

AutoDBT: A Framework for Automatic Testing of Web Database Applications

Lihua Ran, Curtis Dyreson, Anneliese Andrews
 School of E.E. and Computer Science
 Washington State University
 USA

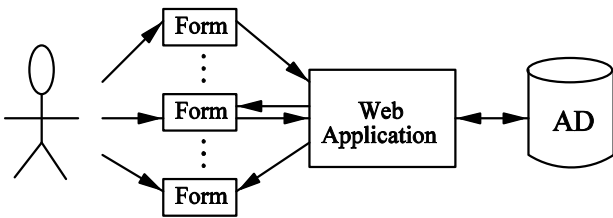
WISE 2004 - Brisbane

Outline

- Motivation
- Two approaches
- AutoDBT
- Implementation plan
- Related work
- Conclusion and future work

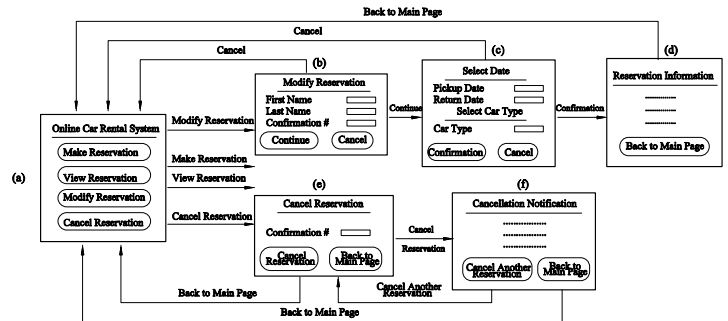
AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Architecture of Web Applications



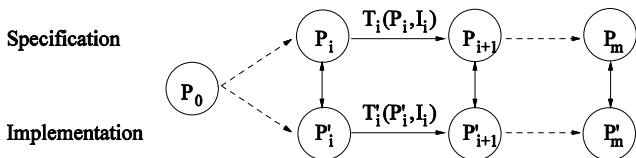
AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Example Web Application



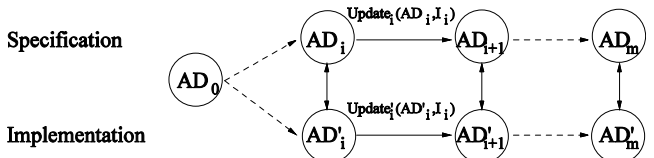
AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Black Box Testing



AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Testing a Web Database Application



AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Additional Problems

- Evolving database state
 - Non-database: each test starts from same initial state
 - Database: database states differ for each test
- Modeling of database updates
- Correctly choosing test data
 - Non-database: manufactured as needed
 - Database: chosen from two dynamic changing sources (AD & SD)
- Data Dependency between transitions
- Correctness of each AD state
 - Non-database: expected page
 - Database: page + AD state

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Outline

- Motivation
- Two approaches
- AutoDBT
- Implementation plan
- Related work
- Conclusion and future work

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Two Complementary Approaches

- Dirty Test Suite
 - Generated in advance
 - Based on the initial database state
- Clean Test Case (AutoDBT)
 - Generated on the fly
 - Based on the most current database state

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Dirty Test Suite Approach

- Why “dirty”
 - Too expensive to keep copies of AD0
 - Practical solution: run each test case on ADi
 - ADi <> AD0
 - The preset test case might be invalid based on ADi
- Solution
 - Filter query

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Comparing Approaches

- Dirty test suite approach is more efficient
 - No overhead at run time
 - Reused to retest the same application
- Clean test case approach is guaranteed to make progress

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Outline

- Motivation
- Two approaches
- AutoDBT
- Implementation plan
- Related work
- Conclusion and future work

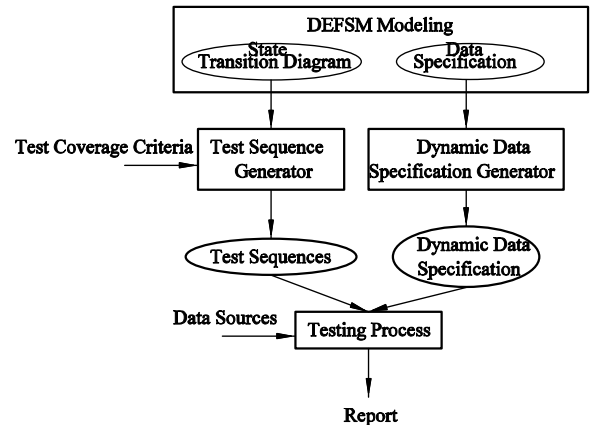
AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

AutoDBT's Framework

- DEFSM Modeling
 - State Transition Diagram
 - Data Specification
- Test Sequence and Dynamic Data Specification Generation
- Testing Process

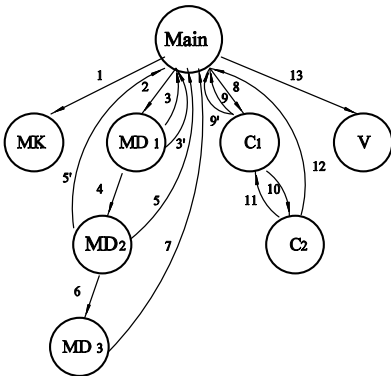
AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

AutoDBT's Framework



AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Step 1: State Transition Diagram



AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Step 2: Data Specification

- Input Specification
 - Data Sources
 - Application Database (AD): existing data
 - Synthetic Database (SD): new data
 - Data Flow (Data Inheritance)
- Update Specification
- Specification Language: Prolog
 - Well-defined declarative semantics
 - Evaluation model: Top-down, tuple-at-a-time.

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Data Specification

- Assumption
 - AD/SD schema are known
 - Same schemas for AD and SD (can be relaxed)
- Populate AD/SD
 - Gray et al.

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

An Example Application Database (AD)

Customer		
FirstName	LastName	CustomerID
john	smith	c0001
mike	green	c0002
rick	reed	c0003
kate	brown	c0004

Reserves	
CustomerID	Confirmation#
c0001	0001
c0002	0002
c0003	0003
c0004	0004
c0004	0005

Reservation			
Confirmation#	PickupDate	ReturnDate	CarType
0001	10/01/03	10/03/03	economy
0002	10/02/03	10/05/03	compact
0003	10/15/10	10/23/03	full size
0004	11/03/03	11/30/03	minivan
0005	11/03/03	11/30/03	full size

Available_Car_Type	
CarType	CarNumber
economy	50
compact	40
full size	60
minivan	45
luxury	20
convertible	30

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

An Example Synthetic Database (SD)

Customer		
FirstName	LastName	CustomerID
franklin	bond	c0006
alicia	wong	c0007

Reserves	
CustomerID	Confirmation#
c0006	0006
c0007	0007

Reservation			
Confirmation#	PickupDate	ReturnDate	CarType
0006	12/04/03	12/10/03	luxury
0007	12/06/03	12/08/03	convertible

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Input Specification Definition

- A button name or a Prolog rule of the following form followed by a button name:

$$input_i(V_{i1}, \dots, V_{in}) :- Predicate_1, \dots, Predicate_m.$$

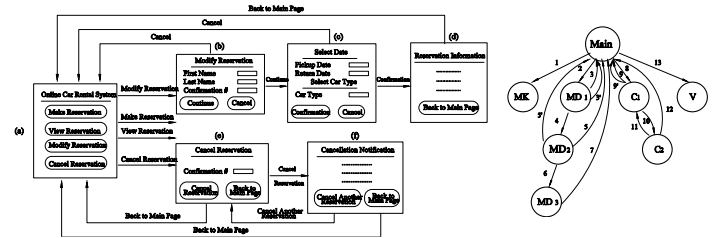
AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Input Specification

Transition	Input Data Flow Specification
2	Button(Modify Reservation).
3	$input_3(Fn, Ln, C\#) :- AD_Customer(Fn, Ln, Cid), AD_Reserves(Cid, C\#).$ Button(Cancel).
3'	$input_{3'}(Fn, Ln, C\#) :- SD_Customer(Fn, Ln, Cid), SD_Reserves(Cid, C\#).$ Button(Cancel).
4	$input_4(Fn, Ln, C\#) :- AD_Customer(Fn, Ln, Cid), AD_Reserves(Cid, C\#).$ Button(Continue).
5	$input_5(Pd, Rd, Ct) :- AD_Reservation(., Pd, Rd, Ct).$ Button(Cancel).
5'	$input_{5'}(Pd, Rd, Ct) :- SD_Reservation(., Pd, Rd, Ct).$ Button(Cancel).
6	$input_6(Pd, Rd, Ct) :- SD_Reservation(., Pd, Rd, Ct).$ Button(Confirmation).
7	Button(Back to Main Page).
8	Button(Cancel Reservation).
9	$input_9(C\#) :- AD_Reservation(C\#, .., ..).$ Button(Back to Main Page).
9'	$input_{9'}(C\#) :- SD_Reservation(C\#, .., ..).$ Button(Back to Main Page).
10	$input_{10}(C\#) :- AD_Reservation(C\#, .., ..).$ Button(Back to Main Page).
11	Button(Cancel Another Reservation).
12	Button(Back to Main Page).

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

OCRS



AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Input Specification

Transition	Input Data Flow Specification
2	Button(Modify Reservation).
3	$input_3(Fn, Ln, C\#) :- AD_Customer(Fn, Ln, Cid), AD_Reserves(Cid, C\#).$ Button(Cancel).
3'	$input_{3'}(Fn, Ln, C\#) :- SD_Customer(Fn, Ln, Cid), SD_Reserves(Cid, C\#).$ Button(Cancel).
4	$input_4(Fn, Ln, C\#) :- AD_Customer(Fn, Ln, Cid), AD_Reserves(Cid, C\#).$ Button(Continue).

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Update Specification Definition

- The update specification is one or more Prolog rules of the following two forms.

$delete_AD_Relation(A_1, \dots, A_n) :- Predicate_1, \dots, Predicate_m.$

$insert_AD_Relation(A_1, \dots, A_n) :- Predicate_1, \dots, Predicate_m.$

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Update Specification Example

Transition	Update Specification
6	delete_AD_Reservation($C\#, Pd, Rd, Ct$) :- $input_4(_, _, C\#)$, AD_Reservation($C\#, Pd, Rd, Ct$). insert_AD_Reservation($C\#, Pd, Rd, Ct$) :- $input_4(_, _, C\#)$, $input_6(Pd, Rd, Ct)$.
10	delete_AD_Reservation($C\#, Pd, Rd, Ct$) :- $input_{10}(C\#)$, AD_Reservation($C\#, Pd, Rd, Ct$). delete_AD_Reserves($Cid, C\#$) :- $input_{10}(C\#)$, AD_Reserves($Cid, C\#$).

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Step 3: Test Sequence Generation

- Test Sequence Generation
 - A test sequence: a sequence of transitions
 - Apply graph theory algorithm on the state transition diagram

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

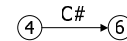
Step 4: Dynamic Data Generation

- Test case
 - An instantiation of the input parameters for the entire test sequence
 - Generated based on a sequence of dynamic changing AD states
- Dynamic data specification
 - Combines the input and update specification
 - Two Components:
 - ♦ Data Inheritance Graph (DIG)
 - ♦ Dynamic Input Specification (DIS)

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Data Inheritance Graph (DIG)

- Model data inheritance
- Generated based on data specification
- Example:



Transition	Update Specification
6	delete_AD_Reservation($C\#, Pd, Rd, Ct$) :- $input_4(_, _, C\#)$, AD_Reservation($C\#, Pd, Rd, Ct$). insert_AD_Reservation($C\#, Pd, Rd, Ct$) :- $input_4(_, _, C\#)$, $input_6(Pd, Rd, Ct)$.
10	delete_AD_Reservation($C\#, Pd, Rd, Ct$) :- $input_{10}(C\#)$, AD_Reservation($C\#, Pd, Rd, Ct$). delete_AD_Reserves($Cid, C\#$) :- $input_{10}(C\#)$, AD_Reserves($Cid, C\#$).

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Dynamic Input Specification (DIS)

- Specify new versions of the AD/SD
- Automatically merging the input and update specifications
- Example

$input_6(Pd, Rd, Ct, C\#, AD_{in}, AD_{out}, SD_{in}, SD_{out})$:-
 SD_Reservation($_, Pd, Rd, Ct, SD_{in}$),
 AD_Reservation($C\#, Pd', Rd', Cr', AD_{in}$),
 delete_AD(Reservation($C\#, Pd', Rd', Cr'$), AD_{in}, AD_t),
 insert_AD(Reservation($C\#, Pd, Rd, Ct$), AD_t, AD_{out}),
 delete_SD(Reservation($_, Pd, Rd, Ct$), SD_{in}, SD_{out}).

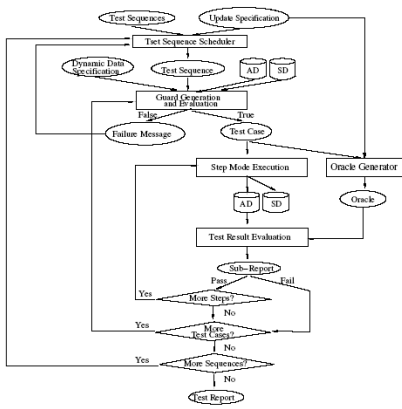
AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Step 5: Testing Process

- Test sequence scheduler
- Guard generation and evaluation
- Oracle generation
- Step mode execution
- Test result evaluation

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Testing Process



AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Test Sequence Scheduler

- Motivation: Mitigate the impact of prior tests
- Strategy: Delay deletion
- Method:
 - Test sequences are classified into four groups: read-only, insertion-only, mixed insertion and deletion, and deletion-only
 - Scheduled in that order

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Guard Generation

- Motivation
 - Maintain data inheritance
 - Generate each test case as a whole
- Strategy: Guard
 - Prolog query
 - Determine test case generation possibility
 - Constructed by concatenating the heads of corresponding Dynamic Input Specification rules

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Guard Generation (Example)

- Test sequence for Modify Reservation path:
- 2->3->2->3'->2->4->5->2->4->5'->2->4->6->7
- Corresponding guard query:

$$\begin{aligned} &?- \text{input}_3(Fn_1, Ln_1, C\#_1, AD_{in}, AD_1), \\ &\text{input}_{3'}(Fn_2, Ln_2, C\#_2, SD_{in}, SD_1), \\ &\text{input}_4(Fn_3, Ln_3, C\#_3, AD_1, AD_2), \\ &\text{input}_5(Pd_1, Rd_1, Ct_1, AD_2, AD_3), \\ &\text{input}_4(Fn_4, Ln_4, C\#_4, AD_3, AD_4), \\ &\text{input}_{5'}(Pd_2, Rd_2, Ct_2, SD_1, SD_2), \\ &\text{input}_4(Fn_5, Ln_5, C\#_5, AD_4, AD_5), \\ &\text{input}_6(Pd_3, Rd_3, Ct_3, C\#_5, AD_5, AD_6, SD_2, SD_3). \end{aligned}$$

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Guard Evaluation

- Using the Prolog interpreter
- Successful evaluation
 - Test Case
- Unsuccessful evaluation
 - ADi: bad state for test case generation
 - Test sequence: put back to the queue.

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

Test Result Evaluation

- Step mode execution
 - Transaction by transaction
- Test Result Evaluation
 - After each transaction commits
 - ADi: checked by an oracle

AutoDBT: Testing a Web Database Application - Ran, Dyreson, Andrews

