PHILIPS

sense and simplicity

Challenges and solutions in test automation of medical visualization applications

Kiran Kumar N, Narayanan AK, Pattabhirama Pandit, Rajesh Arasu, Sanjay Gupta
Philips Electronics India Ltd, Philips Healthcare, BCoC

Oct 01, 2009

PHILIPS

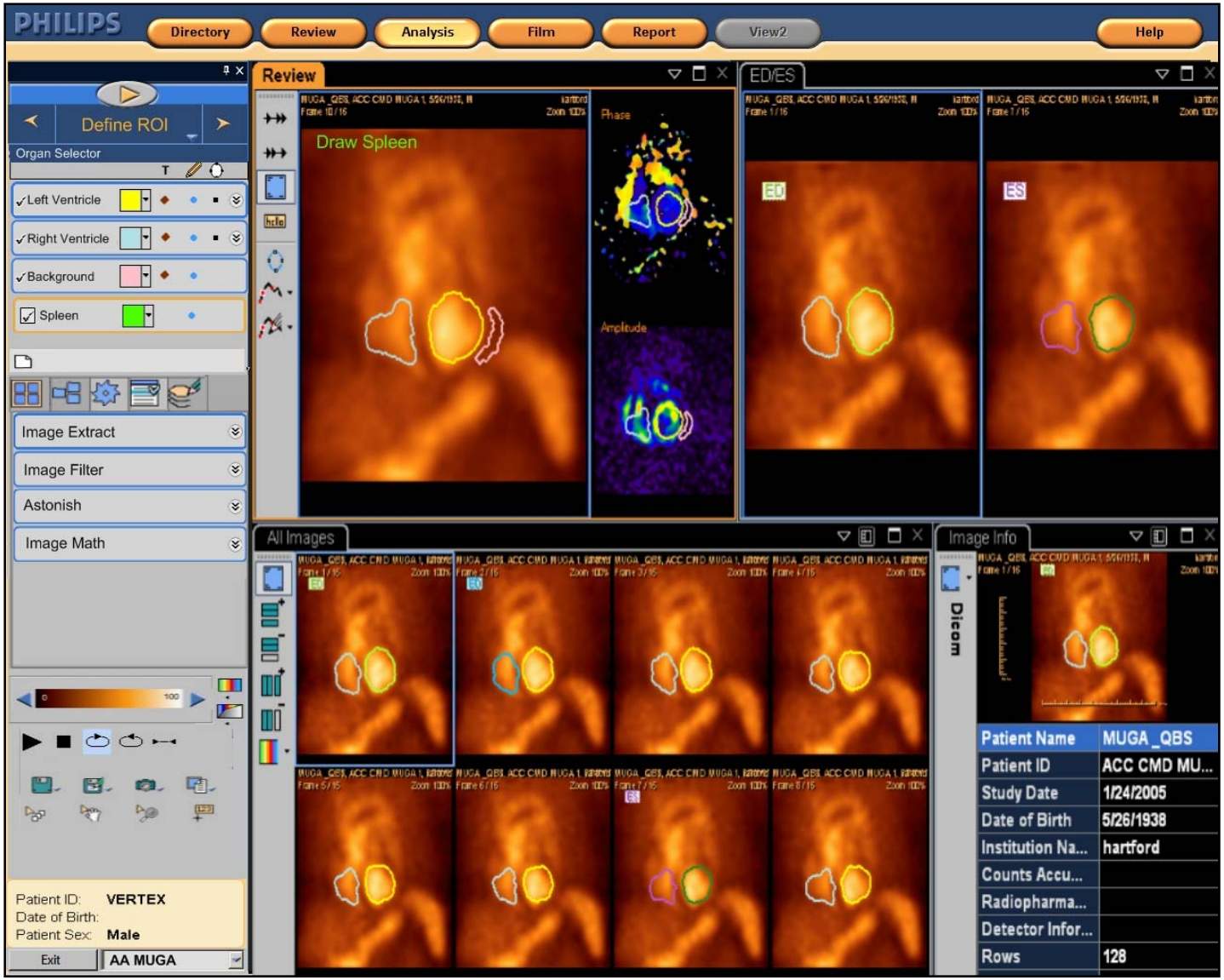
Agenda

- Motivation
- Challenges
- Details of test automation framework
- Results & Lessons Learnt
- References

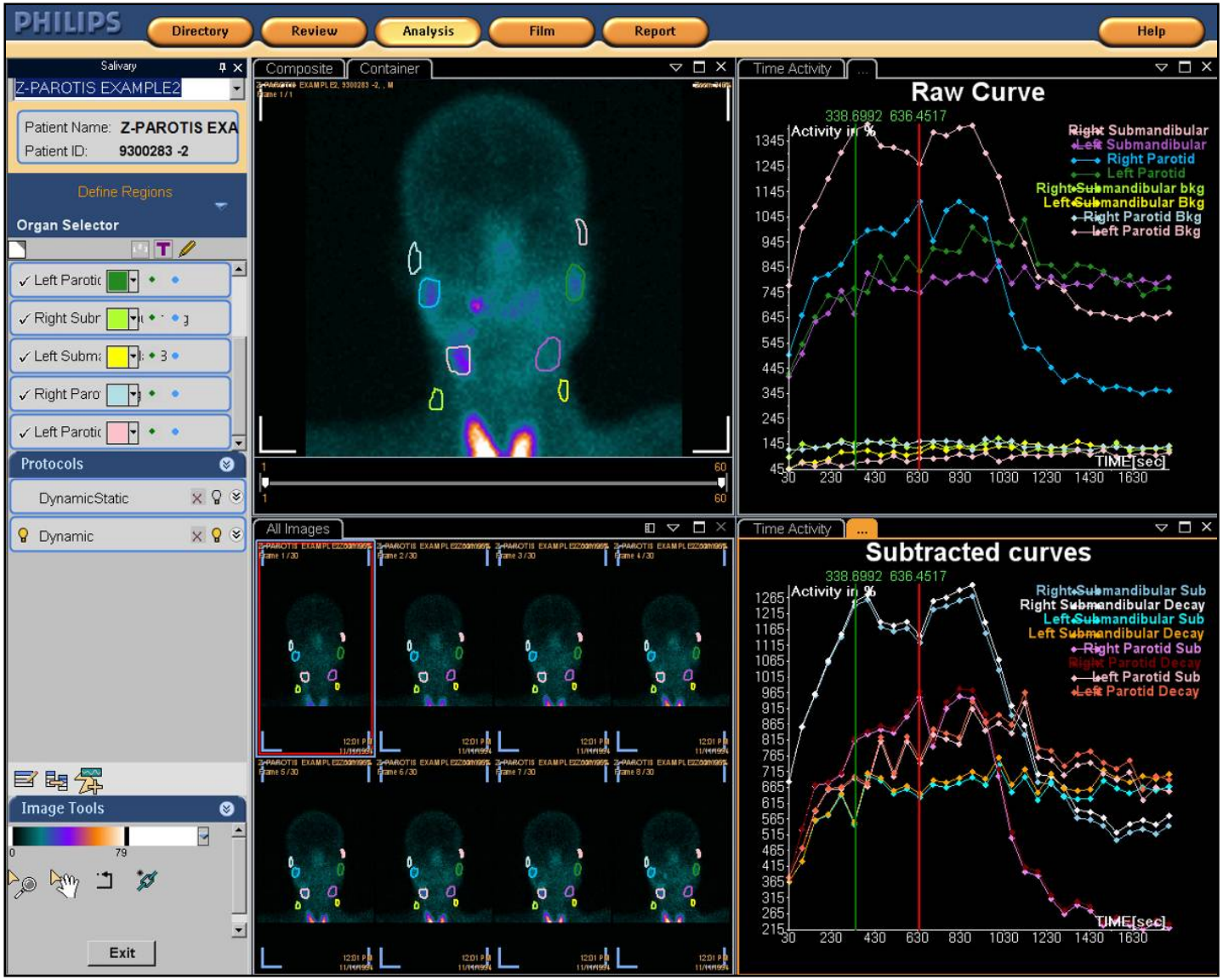
Motivation

- Most PH applications(Application Under Test) are GUI intensive and built on homegrown frameworks.
- Issues in using COTS tools
- Complete test automation solution of these applications is not possible with Commercially Off-the-Shelf(COTS) tools alone

Example of Clinical Application



Example of Clinical Application – contd.



PHILIPS

Challenges in using COTS tools

- Automation scripts built using COTS tools will not function on applications built against PH specific strong names
- Recording is in analog mode for custom GUI Controls
  - Record and playback test scripts are not reliable and need frequent modifications
- Does not meet more than 50% of our needs

Control Types in Our applications	Standard Vendor Support
Standard Controls(35%)	Supported
Compound Controls(20%) .(ex: ComboEditBox,ListView with Tree view Embedded in each row)	Not supported
Custom Controls(45%)	Not Supported

- Relatively big foot print on test systems
- Lack of seamless integration with development environment

PHILIPS

Challenges specific to visualization applications

- Recognition of “Image Viewer” control along with its nested controls.
- Recognition of graphic and textual annotations on images
  - Ellipse
  - Point
  - Polygon
  - Line
  - Text
- Validation of image data (image quality and algorithm results)

Key aspects of the solution

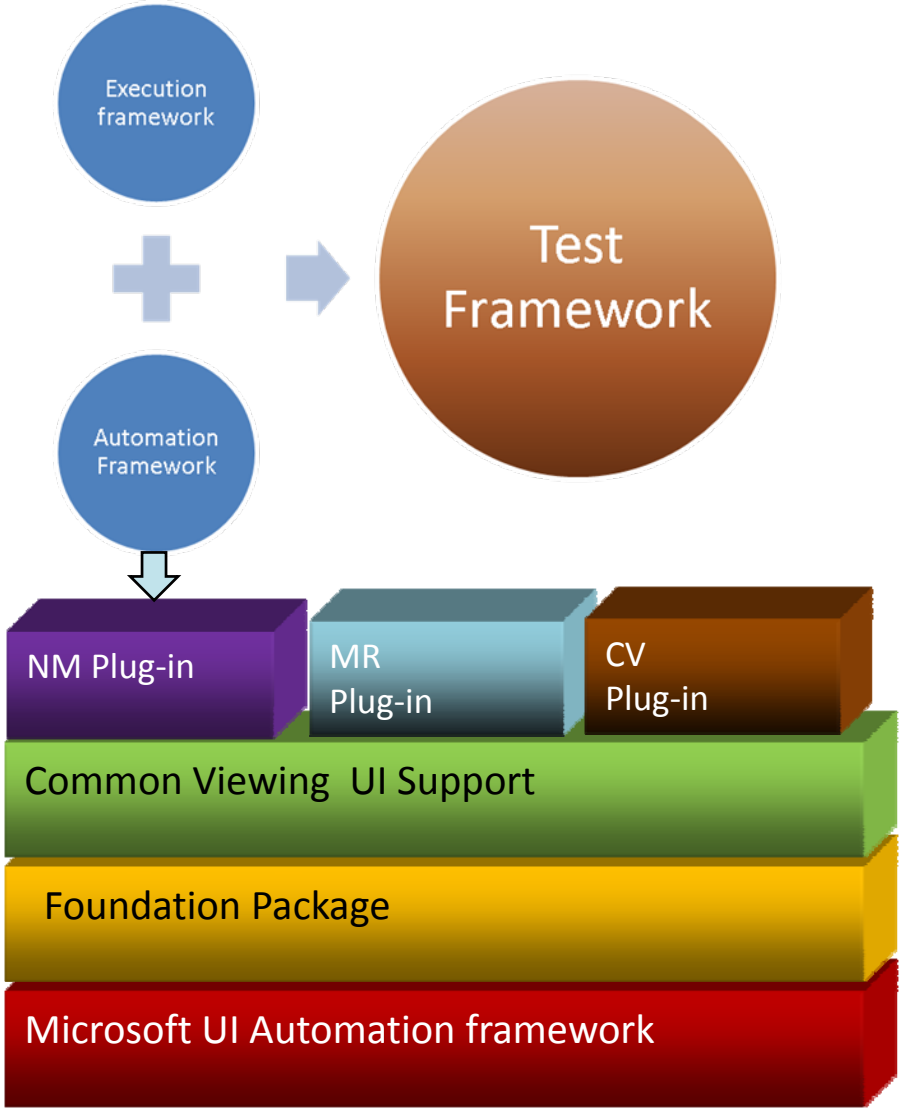
- Hybrid solution combining aspects of the Microsoft automation framework and home-grown automation mechanisms
- Comprehensive custom recognition mechanism for GUI and non-GUI objects
- Application Programmable Interface (API) support for frequently used clinical operations on medical images
- Extensibility mechanism for modality(CT/US/NM/CV) specific plug-ins.
- Customized reporting mechanisms compliant to regulatory requirements (e.g. FDA)

Key aspects of the solution

- Tools to build test cases and clinical workflows
- X-Copy deployment across multiple test nodes
- Mechanism that enables concurrent execution across multiple test nodes
- Low foot-print

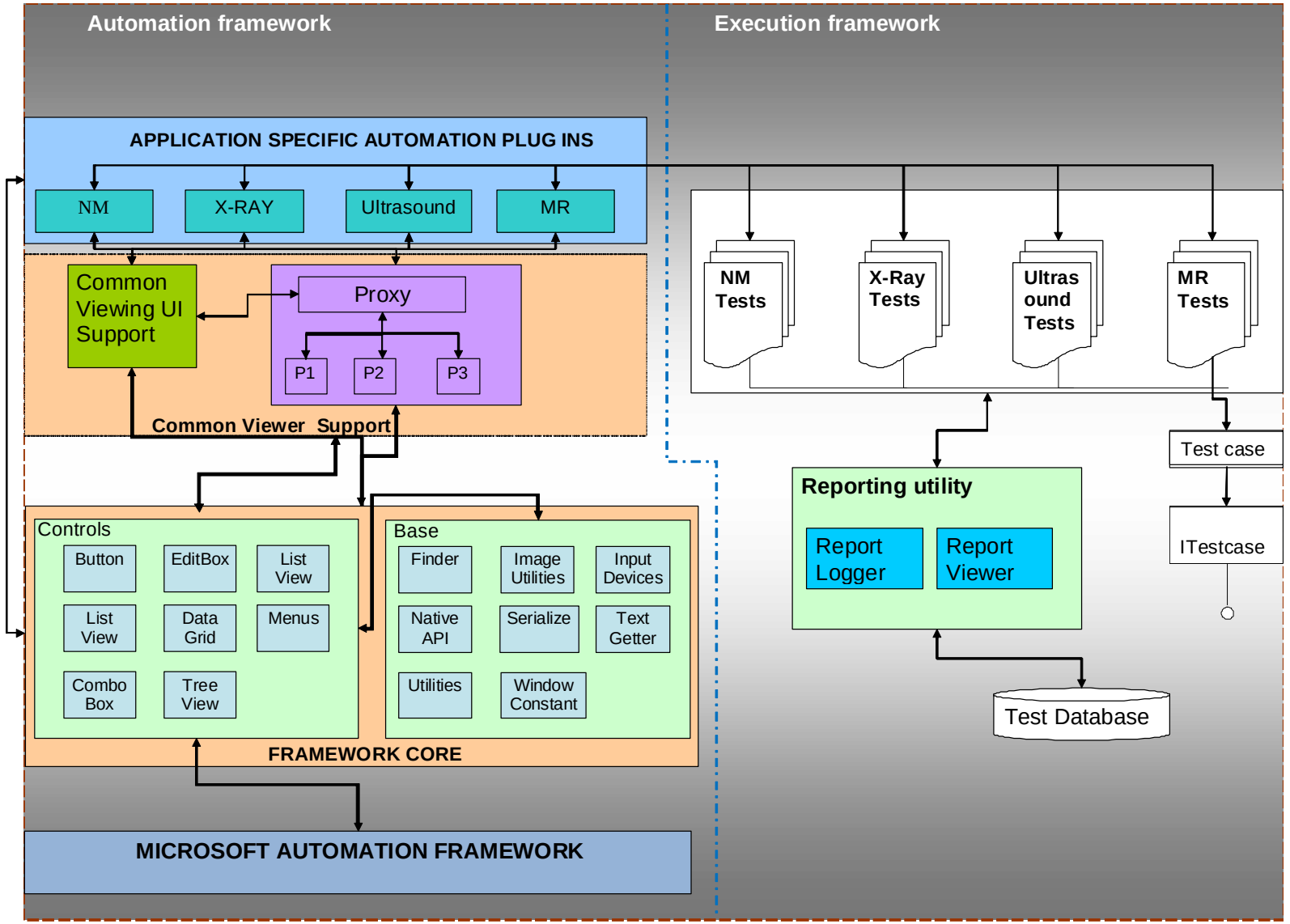
PHILIPS

Architecture of the test framework



PHILIPS

Architecture of the test framework - contd

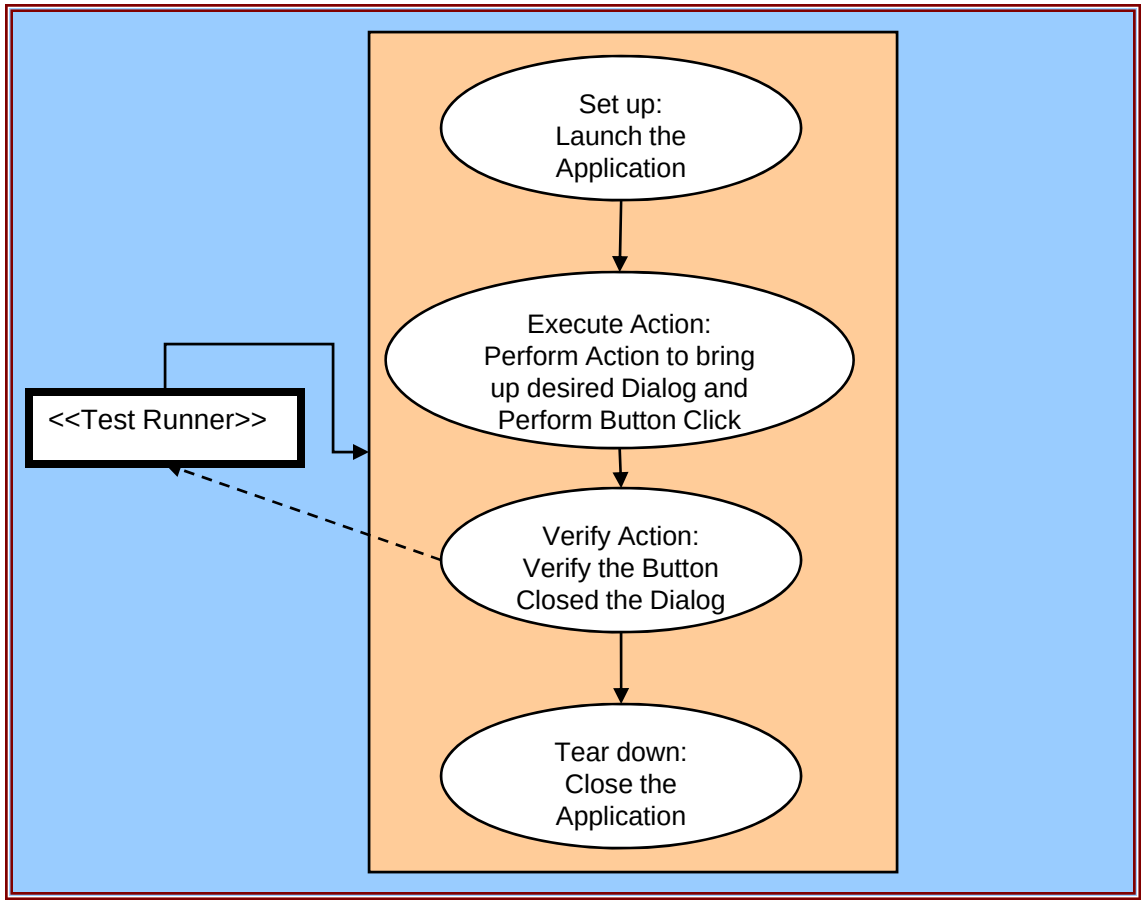


Steps in developing a test case using test framework

- Segmented Test Automation Approach
  - Test is divided into 4 parts
    - Pre condition
    - Action
    - Verification of Action Performed
    - Post Condition
- Define a test Case Model.
- Every test (each Test case is an Object) will be atomic in nature to enable work flow chaining
- Support execution from Standard runners like NUnit by providing interface generators.

PHILIPS

Example of a test case written using test automation framework



PHILIPS

Example of a test case written using test automation framework

```
public class Test001 : TestCase, ITestCase
{
    #region ITestCase Members
    [TestcaseSetup]
    public override object Setup(params object[] parameters)
    {
        ClientApplication.LaunchApplication(true);
        return null;
    }
    [TestcaseExecute]
    public override object ExecuteAction(params object[] parmaters)
    {
        return null;
    }
    [TestcaseVerify("VERIFY")]
    public override object VerifyAction(params object[] parameters)
    {
        bool isDisplayed =
             Navigator.IsDisplayed(Navigator.Button.Forward);
        return null;

    }
    [TestcaseTeardown]
    public override object Teardown(params object[] parmaters)
    {
        ClientApplication.CloseApplication();
        return null;
    }
    [ReUsable]
    public override object ReusableFunction(params object[] paraters)
    {
        return null;
    }
    #endregion
}
```

Results

- Deployment scenario1:
  - 4700 integration test cases automated, with an execution time of 26 hours
  - manual effort of 80 person days.
  - Also deployed against five versions of the product.
- Deployment scenario 2:
  - one suite of clinical applications automated, i.e. about 750 test cases, with an execution time of about 5 hours
  - manual effort of 8 person days.

Lessons Learnt

- End-to-end-automation solution: A hybrid solution combining aspects of standard automation solutions (e.g. Microsoft UI Automation) and home-grown solutions is more likely to address end to end automation needs
- Plug-in architecture: The concept of an automation framework to provide plug-in mechanisms so that application specific extensions can be met, is an important design consideration
- Cost Benefit: Initial investment on the framework development is high, which pays off during subsequent test case development
- Ease of use: the framework should be so designed that it can be made an integral part of the testing activity; should not call for special skills from the testing community (e.g. programming) to use

PHILIPS

References

- www.codeplex.com
- UI Automation Overview: http://msdn.microsoft.com/en-us/library/ms747327.aspx
- Nunit: www.nunit.org


PHILIPS