More Important than Techniques

Michael Stahl, Intel

SIGiST 2012

(c) Michael Stahl, 2012, All Rights Reserved

Seriously…

How to design a Test?

(c) Michael Stahl, 2012, All Rights Reserved

IEEE 829-1998

A test case specification shall have the following structure:

a) Test case specification identifier;
b) Test items;
c) Input specifications;
d) Output specifications;
e) Environmental needs;
f) Special procedural requirements;
g) Intercase dependencies.

(c) Michael Stahl, 2012, All Rights Reserved

IEEE 829-2008

Level Test Case Outline (full example)

1. Introduction (once per document)

1.1. Document identifier

1.2. Scope

1.3. References

1.4. Context

1.5. Notation for description

2. Details (once per test case)

2.1. Test case identifier

2.2. Objective

2.3. Inputs

2.4. Outcome(s)

2.5. Environmental needs

2.6. Special procedural requirements

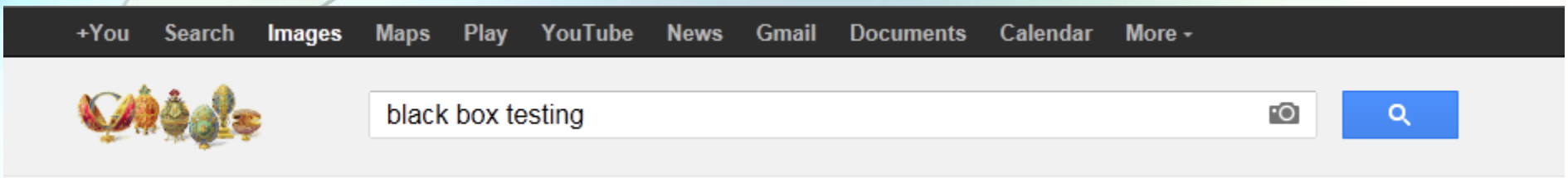
2.7. Intercase dependencies

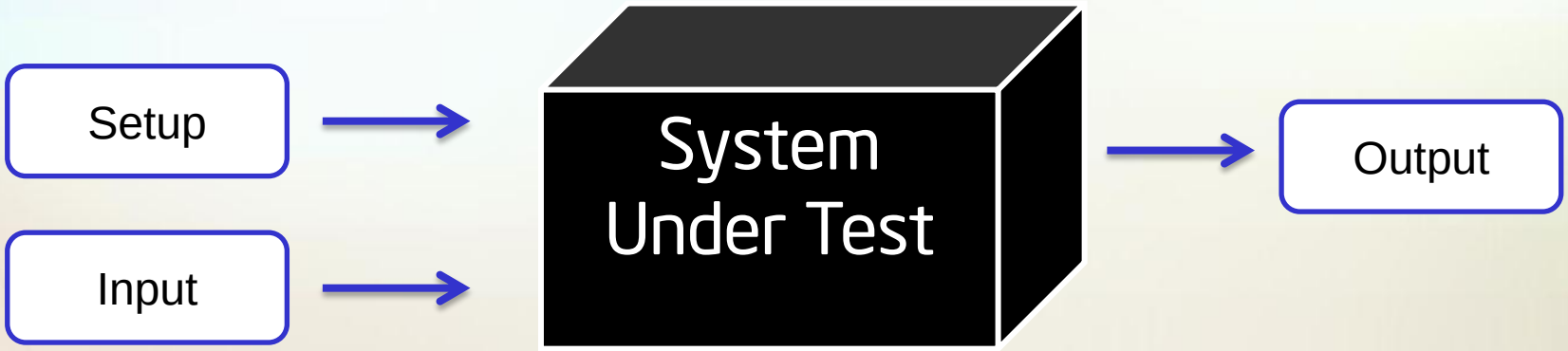
3. Global (once per document)

3.1. Glossary

3.2. Document change procedures and history

(c) Michael Stahl, 2012, All Rights Reserved





(c) Michael Stahl, 2012, All Rights Reserved

The ISTQB Test Techniques

- Equivalence Partitioning
- Boundary Values Analysis
- Decision Table Testing
- State Transition Testing
- Use Case Testing
- Experience-based Testing
- Etc. etc.

Source: ISTQB, Foundation level syllabus

(c) Michael Stahl, 2012, All Rights Reserved

The "standard" Test Design

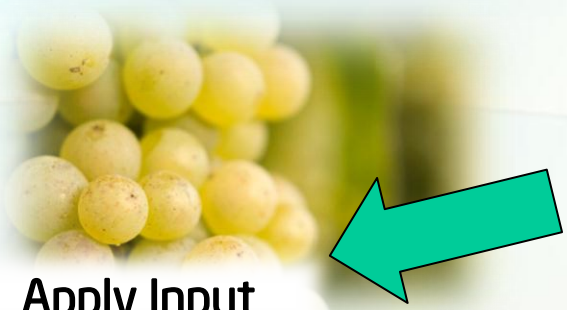
Apply Input

Expected Results

Teardown

(c) Michael Stahl, 2012, All Rights Reserved

The "standard" Test Design



Test Techniques

- Equivalence Partitioning
- Boundary Values Analysis
- Decision Table Testing
- State Transition Testing
- Use Case Testing
- Experience-based Testing
- Etc. etc.

Teardown

(c) Michael Stahl, 2012, All Rights Reserved

So…

How to design a Test?

(c) Michael Stahl, 2012, All Rights Reserved

Outline

- Input and Output basics

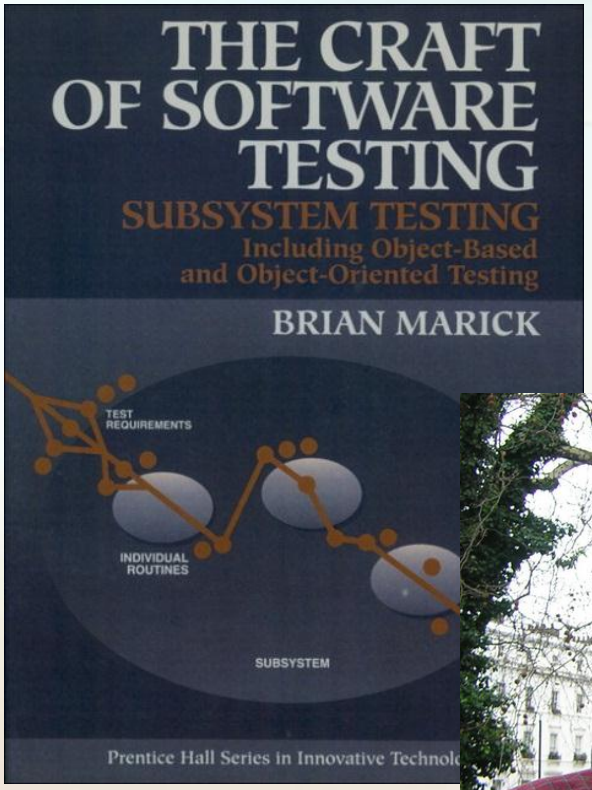
- Preconditions, Postconditions
- Foreground, Background
- What's the limit?
- Test flow
- Test your test

- Summary

(c) Michael Stahl, 2012, All Rights Reserved

המביא דבר בשם אומרו מביא גאולה לעולם

אבות ו' ו

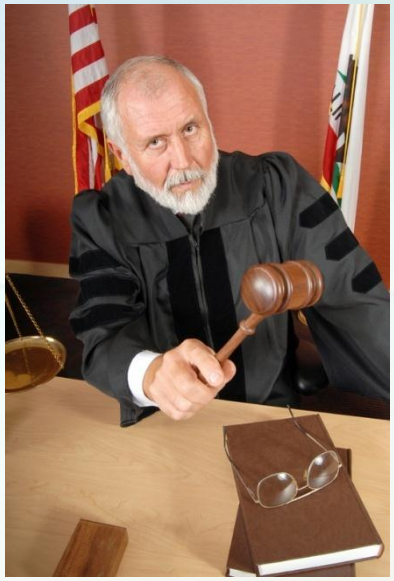


Whosoever tells a thing in the name of him that said it, brings redemption to the world

Ethics Of The Fathers 6, 6

(c) Michael Stahl, 2012, All Rights Reserved

Disclaimers



- Names and brands referenced herein may be claimed as the property of third parties
- The views expressed in this presentation are solely my own, and do not in any manner represent the views of my employer
- Information in this presentation is provided "AS IS" without any warranties or representations of any kind

(c) Michael Stahl, 2012, All Rights Reserved

Input

(c) Michael Stahl, 2012, All Rights Reserved

Input basics: Valid

Range

Enumeration

Input

Combinations (if applicable)

(c) Michael Stahl, 2012, All Rights Reserved

Input basics: Invalid

Range

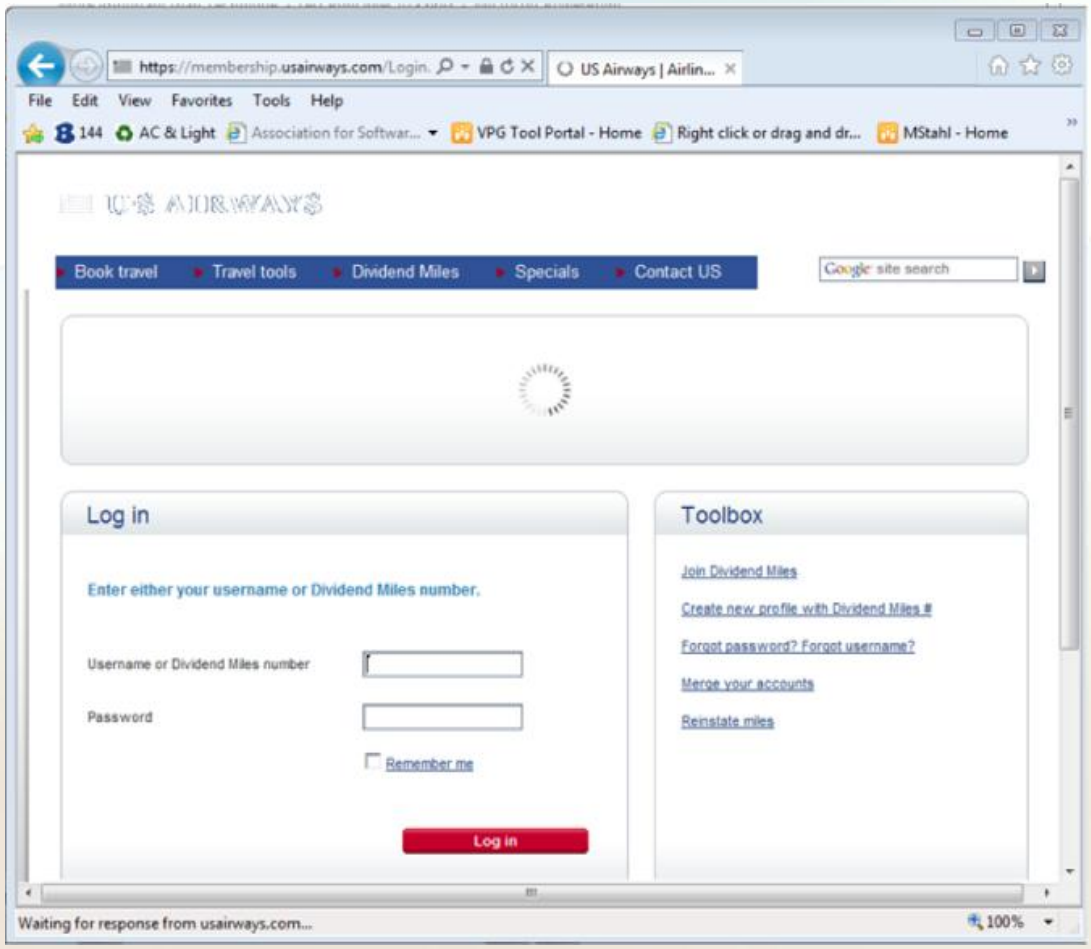
Enumeration

Combinations (in rare cases!)

Input

(c) Michael Stahl, 2012, All Rights Reserved

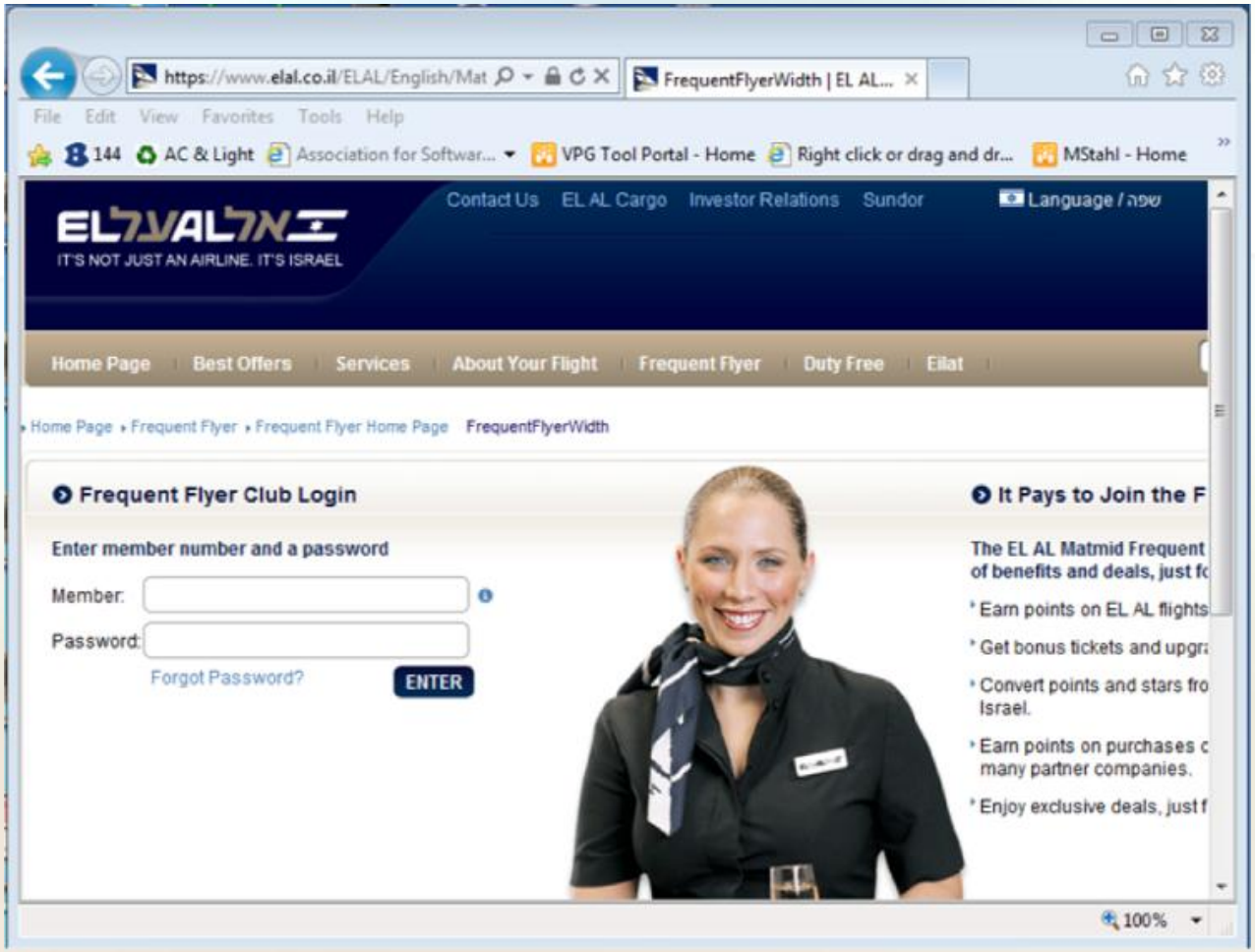
Input basics: Invalid



Invalid Combinations (in rare cases!)

(c) Michael Stahl, 2012, All Rights Reserved

Input basics: Invalid



Invalid Combinations (in rare cases!)

(c) Michael Stahl, 2012, All Rights Reserved

Output

(c) Michael Stahl, 2012, All Rights Reserved

Output basics: Valid

Range

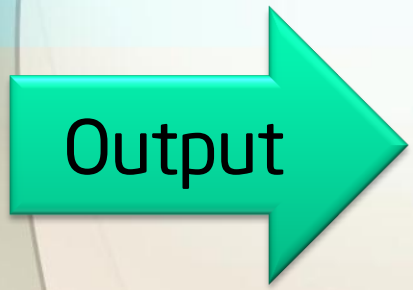
Output

Enumeration

Error codes or messages

(c) Michael Stahl, 2012, All Rights Reserved

Output basics: Invalid



(c) Michael Stahl, 2012, All Rights Reserved

A "Wider" look at Input: Preconditions

(c) Michael Stahl, 2012, All Rights Reserved

Input is one part of Preconditions

Environment

Input

System under test

Input

Configuration

(c) Michael Stahl, 2012, All Rights Reserved

Preconditions are Assumed or Validated

Validated

Assumed

Input

(c) Michael Stahl, 2012, All Rights Reserved

Assumed Preconditions

The system has the needed fonts

The system has a printer

The user can read

There IS a user

(c) Michael Stahl, 2012, All Rights Reserved

The Environment: Control, Set, Cause or Monitor?

Environment

Control

Set

Cause

Monitor

Input

System under test

Input

Configuration

(c) Michael Stahl, 2012, All Rights Reserved

The Environment:
~~Control, Set, Cause or Monitor?~~

Environment

Control *Set* *Cause* *Monitor* *Ignore*

Input

System under test

Input Configuration

(c) Michael Stahl, 2012, All Rights Reserved

A "Wider" look at Output: Postconditions

(c) Michael Stahl, 2012, All Rights Reserved

Output is one part of Postconditions

Output

Environment

System under test

Output

Configuration

(c) Michael Stahl, 2012, All Rights Reserved

Postconditions have Variants and Invariants

Output

Variants

Invariants

(c) Michael Stahl, 2012, All Rights Reserved

Post Conditions: Variants and Invariants

(c) Michael Stahl, 2012, All Rights Reserved

Creating Preconditions

Obvious:

- User inputs
- CLI arguments

Less obvious:

- Configuration (Files, Windows registry)
- API calls
- Sensors
- The Environment
  - OS, HW, installed stuff, interrupts
  - External stuff (temperature, radiation, comm…)

(c) Michael Stahl, 2012, All Rights Reserved

Checking Postconditions

- Variants:
  - "Expected Results"
  - Configurations; Environment
- Invariants:
  - Configurations; Environment
  - System resources
    - Memory
    - File handles
  - Pay special attention after Exceptions!

(c) Michael Stahl, 2012, All Rights Reserved

Checking Postconditions

- Background
  - Known state – before the test
- Foreground
  - Stuff that changes

(c) Michael Stahl, 2012, All Rights Reserved

Checking Postconditions

Before:

After:

(c) Michael Stahl, 2012, All Rights Reserved

Checking Postconditions

Before:

After:

(c) Michael Stahl, 2012, All Rights Reserved

Checking Postconditions

- Set background value to highlight change

- Set inputs to highlight mistakes
  - 2+2 = 2*2 !
  - 2-2 = 2-2 !

(c) Michael Stahl, 2012, All Rights Reserved

Output: Background and Foreground

```
Int Foo(int x)  {

   int status = SUCCESS;

   if (x > 1000) retVal = ERROR_OUT_OF_RANGE
   else          process(x, &status);

   return status;
}
```

What if process() does nothing ?

(c) Michael Stahl, 2012, All Rights Reserved

Output: Background and Foreground

```
Int Foo(int x)  {

   int status = UNUSED_VALUE;

   if (x > 1000) retVal = ERROR_OUT_OF_RANGE
   else          process(x, &status);

   return status;
}
```

(c) Michael Stahl, 2012, All Rights Reserved

Output: Background and Foreground

Array testing:

```
char src[16]={0}, dst[16]={0};
// src gets value from user
strncpy(src, input, sizeof(input));
// src is copied to dst
strncpy(dst, src, sizeof(dst));
```

(c) Michael Stahl, 2012, All Rights Reserved

Output: Background and Foreground

Array testing:

```
char src[16]={0}, dst[16]={0};
// src gets value from user
strncpy(src, input, sizeof(input));
// src is copied to dst
strncpy(dst, src, sizeof(dst)); //bug!
```

(c) Michael Stahl, 2012, All Rights Reserved

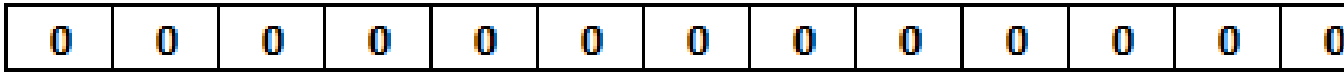
Output: Background and Foreground

Array testing:

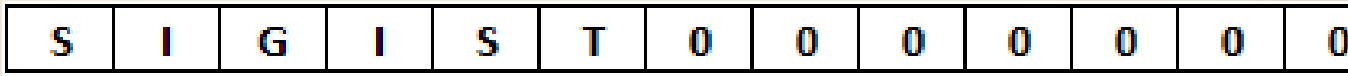
Incorrect test:

1. Enter “SIGIST” in input field
2. Verify dst = “SIGIST”

dst before:



dst after:



Test Result = PASS // ☹

(c) Michael Stahl, 2012, All Rights Reserved

Output: Background and Foreground

Array testing:

Correct test:

1. Set dst to {A}
2. Enter "SIGIST" in input field
3. Verify dst = "SIGISTAAAAAAAAAA"

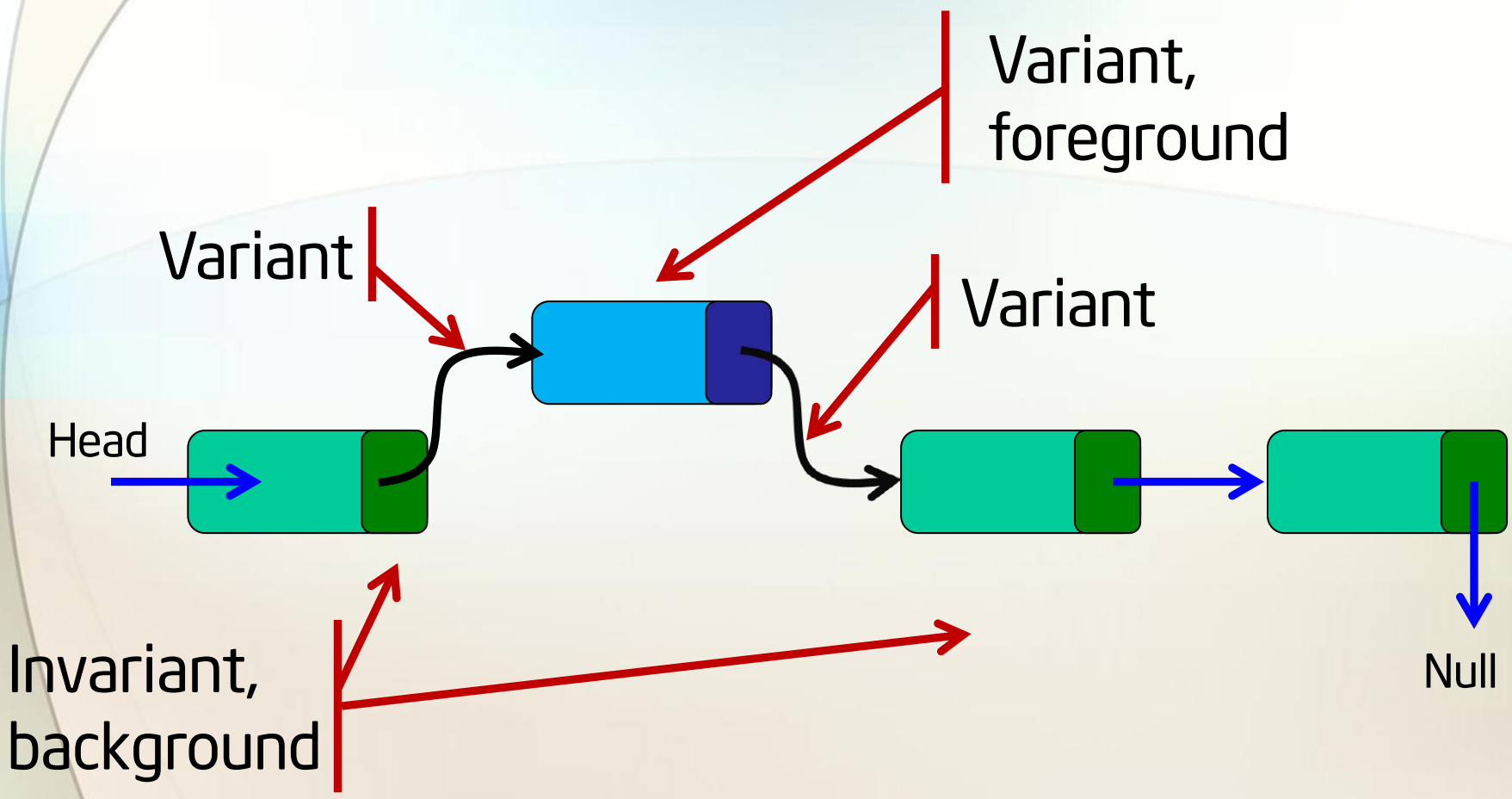
dst before:

dst after:

Test Result = FAIL // ☺

(c) Michael Stahl, 2012, All Rights Reserved

Everything together: Linked-list "Add"



(c) Michael Stahl, 2012, All Rights Reserved

Output: Non-functional attributes limits

Set/Verify execution conditions

Set background

Apply Input

DUT

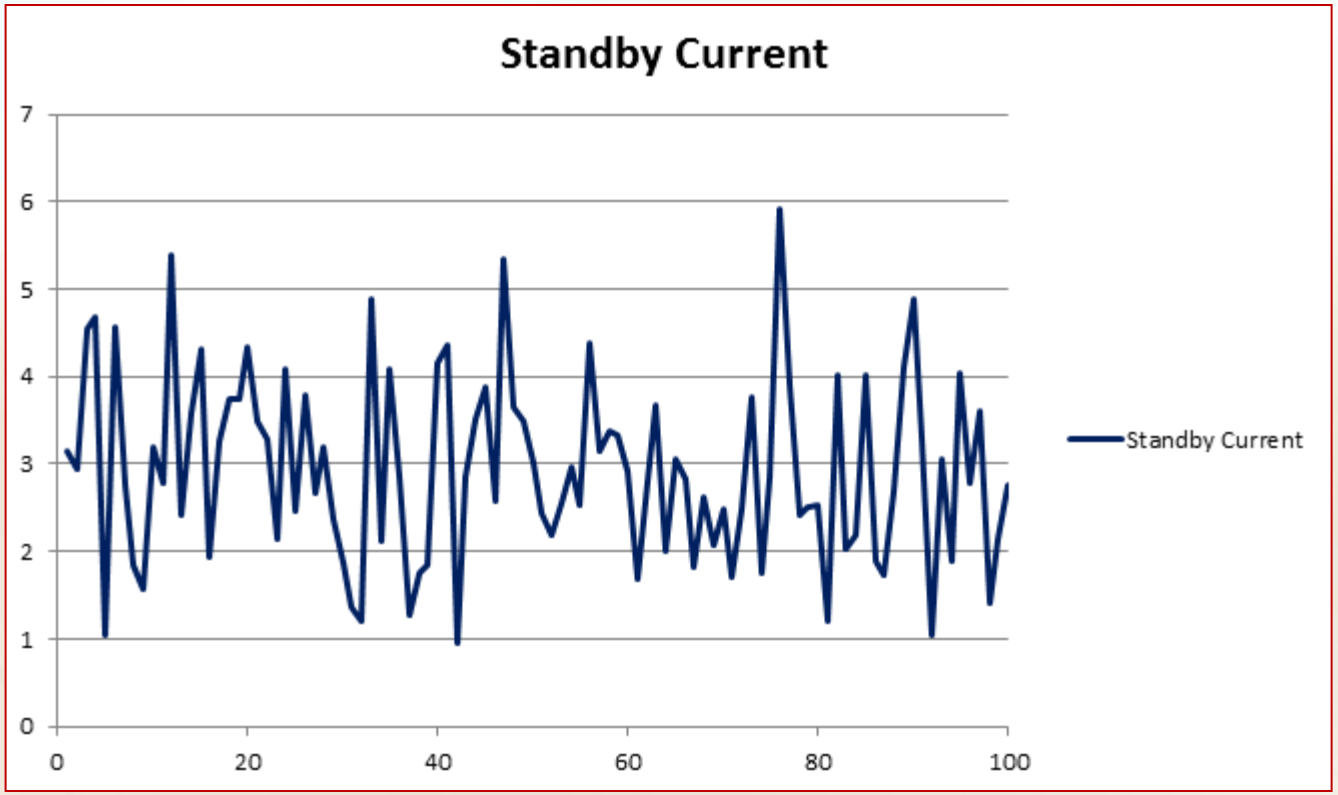
Check Output

(c) Michael Stahl, 2012, All Rights Reserved

(c) Michael Stahl, 2012, All Rights Reserved

What's the limit?

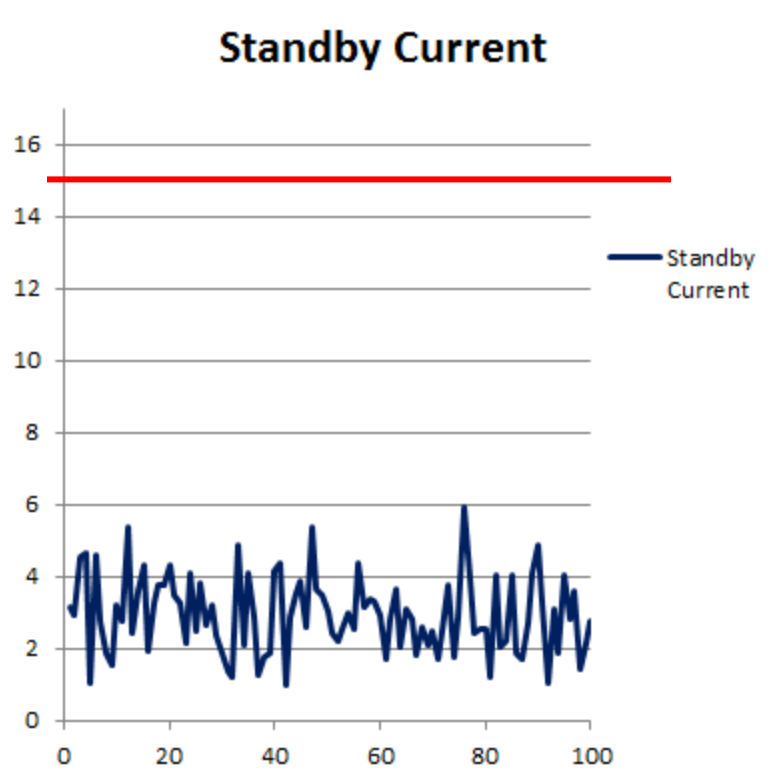
Spec: Standby Current < 15.0

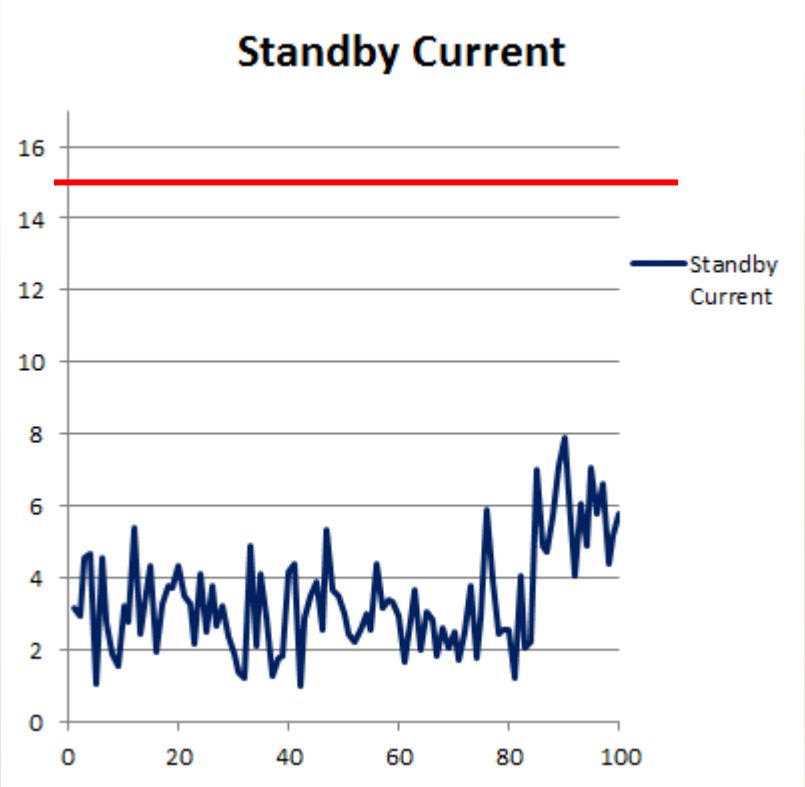


(c) Michael Stahl, 2012, All Rights Reserved

What's the limit?

Spec: < 15.0





(c) Michael Stahl, 2012, All Rights Reserved

Test the Capability

- Expected Results:
  - Based on capability – not on Spec*
- Rationale:
  - Increase the test sensitivity
  - Find the real limit

* Unless the capabilities are worse than Spec…

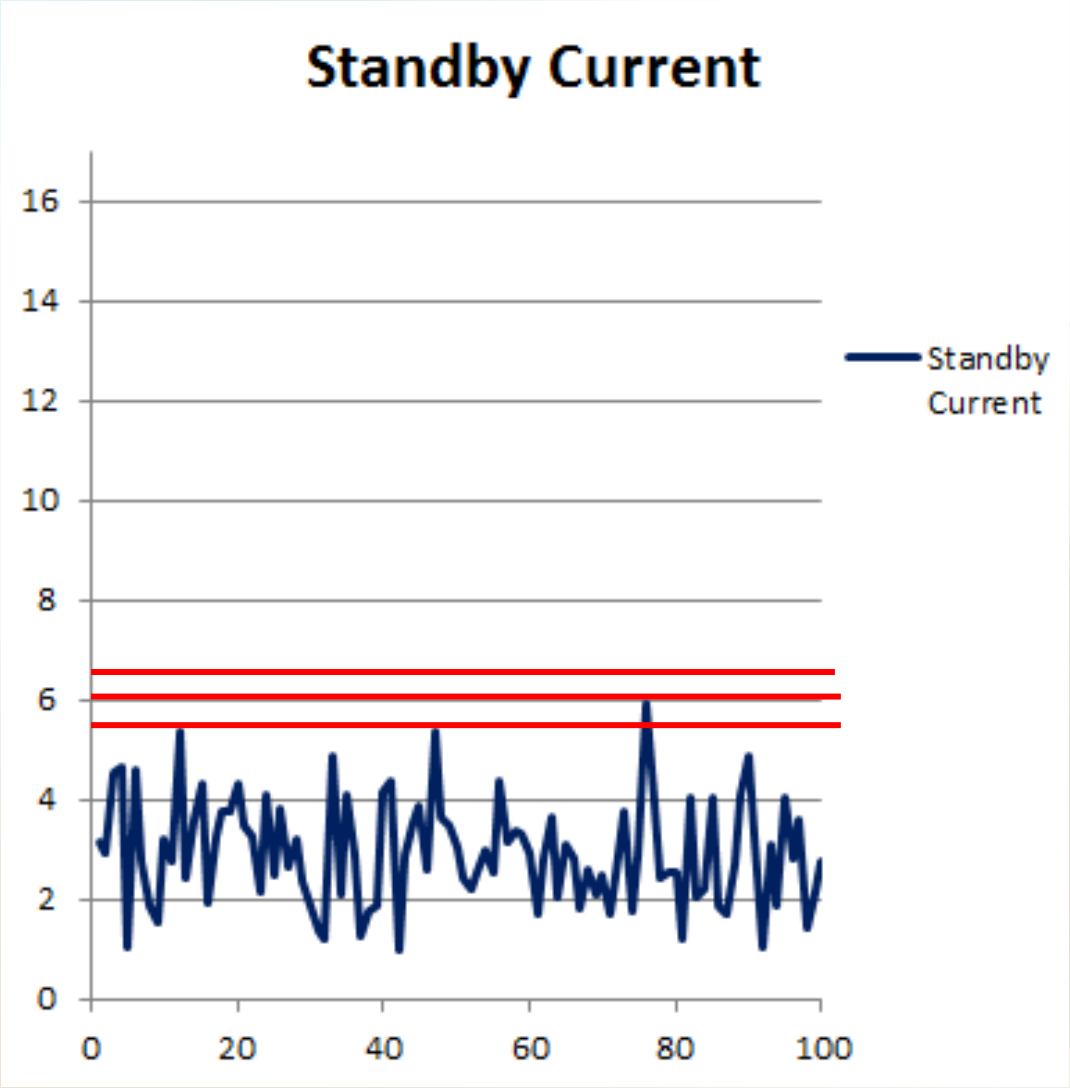
(c) Michael Stahl, 2012, All Rights Reserved

Increasing the test sensitivity

Fail if > 5.5?

6?

6.5?



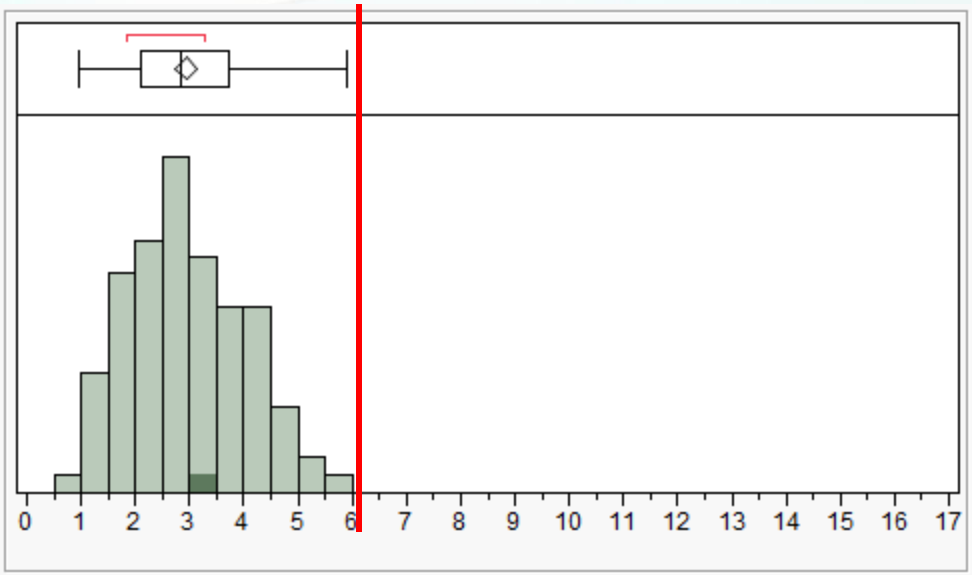
Risk: False alarms

(c) Michael Stahl, 2012, All Rights Reserved

Increasing the test sensitivity

Spec: < 15.0

Where to put the test limit?



Quantiles

100.0%	maximum	5.91164
99.5%		5.91164
97.5%		5.3671
90.0%		4.36426
75.0%	quartile	3.7218
50.0%	median	2.84308
25.0%	quartile	2.1151
10.0%		1.67605
2.5%		1.04148
0.5%		0.95798
0.0%	minimum	0.95798

Moments

Mean	2.9371787
Std Dev	1.0666257
Std Err Mean	0.1066626
Upper 95% Mean	3.1488204
Lower 95% Mean	2.725537
N	100

Mean+1σ = 4; Mean+2σ = 5; Mean+3σ = 6

(c) Michael Stahl, 2012, All Rights Reserved

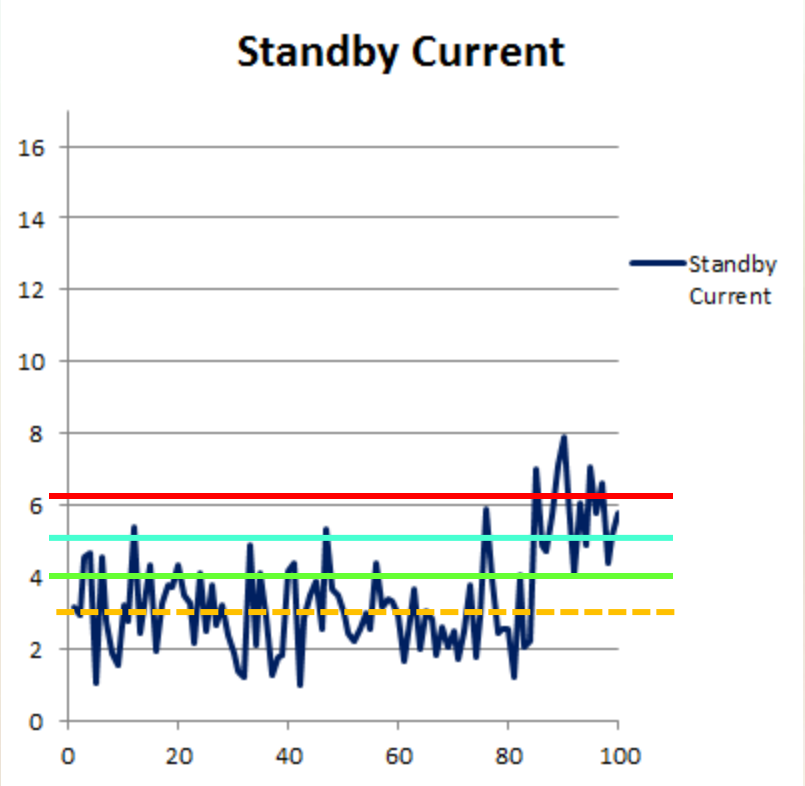
What's a Fail then?

Use Process Control methods and rules

Example: Fail if:

- One measurement above +3σ
- Two consecutive points falling above the +2σ limit
- Three consecutive points falling above the +1σ limit
- Seven consecutive points falling above the centerline
- Ten consecutive points falling below the centerline
- …

Source: Wikipedia - Western Electric rules

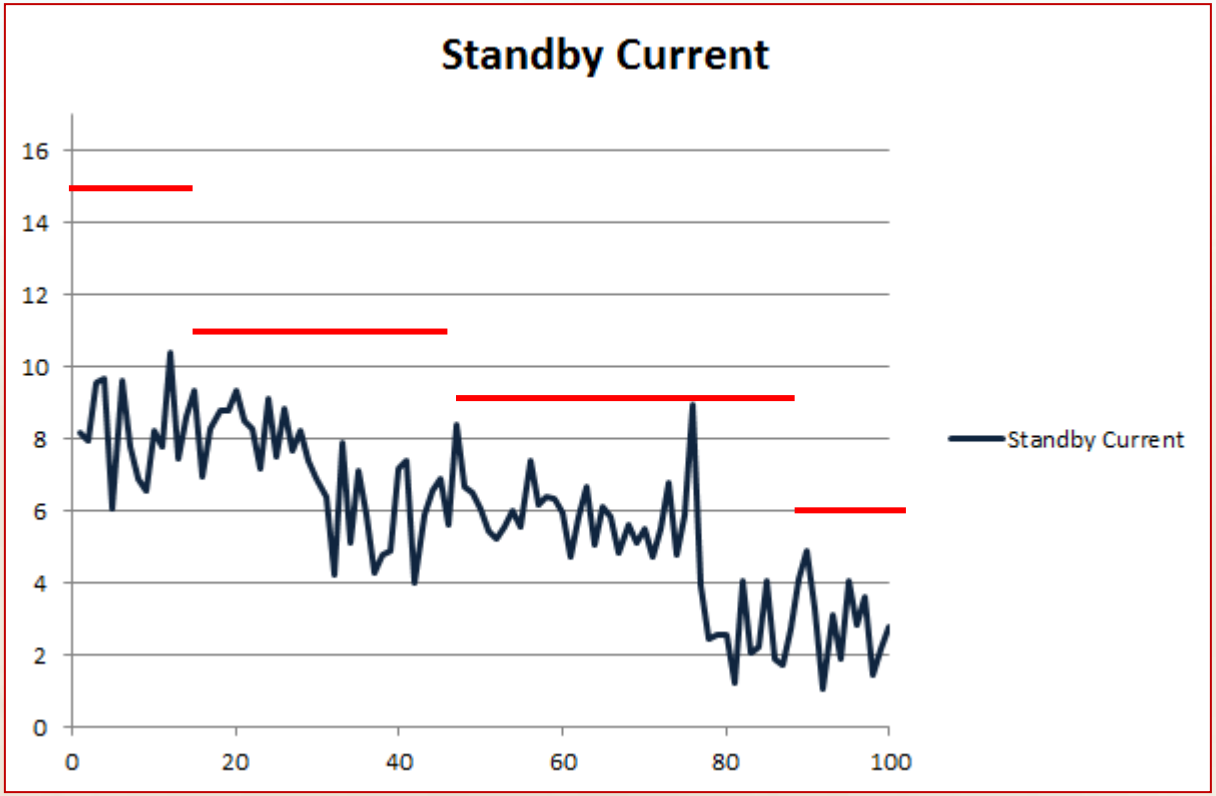


(c) Michael Stahl, 2012, All Rights Reserved

"Tighten the Screws"

Spec: < 15.0

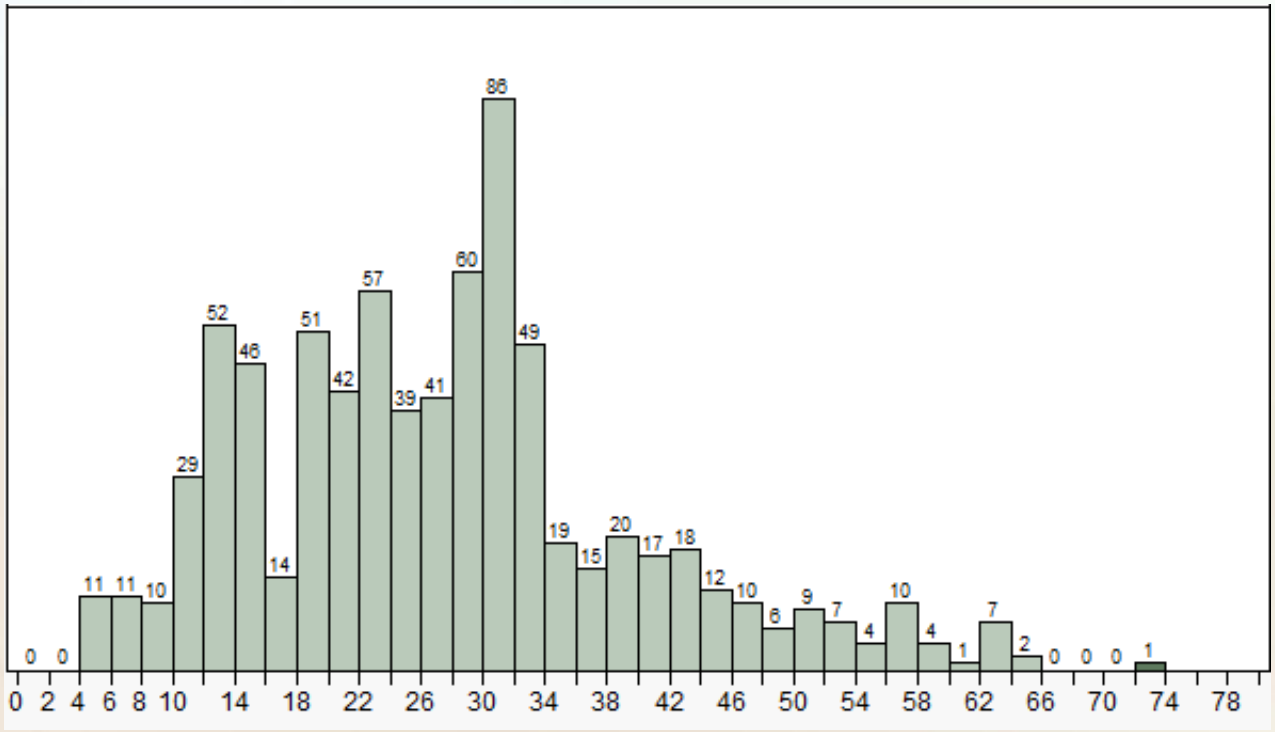
How much regression do you allow?



(c) Michael Stahl, 2012, All Rights Reserved

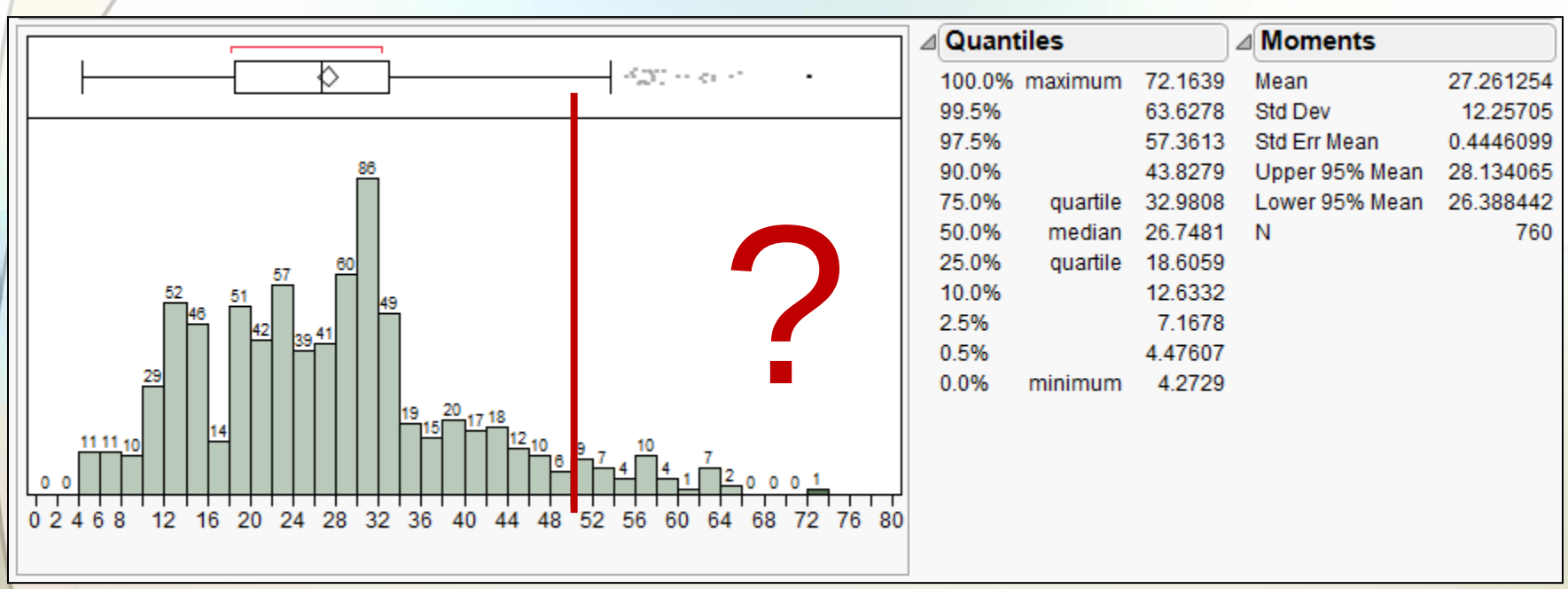
Example – real case (Atom processor)

- Some measurements ☺
- The spec: PASS if below 50%
- Capability (760 measurements):



(c) Michael Stahl, 2012, All Rights Reserved

Where would YOU put the limit?



Mean+1σ = 39.5

Mean+2σ = 52.25 ☹

Mean+3σ = 64 ☹ $#^%&#@!!!

(c) Michael Stahl, 2012, All Rights Reserved

Additional info

Different results due to different scenario

x

Scenario	1	2	3	4	...	19
Run 1	##	##	##	##		##
Run 2	##	##	##	##		##
Run 3	##	##	##	##		##
Run 4	##	##	##	##		##
Run 5	##	##	##	##		##
Run 6	##	##	##	##		##
Run 7	##	##	##	##		##
Run 8	##	##	##	##		##
Run 9	##	##	##	##		##
Run 10	##	##	##	##		##

= 760

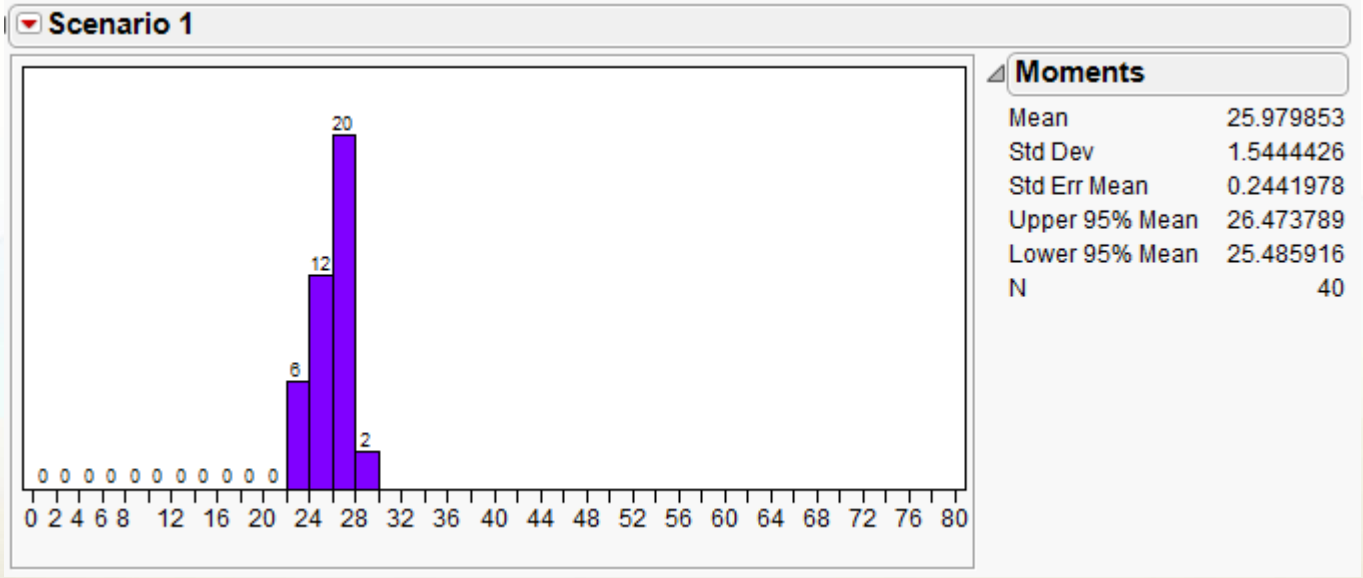
Different results due to "system capability"

(c) Michael Stahl, 2012, All Rights Reserved

“Use a Larger Magnifying Glass”

(c) Michael Stahl, 2012, All Rights Reserved

Per-scenario distribution



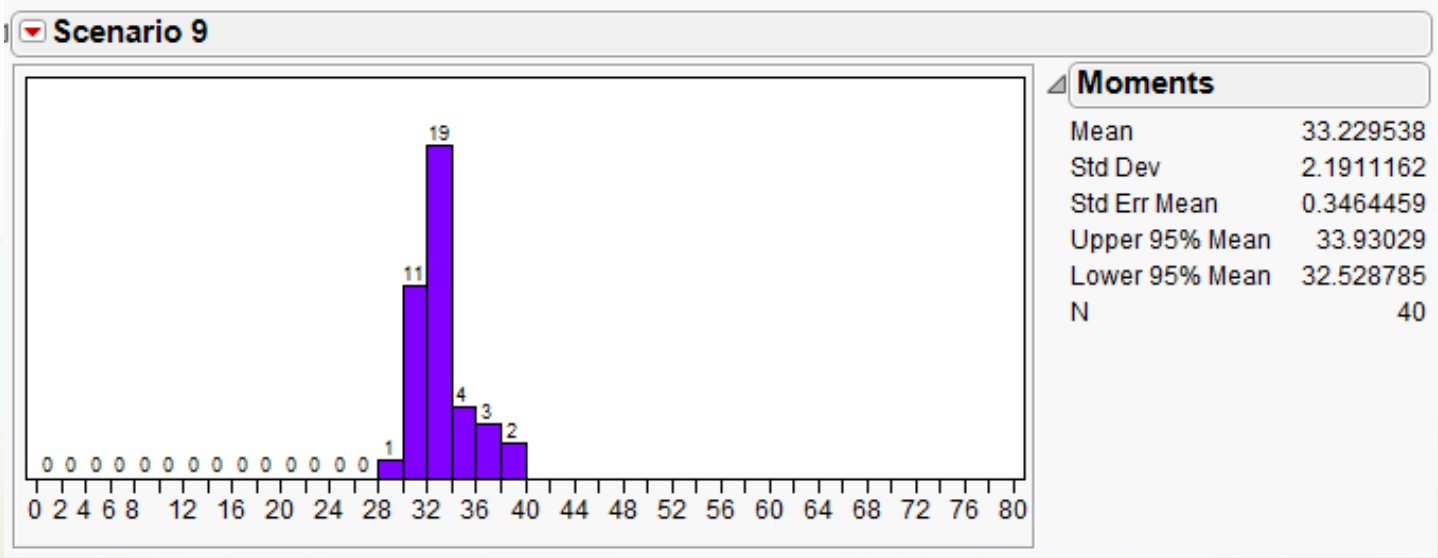
Mean+1σ = 27.5

Mean+2σ = 29.1

Mean+3σ = 30.6 ☺ !!!

(c) Michael Stahl, 2012, All Rights Reserved

Per-scenario distribution



Mean+1σ = 35.4

Mean+2σ = 37.6

Mean+3σ = 39.8 ☺ !!!

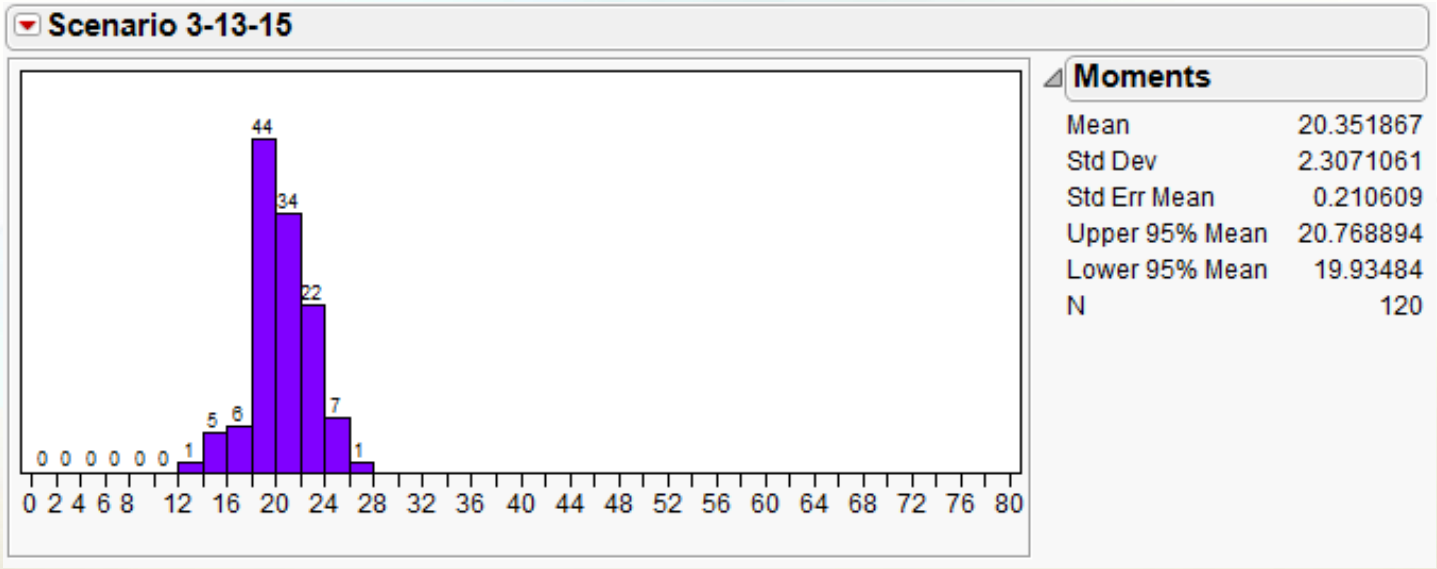
Mean-1σ = 31.0

Mean-2σ = 28.8

Mean-3σ = 26.6

(c) Michael Stahl, 2012, All Rights Reserved

Per group-of-scenarios distribution



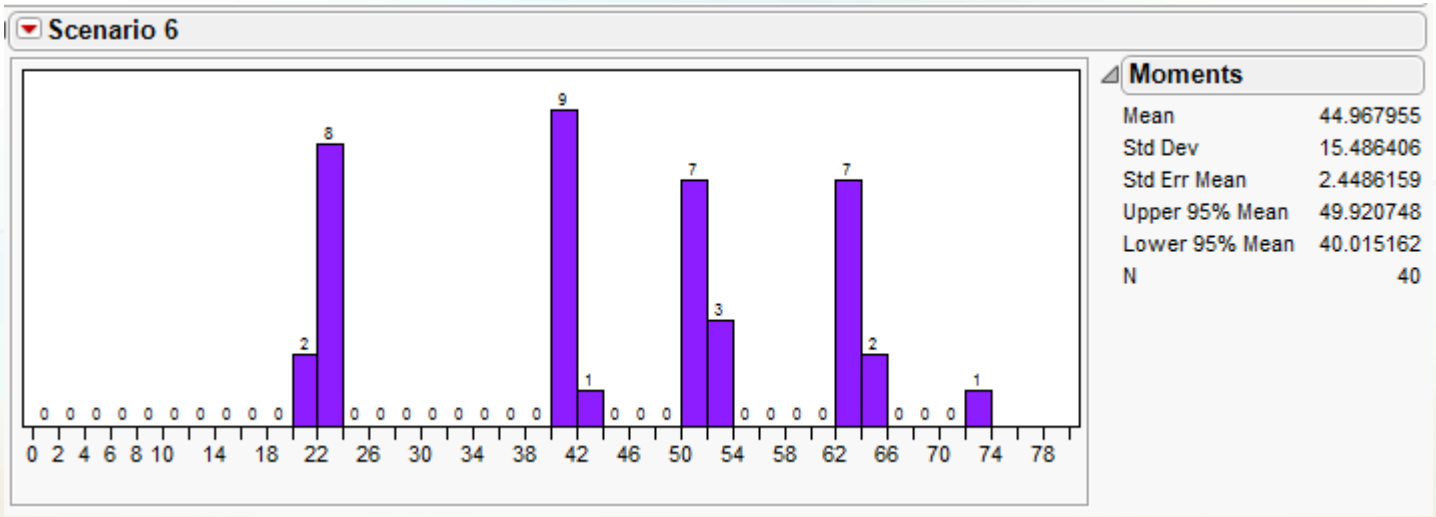
Mean+1σ = 22.6

Mean+2σ = 24.9

Mean+3σ = 27.2 ☺ !!!

(c) Michael Stahl, 2012, All Rights Reserved

Per group-of-scenarios distribution



Something is interesting...

Or wrong...

(c) Michael Stahl, 2012, All Rights Reserved

Setup

(c) Michael Stahl, 2012, All Rights Reserved

Test flow (naïve)

Did "Set" work?

Set execution conditions

Apply Input

Check Output

(c) Michael Stahl, 2012, All Rights Reserved

Validate "Set"!

Get execution conditions

(record the background)

Set to different values

(change the background)

Get execution conditions

(validate the foreground)

Set-able variables are Input!

- Apply the "Basics"
- Do you allow setting invalid values?

(c) Michael Stahl, 2012, All Rights Reserved

Validate "Set" and "Get"!

Set test app internals to different values

(set the background)

Get execution conditions

Validate the foreground + set background for next step

Set the DUT to different values

(change the background)

Get con

(val fore

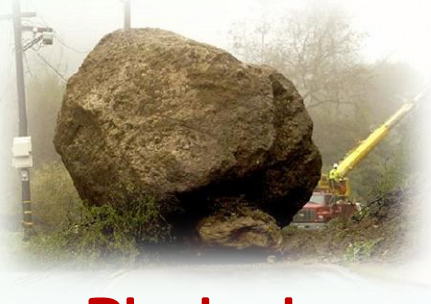
(c) Michael Stahl, 2012, All Rights Reserved

Test flow

Set execution conditions

Verify execution conditions

Fail?



Blocked!

Set background

Apply Input

DUT

Check Output

Teardown

(c) Michael Stahl, 2012, All Rights Reserved

Teardown

(c) Michael Stahl, 2012, All Rights Reserved

Teardown

Reset conditions to default

Clear background

Clean the environment

(c) Michael Stahl, 2012, All Rights Reserved

Teardown & Test efficiency

- State-aware Setup
- Combine tests

(c) Michael Stahl, 2012, All Rights Reserved

Validate your test!

(c) Michael Stahl, 2012, All Rights Reserved

Validate your test!

Set/Verify execution conditions

Set background

Apply Input

DUT

Check Output

(c) Michael Stahl, 2012, All Rights Reserved

Validate your test!

Set/Verify execution conditions

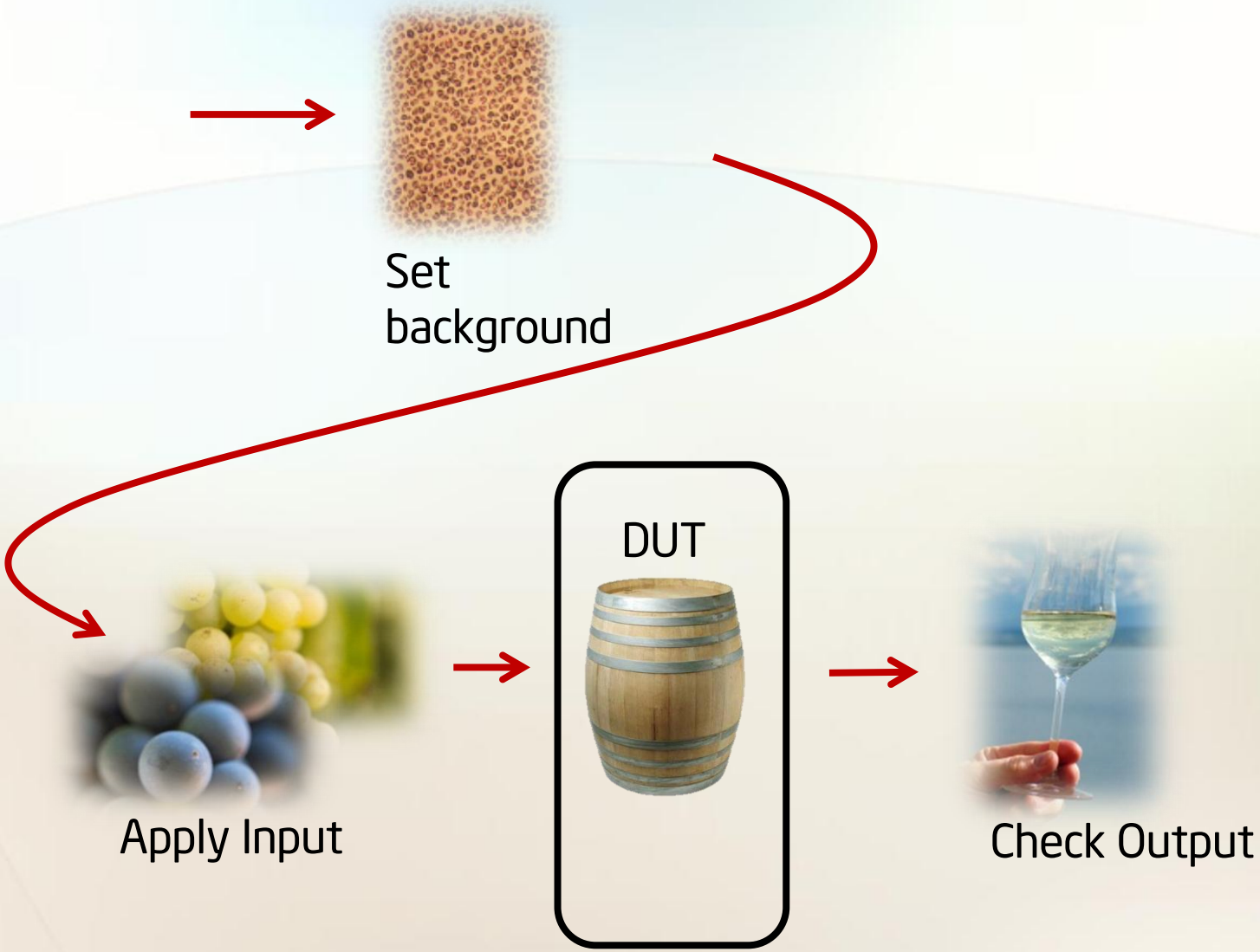
Set background

Apply Input

Check Output

(c) Michael Stahl, 2012, All Rights Reserved

Validate your test!



(c) Michael Stahl, 2012, All Rights Reserved

Validate your test!

Set/Verify execution conditions

Apply Input

DUT

Check Output

(c) Michael Stahl, 2012, All Rights Reserved

Validate your test!

Set/Verify execution conditions

Set background

Apply Input

DUT

Check Output

(c) Michael Stahl, 2012, All Rights Reserved

Validate your test!

Set/Verify execution conditions

Set background

Apply Input

DUT

Check Output

(c) Michael Stahl, 2012, All Rights Reserved

Validate your test!

Set/Verify execution conditions

Set background

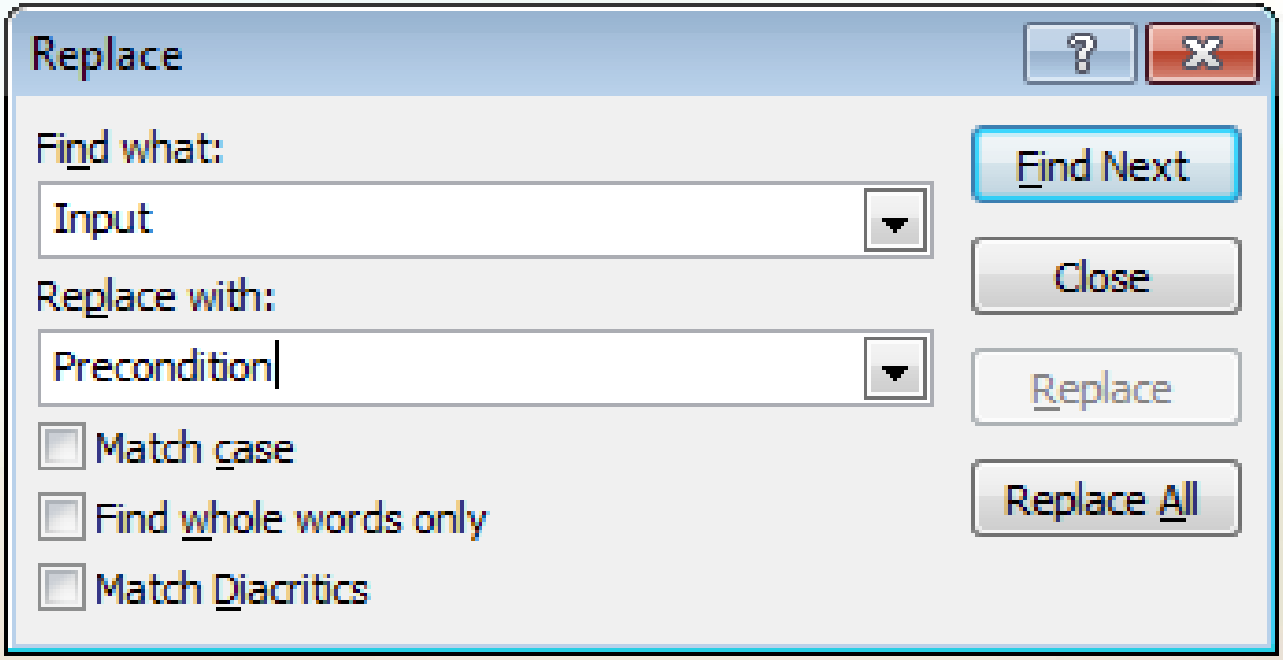
Apply Input

DUT

Check Output

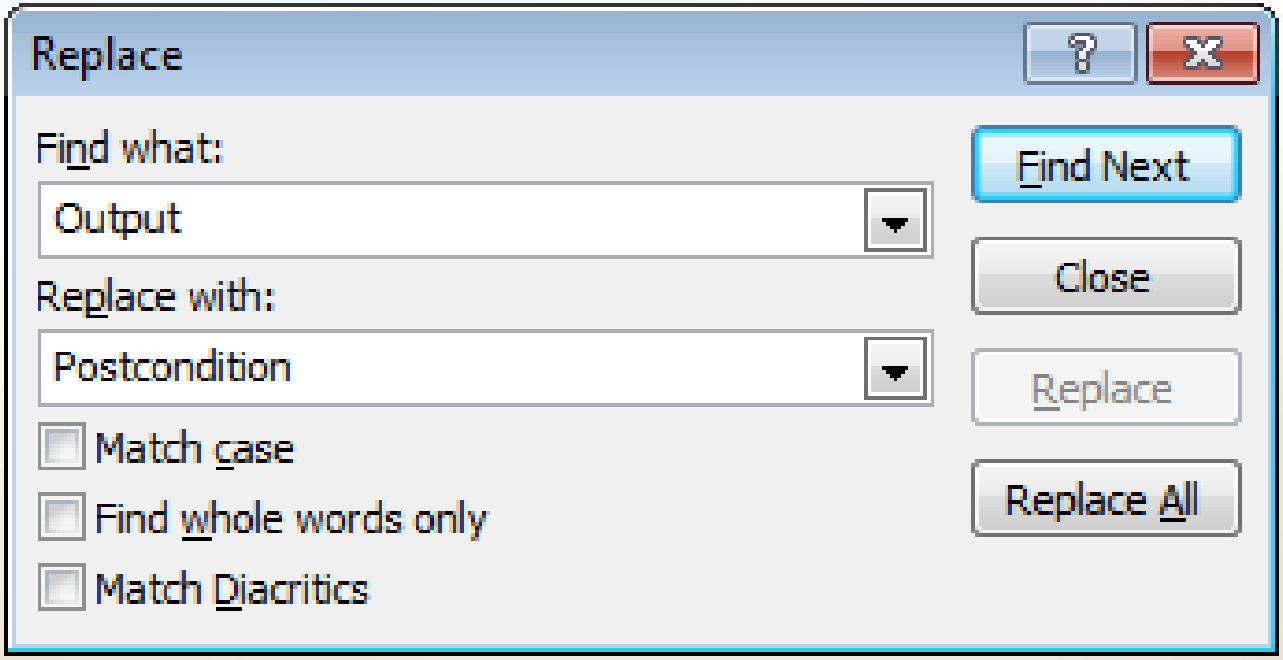
(c) Michael Stahl, 2012, All Rights Reserved

Summary



(c) Michael Stahl, 2012, All Rights Reserved

Summary



(c) Michael Stahl, 2012, All Rights Reserved

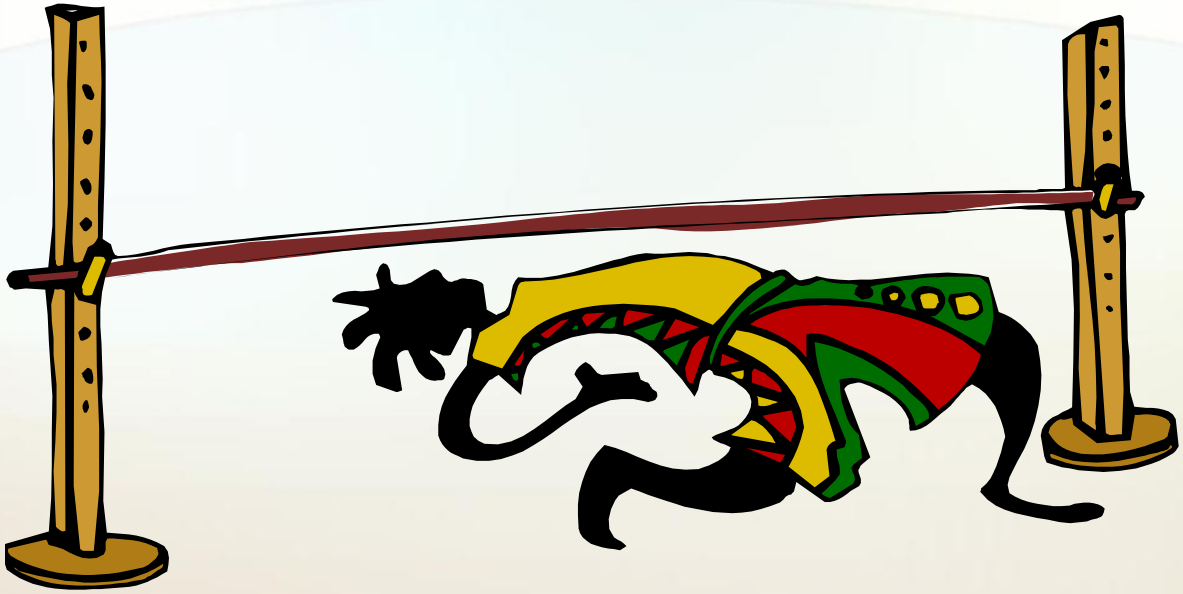
Summary

(c) Michael Stahl, 2012, All Rights Reserved

Summary

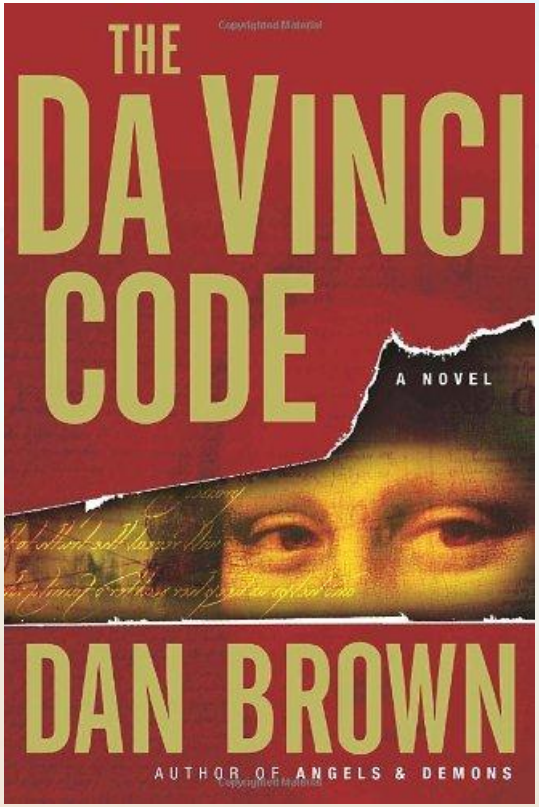
(c) Michael Stahl, 2012, All Rights Reserved

Summary



(c) Michael Stahl, 2012, All Rights Reserved

Summary



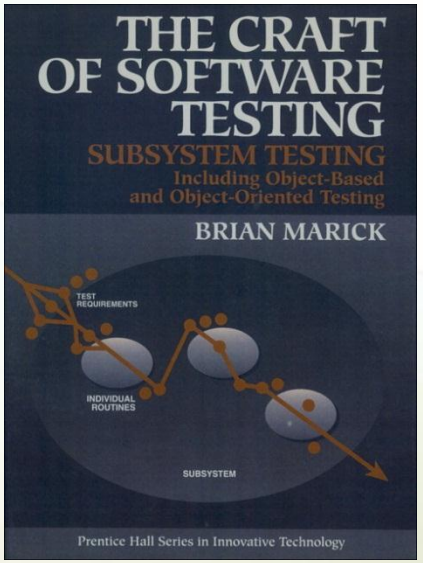
Quis custodiet ipsos custodes?

(c) Michael Stahl, 2012, All Rights Reserved

Acknowledgements / Resources

Resource:

- The Craft of Software Testing – **Brian Marick**



- Why Tests Don't Pass (or Fail), Hoffman, Doug, STPCon 2012

Thanks!

- Intel colleagues **Shmuel Gershon, Dan Alloun , Bet-Sheva Honigstein-Magen, Issi Hazan** – for suffering through the initial version of this presentation, and making it much better

(c) Michael Stahl, 2012, All Rights Reserved

Thank You!