Standards and Evolution

Lessons Learned on Quality (of) Standards

Rainer Gerlich, Ralf Gerlich (BSSE)

DASIA'11

17.-20.05.2011, Malta

Dr. Rainer Gerlich BSSE System and Software Engineering
Auf dem Ruhbühl 181
88090 Immenstaad
Germany

Tel. +49/7545/91.12.58
Fax +49/7545/91.12.40
Mobil +49/171/80.20.659
email Rainer.Gerlich@bsse.biz

Talk is on Software Standards

We are talking about software standards!

We won‘t talk about hardware standards
like
analog/digital IO‘s on a board
or
sizes of connectors and boxes!

We will talk about potential non-compliances with the standards as a matter of evolution!

Focus is put on Unit Testing

Quality

degree to which a set of inherent characteristics fulfills requirements

ISO 9000
ECSS Glossary of Terms)
ECSS P-001, 3.160

100% -
degree to which requirements could not be fulfilled

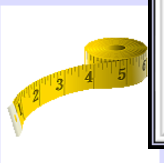
Any deviation from fulfillment is called

a fault

100% -
degree to which faults could not be removed

Fault-centric approach

- dormant faults
- sporadic faults
- non-anticipated faults
- *Last but not least* anticipated faults



towards higher fault identification sensitivity

Our Motivation – Higher Efficiency in Testing

BSSE System and Software Engineering

We are successful with current standards!

At what cost?

Goals

More test stimuli per €

Robustness, stress testing

Higher Code Coverage per €

Higher Fault Coverage per €

Means

A significant higher throughput can be achieved by full automation of the test cycle "human fully out of the test loop"

Fully automated test cycle = Test case identifcation by an automaton

10+ test cases and 1000+ stimuli per minute on average

at max. test coverage (block /statement + MC/DC)

incl. test drivers + test report, statistical evaluation

alternative, complementary fault detection methods for unit testing

More than comparing test vectors to the specification and more than coverage analysis!

Why an automaton is needed …

BSSE System and Software Engineering

```
void myFunc(int il, int iu)
{
   int ii;
   if ((iu-il)>100)
       return;
   for (ii=il;ii<iu;ii++)
       ;
   return;
}
```

What is the WCET when the execution of a loop cycle takes 1 ms?

100 ms?

It is about 1.3 years! (4294967295 ms)

Why?

Example:

il =-2147483648

iu = 2147483647

iu-il =-1

iu-il>100=false

Found by an (unbiased) automaton

We can learn more than just to take this exotic test case!

It is about the difference between theory and practice!

The automaton identified a non-anticipated fault!

Pure test coverage would not have required to find this test case!

Who would have found this fault due to requirements-based testing?

For those who may ask

We will put the focus on unit testing here.
But
similar results are available for
fully automated model-driven testing!

Testing Requirements – Discussion ECSS vs. Full-Auto-Test Cycle

Deriving proposals for test cases by an automaton is much more efficient
millions of test stimuli can be generated per hour on a PC

ECSS

E-40, 4.2.6 (5)
This process can include a test readiness review (TRR) to verify that all test facilities and test cases and procedures are available before each significant test campaign, and under configuration control.

E-40, 5.5.3.2
The supplier shall develop and document the test procedures and data for testing each software unit.
The supplier shall test each software unit ensuring that it satisfies its requirements and document the test results.

E-40, 5.5.2.9
The supplier shall define and document ..., test design and test case specification for testing software units

E-40, 5.6.3.1
The supplier shall develop and document, for each requirement of the software item in TS (including ICD), a set of tests, test cases (inputs, outputs, test criteria) and test procedures including: [...]

Fully Automated Test Cycle

Requires test-cases to be provided before test.

Automated cycle requires test generation and execution before comparison with specification.

What shall be documented in case of

millions of test stimuli?

Suggests that test cases are to be (manually) derived from a specification

implying manual setup of test environment

BUT as quality goal it is sufficient:
test vectors shall be compared with the specification

overspecification implies technology dependence

Testing Requirements – Discussion DO178B vs. FullyAuto-Test Cycle

DO178-B

6.3.6.b
The objective is to verify that the test cases were accurately developed into test procedures and expected results.

6.4.2
Requirements-based testing is emphasized because this strategy has been found to be the most effective at revealing errors

6.4.4.2
The requirements-based test cases may not have completely exercised the code structure, so structural coverage analysis is performed and additional verification produced to provide structural coverage.

6.4.4.3
Structural coverage analysis may reveal code structure that was not exercised during testing. Resolution would require additional software verification process activity

Fully Automated Test Cycle

The test setup is generated automatically.

Test vectors are automatically derived.

Test cases must be derived from the specification

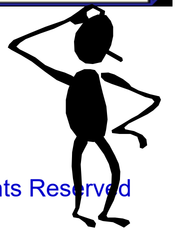
No evidence given for effectiveness assumption

Suggests that test cases are (manually) derived from the specification (top-down)

implying manual setup of test environment

Admits that specification-based testing may not be sufficient to achieve full coverage.

Systematic auto-generation of test vectors (bottom-up) will mitigate this issue.



Are deviations for improvements really allowed?

Will I get an OK from PA before or at the end?

Fault Identification - Platforms

To detect (true) faults everything is allowed.
Non-representative platforms may increase fault visibility.

ECSS + ISVV

Q-80, 7.3.6.a
Where the components developed for reuse are developed to be reusable on different platforms, the testing of the software shall be performed on all those platforms.

E-ST-10-02, 5.2.2.1.c
Verification of software shall include testing in the target hardware environment

ECSS requires target-test, but no other environments. DO178-B declares target „excellent“ environment, suggesting that others are inferior.

Fully Automated Test Cycle

Dormant faults can efficiently be detected on a platform other than the target platform.

Pre-condition for efficiency:
auto-porting to another platform

DO178-B

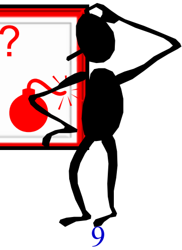
6.3.1.c
The objective is to ensure that no conflicts exist between the high-level requirements and the hardware/software features of the target computer, especially, system response times and input/output hardware.

6.4
To verify correct operation of the software in the target computer environment.

6.4.1
More than one test environment may be needed to satisfy the objectives for software testing. An excellent test environment includes the software loaded into the target computer and tested In a high fidelity simulation of the target computer environment.

6.4.1
Selected tests should be performed in the integrated target computer environment, since some errors are only detected in this environment.

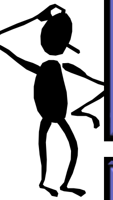
Are deviations for improvements really allowed?
Will I get an OK from PA before or at the end?



Independence

We are going a step ahead!
We introduce metrics to assess the benefit of another tool!

BSSE System and Software Engineering



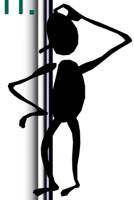
Does supplier and design independence imply complementary or equivalent fault identification capabilities?

Which faults can be detected at all?

No metrics on independence!

The fault identification capabilities should be known. Tools should be selected to be complementary and/or equivalent in a deterministic manner.

The resulting toolset should fully cover known fault types at least.



Can I proceed this way?

ECSS + ISVV

Q-80, 5.6.1.1.a
Methods and tools to be used […](including […] validation, testing, […]) shall be identified by the supplier and agreed by the customer

ISVV + IEEE 1012
For software tools, technical independence means that the IV&V effort uses or develops its own set of test and analysis tools separate from the developer's tools

Q-80, 5.6.1.2.a
The choice of development methods and tools shall be justified by demonstrating through testing or documented assessment that:[…]
2. the tools and methods are appropriate for the functional and operational characteristics of the product,

Q-80, 5.6.1.3.a
The correct use of methods and tools shall be verified and reported.

DO178-B

4.4.1.b
The use of qualified tools or combinations of tools and parts of the software development environment should be chosen to achieve the necessary level of confidence that an error introduced by one part would be detected by another. An acceptable environment is produced when both parts are consistently used together

12.3.3.4
Each tool is to be obtained from a different developer
Tool designs have to be dissimilar

12.2.2
The qualification criteria for software verification tools should be achieved by demonstration that the tool complies with its Tool Operational Requirements under normal operational conditions.

12.2.3.2
Tool Operational Requirements describe the tool's operational functionality. This data should include:
A description of the tool's functions and technical features.

Summary

Evolution may have a chance, but .

process compliance means safe harbour

driver is process compliance not product quality

Standards difficult to waive and to invest in resources for more efficient methods and tools

standards define quasi-state-of-the-art for PA, not less, but also **not more** in quality

Examples discussed

If the presented cases could be covered, what is the process? Is an official statement possible within a short-term perspective?

- extension of the curre eneration of test environment and test cases
- proper selection of tools for to g ve nd ISVV
- proper selection of test platforms

Guidance towards higher efficiency

Focus on efficiency of methods and tools guided by measurable product quality

Benchmarking of standards

metrics need to be applied allowing to derive efficiency figures in terms of fault identification sensitivity and effort

Backup

Views

We are talking about software standards!

Focus

of current standards

Process Quality

Quality = Process Compliance

Metrics on Process Compliance

Spec-based Testing only

Compromise Quality vs. Feasibility

Product Quality

Efficiency = Quality / €

Metrics on Product Quality

Systematic Fault Identification

Evolution Making More Feasible

Not in Focus

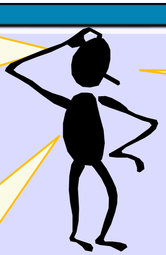
but our focus

Does process-quality imply product-quality?

Standards and the development process

We want to drive the process towards better control of quality

BSSE System and Software Engineering



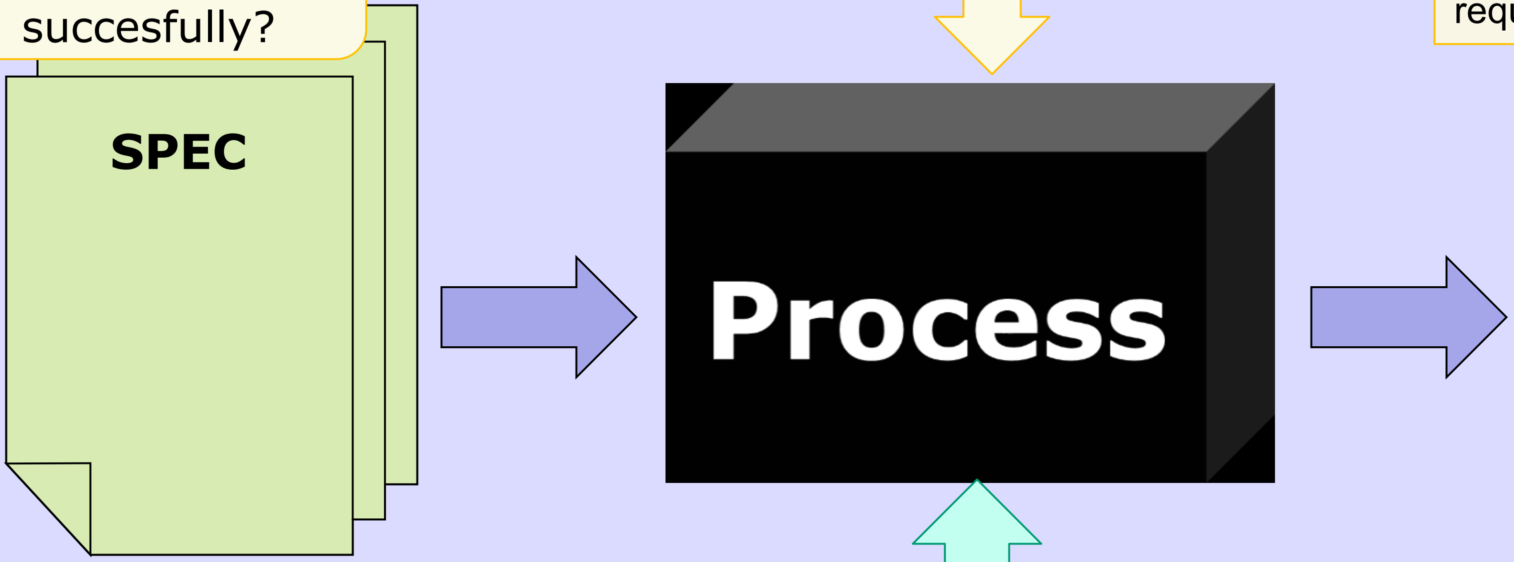
Did I document my test-cases?

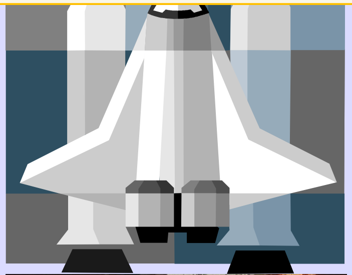
Is my code sufficenty readable?

Was my code reviewed succesfully?

Space standards
Level of quality does not inherently follow from process compliance.
Inspection of every product still required

Process-centric goal:
Focus on process execution

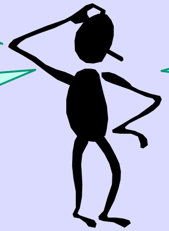




Property-centric goal:
Focus on product properties

Mass production
process fully ensures high quality,
no reviews on products or process

Is my V&V environment sufficient to detect all faults?



Does my process sufficiently guide to avoid faults (early enough)?

Does my process sufficiently guide to find faults and to fix them?

Standards vs. Innovation and Evolution

Technology Evolution

at the beginning: standards are rather challenging

Standards call for innovation:

high demands, nearly impossible to fulfill

after a while: technology has made progress

Standards block or stifle innovation:
Industry has invested in existing standard
Standards compliance is „safe harbour“
new tech. vs standard + "established practices“

Quality Assessment

(implicit) assumption: process compliance is main driver of product quality

Lack of justification for this supposed correlation

Even with all the specialisations allowed by an (ECSS) process the desired quality and efficiency might be unreachable (⇒ budget overruns, failures)

There may be more efficient processes which are not in the set of allowed instantiations

Metrics for decision?

A lot of obstacles do exist, preventing benefit from innovation!

Our Motivation –
Risk Reduction and Fault Identification

Our Goals

Ensure coverage of all known fault types in V&V

Provide guidance on how to detect unknown/hiding fault types more easily

Test right from beginning of coding
when test effort is sufficiently low

Apply massive test stimulation (millions of stimuli), statistics

Fault Identification

Earlier Detection of Weaknesses (robustness, testability)

Stress Testing and Robustness Testing

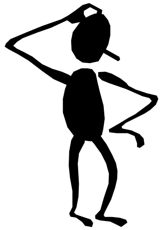
Current Standards

No guidance on how to detect all known fault types, they are not even classified

Apply some tool, this already complies with standards not knowing what is really supported

Unit Testing already during coding is not current practice possibly due to the high effort required for manual test preparation

Suggested manual approach implies poor test throughput, poor evaluation of robustness, statistical insignificance



Are deviations for improvements really allowed?

Will I get an OK from PA before or at the end?

What is the process for evolution?