Preventing Data Errors with Continuous Testing

Kıvanç Muşlu  Yuriy Brun  Alexandra Meliou

University of Washington  University of Massachusetts
Data entry error raises flag on life insurance

McShane's insurance was set to run out in December.

At 26 years old he had no kids and no car, he

McShane's 3 year old son's room was one door
down — a room McShane and his wife had
turned into a home office.

She tried to improve it by filling with

McShane's insurance policy is not to count
on it.

February 9, 2013

Recently, we informed you of a data error that inflated
all school costs by a cumulative total of $7.4 million. We also
want to clarify what happened, the impact it had, and our
response to this situation.

Prior to Winter Break, site administrators reviewed a one-page
total allocations for each site. The summary indicated
that allocations, was flat, essentially the same as the current
allocations. The data was used to plan expenditures. To
present the full picture of expenditures, insured the site
level.

In this instance, our data error, allocated a mistake and made the site budget

Vernon Hall, Deputy Superintendent, Business & Operations

Data entry is a top cause of medication errors

Training and design are seen as keys to reducing electronic prescribing errors.

By ANDIS ROBEZNIKS — Posted Jan. 24, 2005

Computerized prescribing systems might cut
the quantity and severity of medication
mistakes, but they can’t eliminate them
entirely, said patient safety experts who
reviewed the U.S. Pharmacopeia’s 5th
annual study of medication error reports.

The study of the more than 235,000 error
reports submitted in 2003 by 570 health care
facilities was the largest ever by USP. And
as the number of reported errors goes up, the percentage that causes patient
harm has gone down. But the findings that generated the most discussion are
those indicating that electronic prescribing is creating new types of errors.

"Computer entry" was the fourth-leading cause of errors, accounting for
13% (27,711) of the medication errors reported in 2003. In contrast, illegible
or unclear handwriting was the 15th-leading cause, and accounted for 2.9%
(6,134) of reported errors.
Data discrepancies

American Airlines Flight Number 119 (AA119)

FLIGHT TRACKER

6:15 PM

Origin Terminal A / Gate 32 / Newark Liberty Intl (KEWR - track or info)

Destination Terminal 4 / Gate 42B / Los Angeles Intl (KLAX - track or info)

Aircraft Boeing 737-800 (twin-jet) (B738/Q - track or photos)

Departure 6:15 PM, Dec 08
Takeoff Time 6:53 PM, Dec 08
Terminal - Gate: Terminal A - 32

Arrival Status: In Air

Airport
Scheduled Time: 9:40 PM, Dec 08
Takeoff Time: 9:42 PM, Dec 08

Estimated Time: 8:33 PM

Time Remaining: 25 min
Terminal - Gate: Terminal 4 - 42
Baggage Claim: 4

Duration 5 hours 43 minutes
20 minutes left
5 hours 23 minutes

Arrival 9:40 PM

9:40 PM

滑翔

slide credit: Luna Dong and Divesh Srivastava
“Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?”

Charles Babbage, from *Passages from the Life of a Philosopher*
Dealing with software errors

• program analysis
• language features
• code reviews
• formal verification
  ... and
     testing
Key idea: Testing for data-intensive systems

- Identifying system failures caused by well-formed but incorrect data
- Using application-specific execution information
- Integrating into the system usage workflow

Bringing testing to the data domain
How is data testing different?

System administrators and users are not software engineers

• Data semantics
• Test query generation
• Timeliness
• Unobtrusive and precise interface
Data semantics

• Valid data can span multiple orders of magnitude
  – thwarts statistical outlier detection
• Semantics of nearly identical data differ vastly:
Test query generation

- Administrators and users have not (yet) bought into testing
- Manually written tests will come
  - developers can ship these with application
- But automatic generation is needed now
  - mine queries from code
  - record historical queries
  - adapt related work on database-use test generation
Timeliness

• Data testing is a runtime activity

• Administrators and users don’t troubleshoot unless something goes wrong

• Learning about an error sooner means error has less impact
Unobtrusive and precise user interface

• Administrators and users may not understand test outcomes
• Tests must integrate into workflow
• Results must link to
  – actions that caused failure, or
  – data values relevant to failure
Example scenario: car dealership

<table>
<thead>
<tr>
<th>carID</th>
<th>make</th>
<th>model</th>
<th>year</th>
<th>inventory</th>
<th>cost</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>121</td>
<td>Nissan</td>
<td>Versa</td>
<td>2014</td>
<td>23</td>
<td>$10,990</td>
<td>$13,199</td>
</tr>
<tr>
<td>96</td>
<td>Smart</td>
<td>fortwo Pure</td>
<td>2014</td>
<td>21</td>
<td>$12,490</td>
<td>$14,999</td>
</tr>
<tr>
<td>227</td>
<td>Ford</td>
<td>Fiesta</td>
<td>2014</td>
<td>9</td>
<td>$13,200</td>
<td>$15,799</td>
</tr>
<tr>
<td>160</td>
<td>Suzuki</td>
<td>SX4</td>
<td>2014</td>
<td>27</td>
<td>$13,699</td>
<td>$16,499</td>
</tr>
<tr>
<td>82</td>
<td>Chevrolet</td>
<td>Sonic</td>
<td>2014</td>
<td>15</td>
<td>$13,735</td>
<td>$16,499</td>
</tr>
<tr>
<td>311</td>
<td>KIA</td>
<td>Soul</td>
<td>2013</td>
<td>3</td>
<td>$13,300</td>
<td>$14,699</td>
</tr>
<tr>
<td>319</td>
<td>KIA</td>
<td>Soul</td>
<td>2014</td>
<td>22</td>
<td>$13,900</td>
<td>$16,999</td>
</tr>
<tr>
<td>286</td>
<td>Toyota</td>
<td>Yaris</td>
<td>2013</td>
<td>1</td>
<td>$13,980</td>
<td>$15,199</td>
</tr>
<tr>
<td>295</td>
<td>Toyota</td>
<td>Yaris</td>
<td>2014</td>
<td>11</td>
<td>$14,115</td>
<td>$16,999</td>
</tr>
<tr>
<td>511</td>
<td>Mercedes</td>
<td>C-Class</td>
<td>2014</td>
<td>21</td>
<td>$35,800</td>
<td>$45,999</td>
</tr>
<tr>
<td>513</td>
<td>Mercedes</td>
<td>R-Class</td>
<td>2014</td>
<td>7</td>
<td>$52,690</td>
<td>$62,899</td>
</tr>
<tr>
<td>799</td>
<td>Maserati</td>
<td>Quattroporte</td>
<td>2014</td>
<td>8</td>
<td>$102,500</td>
<td>$122,999</td>
</tr>
<tr>
<td>808</td>
<td>Maserati</td>
<td>GranTurismo</td>
<td>2014</td>
<td>12</td>
<td>$126,500</td>
<td>$149,999</td>
</tr>
</tbody>
</table>

- Manager wants to put cars between $10K and $15K on a 30%-off sale

```
UPDATE cars SET price=0.7*price
WHERE price BETWEEN 10000 AND 150000
```
Car dealership challenges

- Data semantics
  reducing all prices by 30% seems like a valid change
- Test query generation
  manager doesn’t know about writing tests
- Timeliness
  manager doesn’t know to run tests
  reporting delay causes financial losses
- Unobtrusive and precise interface
  problem cannot be reported as “test failed”
Continuous Data Testing (CDT)

• Data semantics and test query generation mines source code for queries allows manually written and history-mined tests

• Timeliness run tests continuously optimizations to trigger proper tests at proper times

• Unobtrusive and precise interface delegates problem to system designer highlights data involved in tests
Making CDT possible (optimizations)

- NaïveCDT: run tests continuously
- SimpleCDT: only run tests after updates
- SmartCDT: only relevant tests after relevant changes
- SmartCDT$_{TC}$: test compression
- SmartCDT$_{IT}$: incremental test query execution
- SmartCDT$_{TC+IT}$: all of the above
How effective is CDT at preventing data entry errors?

Do false positives reduce CDT’s effectiveness?

CDT’s and integrity constraints’ effect on data entry speed
Crowd study
Amazon Mechanical Turk

Data came from real-world spreadsheets from data.gov and the EUSES corpus; tests from formulae in the spreadsheets.

96 distinct users were asked to copy numeric values into the corresponding cells.

---

Please copy the exact data you see in the above image into the cells below. If a cell is highlighted by default, it is possible that you made an error. When you finish, check the "I agree to voluntarily enter this study" box and then click the "Submit" button. Make sure that you fill all cells.

Thanks,

Informed consent: ConsentForm.pdf

☐ I agree to voluntarily enter this study.

Submit

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.536338109</td>
<td>5.537153072</td>
<td>5.53693926</td>
<td>5.536773001</td>
<td>5.536624876</td>
<td>5.536385728</td>
<td>5.536385728</td>
<td>5.536385728</td>
<td>5.536385728</td>
<td>5.536385728</td>
<td>5.536385728</td>
<td>5.536385728</td>
</tr>
</tbody>
</table>

---

Group 1: control
No highlighting. Submitting errors allowed.

Group 2: CDT
Data involved in failing tests **highlighted**. Submitting errors OK.

Group 3: CDT with false positives
Data involved in failing test and 40% extra data **highlighted**. Submitting errors OK.

Group 4: integrity constraints
Data involved in failing integrity constraints **highlighted**. No submitting errors.
CDT, even with false positives, successfully prevented data errors.

CDT much faster than integrity constraints.
CDT efficiency

CDT’s effect on performance of performance-intensive applications

- NaïveCDT
- SimpleCDT
- SmartCDT
- SmartCDT_{TC}
- SmartCDT_{IT}
- SmartCDT_{TC+IT}

More frequent updates = more overhead

Even with frequent updates, overhead is manageable
Related work

• Generating tests for systems that use databases


  Li and Csallner. Dynamic symbolic database application testing. DBTest 2010.

  Pan, Wu, and Xie. Database state generation via dynamic symbolic execution for coverage criteria. DBTest 2011.

  Pan, Wu, and Xie. Generating program inputs for database application testing. ASE 2011.
Related work

• Generating tests for systems that use databases

• Continuous testing


Related work

- Generating tests for systems that use databases
- Continuous testing
- Effects of continuous feedback


Related work

• Generating tests for systems that use databases
• Continuous testing
• Effects of continuous feedback
• Errors in spreadsheets

Badame and Dig. Refactoring meets spreadsheet formulas. ICSM 2012.

Barowy, Gochev, and Berger. CheckCell: Data debugging for spreadsheets. OOPSLA 2014.

Hermans and Dig. BumbleBee: A refactoring environment for spreadsheet formulas. FSE Tool Demo 2014.
Related work

• Generating tests for systems that use databases
• Continuous testing
• Effects of continuous testing
• Errors in spreadsheet
• Data cleaning


Related work

- Generating tests for systems with databases
- Continuous testing
- Effects of continuous feedback
- Errors in spreadsheets
- Data cleaning
- Understanding errors in databases

Meliou, Gatterbauer, Moore, and Suciu. The complexity of causality and responsibility for query answers and non-answers. PVLDB 2010.


Related work

• Generating tests for systems that use databases
• Continuous testing
• Effects of continuous feedback
• Errors in spreadsheets
• Data cleaning
• Understanding errors in databases
• A truckload of automated test generation work
Contributions

• Four challenges of data testing:
  1. data semantics
  2. test generation
  3. timeliness
  4. interface

• Continuous Data Testing prototype for PostgreSQL

• Optimizations for which tests to run when

• CDT efficient and effective at preventing errors, even when low-quality tests result in false positives

https://bitbucket.org/ameli/contest