BERT: BEnhavioral Regression Testing

Alessandro (Alex) Orso
School of CS -- College of Computing
Georgia Institute of Technology
http://www.cc.gatech.edu/~orso/

Tao Xie
Department of Computer Science
North Carolina State University
http://people.engr.ncsu.edu/txie/

Partially supported by: NSF, DHS, and US Air Force
2nd UPDATE: Amazon.com Web Site Down For Technical Reasons
June 06, 2008: 04:02 PM EST

(Updated to add information from a company customer-service representative.)

NEW YORK -(Dow Jones)- Amazon.com Inc.’s (AMZN) Web site was down for more than an hour Friday afternoon and remained dysfunctional at the time of this report.

An Amazon.com customer-service representative said the site wouldn’t be fully functional for another one or two hours. She said the outage was due to an upgrade of the company’s Web site, but didn’t provide further details.

The company’s Web site was completely down between at least 1:40 p.m. EDT to 3 p.m. before reappearing with partial functionality. Dow Jones Newswires employees were still unable to complete a full transaction on the site before getting an error message at the time of this report.
2nd UPDATE: Amazon.com Web Site Down For Technical Reasons
June 06, 2008: 04:02 PM EST

(Updated to add information from a company customer-service representative.)

NEW YORK -(Dow Jones)- Amazon.com Inc.’s (AMZN) Web site was down for more than an hour Friday afternoon and remained dysfunctional at the time of this report.

An Amazon.com customer-service representative said the site wouldn’t be fully functional for another one or two hours. She said the outage was due to an upgrade of the company’s Web site, but didn’t provide further details.

The company’s Web site was completely down between at least 1:40 p.m. EDT to 3 p.m. before reappearing with partial functionality. Dow Jones Newswires employees were still unable to complete a full transaction on the site before getting an error message at the time of this report.
[...] the outage was due to an upgrade of the company’s Web site [...]
“If only the kernel had a regression testsuite, everything would be better.”

-- Greg Kroah-Hartman keynote on the Linux kernel at OLS 2006
Regression Testing
Process and Issues
Regression Testing
Process and Issues

Test suite T

Program P → Program P'
Regression Testing
Process and Issues

Test suite $T$

Program $P$

Program $P'$
Regression Testing
Process and Issues
Regression Testing
Process and Issues

Test-suite maintenance

Test suite T

Obsolete test cases

Test suite Tval
Regression Testing

Process and Issues

Test-suite maintenance

Test suite T

Test suite Tval

Regression test selection

Test suite T'

Obsolete test cases
Regression Testing
Process and Issues

Test-suite maintenance

Test suite T

Regression test selection

Test suite T'

Test-suite prioritization

Prioritized Test suite T'

Obsolete test cases
Regression Testing
Process and Issues

Test suite $T$

Test-suite maintenance
Test suite $T_{val}$
Regression test selection
Test suite $T'$
Test-suite prioritization
Prioritized Test suite $T'$

Test suite $T_{aug}$

Obsolete test cases
Regression Testing Process and Issues

- Test-suite maintenance
- Regression test selection
- Test-suite prioritization
- Prioritized Test suite T'
- Test-suite augmentation
- Minimized test suite
- Test-suite minimization
- Redundant test cases
- Obsolete test cases
Regression Testing Process and Issues

Test suite T

Test-suite maintenance

Test suite Tval

Regression test selection

Test suite T'

Test-suite prioritization

Prioritized Test suite T'

Modified test suite

Test-case manipulation

Minimized test suite

Test-suite minimization

Test suite Taug

Test suite T

Obsolete test cases

Redundant test cases

Test-suite augmentation
Regression Testing Process and Issues

- Test-suite maintenance
- Regression test selection
- Test-suite prioritization
- Test-case manipulation
- Minimized test suite
- Redundant test cases
- Augmented test suite
- Prioritized test suite T'
- Modified test suite
Outline

• Introduction
• Our technique
• Experience
• Conclusion and future work
Outline

• Introduction
• Our technique
• Experience
• Conclusion and future work
Outline

- Introduction
- Our technique
- Experience
- Conclusion and future work
Traditional regression testing
Traditional regression testing
Traditional regression testing

Program P

Program P'

Test runner & Oracle checker

Test suite T

Regression errors

Traditional regression testing
class BankAccount {
    double balance;

    bool deposit(double amount) {
        if (amount > 0.00) {
            balance = balance + amount;
            return true;
        } else {
            print("negative amount");
            return false;
        }
    }

    bool withdraw(double amount) {
        if (amount <= 0) {
            print("negative amount");
            return false;
        } else {
            if (balance < 0)
                print("account overdraft");
            return false;
        }
        balance = balance - amount;

        return true;
    }
}
class BankAccount {
    double balance;

    bool deposit(double amount) {
        if (amount > 0.00) {
            balance = balance + amount;
            return true;
        } else {
            print("negative amount");
            return false;
        }
    }

    bool withdraw(double amount) {
        if (amount <= 0) {
            print("negative amount");
            return false;
        }
        if (balance < 0)
            print("account overdraft");
            return false;
    }
    balance = balance - amount;

    return true;
}
class BankAccount {

double balance;

bool deposit(double amount) {
    if (amount > 0.00) {
        balance = balance + amount;
        return true;
    } else {
        print("negative amount");
        return false;
    }
}

bool withdraw(double amount) {
    if (amount <= 0) {
        print("negative amount");
        return false;
    }
    if (balance < 0)
        print("account overdraft");
    balance = balance - amount;
    return true;
}
class BankAccount {

double balance;

bool deposit(double amount) {
    if (amount > 0.00) {
        balance = balance + amount;
        return true;
    } else {
        print("negative amount");
        return false;
    }
}

bool withdraw(double amount) {
    if (amount <= 0) {
        print("negative amount");
        return false;
    } else { // Removed the if statement for withdrawing
        if (balance < 0) {
            print("account overdraft");
            return false;
        }
        balance = balance - amount;
        return true;
    }
}
}

class BankAccount {

double balance;
bool isOverdraft;

bool deposit(double amount) {
    if (amount > 0.00) {
        balance = balance + amount;
        return true;
    } else {
        print("negative amount");
        return false;
    }
}

bool withdraw(double amount) {
    if (amount <= 0) {
        print("negative amount");
        return false;
    }
    if (balance < 0) {
        print("account overdraft");
        return false;
    }
    balance = balance - amount;
    if (balance < 0)
        isOverdraft = true;
    return true;
}
}
class BankAccount {
    double balance;

    bool deposit(double amount) {
        if (amount > 0.00) {
            balance = balance + amount;
            return true;
        } else {
            print("negative amount");
            return false;
        }
    }

    bool withdraw(double amount) {
        if (amount <= 0) {
            print("negative amount");
            return false;
        }
        if (balance < 0) {
            print("account overdraft");
            return false;
        }
        balance = balance - amount;
        return true;
    }
}
class BankAccount {
    double balance;

    bool deposit(double amount) {
        if (amount > 0.00) {
            balance = balance + amount;
            return true;
        } else {
            print("negative amount");
            return false;
        }
    }

    bool withdraw(double amount) {
        if (amount <= 0) {
            print("negative amount");
            return false;
        } else {
            print("account overdraft");
            isOverdraft = true;
            return true;
        }
    }
}

class BankAccount {
    double balance;
    bool isOverdraft;

    bool deposit(double amount) {
        if (amount > 0.00) {
            balance = balance + amount;
            return true;
        } else {
            print("negative amount");
            return false;
        }
    }

    bool withdraw(double amount) {
        if (amount <= 0) {
            print("negative amount");
            return false;
        }
        if (isOverdraft) {
            print("account overdraft");
            return false;
        }
        balance = balance - amount;
        return true;
    }
}

class BankAccount {

double balance;

bool deposit(double amount) {
    if (amount > 0.00) {
        balance = balance + amount;
        return true;
    } else {
        print("negative amount");
        return false;
    }
}

bool withdraw(double amount) {
    if (amount <= 0) {
        print("negative amount");
        return false;
    }
    if (balance < 0) {
        print("account overdraft");
        return false;
    }
    balance = balance - amount;
    return true;
}
class BankAccount {
    double balance;
    bool isOverdraft;

    bool deposit(double amount) {
        if (amount > 0.00) {
            balance = balance + amount;
            return true;
        } else {
            print("negative amount");
            return false;
        }
    }

    bool withdraw(double amount) {
        if (amount <= 0) {
            print("negative amount");
            return false;
        }
        if (isOverdraft) {
            print("account overdraft");
            return false;
        }
        balance = balance - amount;
        if (balance < 0)
            isOverdraft = true;
        return true;
    }
}

Where is the fault?
class BankAccount {
    double balance;
    bool isOverdraft;

    bool deposit(double amount) {
        if (amount > 0.00) {
            balance = balance + amount;
            return true;
        } else {
            print("negative amount");
            return false;
        }
    }

    bool withdraw(double amount) {
        if (amount <= 0) {
            print("negative amount");
            return false;
        }
        if (isOverdraft) {
            print("account overdraft");
            return false;
        }
        balance = balance - amount;
        if (balance < 0)
            isOverdraft = true;
        return true;
    }
}
```java
class BankAccount {
  double balance;
  bool isOverdraft;

  bool deposit(double amount) {
    if (amount > 0.00) {
      balance = balance + amount;
      return true;
    } else {
      print("negative amount");
      return false;
    }
  }

  bool withdraw(double amount) {
    if (amount <= 0) {
      print("negative amount");
      return false;
    }
    if (isOverdraft) {
      print("account overdraft");
      return false;
    }
    balance = balance - amount;
    if (balance < 0)
      isOverdraft = true;
    return true;
  }
}

void testBehavioralDifference() {
  BankAccount a = new BankAccount();
  a.deposit(10.00);
  a.withdraw(20.00);
  a.deposit(50.00);
  bool result = a.withdraw(20.00);
  assertEquals(result, true);
}
```

Such a test may not be in T

• 100% stmt coverage without it

• Specific sequence of calls/params

• Or its oracle may be inadequate
Traditional regression testing

Existing test suite typically targets a small subset of the program behavior

- Tests focus on core functionality
- Oracles often approximated
Traditional regression testing

Program P

Test runner & Oracle checker

Test suite T

Regression errors

Program P′

Program P

Program P′

Test suite T

BERT
Phase I: Generation of test cases for changed code

Program P

Program P'

Test suite T

BERT
**Phase I:**
Generation of test cases for changed code

Program P

Program P'

Test suite T

Change analyzer
Phase I: Generation of test cases for changed code

Program P

Program P'

Test suite T
Phase I: Generation of test cases for changed code
Phase I: Generation of test cases for changed code

Program P

Program P’

Tests for C TC

Test suite T

Change analyzer

Code changes C

Test case generator

BERT
**Phase I:**
Generation of test cases for changed code

**Change analyzer**

- Given two versions, produces a list of changed classes
- Can use any differencing tool
- Currently: Eclipse’s change information
Phase I: Generation of test cases for changed code

- Change analyzer
- Code changes C
- Test case generator

Program P

Program P'

Tests for C TC

Test suite T
**Phase I:**
Generation of test cases for changed code

**Test case generator**
- Given a class, generates a set of test cases for the class
- BERT can use one or more generators
- Currently: JUnit Factory and Randoop
Phase I: Generation of test cases for changed code

Test case generator

- Given a class, generates a set of test cases for the class
- BERT can use one or more generators
- Currently: JUnit Factory and Randoop
Phase II: Behavioral comparison
Phase II: Behavioral comparison

- Program P
- Program P'
- Test suite T
- Code changes C
- Test runner & Behavioral comparator
- Test case generator
- Change analyzer
- BERT

Tests for C TC

Phase II: Behavioral comparison
**Phase II: Behavioral comparison**

- **Change analyzer**
- **Program P**
- **Program P'**
- **Tests for C TC**
- **Test runner & Behavioral comparator**
- **Test suite T**

- **BERT**
  - **Code changes C**
  - **Test case generator**

- **Raw behavioral differences**
Test runner & Behavioral comparator

- $\forall c$ and $t$ for $c$, runs $t$ on old and new versions of $c$, logging state, return values, outputs
- Compares and logs differences and relevant context

Phase II: Behavioral comparison
BERT

Change analyzer

Test case generator

Code changes C

Program P

Program P'

Tests for C TC

Test runner & Behavioral comparator

Test suite T

Raw behavioral differences

Change analyzer

Program P

Program P'

Tests for C TC

Test runner & Behavioral comparator

Test suite T

Raw behavioral differences
Phase III:
Differential behavior analysis and reporting
Phase III: Differential behavior analysis and reporting
Phase III: Differential behavior analysis and reporting
Behavioral differences analyzer

- Simplifies and refines raw data through abstraction and redundancy elimination

- Reports behavioral differences between \( c_{v0} \) and \( c_{v1} \) and test cases that reveal them

  - fields with \( \neq \) values
  - methods returning \( \neq \) values
  - differences in graphical and textual output

**Phase III:** Differential behavior analysis and reporting
**Phase I:** Generation of test cases for changed code

- Change analyzer
- Test case generator
- Code changes C
- Program P
- Program P'
- Tests for C TC

**Phase II:** Behavioral comparison

- Test runner & Behavioral comparator
- Test suite T
- Raw behavioral differences

**Phase III:** Differential behavior analysis and reporting

- Behavioral differences analyzer
- Behavioral differences

**BERT**
Code changes C

Raw behavioral differences

Program P

Program P'

Test suite T

Test runner & Behavioral comparator

Tests for C TC

Raw behavioral differences

Behavioral differences analyzer

Behavioral differences

Change analyzer

Test case generator
Change-centric automatic generation of test cases

Program P

Program P'

Tests for C TC

Change analyzer

Test case generator

Code changes C

Test runner & Behavioral comparator

Test suite T

Raw behavioral differences

Behavioral differences analyzer

Behavioral differences

BERT
Change-centric automatic generation of test cases

Focus on differential behavior
Outline

- Introduction
- Our technique
- Experience
- Conclusion and future work
Outline

- Introduction
- Our technique
- Experience
- Conclusion and future work
Proof of Concept Evaluation

- Built BERT prototype
- Phase I & II
- Reflection and scaffolding instrumentation
- Applied BERT to BankAccount example
- Fed BankAccount to BERT
- Generated 2,569 test inputs
  (< 1 sec to execute)
Results

- 60% of the inputs (1,557) showed a behavioral difference that revealed the regression error
  - `withdraw` returned different values
  - `withdraw` produced different output
Results

- 60% of the inputs (1,557) showed a behavioral difference that revealed the regression error
- `withdraw` returned different values
- `withdraw` produced different output

```java
public void testclasses3() throws Throwable {
    BankAccount var0 = new BankAccount();
    double var1 = (double)1.0;
    boolean var2 = var0.deposit((double)var1);
    double var3 = (double)2.0;
    boolean var4 = var0.withdraw((double)var3);
    double var5 = (double)1.0;
    boolean var6 = var0.deposit((double)var5);
    double var7 = (double)2.0;
    boolean var8 = var0.withdraw((double)var7);
}
```
Results

• 60% of the inputs (1,557) showed a behavioral difference that revealed the regression error
  • `withdraw` returned different values
  • `withdraw` produced different output

• Some considerations
  • No state difference reported despite addition of field `isOverdraft` (intentional)
  • Results obtained in a fully-automated way thanks to BERT’s characteristics
  • Generation of large number of tests for changed code
  • Automatic identification of behavioral differences
Outline

• Introduction
• Our technique
• Experience
• Conclusion and future work
Outline

• Introduction
• Our technique
• Experience
• Conclusion and future work
Related Work

• Rich literature on related areas
• Test suite augmentation
• Impact analysis
• Differential testing
• Regression testing in general
• Too many to mention individually
• Thorough discussion in the paper
Conclusion

- **BERT**: a regression testing approach that
  - Generates many tests for changed code
  - Runs test on old and new (changed) code
  - Analyzes and reports differences in behavior
- Two key novelties
  - Focus on a small code fraction ⇒ thorough
  - Leverage differential behavior ⇒ no oracles
Open Issues

• Main issue: false positives
• More studies are needed to assess the issue
• Some ideas to address it
  • More aggressive abstraction, clustering, and filtering based on change information
  • Ranking based on change information
  • Combination with automated debugging
Thank You!