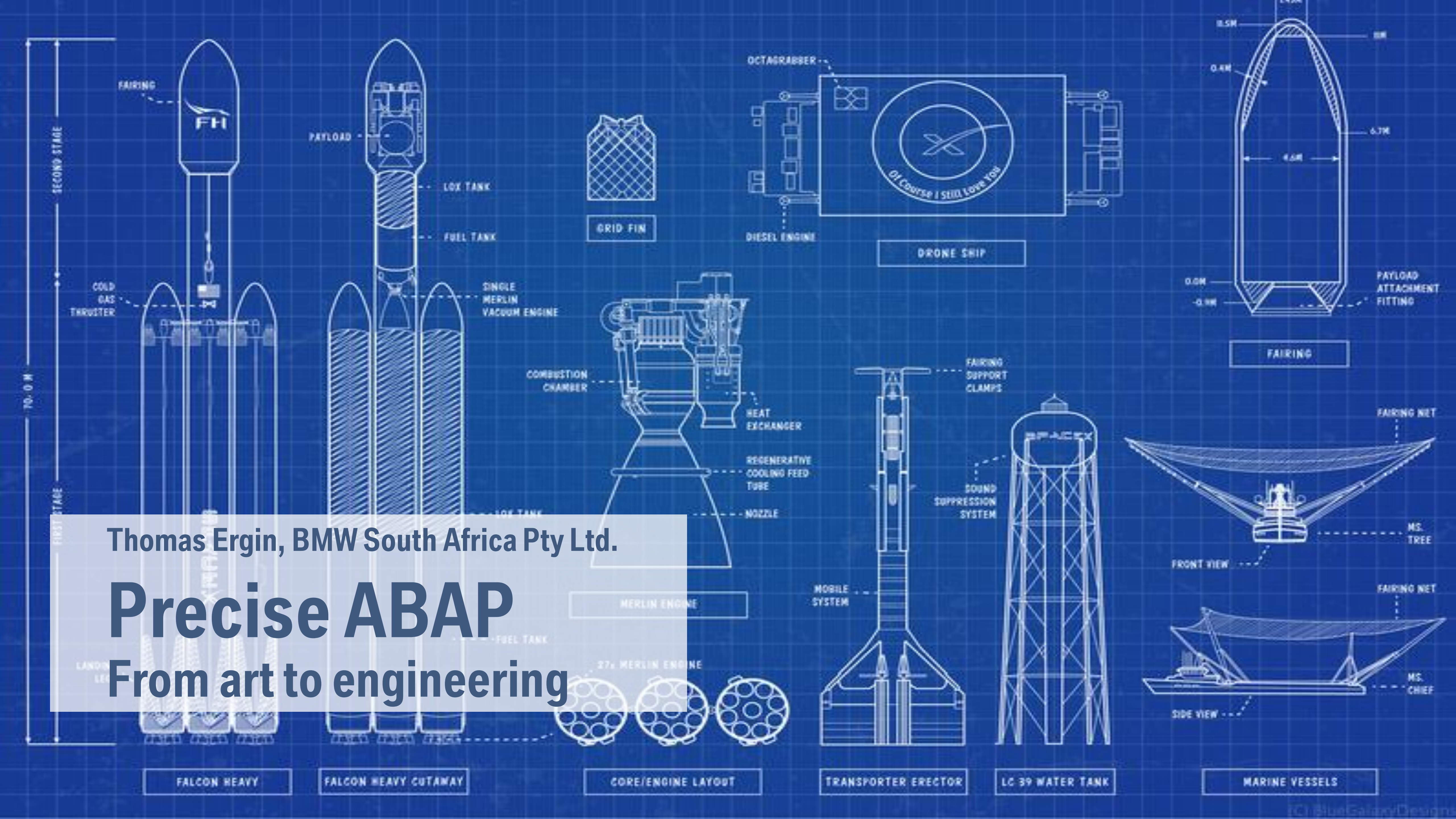


Thomas Ergin, BMW South Africa Pty Ltd.

Precise ABAP

From art to engineering

Clean Code tells a story…

```
DATA: xf_makwa(7) TYPE c.
DATA: xf_trp(1) TYPE c.
WRITE pw_afolg-matnr TO xf_objctkey.
xf_tab = 'MARA'.
xf_classnum = 'PW'.
CALL FUNCTION 'BAPI_OBJCL_GETDETAIL’
TABLES
ALLOCVALUESNUM = lt_ALLOCVALUESNUM
RETURN = lt_baret.
LOOP AT lt_ALLOCVALUESCHAR INTO
lw_ALLOCVALUESCHAR.
IF lw_ALLOCVALUESCHAR-charact EQ
'PW_GRADEINSTELLUNG1'.
WRITE lw_ALLOCVALUESCHAR-value_neutral TO
IF pw_afvu-slwid EQ 'ZPP0002'.
```

```
CHECK is_user_authorized( ).

todays_orders = get_sales_orders( today ).

DATA(preferences) = get_user_preferences( ).

SORT todays_orders BY (preferences-sorting).

display( todays_orders ).
```

Precise ABAP developers…

- Look beyond the SAP ecosystem
- Leverage existing library functions
- Avoid writing generic solution
- Design solutions with reusability in mind
- Do not maintain a "clean" and an "unclean" style
- Do not maintain a "performant" and a "default" style

Sequencing I – Fail fast!

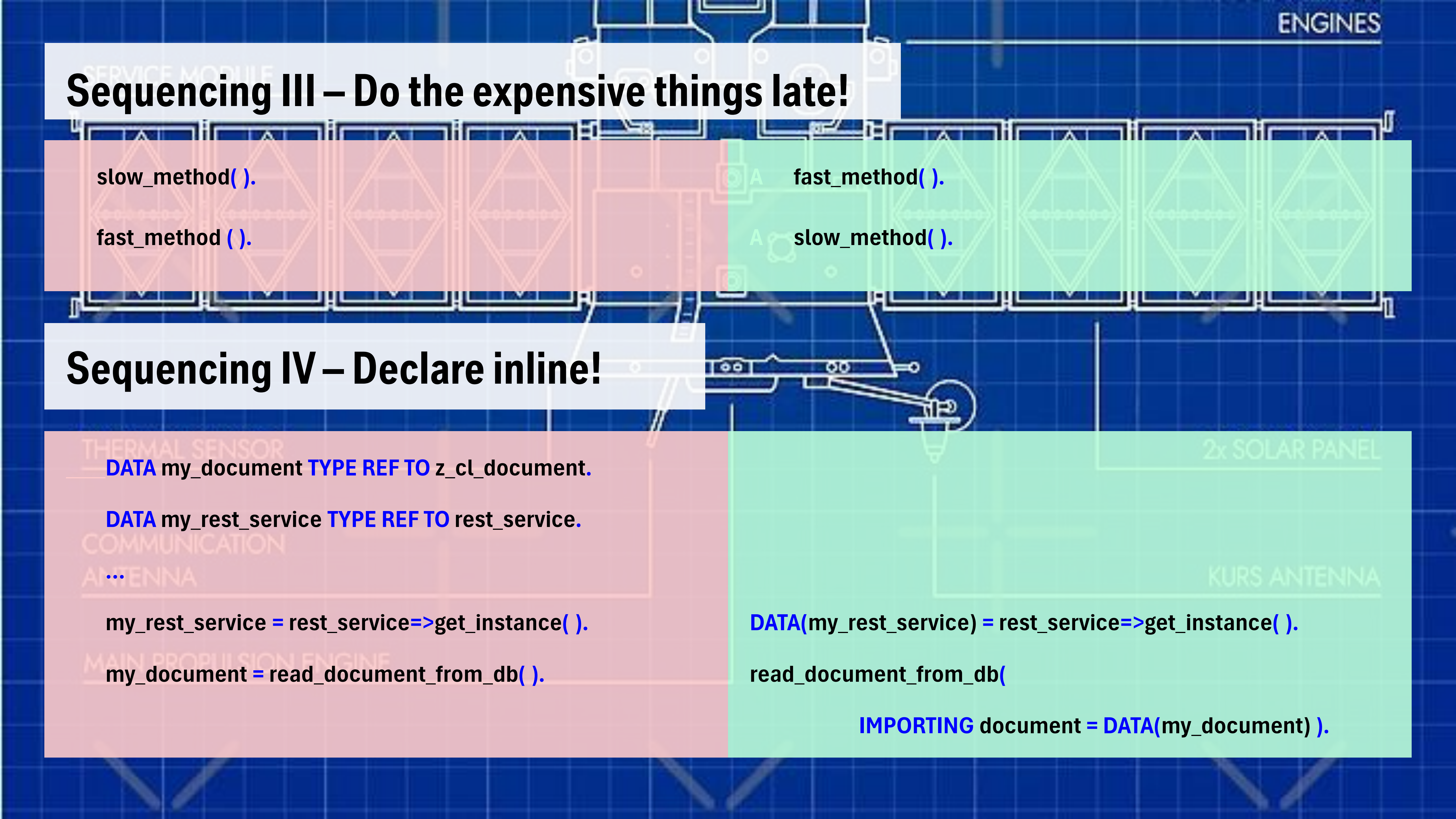
```
IF is_check_a_succesful( ).
    IF is_check_b_succesful( ).
      ...
      do_the_processing( ).
      ...
    ELSE.
      RAISE EXCEPTION TYPE cx_exception_b.
    ENDIF.
ELSE.
  RAISE EXCEPTION TYPE cx_exception_a.
ENDIF.
```

```
CASE abap_false.
  WHEN is_check_a_succesful( ).
    RAISE EXCEPTION TYPE cx_exception_a.
  WHEN is_check_b_succesful( ).
    RAISE EXCEPTION TYPE cx_exception_b.
ENDCASE.
...
do_the_processing( ).
...
```

Sequencing II – Check at method start!

```
METHOD perform_action.

...

ENDMETHOD.

...

IF is_valid( input ).
    perform_action( input ).
ENDIF.

...
```

```
METHOD perform_action.
  CHECK is_valid( input ).
...

ENDMETHOD.

...


perform_action( input ).

...
```



Sequencing III – Do the expensive things late!

```
slow_method( ).

fast_method ( ).
```

```
fast_method( ).

slow_method( ).
```

Sequencing IV – Declare inline!

```
DATA my_document TYPE REF TO z_cl_document.

DATA my_rest_service TYPE REF TO rest_service.

…

my_rest_service = rest_service=>get_instance( ).

my_document = read_document_from_db( ).
```

```
DATA(my_rest_service) = rest_service=>get_instance( ).

read_document_from_db(

            IMPORTING document = DATA(my_document) ).
```

Parameters – Keep the signature slim!

```
create_order(
    IMPORTING customer_order = DATA(customer_order)
    EXPORTING
        data = 'T. Ergin’
        item = 'Pencil’
        amount = 2 )
    CHANGING number_of_orders = number_of_orders.
```

```
DATA(customer_order) =  create_order(
  VALUE order(
      customer = 'T. Ergin'
      item = 'Pencil’
      amount = 2 ) ).

increase_number_of_orders( number_of_orders ).
```

Error handling I – Exceptions should stay exceptional!

```
IF retrieve_lock( ) = abap_false.

    RAISE EXCEPTION TYPE no_lock_exception.


ENDIF.
```

```
IF retrieve_lock( ) = abap_false.

    issue_message( ‘Can not lock…’ ).

    RETURN.

ENDIF.
```

Error handling II – Rather a dump than an undefined application state!

```
TRY.

   DATA(guid) = cl_system_uuid=>create_uuid_x16_static( ).

CATCH cx_uuid_error INTO DATA(guid_exception).

   "to be done

ENDTRY.
```

```
DATA(guid) = cl_system_uuid=>create_uuid_x16_static( ).
```

Error handling III – Enable the where-used-list!

```
issue_message( ‘Something went wrong’ ).
```

```
MESSAGE e000(FIRU_DCS) “Something went wrong
   INTO DATA(error_message).
issue_message(error_message ).
```

Error handling IV – Use specific messages

```
MESSAGE e001(O1) WITH ‘No authorization’. "&1 &2 &3 &4
```

```
MESSAGE e604(42). “No authorization
```

Resilience – Retry!

```
status = send_http_request( ).

IF status = not_send.
   ...
ENDIF.
```

```
WHILE status = not_send AND sy-index <= max_retries.
   status = send_http_request( ).
ENDDO.
IF status = not_send.
   ...
ENDIF.
```

Resilience – Timeout!

```
WHILE status = not_send.
   status = send_http_request( ).
ENDDO.
```

```
timeout = cl_abap_context_info=>get_system_time( ) + 5_seconds.
WHILE status = not_send AND cl_abap_context_info=>get_system_time( ) < timeout.
   status = send_http_request( ).
ENDDO.
```

Resilience – Circuit Breaker!

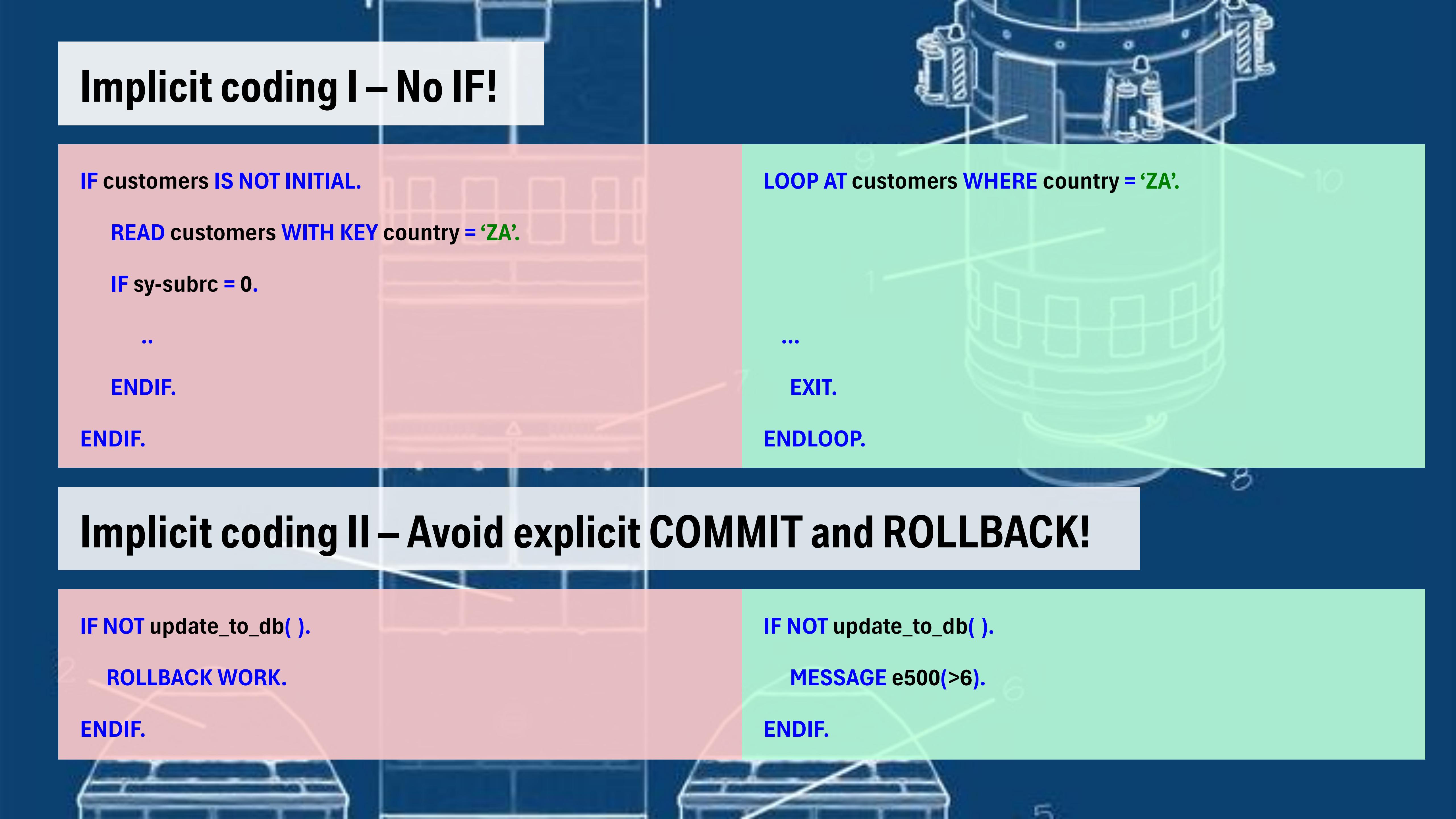
```
WHILE status = not_send.
    status = send_http_request( ).
ENDDO.
```

```
IF circuit_breaker->circuit_is_open( ).
    status = send_http_request( ).
ENDIF.
```

Resilience – Fallback!

```
DATA(user_language) = get_user_language( ).
IF user_language IS INITIAL.
   raise( ‘No default language set for user.’ ).
ENDIF.
```

```
DATA(user_language) = get_user_language( ).
IF user_language IS INITIAL.
   user_language = cl_wb2_const=>english.
ENDIF.
```



Implicit coding I – No IF!

```
IF customers IS NOT INITIAL.
  READ customers WITH KEY country = 'ZA'.
  IF sy-subrc = 0.
    ..
  ENDIF.
ENDIF.
```

```
LOOP AT customers WHERE country = 'ZA'.


 ...
  EXIT.
ENDLOOP.
```

Implicit coding II – Avoid explicit COMMIT and ROLLBACK!

```
IF NOT update_to_db( ).
  ROLLBACK WORK.
ENDIF.
```

```
IF NOT update_to_db( ).
  MESSAGE e500(>6).
ENDIF.
```

Implicit coding III – Don‘t CLEAR / UNASSIGN!

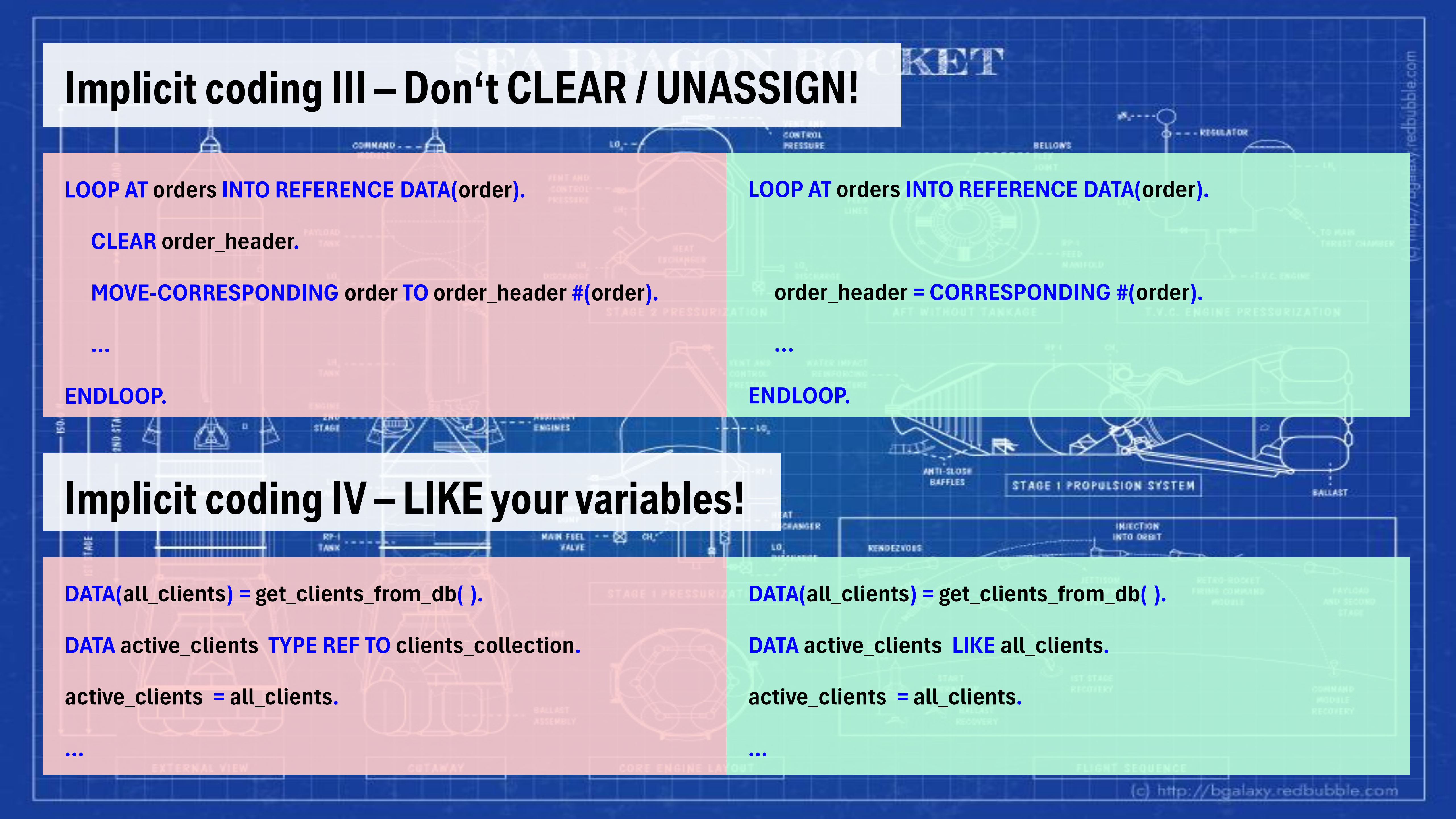
```
LOOP AT orders INTO REFERENCE DATA(order).
  CLEAR order_header.
  MOVE-CORRESPONDING order TO order_header #(order).
  ...
ENDLOOP.
```

```
LOOP AT orders INTO REFERENCE DATA(order).

  order_header = CORRESPONDING #(order).
  ...
ENDLOOP.
```

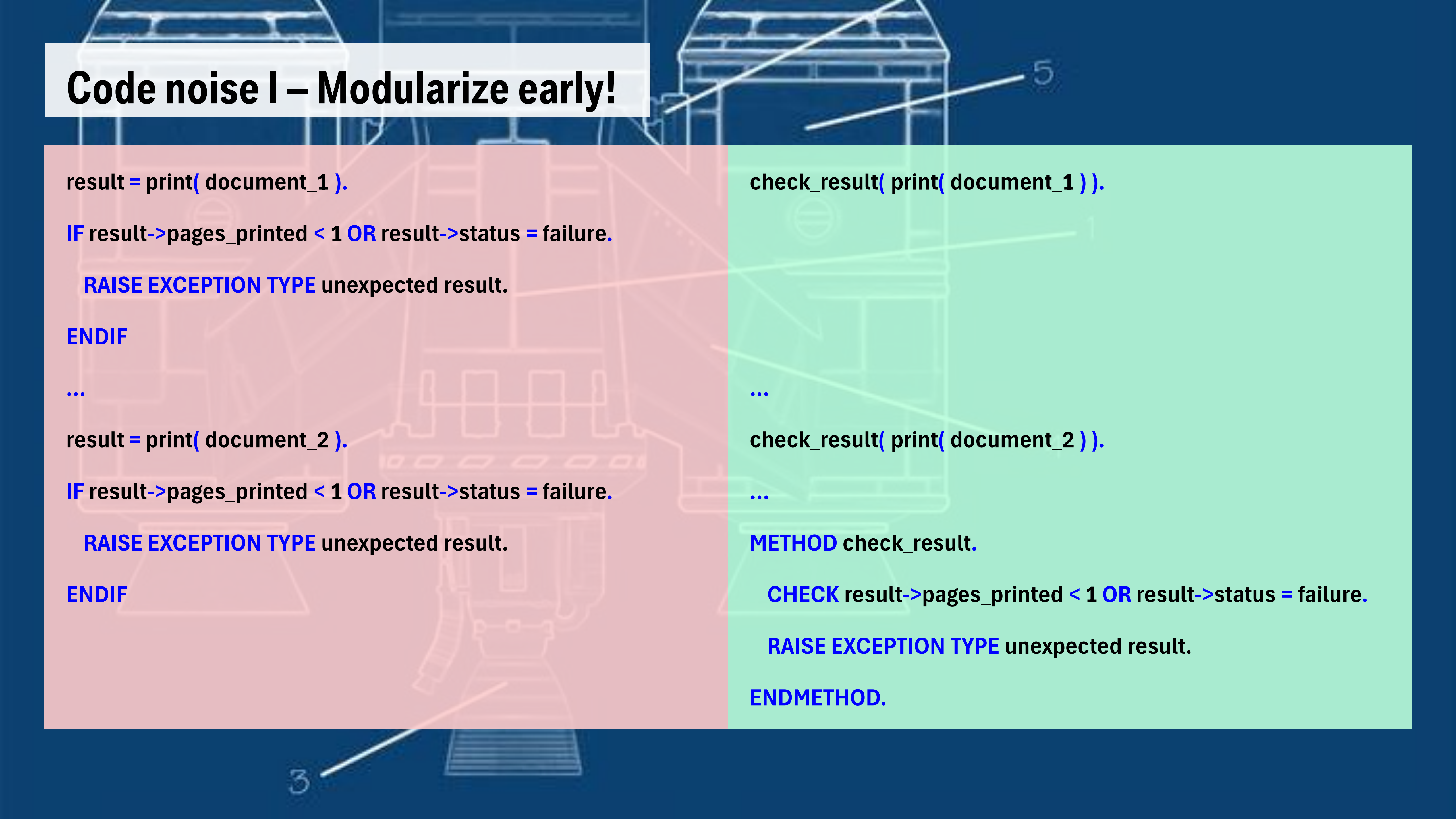
Implicit coding IV – LIKE your variables!

```
DATA(all_clients) = get_clients_from_db( ).
DATA active_clients  TYPE REF TO clients_collection.
active_clients  = all_clients.
...
```

```
DATA(all_clients) = get_clients_from_db( ).
DATA active_clients  LIKE all_clients.
active_clients  = all_clients.
...
```



Code noise I – Modularize early!



```
result = print( document_1 ).
IF result->pages_printed < 1 OR result->status = failure.
   RAISE EXCEPTION TYPE unexpected result.
ENDIF
...
result = print( document_2 ).
IF result->pages_printed < 1 OR result->status = failure.
   RAISE EXCEPTION TYPE unexpected result.
ENDIF
```

```
check_result( print( document_1 ) ).
...
check_result( print( document_2 ) ).
...
METHOD check_result.
   CHECK result->pages_printed < 1 OR result->status = failure.
   RAISE EXCEPTION TYPE unexpected result.
ENDMETHOD.
```

Code noise II – Do not repeat yourself!

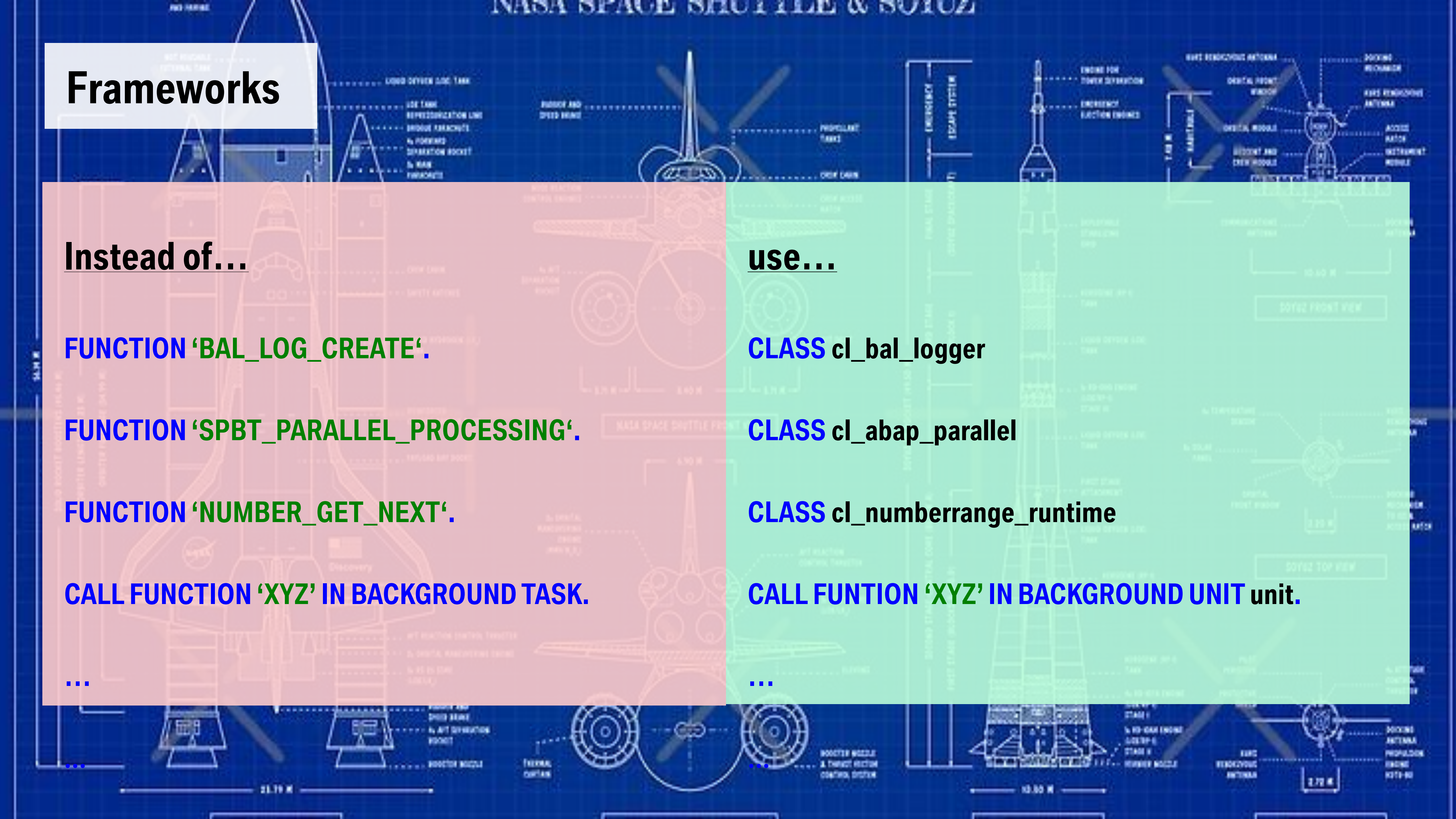
```
DATA(todays_orders) = read_orders(
    cl_abap_context_info=>get_system_date( ) ).
check_orders( todays_orders ).
print_orders( todays_orders ).
...
DATA(todays_orders) = read_orders(
    cl_abap_context_info=>get_system_date( ) – 1 ).
check_orders( yesterdays_orders ).
print_orders( yesterdays_orders ).
```

```
DATA(date) = cl_abap_context_info=>get_system_date( ) .
DO 2 TIMES.
    DATA(orders) = read_orders( date ).
    check_orders( orders ).
    print_orders(orders ).
    date -= 1.
ENDDO.
```

Code noise III – Focus on the function!

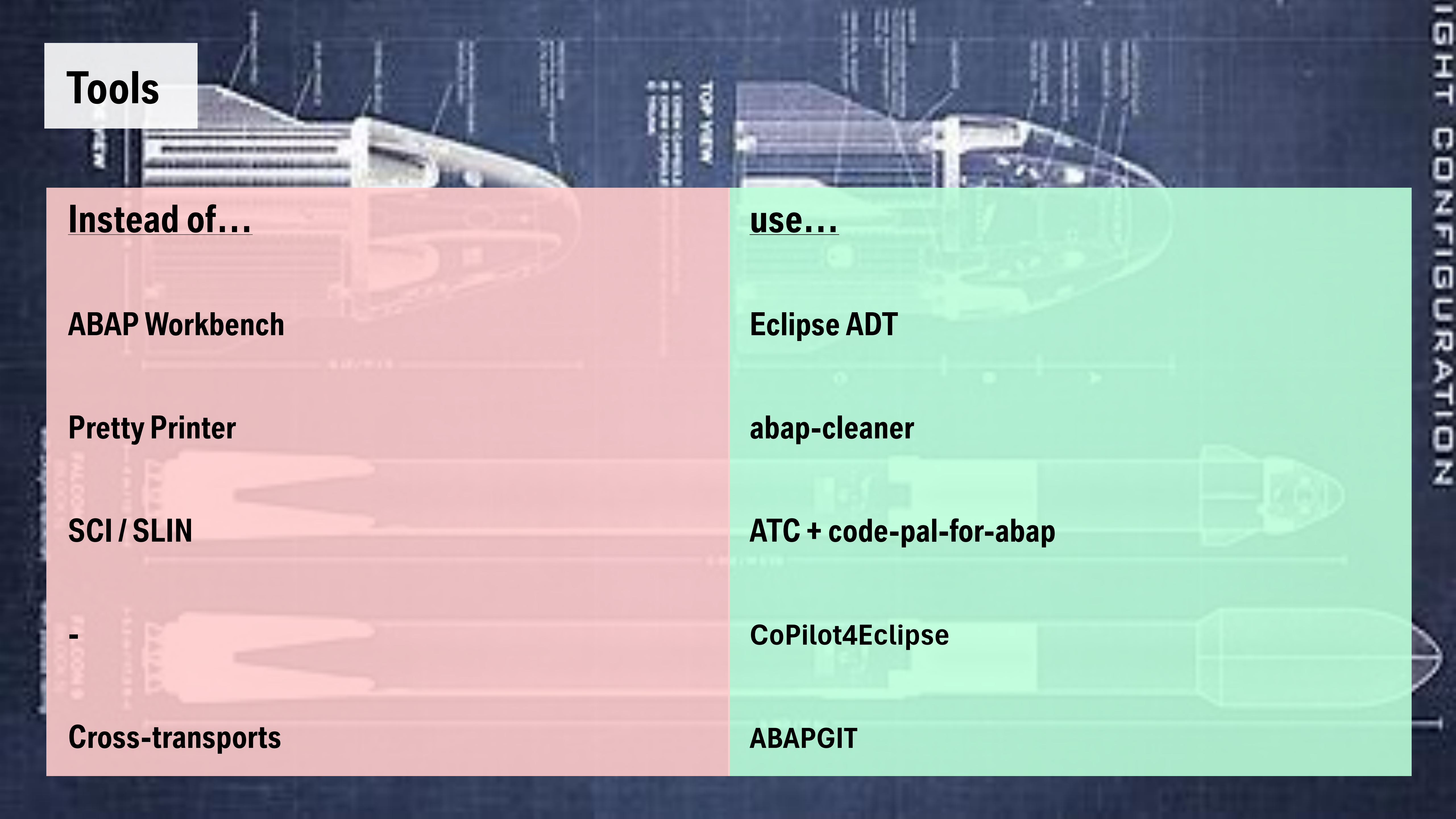
```
input-value  = 42.

input-meaning  = ‘Purpose of life’ .

input-source  = ‘The Hitchiker’s Guide to the Galaxy’ .

input-author  = ‘Douglas Adams’ .

result = do_action( input ).

CASE result.

   WHEN ok_status.

...

   WHEN failure_status.

...

   WHEN unkown_status.

ENDCASE.
```

```
input = setup_input( ).




result = do_action( input ).

handle_result( result ).


METHOD setup_input.

    input-value  = 42.

    input-meaning  = ‘Purpose of life’ .

    input-source  = ‘The Hitchiker’s Guide to the Galaxy’ .

...
```



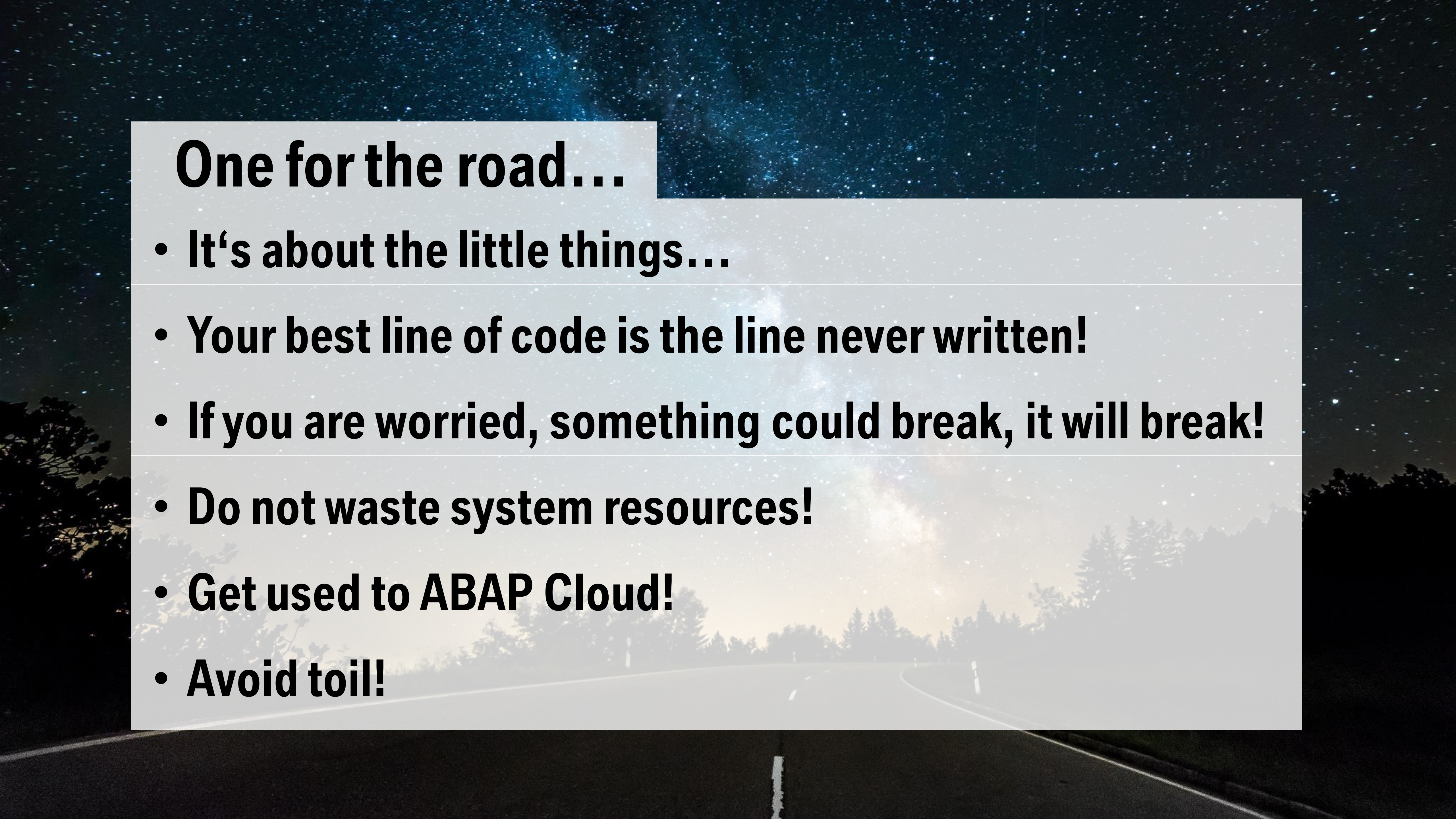
Frameworks

Instead of…	use…
FUNCTION ‘BAL_LOG_CREATE‘.	CLASS cl_bal_logger
FUNCTION ‘SPBT_PARALLEL_PROCESSING‘.	CLASS cl_abap_parallel
FUNCTION ‘NUMBER_GET_NEXT‘.	CLASS cl_numberrange_runtime
CALL FUNCTION ‘XYZ’ IN BACKGROUND TASK.	CALL FUNTION ‘XYZ’ IN BACKGROUND UNIT unit.
…	…



Tools

Instead of…	use…
ABAP Workbench	Eclipse ADT
Pretty Printer	abap-cleaner
SCI / SLIN	ATC + code-pal-for-abap
-	CoPilot4Eclipse
Cross-transports	ABAPGIT



One for the road…

- It‘s about the little things…
- Your best line of code is the line never written!
- If you are worried, something could break, it will break!
- Do not waste system resources!
- Get used to ABAP Cloud!
- Avoid toil!