(12)Indian Patent Application

---

(21) Application Number: 711/CHE/2015

(22) Filing Date: 13/02/2015 (43) Publication Date: 26/08/2016

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

(72) Inventor(s): BALASUBRAMANYA BHARADWAJ
RAVICHANDRA B M
SHAILAJA D
MRN MURTHY

(51) International Classifications: H01M 8/00

(54) Title: A METHOD AND SYSTEM OF GENERATING TEST CASES FOR ASSEMBLY FILE

(57) Abstract: A Method and System of generating test cases for assembly file The invention relates to a method and system for generating test cases for assembly file. The method provides an application to read assembly file, selecting an executable file or library from the assembly file, listing classes and functions of the selected executable file or library in a tree structure, selecting a class or function of the executable file or library from the tree structure and generating test cases for the selected function or class of the executable file or library.

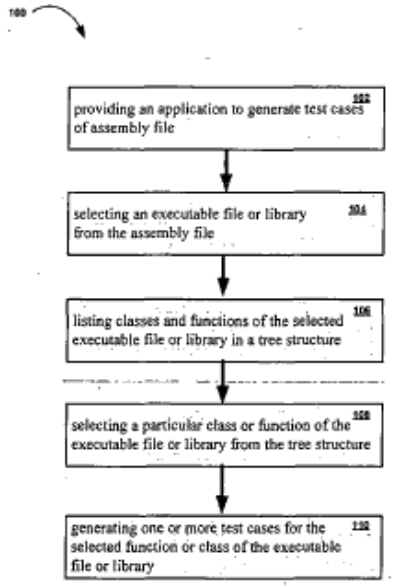

Figure 1

ABSTRACT

A Method and System of generating test cases for assembly file

The invention relates to a method and system for generating test cases for assembly file. The method provides an application to read assembly file, selecting an executable file or library from the assembly file, listing classes and functions of the selected executable file or library in a tree structure, selecting a class or function of the executable file or library from the tree structure and generating test cases for the selected function or class of the executable file or library.

We Claim:

1. A method for generating one or more test cases for one or more assembly files, the method comprising:

providing an application to read the one or more assembly files, each assembly file containing one or more executable files or libraries;

selecting an executable file or library from an assembly file of the one or more assembly files;

listing classes and functions of the selected executable file or library in a tree structure;

selecting a class or function of the executable file or library from the tree structure; and

generating one or more test cases for the selected function or class of the executable file or library.

2. The method of claim 1, wherein the generating one or more test cases is performed automatically.

3. The method of claim 1, further comprises using Microsoft .NET framework to develop the application.

4. The method of claim 1, wherein the generating one or more test cases for the selected function or class is based on function signature.

5. A system comprising:

A .NET framework-based platform to develop a test application; and

the test application to test a software project, wherein the software project comprises a plurality of assembly files.

6. The system of claim 5, wherein the testing one or more of the plurality of assembly files of the software project further comprising:

receiving user selection of an assembly file through a graphical user interface;

generating one or more test cases in response to the user selection; and

storing the generated test cases in a data store.

7. The system of claim 5, further comprising validating the assembly file before generating one or more test cases.

8. A system for generating one or more test cases for one or more assembly files, the system comprising:

a test application module to develop an application to read the one or more assembly files, each assembly file containing one or more executable files or libraries;

a test execution module to select an executable file or library from an assembly file of the one or more assembly files, list classes and functions of the selected executable file or library in a tree structure and select a class or function of the executable file or library from the tree structure; and

a processor to generate one or more test cases for the selected function or class of the executable file or library.

9. The system of claim 8, wherein the generating one or more test cases is performed automatically.

10. The system of claim 8, further comprises using Microsoft .NET framework to develop the application.

Dated this 13th day of February 2015

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

FIELD OF INVENTION

This invention relates to system and method for software testing and in particular to, unit testing of assembly files of an application.

BACKGROUND

Modern software typically involves many components often developed by large teams of software developers. The days of procedural programming in which a single developer could write an application that simply executed from start to finish performing a single, well-defined task are gone. The software developer often uses libraries, components, frameworks, and other bodies of code written by other developers. The chances for mistakes or misunderstanding how to use a particular external function or module are higher than ever.

One verification method for increasing the correctness of the software is called unit testing. When unit testing, the software developer attempts to isolate a particular unit of the software from external influences in a way that the unit can be tested in isolation. In theory, if each unit of the software application is correct, then the only room left for error is in communication between components, so the unit testing can dramatically reduce software error rates.

Test automation is the use of software to control the execution of tests, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control and test reporting functions. Test automation tools assist the software developer to quickly create applications that have dramatically improved the software developer's productivity, as well as decreasing the pressure on testers, who are often perceived as bottlenecks to the delivery of

software products. Test automation is one way to keep up with the increasing amounts of programming code that requires testing, as manual testing is time consuming.

While there is so much discussion on testing of software code, there is relatively little on generation of test cases of assembly file (.exe, .dll) for aiding in testing of the application. When a test report shows a bug in some module of the application where most of features are achieved by library referred, it is difficult to locate the bug as well as to find out what exactly created the bug.

Therefore, there may a need for an automated system and method for generating test cases for assembly files of an application.

SUMMARY OF THE INVENTION

Exemplary embodiments of the invention disclose a method and system for automatically generating one or more test cases for one or more assembly files of an application. According to an embodiment of the invention, a system and method for providing/developing an application to generate one or more test cases for the one or more assembly files is disclosed. Each assembly file contains one or more executable files or libraries. According to another embodiment, an executable file or library from one of the one or more assembly files is selected and a list of classes and functions of the selected executable file or library is displayed in a tree structure. A class or function of the executable file or library is selected from the tree structure. One or more test cases are generated by the test application for the selected function or class of the executable file or library.

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 block diagram of a process for automatic generation of test cases for assembly files of an application according to an exemplary embodiment of the invention;

Figure 2 illustrates an exemplary system for generating test cases for assembly files, according to one embodiment of the invention; and

Figure 3 illustrates a block diagram of a process for generating test cases for a .NET assembly file 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 as defined by the claims and their equivalents. 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.

According to embodiments of the invention, a system and method of generating test cases for assembly file of an application is disclosed.

FIG. 1 illustrates a block diagram of process 100 for automatic generation of test cases for an assembly file of an application according to an embodiment of the invention. At step 102, an application may be provided/ developed to generate one or more test cases for one or more assembly files. According to an embodiment, an assembly file may be an executable file or library. According to another embodiment, the application may be developed using a Microsoft .NET framework. According to further embodiment, the application may enable a test driven development process. According to yet another embodiment, the assemblies may be used as building blocks for the .NET Framework. Furthermore, the assemblies may provide the common language runtime with the information needed to be aware of type implementations. Stated differently, the assemblies may be a collection of types or resources, each of which may form a logical and distinct unit of functionality of a software project, that are built to work together.

According to one embodiment, a software project may be any software application or program designed for various purposes. An application may have one or more assemblies, and each of the assemblies having one or more functions. Each of the assemblies and functions may represent a distinct unit or functionality of the project. According to one embodiment, each of the assemblies may have an assembly manifest, which may be similar to a table of contents, comprising one or more of the following: (1) the assembly's identity, i.e., its name and version; (2) a file table describing files that make up the assembly, including, for example, any other assemblies that may have been created that the executable (.exe) or dynamic link library (.dll) files, or even bitmap or Readme files, may rely on; (3) an assembly reference list, which may

be a list of external dependencies, .dll or other files, that the application may need; and, (4) information with regard to content, versioning, and dependencies.

At step 104, an executable file or library may be selected from an assembly file of the one or more assembly files. According to one embodiment, the selection may be done through a graphical user interface (GUI). According to yet another embodiment, the selection may be automatic based on one or more predefined criteria. A software developer/tester may set the predefined criteria before starting the process of automatic generation of test cases.

According to one embodiment, a software developer/tester may be a programmer, a software engineer, a software specialist, a group of programmers or developer or engineers or specialists, a QA department or team within an organization or externally hired or contracted to develop software, to program, and to test to ensure that various software and/or hardware products and/or systems perform as originally specified and/or designed. A developer may have access to certain databases, sources, resources, components, and functions of a system or organization that may be restricted to non-developers.

At step 106, classes and functions of the selected executable file or library may be listed in a tree structure. According to an embodiment, the listing of the classes and functions of the executable file or library may be on a graphical user interface (GUI). The tree structure may be a collection of nodes, where each node is a data structure consisting of a value, together with a list of references to nodes, with the constraints that no reference is duplicated. According to yet another embodiment, the tree structure display format may be according to predefined criteria for displaying the classes and functions of the executable file or library.

At step 108, a class or function of the executable file or library may be selected from the tree structure. According to one embodiment, the selection of the class or function may be done through a graphical user interface (GUI). According to yet another embodiment, the selection of class or function may be automatic based on one or more predefined criteria defined by the software developer/tester.

At step 110, one or more test cases may be generated for the selected function or class of the executable file or library. According to one embodiment, the test cases may be generated based on function signature. According to another embodiment, the test cases may be generated for normal, abnormal, boundary and exception conditions. According to yet another embodiment, the test cases may be generated to test functionality, stability and performance of function based on inputs and function parameters. According to another embodiment, the software developer/tester may add or modify the unit test cases. According to another embodiment, the generated test cases may be stored in a unit test store.

The invention may also be extended to test case generation of .NET source code files.

FIG. 2 illustrates an exemplary system 200 for generation of test cases for assembly files, according to one embodiment of the present invention. The system 200 may include a test application module 210 to develop an application for testing assembly files. The test application module 210 may read one or more assembly files. Each assembly file may contain one or more executable files or libraries. According to one embodiment, the test application module may use Microsoft .NET framework to develop the application. According to another embodiment, the assembly files may be .NET assembly files. The system 200 may include a test execution module 212 to select an executable file or library from an assembly file of the

one or more assembly files, list classes and functions of the selected executable file or library in a tree structure and select a class or function of the executable file or library from the tree structure. The system 200 may include a processor 206 for generation of the test cases for the selected function or class of the executable file or library. According to an embodiment, the assembly files may be stored in a repository of executable files 202 or repository of library files 204. According to another embodiment, a .NET assembly file may be provided as input to the processor 206. The processor 206 may be a part of an application running on software operating platform such as, but not limited to, windows platform. According to an embodiment, the processor 206 may generate output in the form of unit test cases for the .NET assembly file on a display device 208. 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). According to yet another embodiment, the system 200 may store the generated unit test cases in a data store.

According to one embodiment, the assembly file may include one or more of executable files stored in 202 or library files stored in 204 that make up, for example, a Visual Studio application. When the Visual Studio source files are compiled, assemblies may be created automatically. Each Visual Studio application may have multiple assemblies ranging in number between ten to twenty. To use one assembly according to one embodiment, a reference may be added to the assembly. For example, the Visual Studio .NET provides a listing of all .NET Framework and other components available for referencing, and a listing of all reusable components created in local projects. Furthermore, the assembly may contain multiple namespaces. The namespaces may be used to organize the objects defined in the assembly to

prevent ambiguity and to simplify references when using large groups of objects, such as Microsoft .NET class libraries.

Figure 3 illustrates an exemplary method 300 for generating test cases according to embodiments of the invention. The exemplary method includes generating test cases for a .NET assembly file. As illustrated, at step 302, a .NET assembly file may be provided to the system 200. The system 200 may exemplarily validate whether file is proper .NET assembly or not. According to an embodiment, the validation may be using .NET reflections Assembly class.

At step 304, the system 200 on validating that the file is a proper assembly file, loads all the information including classes and functions from the .NET assembly file.

At step 306, the system 200 reads information about the classes and functions from the .NET assembly file. According to one embodiment, the system 200 may store the read information about the classes and functions in a list.

At step 308, the system 200 displays the class details in a tree structure. The displaying of the class details may enable a user to select required classes and functions to test. According to an embodiment, the listing of the classes and functions of the .NET assembly file may be on the display device 208. The display device 208 may include a graphical user interface (GUI). The tree structure may be a collection of nodes, where each node is a data structure consisting of a value, together with a list of references to nodes, with the constraints that no reference is duplicated. According to another embodiment, the tree structure display format may be

according to predefined criteria for displaying the classes and functions of the .NET assembly file.

At step 310, the system 200 receives selections of classes and functions to test. According to one embodiment, the selection may be through a graphical user interface (GUI). According to another embodiment, a user may select the classes and functions to test through the GUI. According to yet another embodiment, the selection may be automatic based on one or more predefined criteria.

At step 312, the system 200 generates unit test class and adds unit test functions to the unit test class. According to one embodiment, the unit test functions with predefined test cases may be added to the unit test class by using .NET CodeDOM class.

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 embodiments of the present invention may be provided as a computer program product that may include a machine-readable medium, having stored thereon instructions which may be used to program a computer (or other electronic devices) to perform a process according to the present invention. The

machine-readable medium may include, but is not limited to, floppy diskettes, optical disks, compact disc read-only memories (CD-ROMs), and magneto-optical disks, ROMs, random access memories (RAMs), erasable programmable read-only memories (EPROMs), electrically erasable programmable read-only memories (EEPROMs), magnetic or optical cards, flash memory, or other type of media/machine-readable medium suitable for storing electronic instructions. In addition to an embodiment consisting of specifically designed integrated circuits or other electronics, 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. The invention may also be implemented by the preparation of application specific integrated circuits or by interconnecting an appropriate network of conventional component circuits, as will be readily apparent to those skilled in the art.

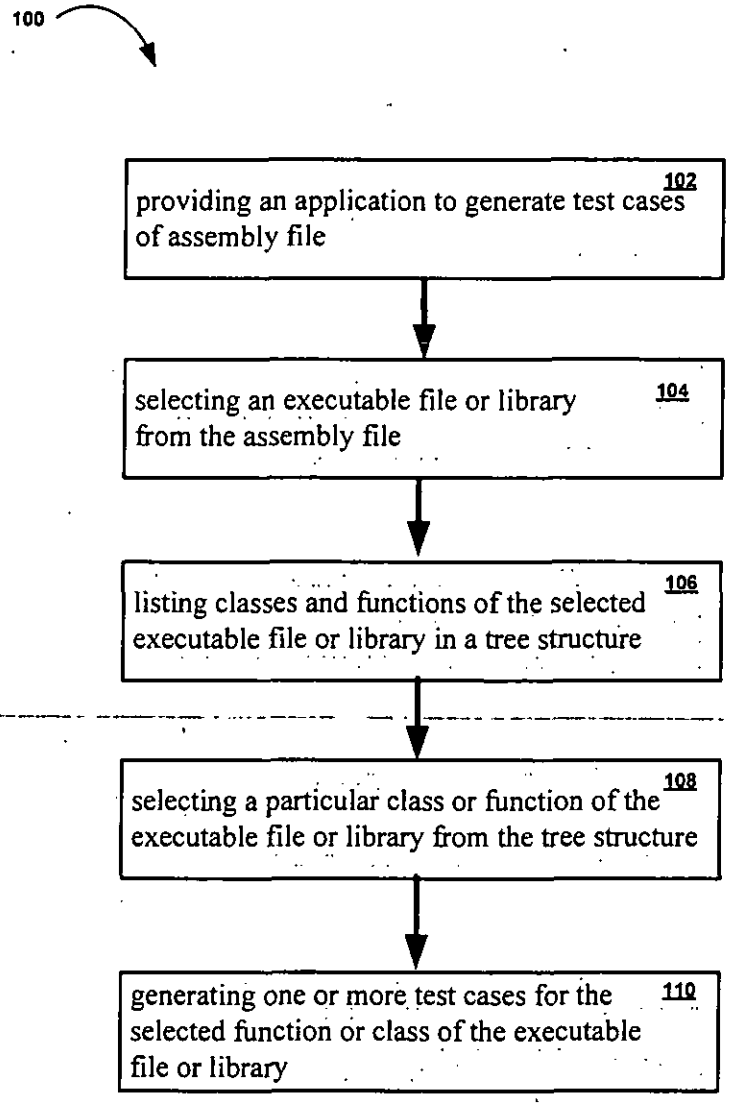

Figure 1

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

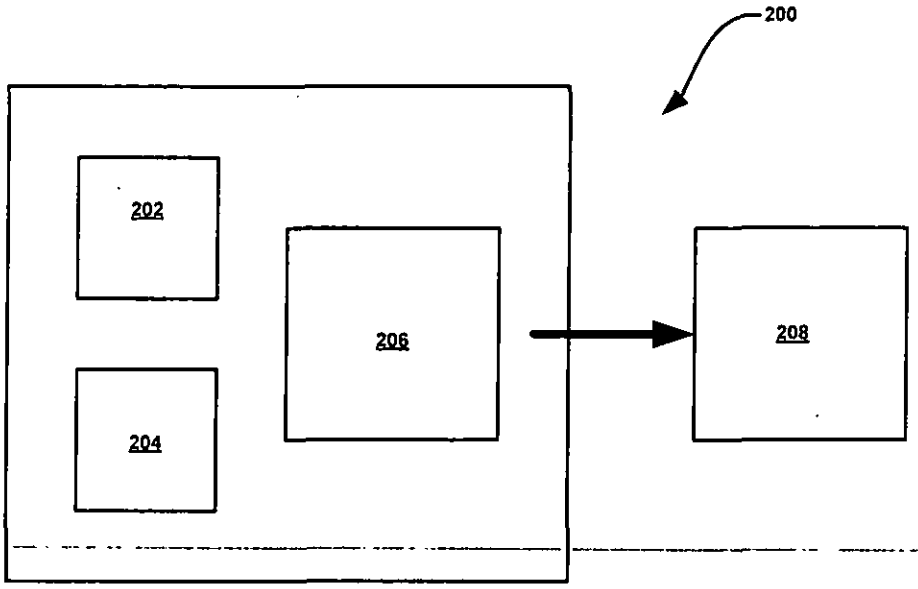

Figure 2

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

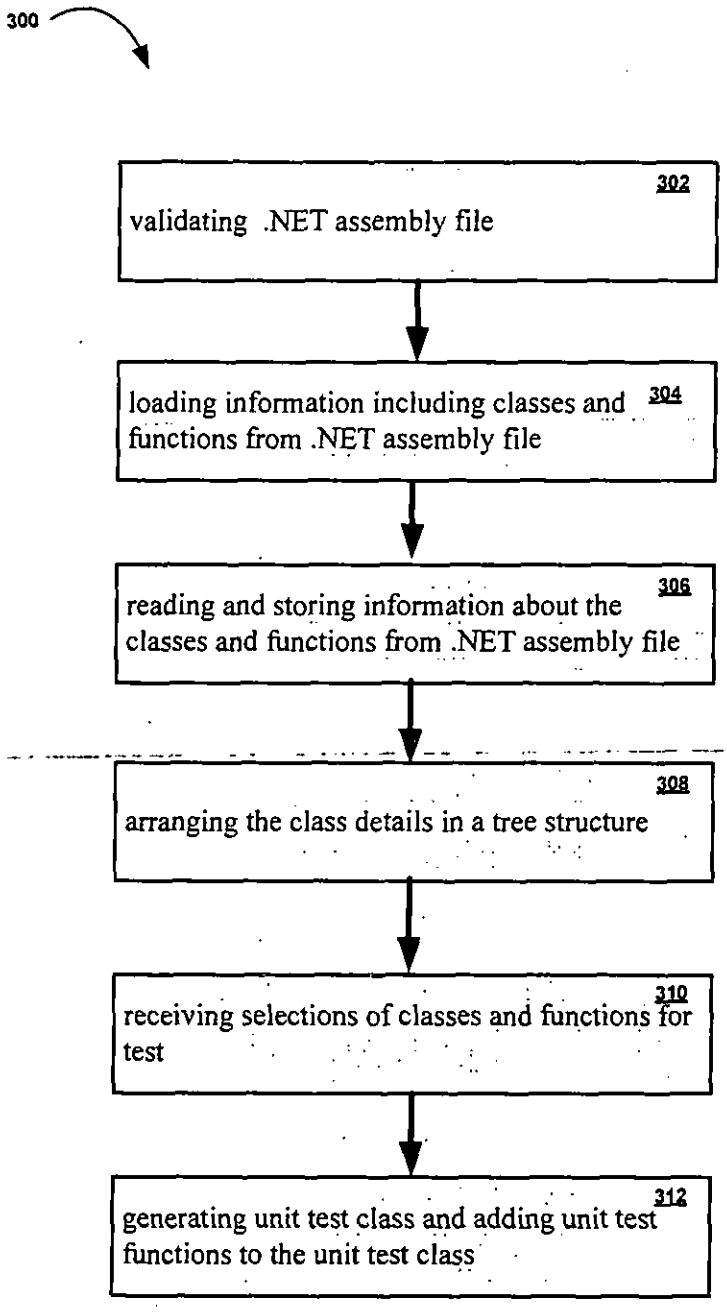

Figure 3

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