Eiropas Sociālā fonda projekts
“Datorzinātnes pielietojumi un tās saiknes ar kvantu fiziku”
Nr.2009/0216/1DP/1.1.1.2.0/09/APIA/VIAA/044

ESF EIROPAS SOCIĀLAIS FONDS
EIROPAS SAVIENĪBA
IEGULDĪJUMS TAVĀ NĀKOTNĒ

An Implementation of Self-Testing

Edgars Diebelis
Prof. Dr. Janis Bicevskis

07.07.2010

Self-testing

- Ability to execute stored test examples to ensure that software.
- Test approach:
  - Manual testing.
  - Automated testing with testing tools.
  - Self-testing.

Content

- Smart Technologies.
- Method of Self-Testing.
- Phases of system testing.
- Modes of system actions.
- Test Point.
- Implementation of Self-Testing.
- Self-testing example.
- Conclusions.

Smart technology

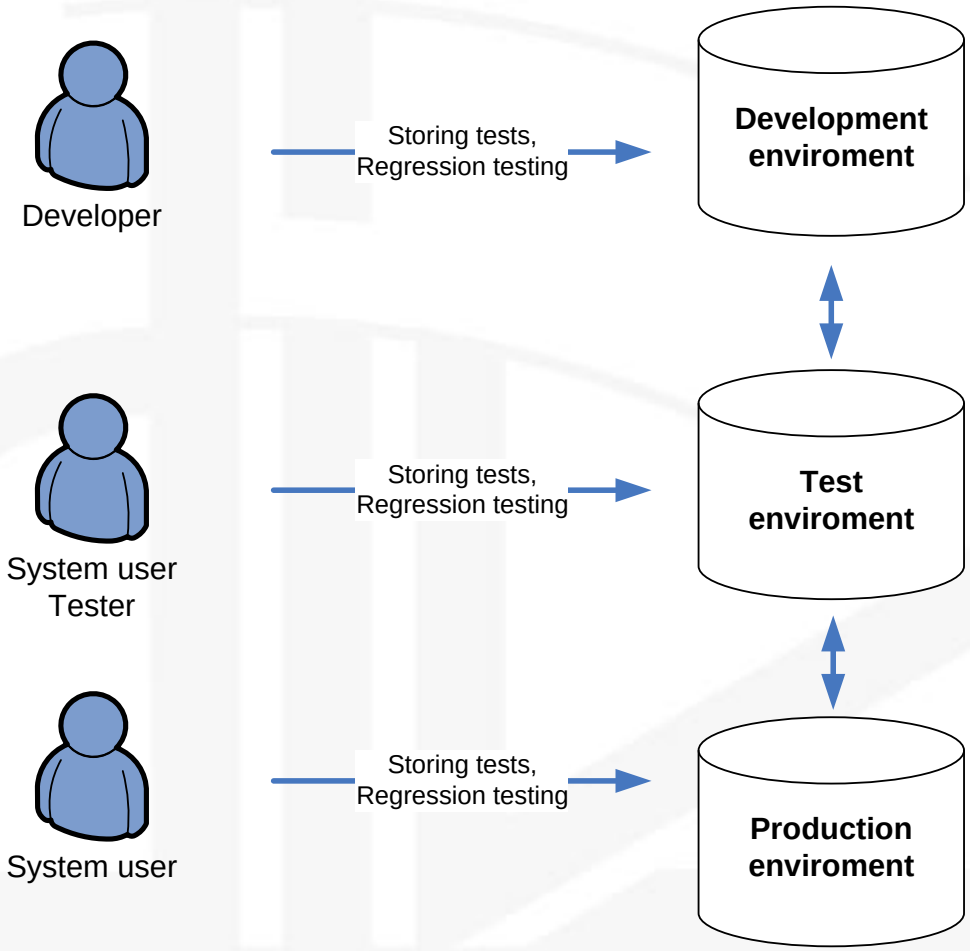
The idea behind this software is that it „manages itself”, i.e., controls external and internal factors and reacts on them adequately

- The software complying with smart technology principles has the following features:
  - Ability to check the target environment (testing of the environment).
  - Ability to ensure sound functioning of software (self-tests of basic functionality).
  - Ability to automatically retrieve the newest version of software from server and convert the accumulated data in compliance with requirements of the newest version (versioning).
  - Ability to monitor the security and data quality of software (monitoring).

Method of Self-Testing

- Self-testing contains two components:
  - Test cases of system's critical functionality to check functions, which are substantial for system using.
  - Built-in mechanism (software component) for automated software testing (regression testing) that provides automated running of test cases and comparing test results with standard values.
- Characteristics of self-testing:
  - Software is delivered together with the test cases used in automated self-testing of the system.
  - Testing can be repeated in production, without impact on production database.

Phases of system testing



Self-testing in Development environment

- Preparing test cases to cover the critical functionality of system.
- Import/export of test cases from/into test or production environments.
- The independent testing might detect misinterpretation of specification or nonfunctional requirements of system.
- Self-testing don’t absolve developers of system from system testing.

Self-testing in Test environment

- Test environment should differ from production only by delivered system changes.
- Test environment is meant for accept-testing.
- Accepts or declines tests from development environment.
- Add new test examples.
- Import/export of test cases from/into development or production environments.
- Execute self-testing of delivered system to ensure
  - that previous software functionality will remain unchanged.
  - that all changes are delivered and installed in the test environment.

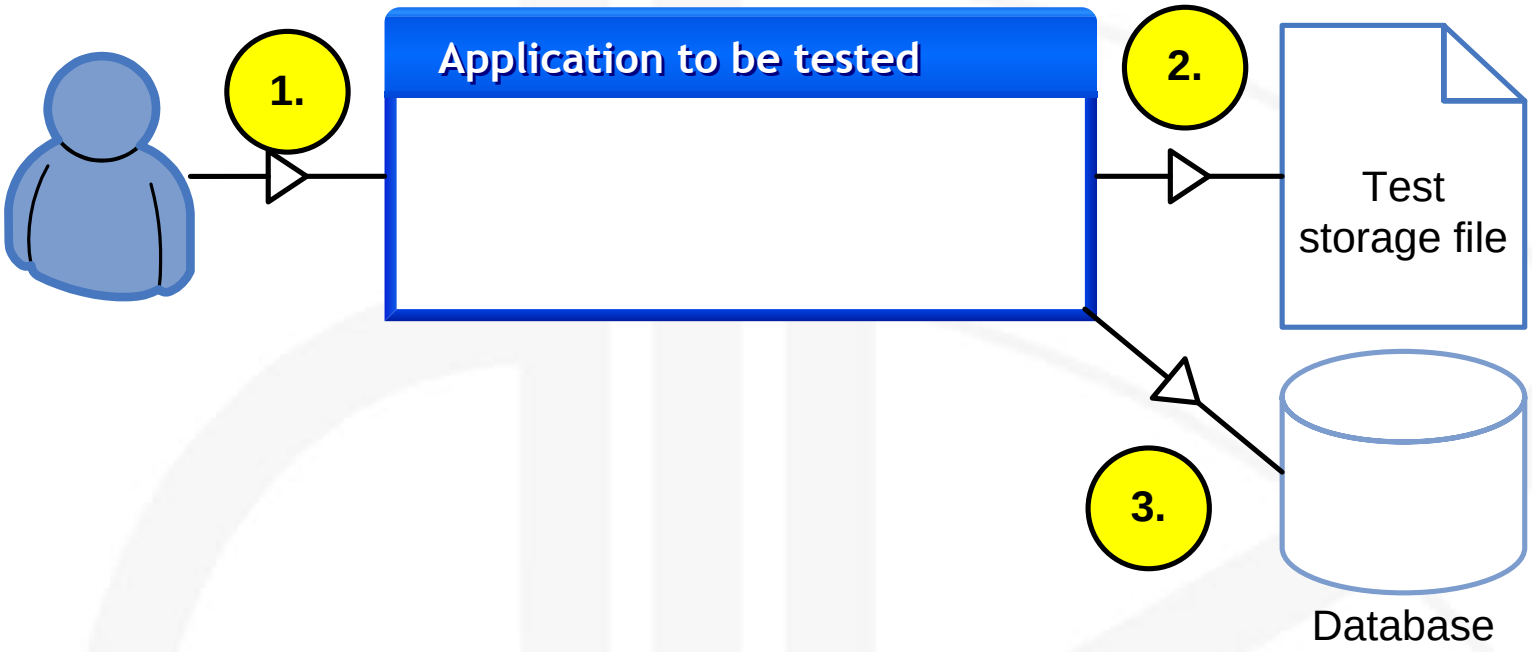
Self-testing in Production

- During production mode there is access to real data.
- Import/export of test cases from/into development or test environments.
- Add new test examples.
- Execute self-testing of delivered system to insure
  - that previous software functionality will remain unchanged.
  - that all changes are delivered and installed in the test environment.
- During production self-testing mode data are used in read-only mode.
- Saving of data is realized in test execution and test storing databases.

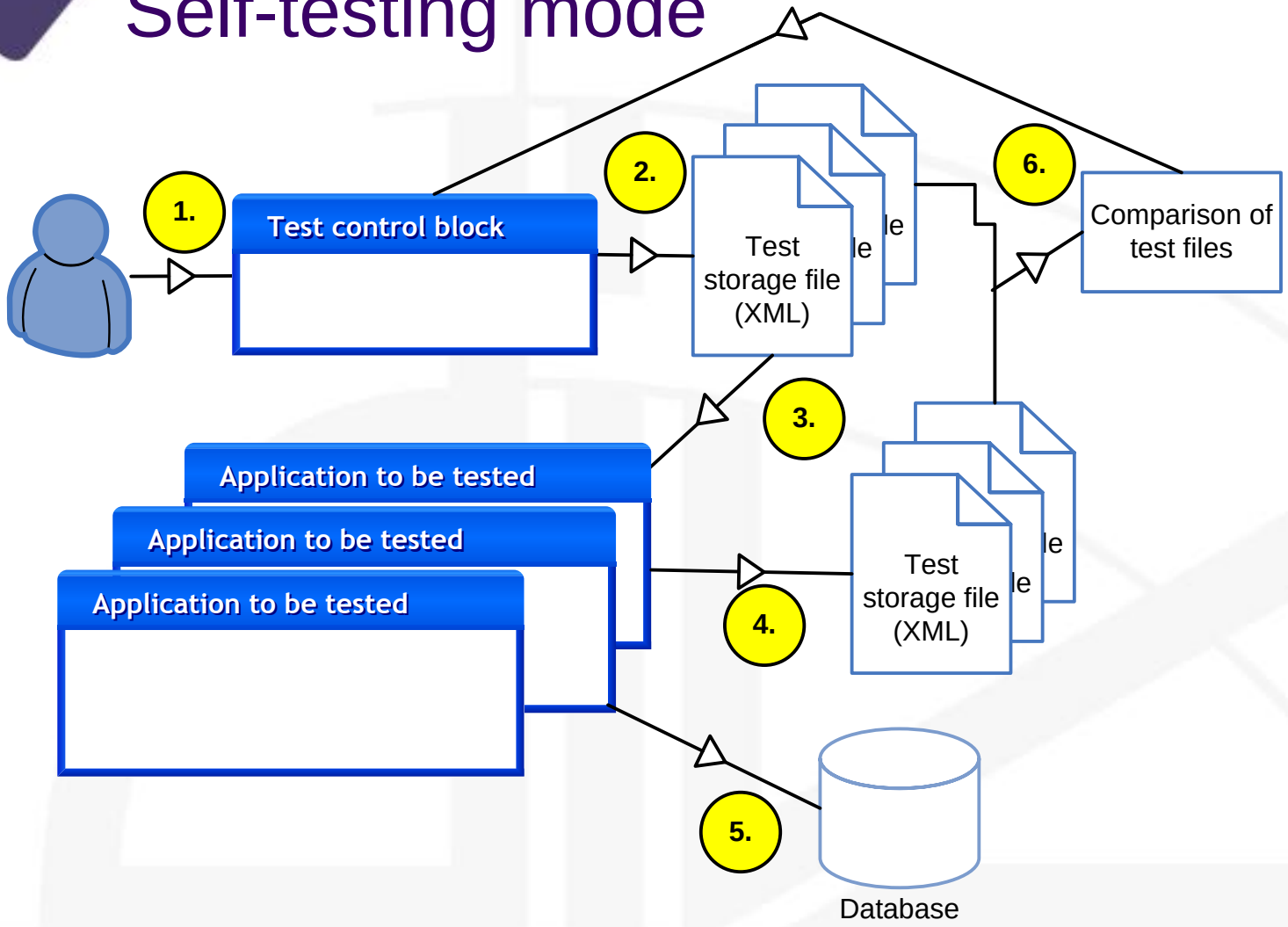
Modes of system actions

- Test storage mode.
- Self-testing mode.
- Use mode.
- Demonstration mode.

Test storage mode



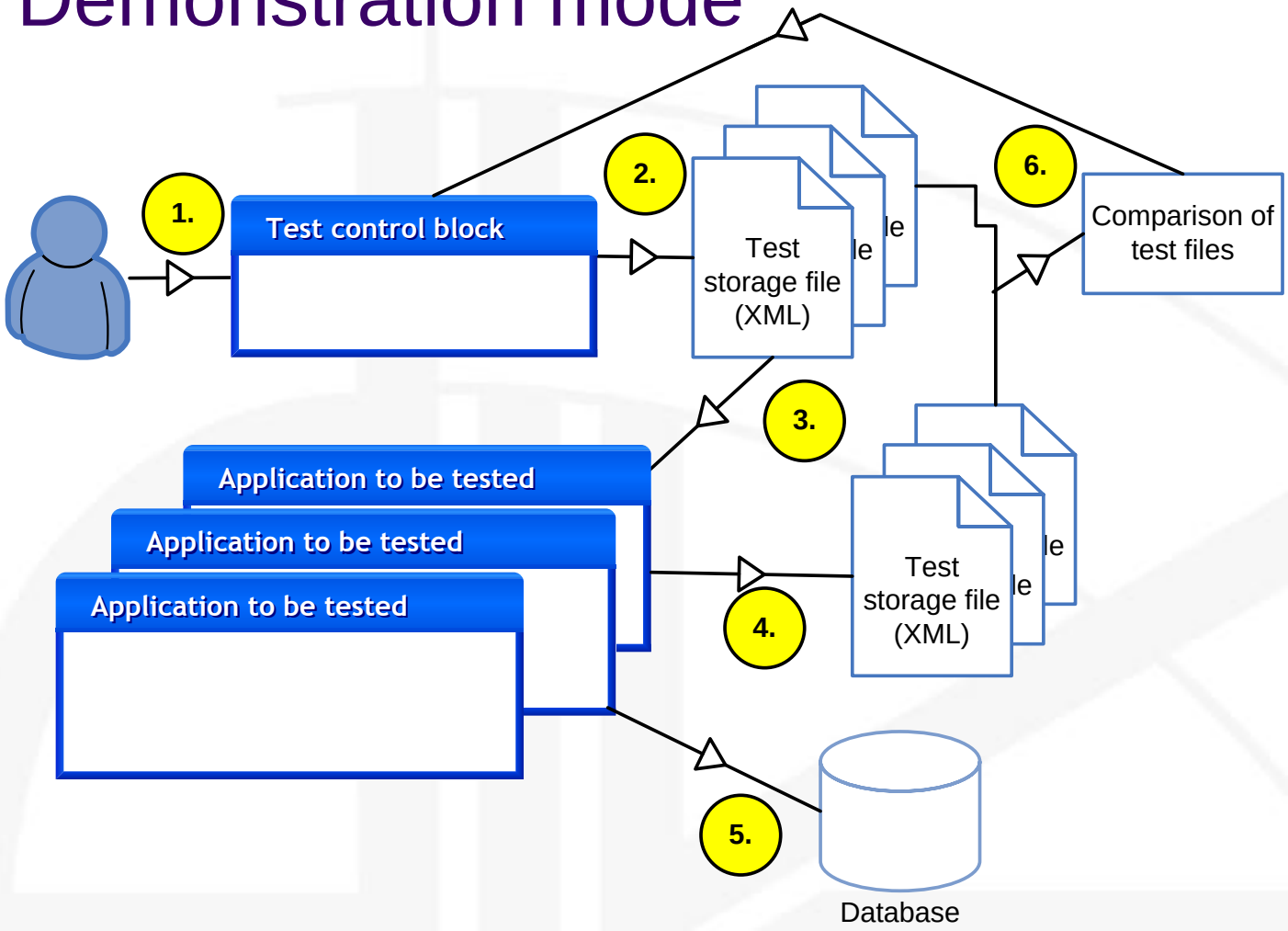
Self-testing mode



Use mode

- During usage mode there are no testing activities. Users use this mode for main functionality of system.

Demonstration mode



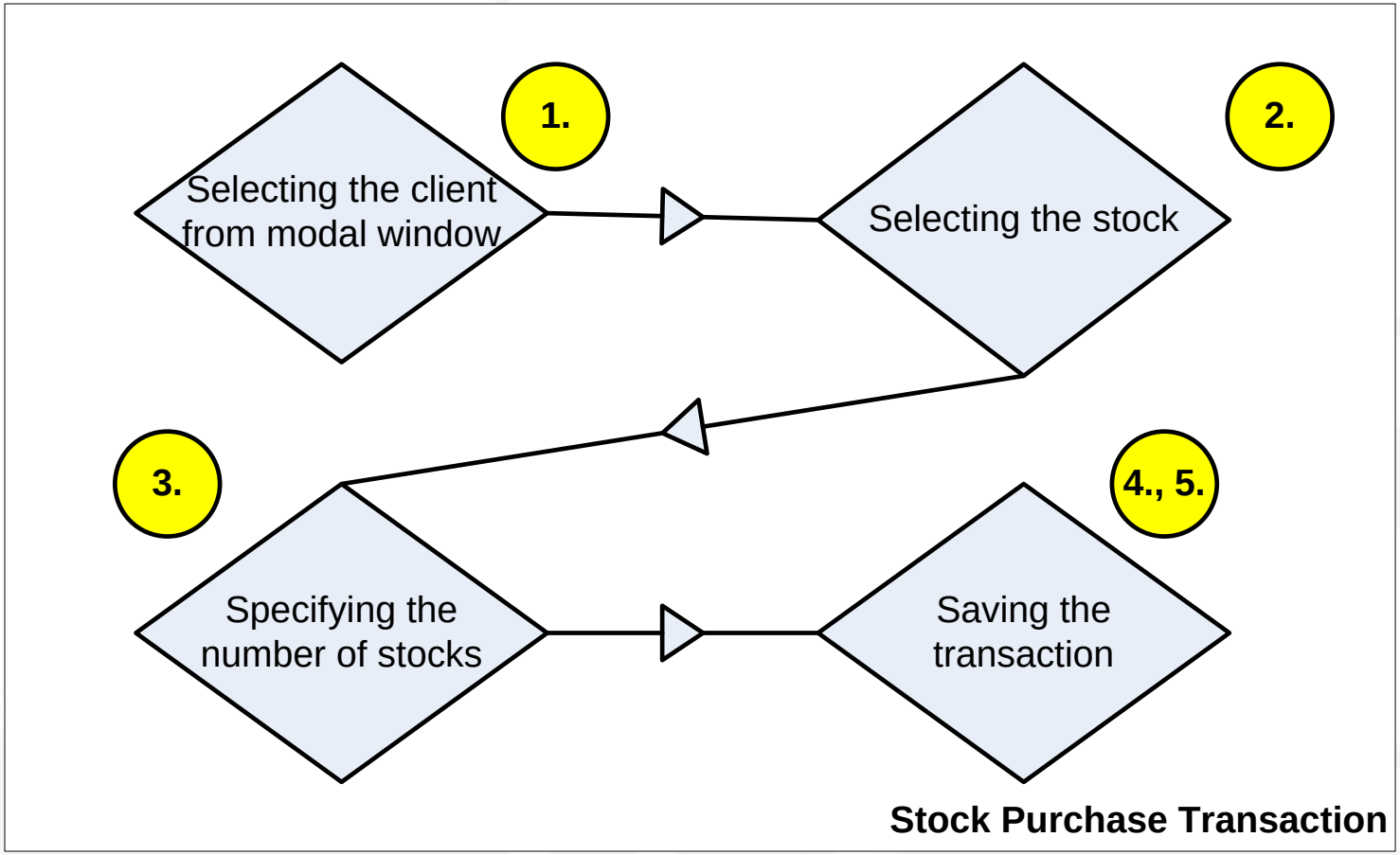
Test point

- Test point is a command upon which system testing actions are executed.
- Test point is a programming language command in the software text.
- Test point ensures that:
  - particular actions and field values are saved when storing tests.
  - the software execution outcome is registered when tests are executed repeatedly.
- By using test points, it is possible to repeat the execution of system events.

Types of Test Point

- Field with value.
- Comparable value.
- MessageBox.
- Modal window.
- SQL query result.
- Application event.
- Test execution criterion.

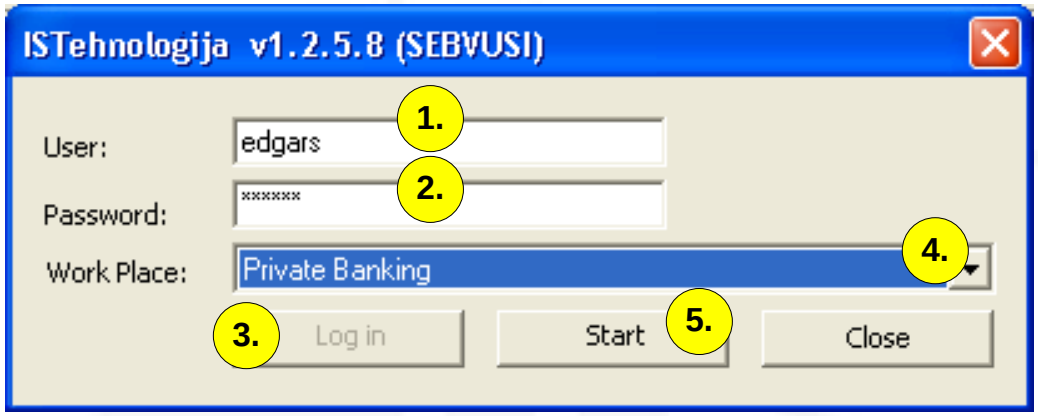
Example


1.
Selecting the client from modal window
2.
Selecting the stock
3.
Specifying the number of stocks
4., 5.
Saving the transaction
Stock Purchase Transaction

Implementation of Self-Testing

- Key components of self-testing software are:
  - Test control block.
  - Library of test actions.
  - Test file (XML file).

Self-Testing example




```
- <Actions>
  - <Control Id="1" Name="Username" Event="Username_TextChanged" ControlType="System.Windows.Forms.TextBox">
      <Value xsi:type="xsd:string">edgars</Value>
    </Control>
  - <Control Id="2" Name="Passwd" Event="Passwd_TextChanged" ControlType="System.Windows.Forms.TextBox">
      <Value xsi:type="xsd:string">123456</Value>
    </Control>
    <Control Id="3" Name="btnAuth" Event="btnAuth_Click" ControlType="System.Windows.Forms.Button" />
  - <Control Id="4" Name="Workplace" Event="Workplace_SelectedIndexChanged" ControlType="System.Windows.Forms.ComboBox">
      <Value xsi:type="xsd:int">20</Value>
    </Control>
    <Control Id="5" Name="StartWork" Event="StartWork_Click" ControlType="System.Windows.Forms.Button" />
  </Actions>
```

Conclusions

- Self-testing does not replace traditional testing of software; it modifies testing process by increasing significantly the role of developer in software testing.
- Self-testing requires additional efforts to integrate functionality of self-testing into software, to develop critical functionality tests and testing procedures.
- Self-testing significantly saves time required for repeated testing (regression) of the existing functionality. This is critical for large systems, where minor modifications can cause fatal errors and impact system's usability.
- Self-testing ensures that functionality of unmodified software will remain unchanged, while new software modules will perform according to the new tests confirmed by system user. As a result, system user can be sure that modifications in software will not cause incidents, while developer can be sure about quality of delivered software.

Conclusions (continued)

- Using the self-testing features the complete functionality of system is tested, thus leaving no room for errors during independent testing. The independent testing can indicate only differences in interpretation of specification, but not in technical implementation.
- Introduction of self-testing functionality is more useful in incremental development model, especially gradually developed systems and systems with long-term maintenance; and less useful in linear development model.
- Test points make test recording and automatic execution much easier. Test points ensure that tests can be recorded in a convenient and easy-to-read manner.
- Test execution criteria test point determines the possibility to execute the test using the available data set
- If test execution criteria test points are used, it is not necessary to maintain the data set which was used to register the test.
- Test execution criteria test point provides a possibility to execute tests in random order.

Thank you!