Manage Risk by Risk-Driven Continual Regression Testing

Yanping Chen

School of Information Technology and Engineering, University of Ottawa

Outline

- Risk and risk-based testing
- Regression testing and risk-based continual regression testing
- Risk-based regression test case selection
- Risk-based end-to-end scenario selection
- Real experience to date
- Summary and recommendations
- Reference

Risk and Risk-based Testing

- Risk: event that has some probability of happening, and that if it occurs, will result in some loss
- Risk-based testing: do heavier testing of those parts that may bring higher risk
- Risk-based testing actions
  - Identify risk for functions or features
  - Quantify risk and create ranked list of functions or features
  - Design test cases based on ranked list

Why Risk-based Testing?

- *All testing is motivated by risk: Tester's job is finding high-priority problems to avoid risk*
- Traditional testers have always used risk-based testing, but in ad hoc fashion based on their personal judgment [4]
- Using risk to measure quality of test suite is reasonable

"Risk-based testing" vs. "~~Food~~-based living"
Air

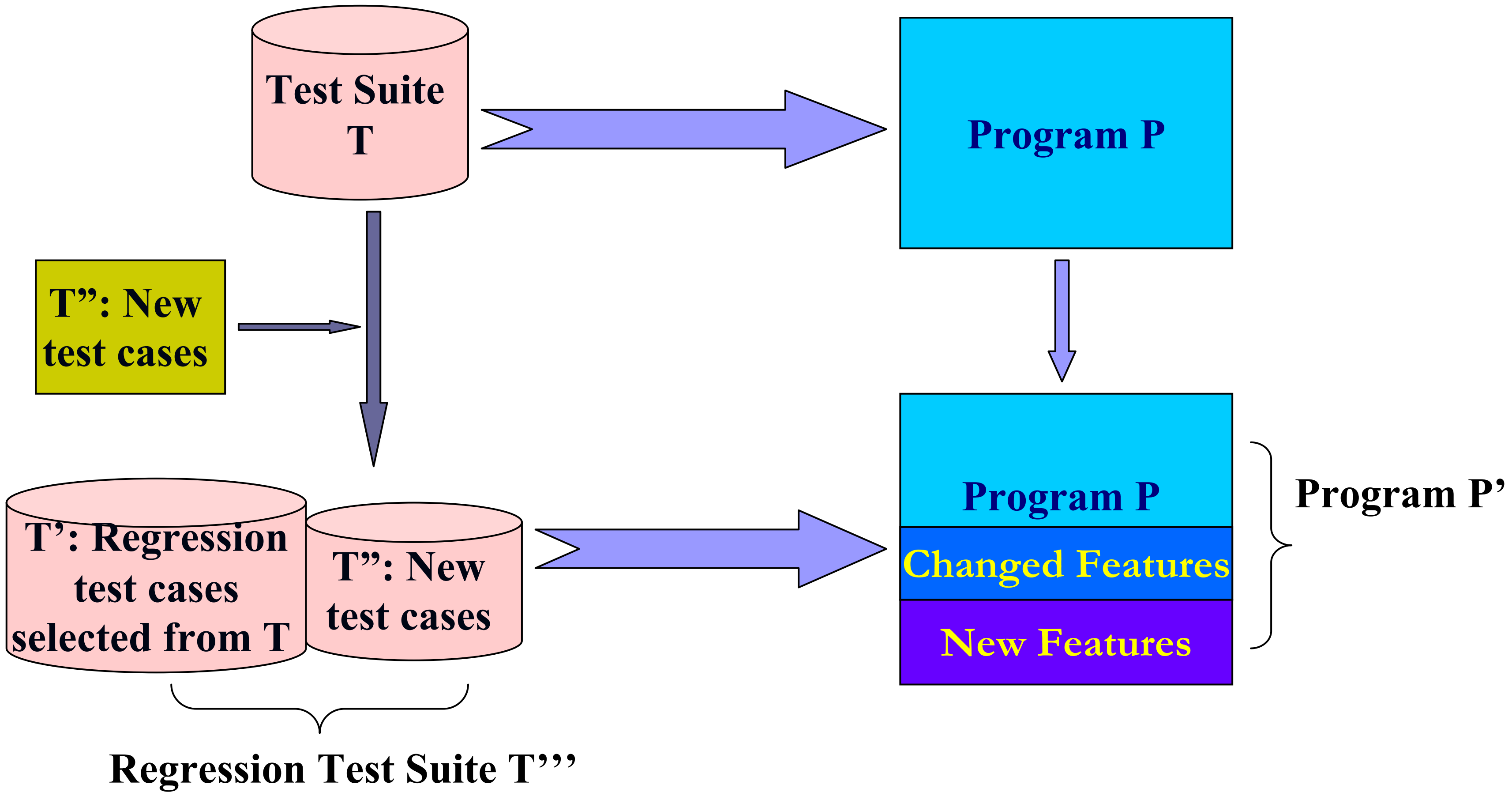
Questions to asked for risk-based approach

- Which areas are significant?
- How much testing is enough for average area?
- What are *risks* involved in leaving certain bug unresolved?
- At what point can product be considered adequately tested and ready for market?

Continual Regression Testing

- *To ensure that new or modified features do not cause current release to regress after incorporating fixes into product -- ensure customer’s business won’t be at risk*
- Essential to ensure software quality
- In software maintenance: validate modified software
- In O-O software development
  - Ensure quality of successive increments
  - Assess quality of re-used components
- Continual regression testing: execute regression tests every day or on every new build

Typical Regression Test Selection



A Simple Risk Model [2]

- Two elements of Risk Exposure (RE_f):
  - Probability of fault
  - Cost (consequence or impact) of fault in corresponding function if it occurs in production
- $RE_f = P_f \times C_f$
  - RE_f: Risk Exposure of function *f*
  - P_f: probability of fault occurring in function *f*
    - In our model, we consider severity of defects to assess probability
    - **Note: P (*f*) is extended to severity probability**
  - *C_f: cost if fault occurs (in production) in function f*

Risk-based Regression Testing Approach

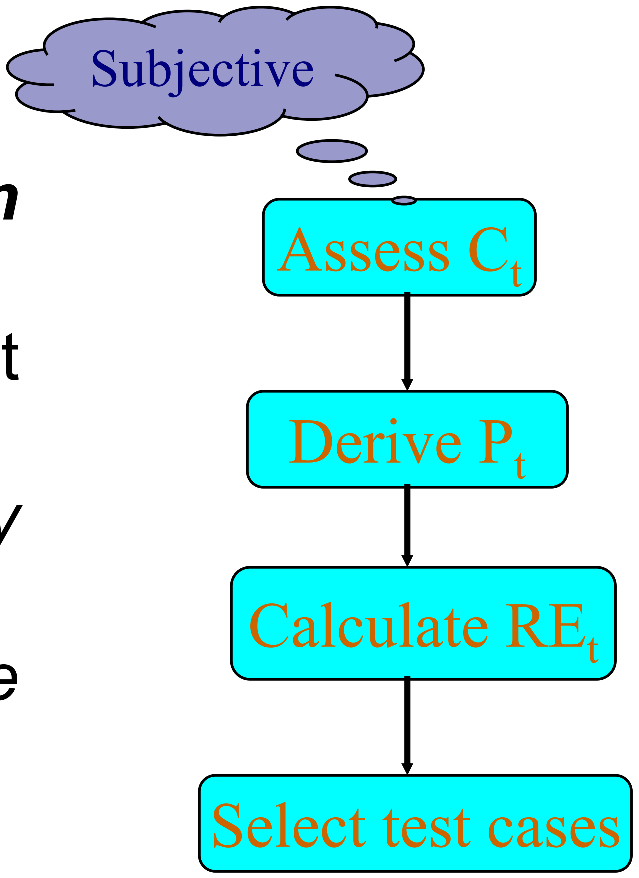
Model-based Tests Selection Method:

Step 1. Assess cost C_t for each test case

Step 2. Derive *severity probability* P_t for each test case

Step 3. Calculate *Risk Exposure* RE_t for each test case

Step 4. Select test cases with top Re_t as regression test cases



Assess *Cost* C_t

- *Two kinds of costs*
  - C_t (*c*): Consequences of fault as seen by customer, i.e., losing market place
  - C_t (*v*): Consequences of fault as seen by vendor, i.e., high software maintenance cost
- C_t is categorized on 1~5 scale (1- low, 5 - high)
  - Weight C_t (*c*) and C_t (*v*) equally
  - $C_t = (C_t\,(c) + C_t\,(v))/2$??

Assess *Cost* C_t (Cont'd)

- C_t (*c*)
  - Test case takes one, specific control flow and includes some data
  - Create questionnaire with questions for both control flow and data
  - Score each test case based on answers for questionnaire as C_t (*c*), on 1~5 scale (1- low, 5 - high)
- C_t (*v*)
  - Cost to fix bugs is dependent on system complexity
  - Use proper questionnaire in assessment
  - Measure C_t (*v*) on 1~5 scale (1- low, 5 - high)

Derive *Severity Probability* P_t

- Summarize number of defects opened for each test case after running full test suite
- Calculate average severity of defects for each test case
- *Use result of Number of Defects (N_t) times Average Severity (S_t) $N_t \times S_t$ to assess severity probability*
- P_t falls into 1~5 range (1 - low, 5 - high)
  - Test cases without any defects in full testing, P_t =1.
  - Test cases with the top 25% estimate $N_t \times S_t$, $P_t = 5$
  - Test cases with the bottom 25% estimate $N_t \times S_t$, $P_t = 2$

Calculate *Risk Exposure* RE_t

Step 1:

Test Case	C_t
t0010	5
t0020	2
…	…
t_n	3

+

Step 2:

Test Case	N_t	S_t	P_t
t0010	3	2	5
t0020	1	1	4
……	…..	……	……
t_n	0	0	1

=

Test Case	C_t	P_t	$RE_t = P_t \times C_t$
t0010	5	5	25
t0020	2	4	8
……	…	……	……
t_n	3	1	3

Select Test Cases with Top RE_t

- Choose test cases with highest value of RE_t
- Reach pre-defined coverage target (e.g., 30% of full test suite)

Test Case	**Full Test Suite**	**Regression Test Suite (30%)**
t0010	1	1
t0020	1	1
t0030	1	0
t0040	1	0
t0050	1	0
t0060	1	1
t0070	1	0
…	…	…

Risk-based End-to-end Regression Test *Scenario* Selection

- Test Scenario
  - Simulate common user profiles of system use
  - More customer-directed
  - Highly effective at finding regression faults
  - Covers sequence of test cases -- *Traceability*
- Selection rules
  - Select scenarios that contain most critical test cases
  - Have test suite of scenarios cover as many test cases as possible

Risk-based Regression Test Scenario Selection

End-to-end Test Scenario Selection Method

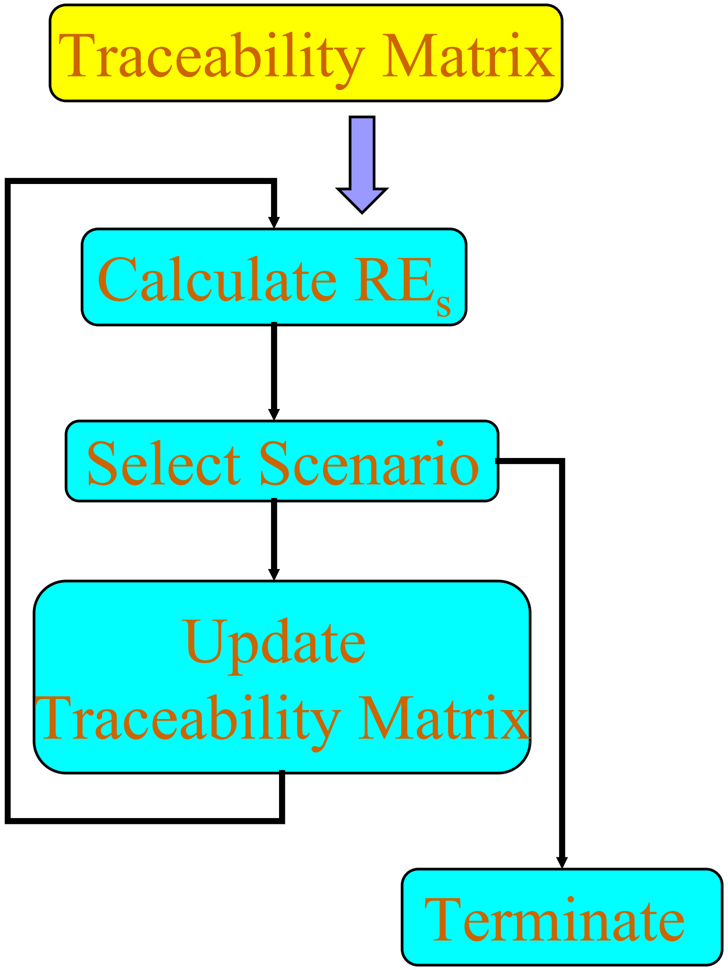
To start: Create traceability matrix between scenarios and test cases

Step 1. Calculate Risk Exposure RE_s for each scenario

Step 2. Select scenario with highest RE_s as regression tests

Step 3. Update traceability matrix and *re-calculate* RE_s

Step 4. Repeat Steps 2 and 3 until out of time and resources



End-to-end Test Scenario Selection Method with Example

- Step 1. Calculate Risk Exposure RE_s for each scenario
  - $RE_s = \sum RE_{t_i}$, $\{1 \leqslant i \leqslant n \mid$ test case t_i is covered by this scenario$\}$
- Step 2. Select scenario with highest RE_s for regression testing

	s001	s002	s003	…
t0010	1	0	0	…
t0020	1	0	0	…
t0030	1	1	0	…
t0040	1	0	1	…
t0050	1	1	0	…
t0060	0	1	0	…
t0070	0	1	1	…
…	…	…	…	…

	C_f	P_f	$RE_f = P_f \times C_f$
t0010	5	5	25
t0020	2	4	8
……	…	……	……
t_n	3	1	3

Scenario	RE_s
s001	985
s002	463
s003	732
s004	213
s005	195
s006	127
s007	70
…	…

- Step 3. Update traceability matrix and *re-calculate* RE_s
  - When running chosen scenario, some test cases will be covered – not necessary to cover again
  - Thus, after chosen scenario has been executed
    - Delete column for chosen scenario
    - Delete rows for all test cases that have been covered by this scenario
  - Based on updated relation table, *re-calculate* RE_s for rest scenarios and *re-build* Risk Exposure table
- Step 4. Repeat Steps 2 and 3 until out of time and resources
  - Size of test suite is dependent on time and resources

	s001	s002	s003	…
~~t0010~~	1	0	0	…
~~t0020~~	1	0	0	…
~~t0030~~	1	1	0	…
~~t0040~~	1	0	1	…
~~t0050~~	1	1	0	…
t0060	0	1	0	…
t0070	0	1	1	…
…	…	…	…	…

Next choice -- s003

Scenario	**REs**
s002	356
s003	611
s004	176
s005	180
s006	96
s007	68
…	…

Case Study with historical data of *IBM WebSphere*

- Three components of IBM WebSphere with different characters
  - Component One: Focus on functionality
  - Component Two: Focus on data
  - Component Three: Both functionality and data are important
- Each component was owned by one experienced tester
- 306 test cases in total

Real Experiences to Date

- High Risk Exposure coverage and average Risk Exposure
- Acceptable specification coverage not our focus
- Only requires straightforward calculation – *can be automated*
- Systematic – *not subjective* !
- Powerful in *selecting effective test cases and finding defects*
  - Caught all defects
  - Omitted fewer test cases that failed in execution

	Risk-based Test Suite	Manual Test Suite	Compared Well
Defects Detected (%)	100%	84.1%	√√
Defect-revealing Test Cases Selected (%)	93.9%	83.1%	√

Summary

- New risk-based regression test technique
  1. *Risk-based* regression *test case* selection
  2. *Risk-based* regression *test scenario* selection
- New objective selection criteria that has good potential to guide regression test selection, even for new or less-experienced test personnel – SYSTEMATIC APPROACH!
- An EFFECTIVE means of QUANTIFYING quality of test suite

Recommendations for Adoption in Process

- Highlight & motivate ***RISK***
  - Analysis
  - Planning
  - Results
- Collect risk data
  - Test plan
    - Cost of test cases
    - Scenarios *vs.* test cases
  - Test profile
    - Number of defects by test case
    - Defect severity
- Measure efficiency & effectiveness
  - % defect detection
  - % defect-revealing test case coverage

Reference

[1] John D. McGregor, and David A. Sykes, *A Practical Guide to Testing Object-Oriented Software*, Addison Wesley Inc., 2001

[2] Gregg Rothermel and Mary Jean Harrold, Analyzing Regression Test Selection Techniques, *IEEE Transactions on Software Engineering*, V.22, no. 8, August 1996, pages 529 – 551

[3] Robert V. Binder, *Testing Object-Oriented Systems*, Addison-Wesley Publishing Inc., 2000

[4] Stale Amland, Risk Based Testing and Metrics: Risk analysis fundamentals and metrics for software testing including a financial application case study, *5th International Conference EuroSTAR'99*, November, 1999.

[5] Yanping Chen, *Specification-based Regression Testing Measurement with Risk Analysis*, Masters Thesis, University of Ottawa, Canada, 2002.

[6] Glenford L. Myers, *The Art of Software Testing*, Wiley-Interscience, 1979.

[7] Hung Nguyen, *Testing Applications on the Web*, Wiley-Interscience, 2003.