ABAPConf 2024

How can a Test Pyramid be implemented?

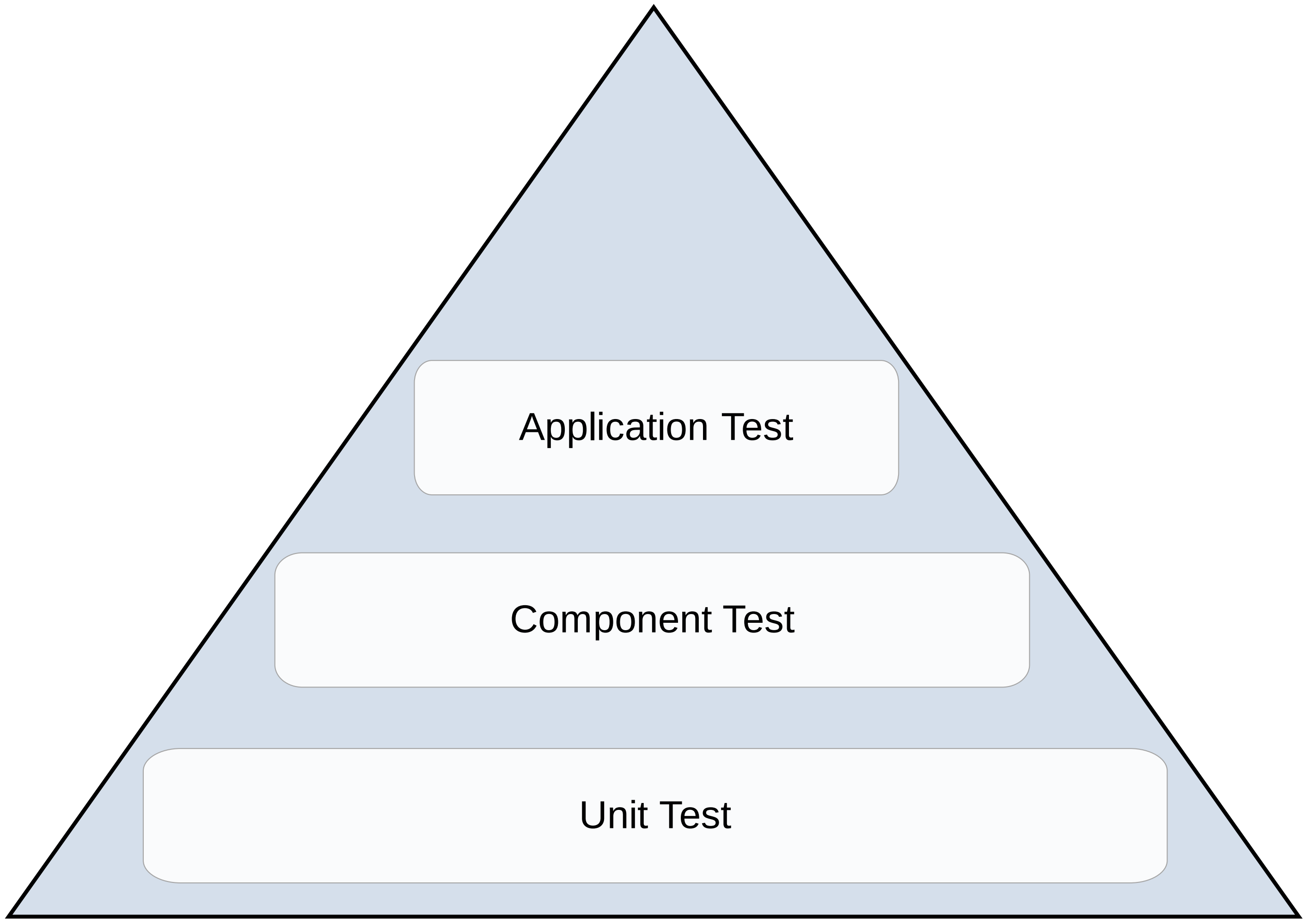
Winfried Schwarzmann, SAP SE | June 6, 2024

PUBLIC

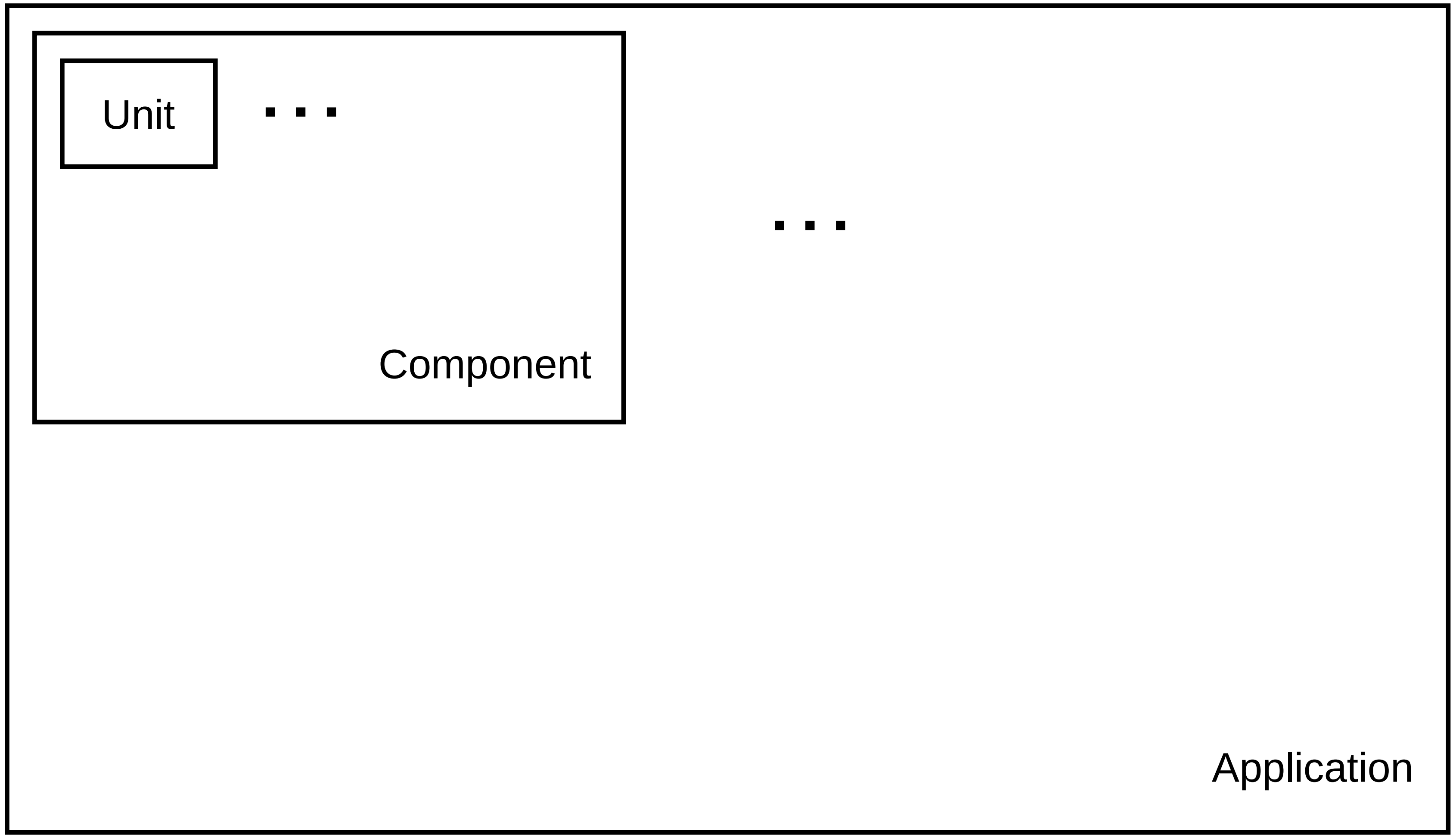
THE BEST RUN SAP®

Agenda

Test Pyramid of an Application

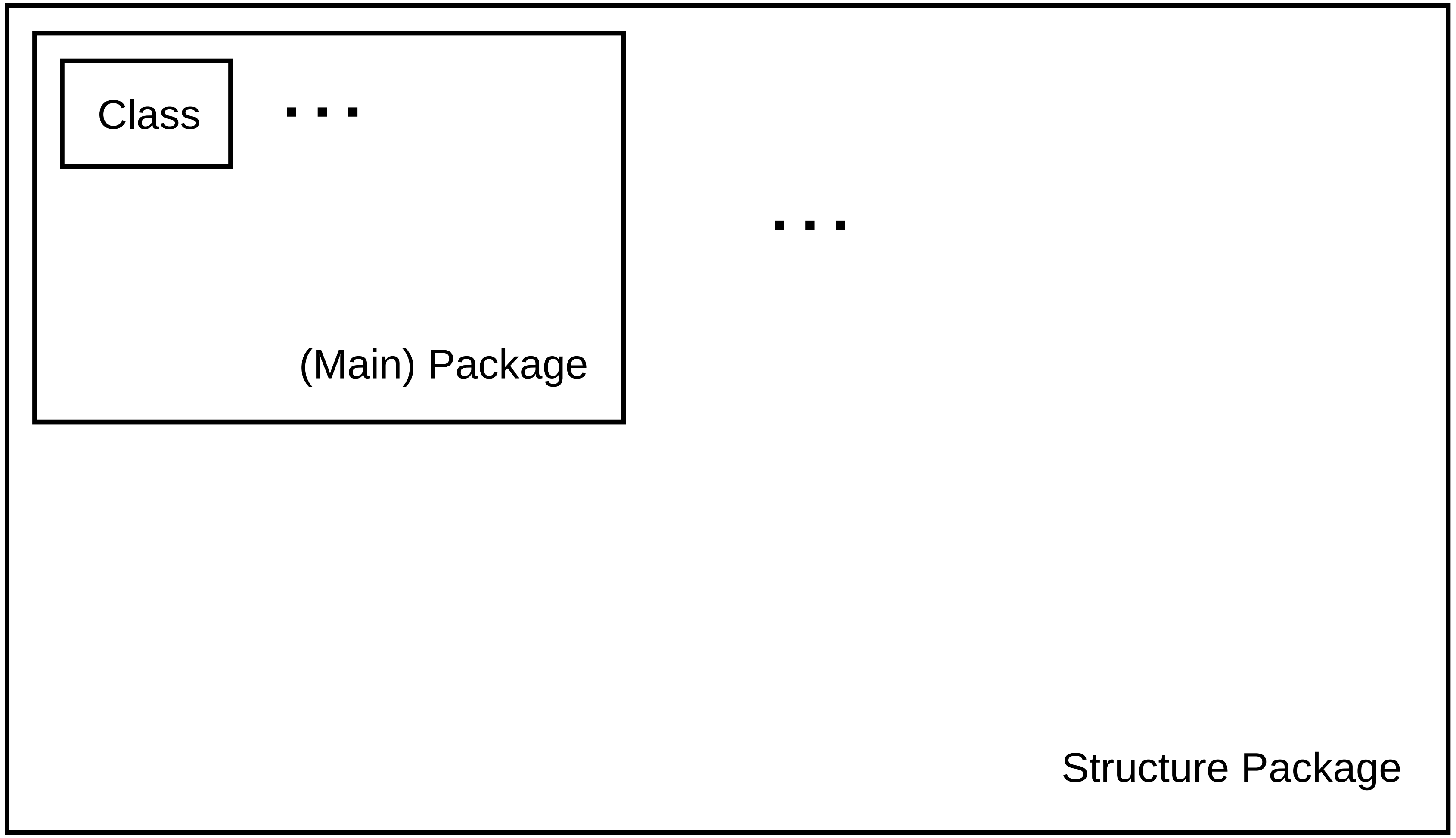


Reasoning



Divide & conquer
the Product

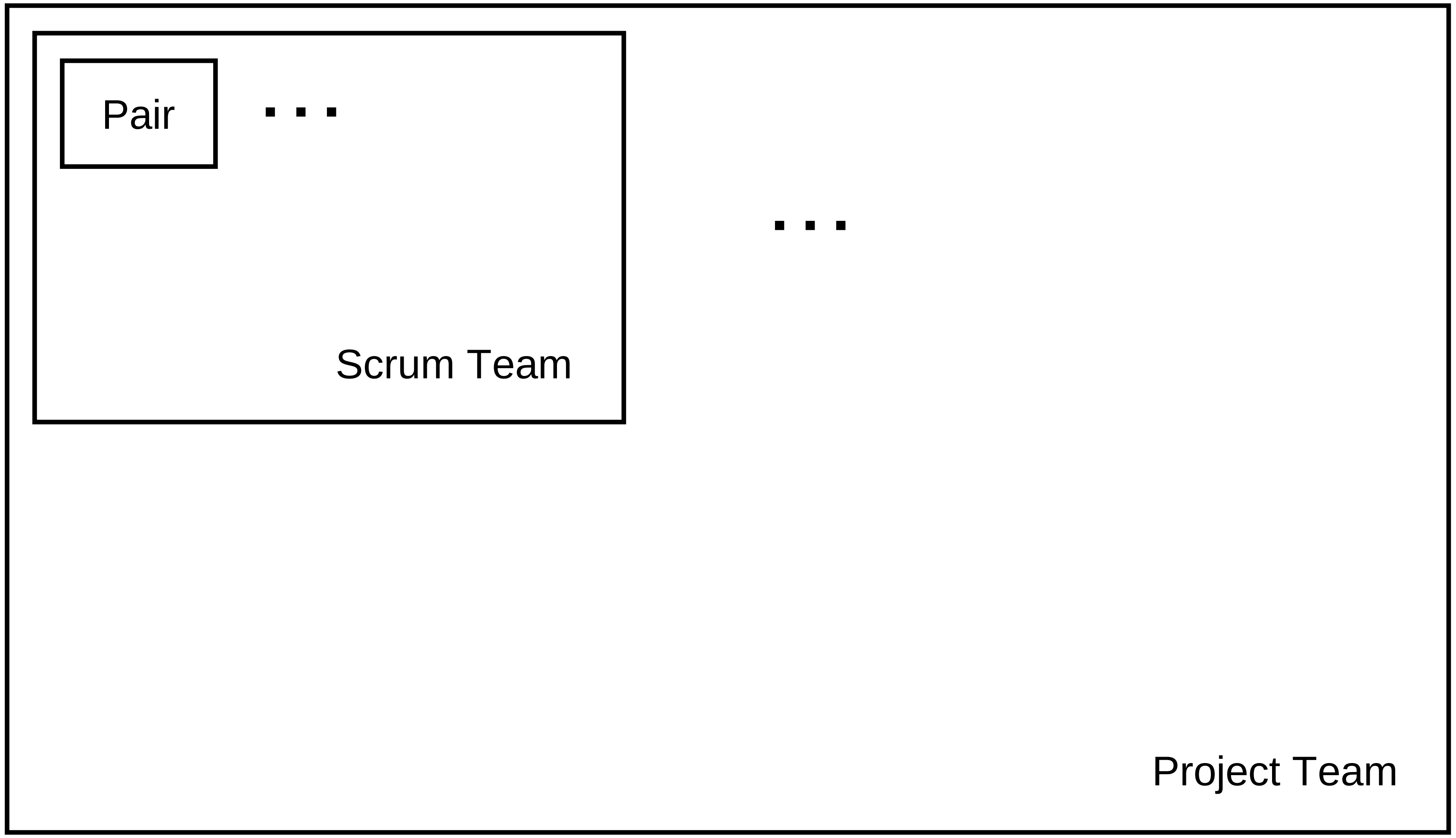
Support



ABAP Objects

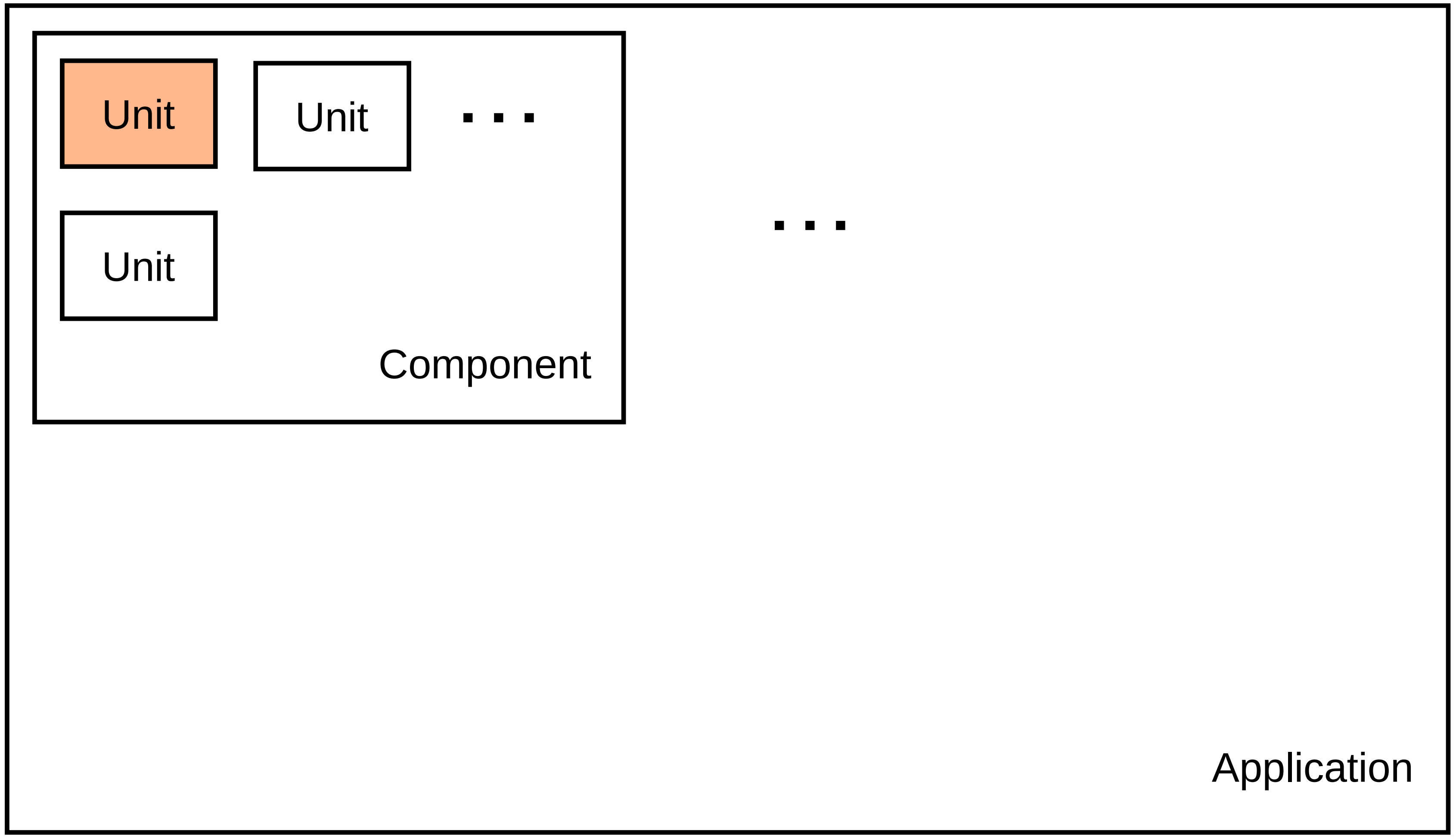
Class design & package concept

Implementation



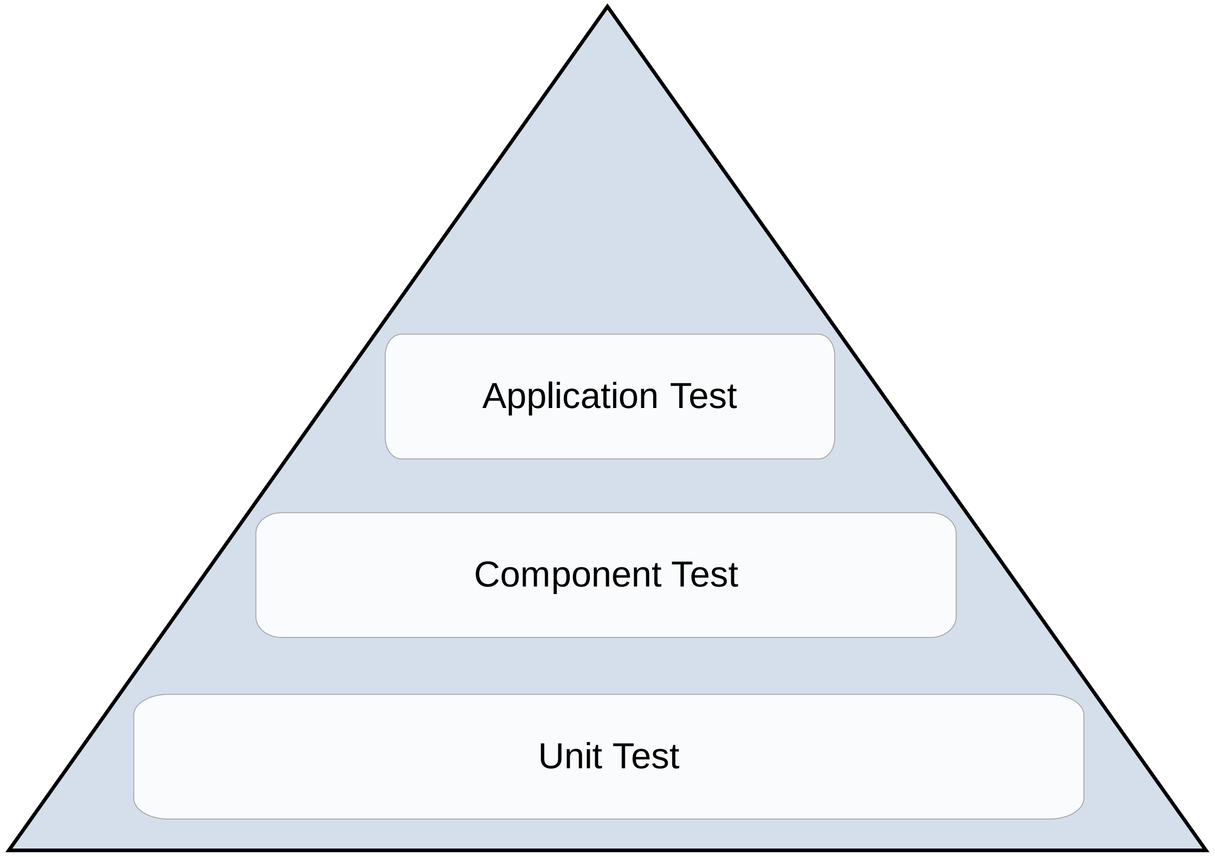
Decoupled collaboration

Efficiency



Independent development with test isolation

Effectiveness

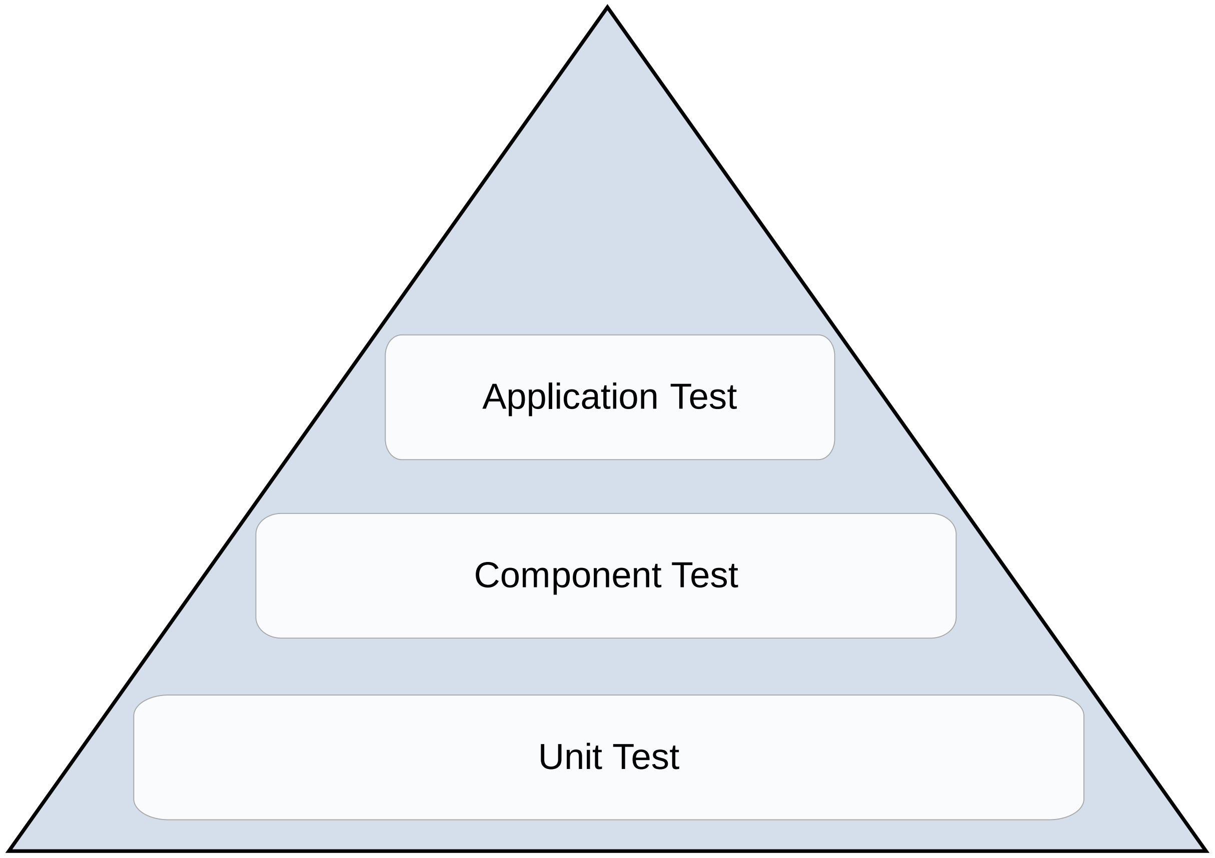


Customer-like scenarios

Test cases based on a test design

Statement & branch coverage

Isolated Test Pyramid



Further benefits:

- Runtime

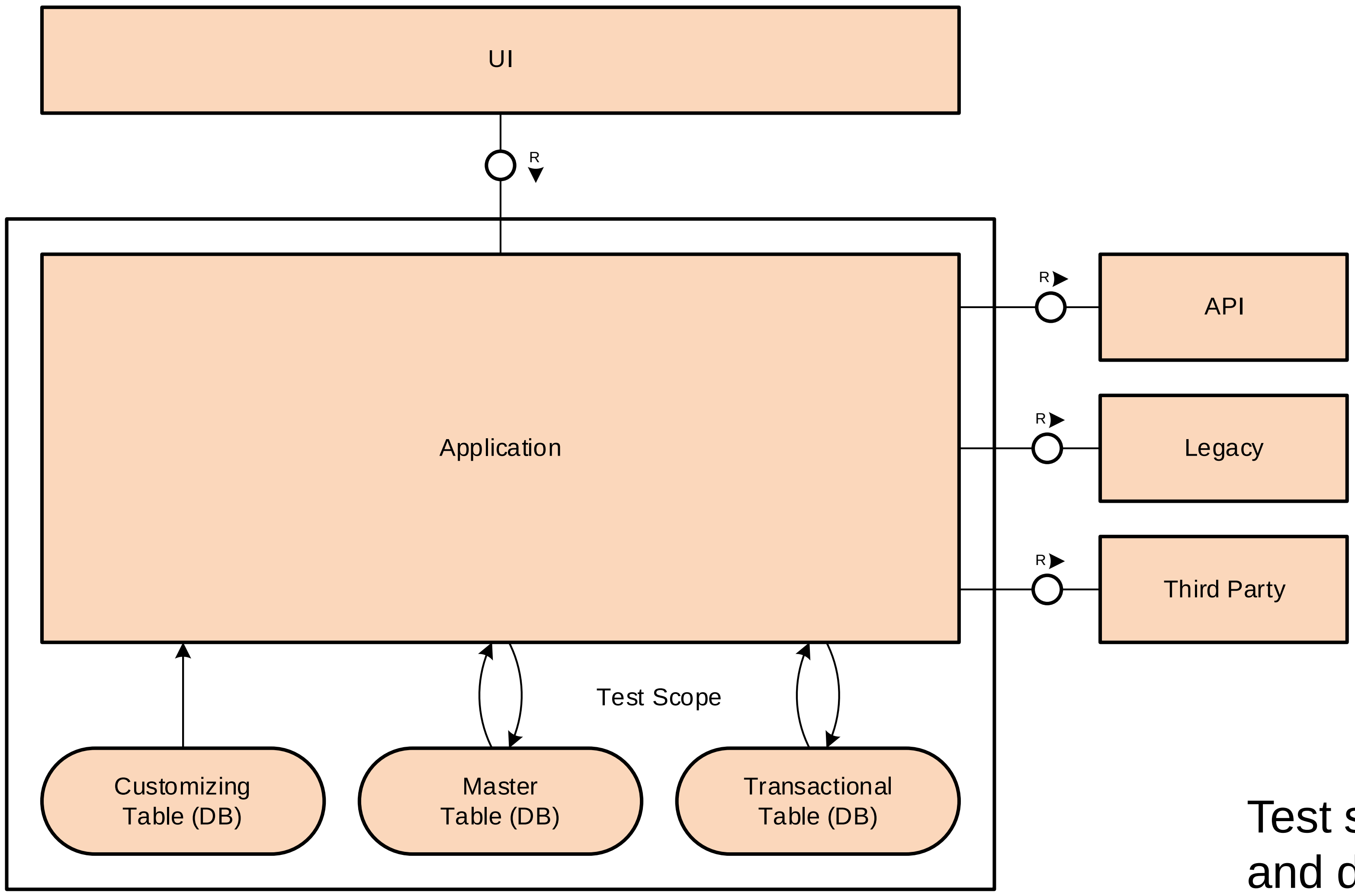
- Defect localization

- Decoupled design

Agenda

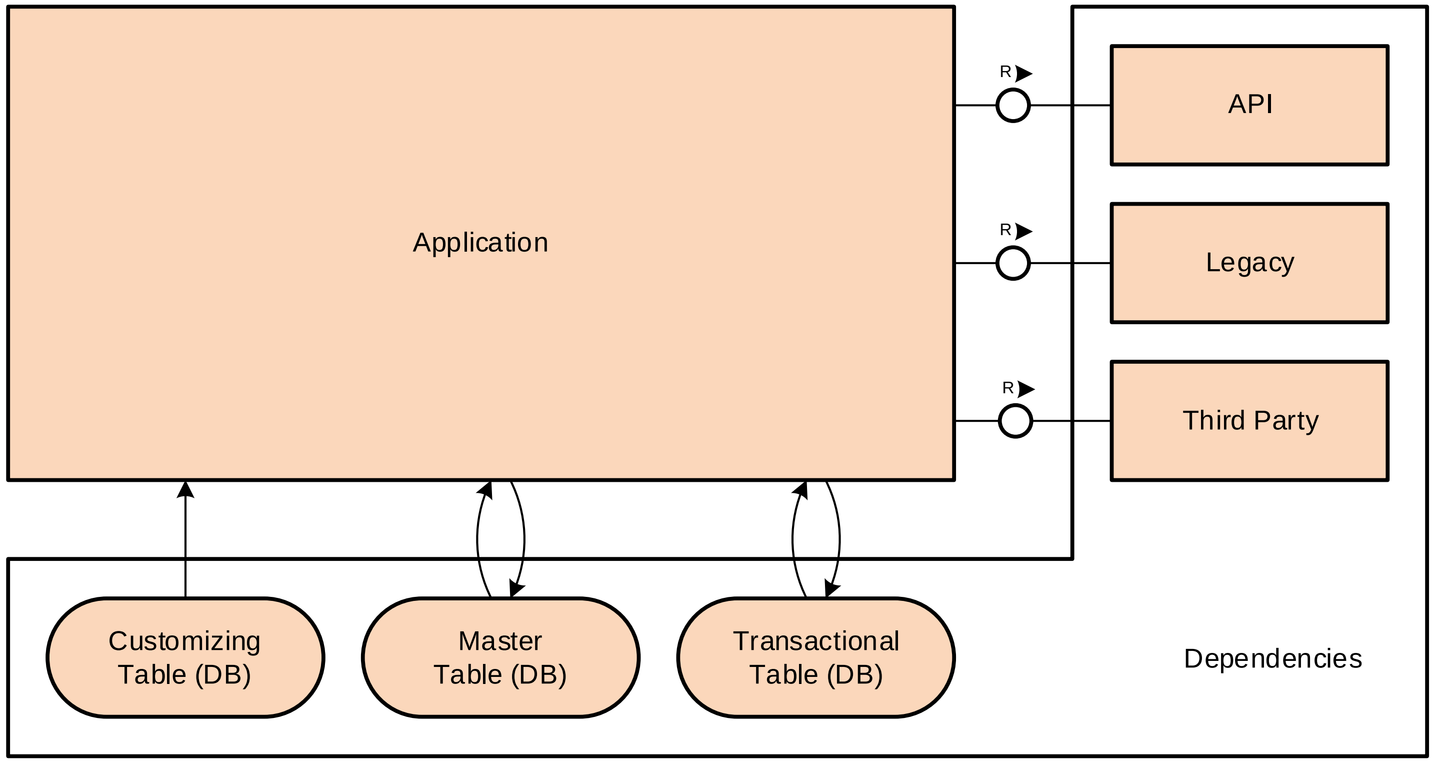
1. **Test Pyramid:** Motivation and **Design**
2. Test-Oriented Improvement Process
3. Clean Design
4. Resources

Test Scope of an Application



Test scope comprises data access and data processing.

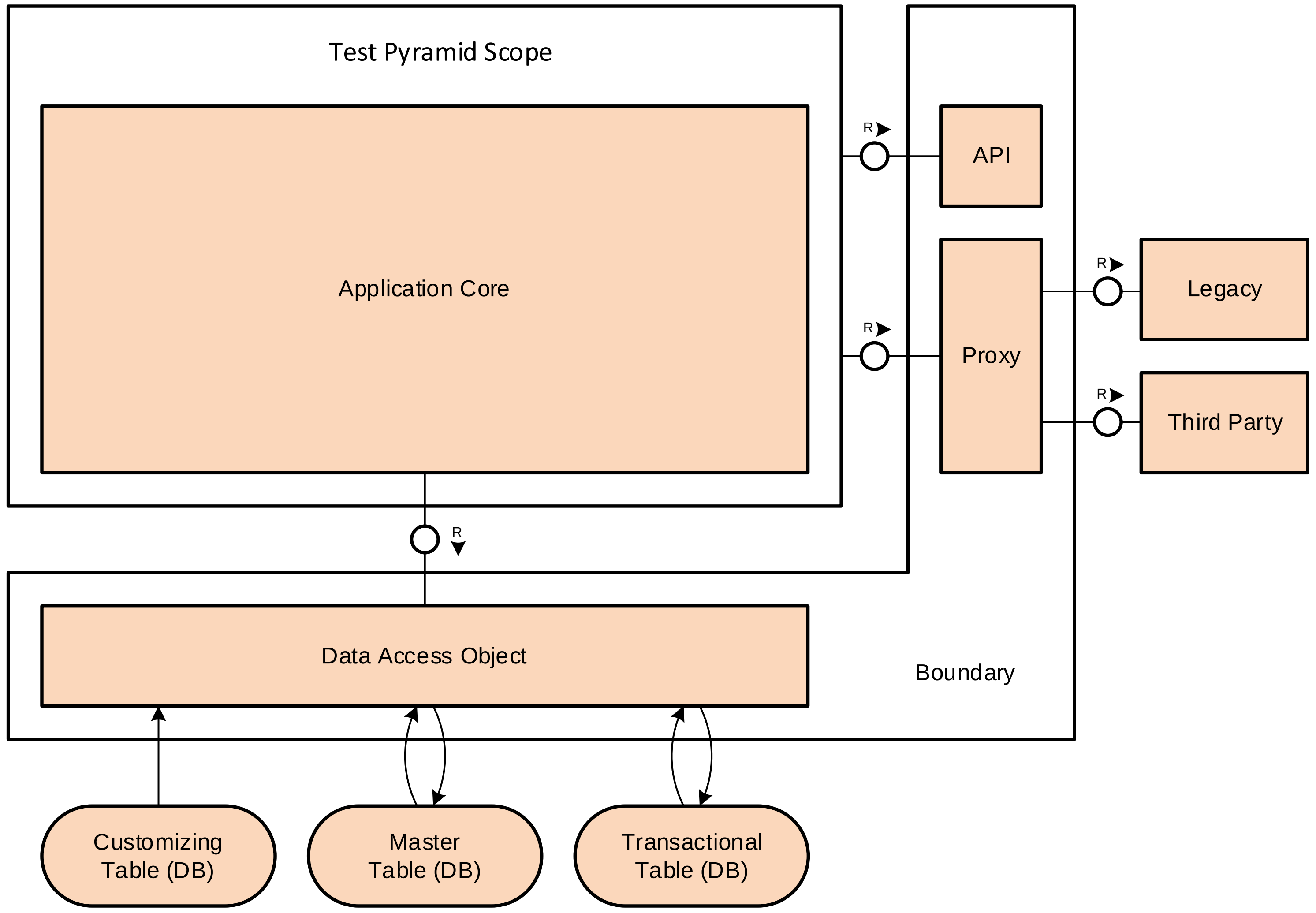
Dependencies of an Application



Functional dependencies

Data dependencies

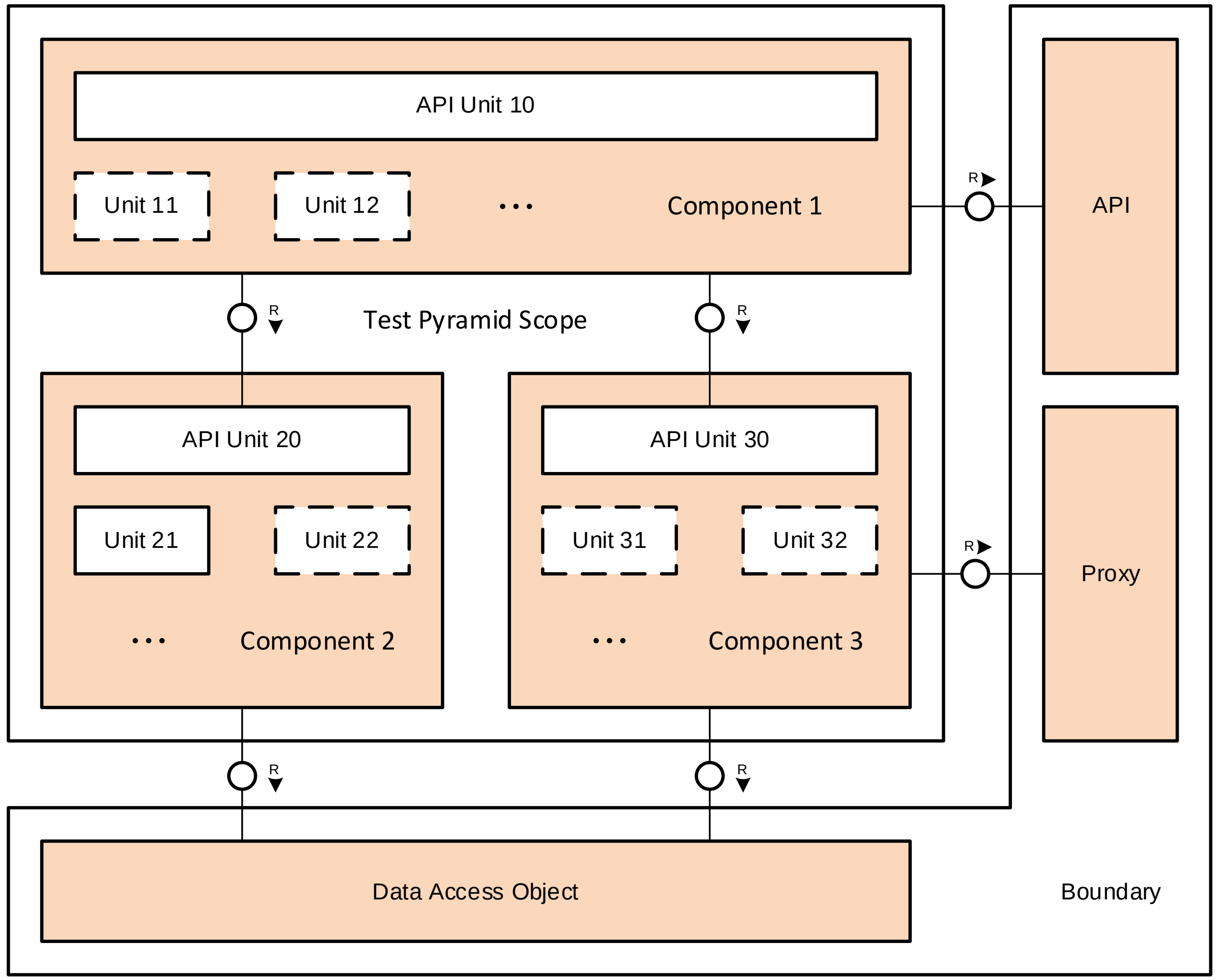
Boundary of an Application



Boundary decouples the test pyramid scope from its dependencies.

Data access object (DAO) is a slim encapsulation of SQL statements.

Components and Units of an Application



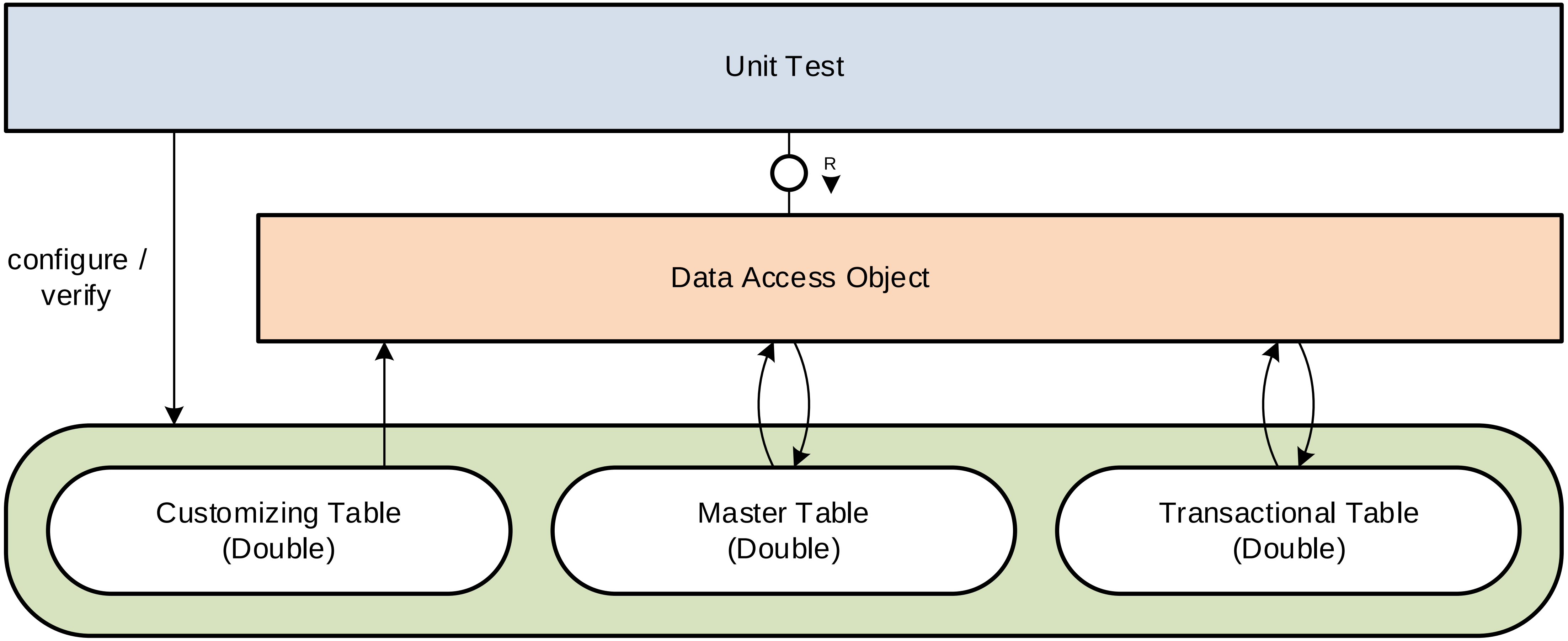
Each component should be represented by a (main) package.

Package should encapsulate everything but an API.

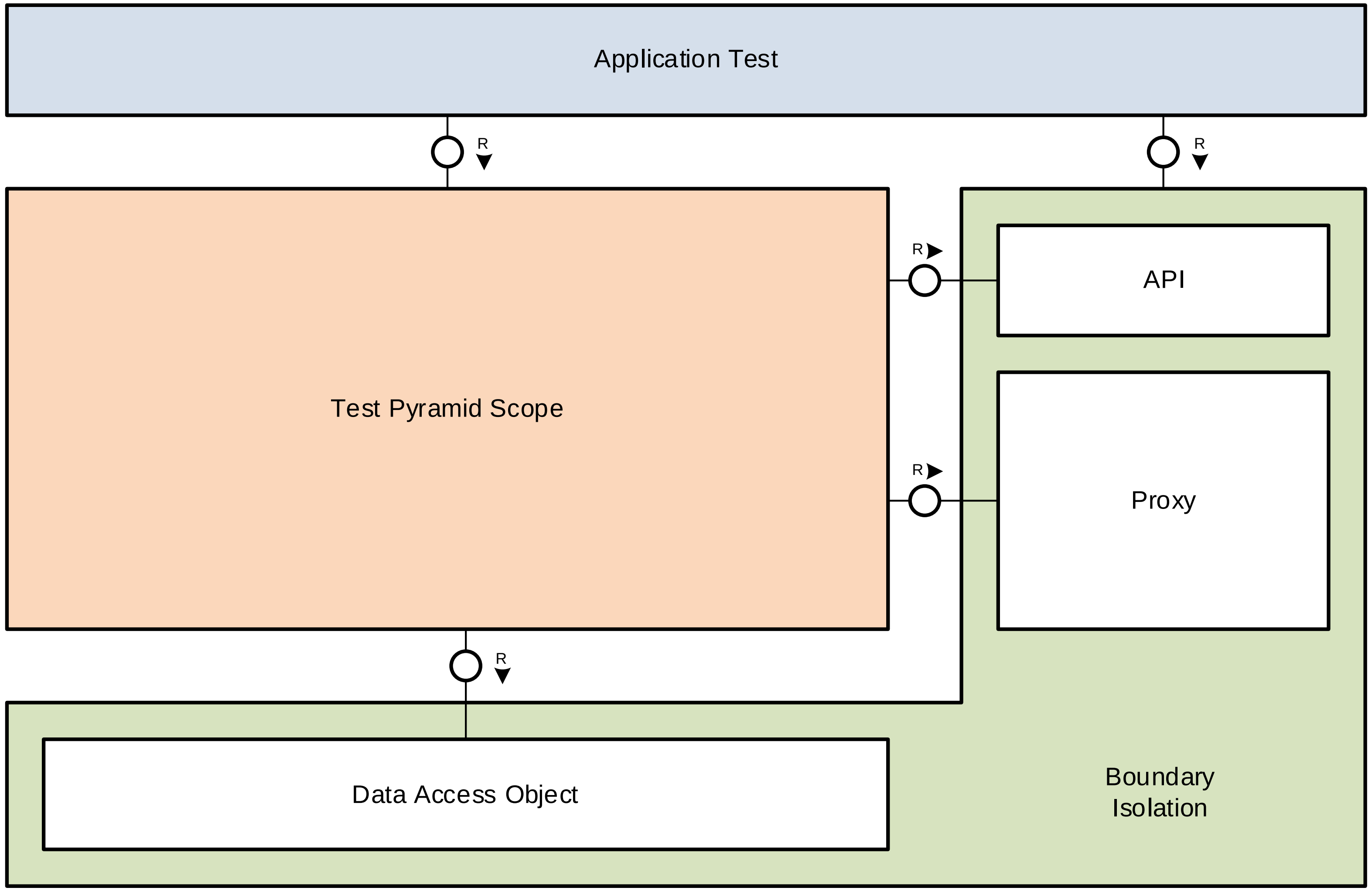
Visualizing Test Isolation

Test	Test classes implementing test cases
Test Infrastructure	Test help classes enabling readability and avoiding duplication of test code
Product	Product code under test
Double	Test double classes or test double frameworks simulating product code that is not under test

Testing the Data Access (Separately)

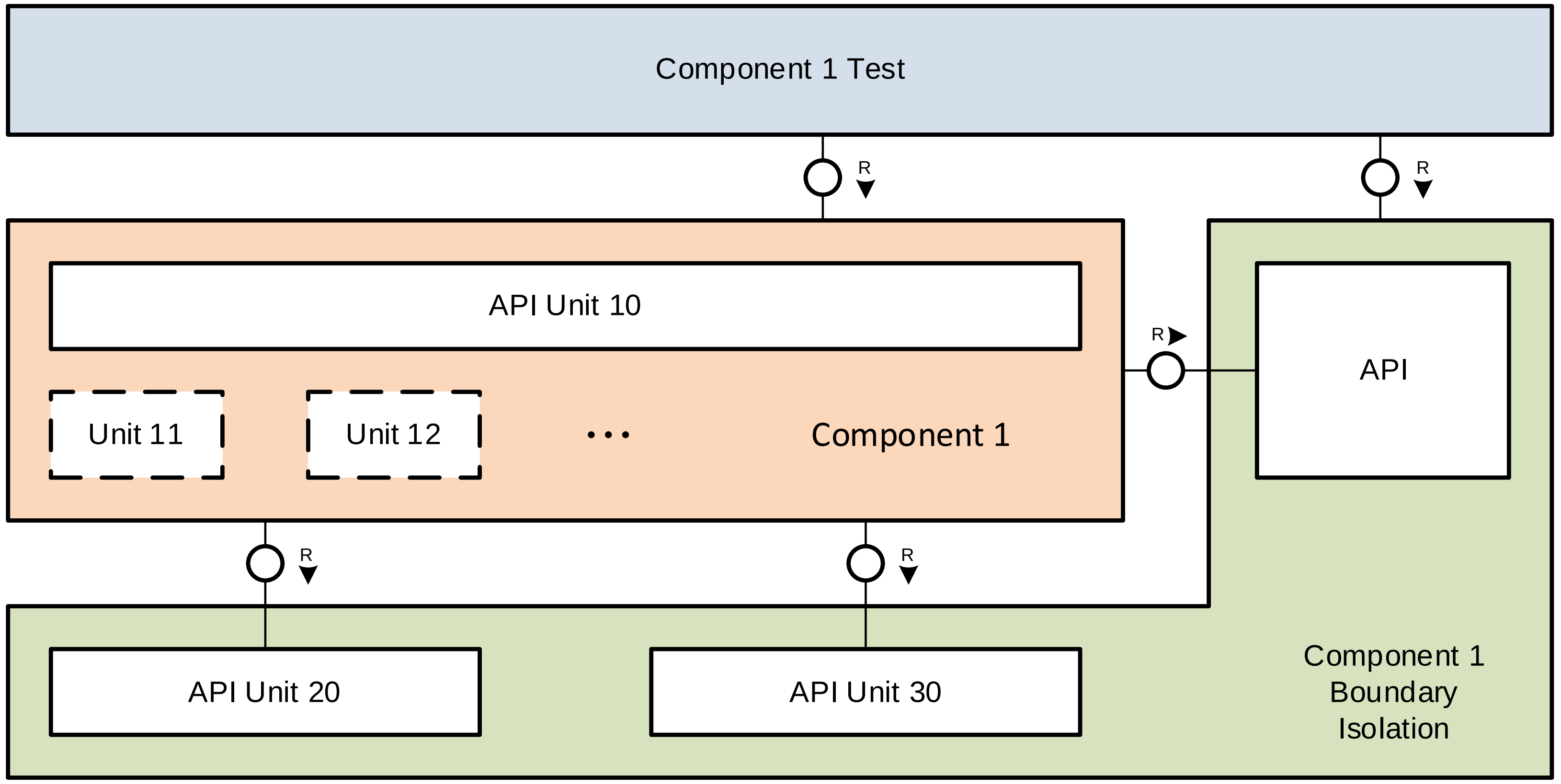


Testing the Application



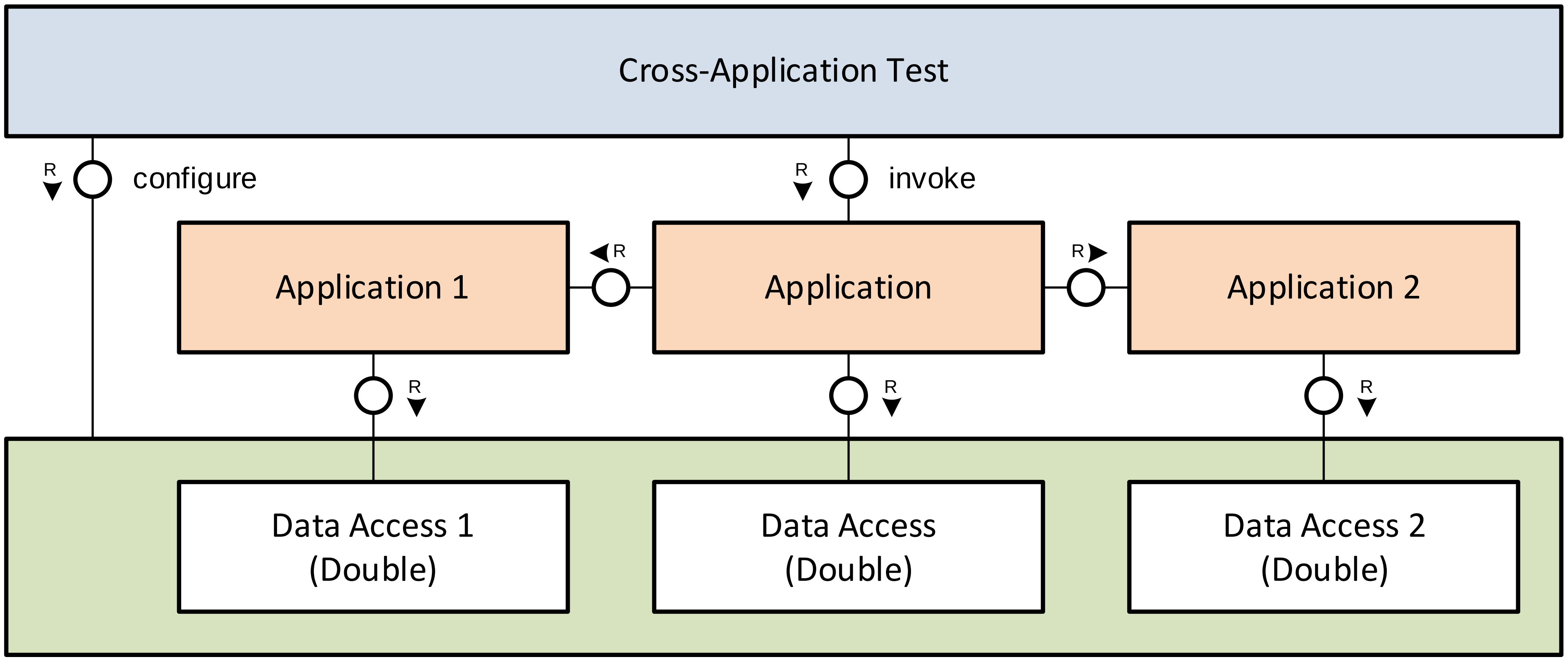
Isolated application tests use global test doubles for all boundary classes.

Testing a Component

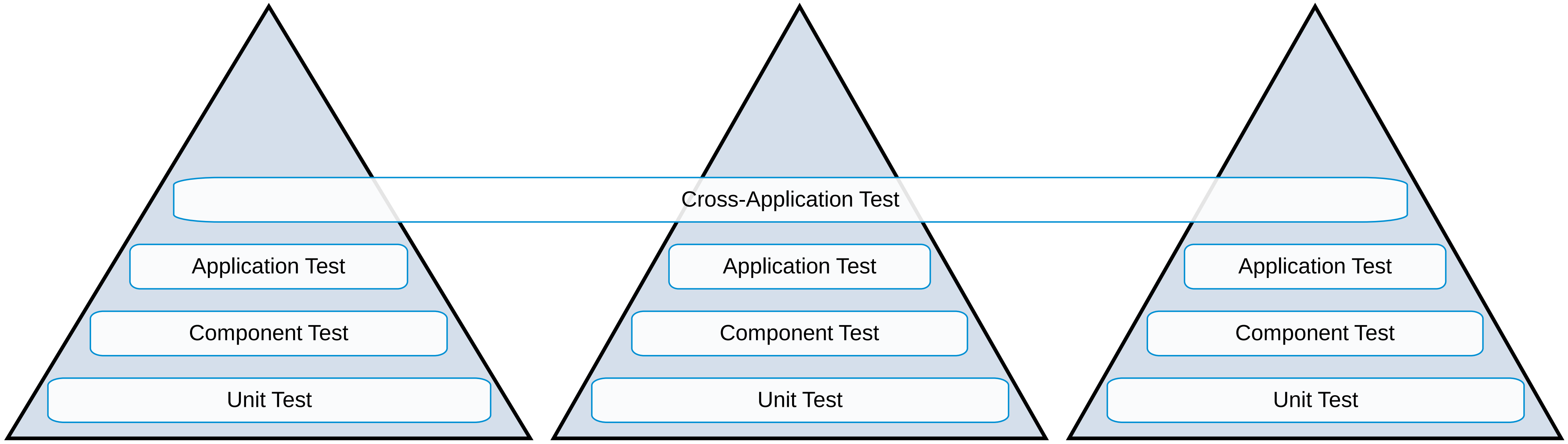


Isolated component tests also (re)use global test doubles for their boundary classes.

Testing Across Applications



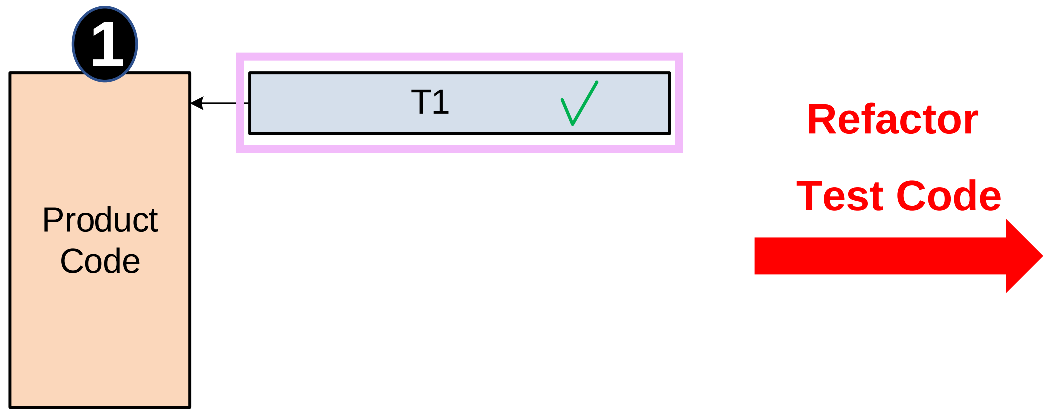
Test Pyramid of a System

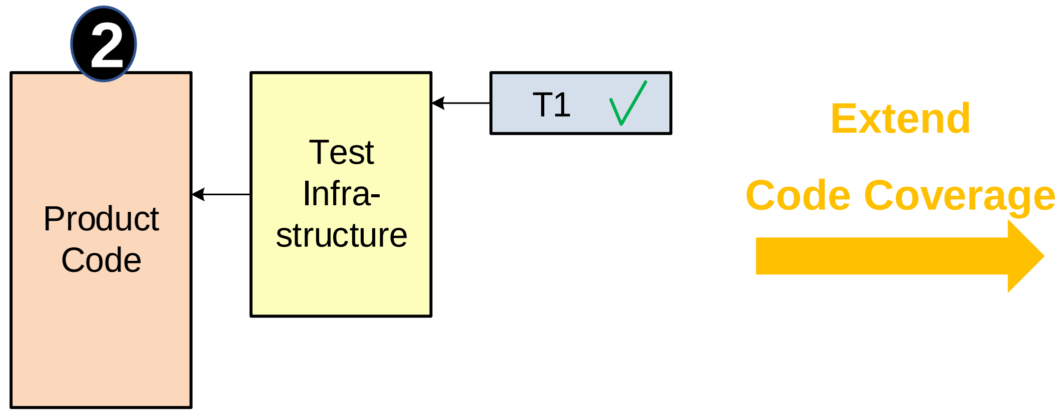


Agenda

1. Test Pyramid
2. **Test-Oriented Improvement Process**
3. Clean Design
4. Resources

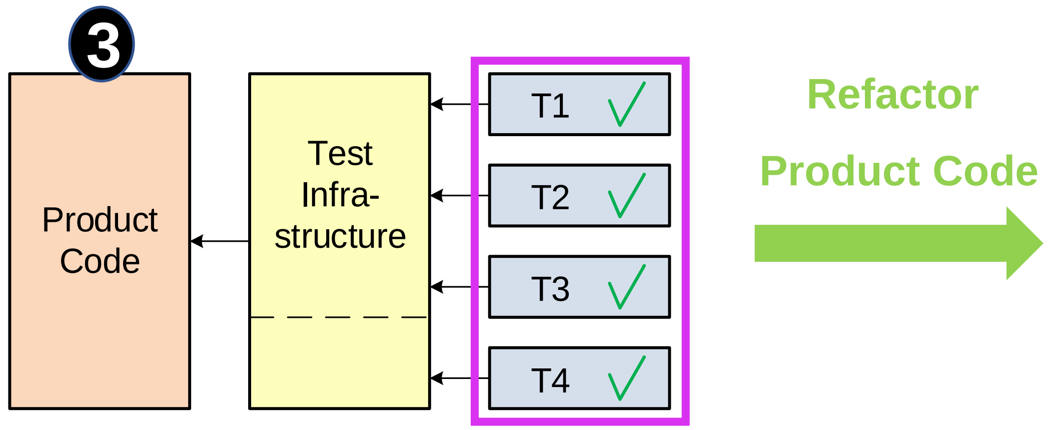
Test-Oriented Improvement Process

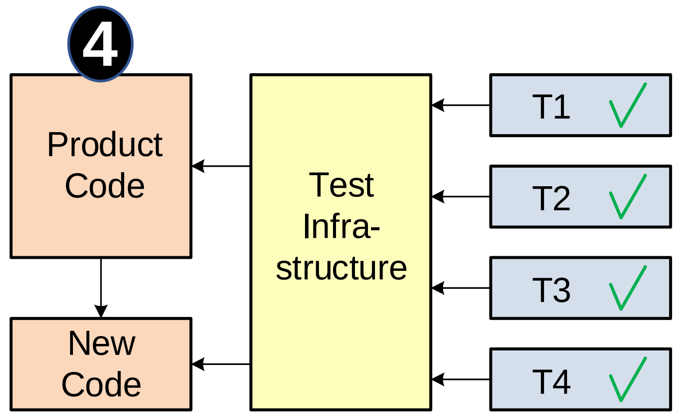




Hard to maintain tests for new or legacy code

Reusable help methods allow for readable tests





Test suite with effective tests

Test suite to reduce product code complexity

Improving the Test Suite

T1

```
METHOD get_saved_pfli_changes.
  DATA:
    lt_entity_type_keys    TYPE usmd_gov_api_ts_ent_tabl,
    ls_entity_type_keys    TYPE usmd_gov_api_s_ent_tabl,
    lt_entity_type_changes TYPE usmd_t_changed_entities,
    lsr_pfli_key           TYPE REF TO data,
    lv_entity_found        TYPE abap_bool.
  FIELD-SYMBOLS:
    <s_key>                 TYPE any,
    <s_pfli_key>            TYPE any,
    <t_pfli_key>            TYPE INDEX TABLE,
    <carr_id>               TYPE mdg_s_carr_id,
    <conn_id>               TYPE mdg_s_conn_id,
    <s_entity_type_changes> TYPE usmd_s_changed_entities,
    <s_entity_changes>      TYPE usmd_s_changed_entity.

  Create and configure the API object to be tested
  mo_conv_api = cl_usmd_conv_som_gov_api=>get_instance( 'SF' ).
  mo_conv_api->set_environment( iv_crequest_id = '3' ).

  Create an entity key for the given flight connection
  mo_conv_api->get_entity_structure(
    EXPORTING
      iv_entity_name = 'PFLI'
    IMPORTING
      er_structure   = lsr_pfli_key
  ).
  ASSIGN lsr_pfli_key->* TO <s_pfli_key>.
  ASSIGN COMPONENT 'CARR' OF STRUCTURE <s_pfli_key> TO <carr_id>.
  <carr_id> = 'AIR'.
  ASSIGN COMPONENT 'PFLI' OF STRUCTURE <s_pfli_key> TO <conn_id>.
  <conn_id> = '0001'.

  Insert entity key into a new entity key table
  ls_entity_type_keys-entity = 'PFLI'.
  mo_conv_api->get_entity_structure(
    EXPORTING
      iv_entity_name = 'PFLI'
    IMPORTING
      er_table       = ls_entity_type_keys-tabl
  ).
  ASSIGN ls_entity_type_keys-tabl->* TO <t_pfli_key>.
  INSERT <s_pfli_key> INTO TABLE <t_pfli_key>.
  INSERT ls_entity_type_keys INTO TABLE lt_entity_type_keys.

  Calculate changes for the entities in the entity key table
  lt_entity_type_changes = mo_conv_api->get_entity_field_changes(
      iv_struct             = zif_usmd_c=>struct_key_attr
      it_entity_keys        = lt_entity_type_keys
      iv_saved_changes      = abap_true
      iv_unsaved_changes    = abap_false
      iv_contained_changes = abap_false
  ).

  Search for changes table for the entity type in question
  READ TABLE lt_entity_type_changes ASSIGNING <s_entity_type_changes>
    WITH KEY entity_type = 'PFLI'
             struct      = zif_usmd_c=>struct_key_attr.
  cl_abap_unit_assert=>assert_subrc( exp = 0 ).

  Search for changes structure for the entity in question
  LOOP AT <s_entity_type_changes>-changed_entities ASSIGNING <s_entity_changes>.
    ASSIGN <s_entity_changes>-entity->* TO <s_key>.
    IF <s_key> = <s_pfli_key>.
      lv_entity_found = abap_true.
      EXIT.
    ENDIF.
  ENDLOOP.
  cl_abap_unit_assert=>assert_true( lv_entity_found ).

  Verify changes of this entity
  cl_abap_unit_assert=>assert_true( <s_entity_changes>-saved_change ).
ENDMETHOD.
```

Refactor

Test Code

Extend

Code Coverage

```
METHOD get_saved_pfli_node_changes.
  add_entity_to_api( mo_any_as_pfli ).

  import_key_of( mo_any_as_pfli ).
  get_nodes_changes( cs_only_saved_changes ).
  assert_only_saved_changes_of( mo_any_as_pfli ).
ENDMETHOD.
```

T1

```
METHOD get_unsaved_pfli_node_changes.
  add_entity_to_api( mo_other_u_pfli ).

  import_key_of( mo_other_u_pfli ).
  get_nodes_changes( cs_only_unsaved_changes ).
  assert_only_unsaved_changes_of( mo_other_u_pfli ).
ENDMETHOD.
```

T2

```
METHOD get_both_flight_node_changes.
  add_entity_to_api( mo_any_u_flight ).

  import_key_of( mo_any_u_flight ).
  get_nodes_changes( cs_both_changes ).
  assert_only_unsaved_changes_of( mo_any_u_flight ).
ENDMETHOD.
```

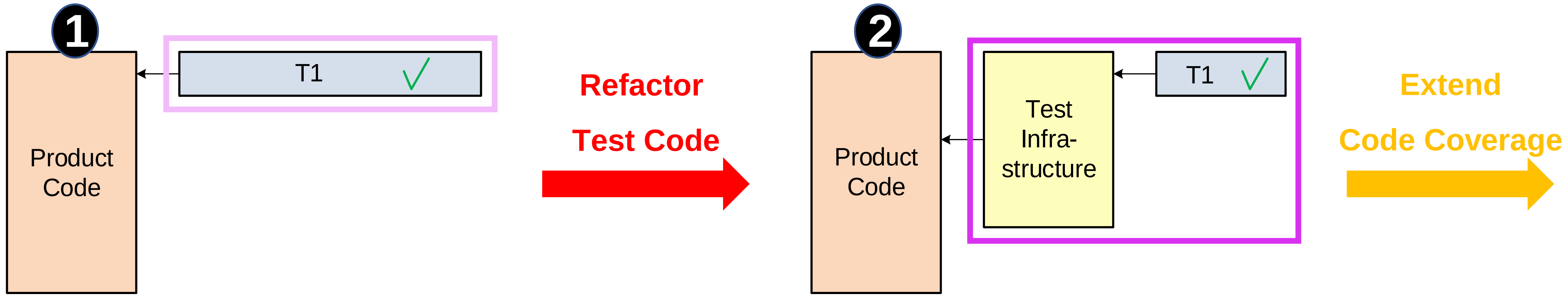
T3

```
METHOD get_unsaved_pfli_tree_changes.
  add_entity_to_api( mo_any_as_pfli ).
  add_entity_to_api( mo_any_u_flight ).

  import_key_of( mo_any_as_pfli ).
  get_trees_changes( cs_only_unsaved_changes ).
  assert_both_changes_of( mo_any_as_pfli ).
ENDMETHOD.
```

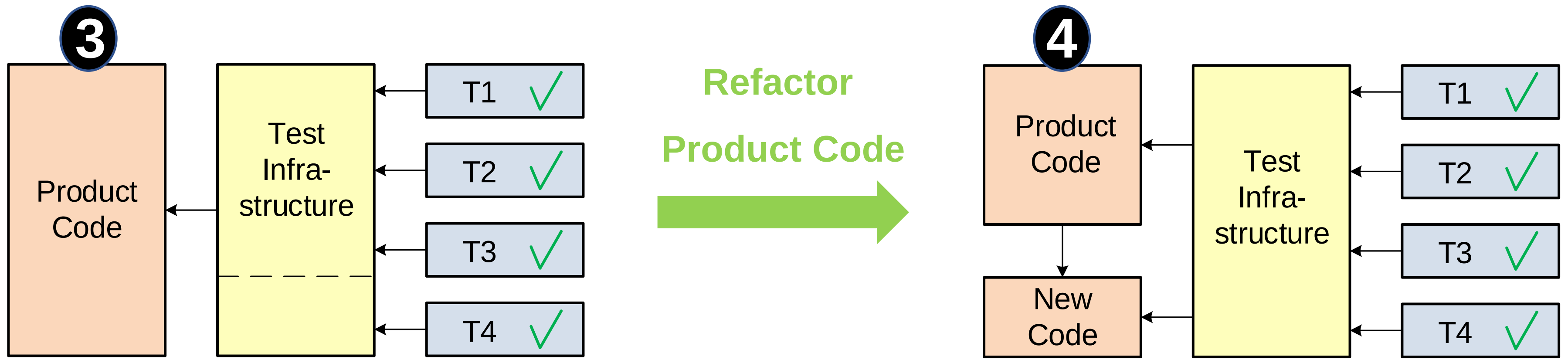
T4

Extracting the Test Infrastructure



Hard to maintain tests for new or legacy code

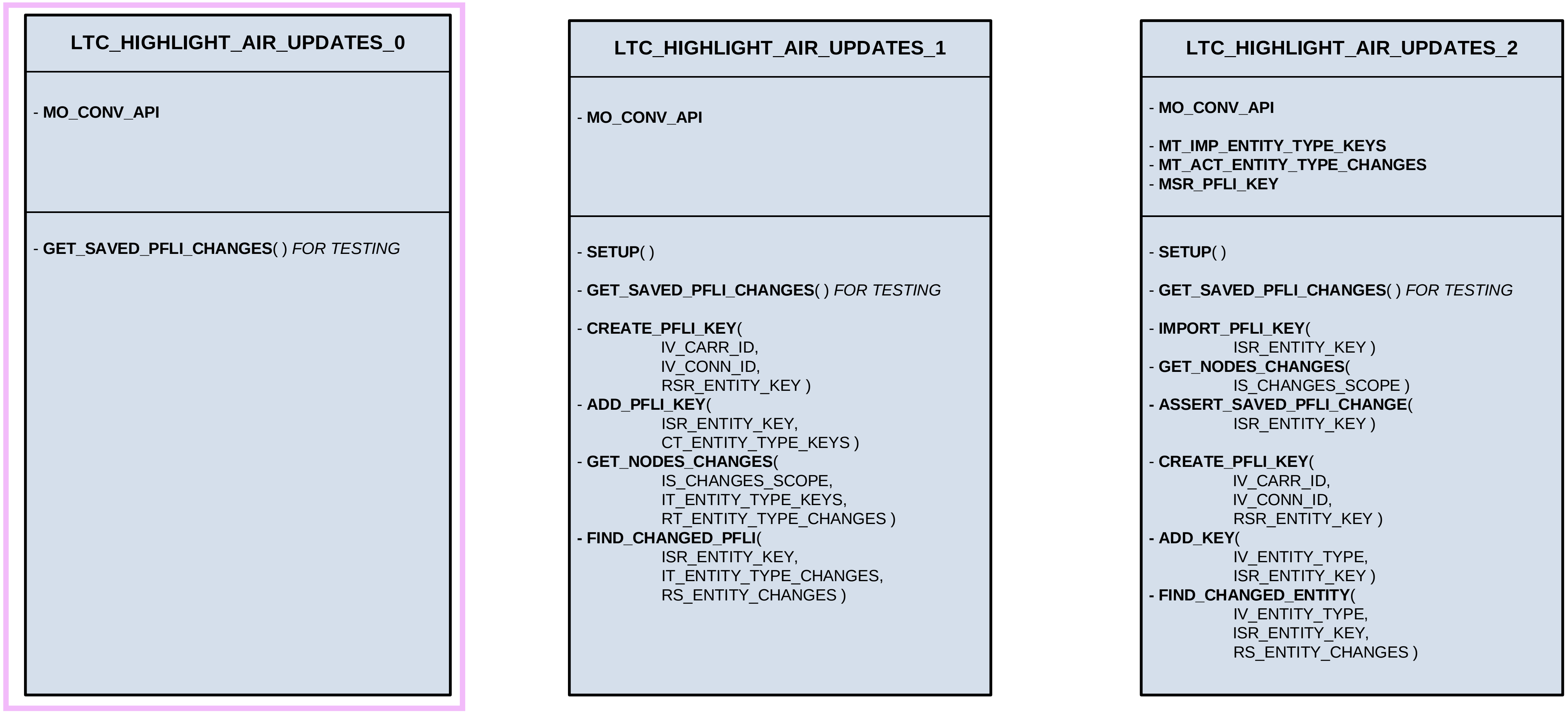
Reusable help methods allow for readable tests



Test suite with effective tests

Test suite to reduce product code complexity

Extracting within the Test Class



LTC_HIGHLIGHT_AIR_UPDATES_0

- **MO_CONV_API**

- **GET_SAVED_PFLI_CHANGES**( ) *FOR TESTING*

LTC_HIGHLIGHT_AIR_UPDATES_1

- **MO_CONV_API**

```
- SETUP( )

- GET_SAVED_PFLI_CHANGES( ) FOR TESTING

- CREATE_PFLI_KEY(
                IV_CARR_ID,
                IV_CONN_ID,
                RSR_ENTITY_KEY )
- ADD_PFLI_KEY(
                ISR_ENTITY_KEY,
                CT_ENTITY_TYPE_KEYS )
- GET_NODES_CHANGES(
                IS_CHANGES_SCOPE,
                IT_ENTITY_TYPE_KEYS,
                RT_ENTITY_TYPE_CHANGES )
- FIND_CHANGED_PFLI(
                ISR_ENTITY_KEY,
                IT_ENTITY_TYPE_CHANGES,
                RS_ENTITY_CHANGES )
```

LTC_HIGHLIGHT_AIR_UPDATES_2

```
- MO_CONV_API

- MT_IMP_ENTITY_TYPE_KEYS
- MT_ACT_ENTITY_TYPE_CHANGES
- MSR_PFLI_KEY
```

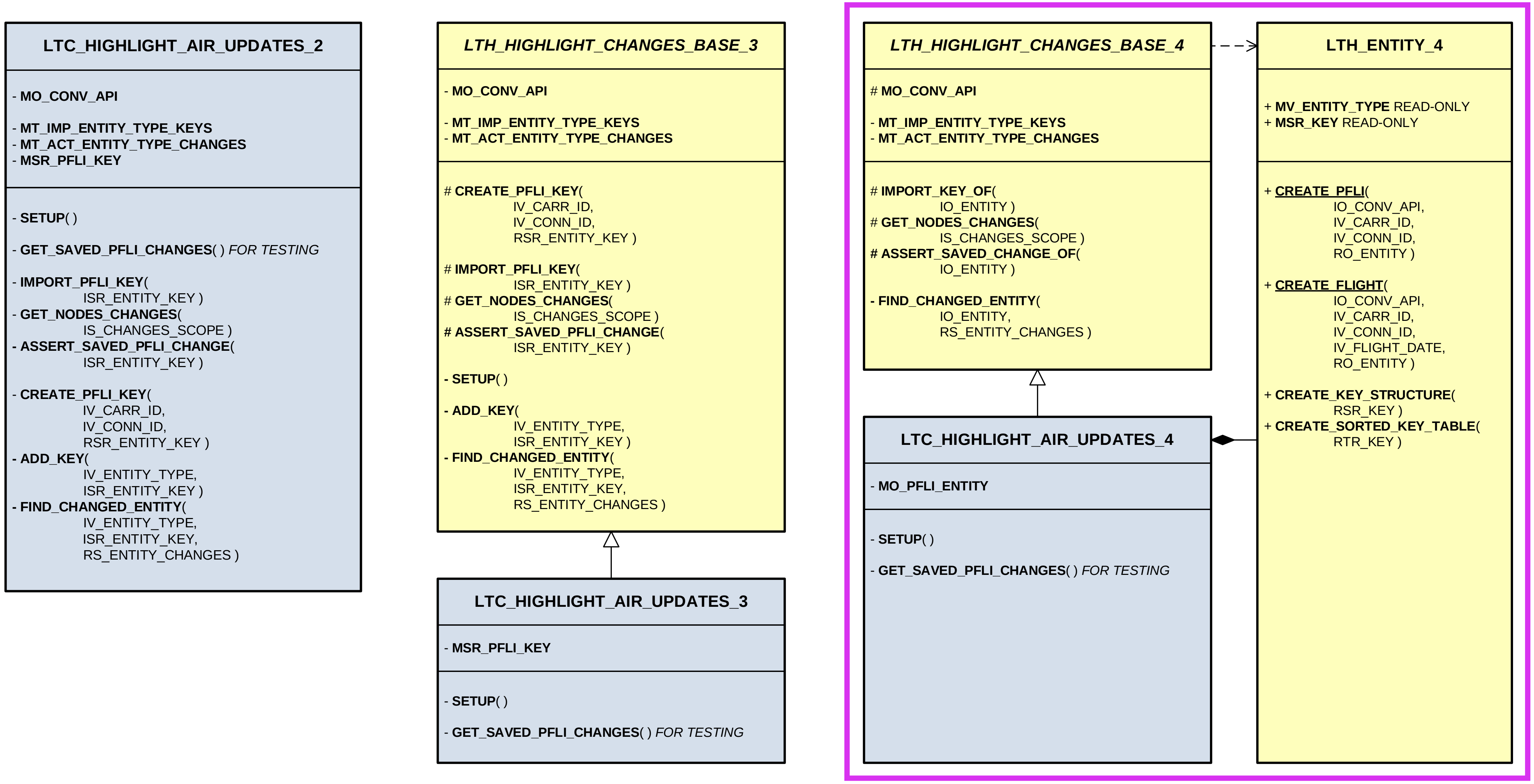
```
- SETUP( )

- GET_SAVED_PFLI_CHANGES( ) FOR TESTING

- IMPORT_PFLI_KEY(
                ISR_ENTITY_KEY )
- GET_NODES_CHANGES(
                IS_CHANGES_SCOPE )
- ASSERT_SAVED_PFLI_CHANGE(
                ISR_ENTITY_KEY )

- CREATE_PFLI_KEY(
                IV_CARR_ID,
                IV_CONN_ID,
                RSR_ENTITY_KEY )
- ADD_KEY(
                IV_ENTITY_TYPE,
                ISR_ENTITY_KEY )
- FIND_CHANGED_ENTITY(
                IV_ENTITY_TYPE,
                ISR_ENTITY_KEY,
                RS_ENTITY_CHANGES )
```

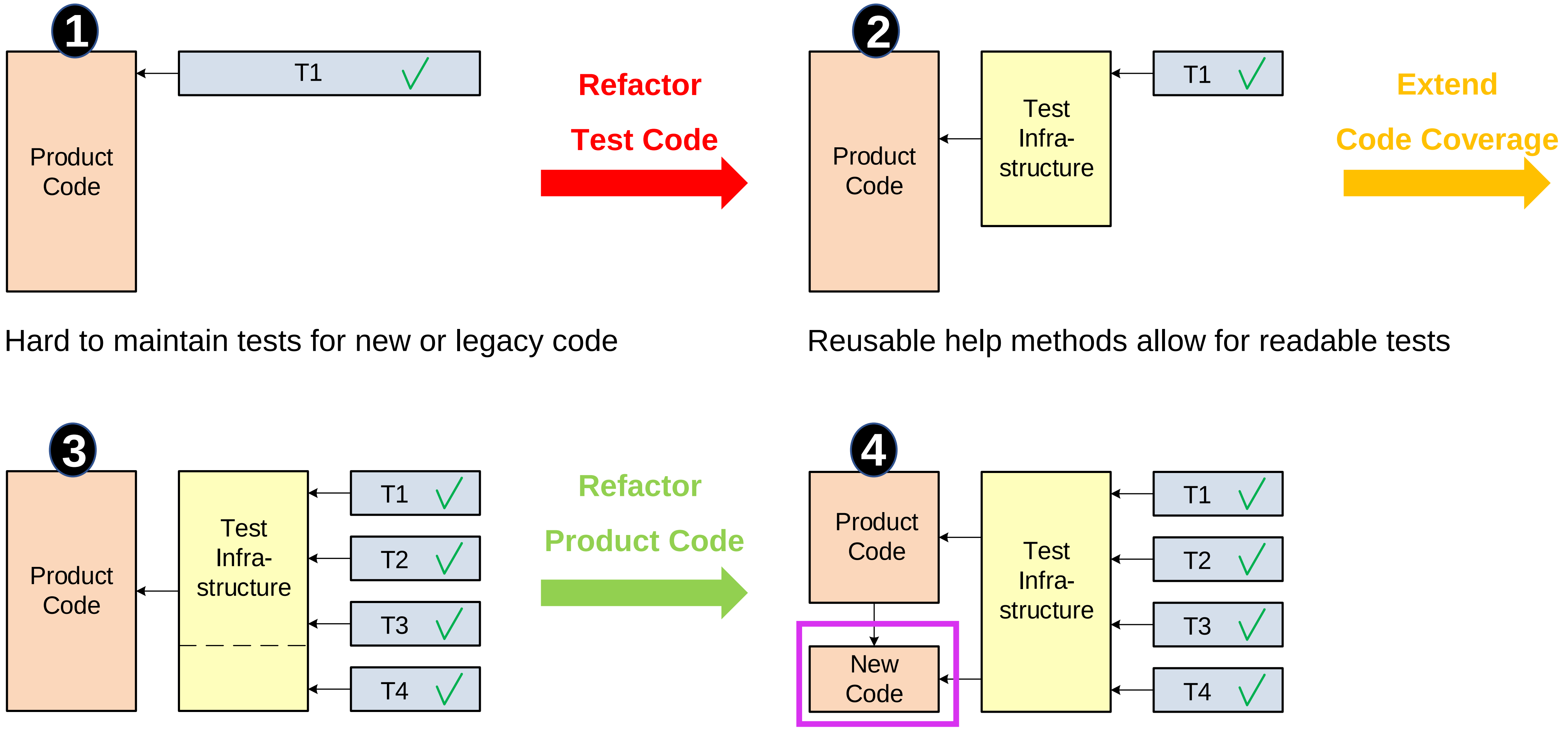
Extracting from the Test Class


LTC_HIGHLIGHT_AIR_UPDATES_2
- MO_CONV_API
- MT_IMP_ENTITY_TYPE_KEYS
- MT_ACT_ENTITY_TYPE_CHANGES
- MSR_PFLI_KEY
- SETUP( )
- GET_SAVED_PFLI_CHANGES( ) FOR TESTING
- IMPORT_PFLI_KEY(ISR_ENTITY_KEY)
- GET_NODES_CHANGES(IS_CHANGES_SCOPE)
- ASSERT_SAVED_PFLI_CHANGE(ISR_ENTITY_KEY)
- CREATE_PFLI_KEY(IV_CARR_ID, IV_CONN_ID, RSR_ENTITY_KEY)
- ADD_KEY(IV_ENTITY_TYPE, ISR_ENTITY_KEY)
- FIND_CHANGED_ENTITY(IV_ENTITY_TYPE, ISR_ENTITY_KEY, RS_ENTITY_CHANGES)
LTH_HIGHLIGHT_CHANGES_BASE_3
- MO_CONV_API
- MT_IMP_ENTITY_TYPE_KEYS
- MT_ACT_ENTITY_TYPE_CHANGES
# CREATE_PFLI_KEY(IV_CARR_ID, IV_CONN_ID, RSR_ENTITY_KEY)
# IMPORT_PFLI_KEY(ISR_ENTITY_KEY)
# GET_NODES_CHANGES(IS_CHANGES_SCOPE)
# ASSERT_SAVED_PFLI_CHANGE(ISR_ENTITY_KEY)
- SETUP( )
- ADD_KEY(IV_ENTITY_TYPE, ISR_ENTITY_KEY)
- FIND_CHANGED_ENTITY(IV_ENTITY_TYPE, ISR_ENTITY_KEY, RS_ENTITY_CHANGES)
LTC_HIGHLIGHT_AIR_UPDATES_3
- MSR_PFLI_KEY
- SETUP( )
- GET_SAVED_PFLI_CHANGES( ) FOR TESTING
LTH_HIGHLIGHT_CHANGES_BASE_4
# MO_CONV_API
- MT_IMP_ENTITY_TYPE_KEYS
- MT_ACT_ENTITY_TYPE_CHANGES
# IMPORT_KEY_OF(IO_ENTITY)
# GET_NODES_CHANGES(IS_CHANGES_SCOPE)
# ASSERT_SAVED_CHANGE_OF(IO_ENTITY)
- FIND_CHANGED_ENTITY(IO_ENTITY, RS_ENTITY_CHANGES)
LTC_HIGHLIGHT_AIR_UPDATES_4
- MO_PFLI_ENTITY
- SETUP( )
- GET_SAVED_PFLI_CHANGES( ) FOR TESTING
LTH_ENTITY_4
+ MV_ENTITY_TYPE READ-ONLY
+ MSR_KEY READ-ONLY
+ CREATE_PFLI(IO_CONV_API, IV_CARR_ID, IV_CONN_ID, RO_ENTITY)
+ CREATE_FLIGHT(IO_CONV_API, IV_CARR_ID, IV_CONN_ID, IV_FLIGHT_DATE, RO_ENTITY)
+ CREATE_KEY_STRUCTURE(RSR_KEY)
+ CREATE_SORTED_KEY_TABLE(RTR_KEY)

Agenda

1. Test Pyramid
2. Test-Oriented Improvement Process
3. **Clean Design**
4. Resources

Refactoring Product Code



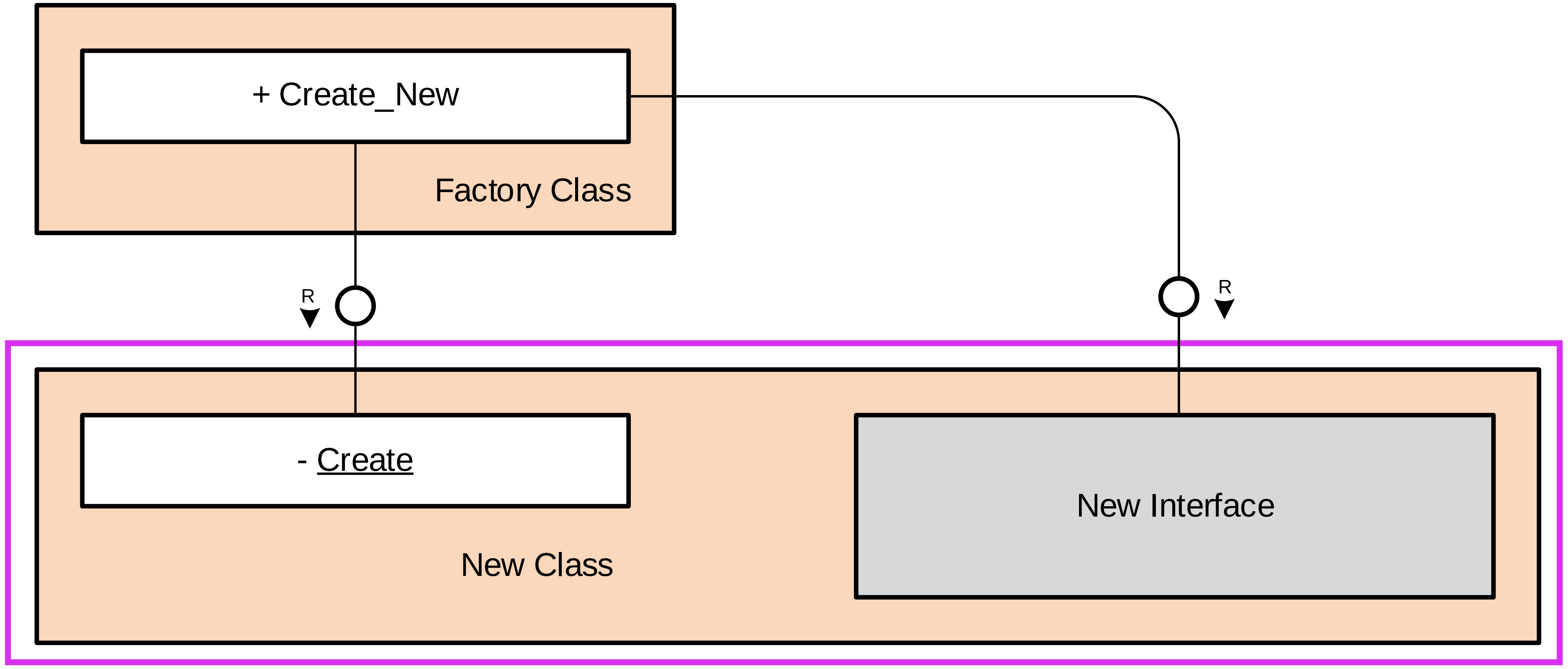
Hard to maintain tests for new or legacy code

Reusable help methods allow for readable tests

Test suite with effective tests

Test suite to reduce product code complexity

Designing a New Class



Implementing the Skeleton of a New Class

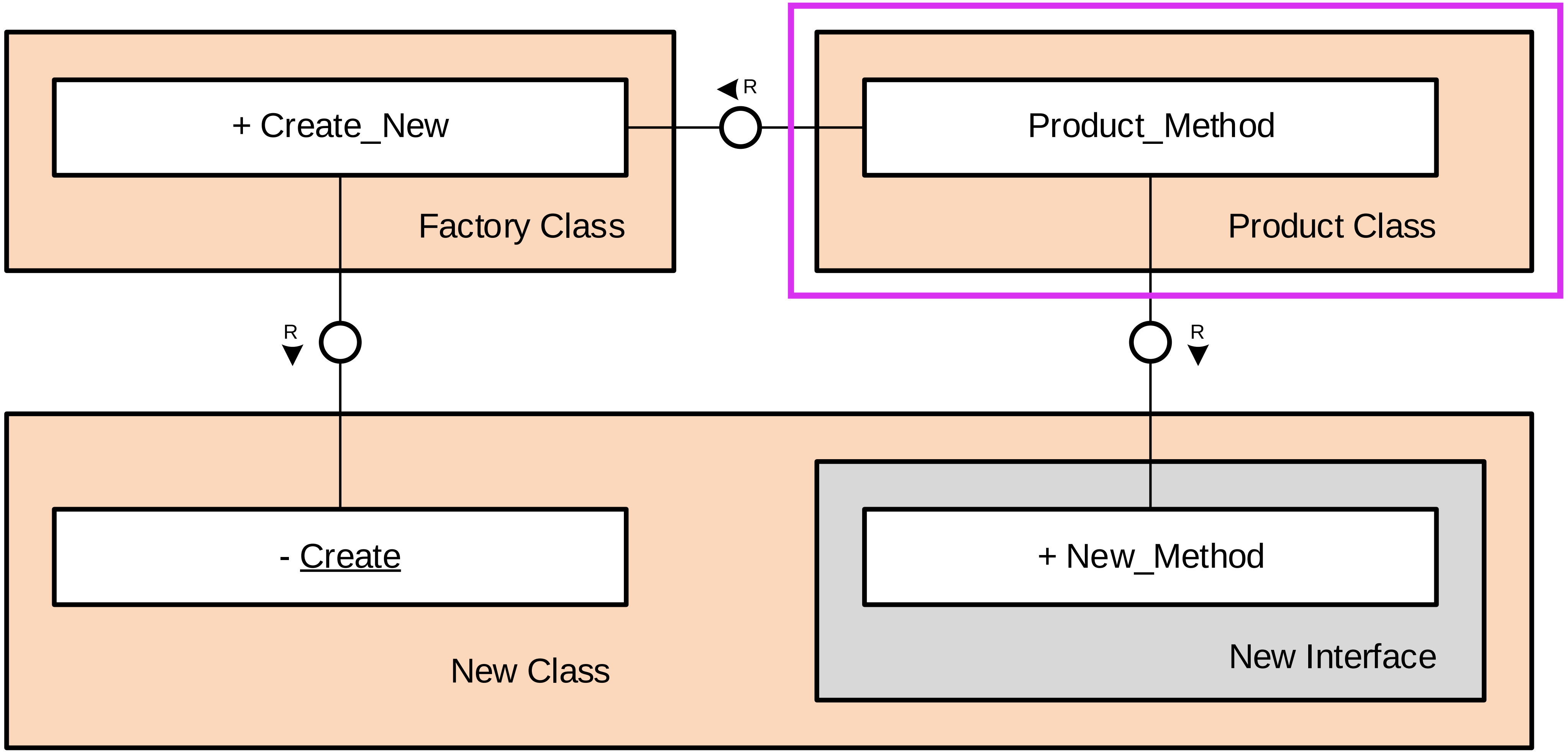
```
INTERFACE if_new PUBLIC.
ENDINTERFACE.


CLASS cl_new DEFINITION PUBLIC FINAL CREATE PRIVATE
GLOBAL FRIENDS cl_factory.
  PUBLIC SECTION.
    INTERFACES if_new.

  PRIVATE SECTION.
    CLASS-METHODS create
      RETURNING VALUE(ro_object) TYPE REF TO cl_new.
ENDCLASS.

CLASS cl_new IMPLEMENTATION.
  METHOD create.
    ro_object = NEW cl_new( ).
  ENDMETHOD.
ENDCLASS.
```

Decoupled from the New Class

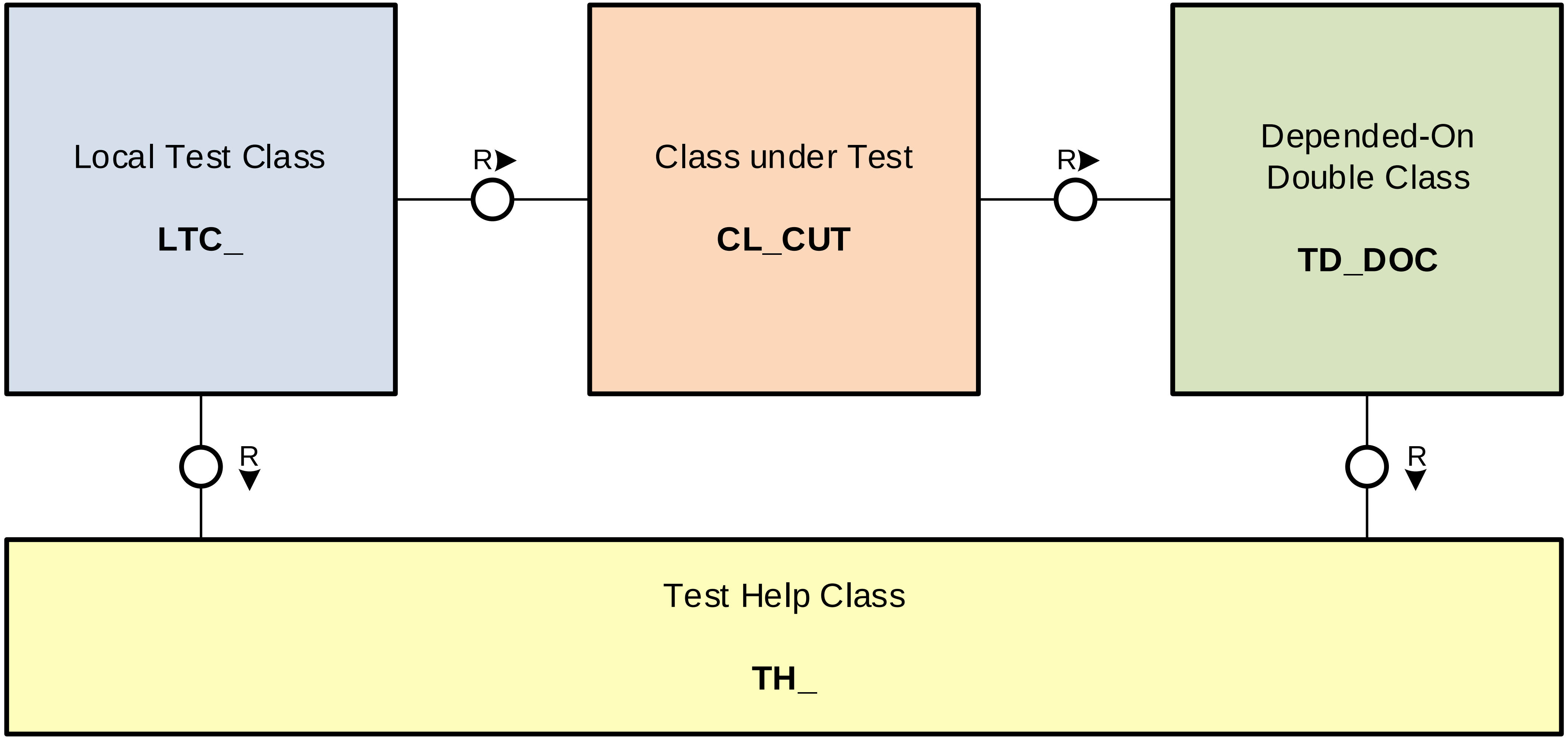


Using the New Class

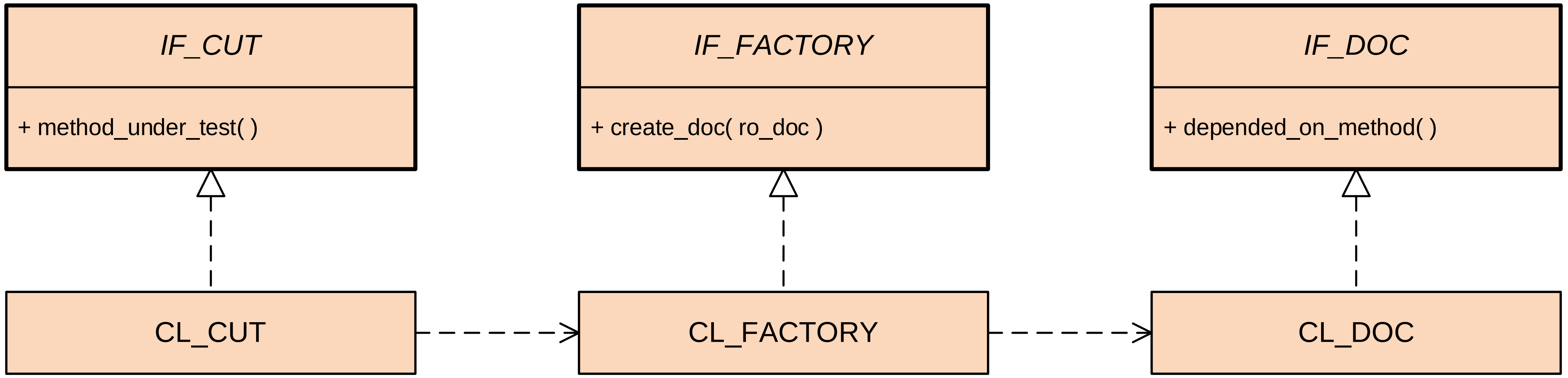
```
CLASS cl_product DEFINITION PUBLIC CREATE PUBLIC.
  PUBLIC SECTION.
    METHODS product_method.
ENDCLASS.


CLASS cl_product IMPLEMENTATION.
 METHOD product_method.
    …
    DATA(lo_new) = cl_factory=>get( )->create_new( ).
    lo_new->new_method( ).
    …
  ENDMETHOD.
ENDCLASS.
```

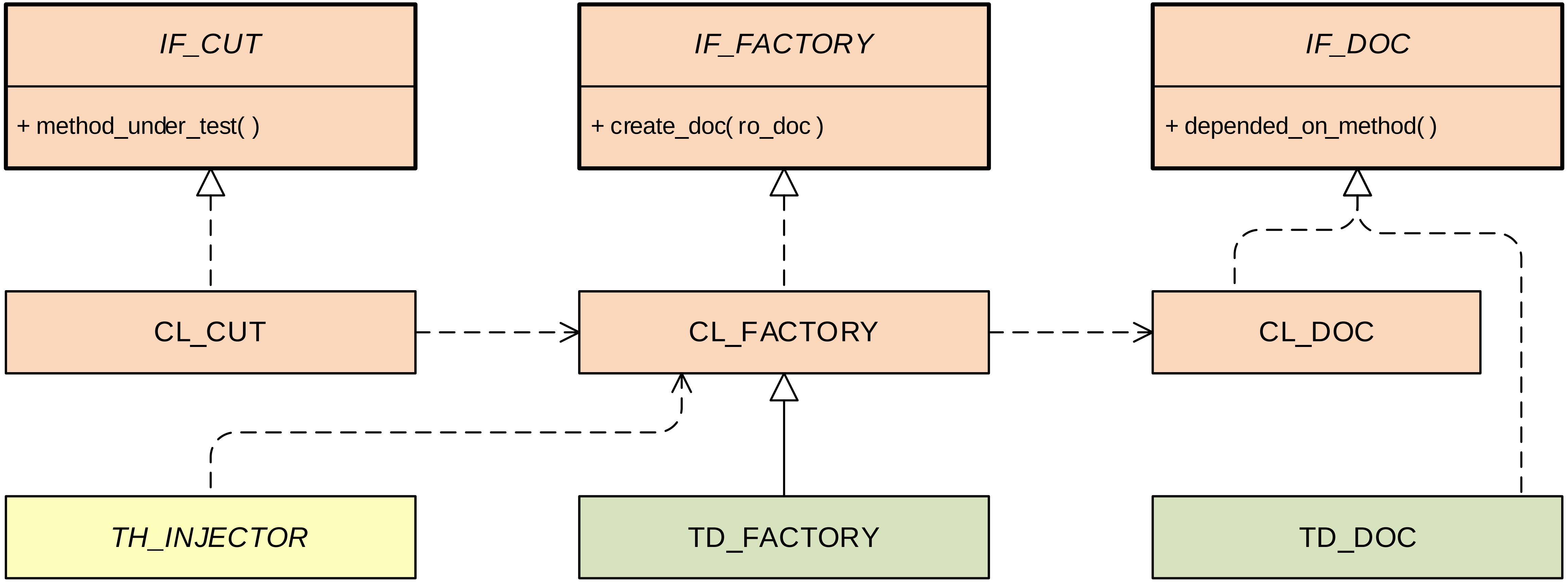
Test Abbreviations



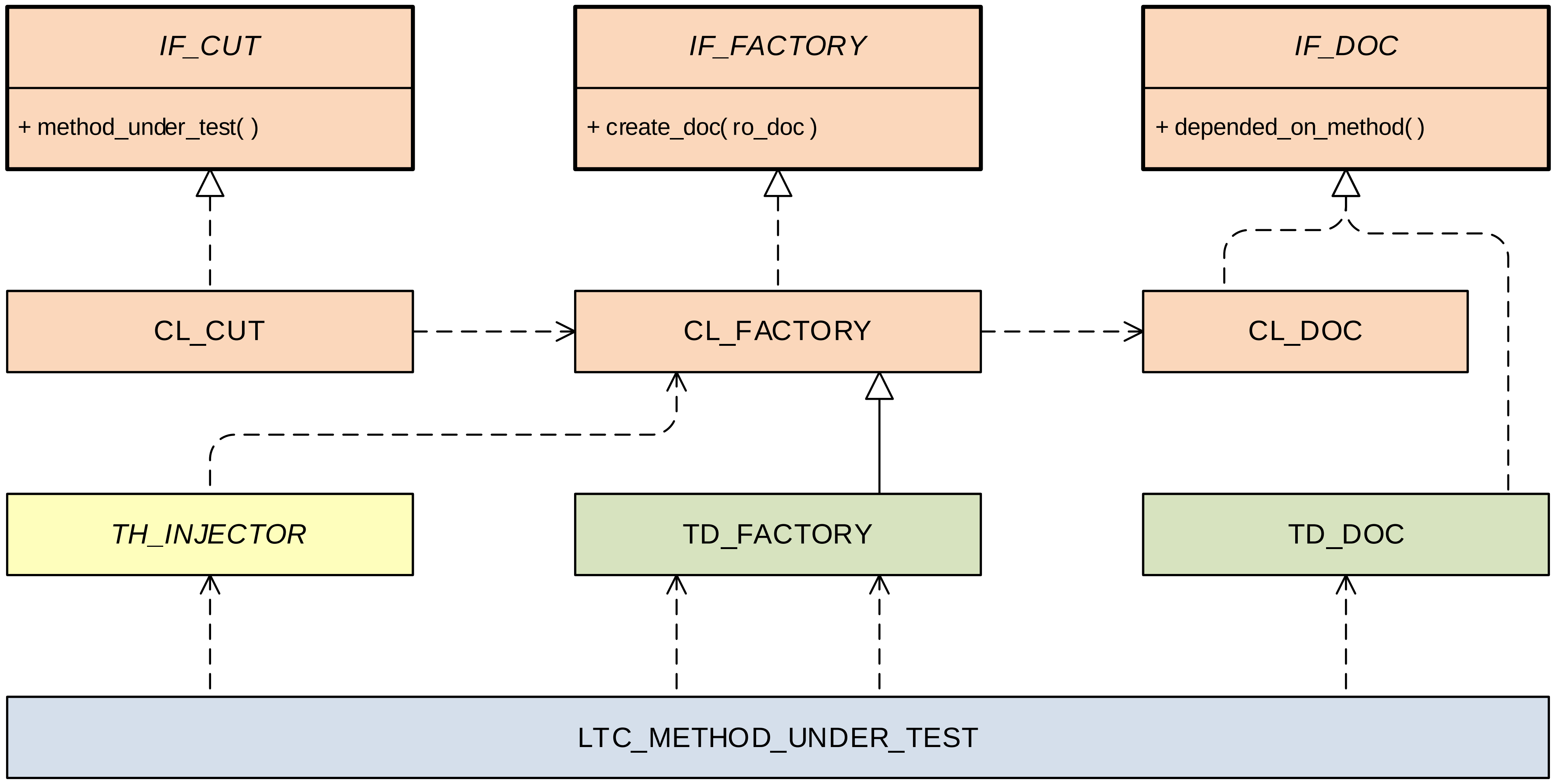
Clean Design: Product Classes



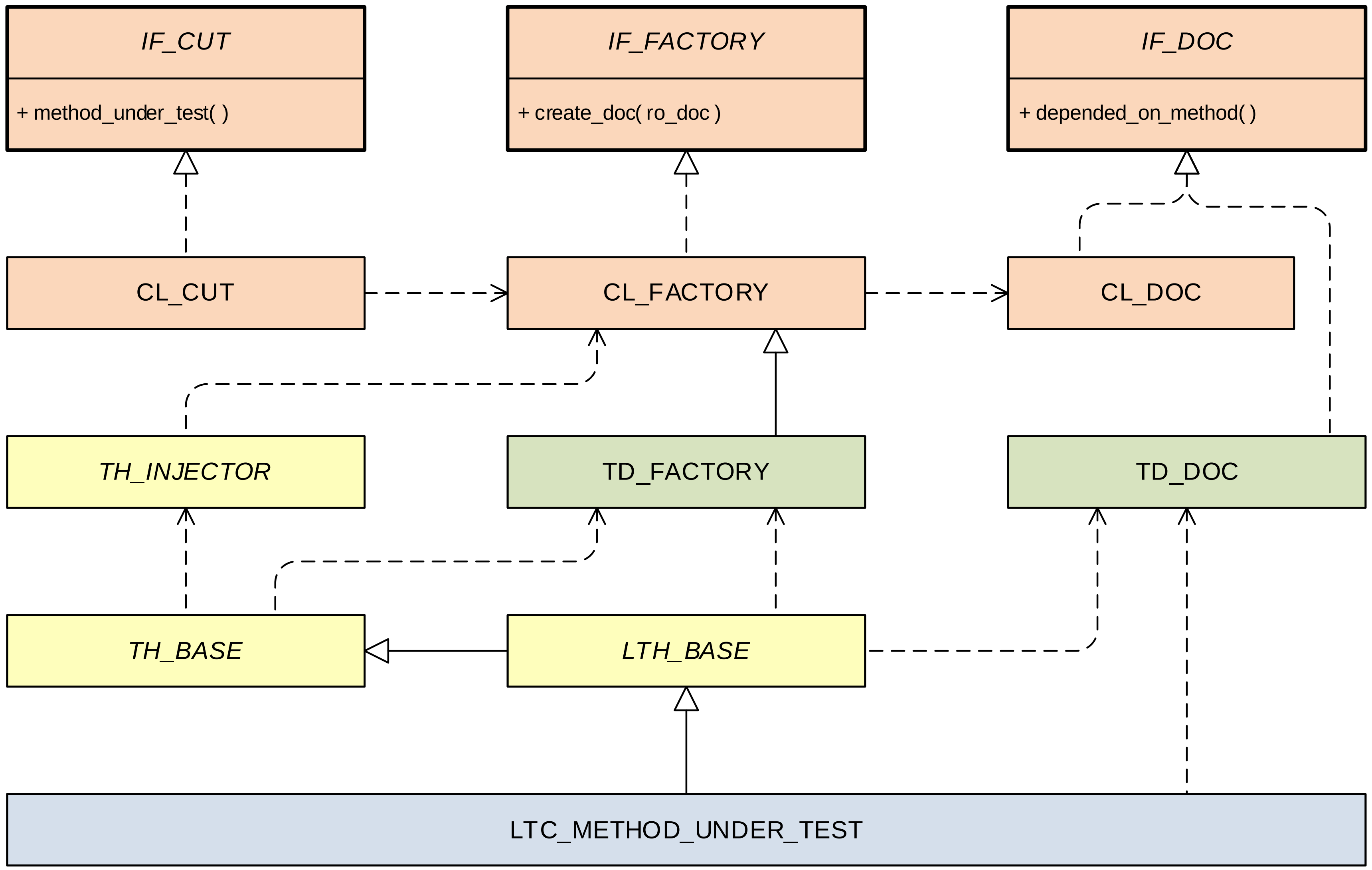
Clean Design: Double Classes



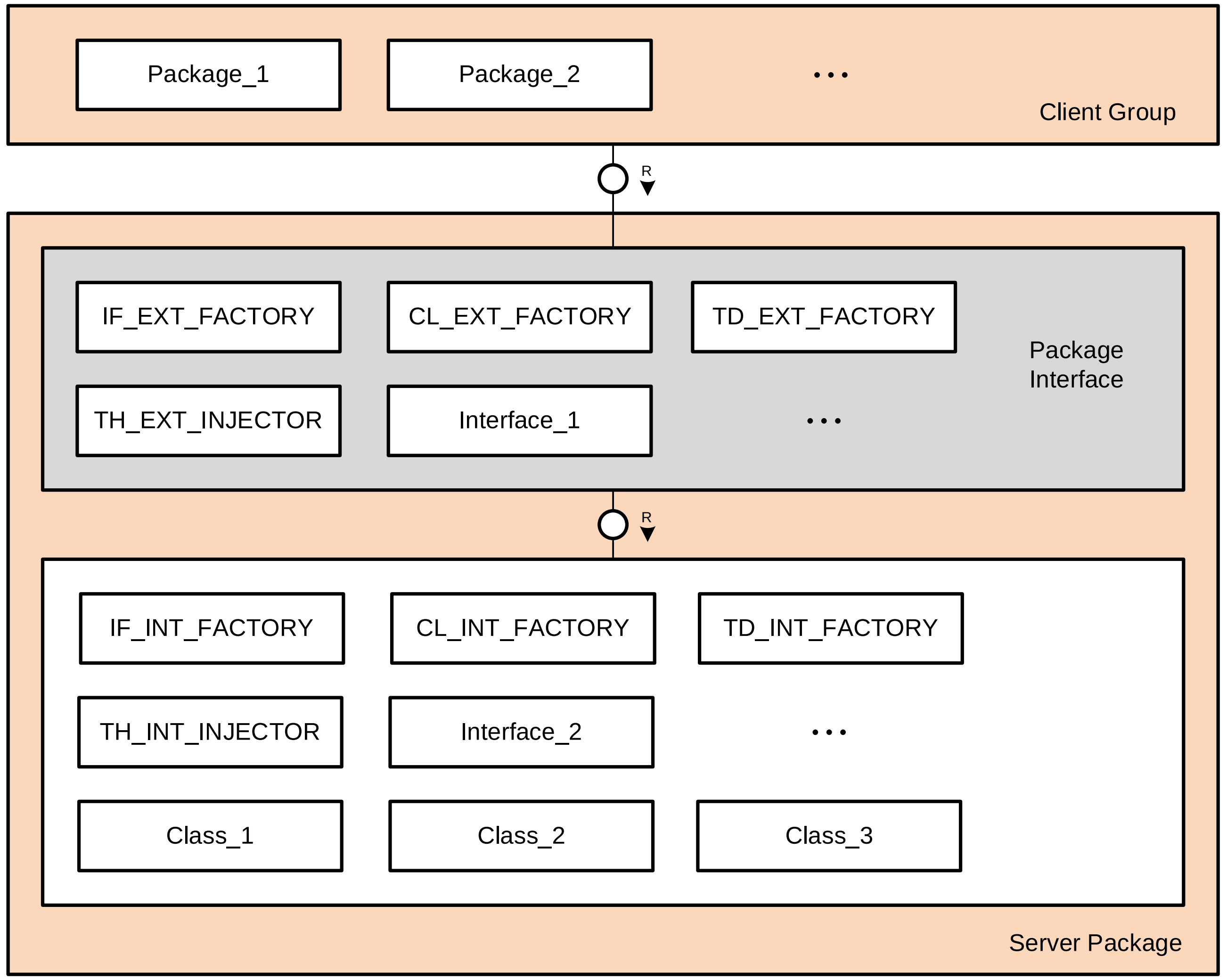
Clean Design: Test Classes



Clean Design: Test Base Classes



Clean Package



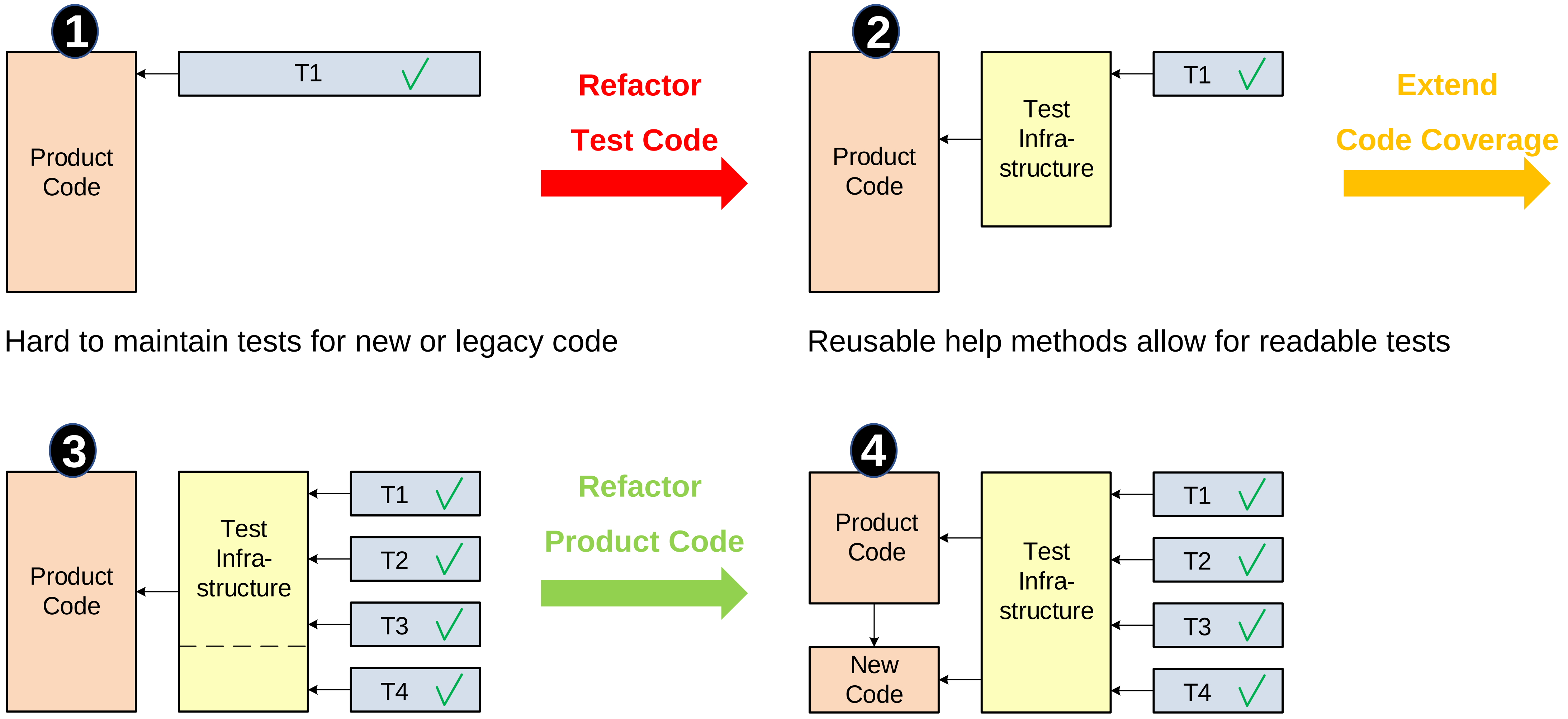
External factory provides access to the interfaces of the API Units.

Internal factory decouples encapsulated units.

Agenda

1. Test Pyramid
2. Test-Oriented Improvement Process
3. Clean Design
4. **Resources**

Test-Oriented Improvement Process



Hard to maintain tests for new or legacy code

Reusable help methods allow for readable tests

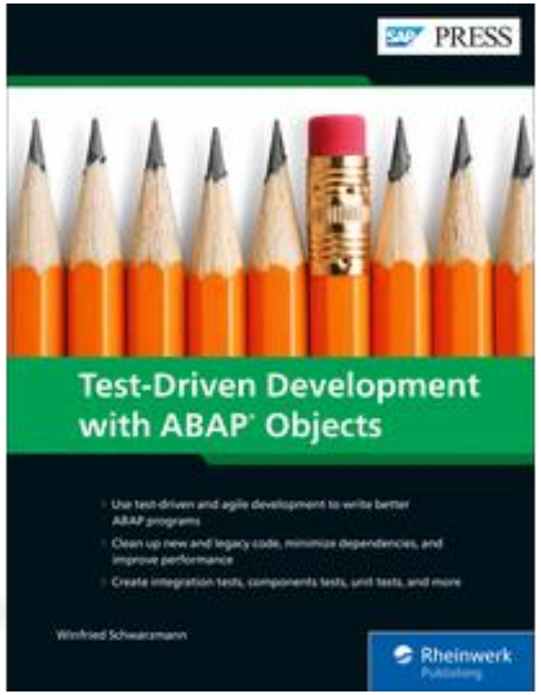
Test suite with effective tests

Test suite to reduce product code complexity

Test-Oriented Improvement Process (Reference)

SAP PRESS
Agile ABAP®-Entwicklung
Bewährte Methoden und Werkzeuge in praktischen Beispielen
Testgetriebene Entwicklung und Wartung von Legacy Code mit ABAP Unit
Regeln und Tipps für effizientes objektorientiertes Design
Winfried Schwarzmann
Rheinwerk Publishing

German edition
SAP Press, 2018


SAP PRESS
Test-Driven Development with ABAP® Objects
Use test-driven and agile development to write better ABAP programs
Clean up new and legacy code, minimize dependencies, and improve performance
Create integration tests, components tests, unit tests, and more
Winfried Schwarzmann
Rheinwerk Publishing

English edition
SAP Press, 2019

Content:

Part I:
Modernization of legacy code

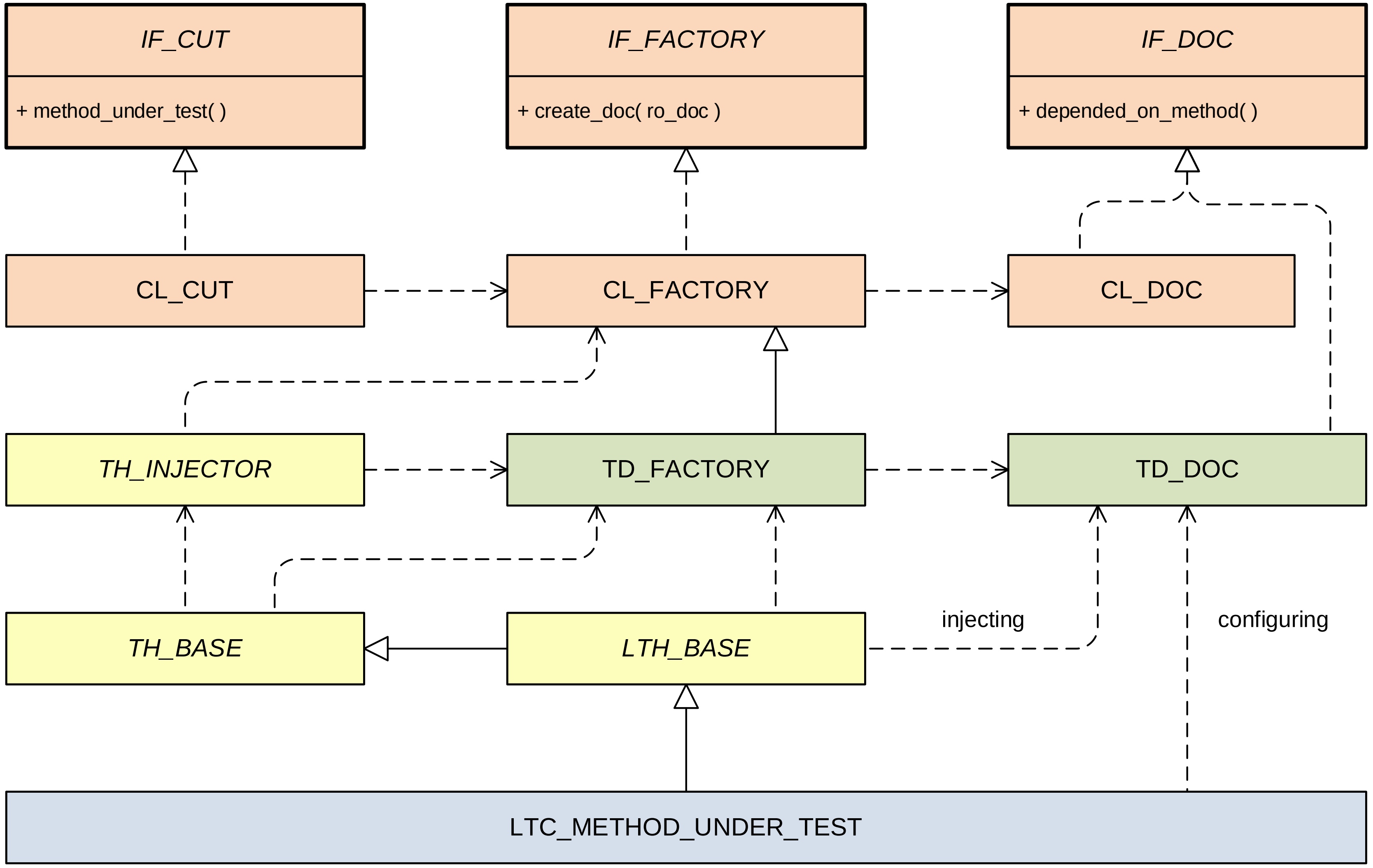
Part II:
Test infrastructure

Part III:
Test-driven development for new code

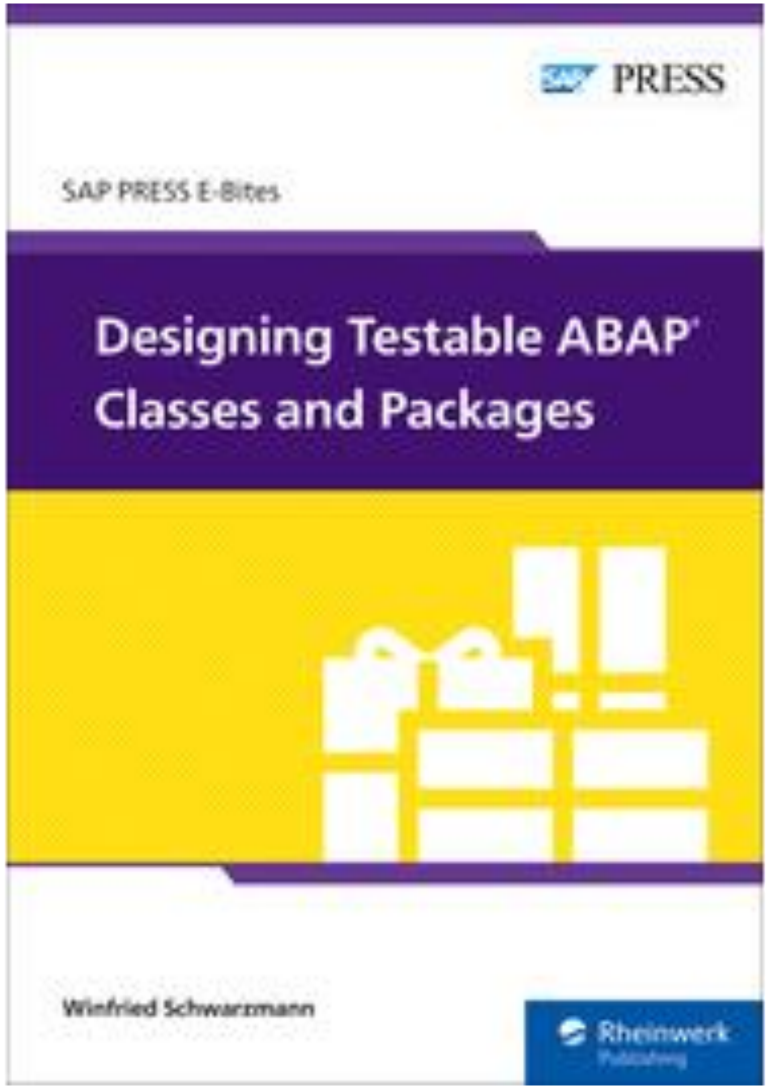
Part IV:
Agile software engineering

Part V:
Development & test tools

Clean Design



Clean Design (Reference)



E-Book
SAP Press
2022

<u>Content:</u>

Part I: Theory

- Classes
- Test Classes
- Packages
- BAdIs

Part II: Training

Exercises with solutions
for individuals and teams

Test Data Classes

All key and mandatory fields need to be defined.

```
DATA(lt_bsp_header) = VALUE t_bsp_header( (
    businesssolutionportfolio = '123'
    bussolnprtfloreference = '456'
) ).
so_environment->insert_test_data( lt_bsp_header ).

DATA(lt_bsp_item) = VALUE t_bsp_item( (
    object_id   = '123'
    number_int  = '100'
    objtype_h   = c_obj_type_service_contract
    currency    = 'EUR'
) (
    object_id   = '123'
    number_int  = '101'
    objtype_h   = c_obj_type_service_order
    currency    = 'EUR'
) ).
so_environment->insert_test_data( lt_bsp_item ).
```

any values

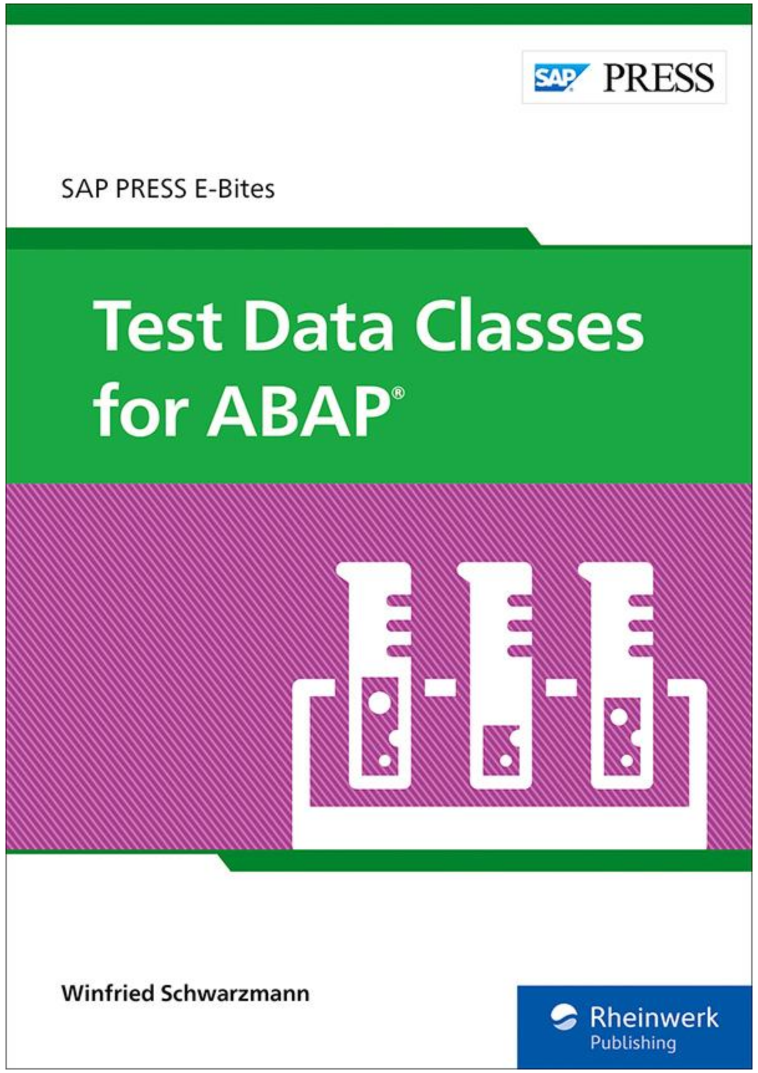
any values

crucial values

Only crucial values and relations are shown.

```
DATA(lo_header) = th_bsp=>create_any( ).
lo_header->add_item( th_item=>create_service_contract( ) ).
lo_header->add_item( th_item=>create_service_order( ) ).
lo_header->insert_fully_into( so_environment ).
```

Test Data Classes (Reference)



E-Book
SAP Press
2021

Content:

Part I: Theory

- Designing and implementing test data classes

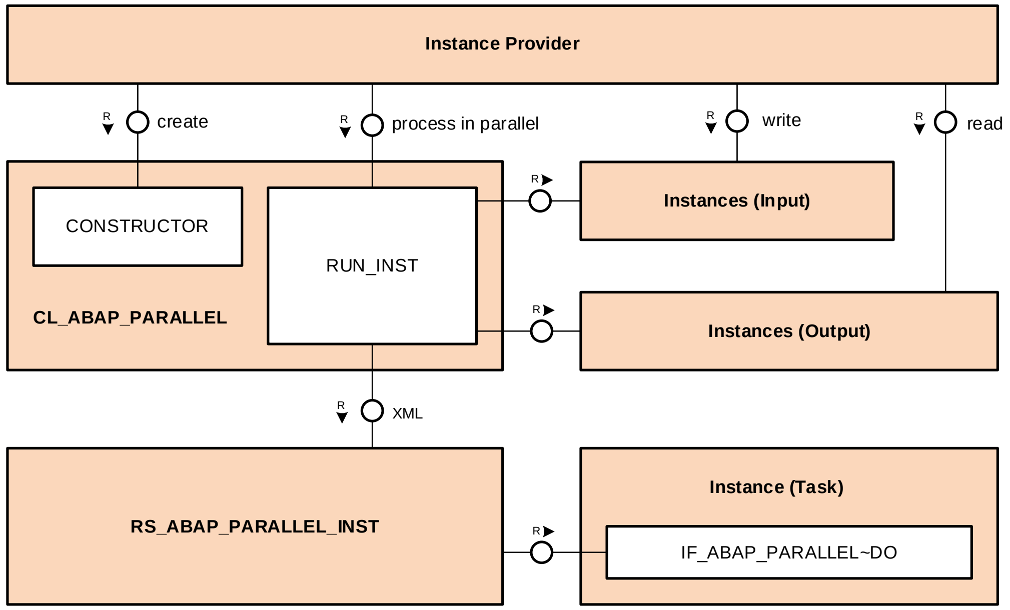
- Using test data classes for the entire test pyramid

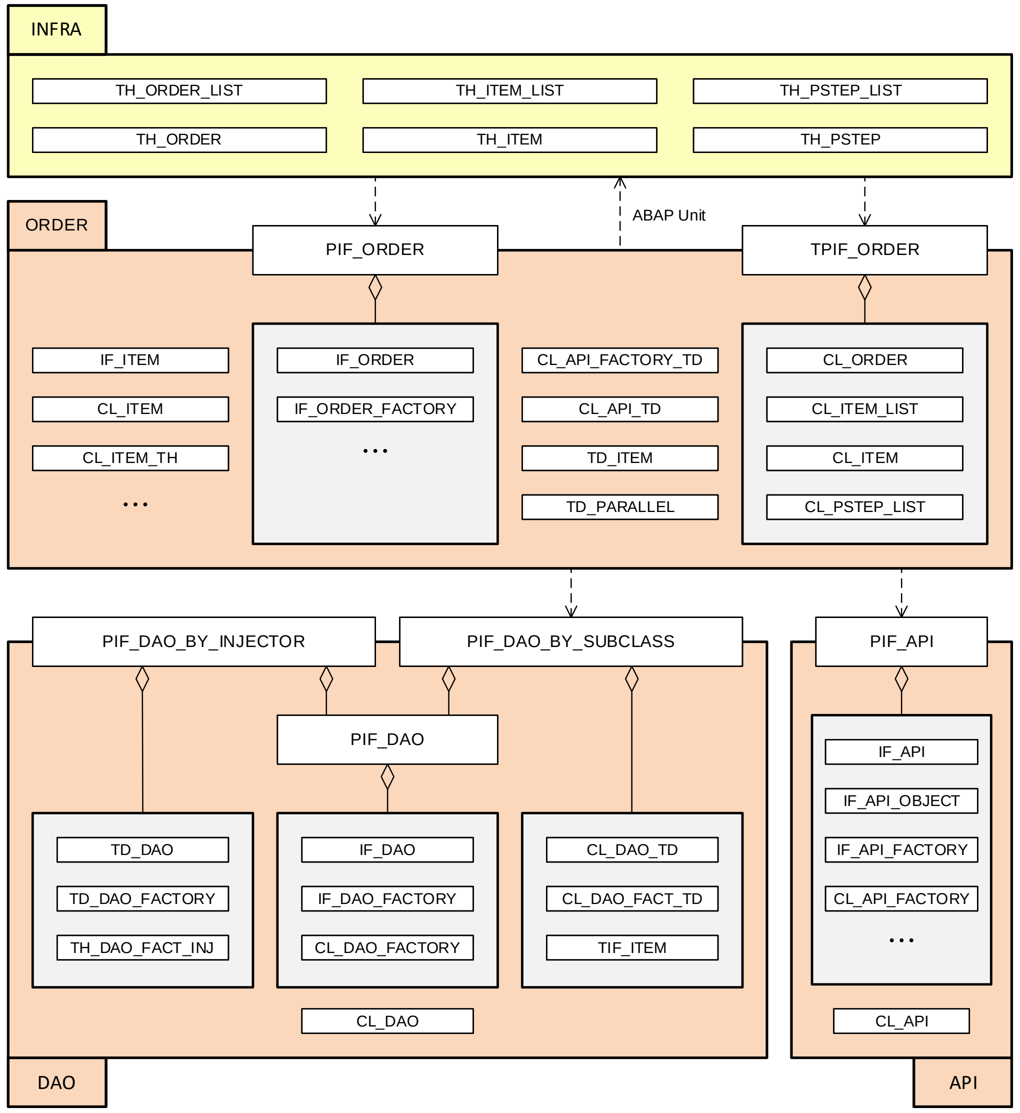
- Using test message classes for verifying error handling

Part II: Training

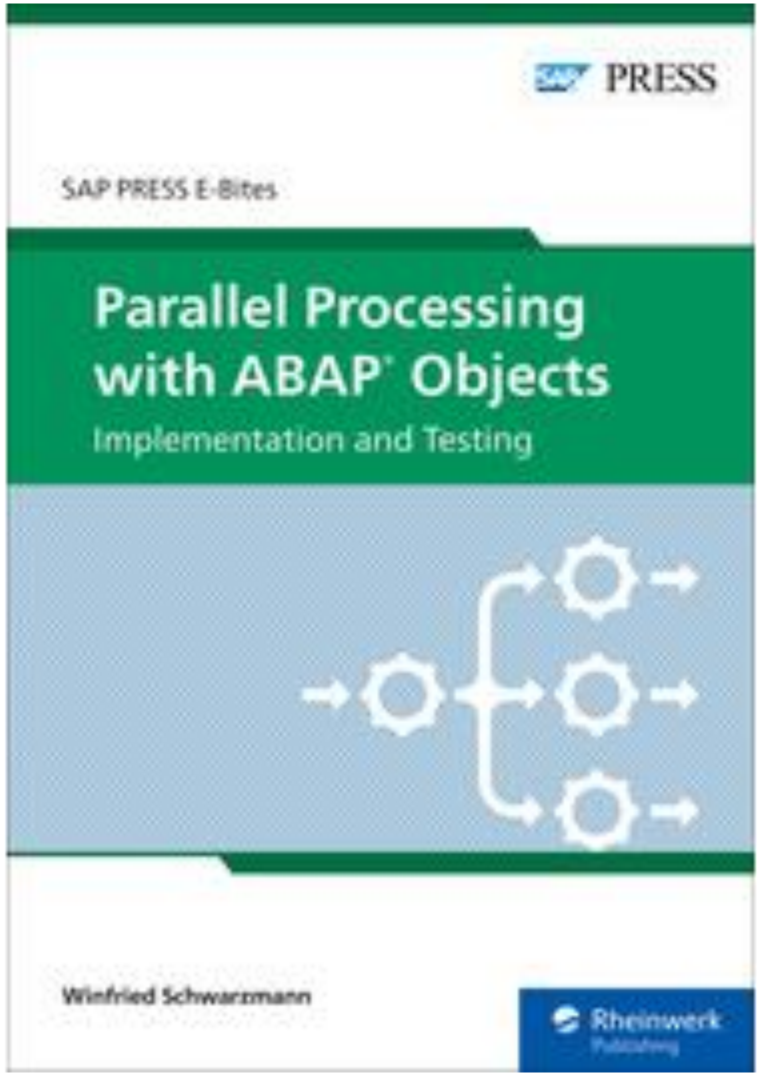
Exercises with solutions
for individuals and teams

Parallel Processing





Parallel Processing (Reference)



E-Book
SAP Press
2022

<u>Content:</u>

Implementing and testing parallel processes

Inheriting test data classes from productive data classes

Implementing, refactoring, and enhancing package design

Winfried Schwarzmann

Cloud ERP S/4HANA Architecture

E-Mail: winfried.schwarzmann@sap.com

Follow us

www.sap.com/contactsap

THE BEST RUN SAP®