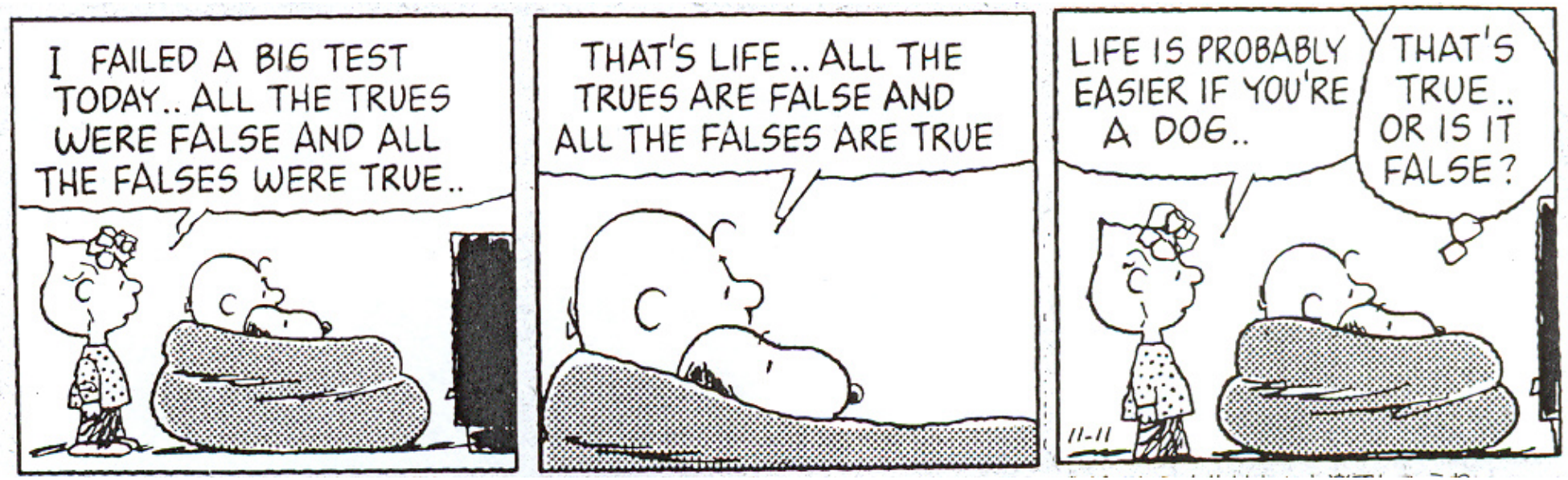
The Non-Virtual Reality of Testing or What's Feasible in Real World Testing

Contents

Karol Frühauf, *INFOGEM AG*, CH-5400 Baden, Karol.Fruehauf@infogem.ch



Seven myths about testing

I Testing is a hobby of quality people

II The quickest way to deployment is ping-pong testing

III Test automation is cheap

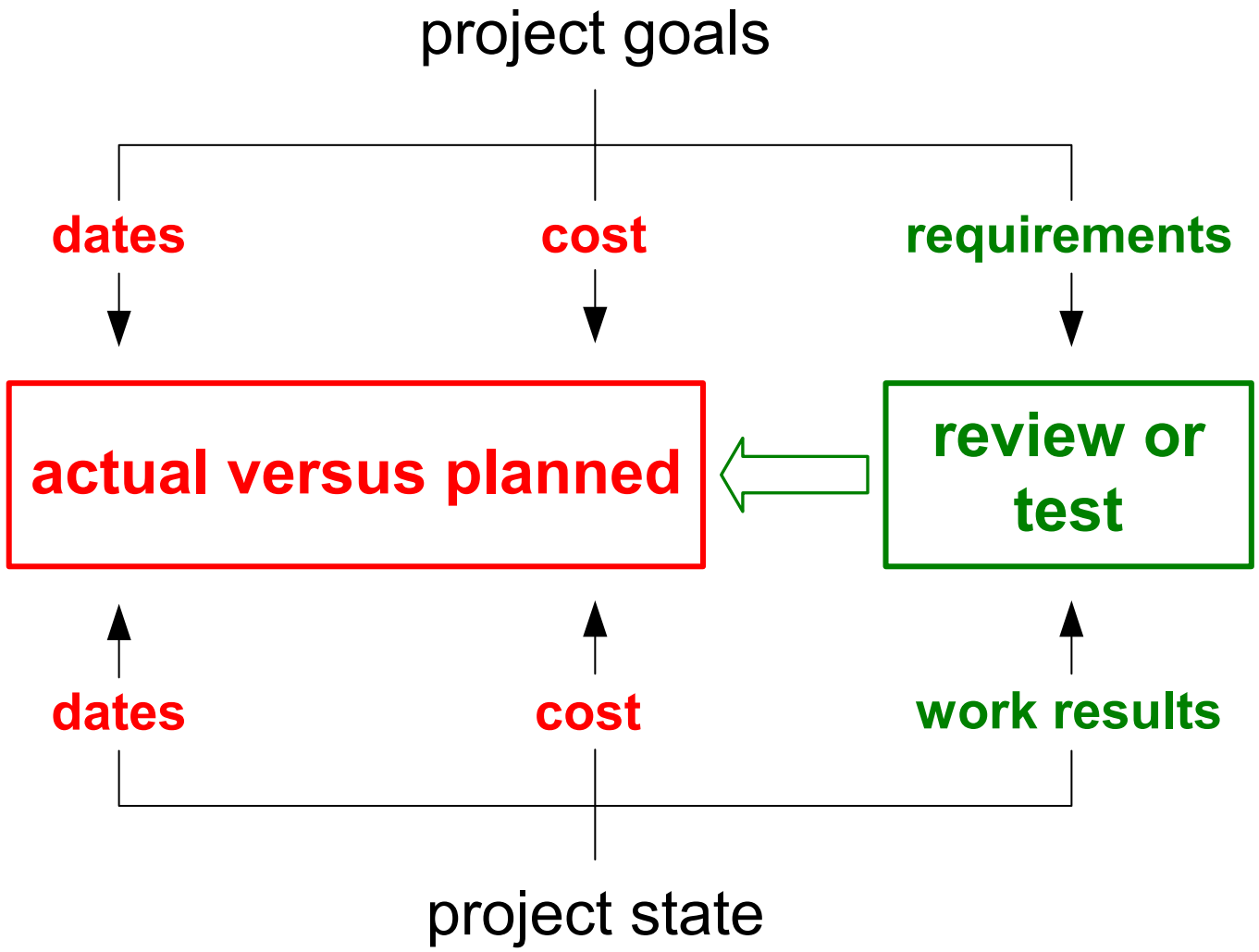
IV You don't need to see what you test

V Integration testing is interface testing

VI Test coverage is a glass box test concept

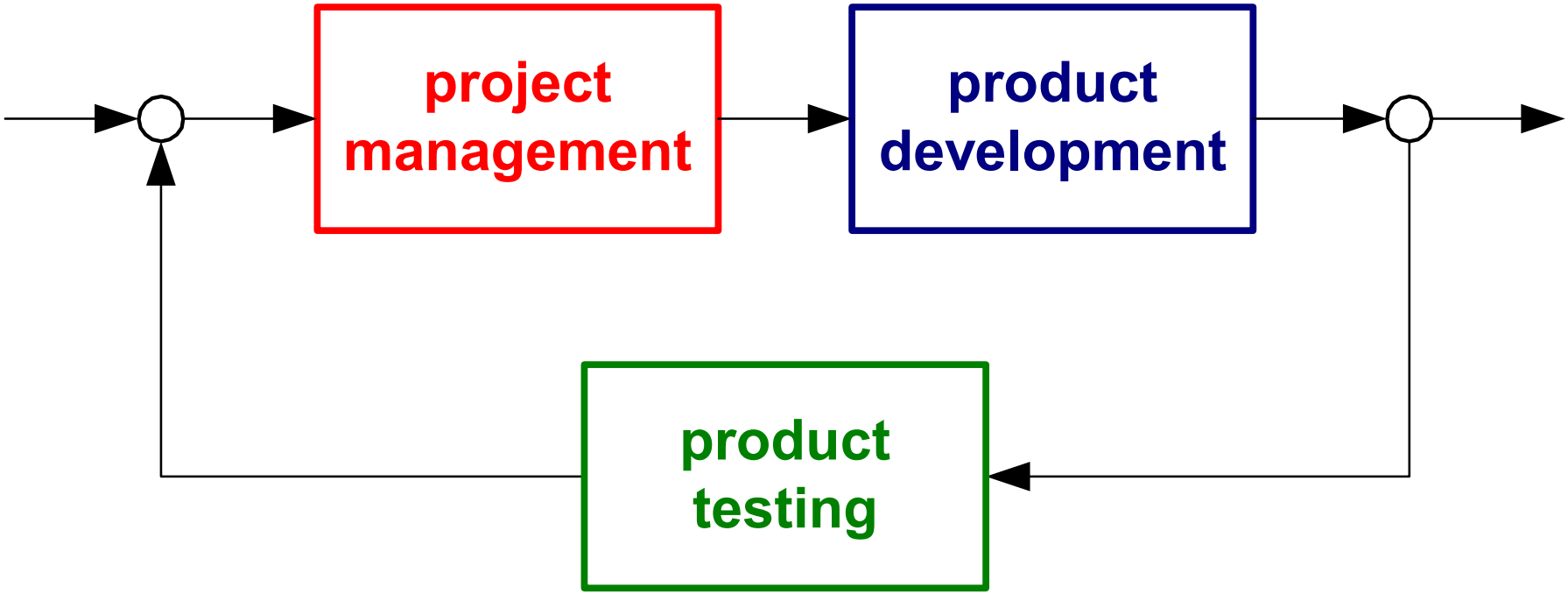
VII Test planning is an easy task

I Testing is a hobby of quality people (1)



⇨ **without review and test no real progress control**

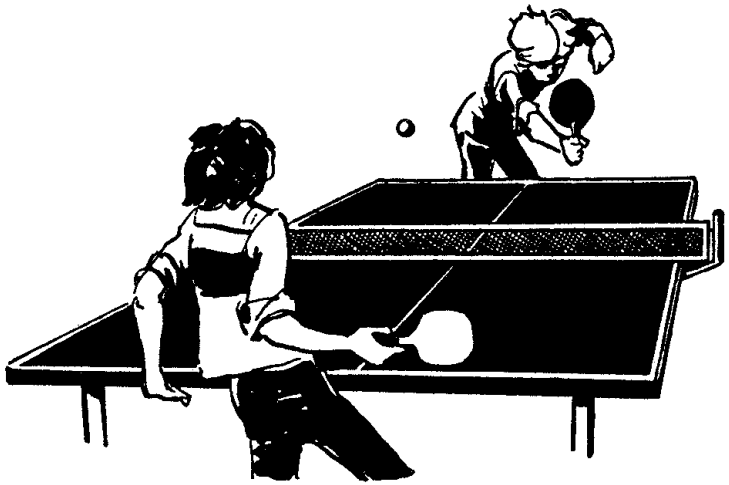
I Testing is a hobby of quality people (2)



⇨ **don't throw defects over the wall to the developer**

II The quickest way to release is ping-pong testing

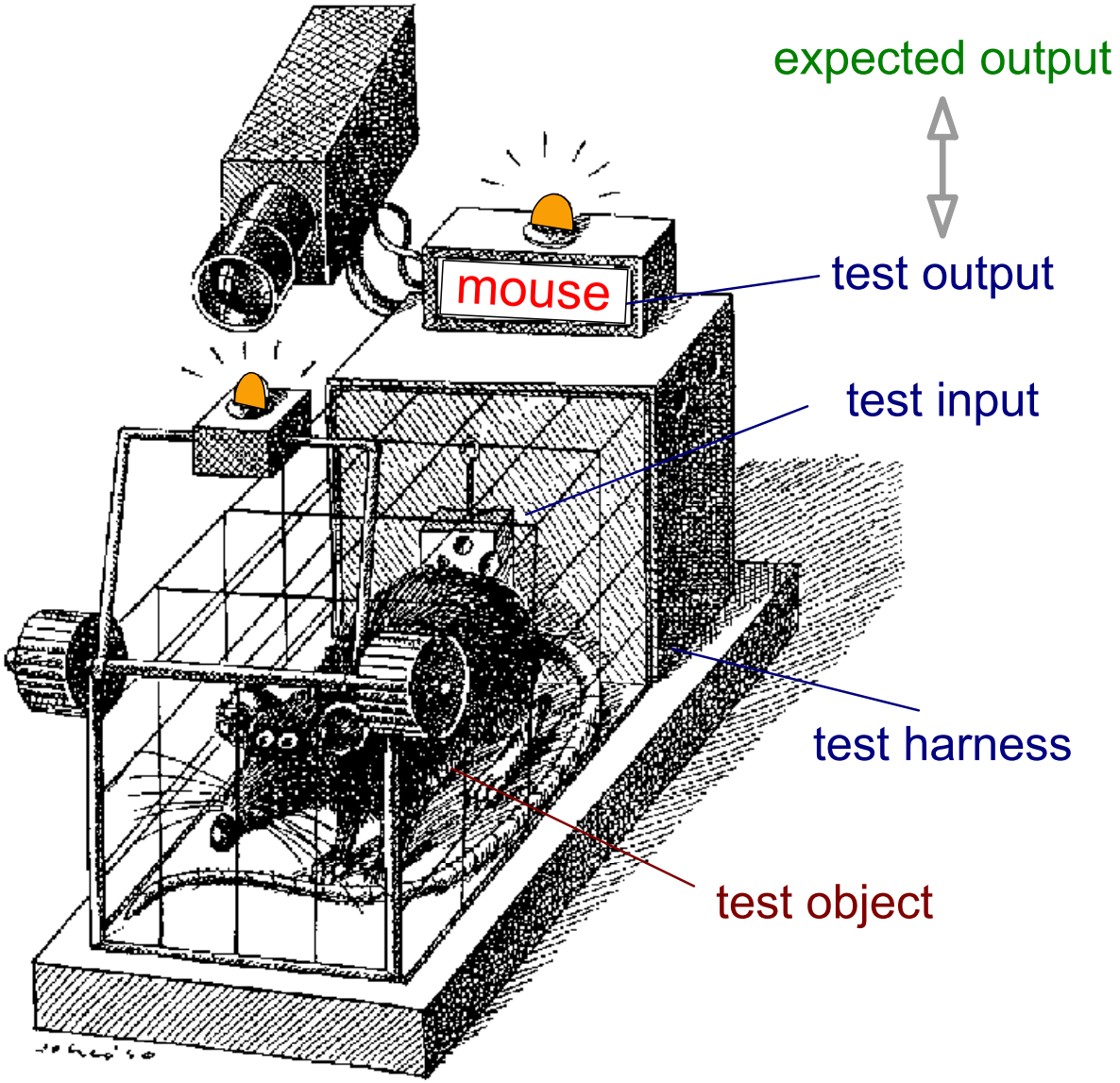
.. as soon as the tester detects a defect



he returns the software to the developer

- ⇨ **we have one defect to fix ...**
- ⇨ **expensive regression tests**
- ⇨ **if** special condition **then** rucksack**;**
- ⇨ **execute all specified test cases, then switch to repair mode**

III Test automation is cheap (1)



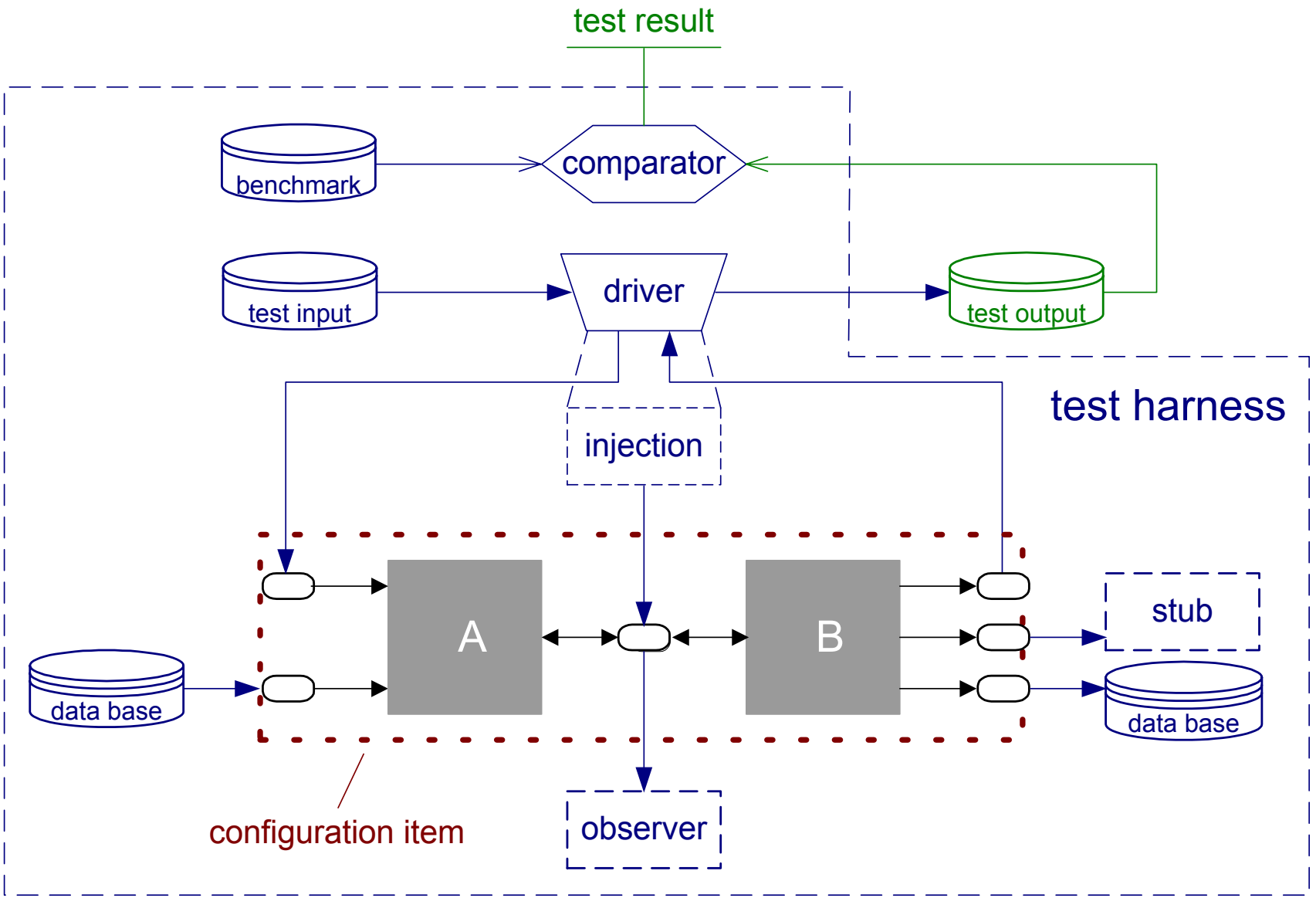
system level

- capture / replay tools
  - \+ capture is cheap
  - – replay is expensive
- test case managers requires strong update discipline

unit level

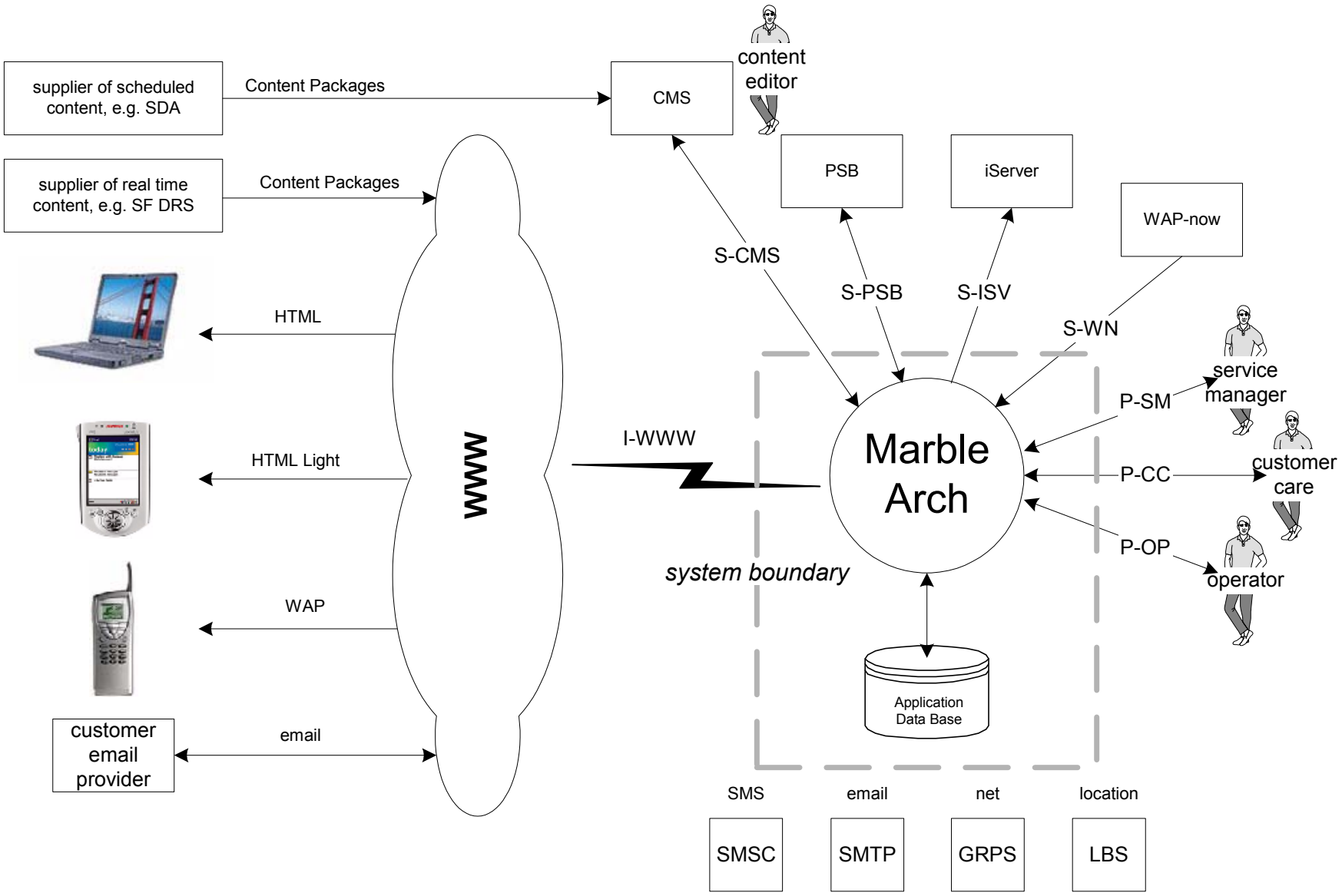
- \+ JUnit etc.
- \+ test coverage profiler
- \+ etc.

III Test automation is cheap (2)

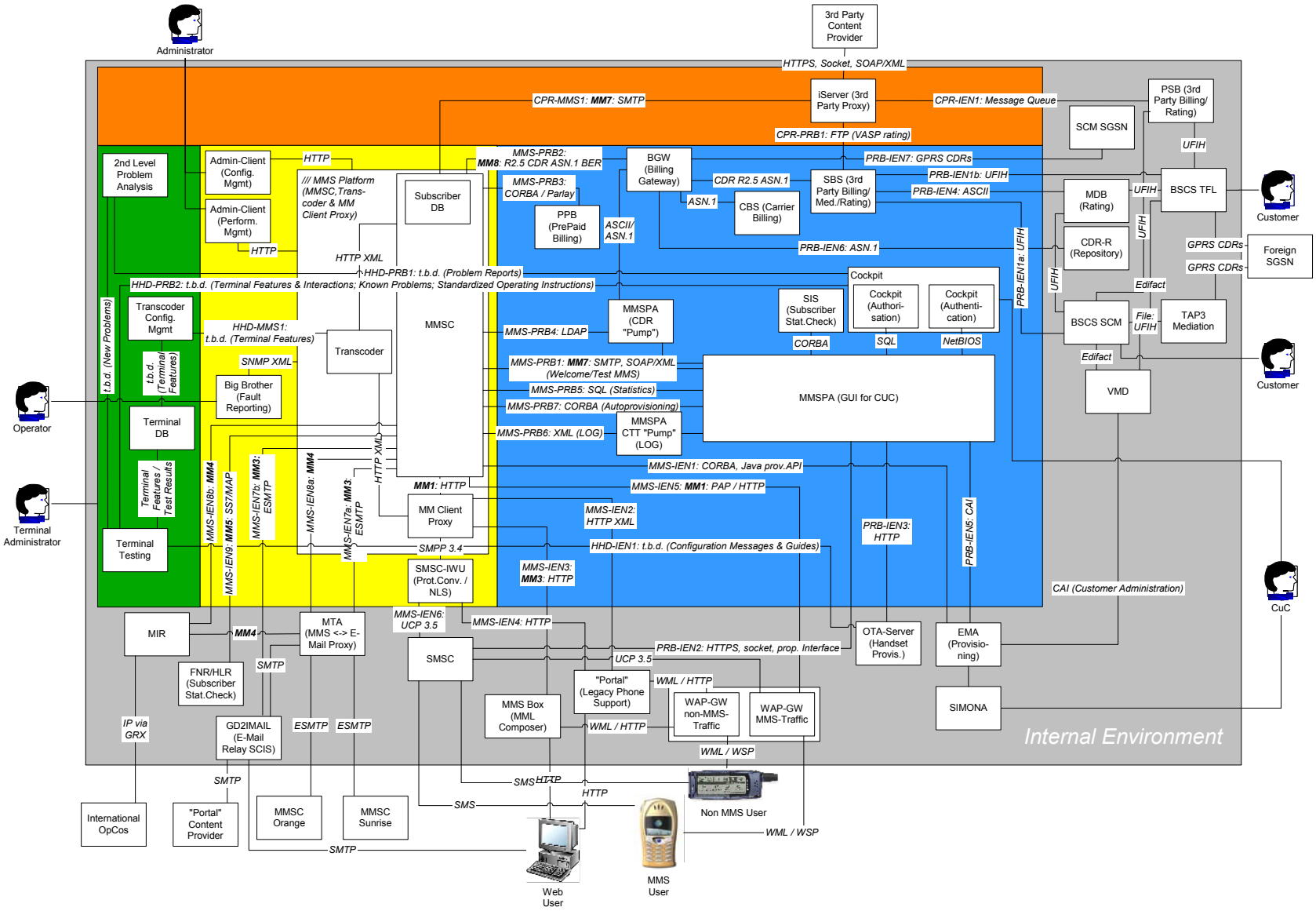


use home made dedicated tools ... keep them up-to-date

IV You don't need to see what you test (1)



IV You don't need to see what you test (2)

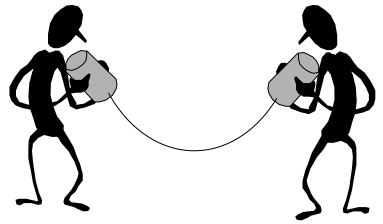


V Integration testing is interface testing (1)

Integration testing: Testing in which software components, hardware components, or both are combined and tested to evaluate the interaction between them.
[IEEE 610.12]

Integration testing: Testing performed to expose faults in the interfaces and in the interaction between integrated components.
Interface testing: Integration testing where the interfaces between system components are tested
[BS7925-1]

Integration testing is the process of verifying the interaction between system components (possibly and hopefully tested already in isolation).
[SWEBOK 1.0]



V Integration testing is interface testing (2)

implementation testing

→ testing in which aggregates are tested with the aim to detect defects caused by errors made during **implementation**

→ concern is the functionality of the aggregate (unit testing) or the interaction of its parts (interface testing)

integration testing

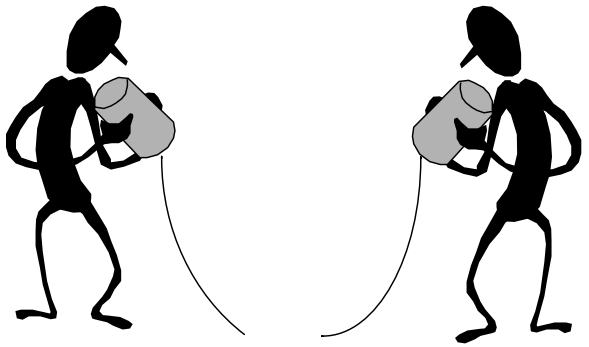
→ testing in which aggregates are tested with the aim to detect defects caused by errors made during **integration**, e.g.

- building
- writing scripts (function test of scripts)
- integration of components to tiers and these to system
- integration of components to subsystems and these to system
- configuration of the system
- installation of the system in the target environment

V Integration testing is interface testing (3)

type of errors integration testing is looking for

- wrong address
- wrong name used
- queue is not set-up
- queue is too small
- file is missing or is in wrong location
- processes are started in a wrong sequence
- a process is not started at all
- wrong setting of configuration parameters or no setting at all
- etc.



VI Test coverage is a glass box test concept (1)

a quite usual conversation ...

A quite usual conversation ...

A: How do you test your programs?
E: In the usual way, like anybody else.
A: I mean, how do you select the test cases that you intend to execute in order to torture your program?
E: Simple, that’s easy.
A: Good. Which method do you apply?
E: Method? I know what I need to test.
A: Of course you know it. I am interested to learn, when do you stop test case selection and specification?
E: When I have enough test cases.
A: Exactly, that is what I want to know, when do you have enough?
E: As soon as I don’t need any more.
A: Yes, of course, but how do you decide that you don’t need any more, that your set of test cases is complete?
E: Man, everybody knows that there is nothing like complete testing.
A: I am convinced there is.
E: Even if it were it’s too expensive, nobody can afford it ... and it doesn't work anyway.
A: Would you agree, then, that you test intuitively?
E: Yes, I do, and I am proud of it.

Example: Black-box test of the Windows clock

Example: A complete set of test cases (1)

	test cases		
output	1	2	3
analogue time display		X	
digital time display			X
font (28 types)		Arial	TnR
display of the Greenwich time		X	
display of the system time			X
display of the title bar		X	
no display of the title bar			X
display of seconds		X	
no display of seconds			X
display of the date			X
no display of the date		X	
display of information	X		

Example: A complete set of test cases (2)

analogue display of time: 8 test cases

	1	2	3	4	5	6	7	8
time display	gch	gch	gch	gch	sys	sys	sys	sys
title bar display	yes	yes	no	no	yes	yes	no	no
seconds display	yes	no	yes	no	yes	no	yes	no
date display	no	no	no	no	no	no	no	no

digital display of time: 448 test cases

date display is possible: doubles the analogue test cases = 16

28 font types available: 16 x 28 = 448

total: analogue display + digital display + info =
8 + 448 + 1 = 457 test cases

VI Test coverage is a glass box test concept (2)

first criterion (3 test cases)

⇨ for all possible types of display at least one of the possible outputs is produced by at least one test case

second criterion (457 test cases)

⇨ all possible combinations of outputs are produced by at least one test case

a possible criterion in between (30 test cases)

⇨ all possible outputs are produced by at least one test case

VI Test coverage is a glass box test concept (3)

- testing is a sampling procedure
  - the sample content depends on risks
    - the sample size is defined by the envisaged "confidence level" of the test

⇨ coverage defines the sample
  ⇨ coverage is a target for the test designer
    ⇨ coverage makes systematic test case selection possible
      ⇨ coverage determines the extent, thus also the cost of testing
        ⇨ coverage enables the project leader / software manager to (better) assess the state of affairs

VII Test planning is an easy task

– we do unit testing, integration testing, system testing

test planning involves

- identify system boundaries and system structure
- define strategy for reviewing, integration, and testing
- analyse risks
- define test objects
- for all test objects define the characteristics of the test
- design the test infrastructure and specify the test harness
- identify all testing activities and estimate the effort
- trade cost and benefit of the tests
- schedule test activities and assign resources

can't be all done at the beginning and not all of what can be done, can be defined with the same level of detail

Characteristics of a single test

test object	one executable unit (or many)
test level	unit or component or system or an aggregate in between
test environment	development or integration or test or production
error types to look for	logic, data entry, navigation, fault tolerance, connection, communication, response time, size, etc.
basis for test case specification	artefact used to gather information about possible test inputs and expected output
basis for test case selection	artefact used to define test coverage criteria used to assess the completeness of the selected test case set
test dimensions	configuration parameters of the run-time environment
test goal	extent of error type and test dimensions coverage
test execution	manually using a checklist, using test procedures, with automatic test logging, completely automated, etc.
tester	user, test engineer, ignorant, expert, etc.
test evaluation	compare with specification (basis), compare with assured results, etc.
test record	completed checklist, manual, test log, automatic test log, etc.

Example: System test planning with variations

WEB	dimension	possible values				cardinality
	OS	NT	2000	XP		3
	browser	Netscape	IE	Firefox		3
	registered user	no	yes			2
	locked user	no	yes			2
	user language	German	French	Italian	English	4

WAP	dimension	possible values				cardinality
	operator	we	foreign			2
	device brands	5 new	15 legacy			20
	registered user	no	yes			2
	locked user	no	yes			2
	user language	German	French	Italian	English	4

	WEB	WAP
minimal number of variations	4	20
theoretically maximal number of variations	144	640

A kind of conclusion ...

tesztelést fejtegettem
kevertem tekertem
remélem
elég értelmesen értékeltem
eme nemes cselekedést

esetleg nem érzékeltem
kedves teremben elhelyezettek
értékes nézetét
e tett szenvedésért
elnézést kérek

kellemes rendhelytelenség
kegyetlen keresését
tisztelt tesztelök