XML based Test Automation Framework

Proposal for Graduate Project

Submitted By:

First, Last
Someone@my.csun.edu
Student ID: xxxxxxxxx
September 19, 2011
Committee Chair: Dr. Shan Barkataki

Approved by: _____
Dr. Shan Barkataki

Table of Contents

OBJECTIVE

To design and implement test scheduling and metrics into an open source XML based automation framework called “iValidator”. XML based test metrics from test results and test scripts will be enabled. An open source static code analyzer will be integrated into the tool to enable code analysis. Based on test metrics and static code analysis a schedule for tests to be run on a priority basis will be generated. The test metrics and results shall be displayed in the GUI.

To design and implement the following capabilities into an open source XML based automation test framework called “iValidator”:

- Integrating a static code analyzer
- A test scheduling algorithm to generate the test execution schedule
- Software metrics measurements
- GUI to view test reports that combines test results and test metrics

INTRODUCTION

There are many approaches to software testing, but effective software testing is a time and effort intensive task that, given the size of modern systems, cannot be readily performed without the help of tools [3]. Many of the available automation test tools are language dependent which mandates tester knowledge of the language and minimal portability. XML based testing allows software testers to develop test procedures that is not in tester's native language [3].

The advantages from the use of XML are: [2]

- Test scripts are easy to edit and maintain
- Content review becomes much easier, as the relevant aspects of the script can be rendered in a readable form, such as HTML.
- Global changes become feasible, as the structured nature of XML documents makes it possible to search for specific element content.
- The structure of the XML document can itself be checked

iValidator is an open source efficient testing tool for professional testers and developers.Tests can be run on different platforms without any modification to the test scripts [5]. iValidator has the following features [5]:

- Supports the following types of testing:
    - Unit tests
    - Integration tests
    - Acceptance tests
- Test case Description:
  The framework allows reading test descriptions from any source (e.g. databases or Excel sheets). Accordingly the results of the test runs can be written to any target.
- Validation, Reporting, Logging:

iValidator separates the test execution from the validation of the results. Special checkup classes validate the results and thus allow a context-dependent evaluation within various scenarios.
- Adapters are used to wrap the access to the system under test (SuT) and other external systems.

I propose to extend this framework to include:
- A scheduling algorithm to prioritize the order in which test cases should be executed
- Integrating open source static code analyzer to perform code analysis
- Develop software metrics measurement component into the iValidator
- GUI to view test report which combines test results and test metrics
- Provide ability to perform analysis from the test report over a period of time

The proposed test framework has below mentioned advantages:
- XML based scripts and data are easy to create, maintain, re-use and portable
- The tool acts as a common platform to analyze error reported.
- Producing more systematic and repeatable software testing and generating more consistent test results [4]
- Acts as a single platform for developing and modifying test scripts, and to report test results.
- Integrating static code analysis and test metrics to schedule priority of test cases to be executed will provide risk analysis and risk identification for modules that are under test.
- Display of test report by combining test results and metrics helps in decision making
- Automatic scheduling of test cases from a test suite depending on the complexity, stability factor, frequency of changes, design issues, failure rate etc., reduces manual intervention in prioritizing test cases.

TECHNICAL APPROACH

The current architecture of iValidator is as shown below:

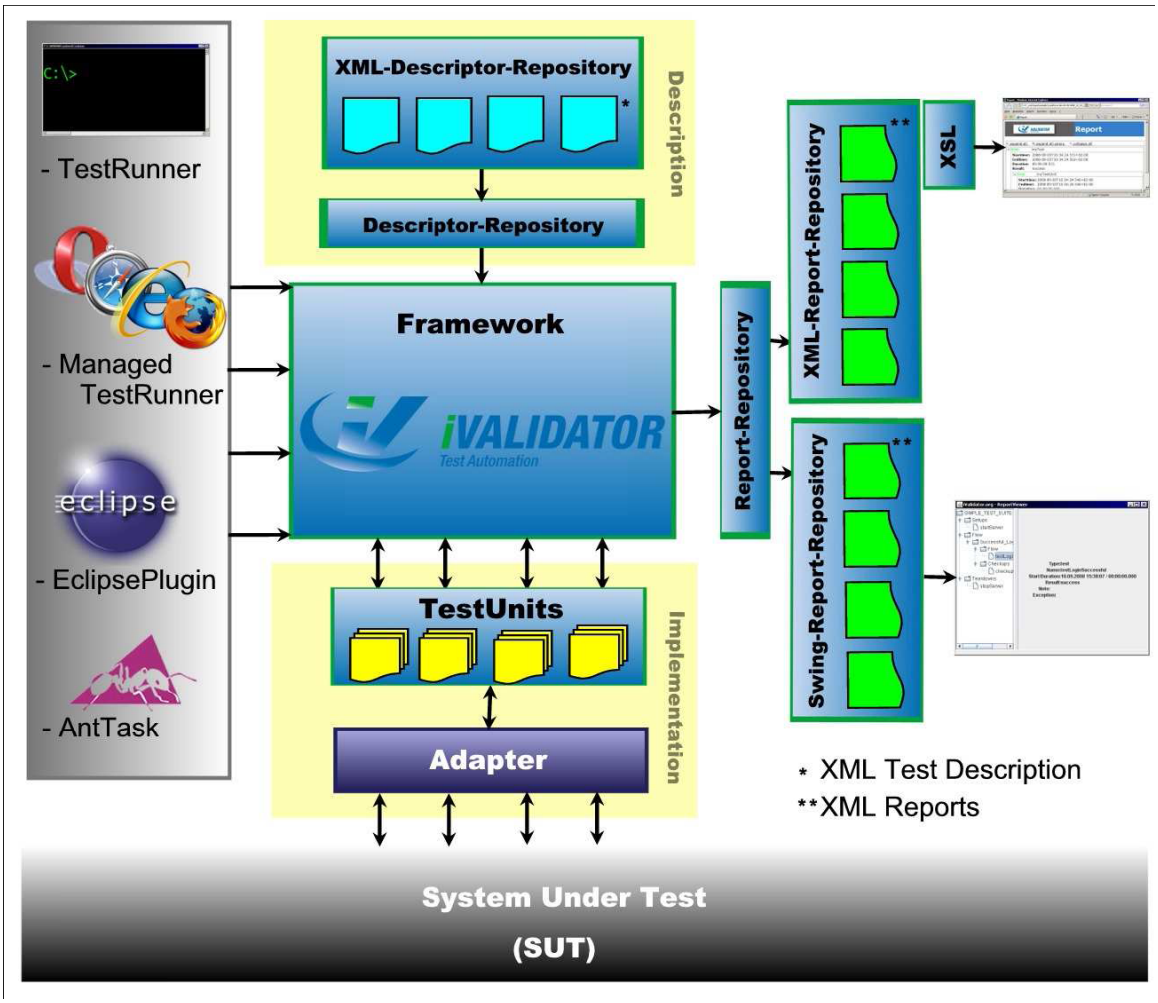
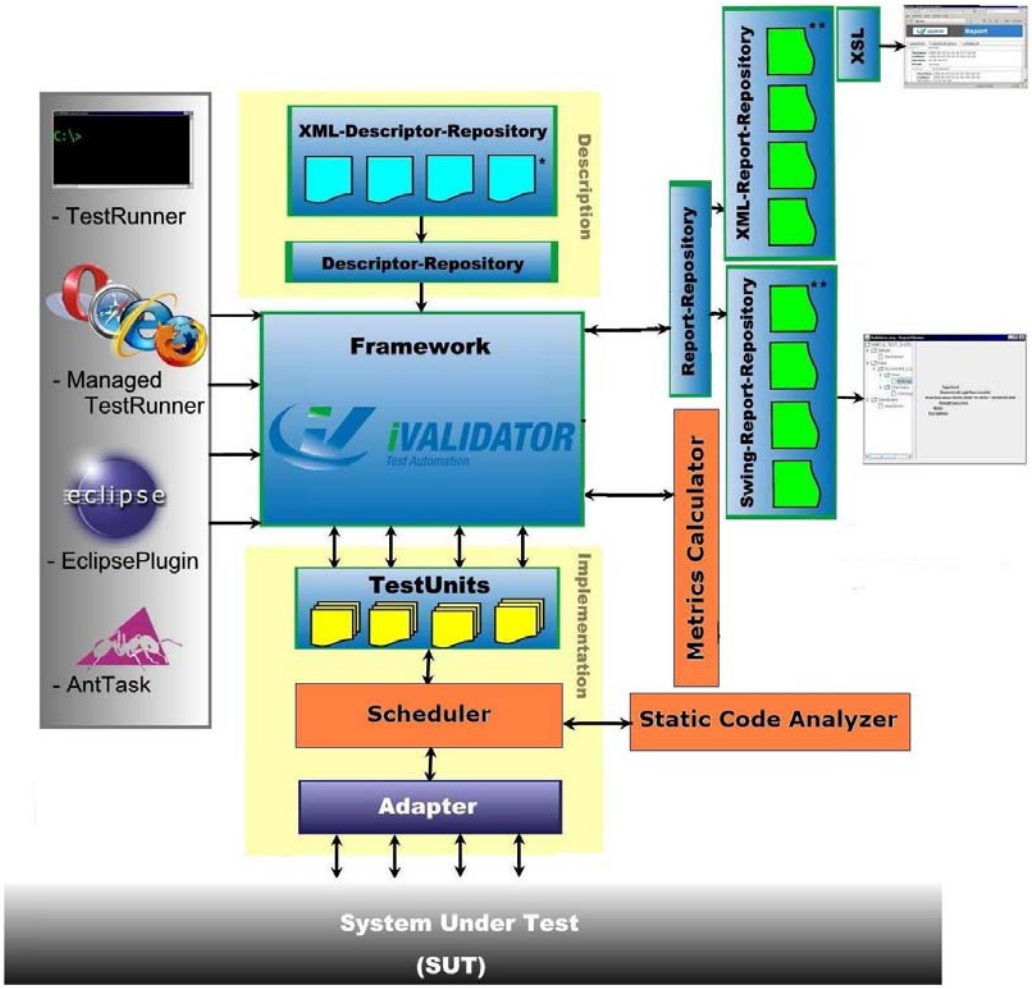


Figure 1: Architecture of iValidator [5]

The proposed architecture is as given below:



Modules to be developed

Figure 2: Proposed Architecture of iValidator

The features of modules to be integrated with iValidator are as below:

- Scheduler: This module will generate a list of prioritized test cases to be executed.
  Input:
  - XML test scripts for test cases to be executed
  - Test metrics from Metrics Calculator
  - Results in XML format from Static Code Analyzer

  Outputs:
  - The prioritized order in which the test cases should be executed
  - Return the test results to the framework

  Functionality:
  - A built in algorithm would perform analysis on test metrics, static code analysis results, and the test scripts to determine the prioritized order in which test scripts should be executed.

- Send prioritized order of test cases to the adapter for execution
- Invoke the static code analyzer

- Static Code Analyzer
  - Integrate an open source tool like TPTP plug-in with Eclipse to perform static code analysis of Java code
  - Generate the results in XML format

- Metrics Calculator: This module calculates software metrics based on defined formulas and stores it in XML format.

  Input:
  - History of test results from previous runs
  - Current test run results

  Output:
  - Different test metrics shall be calculated:
    - Number of test cases run
    - Number of test cases failed
    - Number of test cases passed
    - Number of unstable modules
    - Number of stable modules
    - Complexity of each module
    - Number of times a test script has been modified
    - Different types of bugs and number per module
    - A graph displaying a trend on the bugs found vs. runs over a period of time for each test case
    - A graph of complexity vs. unstable modules

  Functionality:
  - The metrics calculator will analyze the inputs and calculate the different types of test metrics
  - The test metrics will be generated in XML format

The development of the framework will follow a comprehensive iterative software development lifecycle.

Literature study: The project will start with literature study to gain information about XML based automation techniques, iValidator tool, software metrics, scheduling algorithm from various sources including journal publications, case studies, forums, websites, and company websites. Open source software static code analysis tool that involves XML based scripting will be analyzed. A document on the conclusions from literature study will be produced as a deliverable.

Requirements and design: The features described will be translated into requirements and design for the framework will be formalized based on the literature study and technical feasibility. The deliverable at the end of this stage will be requirement and design document.

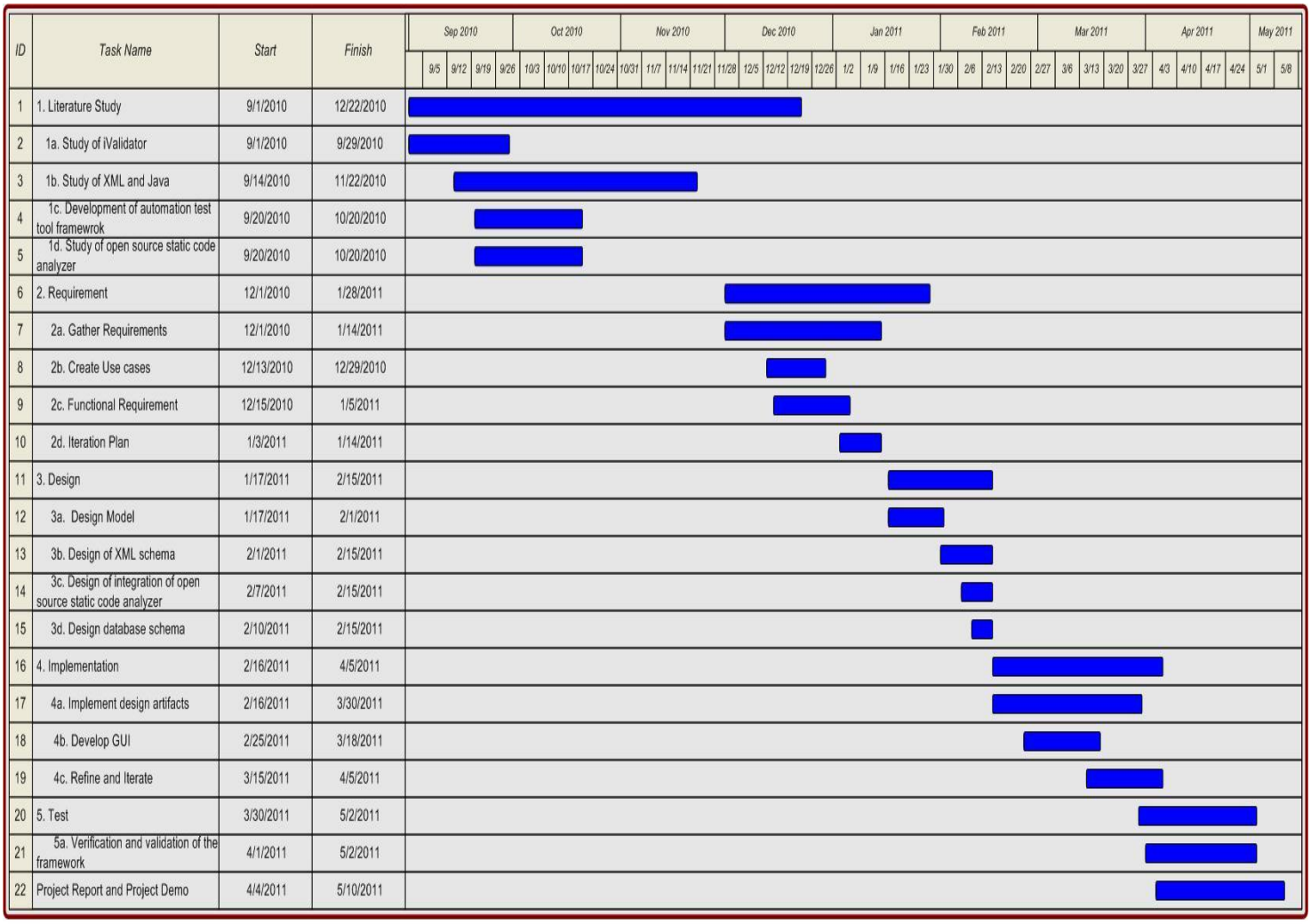
Development: The test framework will be developed based on requirements and design document. XML standards templates and schemas will be generated. The deliverable at the end of this stage is fully functional code.

Test: The developed test framework will be verified and validated by generating test scripts and executing the tests on open source software. The test results and schedule generated by the framework will be verified for correctness and accuracy. The deliverable will be a document containing the test scripts and test results generated by the test framework.

Tasks:

- Literature study
  - Study of iValidator code and architecture in Eclipse
  - Study of XML
  - Study of automation test tool frameworks
  - Study of open source static code analyzer tools
  - Study of software metrics
- Requirement
  - Generate requirements document on the basis of knowledge gained by literature study and expected features for the test framework
- Design
  - Develop high level and detailed design for the components to be added to the framework
  - Design repository schema for test metrics
  - Design XML Schemas for test report and test metrics
  - Design for integrating open source static code analyzer
- Implementation
  - Develop modules as per design
  - GUI for the extended features
- Test
  - Verification and validation of the requirements on an open source software
- Project Demo
- Project Report

SCHEDULE



ID	Task Name	Start	Finish
1	1. Literature Study	9/1/2010	12/22/2010
2	1a. Study of iValidator	9/1/2010	9/29/2010
3	1b. Study of XML and Java	9/14/2010	11/22/2010
4	1c. Development of automation test tool framewrok	9/20/2010	10/20/2010
5	1d. Study of open source static code analyzer	9/20/2010	10/20/2010
6	2. Requirement	12/1/2010	1/28/2011
7	2a. Gather Requirements	12/1/2010	1/14/2011
8	2b. Create Use cases	12/13/2010	12/29/2010
9	2c. Functional Requirement	12/15/2010	1/5/2011
10	2d. Iteration Plan	1/3/2011	1/14/2011
11	3. Design	1/17/2011	2/15/2011
12	3a. Design Model	1/17/2011	2/1/2011
13	3b. Design of XML schema	2/1/2011	2/15/2011
14	3c. Design of integration of open source static code analyzer	2/7/2011	2/15/2011
15	3d. Design database schema	2/10/2011	2/15/2011
16	4. Implementation	2/16/2011	4/5/2011
17	4a. Implement design artifacts	2/16/2011	3/30/2011
18	4b. Develop GUI	2/25/2011	3/18/2011
19	4c. Refine and Iterate	3/15/2011	4/5/2011
20	5. Test	3/30/2011	5/2/2011
21	5a. Verification and validation of the framework	4/1/2011	5/2/2011
22	Project Report and Project Demo	4/4/2011	5/10/2011

CRITERIA FOR SUCCESS

- Provide a detailed explanation and demonstration of iValidator tool
- Successful completion of the requirements analysis to meet the objectives outlined in this proposal
- Creation of the requirements and design work products (in UML) for the proposed project
- Implementation of the design
- Demonstration of a working model of the proposed software application

REFERENCES

1) Antonia Bertolino, Jinghua Gao, Eda Marchetti, Andrea Polini; " TAXI - A Tool for XML-Based Testing", 29th International Conference on Software Engineering (ICSE'07 Companion)
ISBN: 0-7695-2892-9
DOI: 10.1109/ICSECOMPANION.2007.72
Year of Publication: 04 June 2007 2007
Url: http://ieeexplore.ieee.org/Xplore/

2) Colin Bird, Andrew Sermon; "An XML-based approach to automated software testing",
ACM SIGSOFT Software Engineering
Volume 26, Issue 2 (March 2001), pages: 64 - 65
Year of Publication: 2001
ISBN: 0163-5948
Url: http://portal.acm.org/citation.cfm?id=505776.505792

3) R¨udiger Foos, Christian Bunse, Hagen H¨opfner, Torsten Zimmermann; "TML: An XML- based test modeling language", ACM SIGSOFT Software Engineering,
Volume 33, Issue 2 (March 2008)
Article No.: 1
Year of Publication: 2008
ISBN: 0163-5948
Url:http://portal.acm.org/citation.cfm?id=1350809&dl=GUIDE&coll=GUIDE&CFID=99346747&CFTOKEN=54707527

4) Book by Jerry Gao, H.-S. J. Tsao, Ye Wu; "Testing and quality assurance for component- based software", 2003 Artec inhouse Inc
Url: http://books.google.com

5) iValidator tool
Url: http://ivalidator.org/