A Family of Test Adequacy Criteria for Database-Driven Applications

Gregory M. Kapfhammer
Department of Computer Science
University of Pittsburgh

Mary Lou Soffa
Department of Computer Science
University of Pittsburgh

Allegheny College
Motivation

The Risks Digest, Volume 22, Issue 64, 2003

**Jeppesen reports airspace boundary problems**

About 350 airspace boundaries contained in Jeppesen NavData are incorrect, the FAA has warned. The error occurred at Jeppesen after a software upgrade when information was pulled from a database containing 20,000 airspace boundaries worldwide for the March NavData update, which takes effect March 20.
Testing Challenges

- Should consider the environment in which software applications execute
- Must test a program and its interaction with a database
- Database-driven application’s state space is well-structured, but infinite (Chays et al.)
- Need to show program does not violate database integrity, where \( \text{integrity} = \text{consistency} + \text{validity} \) (Motro)
- Must locate program and database coupling points that vary in granularity
→ A program can interact with a database at different levels of granularity
Database-Driven Applications

A program can interact with a database at different levels of granularity.
Database-Driven Applications

A program can interact with a database at different levels of granularity.

<table>
<thead>
<tr>
<th>card_number</th>
<th>pin_number</th>
<th>user_name</th>
<th>acct_lock</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32142</td>
<td>Brian Zorman</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>41601</td>
<td>Robert Roos</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>45322</td>
<td>Marcus Bittman</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>56471</td>
<td>Geoffrey Arnold</td>
<td>0</td>
</tr>
</tbody>
</table>
Database-Driven Applications

A program can interact with a database at different levels of granularity.

```
<table>
<thead>
<tr>
<th>card_number</th>
<th>pin_number</th>
<th>user_name</th>
<th>acct_lock</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32142</td>
<td>Brian Zorman</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>41601</td>
<td>Robert Roos</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>45322</td>
<td>Marcus Bittman</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>56471</td>
<td>Geoffrey Arnold</td>
<td>0</td>
</tr>
</tbody>
</table>
```

Attribute Level
A program can interact with a database at different levels of granularity.
Test Adequacy Criteria

- $P$ violates a database $D_i$'s validity when it:
  - (1-v) inserts entities into $D_i$ that do not reflect real world

- $P$ violates a database $D_i$'s completeness when it:
  - (1-c) deletes entities from $D_i$ that still reflect real world

- In order to verify (1-v) and (1-c), $T$ must cause $P$ to define and then use entities within $D_1, \ldots, D_n$.!
Data Flow Information

→ Interaction point: `UPDATE UserInfo SET acct_lock=1'' + WHERE card_number=''' + c_n + ''' ;''`
→ Database Level: `define(BankDB)`
→ Attribute Level: `define(acct_lock)` and `use(card_number)`
→ Data fbw information varies with respect to the granularity of the database interaction
## Database Entities

### UserInfo

<table>
<thead>
<tr>
<th>card_number</th>
<th>pin_number</th>
<th>user_name</th>
<th>acct_lock</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32142</td>
<td>Brian Zorman</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>41601</td>
<td>Robert Roos</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>45322</td>
<td>Marcus Bittman</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>56471</td>
<td>Geoffrey Arnold</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ A_v(I_r) = \{ 1, 32142, \ldots, Geoffrey Arnold, 0 \} \]
The DICFG: A Unified Representation

- “Database-enhanced” CFG for `lockAccount`
- Define temporaries to represent the program’s interaction at the levels of database and attribute
The DICFG: A Unified Representation

- Database interaction graphs (DIGs) are placed before the database interaction point $I_r$
- Multiple DIGs can be integrated into a single CFG

```java
update_lock = m_connect.createStatement()
if (result_lock == 1)
    completed = true
exit lockAccount
```
Test Adequacy Criteria

- Database interaction association (DIA) involves the definition and use of a database entity.
- DIAs can be located in the DICFG with data flow analysis.
- *all-database-DUs* requires tests to exercise all DIAs for all of the accessed databases.
Generating Test Requirements

- Measured time and space overhead when computing family of test adequacy criteria
- Modified ATM and mp3cd to contain appropriate database interaction points
- Soot 1.2.5 to calculate intraprocedural associations
- GNU/Linux workstation with kernel 2.4.18-smp and dual 1 GHz Pentium III Xeon processors
DIAs at attribute value level represent 16.8% of mp3cd’s and 9.6% of ATM’s total number of intraprocedural associations.
Computing DIAs at the attribute value level incurs no more than a 5 second time overhead.
mp3cd shows a more marked increase in the average number of nodes and edges than ATM
mp3cd shows a significantly greater maximum space overhead than ATM.
Conclusions

- Must test the program’s interaction with the database
- Unique family of test adequacy criteria to detect type (1) violations of database validity and completeness
- Intraprocedural database interactions can be computed from a DICFG with minimal time and space overhead
- High number of hanging definitions indicates that the scope of data flow analysis could be broadened
- This data flow-based test adequacy criteria can serve as the foundation for algorithms that measure test suite quality, automatically generate test cases, and support regression testing

http://cs.allegheny.edu/~gkapfham/research/diatoms/