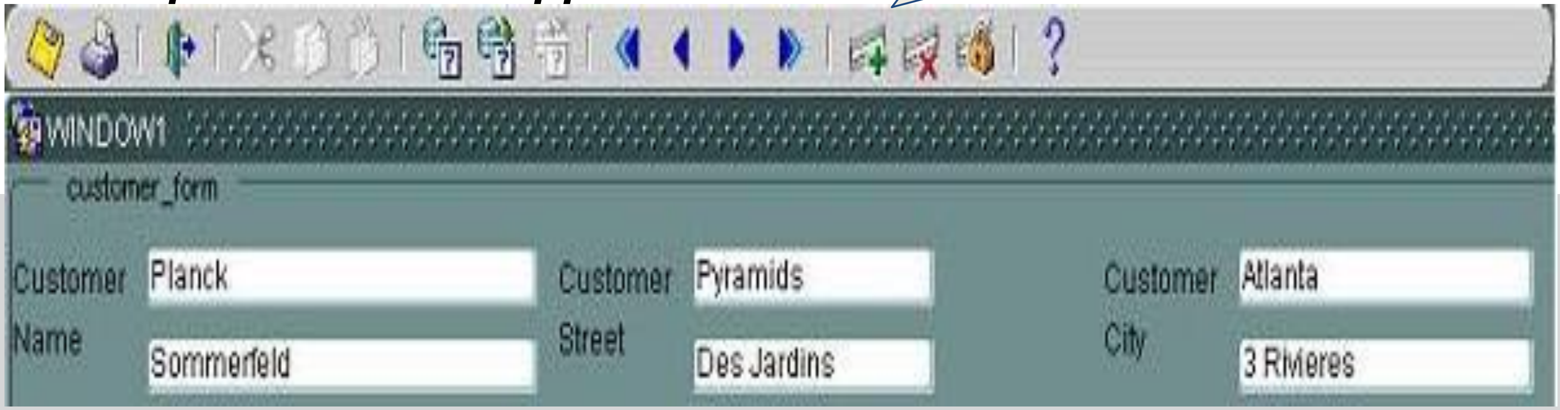
A programming language and development tool for creating desktop applications that interact with Oracle databases

Database tables



CRUD functionality

Desktop Oracle Form application



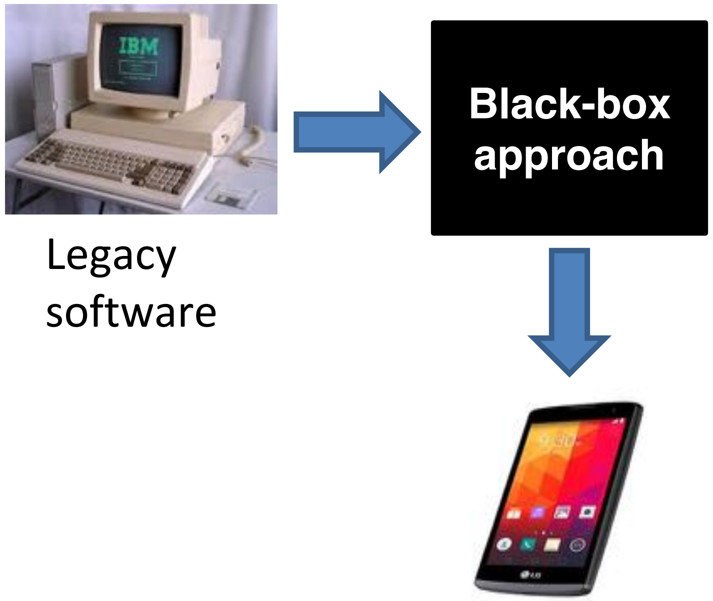
customer_form

Customer	Planck	Customer	Pyramids	Customer	Atlanta
Name	Sommerfeld	Street	Des Jardins	City	3 Rivieres

Project scope

- Master and master/detail forms
  - The basic functionality
    - the graphical interface (except the layout)
    - the CRUD logic
  - the PLSQL code embedded into triggers
- The target technology is JEE

Drawbacks of existing migration tools

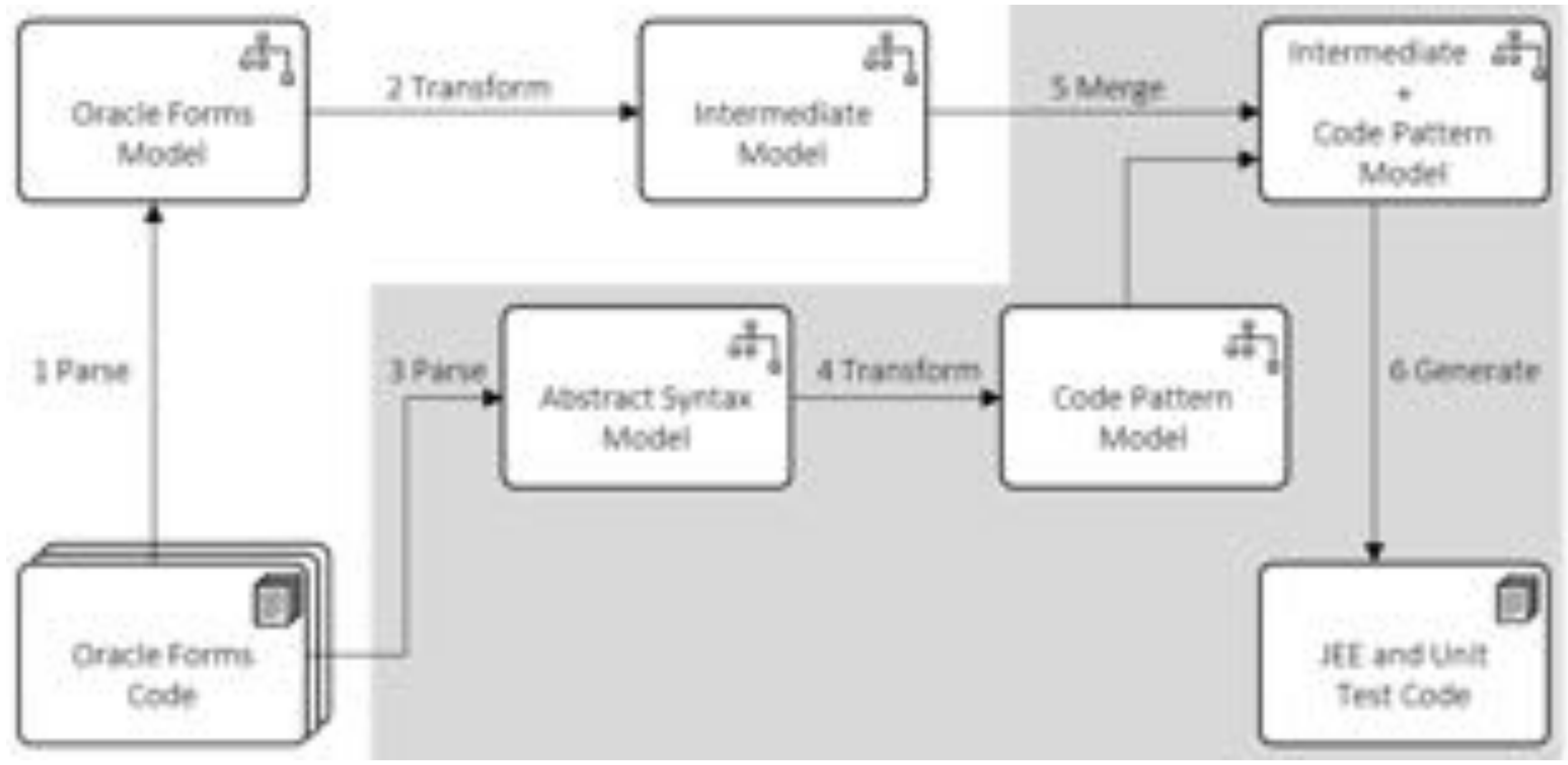


Legacy software

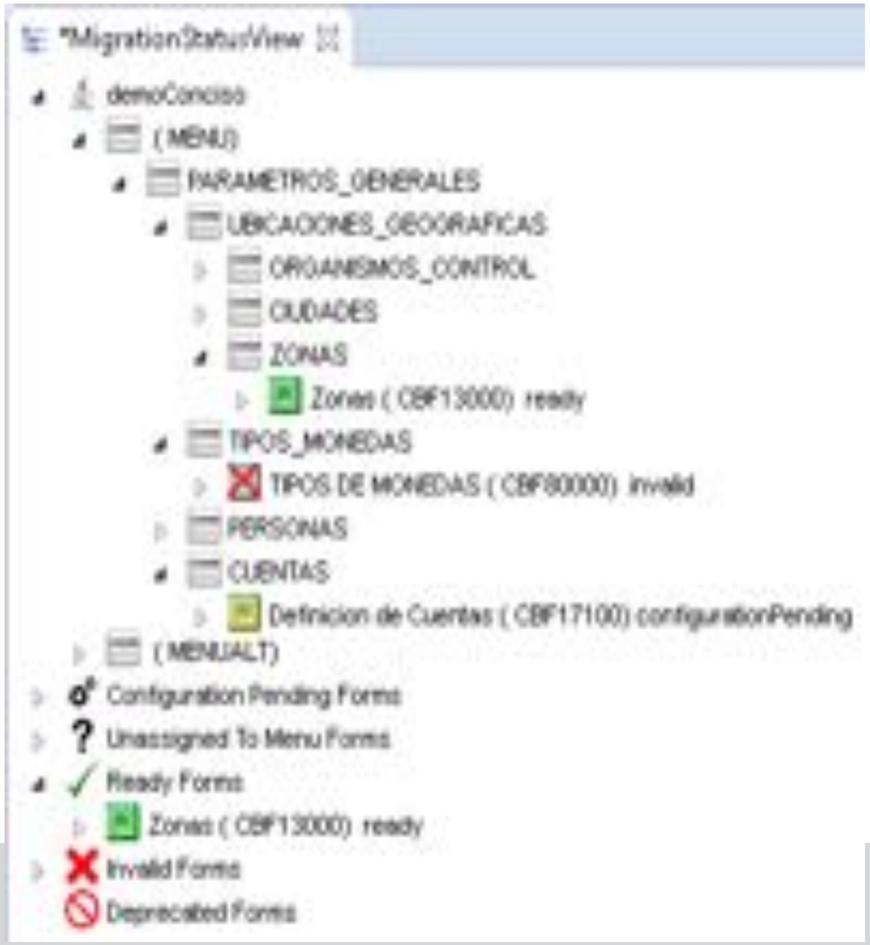
Target software does not operate as expected

1. Lack of information
2. Difficult to maintain
3. Not user friendly
4. Unknown transformation progress
5. Costly approaches

White-box transformation process



Configuring architecture



```
*MigrationStatusView
  demoConciso
    (MENU)
      PARAMETROS_GENERALES
        UBICACIONES_GEOGRAFICAS
          ORGANISMOS_CONTROL
          CIUDADES
          ZONAS
            Zonas ( CBF13000) ready
        TIPOS_MONEDAS
          TIPOS DE MONEDAS ( CBF80000) invalid
        PERSONAS
        CUENTAS
          Definicion de Cuentas ( CBF17100) configurationPending
    (MENUALT)
  Configuration Pending Forms
  Unassigned To Menu Forms
  Ready Forms
    Zonas ( CBF13000) ready
  Invalid Forms
  Deprecated Forms
```

Personalize the features of target architecture through an editor.

- Menu structure definition
  Drawback 3: (Usability)
- Screen classification
  - Configuration pending
    Drawback 1: Data access
  - Unassigned
  - Deprecated

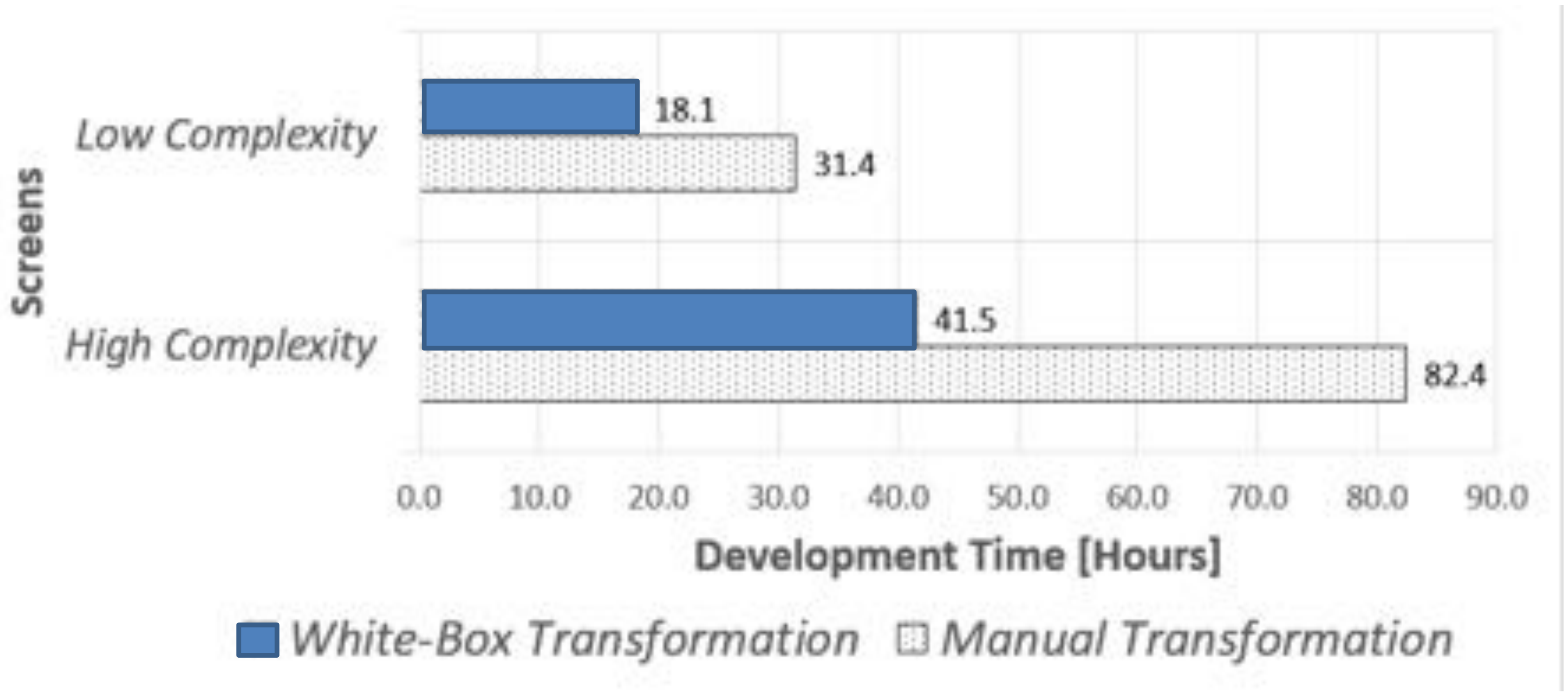
  Drawback 2: Maintainability
  - Ready
    - Drawback 4: configuration process

Evaluation

Pilot study for the basic functionality

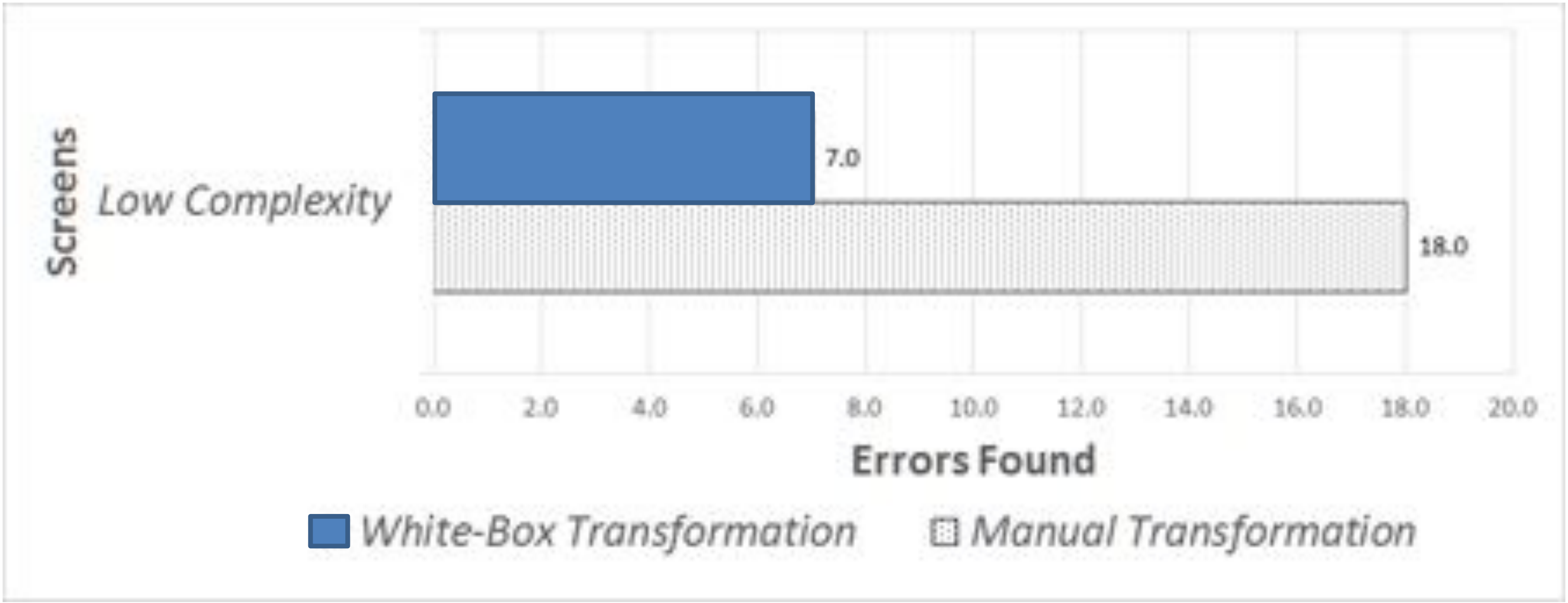
- Purpose: To compare time savings and quality of WBA with these of a manual transformation
  - 4 Asesoftware developers.
  - 2 Teams (1 senior, 1 junior).
  - Insurance application.
    - 2 Forms of different size were chosen (low and high complexity).
  - Task tracking and survey.

Results



"Graphical editor eases the architecture configuration"
"The tool generates a lot of code what result in less development effort for us "

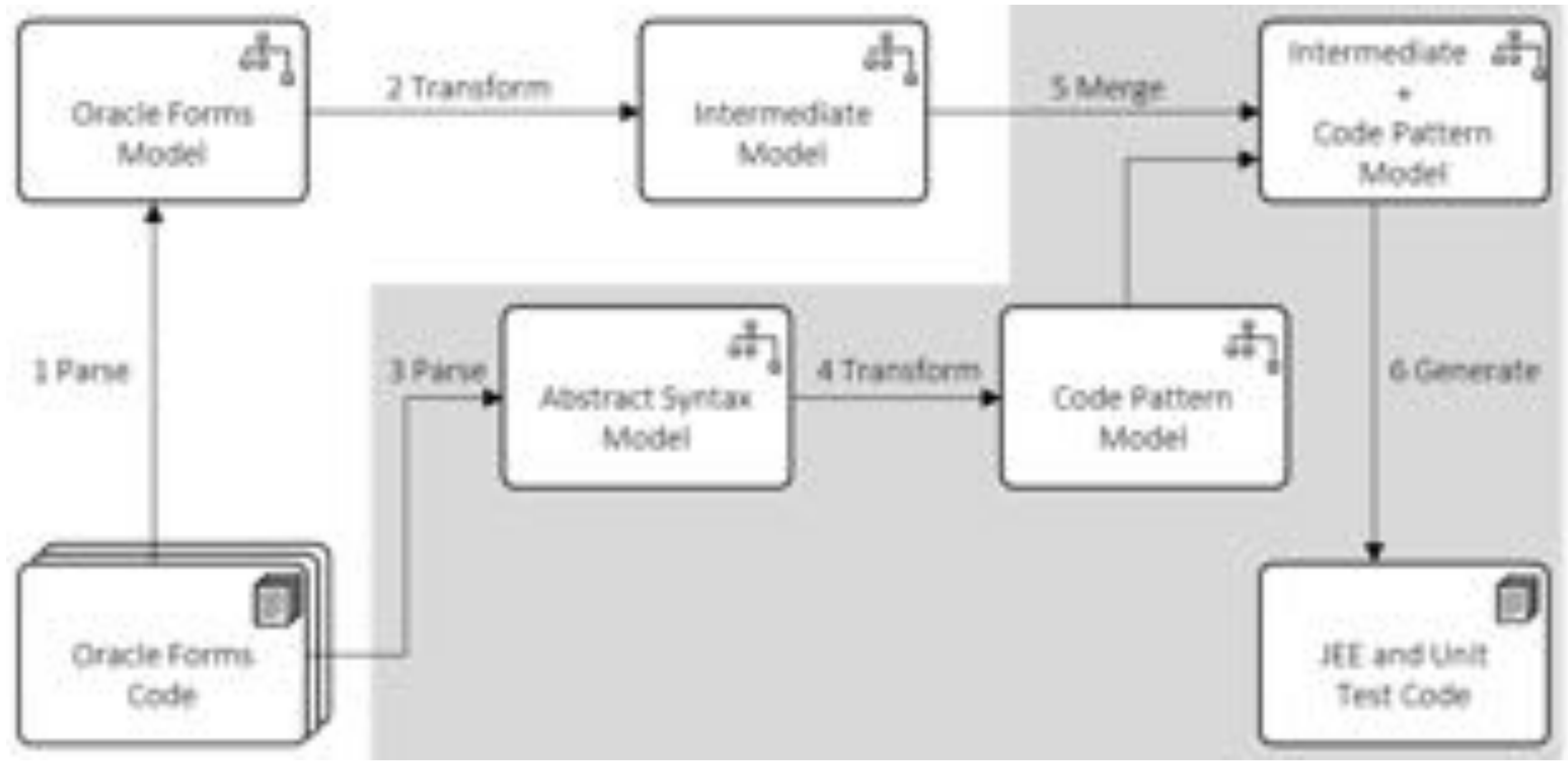
Results



Errors found in the low complexity form for each method

The quality of code is significantly higher when following the white-box transformation than the manual transformation (environ 61%)

White-box transformation process



Code Patterns Catalog

- **Field validation**
- **Field population**
- **Model constraints**
- **Miscellaneous**

20 Patterns

UNQ_VAL, Unique key validation

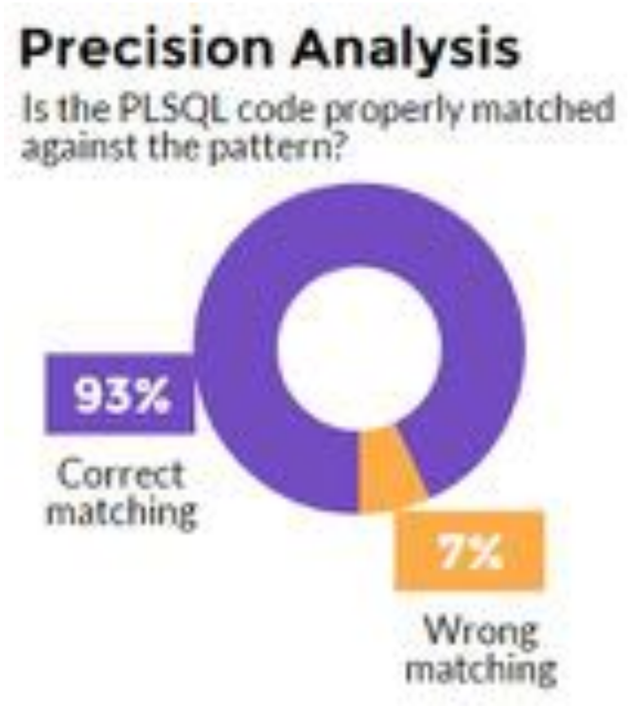
```
SELECT count(1) INTO localVar
FROM tableName
WHERE col1 = fieldA
  AND col2 = fieldB
  AND ....

IF localVar > 0
  SHOW_MESSAGE(msg)
  RAISE_ERROR
```

Evaluation *(PLSQL Migration)*

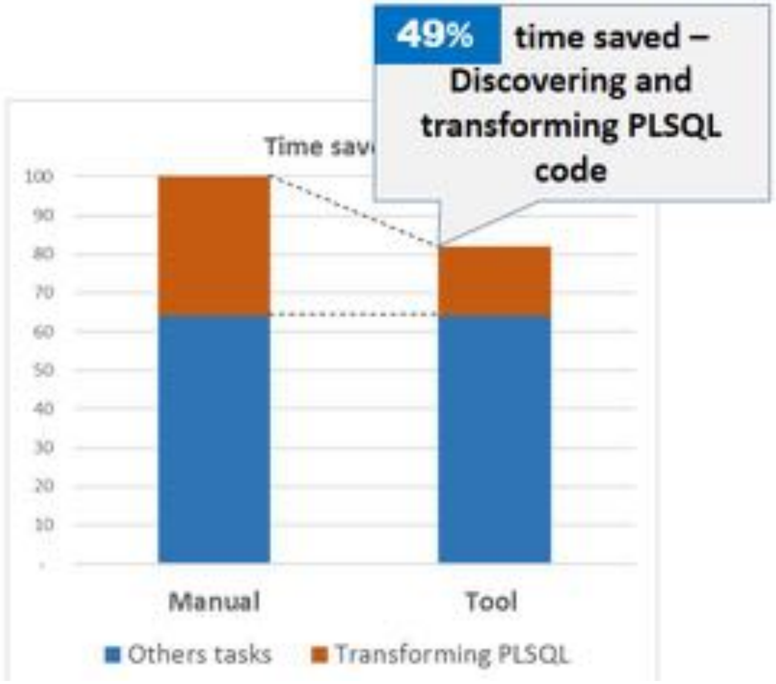
Pilot study for the PLSQL migration

- Purpose: To validate the correctness of the discovered patterns
  - 4 developers
  - 72 code segments reviewed by developers against the tool outcomes
  - 4 applications (Conciliation, Insurance, Bank transfer applications, Treasury)

Results *(PLSQL Migration)*



Time saved



Universidad de los Andes

Lessons Learned

- The success of MDE adoption is significantly affected by factors such as training and commitment to the project.

- Some patterns reflect the application of organizational coding conventions.

- Front code often implement basic data validation (e.g., ranges) and user interface logic.

Conclusions

The value added of our approach relies on

1. Taking architectural decisions at model level
2. Migrating not only the CRUD functionality but also the PLSQL code
3. Generating a clear and understandable target code
4. Applying the best practices of the target technology
5. Decoupling reverse from forward engineering

Developers are more productive when following the white-box modernization than the manual modernization (environ 40%)

This approach has been instrumented in an innovative product called SMoT