Oracle Forms Modernization

Better Tooling Support and Co-creation
to Drive the most Business Value

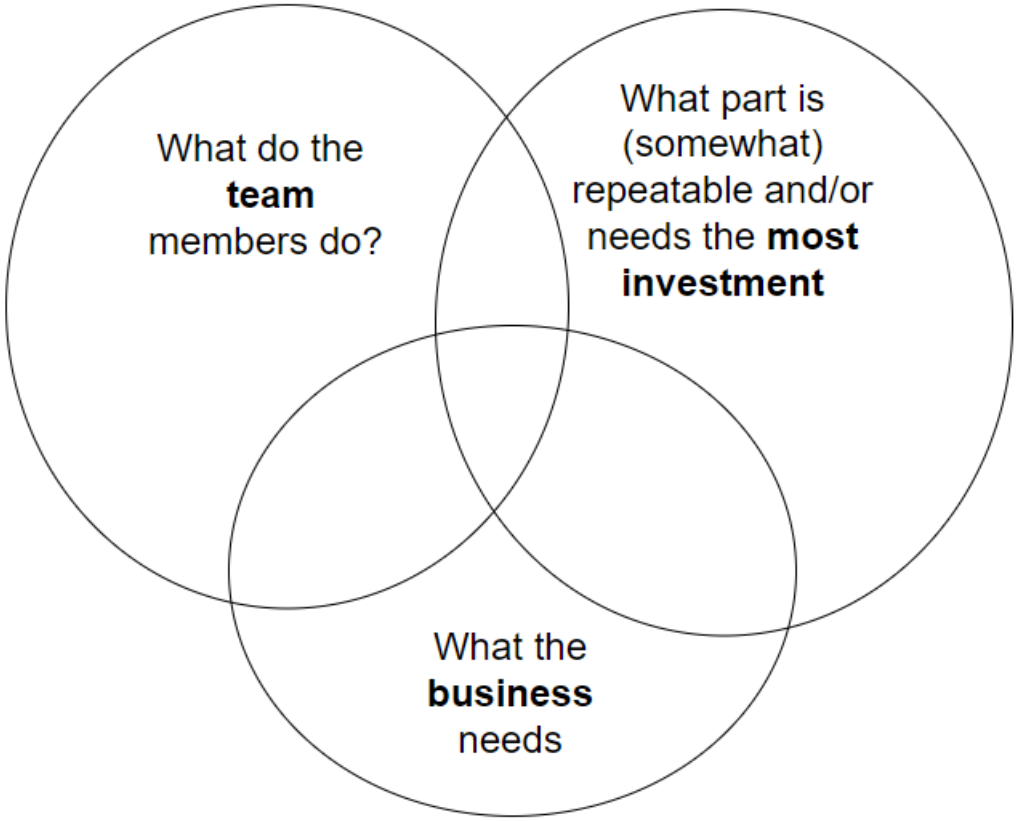
History/Context

- Oracle forms allow you to rapidly create applications by using your oracle db skills.
- Create forms based on underlying tables, drag-and-drops
- Create processing logic by in PL/SQL triggers - called upon specific action in the form. Example.
- Web applications are the norm now and Oracle application express (sometimes referred to as APEX) is the new preferred tool for creating applications on top of Oracle (using just SQL and PL/SQL skills).

History/Context (Summary)

- Oracle Forms Support (Ends 2026/2027)
- 4G data model driven low code system - Oracle Forms
- Replace with Modern Web Development Frameworks

Understanding the Problem First Before Solutioning



- **Interviews**
- **80-20 Rules**
- **Delivers the most Business Values**
  (Cost, Benefit)
  Most Vital Priority
  Most Vital Functions
- Funding

Product Thinking
Interviews, user journey map, workflow
High value need coded in extraction (Parser) and result generation process

Understanding the Problem Space

Time boxed Interviews

BA (recent)

PL/SQL Developer (recent)

Front End Developer (past chat)

Sample Interview Questions (PL/SQL DEV)

Workflow

1. Can you walk me through your typical process for modernizing an Oracle Form?
2. What tools do you currently use for this process?
3. How do you open and analyze Oracle Forms in Oracle Form Builder?
4. What steps do you take to analyze the underlying XML file of a form?
5. How do you write replacement packages based on your analysis?

Identifying Pain Points

Workflow

1. What are the most time-consuming tasks in your workflow?
2. What challenges do you face when analyzing Oracle Forms?
3. Are there any repetitive tasks that you find particularly tedious?
4. What kind of errors or issues do you frequently encounter?
5. How do you handle large and complex forms?

Understanding Gains and Aspirations

Workflow

1. What are the most time-consuming tasks in your workflow?
2. What challenges do you face when analyzing Oracle Forms?
3. Are there any repetitive tasks that you find particularly tedious?
4. What kind of errors or issues do you frequently encounter?
5. How do you handle large and complex forms?

Co-creation not Replacement

1. Use tools (Open Source or Paid)
2. (Semi)automate repeatable, predictable keeping in mind ROI and 80-20 rules
3. Timebox/resource box – “Constraints make us creative.”
4. Existing AI tools : proper prompt engineering, context, few shot learning

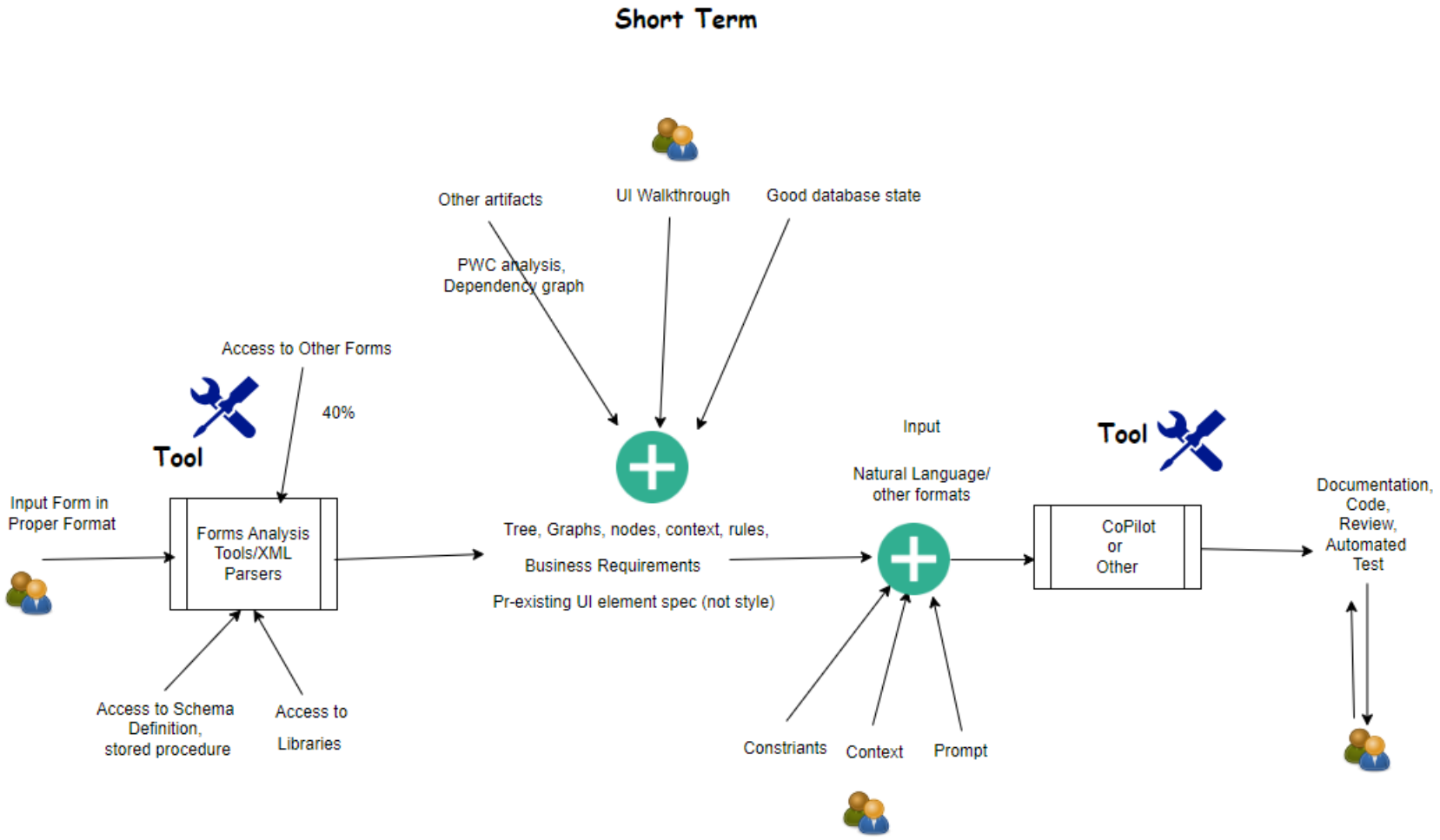
There are tools like: https://github.com/cristianoliveira/FormsOracleAnalyzerLib/tree/master (FMB Analyzer: Has to be adjusted)

BeautifulSoup, XML Parsers – to work on the generated XML file

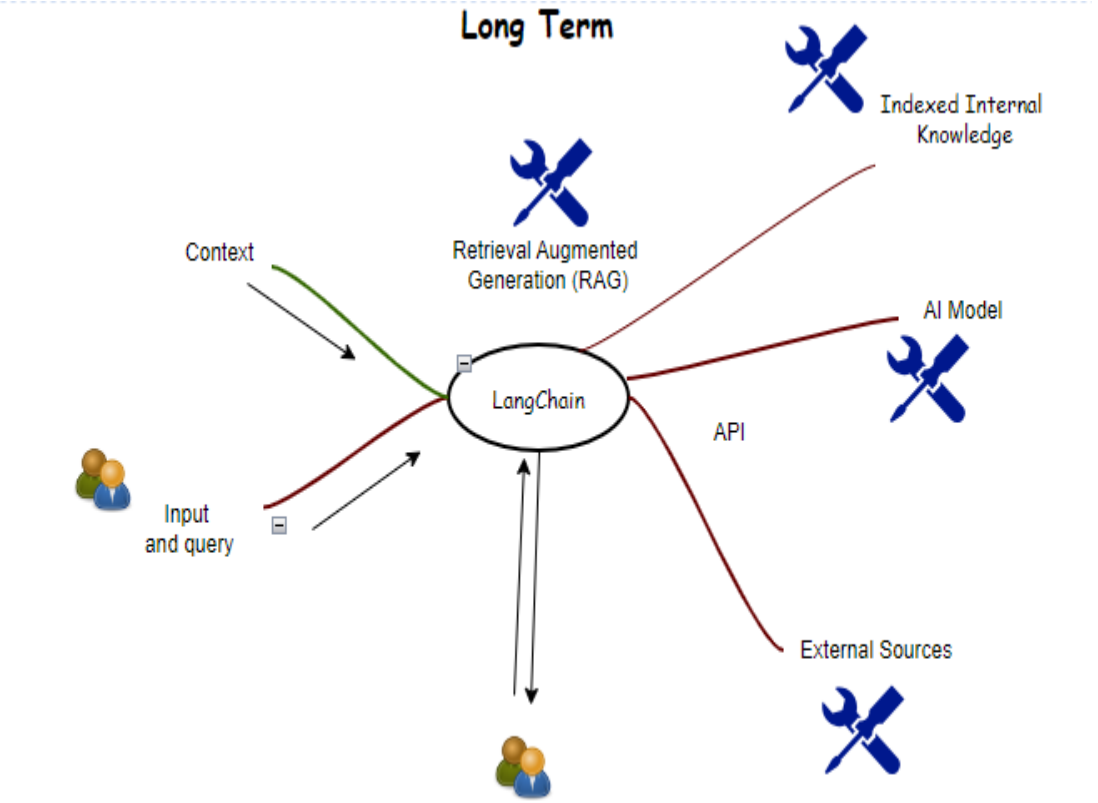
Areas of process improvement

1. More people with access to the tools.
2. Automate the repeatable , predictable keeping in mind ROI and 80-20 rules
3. Timebox/resource box – “Constraints make us creative.”
4. Existing AI tools : proper prompt engineering, context, few shot learning
5. BA – 60% UI walkthrough, **40% tools**
6. PL/SQL DEV – mostly BA work, tools: pre-post queries, complex ones spread across multiple files
7. Front end – Ticket is not in ready state (UX, requirements, endpoint eat significant amount of the time)

Human and Tools Co-creation
Short Term Vision



Human and Tools Co-creation Long Term Vision



Extract from Form Analyzer

- Removed

Documentation (AI)

- Removed

Example Usage (AI)

- Removed

Best Practices for Documentation (AI suggestions: PL/SQL code)

- Documenting PL/SQL code effectively is crucial for maintaining readability, ease of understanding, and facilitating future modifications. Here are some best practices for documenting PL/SQL code:

1. **Header Comments**:
    1. Include a header comment at the beginning of each PL/SQL block (procedure, function, package, etc.) that provides an overview of the code.
    2. Mention the purpose, author, date of creation, and any modifications with dates and descriptions.
2. **Parameter Descriptions**:
    1. Clearly describe each parameter, including its purpose, data type, and any constraints or special considerations.
3. **Inline Comments**:
    1. Use inline comments to explain complex logic or important sections of the code.
    2. Keep comments concise and relevant.
4. **Section Headers**:
    1. Use comments to divide the code into logical sections, making it easier to navigate.
5. **Consistent Formatting**:
    1. Follow a consistent coding style and formatting guidelines to enhance readability.
    2. Use proper indentation and spacing.
6. **Error Handling**:
    1. Document any error handling mechanisms, including the types of exceptions that might be raised and how they are handled.
7. **Version Control**:
    1. Maintain version control comments to track changes over time.
8. **Examples and Usage**:
    1. Provide examples of how to call the procedure or function, especially if it involves complex parameters.

- **Example of Documenting a PL/SQL Function**

Sample Code Skeleton (AI generated)

- Removed

Sample Code Skeleton (AI generated)

- Removed

Opportunities

Streaming Parser (XML or FMB). FMB example -

Documentation not doing it

Automated test and coverage, code review, code generation from comments

Parser – tool

Type of UI elements not style

Demo transcript

Voice to text – faster

Code duplicate removal

Good prompt and context to get good output from AI

User Manual

1 sentence –> user story, acceptance criteria, user signing

Estimation

UI Walkthrough Robot Recorder (not easy for Zone B hosted Oracle Forms)

Risk

Appendix

Not automation but huge cost savers

https://www.orcl-toolbox.com/purchase

https://github.com/cristianoliveira/FormsOracleAnalyzerLib/tree/master (FMB Analyzer: has to be adjusted)

Not automation but huge cost savers

Three of the points that came out of our conversation with BA where there would be huge savings for client is -

1) Demo (like what business is doing on UAT) from the business at the initiation of the project would be a huge time saver.

We would not miss some critical and edge use cases. And we be more confident about coverage. **Can we make it non-negotiable going forward?**

2) Not having data pre-condition set/available on non production system is a huge pain point when the BA tries to gather requirement by playing with the App. Can we make it a pre-requisite before a project start?

3) Frontend developers have to talk to BA, UX, and API person to gather all the specs to do their work. Those should be part of a JIRA task for developer but that is not the case. A ticket should not be considered ready for development until the ticket have those info. <--- **internal process improvement.** (Multiple frontend devs stated the same.)

BA (1 hour interview/user journey mapping to give best help with most ROI)

- 60% Application walkthrough (from SME or business and translating that to natural language document). Using a tool to translate walkthrough into documentation would be harder because of where the Oracle Form is hosted. Some documentation is possible from meeting transcript.
- 40% Forms builder (tool usage to look at POST query, trigger).
- Pre-existing document are obsolete.
- All materials for new employee training not up to date (so we cannot leverage it; how true is it?. Challenge the assumption)

PL/SQL Developer (1 hour interview/user journey mapping to give best help with most ROI)

- Mostly leverages BA's work to write code
- For complex cases she uses FORMS Tools to look at Post query, and global search on multiple downloaded FMB, RDF files to create global concept map in her head (She has the Forms tool already)
- She has access to wider database schema's