IPR ANALYTICS
Pioneer in IP Services

PATENTABILITY SEARCH REPORT

Title: A Debugging and Testing Tool for Supporting Software Evolution Submitted to:

Address:

Email:

Client Reference No:

Date: 10th JUL 2016

Features to Search

E1. The tool enhances the traditional debugging approach by automating the comparison of data structures between two running programs.

E2. The tool allows the reference code and the program being developed to execute on different computer systems by using open distributed systems techniques.

E3. A data visualization facility allows the user to view the differences in data structures and by using the data flow of the code it is possible to locate faulty sections of code rapidly.

Search Strategy

Database: AcclaimIP, USPTO, Patentscope, Espacenet, Google Patents.

Keywords:

Set 1	Automatic software testing, debugging, testing, evolutionary software
Set 2	Comparison, difference, variation
Set 3	Execute, apply
Set 4	Data structures, data variables
Set 5	Program, code, reference code,
Set 6	Faulty sections,

US CLASSIFICATION CODES WITH DEFINITIONS:

717/129 Using breakpoint
714/E11.208 Software debugging

INTERNATIONAL CLASSIFICATION CODES WITH DEFINITIONS:

G06F11/36 Preventing errors by testing or debugging of software

Search Results Reference 1:

Patent/Publication Number: US5838975 **Title:** Method for testing and debugging computer programs **Assignee/Applicant:** Abramson; David Andrew, Sosic; Rok **Filing Date:** 18 Apr 1995 **Priority Date:** 19 Apr 1994 **Also Published as:** None	
Relevant Excerpt for E1	**IN ABSTRACT:** A computerized method of testing and **debugging an executable behaviorally unknown computer program by dynamic comparison with an executable behaviorally known computer program.** The method controls execution of each program and compares related variable values at selected breakpoints in each program.

Relevant Excerpt for E2	**IN DESCRIPTION:** **Col. 3 lines 36-48** Referring to FIG. 1 there is illustrated a comparison method for testing and debugging a computer program in which a user
	selects **an executable behaviourally known computer program or reference program Rp to be executed on a computer Cr shown at step 1.** The user also selects an executable behaviourally unknown computer program which is the program to be tested Tp, shown at step 2, in which Tp is to be executed on a computer Ct. Program Tp has been compiled from the same source code as the reference program Rp. Computers Cr and Ct are linked to communicate with each other and the selection of Rp and Tp includes the identification of their respective file paths or file locations.
Relevant Excerpt for E3	**IN DESCRIPTION:** **Col. 3 line 66 to Col. 4 line 5** Values of variables of Rp and Tp are obtained (read) and then compared at step 5 by a comparison program on Cr. These variables may be selected by the user if so desired and by default all variables are selected**. If a difference results, the differences are output at step 6 to the user by displaying the differences upon a visual display unit,** outputting to a file or to any other output unit.

Reference 2:

Patent/Publication Number: US6002869

Title: System and method for automatically testing software programs

Assignee/Applicant: Novell, Inc.

Filing Date: 26 Feb 1997

Priority Date: 26 Feb 1997

Also Published as: None

Relevant Excerpt for E1	**IN CLAIMS:** 20. A computer program product comprising a computer readable medium having a computer program logic thereon for enabling a processor in a computer system to **automatically perform tests on a software program**, the computer system having a memory, accessible to said processor, in which is stored a test specification file containing state definitions associated with the software program, said software program having a plurality of states the product comprising: test engine means for testing discrete portions of said software program in accordance with the test specification file, said test engine means performing a plurality of test functions on the software program, each of said test functions being associated with and testing one of said discrete portions of said software program, wherein, when each of said test functions are successfully performed, said software program transitions from a current state to a next state, wherein said test engine means determines whether said software program transitioned correctly. **IN DESCRIPTION:** **Col. 11, line 62 to Col. 12 line 6** As shown, the test history file 600 includes, for each state transition, the time from the beginning of the test that the state transition occurred (Elapsed Time). It also includes the Next State, the Test Function, and the argument or value (Value), all of which are defined with respect to the test specification file

	500. For example, at time 0, the test procedure began and transitioned to the START state. At 8.073024 microseconds, the program 104 transitioned to the COMMON CONSOLE OPTIONS state at line 9 of the test specification file 500. **Comparison of the two files 500 and 600 clearly illustrate the manner in which the test engine 206 writes the test results to the test history file 600.**
Relevant Excerpt for E2	**IN DESCRIPTION:** **Col. 13, lines 12-32** A test function executor 810, responsive to the test case generator 808, **executes or applies the test functions 202 to the software program 104 through the conventional software controller 208**. The test case generator 808 instructs the test function executor 810 to **execute a specific test function 202 at a specific time to advance the software program** 104 from a current state to the next at a predetermined rate. The test function executor 810 reads the test functions 202 and arranges them in a function table 812. The test function executor 810, in response to instructions received from the test case generator 808, executes a specific test function 202 in the function table 812. Upon executing each of the test functions 202, the test function executor 810 informs the test case generator 808 whether or not the software program 104 successfully executed the state transition. The test function executor 810 uses the software controller 208 to exercise the software under test 104. The results of the implemented test function are returned from the software controller 208 to the test function executor 810,
	which in turn forwards the results to test case generator 808.

Relevant Excerpt for E3	Not Disclosed

****