EVIDEN

Test Automation

From Theory to Practice

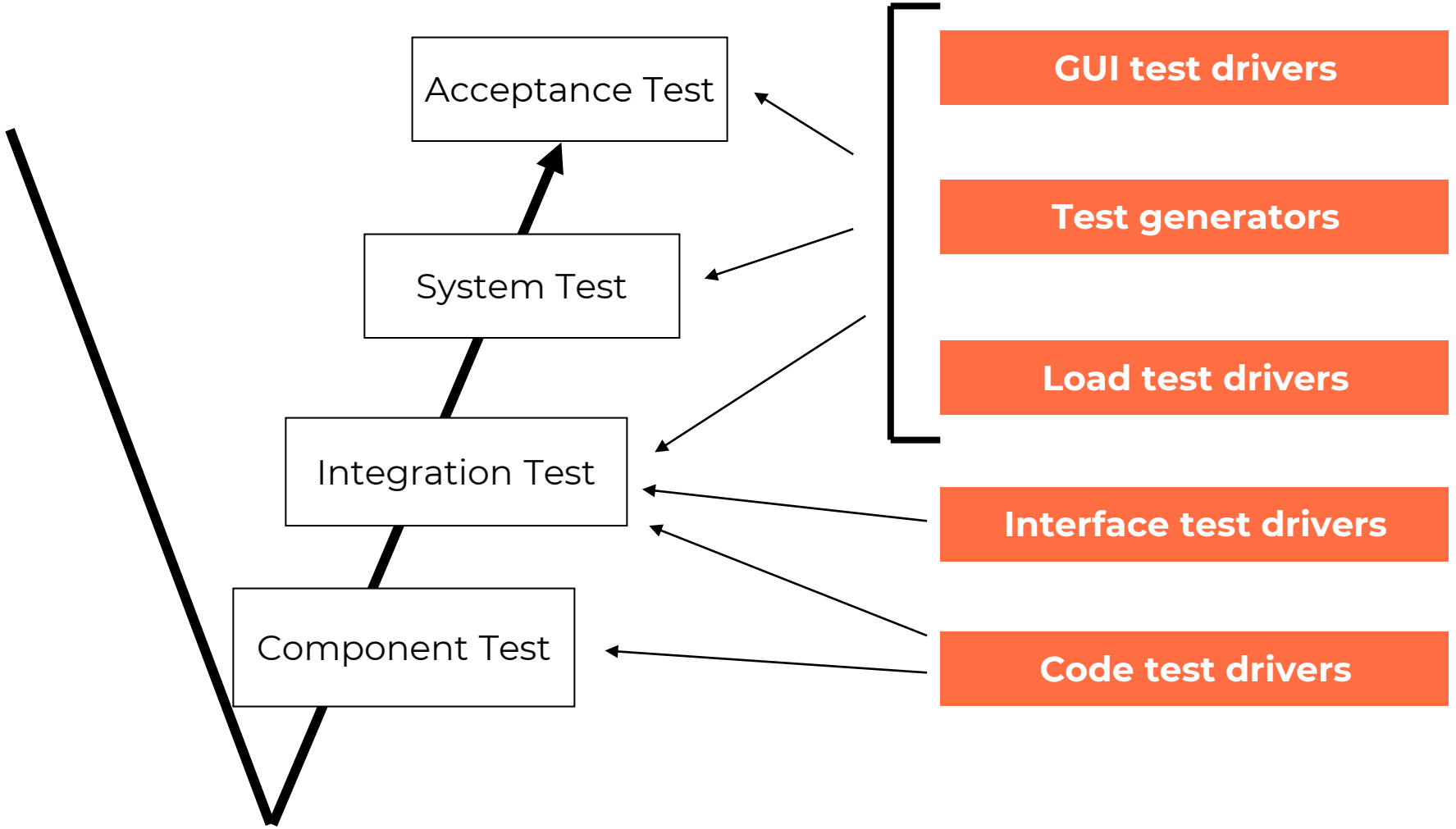
Stefan Mohacsi
Eviden Austria GmbH

an atos business

Categories of Test Automation Tools

- **Code test drivers**
  depend on the programming language
  e.g., Cantata++, JUnit
- **Interface test drivers**
  depend on the interface protocol
  e.g., SoapUI
- **GUI test drivers**
  capture and replay user actions
  e.g., MicroFocus Unified Functional Testing, Ranorex, Selenium, TOSCA
- **Test case / test data generators**
  create test cases or data based on a formal specification
  e.g., Eviden TEMPPO Designer
- **Load test drivers**
  simulate a large number of concurrent users
  e.g., MicroFocus Performance Center, JMeter

Categories of Test Automation Tools



Principles of GUI Test Automation

Functionality of Capture/Replay Tools:

- Recording of test scripts
- Automated execution of test scripts
- Deviations from the expected behavior are detected automatically and are written to a test report

Advantages:

- The effort for test execution is reduced significantly
- Test scripts can run overnight
- Human errors during test execution are avoided

Disadvantage:

- The effort for maintaining the test scripts can be very high

Testability Requirements for GUI Test Automation

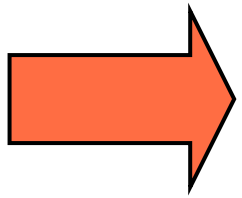
Each tested GUI object must allow the test execution tool to do the following:

1. **Detect** the existence of the object
2. Uniquely **distinguish** it from all similar objects
3. **Simulate** all possible user actions
4. **Verify** the object's properties

1. Recognizability

Firstly, the test tool has to be able to **detect the existence** of the object.

- No problem with GUI objects that are managed directly by the operating system (e.g., Visual C++, Visual Basic)
- A special plug-in is required for GUI environments with object management of their own (e.g., Java, Browsers)

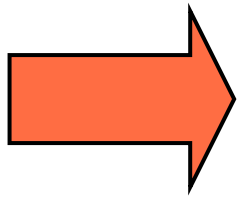


For exotic GUIs no plug-in may be available
=> no automated test possible!

2. Uniqueness

The test tool has to be able to **uniquely distinguish** the GUI object from all similar objects in the same window

- Objects are identified by their properties (label, class name, ID etc.)
- If two or more objects have the same properties, they are identified by their positions

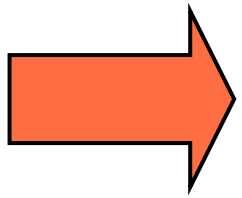


Problems arise if the positions of GUI objects can change dynamically

3. Simulation of User Actions

Basic user actions like simple mouse clicks and keyboard entries can be performed on any object

Object-specific actions (like selecting a particular item in a tree) require that the test tool is familiar with the associated methods. A plug-in may be necessary

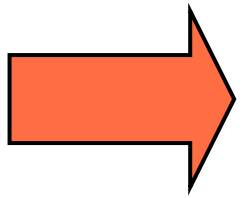


Self-made GUI classes that require complex user interaction are usually NOT supported by test tools!

4. Verification of Object Properties

After each test run the test tool has to be able to **verify** whether the actual results match the expected ones.

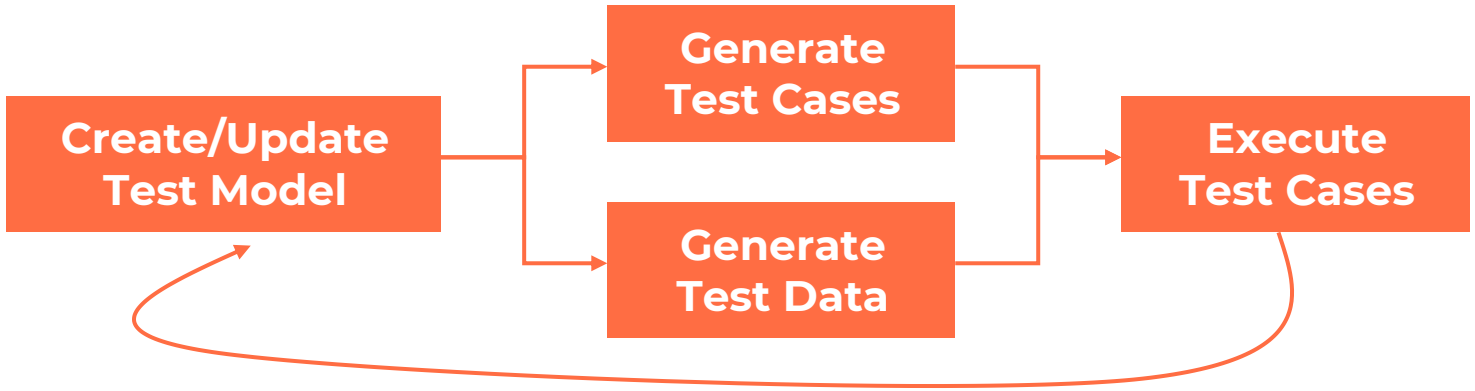
- The object has to provide methods that return the current value of its properties
- The test tool has to be familiar with these methods



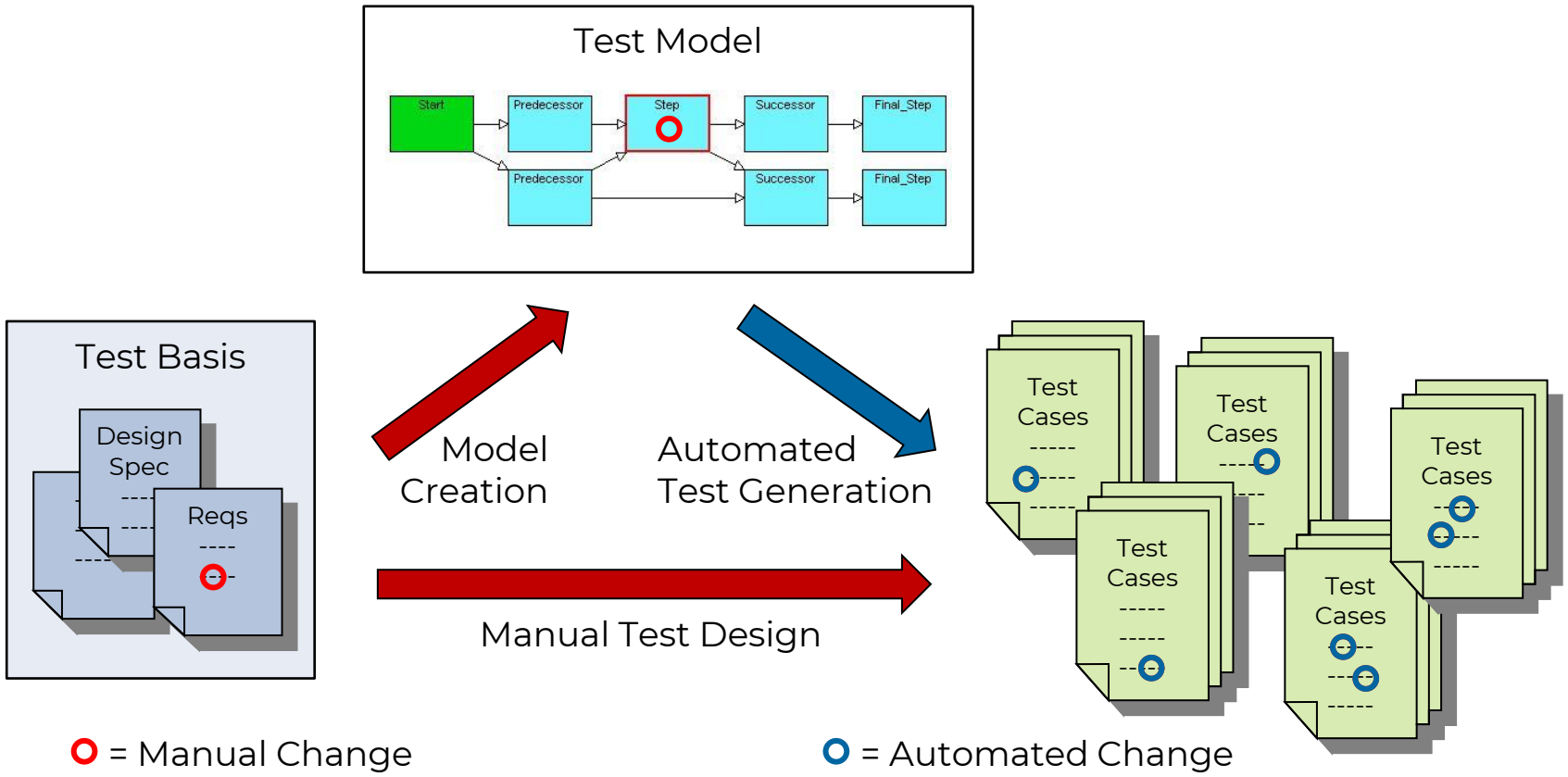
Again, problems with exotic or self-made GUI classes may occur!

What is Model-based Testing (MBT)?

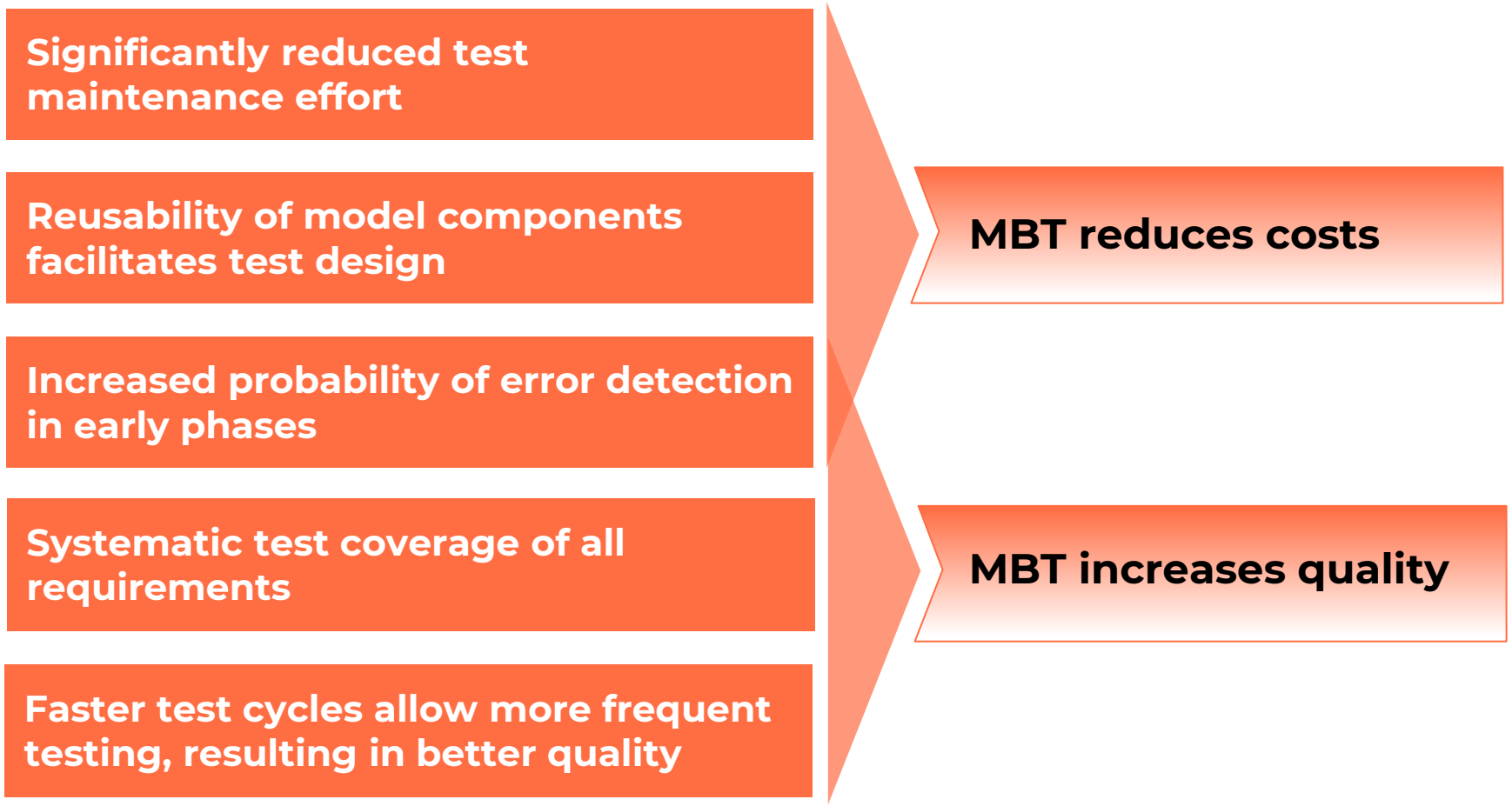
- Test cases and data are **generated automatically** from a model of the system under test
- Tests can be exported as scripts for **automated test** execution tools or as instructions for **manual testing**
- In the next test cycle, **only the model** needs to be **maintained** while the tests are updated automatically



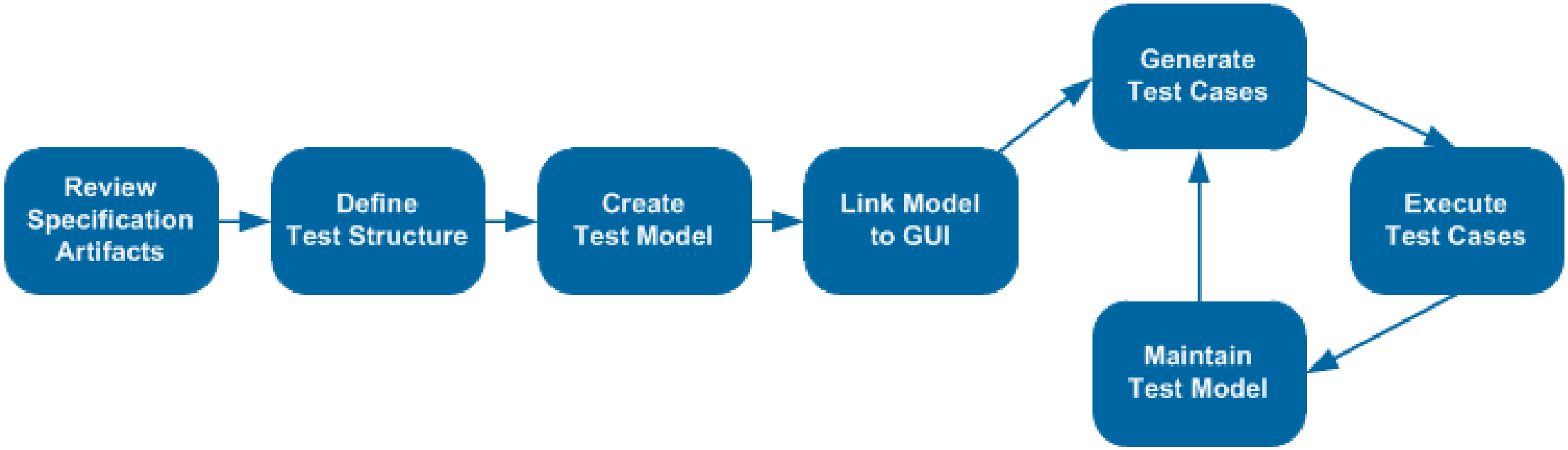
Why is Model-based Testing more Efficient?



Benefits of Model-based Testing



Workflow for Model-based Testing

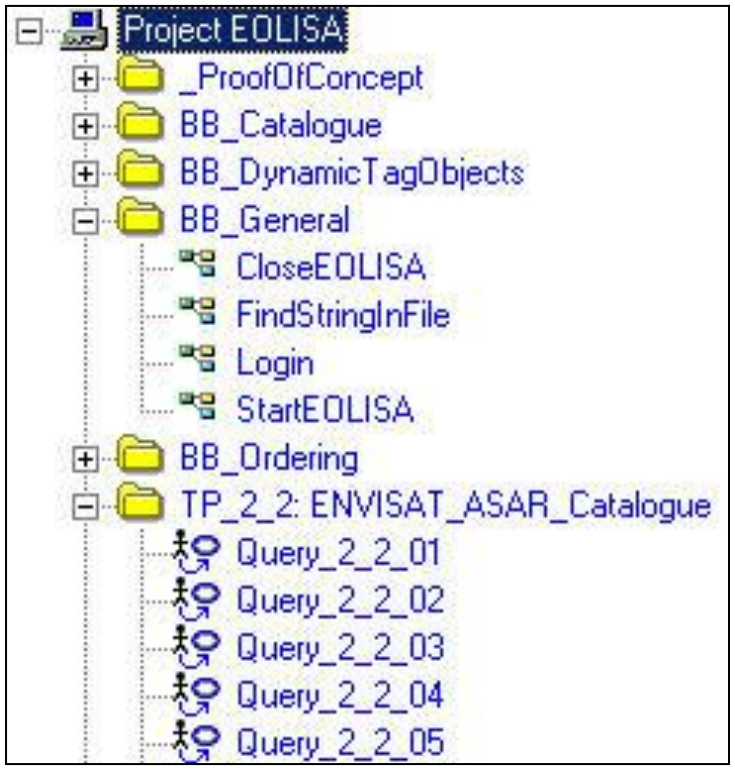


The IDATG Method

IDATG = Integrating Design and Automated Test Case Generation

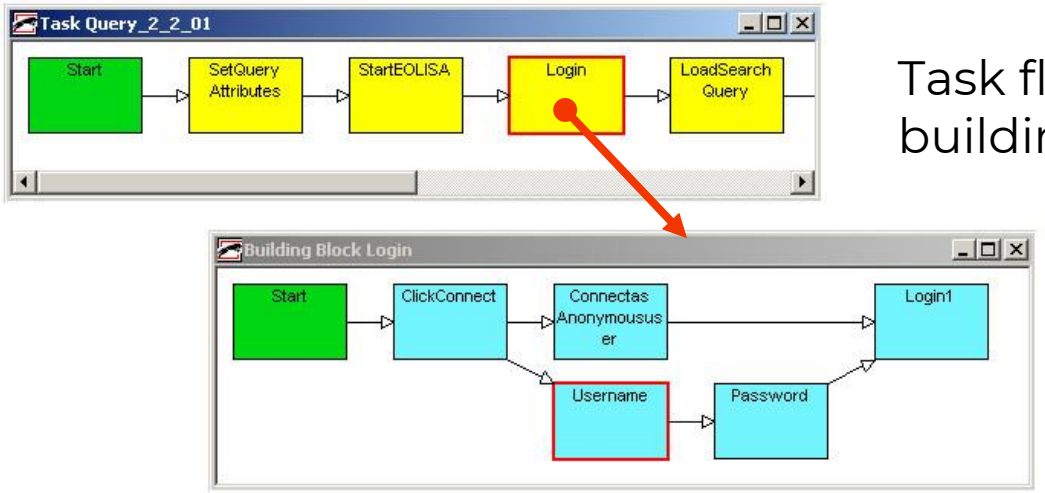
- Model-based test design in a notation optimized for testing using the tool Eviden TEMPPO Designer
- Built-in GUI Spy for mapping steps to GUI elements
- Also applicable for manual tests and testing via non-GUI interfaces
- Generation of test data using systematic methods
- Generation of test scripts in various formats (XML, MicroFocus UFT®, Ranorex®, etc.)

Test Structure Definition



- Start MBT with test scenarios that have the **highest ROI** (many test repetitions, easy to model and automate)
- Definition of re-usable and parametrizable **building blocks** improves clearness and maintainability

Create Test Model



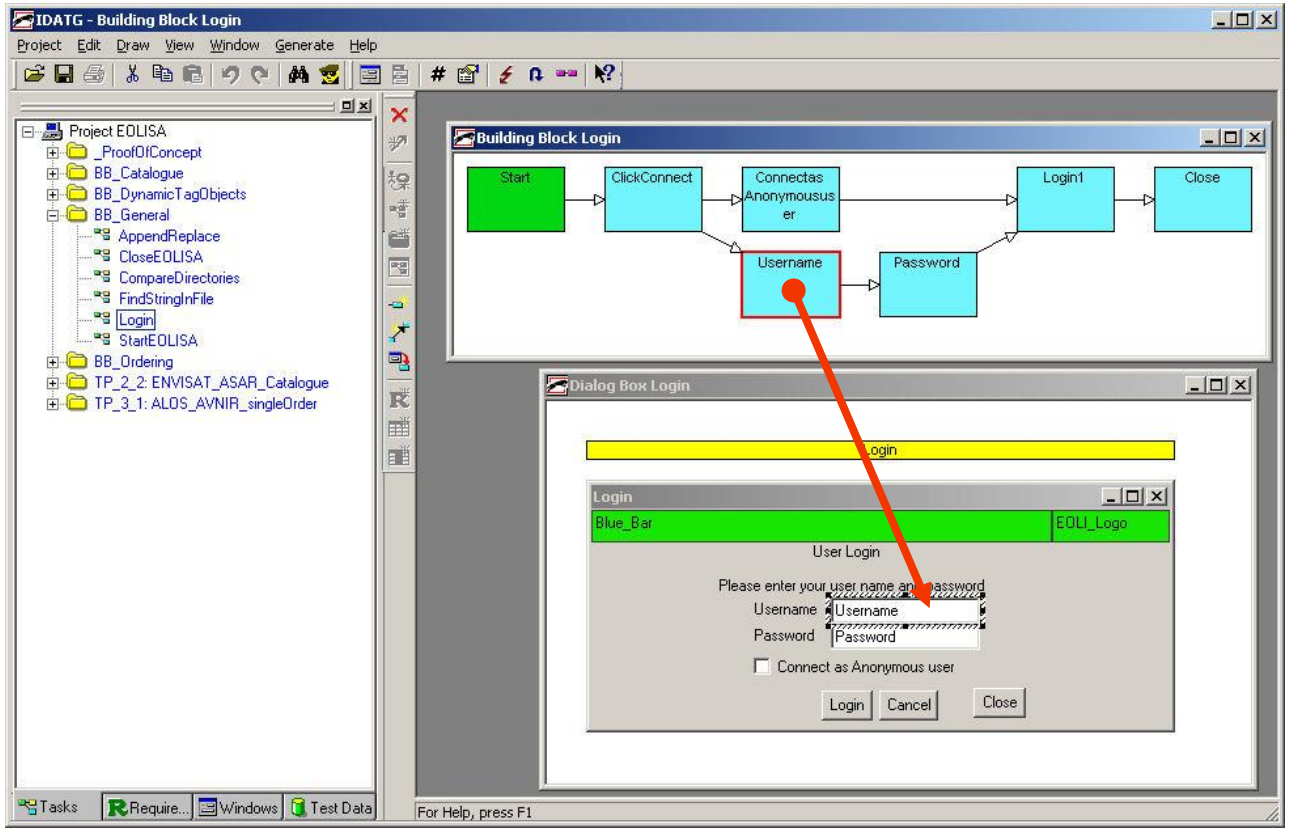
Task flow with building blocks

Reusable building block "Login"

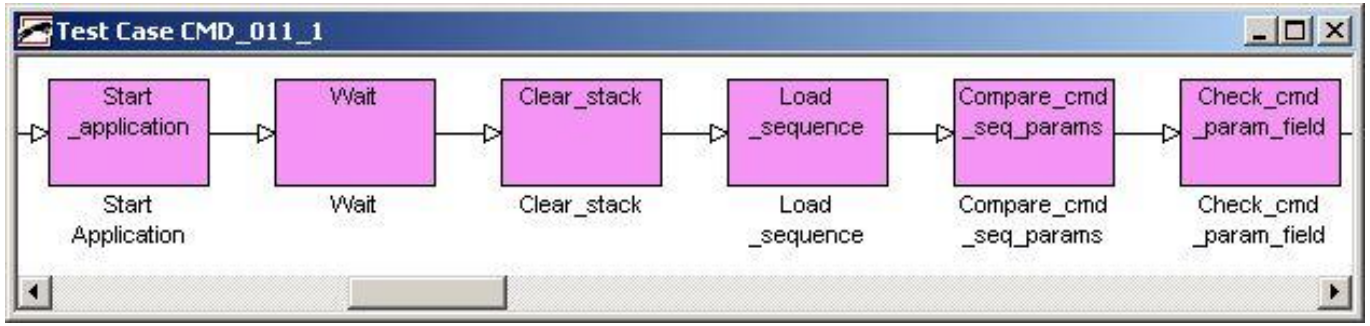
- **Building Block Concept:**
  Each step may either represent an atomic step (blue) or an entire step sequence (yellow)
- Re-use of building blocks **minimizes** the effort for **test maintenance**

Link Model to GUI

- GUI objects can be **recorded or imported**
- Steps can easily be **linked** to GUI objects



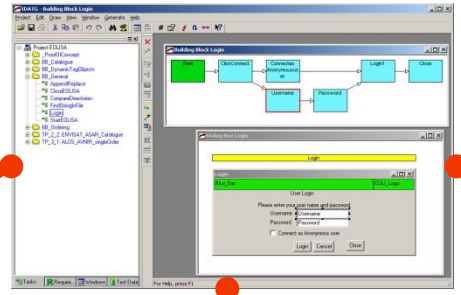
Test Case Generation

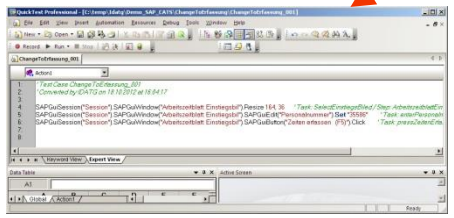


- As soon as part of the application has been specified, it is possible to **generate test cases**
- Generation algorithm can satisfy various **coverage criteria** (step / edge / data coverage / random)
- e.g., for the test package “Commanding” 118 test cases with thousands of steps are generated in less than 3 seconds

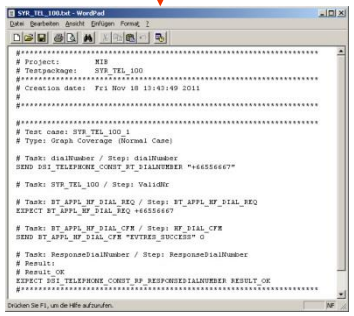
Test Script Export

Eviden TEMPPO Designer

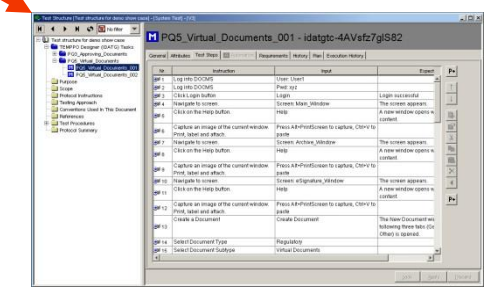




GUI Test Script
(e.g., for MicroFocus UFT®)



Customized Test Script
(e.g., for ESA Test Commander)



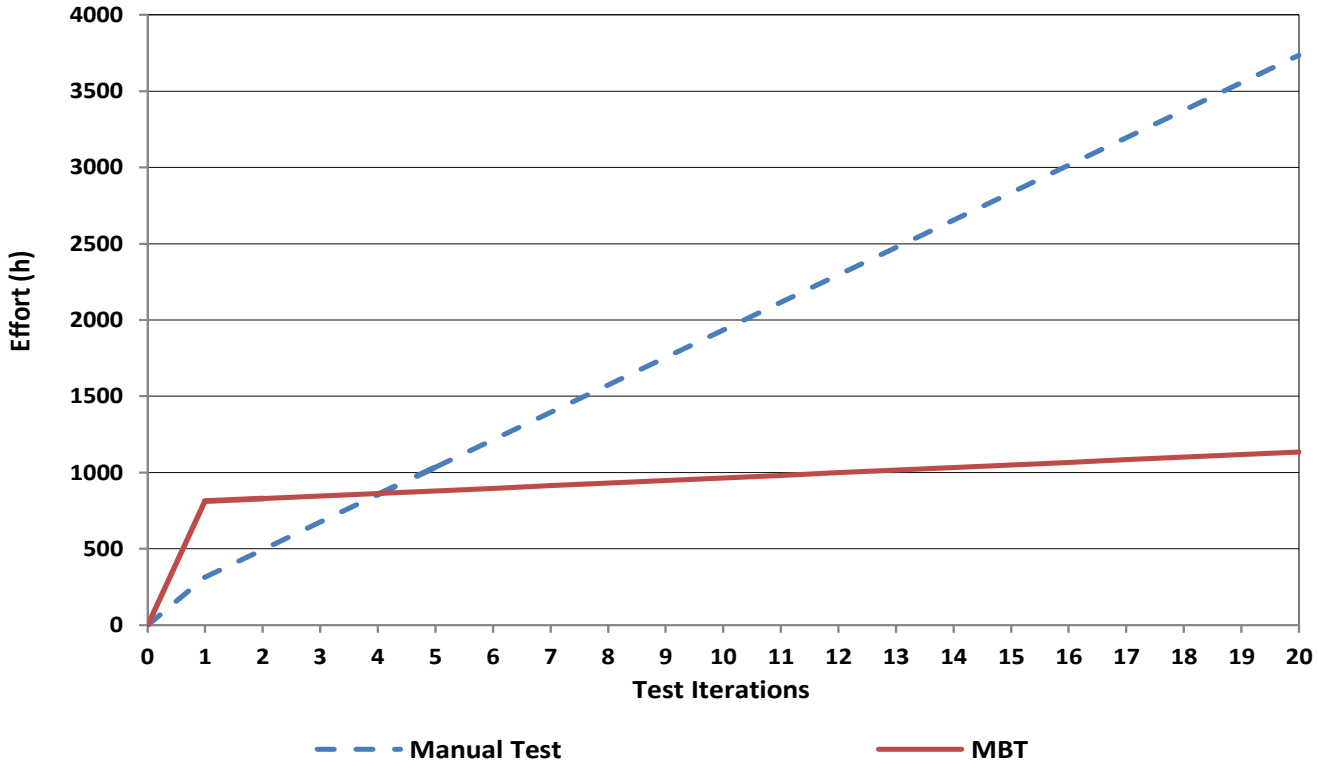
Manual Test Instructions
(e.g., for Test Management Tool)

Test Maintenance

Experiences from ESA MMUS project

- **System changes**, e.g., to the workflow for searching satellite images, often affected hundreds of test cases
- The effort invested into the test model paid off now. Instead of maintaining each single test case like before, **only some parts of the model had to be adapted**. Afterwards, all test cases could be updated automatically by re-generating them.

Total Test Effort

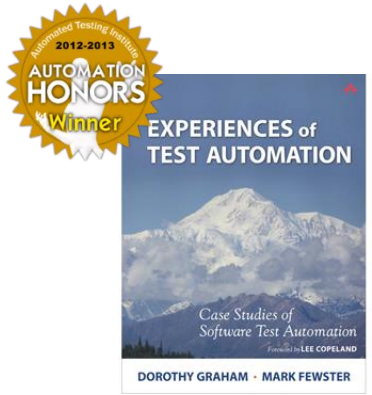


Manual Testing	Model-based Testing
After 1 test cycle: 315h	After 1 test cycle: 812h
2 cycles: 495h	2 cycles: 829h
4 cycles: 855h	4 cycles: 863h
10 cycles: 1935h	10 cycles: 965h
20 cycles: 3735h	20 cycles: 1135h

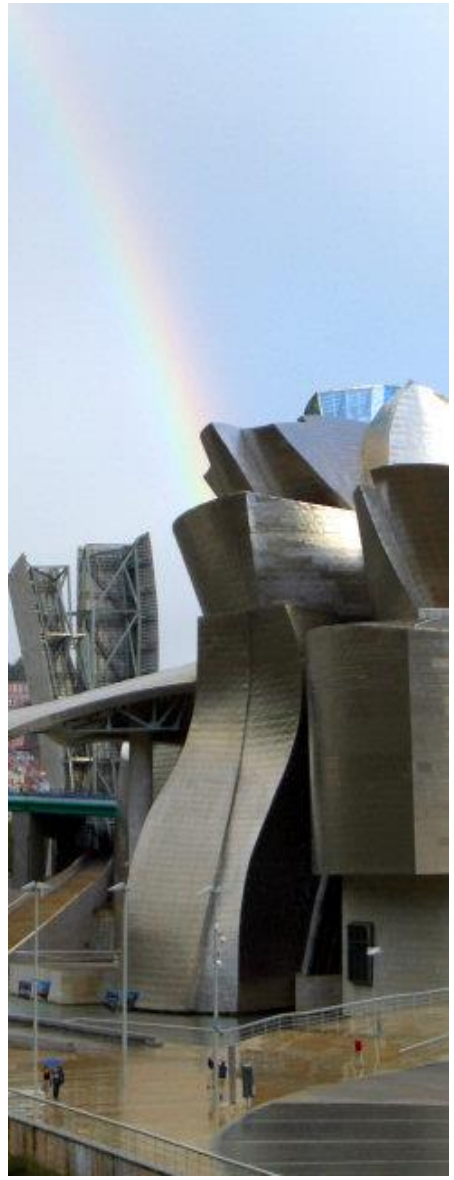
EVIDEN

Summary

- **GUI test automation** is a must for large SW projects
- The GUI's **testability** is a prerequisite for successful test automation
- If applied correctly, MBT can **significantly reduce test maintenance costs**
- Key considerations when creating a test model should be **reusability and maintainability**



For more details, see our chapter in:
Experiences of Test Automation
by Dorothy Graham and Mark Fewster
(Addison-Wesley 2012)



EVIDEN

Questions?

For more information, please contact:
stefan.mohacsi@atos.net