Towards Regression Testing for Database Applications

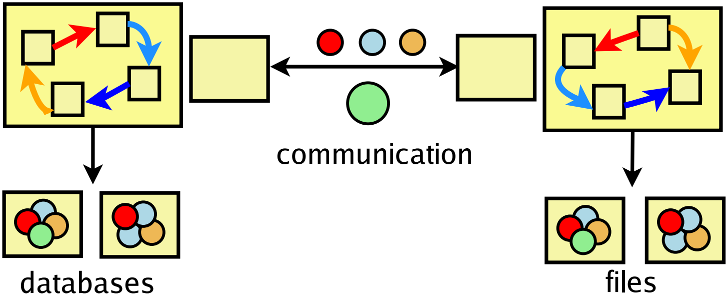
Gregory M. Kapfhammer[†]

Department of Computer Science
Allegheny College, Pennsylvania, USA
http://cs.allegheny.edu/~gkapfham/

ASTReNet and SOSoRNet, King's College London, 2007

[†] In Conjunction with Mary Lou Soffa (UVa/CS), Panos Chrysanthis (Pitt/CS), Bruce Childers (Pitt/CS)

Testing Database Applications



Research Contribution

A **regression testing** framework for traditional database applications. Future research includes service-oriented applications that use Grid-enabled databases.

An Interesting Defect Report

Database Server Crashes

When you run a complex query against Microsoft SQL Server 2000, the SQL Server scheduler may stop responding. Additionally, you receive an error message that resembles the following: **Date Time server Error: 17883 Severity: 1, State: 0 Date Time server Process 52:0 (94c) ...**

An Input-Dependent Defect

This problem occurs when one or more of the following conditions are true: The query contains a `UNION` clause or a `UNION ALL` clause that affects many columns. The query contains several `JOIN` statements. The query has a large estimated cost. **BUG 473858 (SQL Server 8.0)**

An Interesting Defect Report

Database Server Crashes

When you run a complex query against Microsoft SQL Server 2000, the SQL Server scheduler may stop responding. Additionally, you receive an error message that resembles the following: **Date Time server Error: 17883 Severity: 1, State: 0 Date Time server Process 52:0 (94c) ...**

An Input-Dependent Defect

This problem occurs when one or more of the following conditions are true: The query contains a `UNION` clause or a `UNION ALL` clause that affects many columns. The query contains several `JOIN` statements. The query has a large estimated cost. **BUG 473858 (SQL Server 8.0)**

Real World Example

A Severe Defect

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.

An Important Point

Practically all use of databases occurs from within application programs [Silberschatz et al., 2006, pg. 311]

Real World Example

A Severe Defect

The Risks Digest, Volume 22, Issue 64, 2003

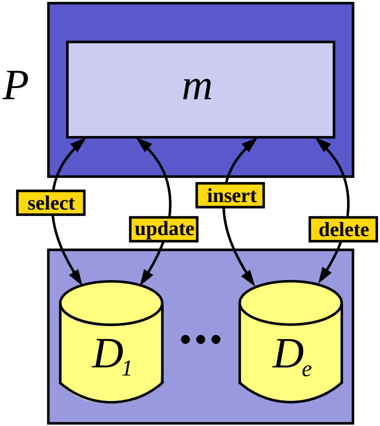
Jeppesen reports airspace boundary problems

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.

An Important Point

Practically all use of databases occurs from within application programs [Silberschatz et al., 2006, pg. 311]

Program and Database Interactions



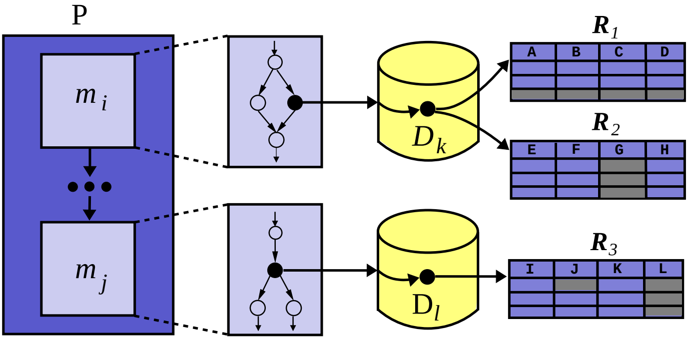
Basic Operation

Program *P* creates SQL statements in order to view and/or modify the state of the relational database

SQL Construction

Static analysis does not reveal the exact SQL command since the program constructs the full SQL statement at run-time

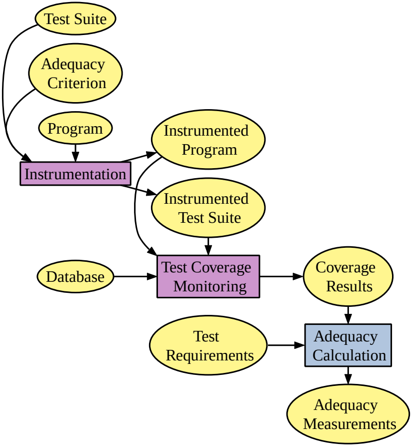
Database Interaction Granularity



Database Interactions

Program *P* interacts with two relational databases D_k and D_l at different levels of granularity (relation, record, attribute, ...)

Overview of the Coverage Monitoring Process



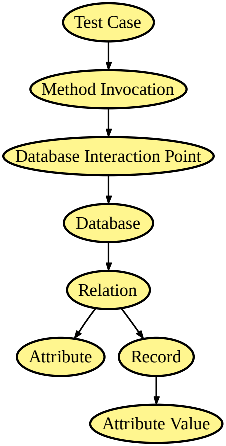
Calculating Coverage

Use instrumentation probes to capture and analyze a program's interaction with the databases

Regression Testing

The adequacy measurements can be used to support both test suite **reduction** and **prioritization**

Database-Aware Coverage Trees



Instrumentation Probes

Use **static** and **dynamic** (load-time) instrumentation techniques to insert coverage monitoring probes

Coverage Trees

Store the coverage results in a tree in order to support the calculation of many types of coverage (e.g., **data flow** or **call tree**)

Comparing the Coverage Trees

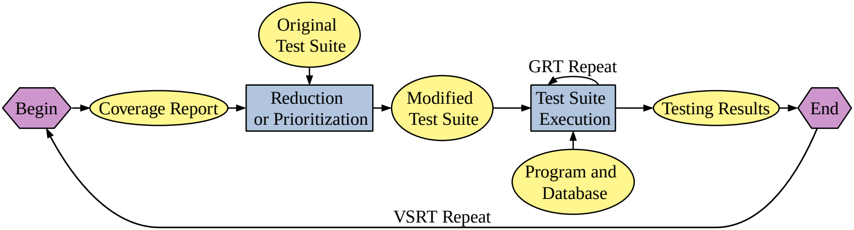
Tree Characteristics

Tree	DB?	Context	Probe Time	Tree Space
CCT	$\times$	Partial	Low - Moderate	Low
DCT	$\times$	Full	Low	Moderate - High
DI-CCT	$\checkmark$	Partial	Moderate	Moderate
DI-DCT	$\checkmark$	Full	Moderate	High

Table Legend

Database? $\in \{\times, \checkmark\}$
Context $\in$ {Partial, Full}
Probe Time Overhead $\in$ {Low, Moderate, High}
Tree Space Overhead $\in$ {Low, Moderate, High}

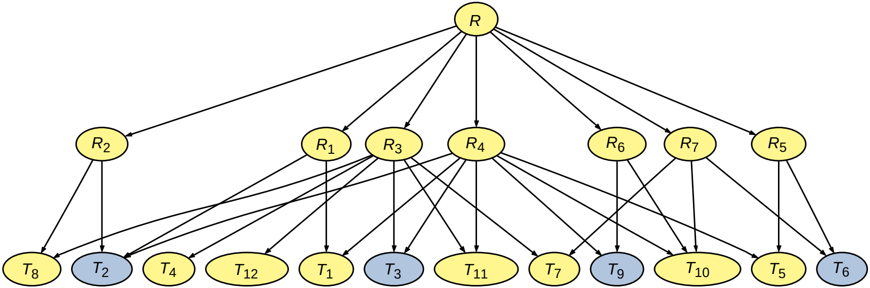
Database-Aware Regression Testing



Regression Testing Overview

Reduction aims to find a smaller test suite that covers the same requirements as the original suite. **Prioritization** re-orders the tests so that they cover the requirements more effectively.

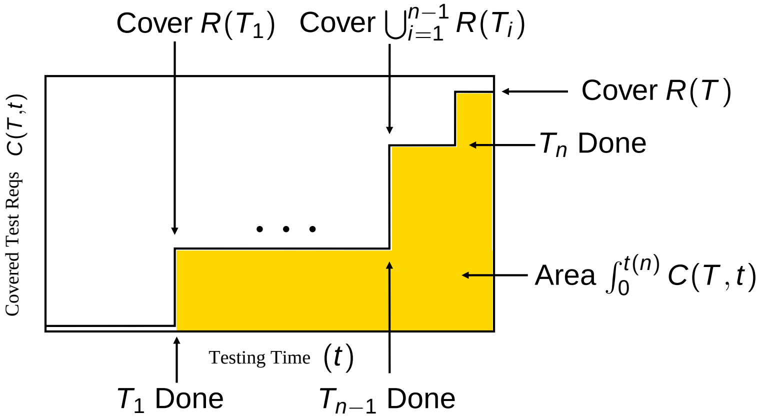
Finding the Overlap in Coverage



Test Suite Reduction

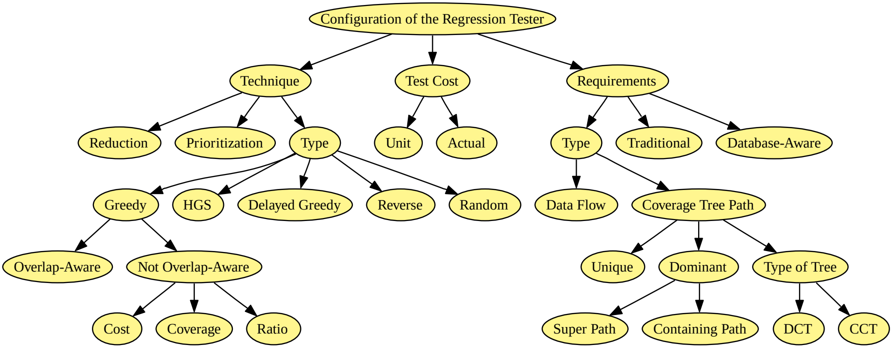
- $R_j \rightarrow T_i$ means that requirement R_j is *covered by* test T_i
- $T = \langle T_2, T_3, T_6, T_9 \rangle$ cover all of the test requirements

Measuring Coverage Effectiveness



- Prioritize to increase the CE of a test suite $CE = \frac{\text{Actual}}{\text{Ideal}}$

Configuring the Regression Testing Framework



- Regression tester uses several algorithms and test requirements

Characterizing the Case Study Applications

Test Suites

Application	# Tests	Test NCSS / Total NCSS
RM	13	$227/548 = 50.5\%$
FF	16	$330/558 = 59.1\%$
PI	15	$203/579 = 35.1\%$
ST	25	$365/620 = 58.9\%$
TM	27	$355/748 = 47.5\%$
GB	51	$769/1455 = 52.8\%$

Details About the Database Interactions

Static Interaction Counts

Application	executeUpdate	executeQuery	Total
RM	3	4	7
FF	3	4	7
PI	3	2	5
ST	4	3	7
TM	36	9	45
GB	11	23	34

Dynamic Interaction Counts

Database interactions that occur in **iterative** or **recursive** computations are executed more frequently

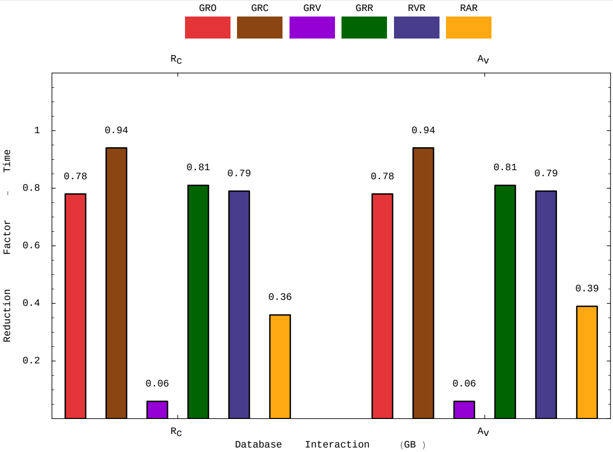
Reducing the Size of the Test Suite

(Size of Reduced Test Suite, Reduction Factor)

App	Rel	Attr	Rec	Attr Value
RM (13)	(7, .462)	(7, .462)	(10, .300)	(9, .308)
FF (16)	(7, .563)	(7, .563)	(11, .313)	(11, .313)
PI (15)	(6, .600)	(6, .600)	(8, .700)	(7, .533)
ST (25)	(5, .800)	(5, .760)	(11, .560)	(10, .600)
TM (27)	(14, .481)	(14, .481)	(15, .449)	(14, .481)
GB (51)	(33, .352)	(33, .352)	(33, .352)	(32, .373)
All (24.5)	(12, .510)	(12.17, .503)	(14.667, .401)	(13.83, .435)

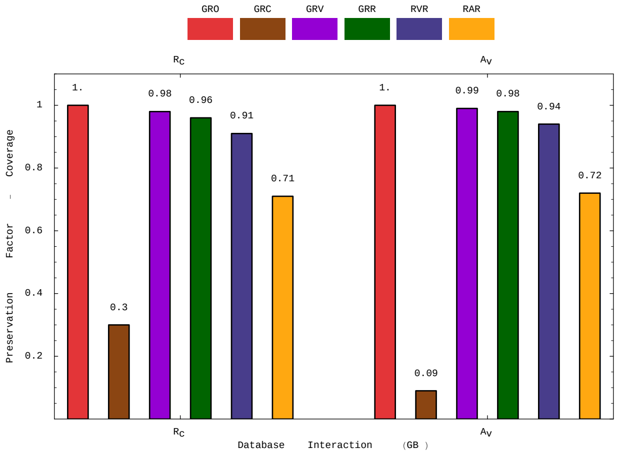
- Reduction factor for test suite size varies from .352 to .8

Reducing the Testing Time



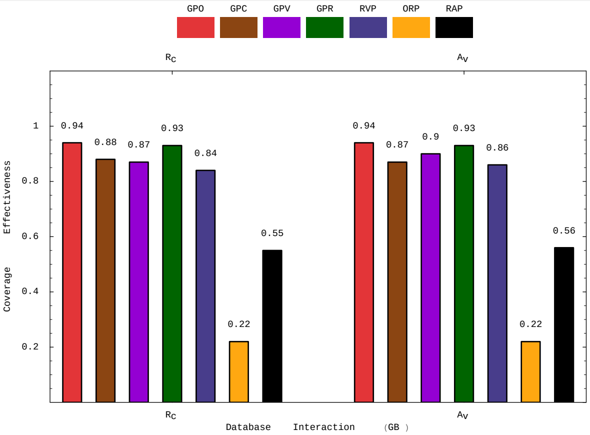
- GRO reduces test execution time even though it removes few tests

Preserving Requirement Coverage



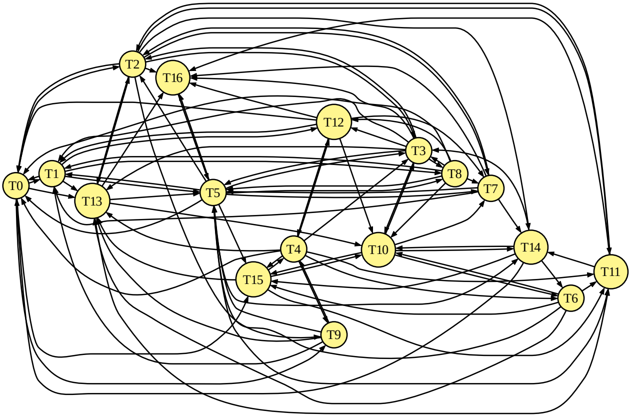
- GRO guarantees coverage preservation while the others do not

Improving Coverage Effectiveness



- GRO is the best choice and the original ordering is poor

Future Work: Avoid Database Restarts



- Use prioritization to reduce testing time by avoiding database restarts

Conclusions and Future Work

Concluding Remarks

- A new **perspective** on software testing and an **efficient** and **effective** method for database-aware regression testing

Future Work

- Challenges associated with grid-enabled databases
- Conduct experiments with larger database applications

Resources

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