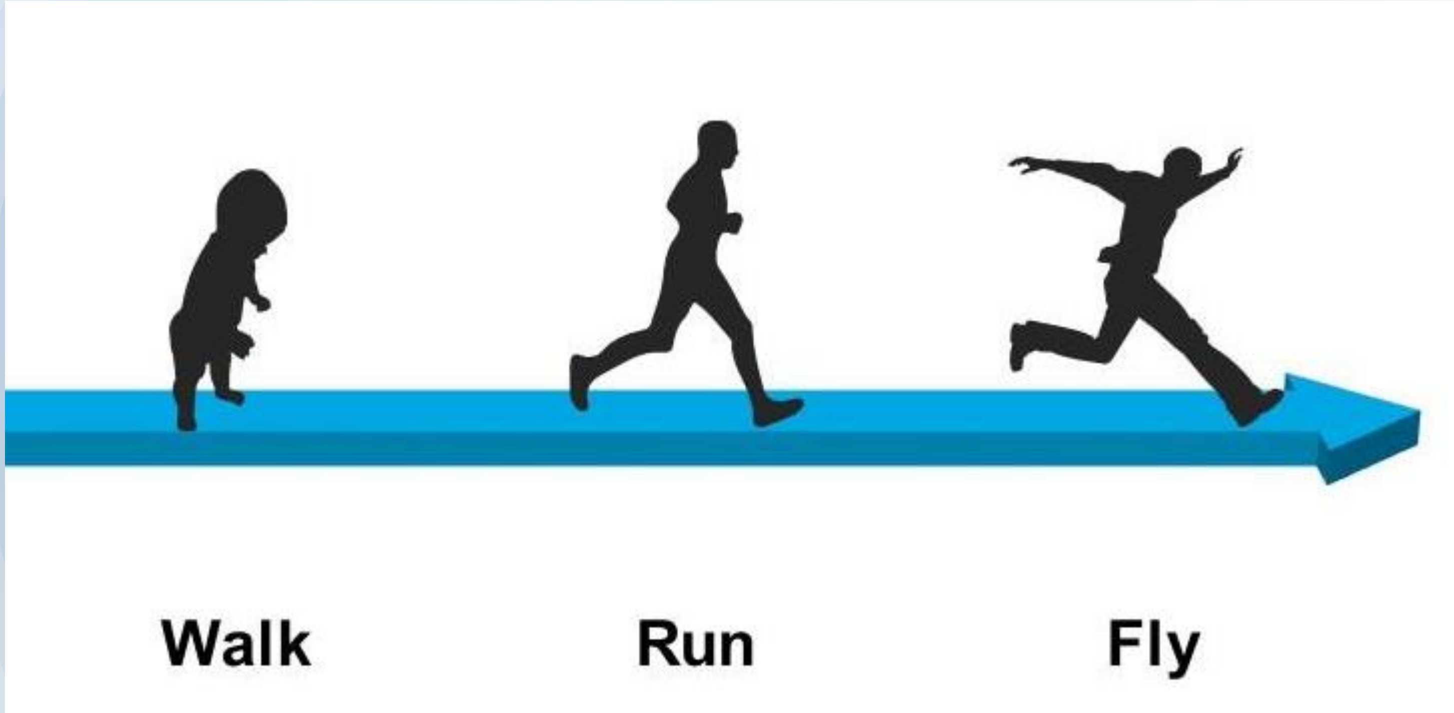
Test Driven Development

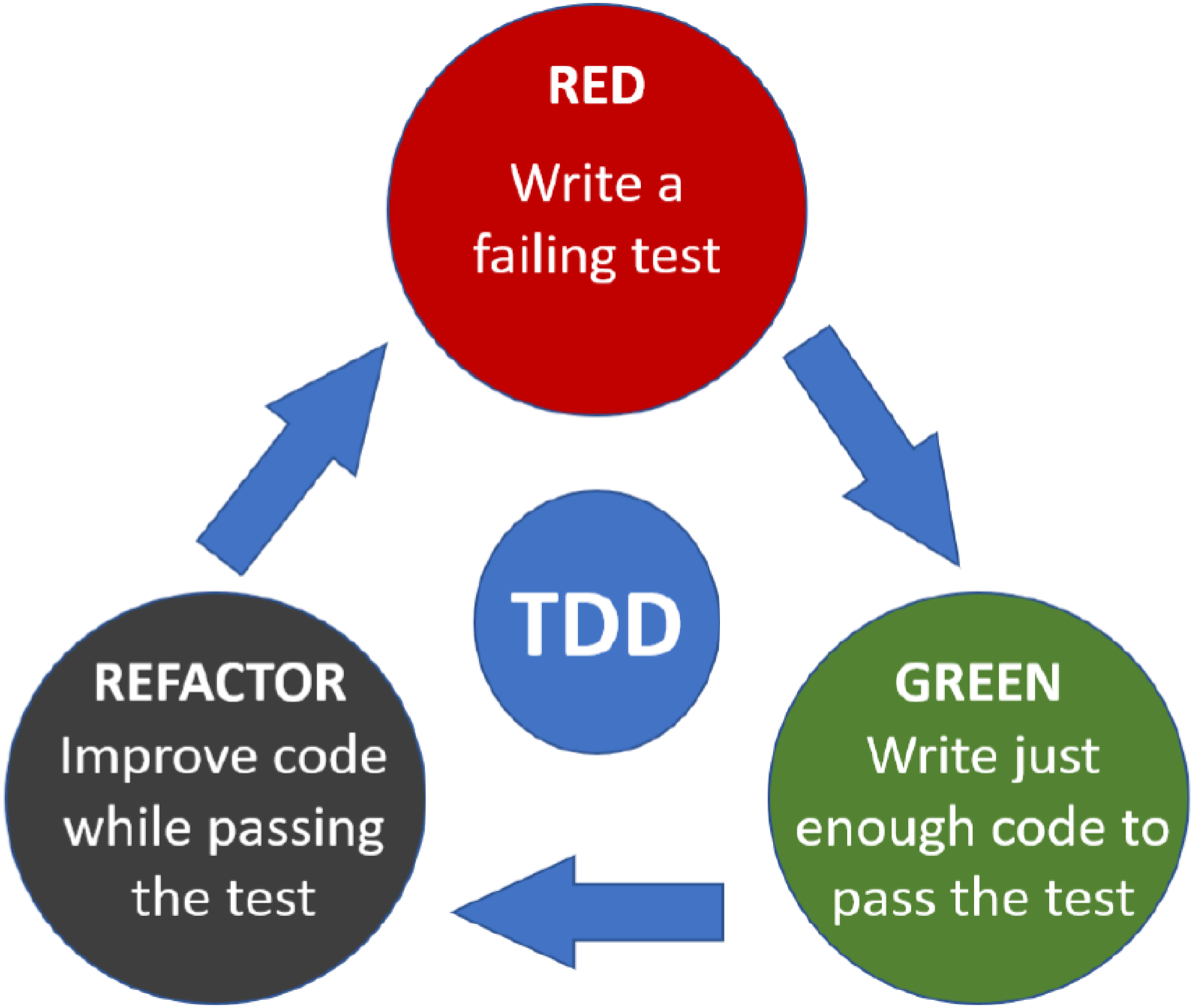
Connascence and Test Doubles

Rune Holen / Bouvet


Walk
Run
Fly

Walking:

- Write red test case
- Code until green
- Refactor



RED
Write a failing test
GREEN
Write just enough code to pass the test
REFACTOR
Improve code while passing the test
TDD

Code smells, like:

- Primitive obsession
- Large class/method
- Feature Envy

SOLID principles:

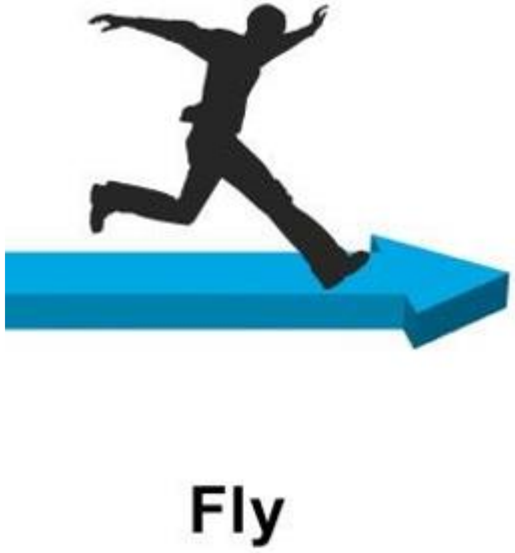
Single Responsibility

Open-Closed

Liskov substitution

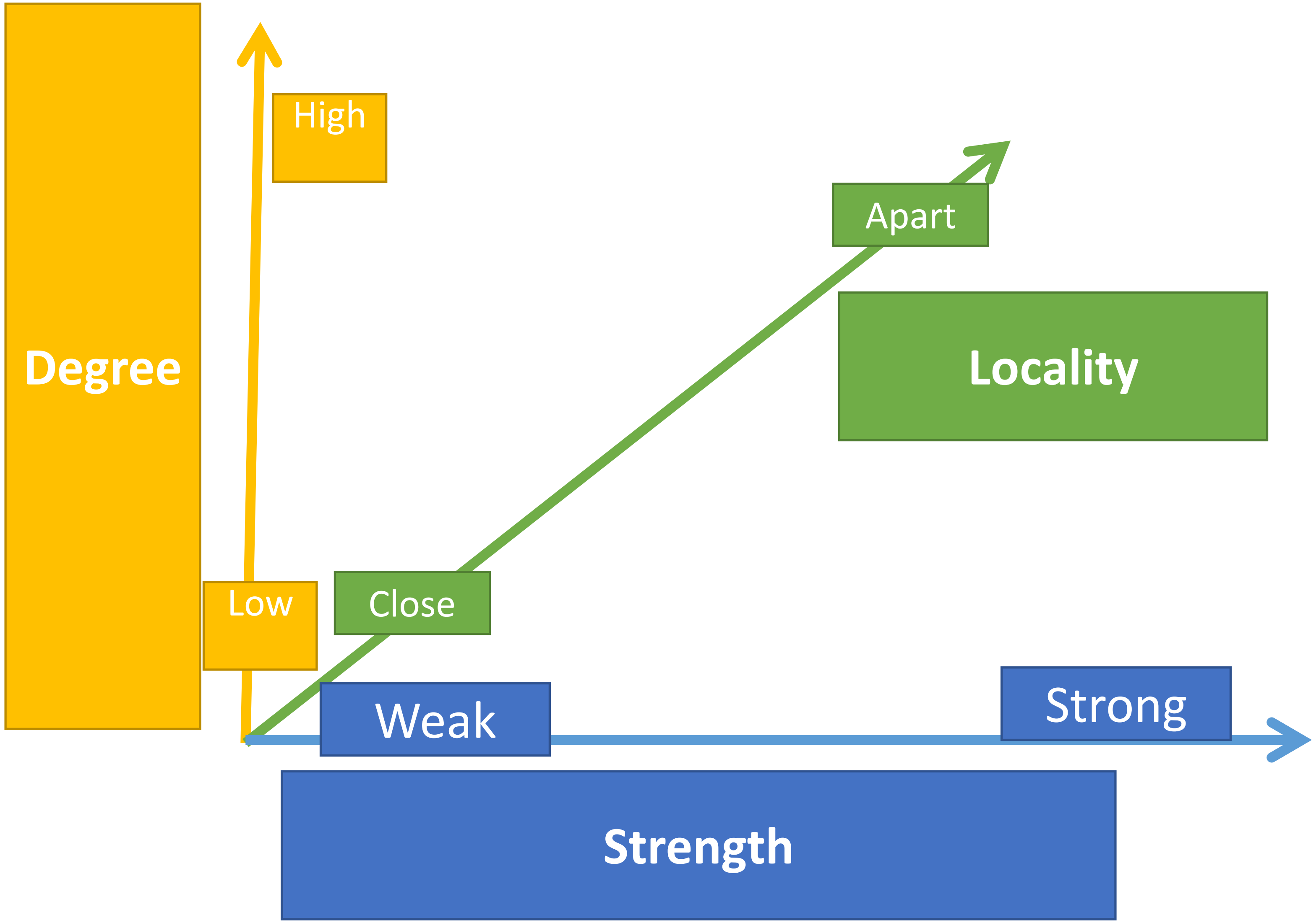
Interface segregation

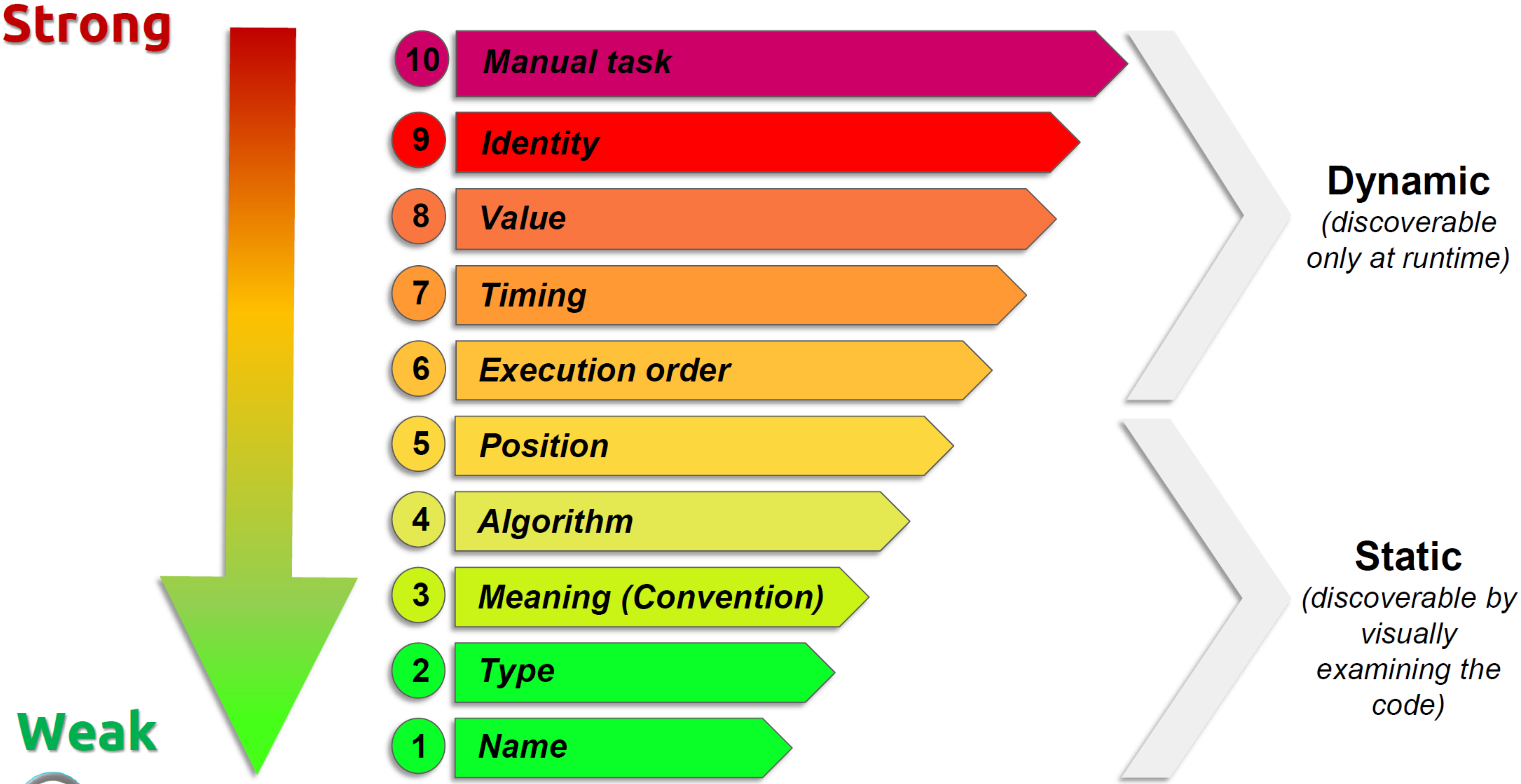
Dependency Inversion



- Connascence
- Test Doubles
- Outside-in Mindset (onion)

Connascence:





Strong

10 *Manual task*

9 *Identity*

8 *Value*

7 *Timing*

6 *Execution order*

5 *Position*

4 *Algorithm*

3 *Meaning (Convention)*

2 *Type*

1 *Name*

Weak

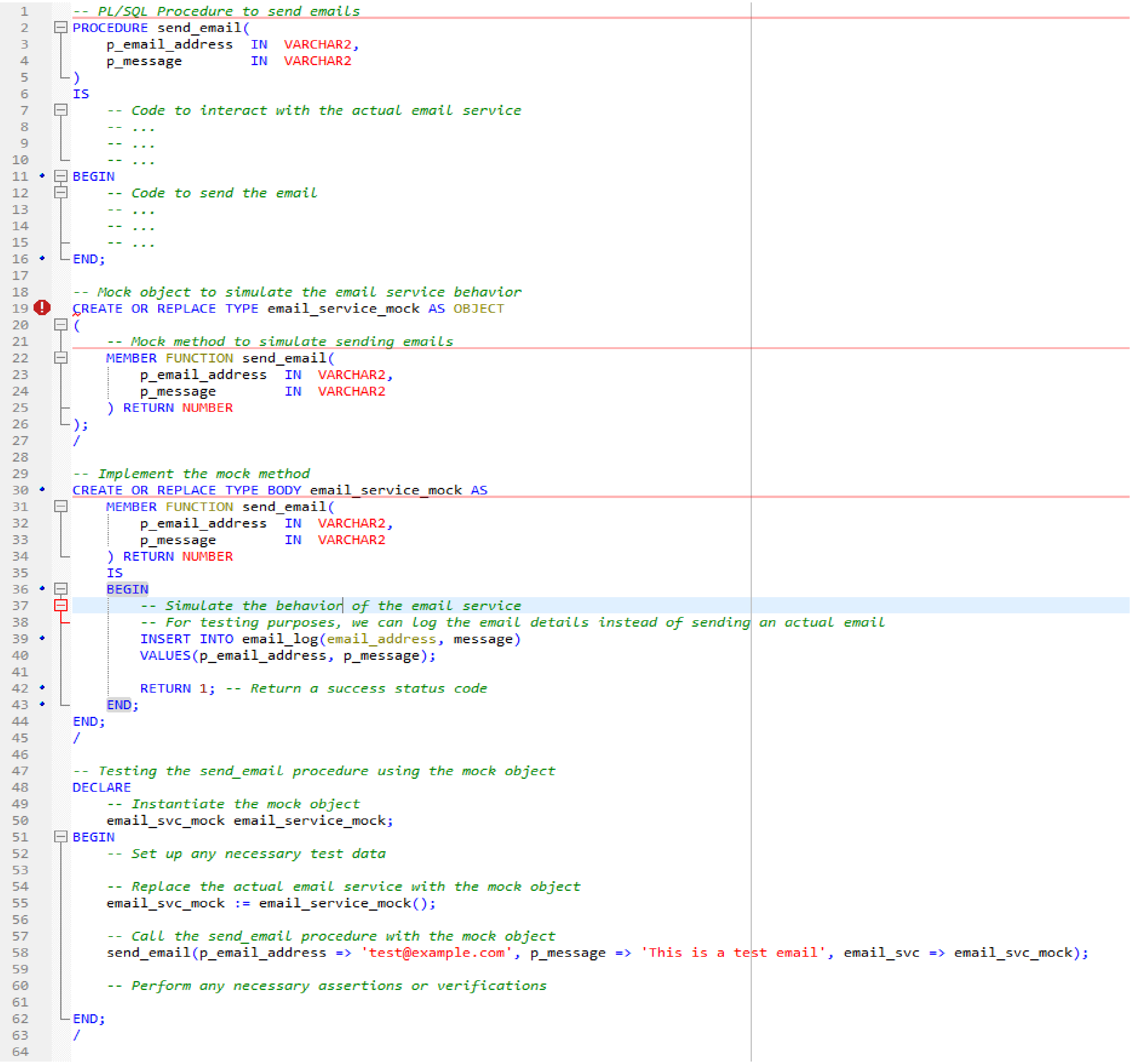
Dynamic
(discoverable only at runtime)

Static
(discoverable by visually examining the code)

Test Doubles

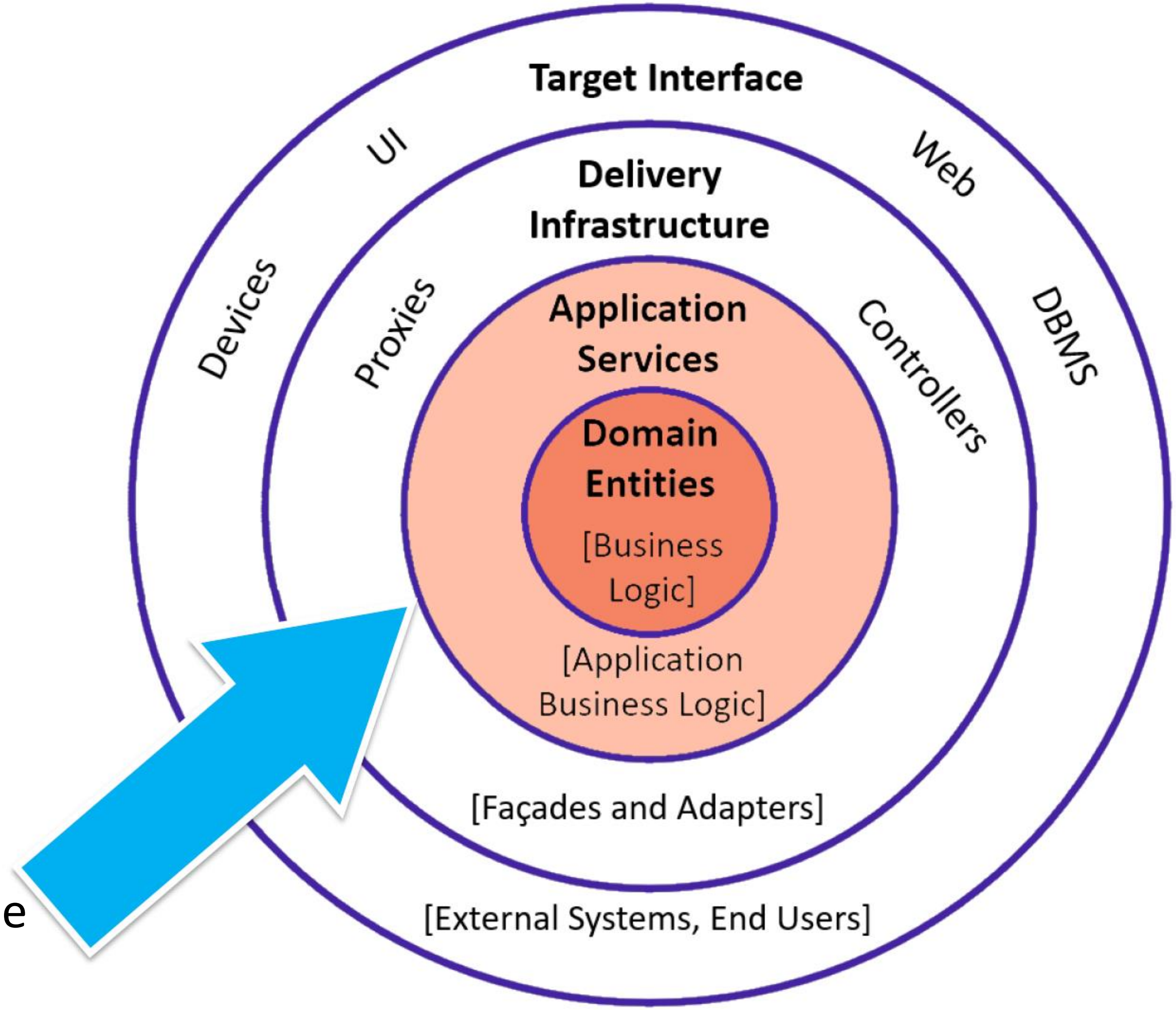
I asked ChatGPT to generate a test double example for PL/SQL

The following example shows a useful framework for a mocked version of send_email(), which logs the email instead of sending via an email service.



```
-- PL/SQL Procedure to send emails
PROCEDURE send_email(
    p_email_address  IN  VARCHAR2,
    p_message        IN  VARCHAR2
)
IS
    -- Code to interact with the actual email service
    -- ...
    -- ...
    -- ...
BEGIN
    -- Code to send the email
    -- ...
    -- ...
    -- ...
END;

-- Mock object to simulate the email service behavior
CREATE OR REPLACE TYPE email_service_mock AS OBJECT
(
    -- Mock method to simulate sending emails
    MEMBER FUNCTION send_email(
        p_email_address  IN  VARCHAR2,
        p_message        IN  VARCHAR2
    ) RETURN NUMBER
);
/

-- Implement the mock method
CREATE OR REPLACE TYPE BODY email_service_mock AS
    MEMBER FUNCTION send_email(
        p_email_address  IN  VARCHAR2,
        p_message        IN  VARCHAR2
    ) RETURN NUMBER
    IS
    BEGIN
        -- Simulate the behavior of the email service
        -- For testing purposes, we can log the email details instead of sending an actual email
        INSERT INTO email_log(email_address, message)
        VALUES(p_email_address, p_message);

        RETURN 1; -- Return a success status code
    END;
END;
/

-- Testing the send_email procedure using the mock object
DECLARE
    -- Instantiate the mock object
    email_svc_mock email_service_mock;
BEGIN
    -- Set up any necessary test data

    -- Replace the actual email service with the mock object
    email_svc_mock := email_service_mock();

    -- Call the send_email procedure with the mock object
    send_email(p_email_address => 'test@example.com', p_message => 'This is a test email', email_svc => email_svc_mock);

    -- Perform any necessary assertions or verifications

END;
/
```

Onion Architecture

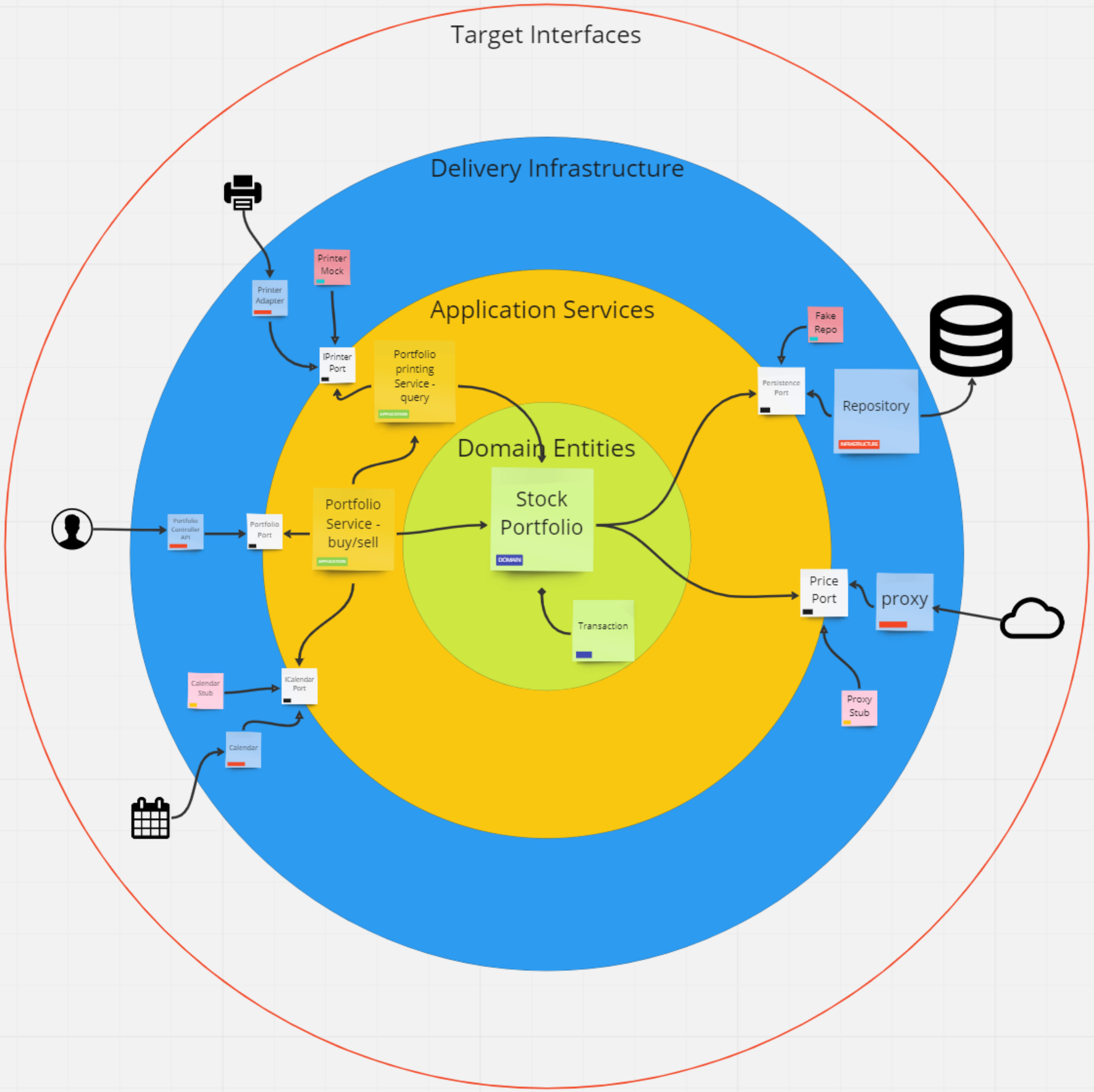


Accessibility

Outside -> inside

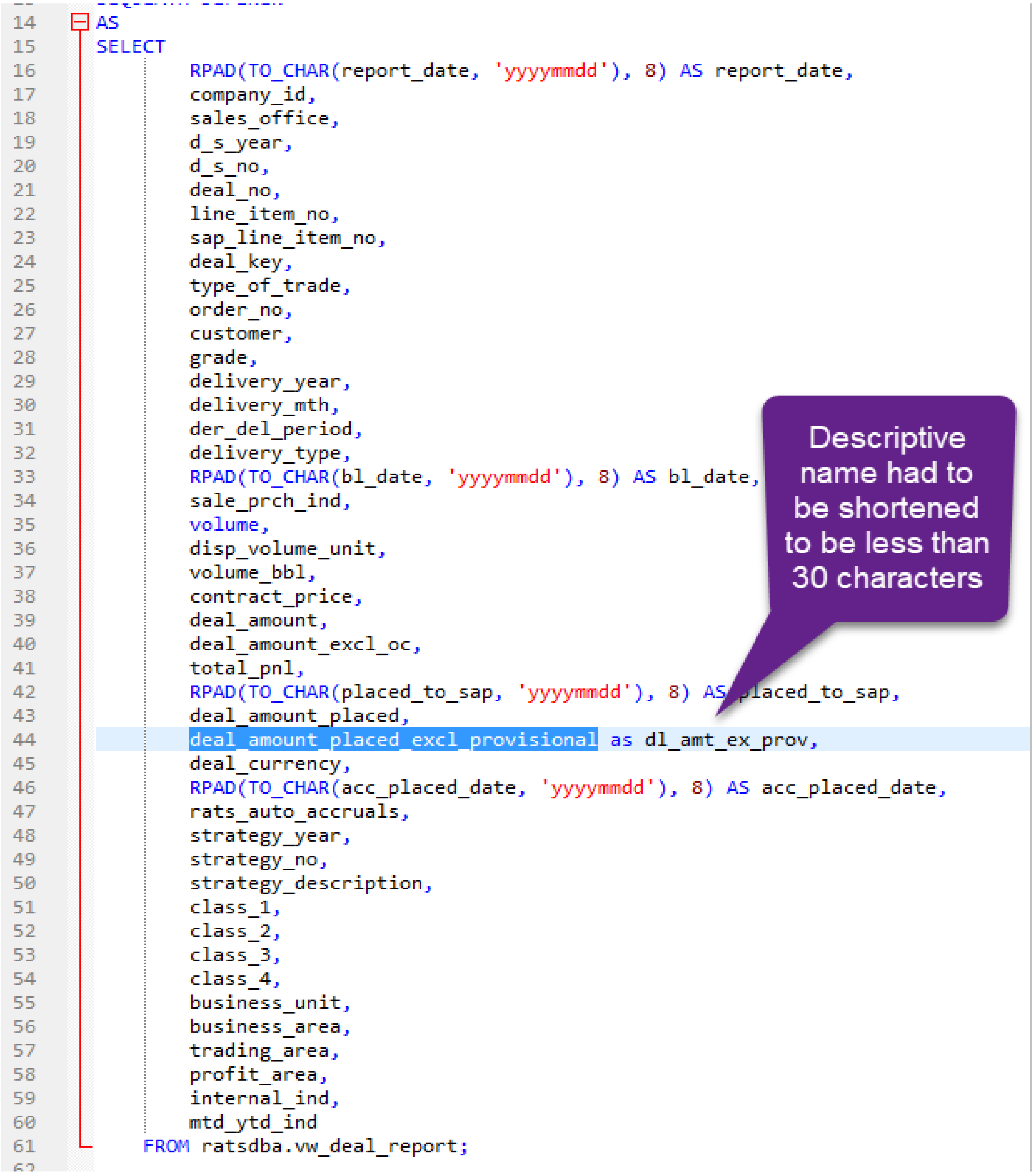
Onion Architecture

Exercise: Stock Portfolio



Missing End-to-End test

An Oracle view has been provided as interface for an external system (SAP) to access a report. View serves as a contract between the two systems, and was successfully released – based on acceptance test.



```
AS
SELECT
        RPAD(TO_CHAR(report_date, 'yyyymmdd'), 8) AS report_date,
        company_id,
        sales_office,
        d_s_year,
        d_s_no,
        deal_no,
        line_item_no,
        sap_line_item_no,
        deal_key,
        type_of_trade,
        order_no,
        customer,
        grade,
        delivery_year,
        delivery_mth,
        der_del_period,
        delivery_type,
        RPAD(TO_CHAR(bl_date, 'yyyymmdd'), 8) AS bl_date,
        sale_prch_ind,
        volume,
        disp_volume_unit,
        volume_bbl,
        contract_price,
        deal_amount,
        deal_amount_excl_oc,
        total_pnl,
        RPAD(TO_CHAR(placed_to_sap, 'yyyymmdd'), 8) AS placed_to_sap,
        deal_amount_placed,
        deal_amount_placed_excl_provisional as dl_amt_ex_prov,
        deal_currency,
        RPAD(TO_CHAR(acc_placed_date, 'yyyymmdd'), 8) AS acc_placed_date,
        rats_auto_accruals,
        strategy_year,
        strategy_no,
        strategy_description,
        class_1,
        class_2,
        class_3,
        class_4,
        business_unit,
        business_area,
        trading_area,
        profit_area,
        internal_ind,
        mtd_ytd_ind
    FROM ratsdba.vw_deal_report;
```

Descriptive name had to be shortened to be less than 30 characters

Questions?

rune.holen@bouvet.no