"Automating Validation –
Or, How to Make Everyone's Life Easier!"

Presentation Overview

- ***Introduction***
  - What is validation?
  - Areas in pharmaceutical industry for validation work
- **Validation of standard programs and macro systems**
  - Preconditions
  - General approach of test case systems
  - Testscript and LOG examples
  - Advantages of test case systems
- **Validation of study programs**
  - Validation document at Bayer
  - Example of document

What is validation?

According to the Capability Maturity Model:

Validation: The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements.

Areas in pharmaceutical industry for validation work

- study-programs created to produce TLFs etc.
- standard programs or macros systems available for everyone

Presentation Overview

- **Introduction**
  - What is validation?
  - Areas in pharmaceutical industry for validation work
- ***Validation of standard programs and macro systems***
  - Preconditions
  - General approach of test case systems
  - Testscript and LOG examples
  - Advantages of test case systems
- **Validation of study programs**
  - Validation document at Bayer
  - Example of document

Validation of standard programs

- standard programs or systems have a life cycle with several main- and sub-versions
- only some aspects of the systems usually change - but everything must be validated for every (sub-)version to ensure that no side effects appear
- validation tasks are repeated every time
- test plans with several smaller tasks useful – converting of system changes to test plan changes are easier
- ideal situation: programs that run the created task automatically with an automatic comparison of results
- solution: Test Case Systems, e.g. Entimo Test Case System or SASUnit

Preconditions

- Specification : clearly defines the requirements that should be tested against and that must be satisfied
- Validation Plan : defines roles and responsibilities
- Validation Report : ideally - definition of all validation tasks
  usually – list of programs/macros to be tested with fields for outcome and signature

General approach of test case systems

- a combilation of macros for different tasks, e.g. for simple messages to the LOG or for more complex tasks like comparison of results
- tests are splitted into several test units
- naming conventions for outputs and exceptions for comparisons are defined in a global document for all testscripts
- one ini-file sets all libraries, macro variables and options for the test environment

Example Testscript

```
/*****************************************************************************************************************
New Test Unit
*****************************************************************************************************************/
%newtestunit (unit = 1, unit_text = 'Parameter Checks')
/*****************************************************************************
New Test Case
*****************************************************************************//**/
%newtestcase('##A','First example testcase')
%testdata;
%maccall;
%let macrocall = &macro(param1=test);
%expectresult();
%put {ERROR: Invalid value for param1 (test).};
%testresult();
%endtestcase(check4unexpectedLogMsg = Y);
/*****************************************************************************************************************
New Test Unit
*****************************************************************************************************************/
%newtestunit (unit = 2, unit_text = 'Usage checks')
/*****************************************************************************
New Test Case
*****************************************************************************//**/
%newtestcase('##A','Second example testcase')
```

Sub-macro details

%expectresults
- the user can define certain strings that must occure in the LOG, e.g. ERROR or WARNING messages
- the testcase systems automatically scans the LOG for specified messages and reports if at least one is missing

%compare_save_reference
- the user can either create reference data sets or compare those references automatically with the created output datasets
- the result of the comparison with the number of differences is printed to the LOG

%compare_outfiles
- can be used to compare lst-, txt-, pdf-,rtf- and ps-files
- if differences occur messages will be printed to the LOG

%dstestresult, %ext_file_rename
- SAS and non-SAS file(s) can be stored following globally defined naming rules

If one of the comparisons fails the testcase will be marked as UNSUCCESSFUL. The reason will be written to the testcase summary to get a quick overview of reasons for the unsuccessful execution.

Example LOG I – Parameter checks

```
********************************************************************************
*  Start of Test Case number 1_1
********************************************************************************
*  EXAMPLE_MACRO Unit 1 'Parameter Checks'
*  Test case 1
*  First example testcase
********************************************************************************
 Test data:
----------

Macro call:
-----------
example_macro(param1=test)

Expected result:
---------------
{ERROR: Invalid value for param1 (test).}

Test result:
------------
ERROR: Invalid value for param1 (test).

********************************************************************************
*  End of Test Case number 1_1
*  Automated check of test results (case insensitive): SUCCESSFUL
********************************************************************************
```

```
********************************************************************************:
*  Start of Test Case number 2_1
********************************************************************************:
*  EXAMPLE_MACRO Unit 2 'Usage checks'
*  Test case 1
*  Second example testcase
********************************************************************************:
Test data:
----------
Macro call:
-----------
example_macro(param1=Hello World!)

Expected result:
---------------
{Hallo World!}


Test result:
------------
Hello World!

********************************************************************************:
*  End of Test Case number 2_1
*  Automated check of test results (case insensitive): UNSUCCESSFUL!!!
*  Expected but not found (upcased):
     HALLO WORLD!
********************************************************************************:
```

Example LOG II – result comparison

```
*******************************************************************************
*  Start of Test Case number 2_2
*******************************************************************************
*  EXAMPLE_MACRO Unit 2 'Usage checks'
*  Test case 2
*  Third example testcase
*******************************************************************************
Test data:
----------
Macro call:
-----------
example_macro(param1=Hello World!,value=1)

Expected result:
---------------
Test result:
------------
Hello World!

Comparison of data sets work.ref_test_2_2 and work.test:
*** No differences
<see dataset TEST_2_2 in libref work>
*******************************************************************************
*  End of Test Case number 2_2
*  Automated check of test results (case insensitive): SUCCESSFUL
*******************************************************************************
```

```
*******************************************************************************
*  Start of Test Case number 2_3
*******************************************************************************
*  EXAMPLE_MACRO Unit 2 'Usage checks'
*  Test case 3
*  Fourth example testcase
*******************************************************************************
Test data:
----------
Macro call:
-----------
example_macro(param1=Hello World!,value=0)

Expected result:
---------------
Test result:
------------
Hello World!

Comparison of data sets work.ref_test_2_3 and work.test:
 Total Number of Values which Compare Unequal: 1.
*** Differences

<see dataset TEST_2_3 in libref work>
*******************************************************************************
*  End of Test Case number 2_3
*  Automated check of test results (case insensitive): UNSUCCESSFUL!!!
*  Expected but not found (upcased):
*  <Differences between Result and Reference Data; see above>
*******************************************************************************
```

Example LOG III – testscript summary

```
*******************************************************************************
* List of test cases (chronological order)
*******************************************************************************
* Test unit: 1 'Parameter Checks'
* with test cases:
*    1_1 First example testcase (automated check)
* Test unit: 2 'Usage checks'
* with test cases:
*    2_1 Second example testcase (automated check)
*    2_2 Third example testcase (automated check)
*    2_3 Fourth example testcase (automated check)
*******************************************************************************
* End list of test cases
*******************************************************************************

*******************************************************************************
* Test results
*******************************************************************************
* Following test cases (1) were executed SUCCESSFULLY (automated check)
*    1_1
*    2_2
* Following test cases (3) were executed UNSUCCESSFULLY (automated check)
*    2_1
*    2_3
*******************************************************************************
* End test results
*******************************************************************************
*******************************************************************************
* Change history of test cases
*    previous to current version
*******************************************************************************
* No changes from previous to current version
*******************************************************************************
* Remarks regarding deleted test cases see documentation
* in header of test case source
* End change history
*******************************************************************************

*******************************************************************************
* Stop EXAMPLE_MACRO test: 07JUN2012 5:55 AM
* Executed by: user as test by 2nd test user
* Log was stored in <path>/test.log
*******************************************************************************
```

This overview can be used as the validation documentation summary.

Advantages of test case systems

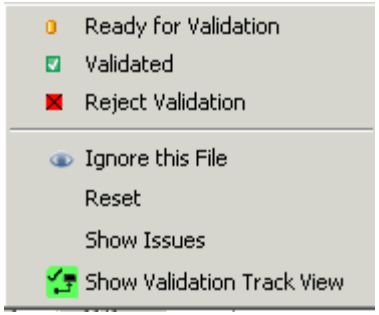
- old functionalities can be easily re-validated
- easy to retrace old validation runs
- maintenance (which requires much more effort that original development) can be done way faster
- test roles can be changed
- test users need less knowledge about the system to be validated
- overview of successful/unsuccessful test cases
- main advantage: lot of the work is done automatically by the system

Presentation Overview

- **Introduction**
  - What is validation?
  - Areas in pharmaceutical industry for validation work
- **Validation of standard programs and macro systems**
  - Preconditions
  - General approach of test case systems
  - Testscript and LOG examples
  - Advantages of test case systems
- **Validation of study programs**
  - Validation document at Bayer
  - Example of document

Validation of study programs at Bayer

- a validation document with all relevant programs and macros needs to be created before every productive run
- that document can be automatically produced by a system macro
- additionally all info about author and reviewer can be derived from system files that hold all stati changes for every program
- status changes can be set via a functionality in the development environment or are automatically set by the system, e.g. if a validated program is changed again



Output of macro for study program validation

Validation Tracking Report

☐ initial document
☐ subsequent documentation (due to corrections, additional requests)

Program Specifications reference

- <path>\test-project\test-study\SAP\<specification.doc>
- alternative scenario: see header of program
- future place: link to Documentum system

Survey of single program validation

Validation levels:
A — Developer validation
B — Reviewer or reference validation
C — Independent double program validation
D — Standard program validation, see also additional validation forms

Program Folder : <path>/pgms

File	Last Saved Date and Time	Validation level	Author	Checked	Reviewer / Reference	Checked	Comments
example_macro.sas	31MAY2012 15:46	B	evmhw (Martin Hoffmann)	2012-05-30	abcde (Test User)	2012-05-31	
example_macro2.sas	31MAY2012 15:46	C	evmhw (Martin Hoffmann)	2012-05-29		.	Reviewer has to be a different Person than the Author