(12)Indian Patent Application

---

(21) Application Number: 201741009637

(22) Filing Date: 20/03/2017 (43) Publication Date: 28/09/2018

(71) Applicant(s): L&T TECHNOLOGY SERVICES LIMITED

(72) Inventor(s): MANJESH, HB

(51) International Classifications: G06F 11/00

(54) Title: ONE-CLICK FRAMEWORK FOR SOFTWARE TESTING

(57) Abstract: A software testing framework for automatically executing a plurality of testing types on a software application is disclosed. The framework consolidates outcome of each of the plurality of testing types in a single output. The framework receives one or more configuration files as an input such that the configuration files define one or more of the testing types to be executed. The received one or more configuration files are scanned and a test bed module is invoked to create a plurality of corresponding test beds for the one or more of the testing types defined in the one or more configuration files. The test bed module installs a build or application under test (AUT), fetches one or more test scripts corresponding to the testing types from a database, installs automation tools corresponding to the testing types and fetches test logic corresponding to the testing types from a generic library module. Further, the test bed module is configured to create a test runner client specific to each of the testing types such that the test runner client executes the test scripts and communicates test reports to a test runner server.

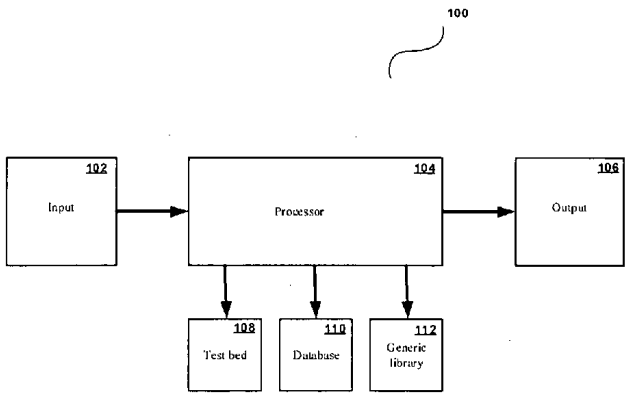
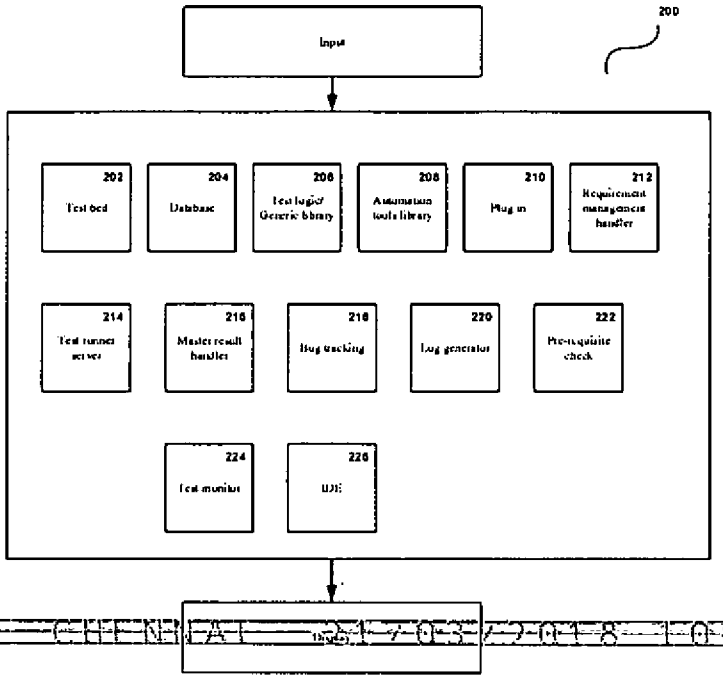


Figure 1

ABSTRACT

One-click Framework for Software Testing

A software testing framework for automatically executing a plurality of testing types on a software application is disclosed. The framework consolidates outcome of each of the plurality of testing types in a single output. The framework receives one or more configuration files as an input such that the configuration files define one or more of the testing types to be executed. The received one or more configuration files are scanned and a test bed module is invoked to create a plurality of corresponding test beds for the one or more of the testing types defined in the one or more configuration files. The test bed module installs a build or application under test (AUT), fetches one or more test scripts corresponding to the testing types from a database, installs automation tools corresponding to the testing types and fetches test logic corresponding to the testing types from a generic library module. Further, the test bed module is configured to create a test runner client specific to each of the testing types such that the test runner client executes the test scripts and communicates test reports to a test runner server.



We Claim:

1. A computer implemented software testing architecture system for automatically executing a plurality of testing types on a software application and consolidating outcome of each of the plurality of testing types in a single output, the system comprising a processor configured to:

receive one or more configuration files as an input such that the configuration files define one or more of the testing types to be executed;

scan the received one or more configuration files and invoke a test bed module to create a plurality of corresponding test beds for the one or more of the testing types defined in the one or more configuration files, the test bed module configured to:

install a build or application under test from information specified in the one or more configuration files;

fetch one or more test scripts corresponding to the one or more testing types from a database;

install one or more automation tools corresponding to the one or more testing types;

fetch test logic corresponding to the one or more testing types from a generic library module; and

create a test runner client specific to each of the one or more testing types such that the test runner client executes the test scripts and communicates one or more test reports to a test runner server.

2. The system as claimed in claim 1, wherein the one or more configuration files is inputted through a user interface.

3. The system as claimed in claim 1, wherein the test beds are created in LAN or cloud based on the information in the configuration files.

4. The system as claimed in claim 1, wherein the plurality of testing types includes both functional and non-functional testing types.

5. The system as claimed in claim 1, wherein the plurality of testing types includes functionality testing, integration testing, compatibility testing, localization/internationalization testing, usability testing, performance testing, security testing and system testing.

6. The system as claimed in claim 1, wherein the test reports may include detailed report on the test results of the one or more testing types specified in the one or more configuration files.

Dated this 20th day of March 2017

Mohammed Faisal (INPA No: 1941)
Head, IPR Dept.
L&T Technology Services Limited
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai, TN, 600089

FIELD OF INVENTION

The invention generally relates to system and methods for software testing and more particularly to providing a single test framework for all types of standard testing in software.

BACKGROUND

A Test Automation Framework is a set of guidelines like coding standards, test-data handling, object repository treatment etc., which when followed during automation scripting produce beneficial outcomes like increase code re-usability, higher portability, reduced script maintenance cost etc.

Various testing automation frameworks are available such as Module Based Testing Framework, Library Architecture Testing Framework, Data Driven Testing Framework, Keyword Driven Testing Framework, Hybrid Testing Framework and Behavior Driven Development Framework. However, the existing testing automation frameworks have their own advantages and drawbacks.

Module Based Testing Framework introduces high level of modularization which leads to easier and cost efficient maintenance. Also, the framework is pretty much scalable. However, the problems of the framework are the separations made between test scripts. Developers usually implement test scripts separately for each module and then embed test data onto them. Testing a different set of data will always require some extra work on test scripts – and this is quite time-consuming.

Like Module Based Framework, Library Architecture Testing framework also introduces high level of modularization and a great degree of re-usability. However, with the introduction of libraries, the framework becomes a little complicated.

Data Driven Testing Framework considerably reduces the total number of scripts required to cover all the possible combinations of test scenarios. Thus, lesser amount of code is required to test a complete set of scenarios. The drawback is that framework is complex and requires an extra effort to come up with the test data sources and reading mechanisms.

In addition to advantages provided by Data Driven testing, Keyword driven framework doesn't require the user to possess scripting knowledge unlike Data Driven Testing. A single keyword can be used across multiple test scripts. However, the user should be well versed with the keyword creation mechanism to be able to efficiently leverage the benefits provided by the framework. The framework becomes complicated gradually as it grows and a number of new keywords are introduced.

Hybrid testing frameworks are basically combinations of at least two different testing frameworks listed above.

Another known testing tool is SOAP-UI. It is an open source cross-platform API Testing tool. SOAP-UI allows testers to execute automated functional, regression, compliance, and load tests on different Web API. SOAP-UI supports all the standard protocols and technologies to test all kinds of API's. However, SOAP-UI is not used for user interface testing.

The above-mentioned framework types specify only standard definitions for expected actions and create a method to drive an application under test. However, a framework is required that supports standard definitions to create test scripts for particular testing type (e.g. localization testing) and creates the method to drive the application under test based on the testing types selected by the tester.

Thus, a new framework for test automation is needed that supports multiple standard testing types in black box testing of software.

The present invention is directed to overcoming one or more of the problems as set forth above.

SUMMARY OF THE INVENTION

Exemplary embodiments of the invention disclose a computer implemented software testing architecture system for automatically executing a plurality of testing types on a software application and consolidating outcome of each of the plurality of testing types in a single output. The system comprises a processor that is configured to receive one or more configuration files as an input such that the configuration files define one or more of the testing types to be executed. The received one or more configuration files are scanned and a test bed module is invoked to create a plurality of corresponding test beds for the one or more of the testing types defined in the one or more configuration files. The test bed module is configured to install a build or application under test (AUT) from information specified in the one or more configuration files. The test bed module fetches one or more test scripts corresponding to the one or more testing types from a database, installs one or more automation tools corresponding to the one or more testing types and fetches test logic corresponding to the one or more testing

types from a generic library module. Further, the test bed module is configured to create a test runner client specific to each of the one or more testing types such that the test runner client executes the test scripts and communicates one or more test reports to a test runner server.

BRIEF DESCRIPTION OF DRAWINGS

Other objects, features, and advantages of the invention will be apparent from the following description when read with reference to the accompanying drawings. In the drawings, wherein like reference numerals denote corresponding parts throughout the several views:

Figure 1 illustrates a software testing architecture system for executing multiple standard testing types, according to an exemplary embodiment of the invention;

Figure 2 illustrates a test automation framework for executing multiple standard testing types, according to an exemplary embodiment of the invention; and

Figure 3 illustrates a block diagram of a process for executing multiple standard testing types using a test automation framework, according to an exemplary embodiment of the invention.

DETAILED DESCRIPTION OF DRAWINGS

The following description with reference to the accompanying drawings is provided to assist in a comprehensive understanding of exemplary embodiments of the invention. It includes various specific details to assist in that understanding but these are to be regarded as merely exemplary. Accordingly, those of ordinary skill in the art will recognize that various changes and modifications of the embodiments described herein can be made without departing from

the scope and spirit of the invention. In addition, descriptions of well-known functions and constructions are omitted for clarity and conciseness.

Exemplary embodiments of the invention disclose a method and system for providing a single framework that supports multiple standard software testing types. The standard testing types may include different types of software testing such as, but not limited to, functionality testing, integration testing, compatibility testing, localization/internationalization testing, usability testing, performance testing, security testing and system testing. According to an exemplary embodiment, on event of a single click on a user interface of the framework, multiple standard testing types supported by the framework are executed and a test report is generated.

According to an embodiment, the framework supports execution of functional testing and performance testing simultaneously.

Figure 1 illustrates an exemplary software testing architecture system 100 for automatically executing a plurality of testing types on a software test application and consolidating outcome of each of the plurality of testing types in a single output, according to an exemplary embodiment of the invention.

The disclosed system 100 may include an input module 102, a processor 104 and an output module 106.

The input module 102 may receive one or more configuration files. According to an embodiment, the one or more configuration files may be received through a user interface. DRAFT According to an exemplary embodiment, the user interface may be a graphical user interface.

The one or more configuration files may define one or more testing types to be executed by the system 100. The testing types may include functional testing as well as non-functional testing. According to one embodiment, the test types may include different types of software testing such as, but not limited to, functionality testing, integration testing, compatibility testing, localization/internationalization testing, usability testing, performance testing, security testing and system testing. According to another embodiment, the configuration file may include IP addresses of machines in which tests are executed. The machines may be on LAN, cloud or any other network. According to an exemplary embodiment, the configuration file may specify to run all the testing types in a single machine or run each testing type on individual machines. According to yet another embodiment, the configuration file may include build link of the test application and unique information required for any testing type.

The processor 104 is configured to receive the one or more configuration files from the input module 102. The processor may scan the received configuration files and invoke a test bed module 108 to create a plurality of corresponding test beds for the testing types defined in the configuration files. The processor 104 may generate test results on an output module 106.

The test bed module 108 may install a build or application under test from information specified in the configuration file. According to an embodiment, the configuration file may specify to run all the testing types in a single machine. The machine may be located in a LAN or cloud. Accordingly, the test bed module 108 may create a setup for all the testing types in the single machine with high configuration. According to another embodiment, the configuration file may specify to run each testing type on each individual machine. Accordingly, the test bed module may initiate a test setup for each testing type in individual machines.

According to another embodiment, the test bed module 108 may fetch requirements corresponding to the one or more testing types from a requirements management module. According to yet another embodiment, the test bed module 108 may fetch test logic corresponding to the testing types from a generic library module 112. The test logic is a set of scripts required to setup /teardown a particular testing type. According to an embodiment, the test bed module 108 may fetch one or more test scripts based on the test logic, from a database 110. According to a further embodiment, the test bed module 108 may install one or more automation tools corresponding to the one or more testing types.

The test bed module 108 may create a test runner client specific to each of the one or more testing types. The test runner client may communicate with a test runner server. The test runner client may send log files of testing type to the test runner server. The test runner client may send test results of testing type to the test runner server. The test runner client may send a bug report of testing type to the test runner server. The test runner client may run the test logic scripts and create a test environment to initiate testing type. The test runner client may run a pre-requisite check to confirm if the test environment is ready for test execution of testing type. The test runner client may communicate test results of pre-requisite check to the test runner server. The test runner client may launch automated tool to start executing test scripts on application or build under test. The test runner client may send a complete test report to the test runner server. The test runner server may collaborate with a result handler module to send the test results to an output module 106. According to an embodiment, the test results may be stored in a data store. According to another embodiment, the test results may be displayed in regular intervals on a display of the output module 106.

According to an exemplary embodiment, the system 100 may report the test results to configured e- mail ids and send an SMS to predefined users for critical defects.

The output module 106 may include a display device. According to one embodiment, the display device may be any display such as, but not limited to, Cathode ray tube display (CRT), Light-emitting diode display (LED), Electroluminescent display (ELD), Plasma display panel (PDP) etc. According to another embodiment, the display may include a graphical user interface (GUI).

Figure 2 illustrates a test automation framework 200 for executing multiple standard testing types, according to an exemplary embodiment of the invention. The framework 200 may support both functional and non-functional tests. The framework 200 may be application independent and easy to expand, maintain and perpetuate. The framework 200 may support execution of the standard testing types in development as well as post-release of any application. According to an exemplary embodiment, one click on a user interface of the framework 200 may be sufficient to initiate functional and non-functional tests in scenario when major change occurs in application after release.

The test automation framework 200 receives one or more configuration files as input. The configuration file may specify the standard testing types that are to be executed by the framework 200. The standard testing types may include different types of software testing such as, but not limited to, functionality testing, integration testing, compatibility testing, localization/internationalization testing, usability testing, performance testing, security testing and system testing.

The test automation framework 200 consists of various modules including such as, but not limited to, test bed module 202, database 204, generic library/test logic module 206, automation tool library module 208, plug in module 210, requirement management handler module 212, test runner server module 214, master result handler module 216, bug tracking tool module 218, log generator module 220, prerequisite check module 222, test monitor module 224 and IDE module 226.

The test bed module 202 may create test beds for one or more standard testing types based on the information provided in the configuration script.

The test bed module 202 may have access to database 204, generic library/test logic module 206, automation tool library module 208, plug in module 210 and requirements management handler module 212.

The database module 204 may contain the test cases and test scripts for all the testing types. The test logic module 206 may also be called generic library module. The generic library module 206 may contain specific scripts of testing type. The generic library module 206 may also provide privilege to a user to add new test logic or specific script for the newly added testing types. The automation tool library module 208 may act as a repository of all the automation tools (.exe or executables) and also enable to add new automation tools. The plug-in module 210 may use automation tool library and enable availability of automation tool at execution time for the testing type. The plug-in module 210 enables availability of licensed as well as non-licensed automation tool. The requirement management handler module 212 may have access to a requirement management tool and provide relevant requirements in response to request from the test bed module at the time of test bed creation for each testing type.

The test bed module 202 may fetch relevant test scripts, test logic scripts and requirements, install automation tool and application under test, launch test runner client and make test environment ready for test execution either in LAN or cloud.

The test bed module 202 may create a test runner client specific to each of the one or more testing types. The test runner server module 214 may communicate with test runner clients of all the testing types. According to an embodiment, test runner server module 214 may communicate with the master result handler module 216, bug tracking tool module 218, log generator module 220, pre-requisite check module 222 and test monitor module 224.

The master result handler module 216 may consolidate all the test results of all the standard testing types that are selected in the configuration file and display the test results on a display or a user interface.

The bug tracking module 218 may be used to create bugs. When the bug tracking module 218 receives a bug report from the test runner server module, the bug tracking module 218 may initiate a bug creation process and display the bug ID in the user interface.

The log generator module 220 may consolidate logs based on a message from the test runner server for all the testing types and display the log details in user interface console in regular intervals.

The prerequisite check module 222 may be initiated after completion of test bed creation to ensure test bed is ready to kick off the test execution.

The test monitor module 224 may monitor health of all the test beds or testing types and display the health parameters on the user interface.

The IDE module 226 may provide integrated development environment to update latest test cases and test scripts related to testing types from a user. According to an embodiment, the user may add test logic for new testing types in the framework.

Figure 3 illustrates a block diagram of a process for executing multiple standard testing types using a test automation framework, according to an exemplary embodiment of the invention.

At step 302, a configuration file is received as an input to a test automation framework. The configuration file may define one or more testing types to be executed by the test automation framework. According to another embodiment, the standard testing types supported by the test automation framework may be such as, but not limited to, functionality testing, integration testing, compatibility testing, localization/internationalization testing, usability testing, performance testing, security testing and system testing.

At step 304, multiple standard testing types supported by the test automation framework are executed. According to an exemplary embodiment, the test automation framework may fetch test logic to support simultaneous execution of functional and performance testing. According to another embodiment, the framework uses various modules mentioned in Figure 2 to execute testing for all standard testing types.

At step 306, a test execution report is generated. According to an embodiment, the test report may include a detailed report on each of the test results of the various standard testing types that are executed by the test automation framework.

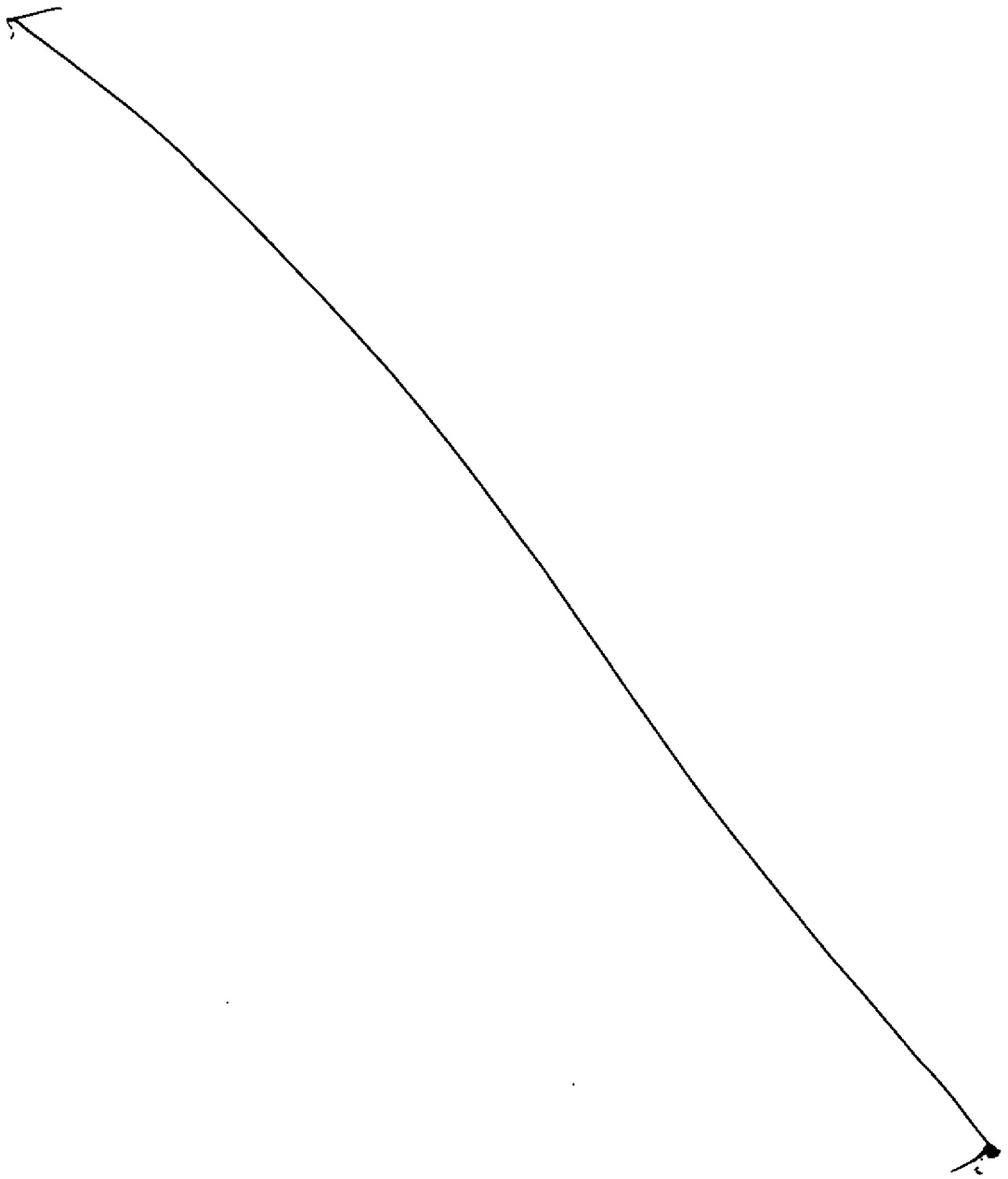
The present invention is directed to overcome problems prevailing in the existing test automation frameworks due to execution of functional tests and non-functional tests separately, such as, but not limited to, defect leakage, schedule impair in test cycle times, more rework post implementation, more time consuming, procuring expensive licenses for functionality and performance tools, requirement of both functional and non-functional testers especially at the time of post-production testing or products that are in maintenance phase, difficulty in running same configuration tests on different operating systems, difficulty in maintaining complex set of code for all the standard software testing types and difficulty in maintaining the large test data for different test configurations for all standard software testing types.

In the drawings and specification there has been set forth preferred embodiments of the invention, and although specific terms are employed, these are used in a generic and descriptive sense only and not for purposes of limitation. Changes in the form and the proportion of parts, as well as in the substitution of equivalents, are contemplated as circumstances may suggest or render expedient without departing from the spirit or scope of the invention.

Throughout the various contexts described in this disclosure, the embodiments of the invention further encompass computer apparatus, computing systems and machine-readable media configured to carry out the foregoing systems and methods. The present invention may be conveniently implemented using a conventional general purpose or a specialized digital

computer or microprocessor programmed according to the teachings of the present disclosure, as will be apparent to those skilled in the computer art.

Appropriate software coding can readily be prepared by skilled programmers based on the teachings of the present disclosure, as will be apparent to those skilled in the software art.



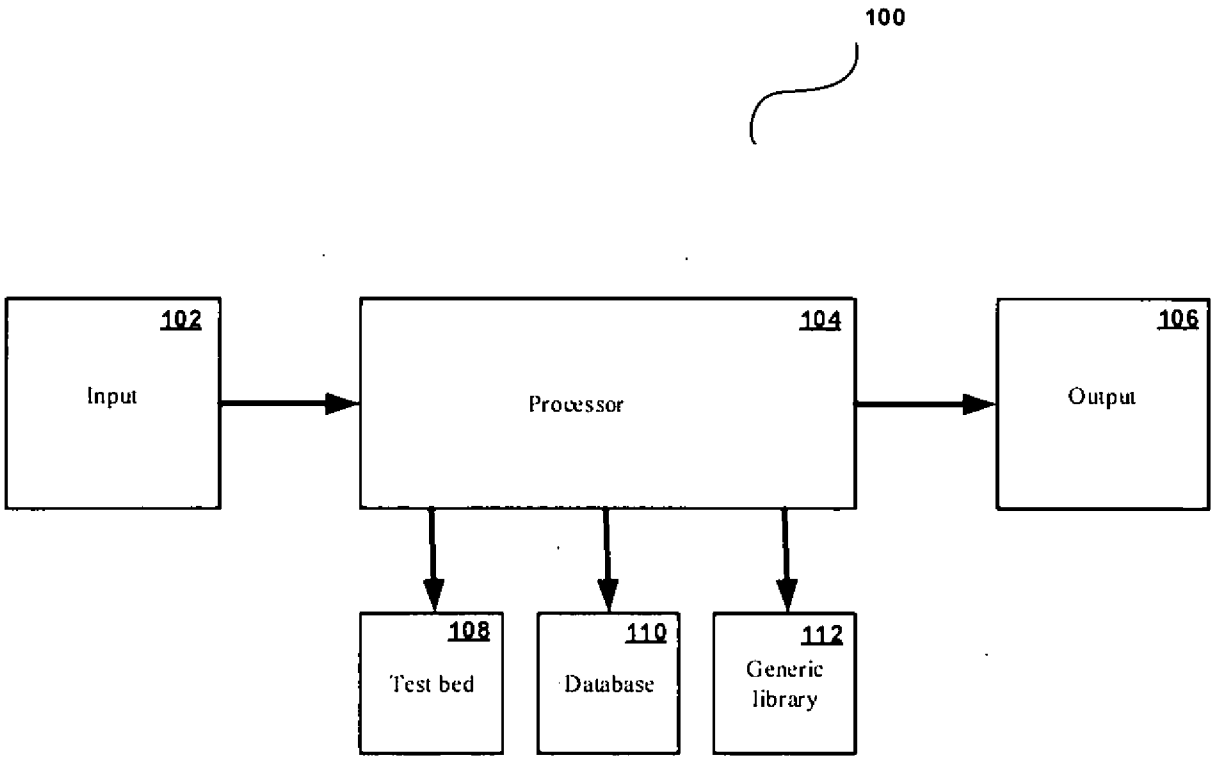


Figure 1

Mohammed Faisal (INPA No: 1941)
Head, IPR Dept.
L&T Technology Services Limited
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai – 600089

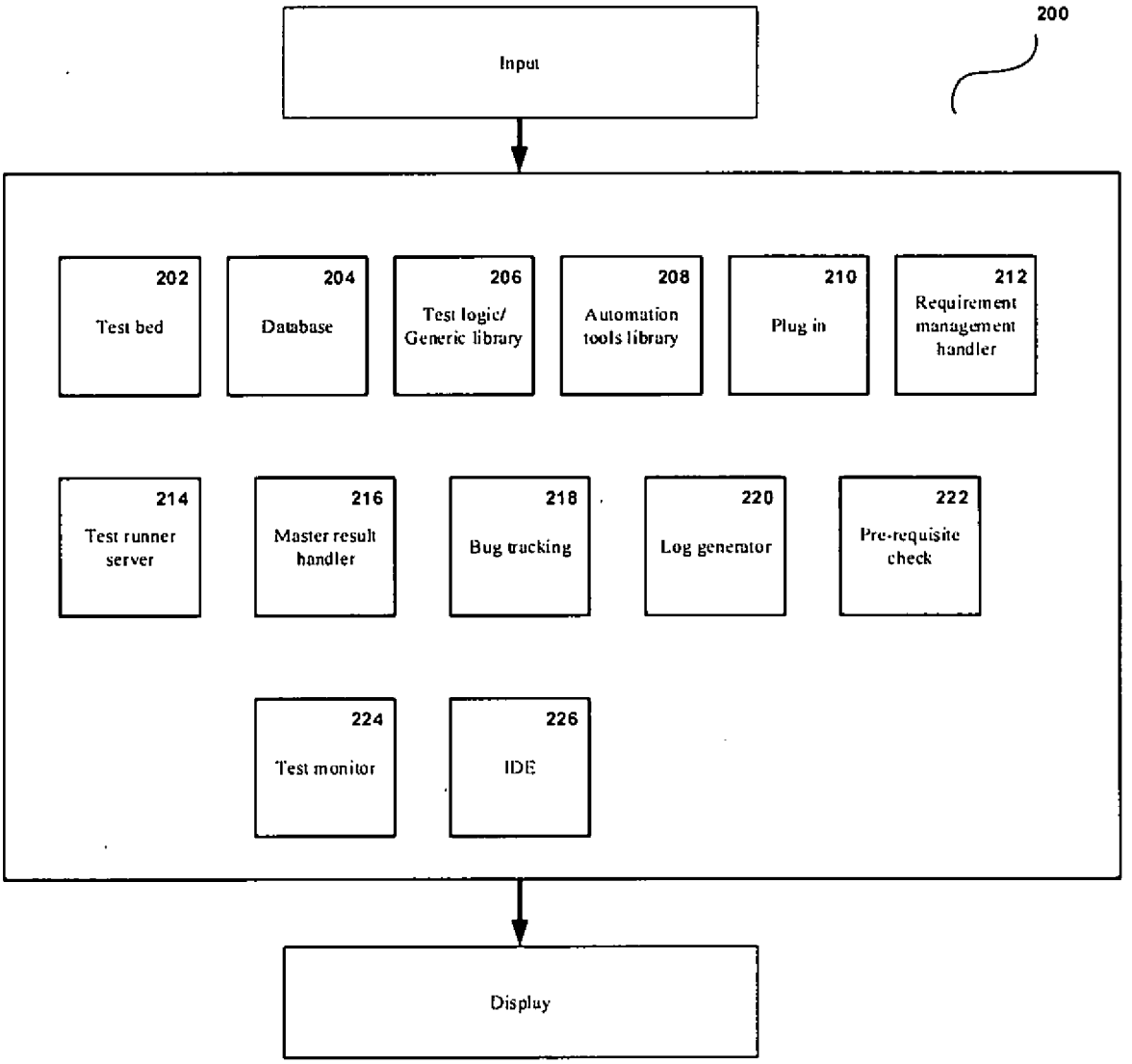


Figure 2

Mohammed Faisal (INPA No: 1941)
Head, IPR Dept.
L&T Technology Services Limited
DEF 3rd Block, 2nd Floor,
Manapakkam, Chennai – 600089

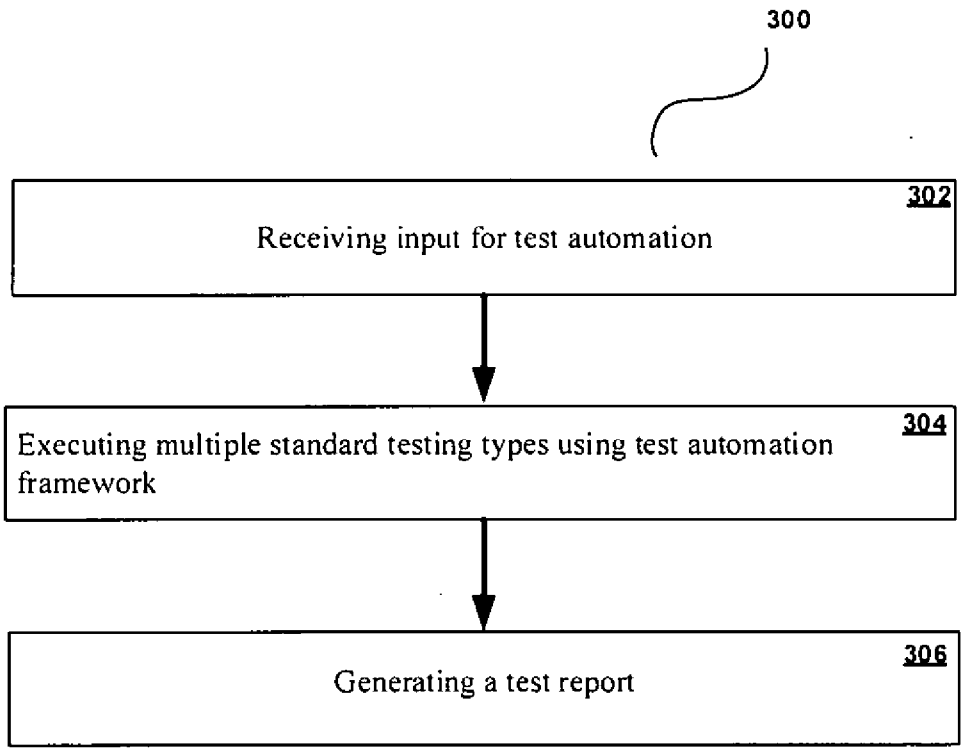


Figure 3

Mohammed Faisal (INPA No: 1941)
Head, IPR Dept.
L&T Technology Services Limited
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai – 600089