Testing for Rule-Based Systems

Inventors: Mark Grechanik, Chicago, IL (US); Chen Fu, Lisle, IL (US)

Assignee: Accenture Global Services Limited, Dublin (IE)

Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 339 days.

Appl. No.: 13/250,348
Filed: Sep. 30, 2011

Prior Publication Data
US 2013/0085979 A1 Apr. 4, 2013

Int. Cl.
G06N 5/02 (2006.01)

U.S. Cl.
CPC ........................................... G06N 5/025 (2013.01)
USPC ............................................ 706/48

Field of Classification Search
CPC ............................................ G06N 5/025
USPC ............................................ 706/48
See application file for complete search history.

References Cited
U.S. PATENT DOCUMENTS

OTHER PUBLICATIONS
Ramaseswamy et al., Using directed hypergraphs to verify rule-based expert systems, IEEE Transactions on Knowledge and Data Engineering, vol. 9, No. 2, Mar.-Apr. 1997. *
Kiper, Structural testing of rule-based expert systems, ACM Transactions on Software Engineering and Methodology, vol. 1, No. 2 (1992). *
Kim et al., Test cases generation from UML state diagrams, IEEE Proc.-Softw., vol. 146, No. 4 (1999). *
Biermann et al., Graphical definition of in-place transformations in the eclipse modeling framework, Model Driven Engineering Languages and Systems, LNCS 4190, pp. 425-439 (2006). *

* cited by examiner

Primary Examiner — Kakali Chaki
Assistant Examiner — Michael Zidanic
(74) Attorney, Agent, or Firm — Finnegan, Henderson, Farabow, Garrett & Dunner, LLP

ABSTRACT
A non-transitory computer-readable medium tangibly embodied in a storage device encoded with instructions that, when executed on a processor, perform a method in a computer system for testing a rule-driven system, the method comprising detecting a potential read-write error or a potential write-write error in the rule-driven system, generating test results based on the detecting, and reporting the test results.

35 Claims, 40 Drawing Sheets

---

Diagram:

1. Analyze rules to determine variable use
2. Identify possible read-write and write-write conflicts
3. Identify path conditions that may cause conflicts
4. Represent rule dependencies
5. Perform test cases