SIEMENS
Ingenuity for life

PLM and Innovation Excellence

Learning Campus

Your partner for Business Learning

Siemens Core Learning Program

Test Automation

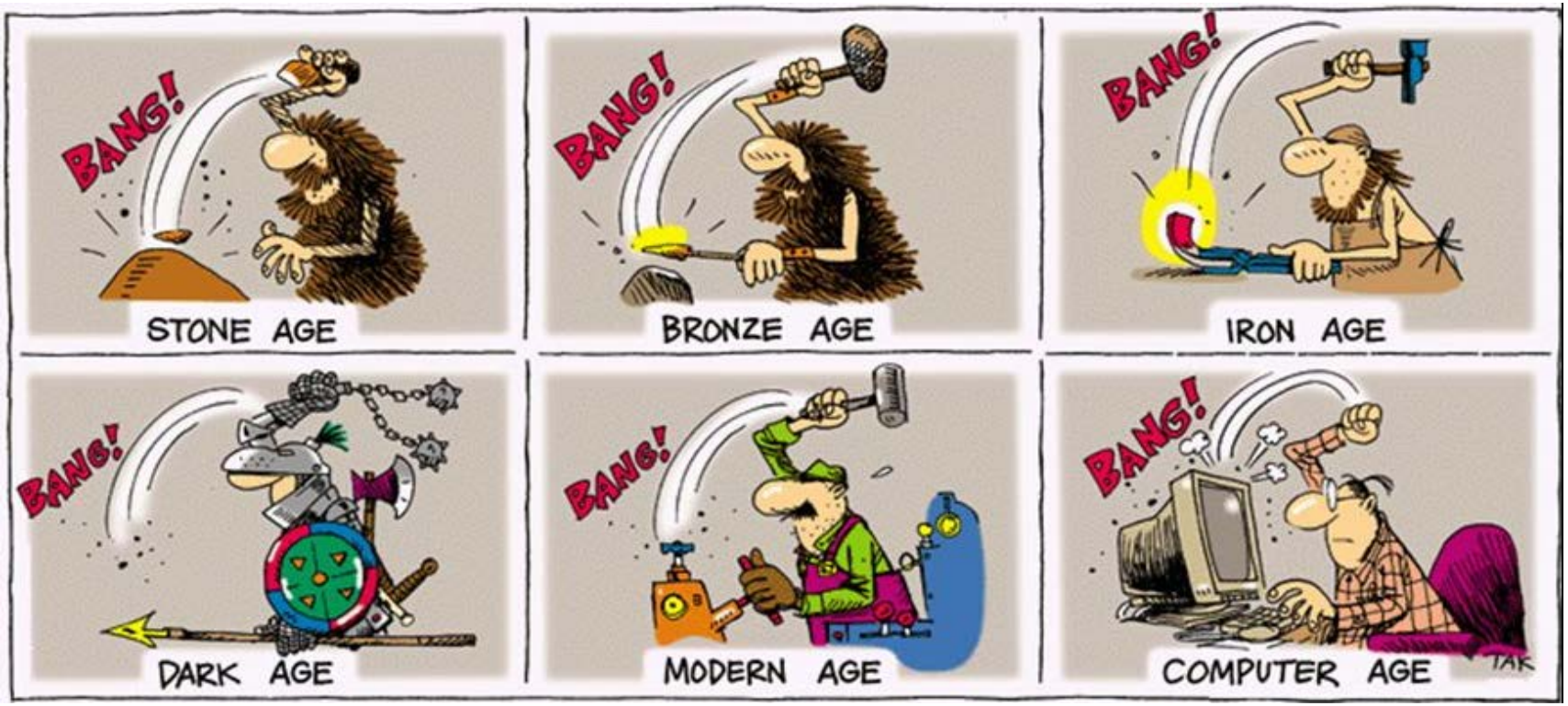
Author:
Peter Zimmerer, CT

SIEMENS
Ingenuity for life

Understand complexity and go for success

Introducing test automation is sometimes like a romance: Stormy, emotional, resulting in either a spectacular flop or a spectacular success.

Bogdan Bereza-Jarocinski, 2000



Learning objectives

- Understand the objectives and value of test automation
- Get to know different test automation strategies and approaches
- Learn how to define and drive a test automation approach

Test Automation

Agenda

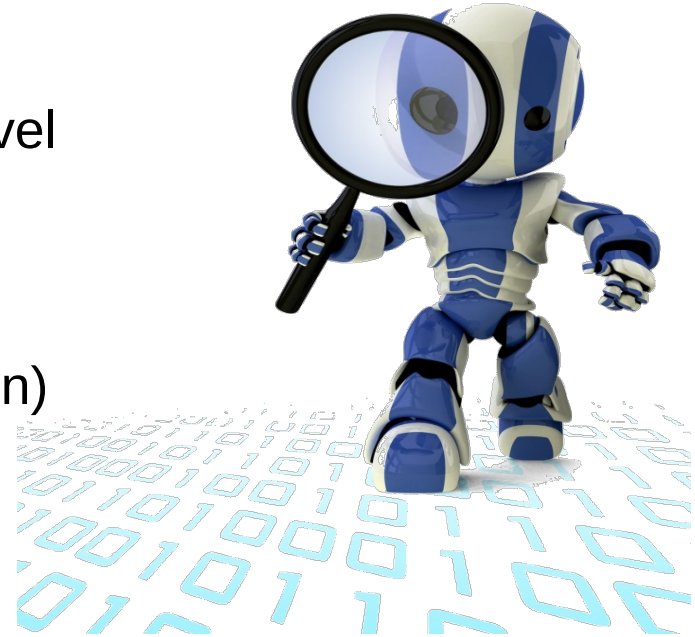
Objectives and Value

Strategies and Approaches

Frameworks and Technologies

Summary

Test automation – What?



Facets

- Support for any testing activity at any testing level
  - Generation of test input data
  - Observation of memory management at runtime
  - Defect tracking
- Automated test (case) execution (and evaluation)
  - Capture / replay
  - Automation of a manual tester

Realization

- Batch scripts, programs, set-up procedures, support tools (e.g. diff)
- Testing tools
  - Commercial
  - Open source
  - In-house development

Elements of test automation – *SEARCH*

***S*etup** is the effort it takes to bring the software to a point where the actual test operation is ready for execution

***E*xecution** is the core of the test – the specific steps necessary to verify functionality, sufficient error handling, or some other relevant task

***A*nalysis** is the process of determining whether the test passes or fails; this is the most important step – and often the most complicated step of a test

***R*eporting** includes display and dissemination of the analysis, for example, log files, database, or other generated files

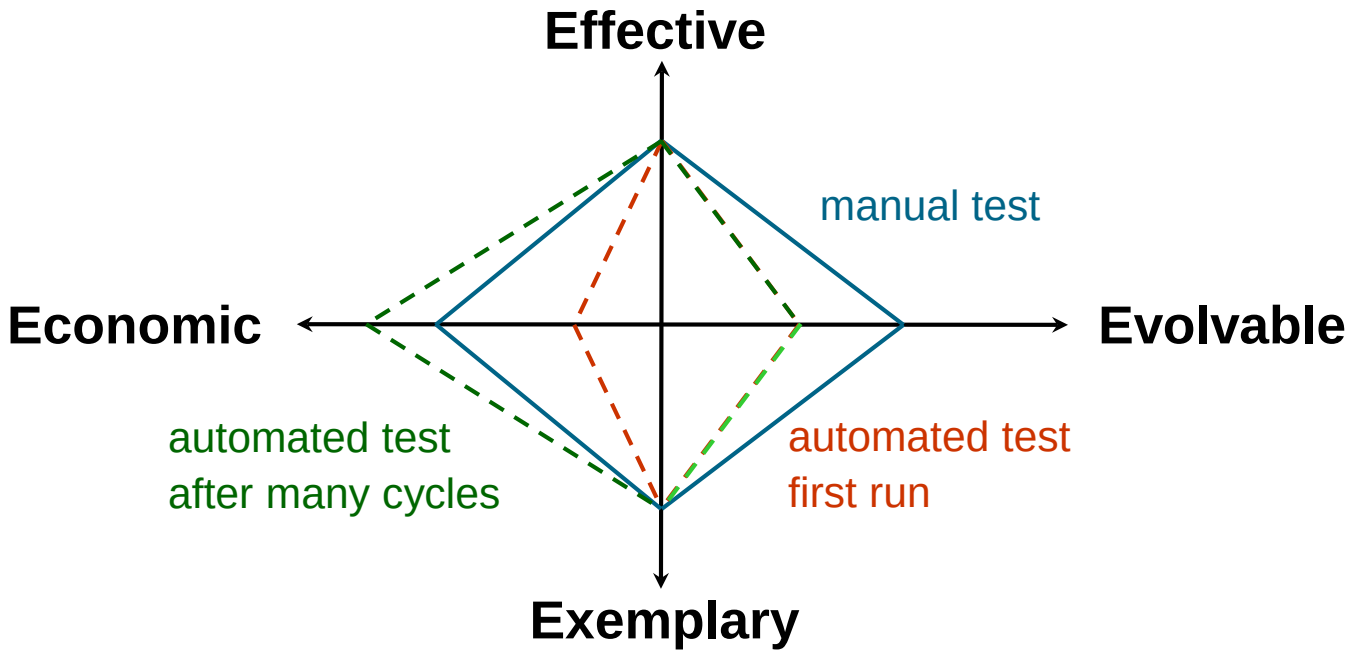
***C*leanup** returns the software to a known state so that the next test can proceed

A ***H*elp** system enables maintenance and robustness of the test case throughout its life

Reference: Keith Stobie and Mark Bergman: How to Automate Testing: The Big Picture, 1992

Test automation – Characteristics of tests

Effective – Reasonable probability for detecting defects
Exemplary – Practical, low redundancy, represents others
Economic – Reasonable cost to develop and perform, cheap to use
Evolvable – Easy to maintain, to change, to adapt



Reference: D. Graham, M. Fewster

SIEMENS
Ingenuity for life

Test automation – Questionable objectives

Find more bugs

Run more tests more often (continuously, overnight, on weekends)

Reduce testing staff

Reduce elapsed time for testing

Automate X% of testing

→ *Testing* and *automation* are different and distinct activities
→ Don't confuse objectives for *testing* with objectives for *automation*

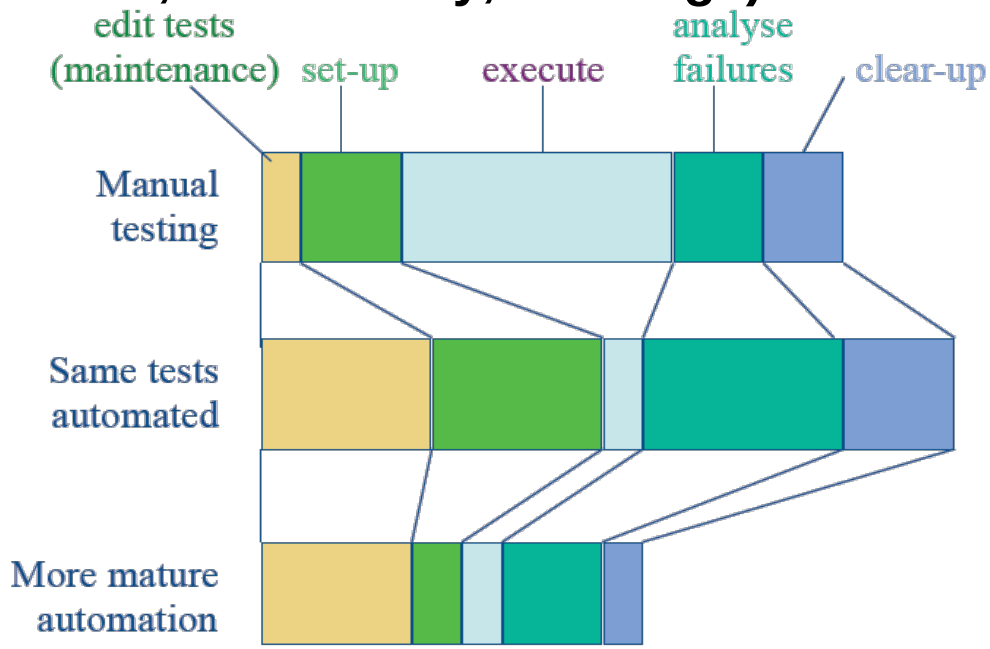
Test automation – Better objectives (1)

Find more *regression* bugs (dependent on *quality* of the tests)

Run the *most important, useful, valuable* tests more often

Control cost of automation effort vs. effort saved by automation (→ increase in execution speed, times run, data variety, coverage)

Reduce elapsed time for *all tool-supported* testing activities: set-up, execute, analyze failures, clear-up, maintenance



Reference:
Dorothy Graham, Mark Fewster:
That's No Reason to Automate! Why Good Objectives Are Critical to Test Execution Automation Better Software, July/August 2009

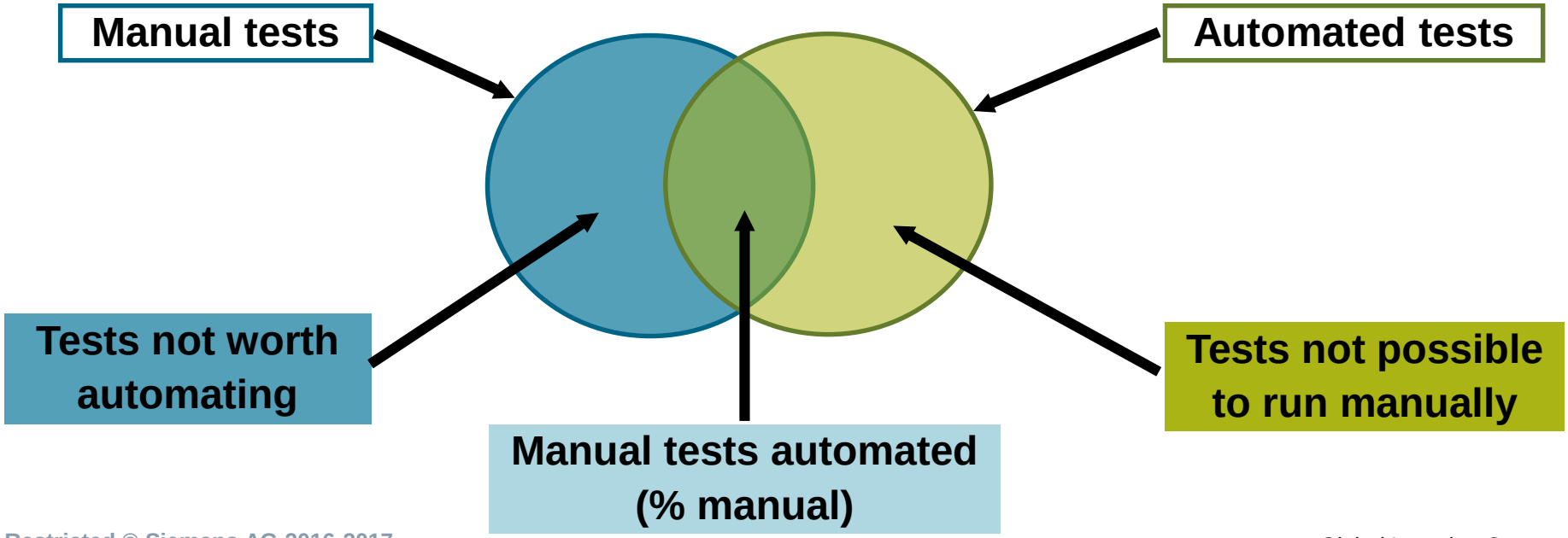
Test automation – Better objectives (2)

SIEMENS
Ingenuity for life

Automation should provide valuable support to testing

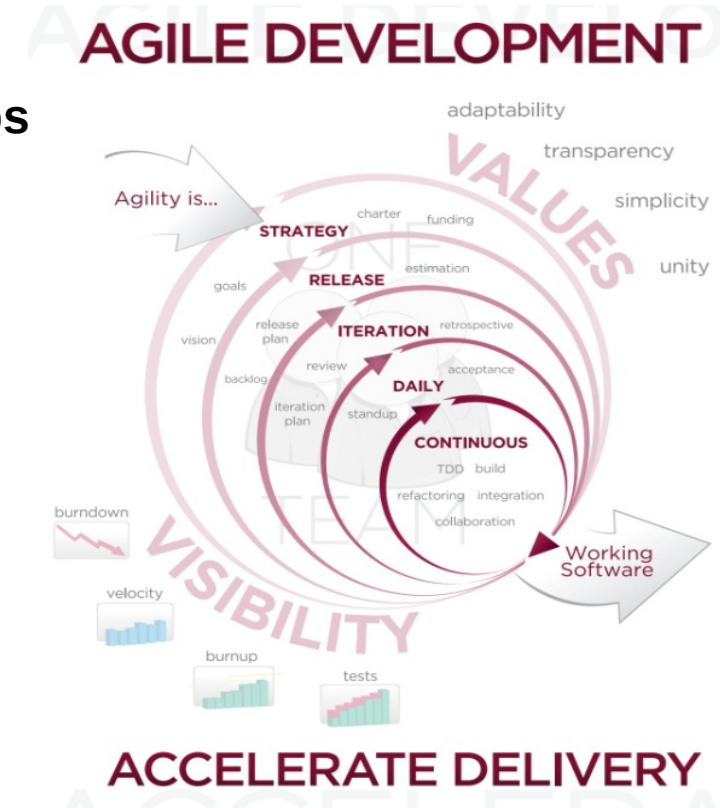
- For example increase number of additional verifications / check points made that couldn't be checked manually
- Automation as a lever to improve effectiveness and efficiency in testing

→ **Automating waste doesn't reduce it, it just hides it**



SIEMENS
Ingenuity for life

Test automation – Why? – When?



Automated testing is the foundation for any kind of iterative, incremental, agile development up to DevOps

- Daily builds and small releases are useless if they cannot be tested
- Speed up development, early and often feedback
- Continuous testing in DevOps

An efficient quality check of software is not possible without the *right* test automation!

- Changing customer requirements, changing code
- Last-minute checks after patches
- Regression testing, safety net
- Cost-effective maintenance, refactoring activities
- Precondition for specific testing activities: white-box coverage, load test, endurance test, test data setup, etc.

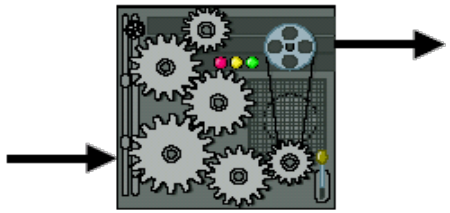
Cost reduction not impossible, but generally only medium-term …
Carefully decide on start / extent of test automation implementation

Test automation strategy

Focus on *risk reduction, coverage,* and *speed of development* instead of *(test-) cost reduction*
No 100% automation – manual and automated testing are different!

TestAutomationROI Example

Testability is the key to cost-effective test automation
Testability is often a better investment than automation



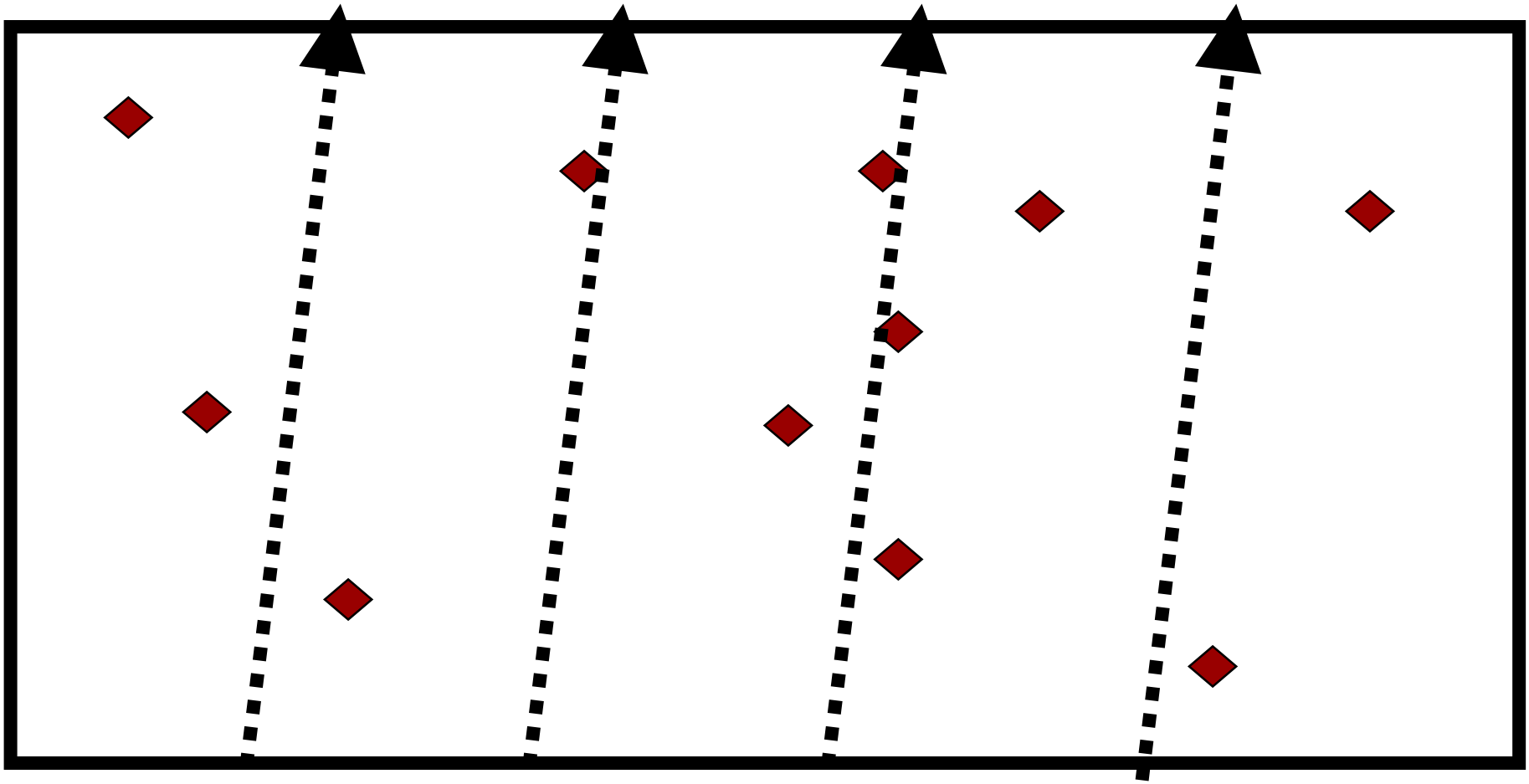
- Test environment: stubs, mocks, fakes, dummies, spies
- Test oracle: assertions (design by contract)

→ Implemented either *inside the SUT* or *inside the test automation*

Areas of test automation

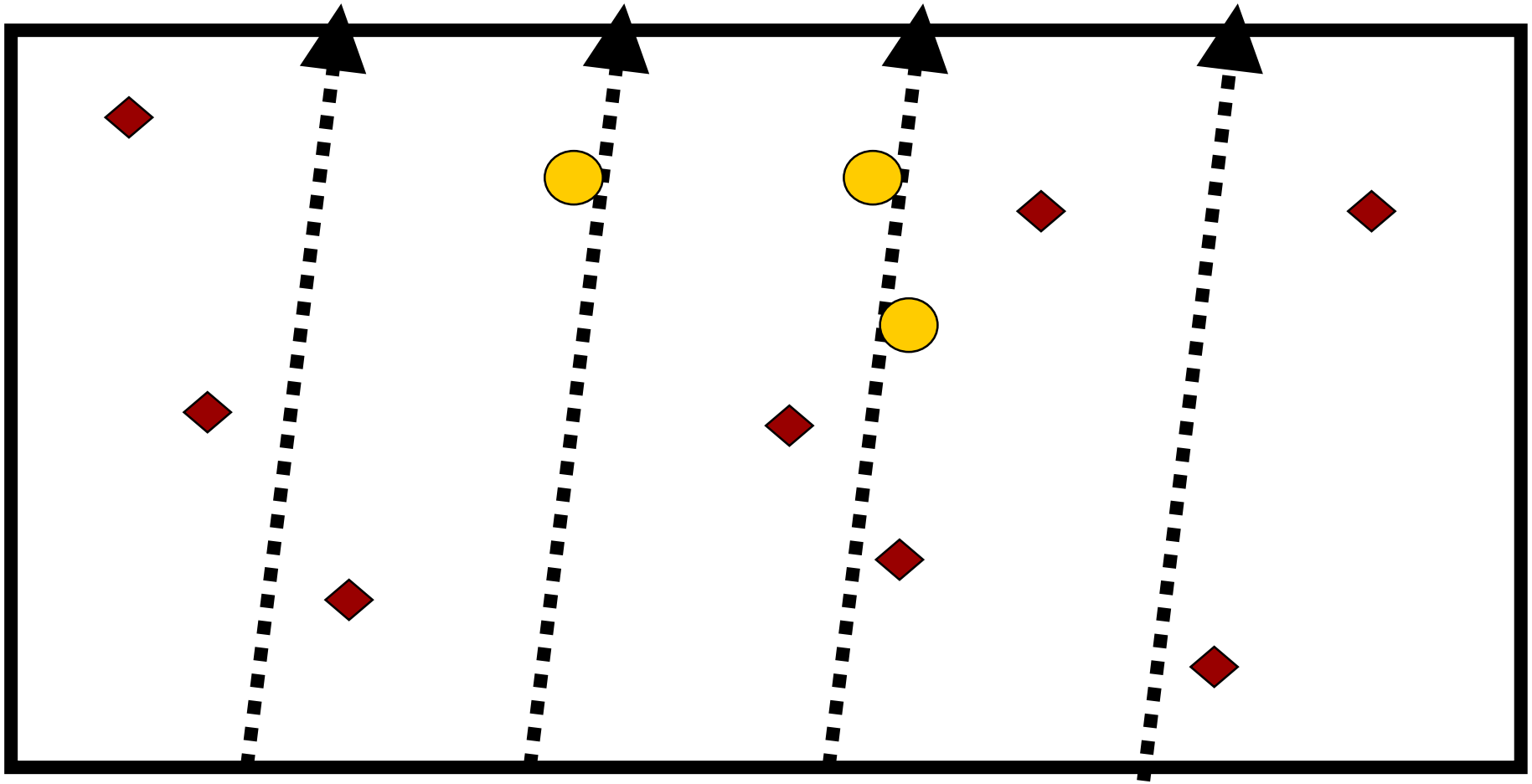
- Testing that requires **repeated effort** of similar test cases (or activities)
  - Regression testing
  - Smoke testing, configuration testing, performance / load / endurance testing
- Needed where **manual tests are impossible** (or very difficult or expensive)
  - Test data setup and administration, system configuration setup, result analysis
  - Fast tests, precise tests, distributed tests, coordinated tests, remote tests

Test automation – Limitations
Example: Minefield metaphor



◆ **Mines**

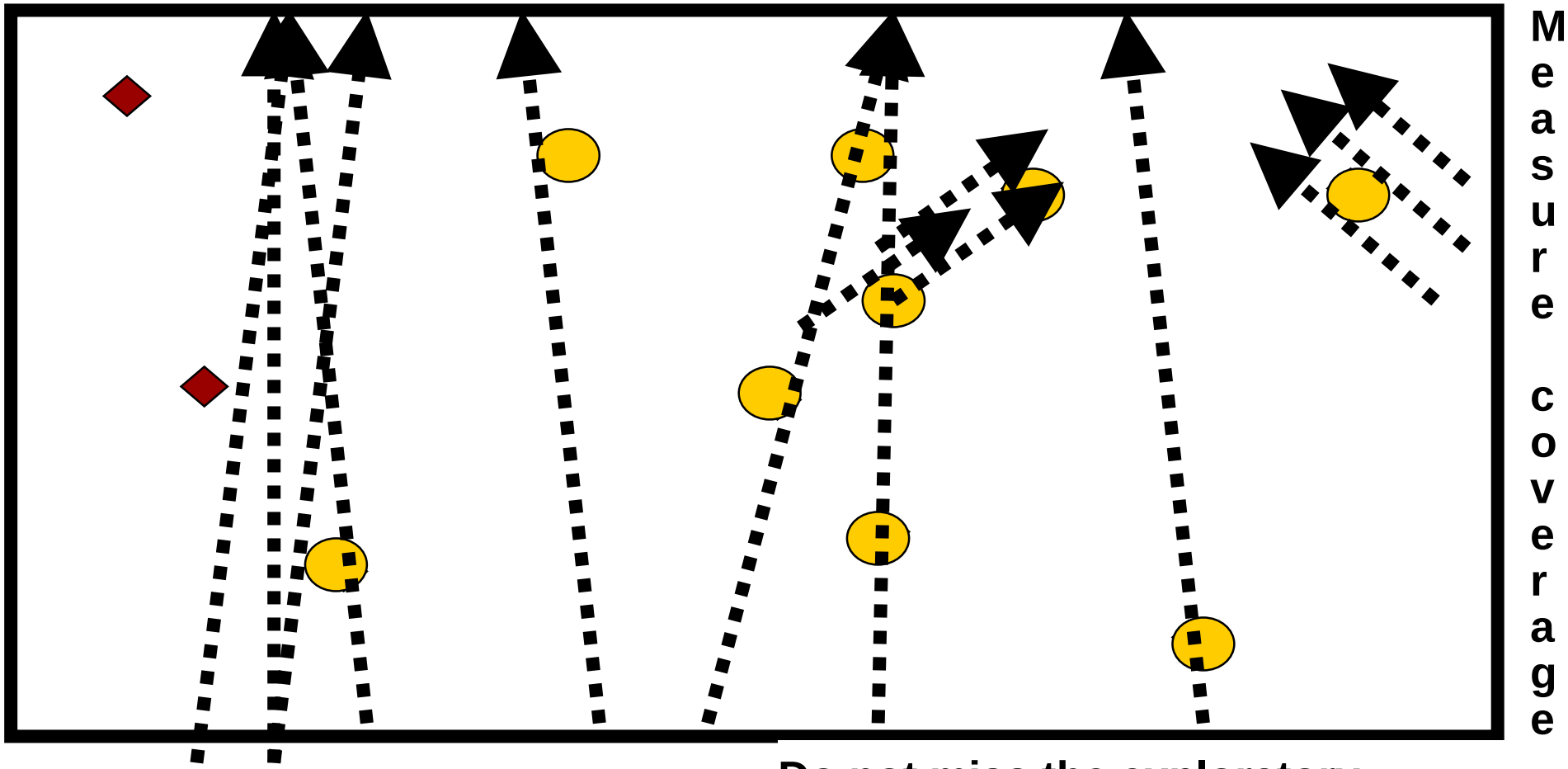
Totally repeatable tests won't clear the minefield …



Mines Fixes

Sometimes variable tests (manual tests or / and new automated test cases) are therefore more effective …

SIEMENS
Ingenuity for life



Mines

Fixes

Do not miss the exploratory, investigative mission of testing

Test Automation

Agenda

Objectives and Value

Strategies and Approaches

Frameworks and Technologies

Summary

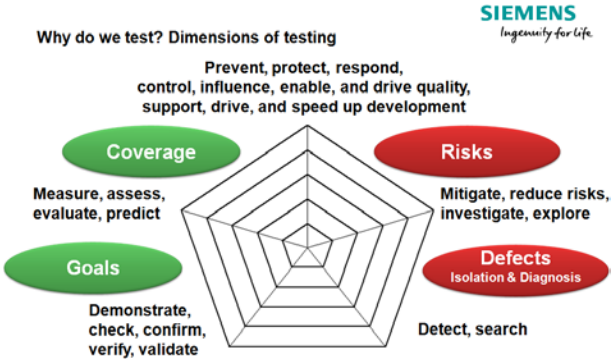
Test automation – Factors to consider

Effort, cost – ROI, level of testing (API ←→ GUI), test oracles (verification points), testability, test environment, test harness, test tools

Test lifetime – frequency of usage, reuse, lifespan of system under test, change, maintenance, likelihood for regression

Benefits, value – finding (new) bugs ←→ regression tests

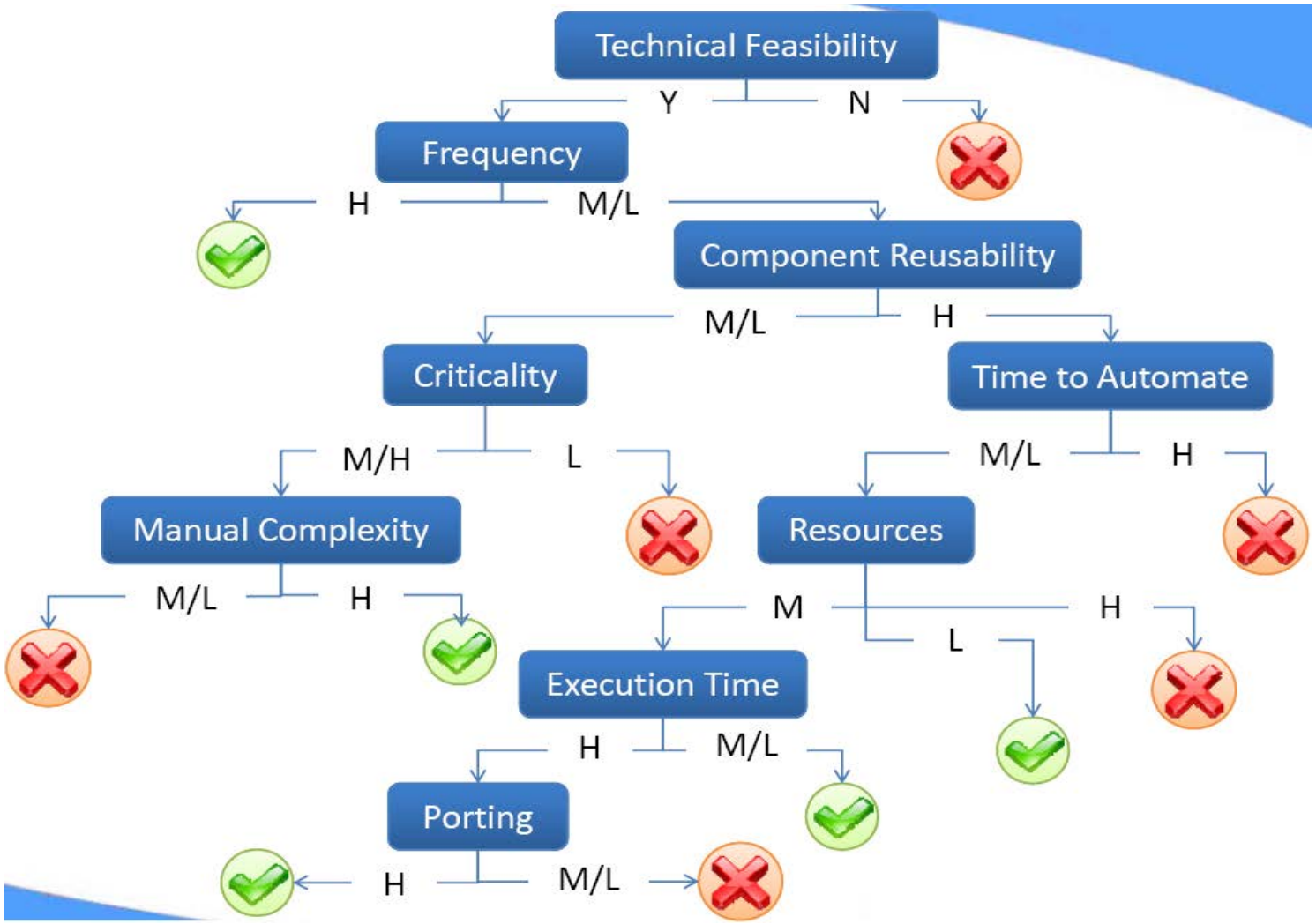
Point of involvement – the earlier the better, late start is risky



Accuracy – difficulties in analyzing results, complaints on reporting false positives (failure is caused by a bug somewhere in the test code rather than in the system under test code), expected results are ambiguous, needs a human brain to interpret the data (usability, intuitiveness)

System context – legacy code, 3rd party software, ever-changing code

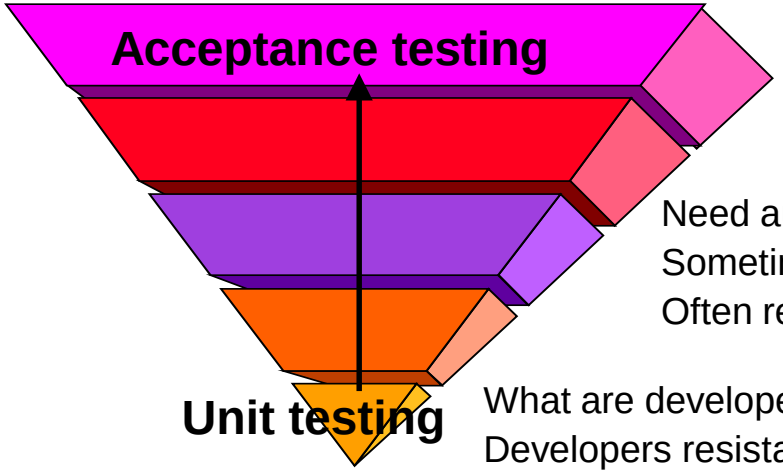
Choosing the right test cases for automation



Reference: Pradeep G, Cognizant Technology Solutions, StarWest Conference 2009

Test automation strategy:
Change / improve your test *execution* pyramid

SIEMENS
Ingenuity for life



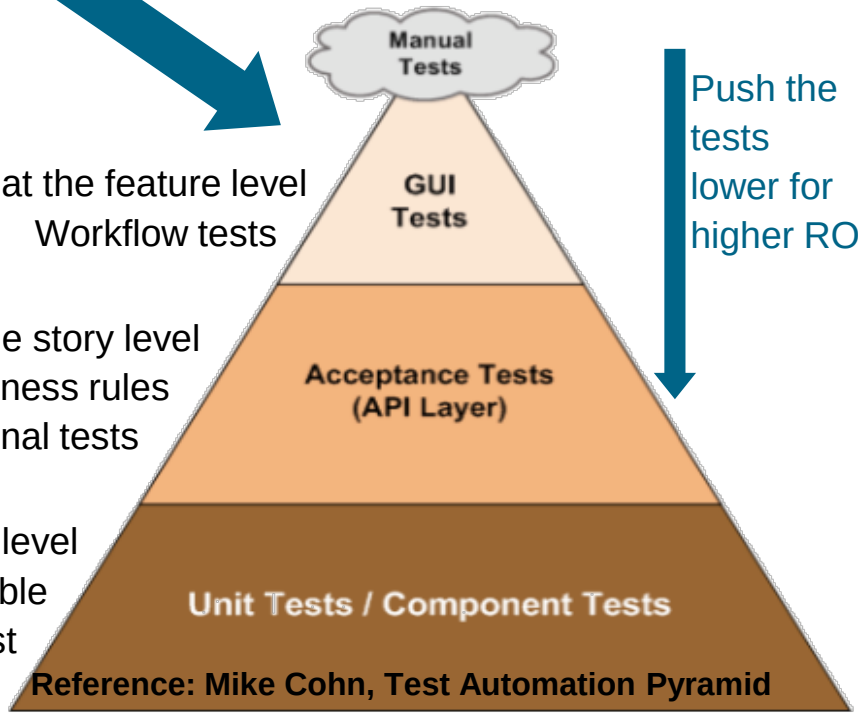
Easy to create, very familiar – what we always do
Typically tedious, manual
How do we know coverage?

Need automation specialists
Sometimes fragile, maintenance problems
Often rewrite, change, adapt

What are developers testing at all?
Developers resistance to writing tests

Testing *the* GUI ✓

~~**Testing *through* the GUI**~~



Push the tests lower for higher ROI

Automate at the feature level
Workflow tests

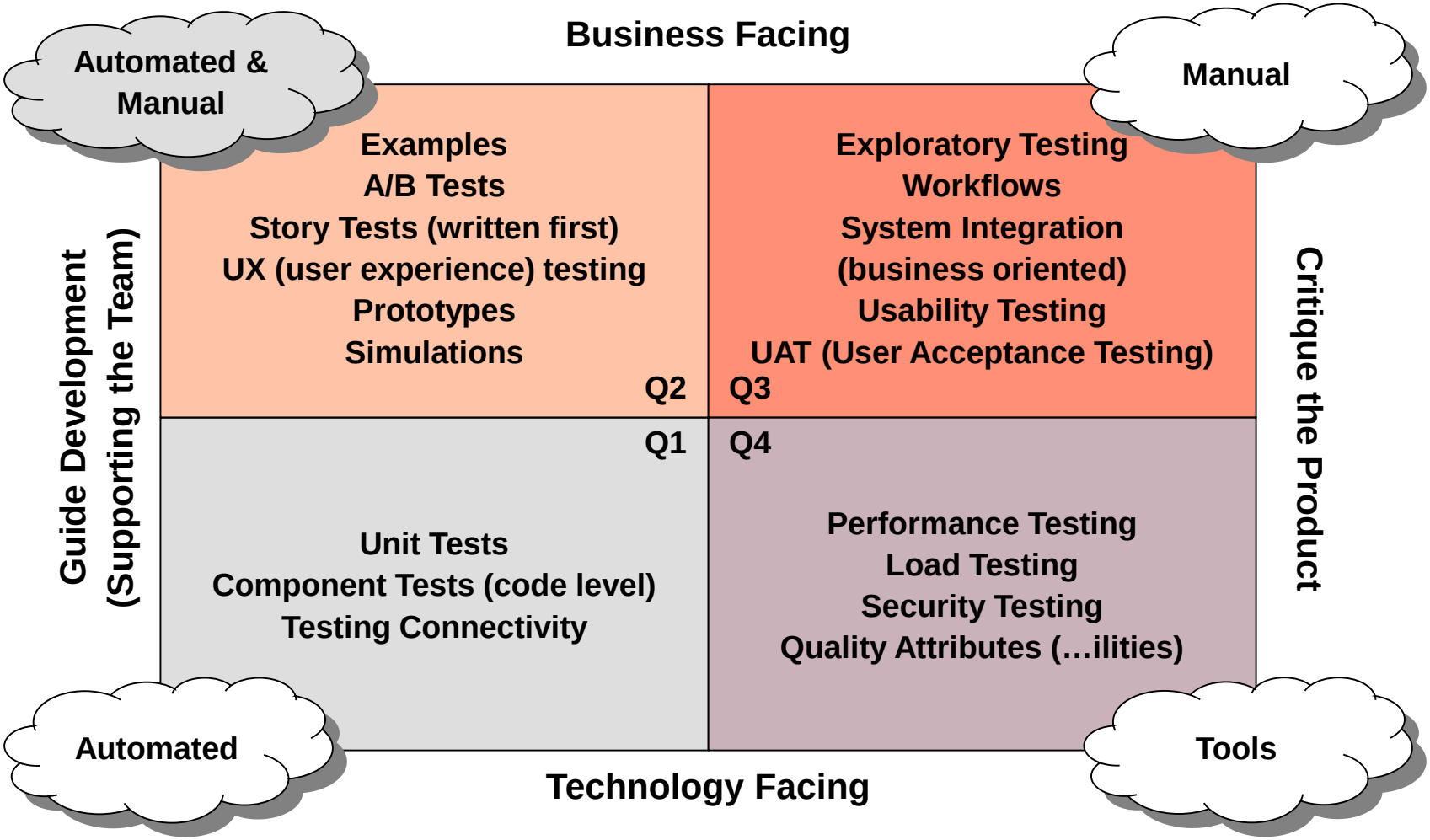
Automate at the story level
Service layer, business rules
Functional tests

Automate at the task level
Features under test more stable
Do these the most

Reference: Mike Cohn, Test Automation Pyramid

Tetris principle:
Test as low as possible

Agile testing quadrants



Reference: Brian Marick, Lisa Crispin, Janet Gregory

Automated build, test, deploy, deliver → DevOps

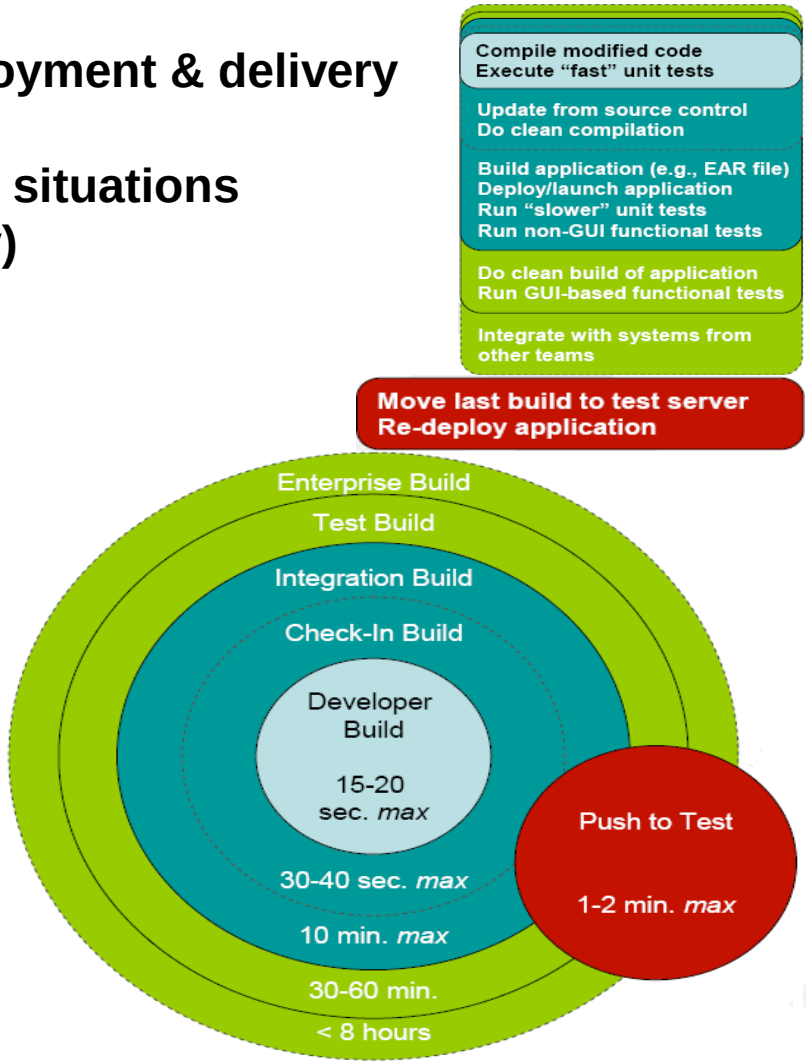
Daily build, continuous integration & test & deployment & delivery

- Tools for example Ant, Maven

You need different builds, optimized for different situations

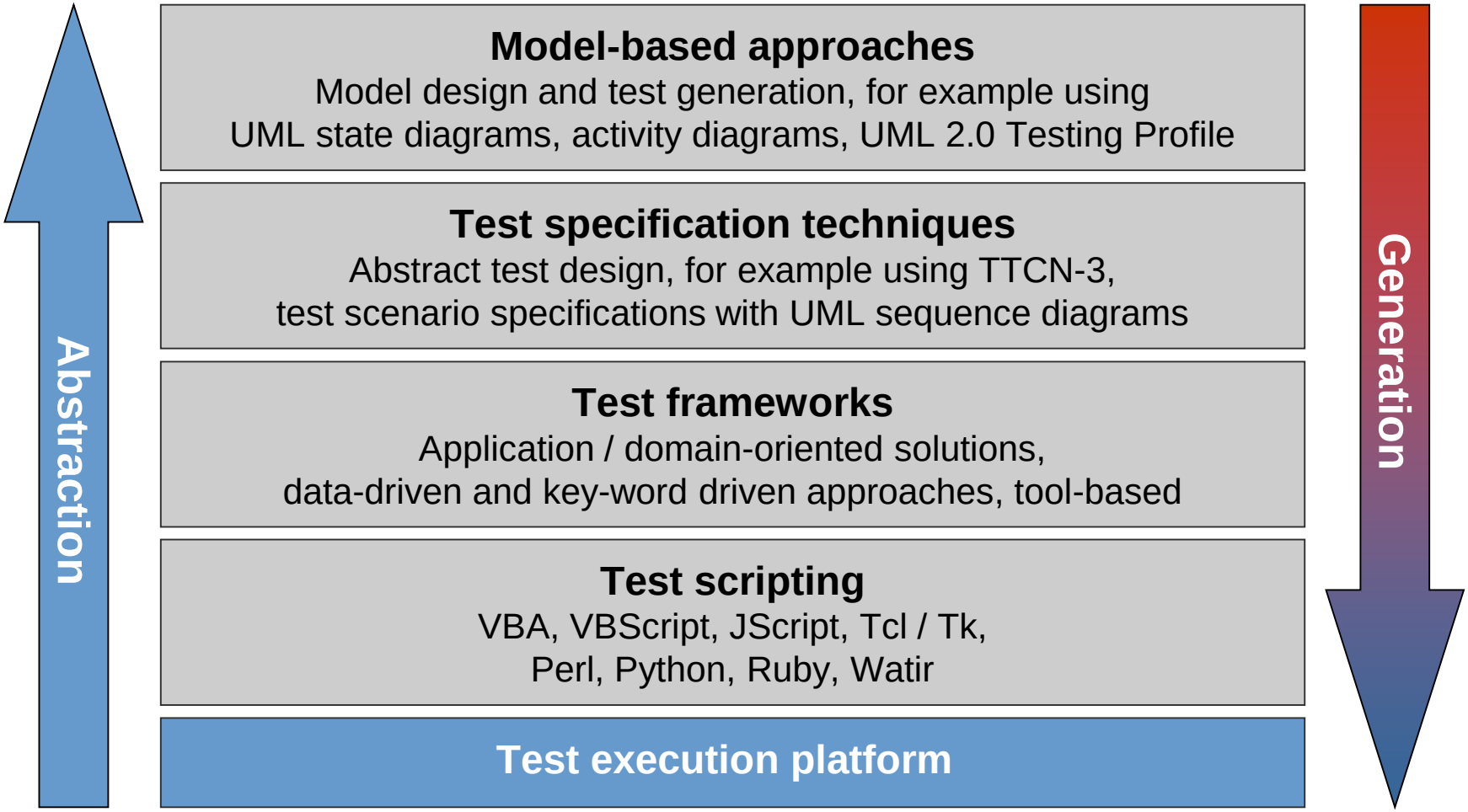
***Build, test, deploy, deliver* (solar system analogy)**

- Developer build
  - Performed "on demand" on local machine
- Check-in build
  - Performed "on demand" at each check-in
- Integration build
  - Kicked off automatically on each check-in
- Test build
  - Performs additional tasks to ensure testable system
- Enterprise build
  - For a large system with multiple teams
- Push to test
  - Kicked off on demand or at pre-scheduled times; draws from last successful test or integration build

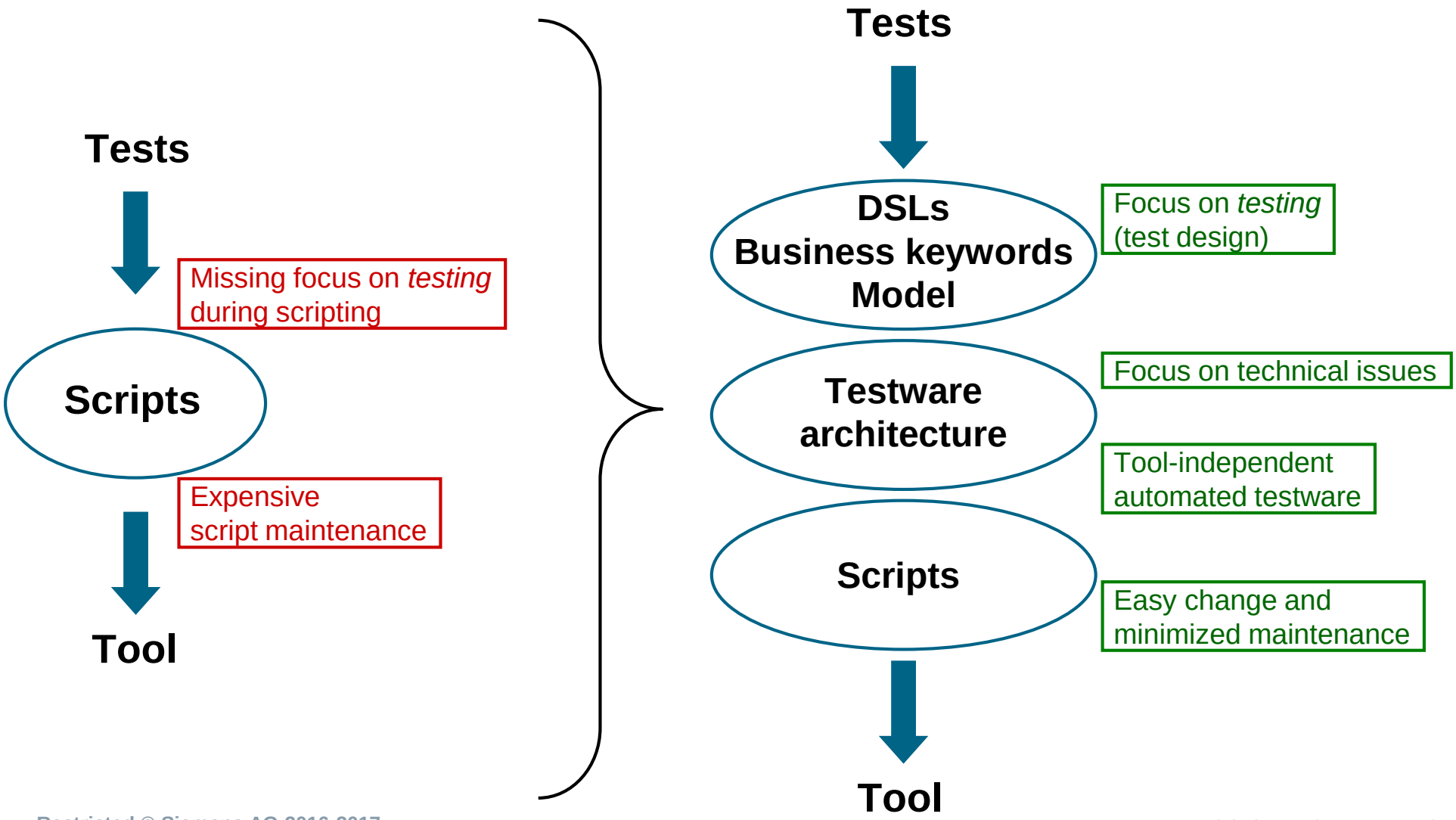


Ref.: Jeff Nielsen (Digital Focus), David DeWolf (2006)

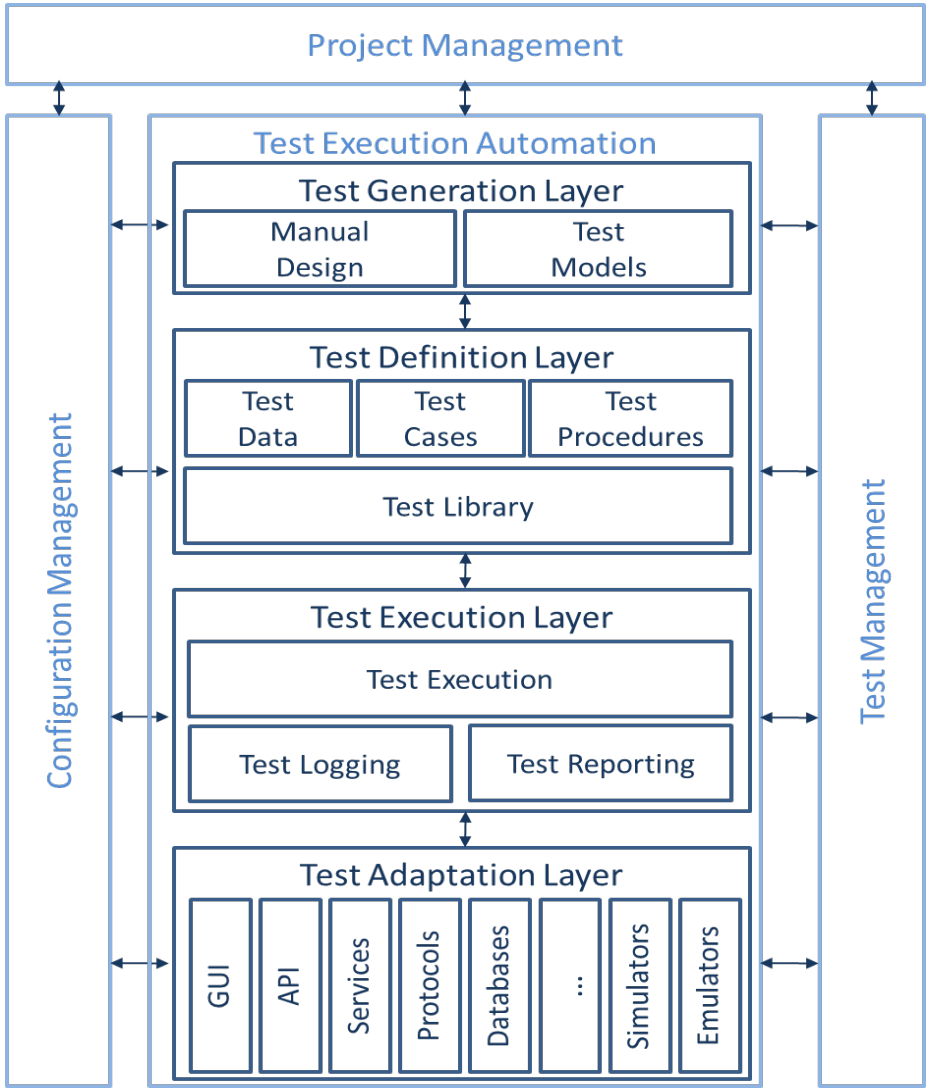
Test automation is software development



Levels of abstraction



ISTQB Generic test automation architecture (gTAA)



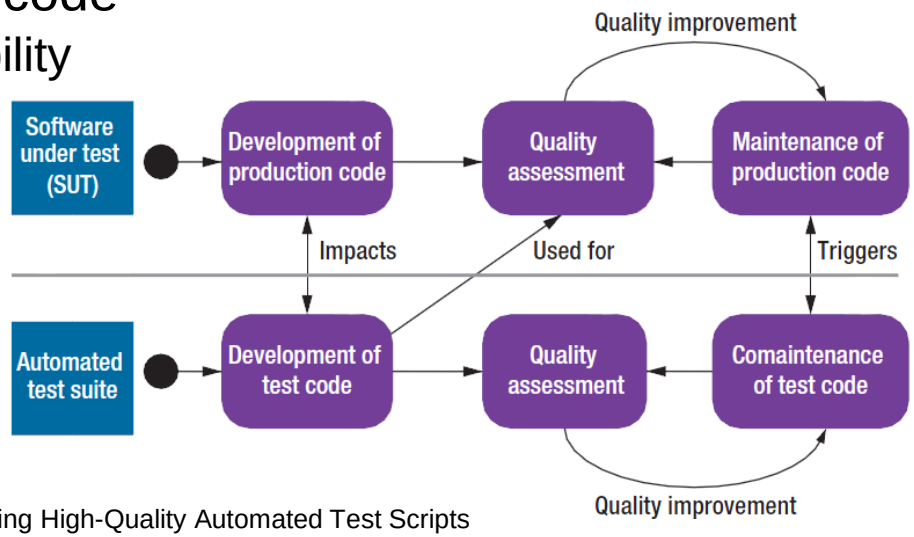
http://www.istqb.org/

Software Test Code Engineering (STCE)

SIEMENS
Ingenuity for life

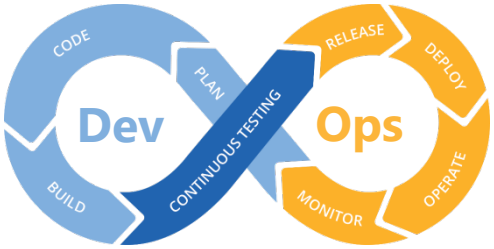
End-to-end test script engineering and test script management

- Use (test) patterns as guidelines to ensure quality
- Functional-quality attributes of test code
  - Correctness in properly testing the SUT
  - Effectiveness in fault detection (→ assess and verify test suite quality)
    - If the test case fails, does the SUT really have a fault?
    - If the SUT has a fault, does the test suite detect it?
- Nonfunctional-quality attributes of test code
  - Maintainability, understandability, readability
  - Reliability
  - Test smells, e.g. test redundancy
- Comaintenance
  - Test antipatterns
  - Determine test case sensitivity
  - Minimize coupling with SUT



Reference: Vahid Garousi, Michael Felderer: Developing, Verifying, and Maintaining High-Quality Automated Test Scripts
IEEE Software, Vol. 33, No. 3, May / Jun 2016: 68-75

New directions in test automation



SIEMENS
Ingenuity for life

Virtualization – hardware, OS, services (application behavior)

- Speed up test setup, configuration, execution, and reporting (virtual test lab)
- Environment virtualization: service, network, data, event, sensor
- Better test reporting and reproducibility: attach context to bugs by snapshots
- More efficient use of hardware resources
- Isolate and virtualize slices of specific behavior (service virtualization)
- Rapidly define and provision test environments (virtual assets)
- But consider impact and influence on performance and timing issues

Continuous testing is *the* enabler for DevOps

Cloud testing: testing *in* the cloud, *from* the cloud, *of* the cloud

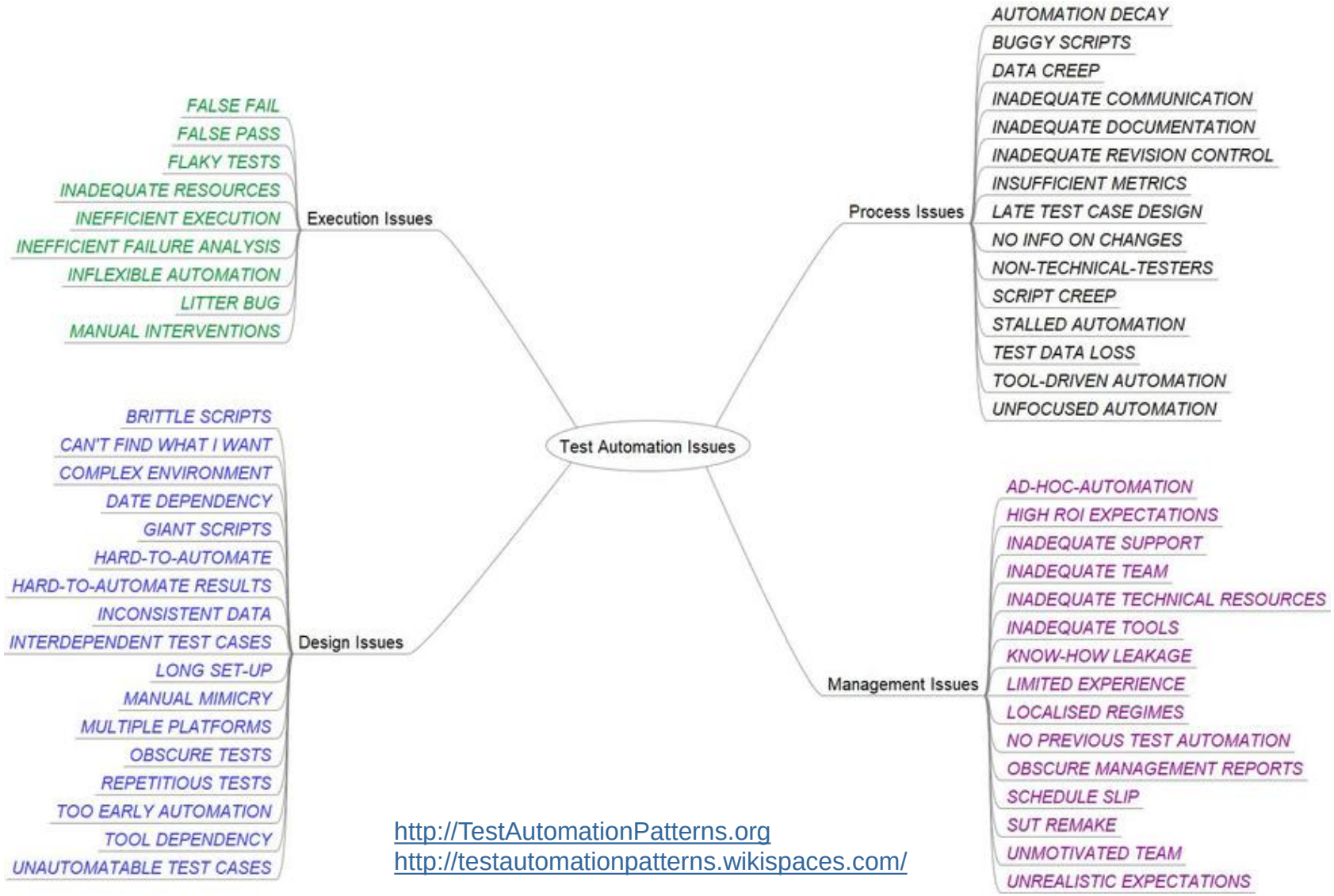
- Leverage cloud computing environments for testing – migrate testing *to* the cloud

Crowd testing (in-the-wild testing, virtualization of test teams)

- Example: Applause (http://www.utest.com/, http://www.applause.com/)
  On-demand testing via a global community of professional testers
- Sourcing relevant people from within your company across disciplines and levels; sourcing end users with specific domain knowledge and user environments

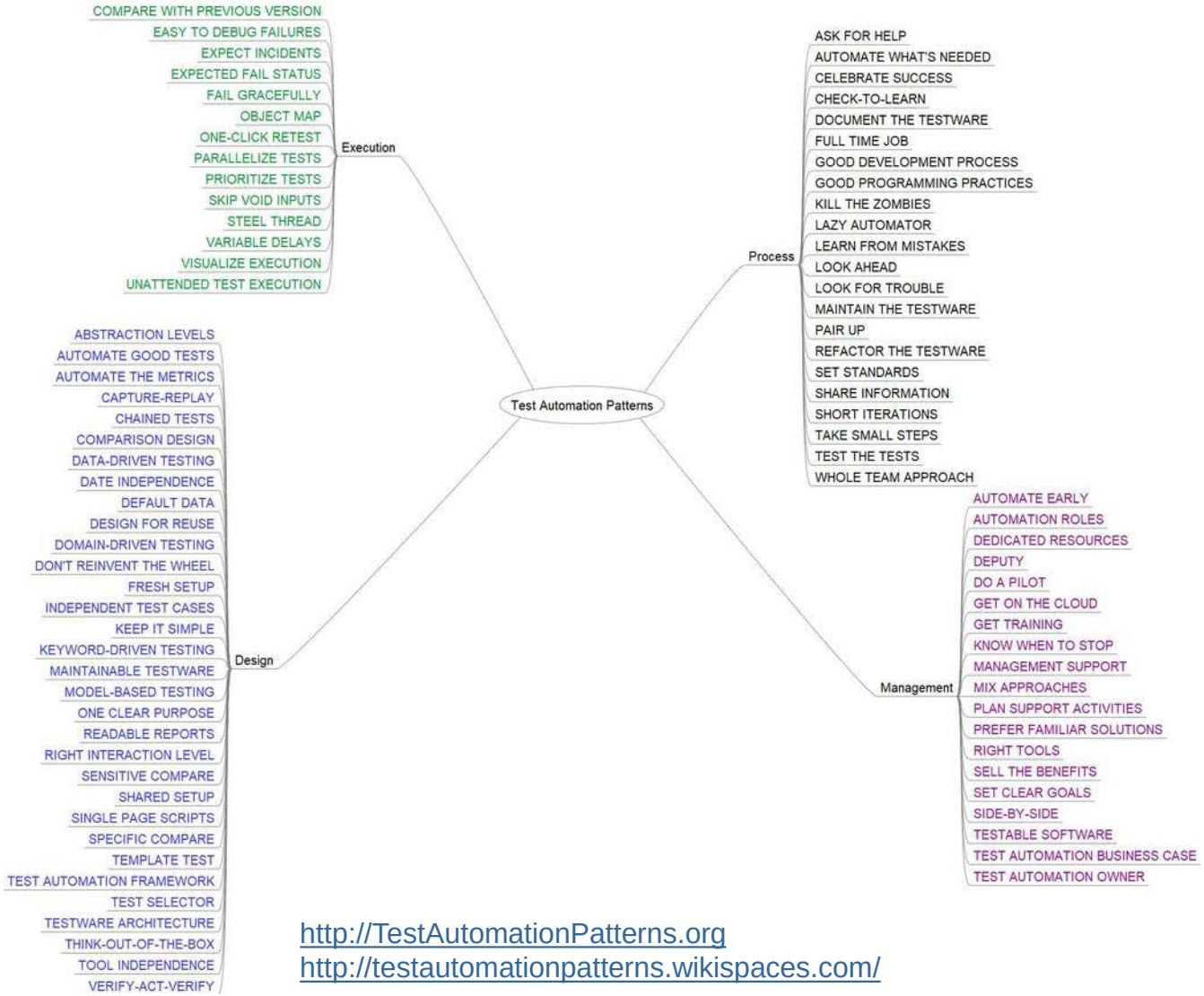
Test automation patterns wiki (1)
Test automation issues

SIEMENS
Ingenuity for life



http://TestAutomationPatterns.org
http://testautomationpatterns.wikispaces.com/

Test automation patterns wiki (2)
Test automation patterns



http://TestAutomationPatterns.org
http://testautomationpatterns.wikispaces.com/

Test Automation

Agenda

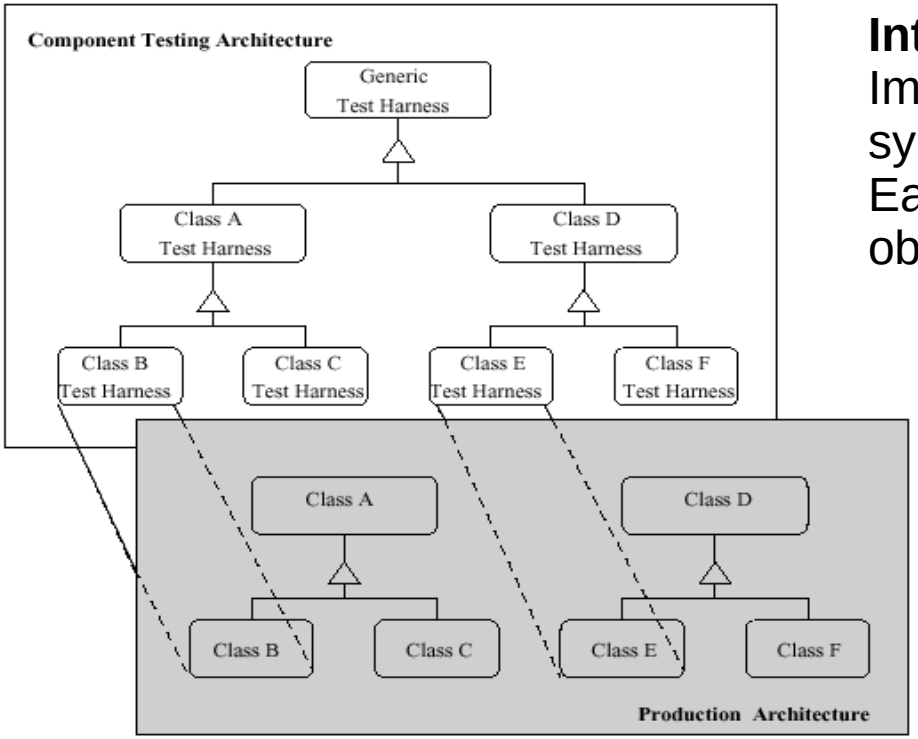
Objectives and Value

Strategies and Approaches

Frameworks and Technologies

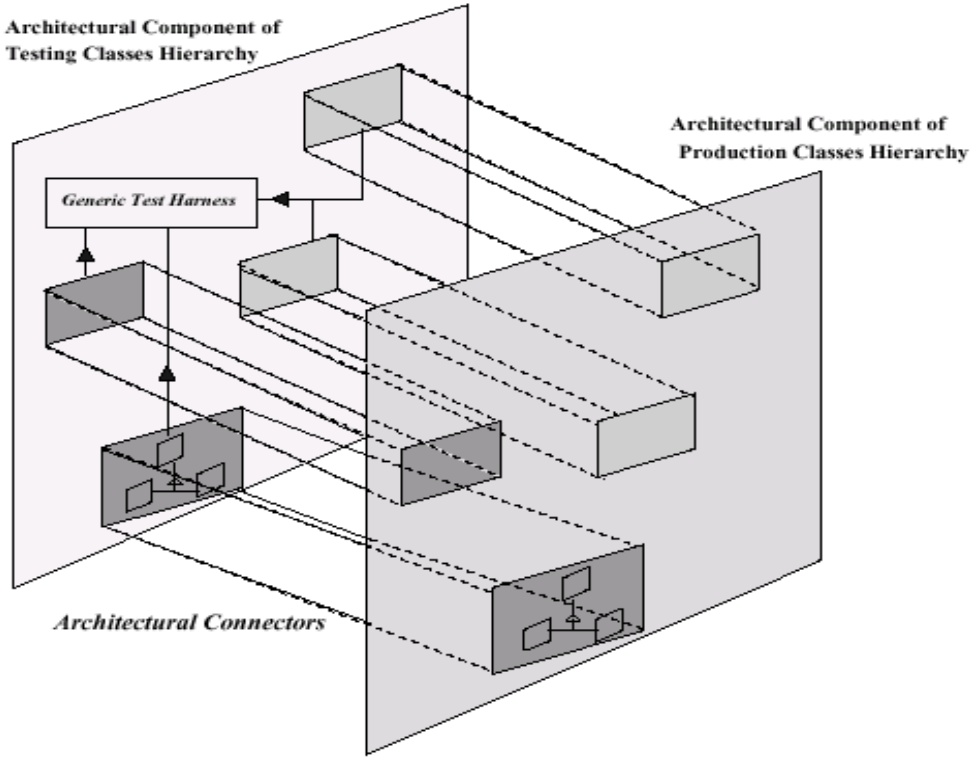
Summary

Parallel Architecture for Component Testing (PACT)



Intent

Implement a driver as a class hierarchy that is symmetric to the hierarchy of the class under test. Each driver class sends a test message to an object of the class under test.



Applicability

This approach is useful at class and cluster scope for unit and integration testing of an application class hierarchy.

Reference: John D. McGregor Clemson University, South Carolina

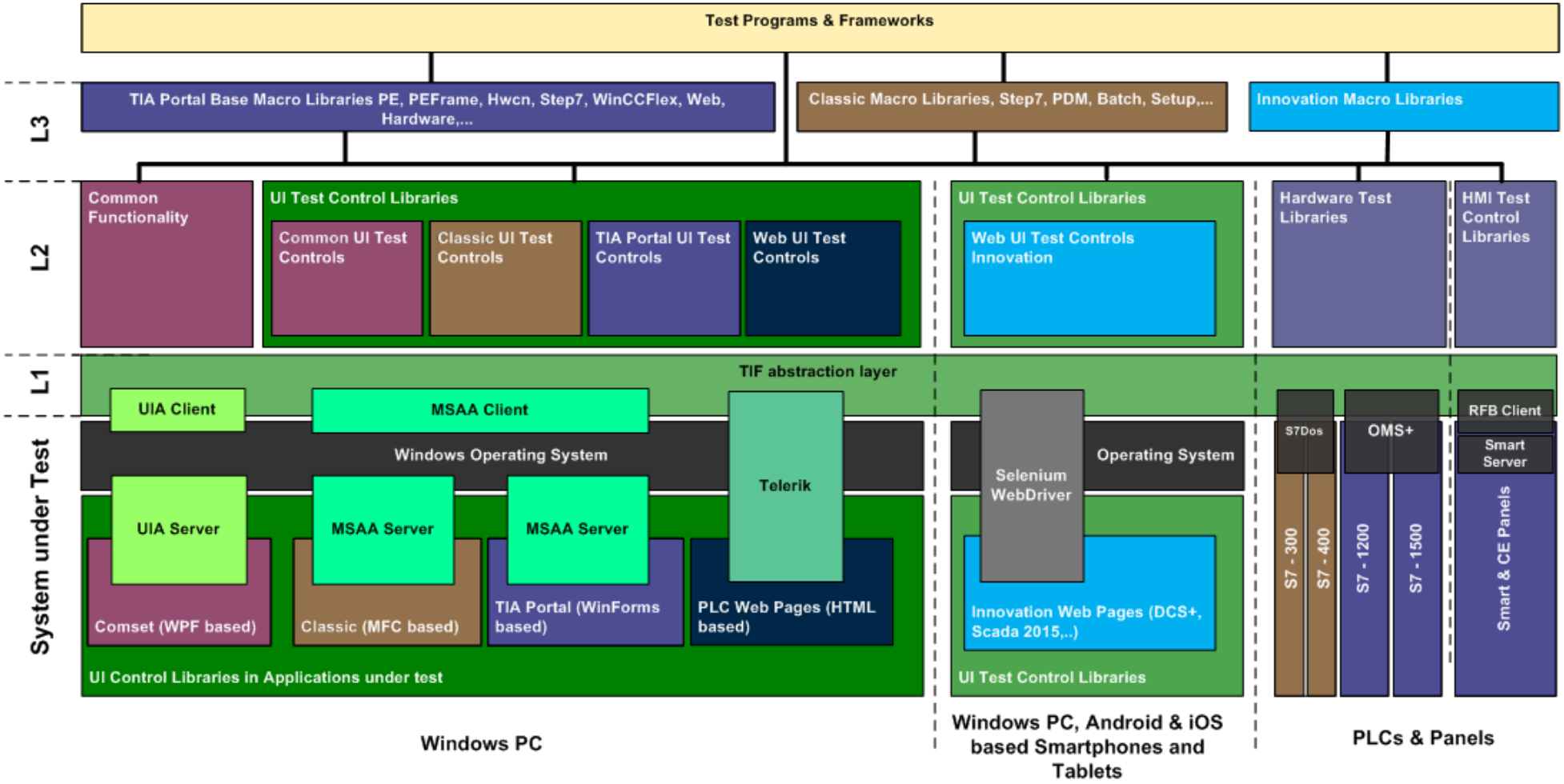
Example: TifLib (Test Interface Library)

- TifLib – Test Interface Library
- License free, standard test automation library at Siemens
- Provides functionality to simulate user interaction
- 3 usage areas of the TifLib:

TIA & Classic TifLib (windows based)	Innovation TifLib (web based)	HW TifLib (without UI)
▪ TIA Portal ▪ SIMATIC PCS 7	▪ DCS+ ▪ SCADA 2015 ▪ Unity UI (MES)	▪ PLCs

Example: TifLib (Test Interface Library) Architecture TifLib

SIEMENS
Ingenuity for life



Example: TifLib (Test Interface Library)
Overview TifLib (based on TIA Portal)

SIEMENS
Ingenuity for life

Execution environment
(TA-Suite, SI-Framework, Nuint,…)

Test programs
(programmed or projected tests using the execution environments)

TifLib Layer 3
(combination of L2 functionality to use case oriented macros)

TifLib Layer 1+2
(1:1 association between TIA UI Controls and UI Test Controls)

MSAA
(MS technology for UI automation of **WinForms** based apps)

TIA Portal

- Provides mouse and keyboard functionality
- Support for custom controls (TIA > 95%)
  - More efficient tests
  - Higher product quality
- Development, maintenance and training at one central place (TA Team)

Example: TifLib (Test Interface Library)
Overview TifLib (based on TIA Portal)

SIEMENS
Ingenuity for life

Execution environment
(TA-Suite, SI-Framework, Nuint,…)

Test programs
(programmed or projected tests using the execution environments)

TifLib Layer 3
(combination of L2 functionality to use case oriented macros)

TifLib Layer 1+2
(1:1 association between TIA UI Controls and UI Test Controls)

MSAA
(MS technology for UI automation of **WinForms** based apps)

TIA Portal

- User oriented macros
- Reduces the complexity
  - Saves time
  - Increases robustness
- Stable tests
  - Most test interface relevant changes can be handled within the macros
- Including full synchronization

Example: TifLib (Test Interface Library) Comparison Layer 2 and Layer 3

SIEMENS
Ingenuity for life

Layer 2: Create project

```
//Create project Layer 2
//L2 - Open "Project new"- dialog
TifPEMainMenuControl mainMenu = TifPEMainMenuControl.Find("MainMenu");
TifPEMenueItemControl projectNewItem = mainMenu.FindItemByPath(
    "Siemens.Automation.FrameApplication:Siemens.Automation.FrameApplication.Menu.MenuService:MenuFileResX\\" +
    "Siemens.Automation.FrameApplication:Siemens.Automation.FrameApplication.Menu.MenuService:MenuFileNewResX");
projectNewItem.MouseLeftClick();

//Wait for "Project new" dialog
TifPEWindowControl projectNewWindow = TifPEWindowControl.Find("DialogFrameControl", 20000);
projectNewWindow.BringToForeground();
TestInterface.SetSearchWindow("DialogFrameControl");

//Fill dialog
TifPETextBoxControl projectNameTextBox = TifPETextBoxControl.Find("Edit_ProjectName");
projectNameTextBox.SetText("L2DemoProject");

TifPETextBoxControl projectPathTextBox = TifPETextBoxControl.Find("Edit_Path");
projectPathTextBox.SetText(@"D:\02_WorkTemp\TifLibDemo");

TifPEButtonControl createButton = TifPEButtonControl.Find("CreateNewProjectDialog.Create");
createButton.MouseLeftClick();
```

Layer 3: Create project

```
//Create Project Layer 3
var projectInfo = new TifProjectInfo("L3DemoProject", @"D:\02_WorkTemp\TifLibDemo");
step7App.Project.Create(projectInfo);
```

SIEMENS
Ingenuity for life

What is TTCN-3?

Testing and Test Control Notation, version 3

- Internationally standardized abstract test specification and test implementation language from ETSI (European Telecommunications Standards Institute)
- Developed based on the experiences from previous TTCN versions

Specifically designed for testing and certification

- Built-in language features
- Data templates allow structuring and reusability of test data
- Matching mechanism to compare expected and received data
- Message-, procedure-based, and real-time ports
- Parallel test components
- Alternative and concurrent behavior, test verdicts, timers

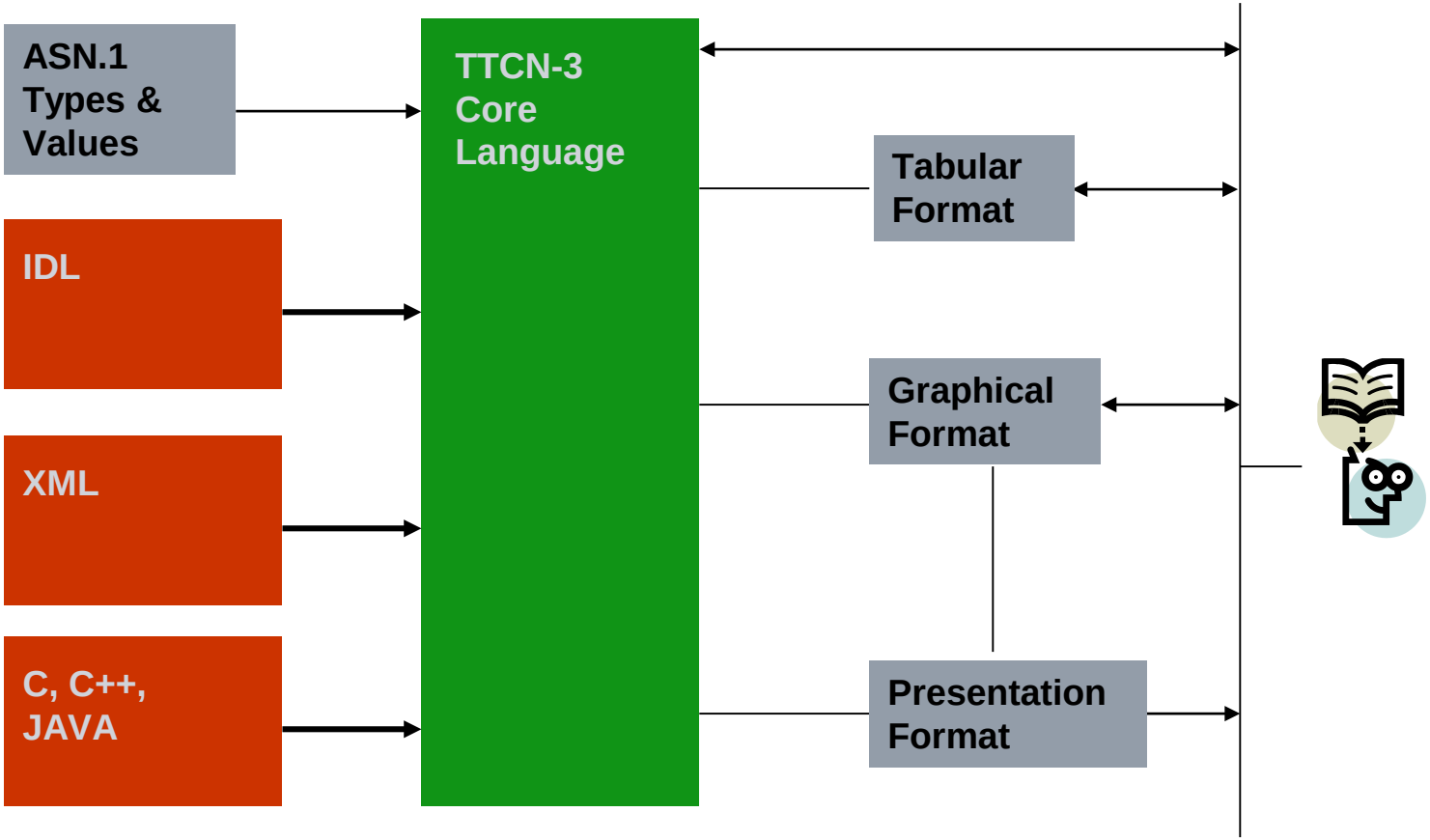
Applicable for all kinds of black-box testing for reactive and distributed / concurrent systems (communication behavior)

- Telecom systems (ISDN, GSM, UMTS)
- Internet (has been applied to IPv6)
- CORBA, CAN, MOST, SW platforms

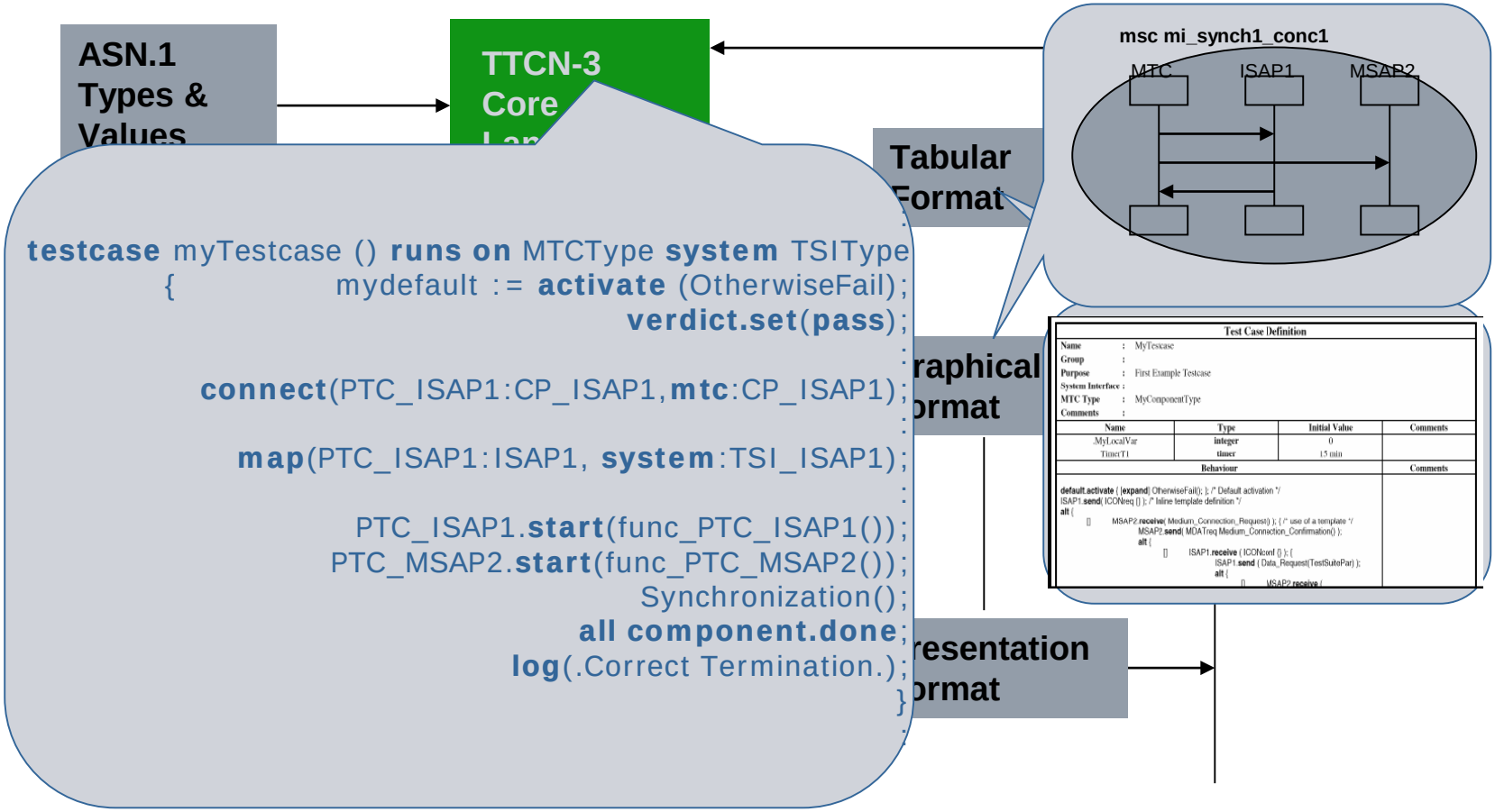
TTCN-3

http://www.ttcn-3.org/

TTCN-3 overview



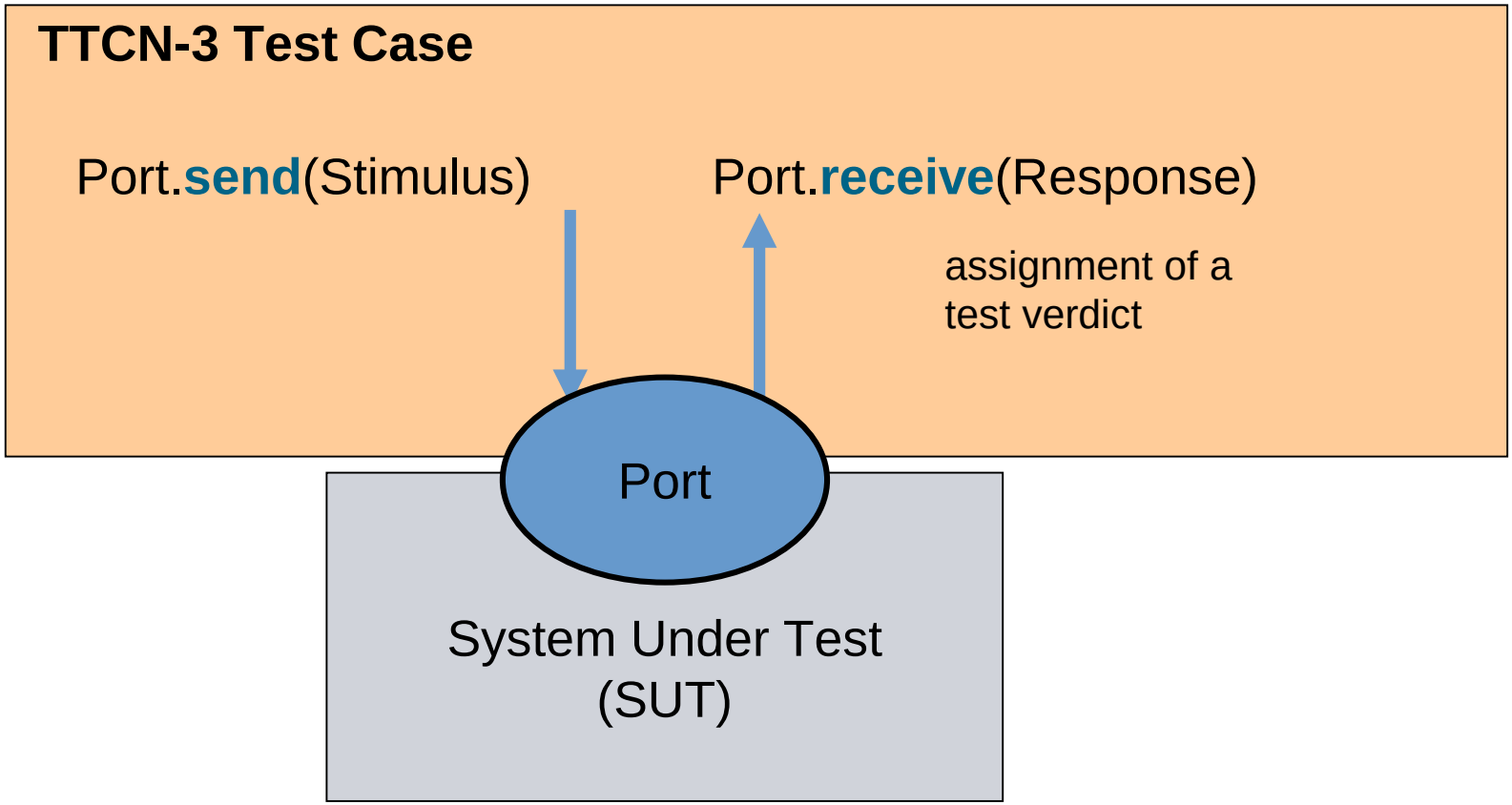
TTCN-3 overview



Hello World in TTCN-3

```
testcase TC_HelloWorld_1 () execute on HelloWorld_Config
    {
        hw.call(hello, 0.1){
            []hw.getreply("Hello World") {setverdict(pass)}
            []hw.catch(hello, hello_exc){setverdict(fail)}
            []hw.catch(timeout) {setverdict(inconc)}
        }
    }
```

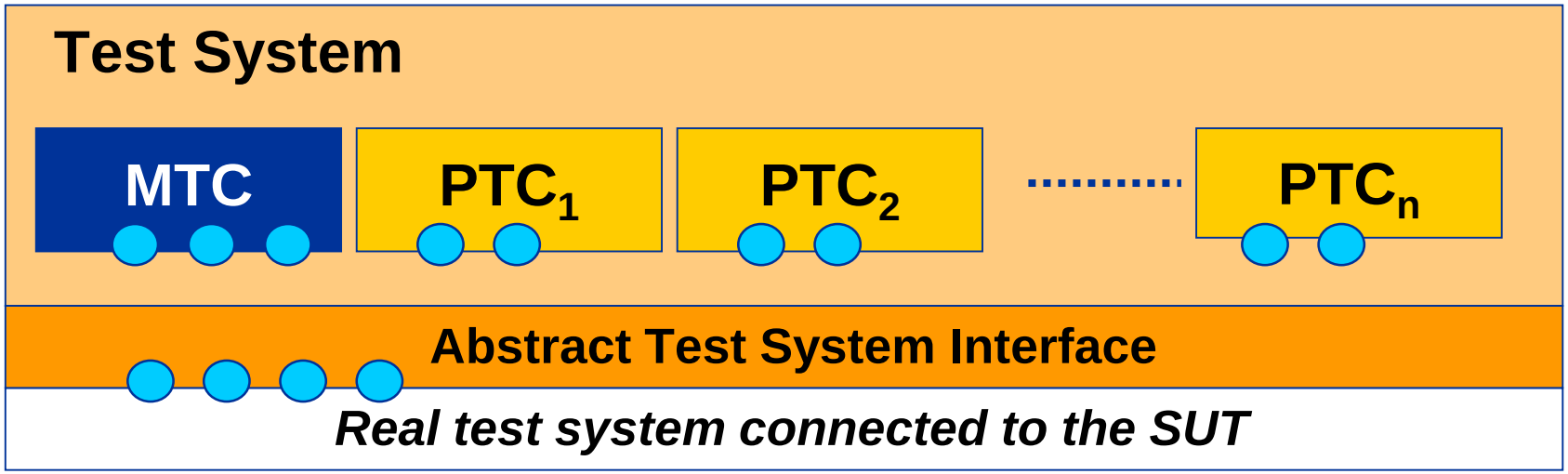
Concepts – Black-box testing



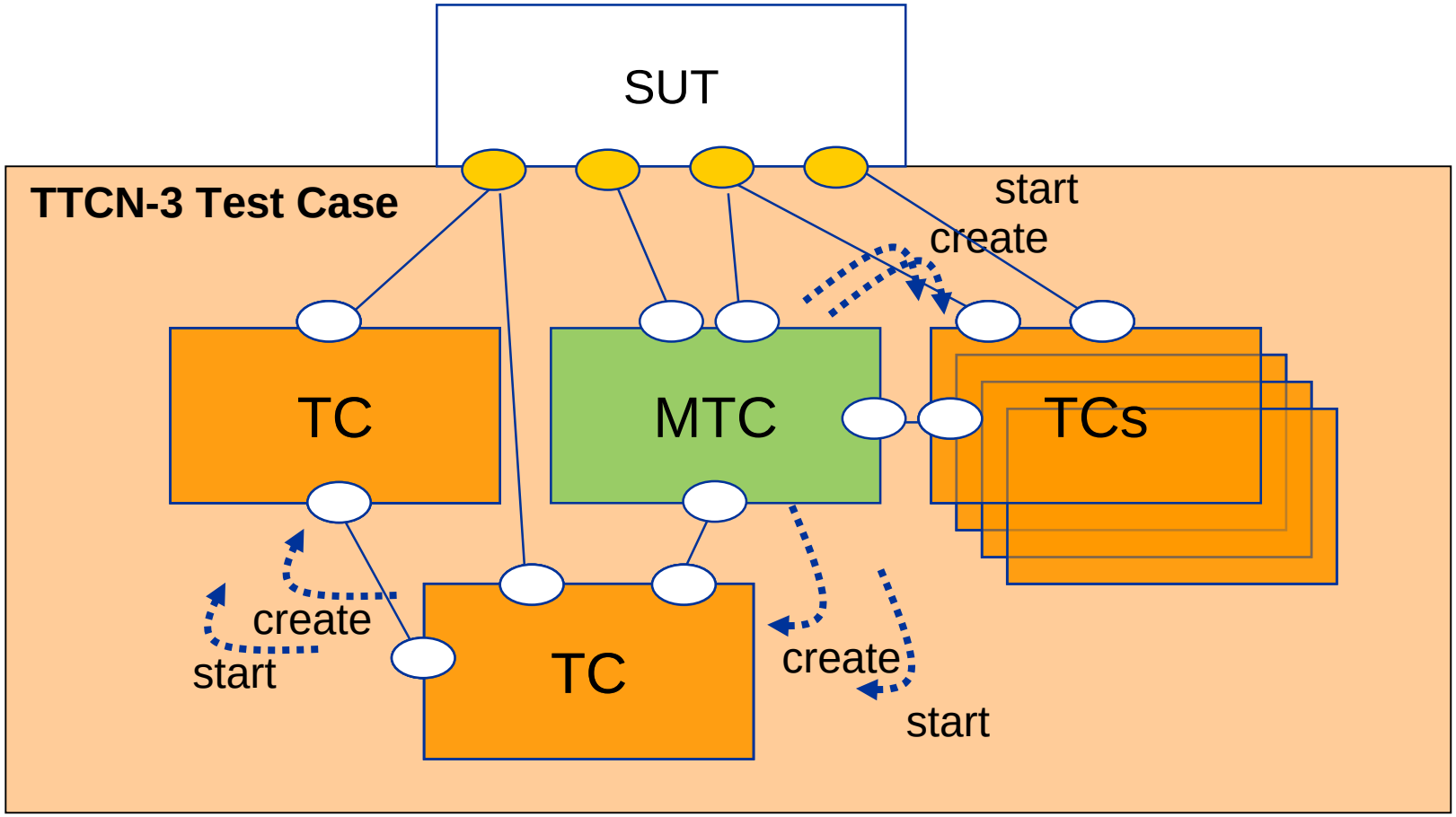
Concepts – Test components

There are three kinds of test components

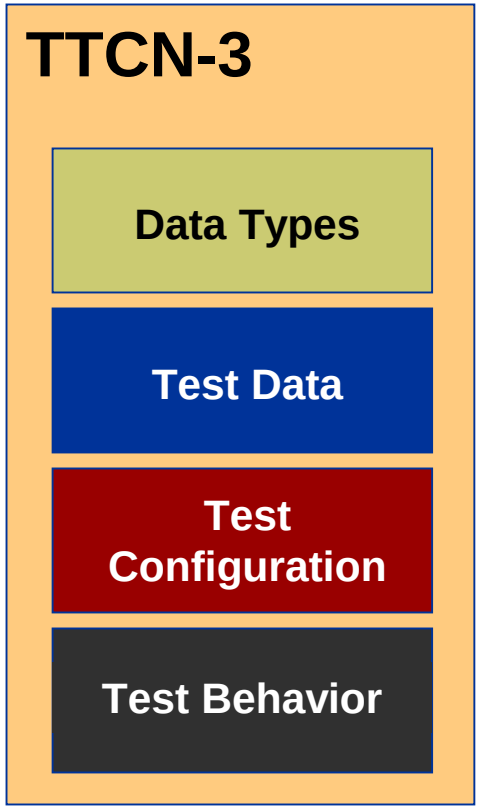
- Abstract Test System Interface defined as a component
- **MTC** (Main Test Component)
- **PTC** (Parallel Test Component)



Concepts – Dynamic test configurations

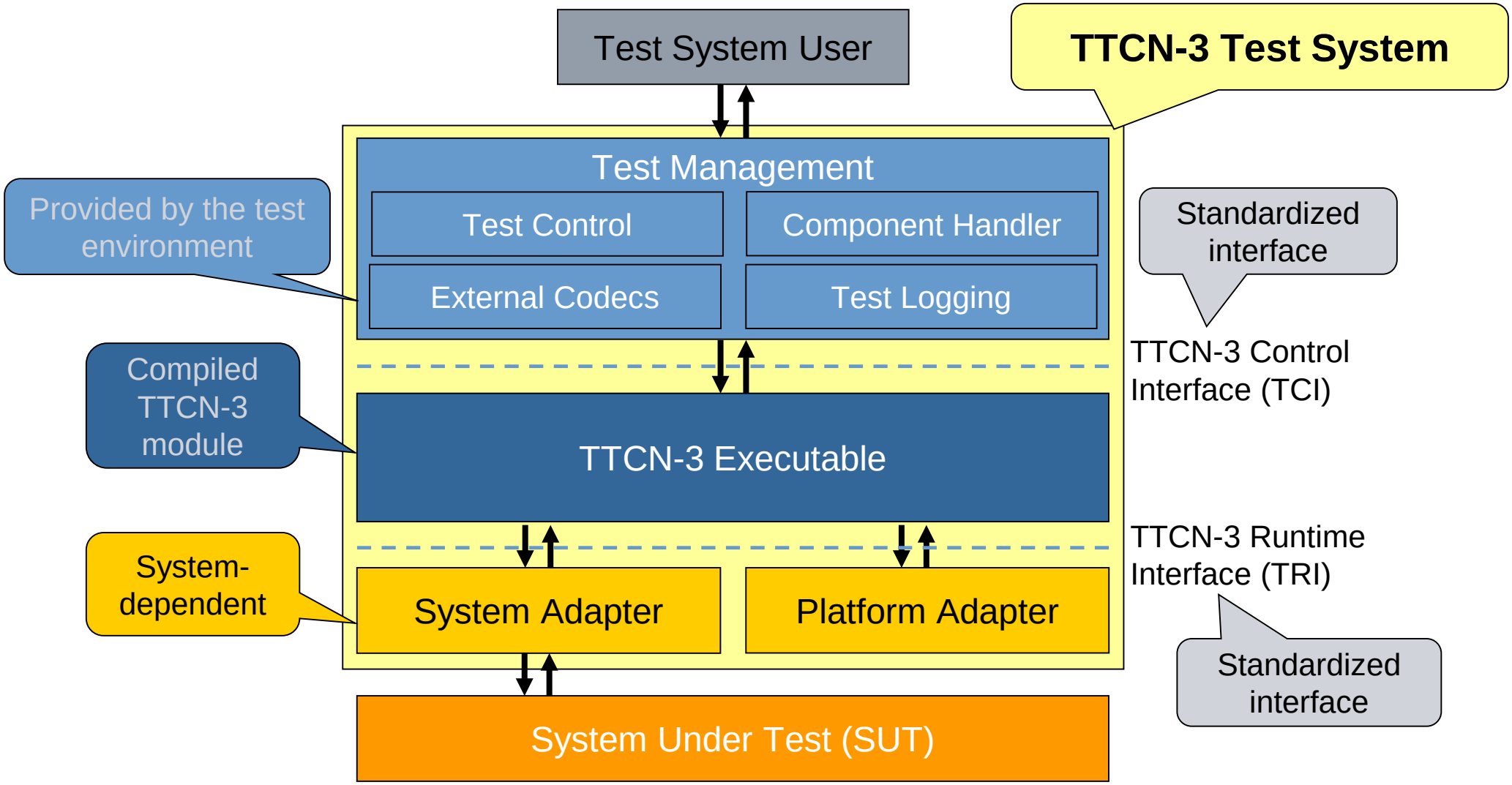


Concepts – Main elements of TTCN-3

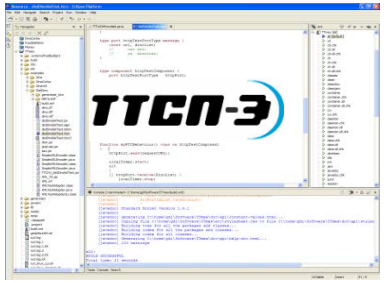


- **Built-in and user-defined generic data types, for example to define messages**
- **Actual test data transmitted/received during testing**
- **Definition of the components and communication ports that are used to build various testing configurations**
- **Specification of the dynamic test system behavior**

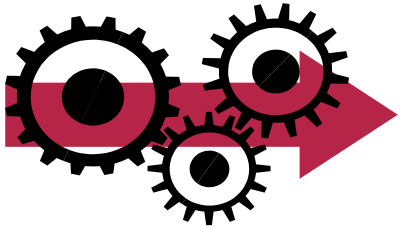
TTCN-3 test architecture



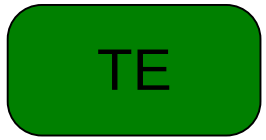
Setting up a TTCN-3 Test System



Abstract Test Suite

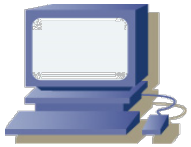


compile

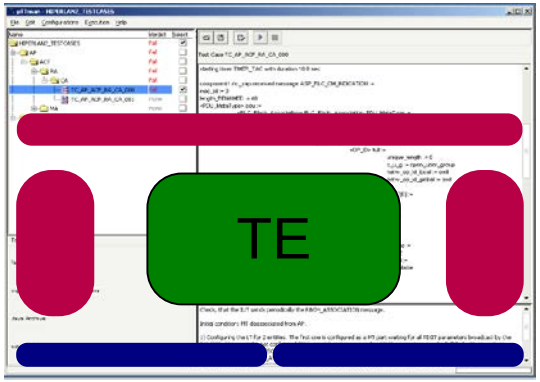


TTCN-3
Executable

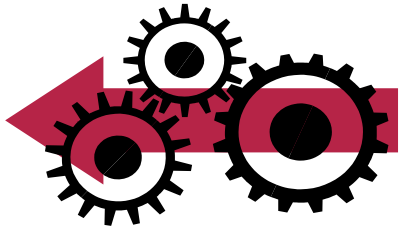
+



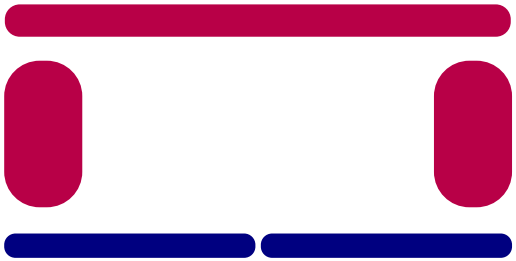
SUT



Executable Test Suite



build



TTCN-3 Runtime System

Relevance of TTCN-3

TTCN-3 is standardized

- Test notation is independent from tool providers
- Test notation keeps pace with arising new technology trends
- Growing TTCN-3 user community
- Test suites for standard protocols become available from ETSI, e.g. for SIP, IPv6, WiMAX

Standardized TTCN-3 interfaces

- Provides quick adaptation to a large variety of systems with predictable costs
- Facilitates reuse of TTCN-3 test suites

TTCN-3 as a universal test notation

- Carries high potential for cost reductions
  - Test suite design
  - Automation of test execution
  - Adaptation to different SUTs
  - Test tool and test suite maintenance
- Supersedes any proprietary ad-hoc solution

Quick Reference Card

http://www.blukaktus.com/

➔ TTCN-3 is a test methodology that is beneficial

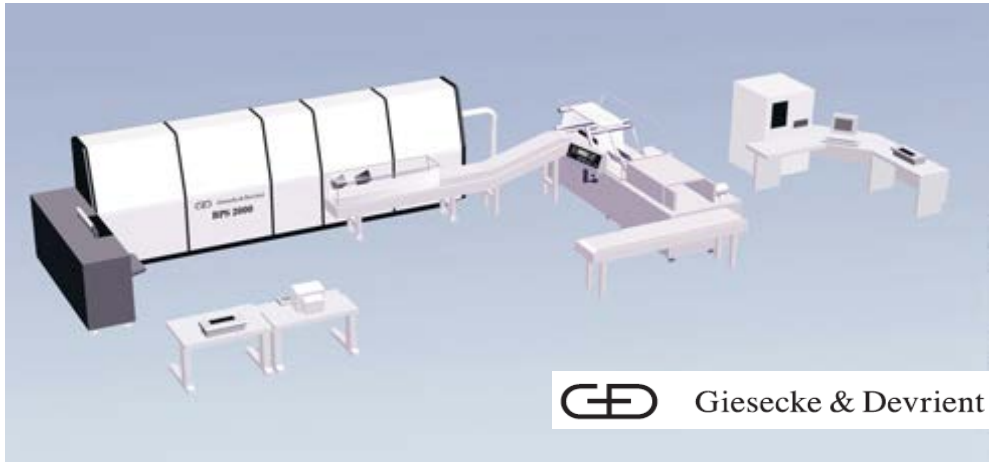
Example – Test automation with TTCN-3 (1)

BPS 3000

These units are deployed at central banks worldwide for

- counting and inspection,
- sorting, packing, and destruction

of banknotes



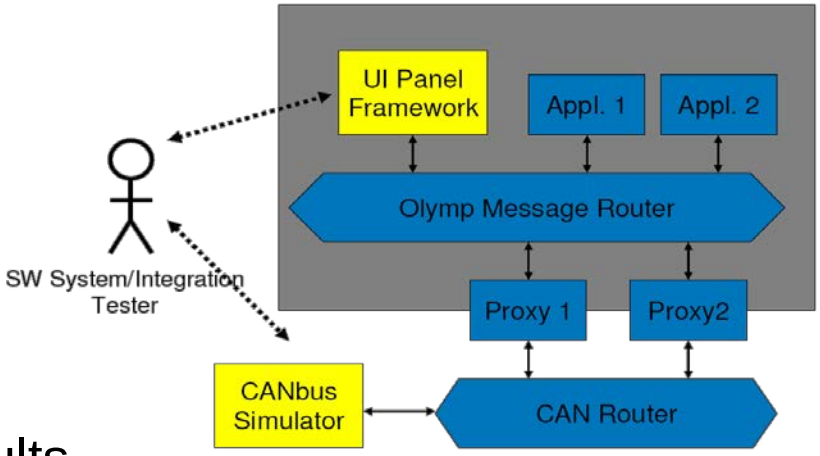
Approach

- Test strategy and test architecture for the system's next generation central control application software
- Integration testing of .NET applications (C#)
- Testing the communication between subsystems
- Usage of TTCN-3 (**T**esting and **T**est **C**ontrol **N**otation Version 3)






Example – Test automation with TTCN-3 (2)



Situation after test automation

- Use of hardware simulator and TTCN-3 test technology enables full test automation
- Test automation integrated into nightly build
- All operation modes tested in software before system integration with machine hardware
- Repeatable test setup and test execution results
- Early defect detection

Benefits

- Faster test execution:
  Test suite runs for **about 4 hours**, manual tests require **about 2 days**
- More frequent test executions:
  Test suite executed **once per day** instead of only **every 2 weeks**

Focus on *risk reduction* and *speed of development*,
(test-) cost reduction typically only medium-term

Test Automation

Agenda

Objectives and Value

Strategies and Approaches

Frameworks and Technologies

Summary

SIEMENS
Ingenuity for life

What we have learned

An efficient quality check of software is not possible without the *right* test automation.

Automation should provide *valuable* support to testing; focus on risk reduction, coverage and speed instead of (test) cost reduction.

Testability and a good test architecture is the key to cost-effective test automation.

Test automation patterns are available for many test automation issues.

You know examples of frameworks as well as new directions: virtualization, cloud testing, crowd testing.

Further readings

Use the SSA Wiki :
https://wiki.ct.siemens.de/x/fReTBQ

and check the "Reading recommendations":
https://wiki.ct.siemens.de/x/-pRgBg

- Architect's Resources:
  - Competence related content
  - Technology related content
  - Design Essays
  - Collection of How-To articles
  - Tools and Templates
  - Reading recommendations
  - Job Profiles for architects
  - External Trainings
  - ... more resources