Regression Test Selection Techniques

Sanghyun Yoon

2012. 10. 12

KU KONKUK UNIVERSITY

DEPENDABLE SOFTWARE LABORATORY

Contents

- What is regression test?
- Classification of regression test
- Concepts related to regression testing
- Graph walk-based technique
- DFA model-based approach

What is regression test?

- Purpose of regression test
  - To ensure that the modifications do not introduce new bugs into previously validated code.

- Regression test mainly carried out **unmodified parts** of the program.

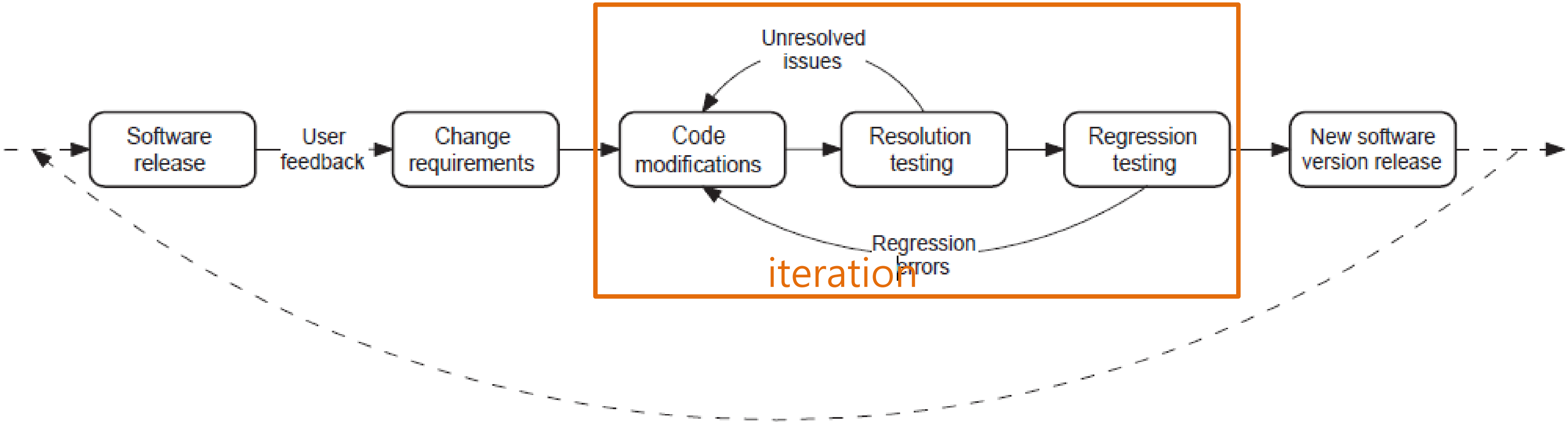
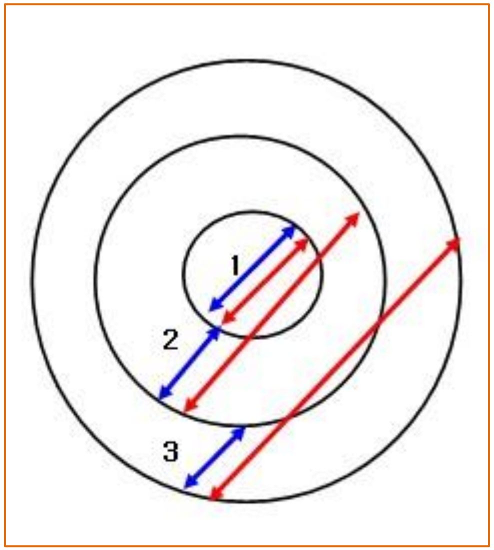


Figure 1: Activities that take place during software maintenance and regression testing.



Blue: program changes
Red: test boundary

- Regression test is a necessary but expensive maintenance activity.
- To optimize regression test, many techniques are proposed.

Classification of regression test – by approach

- Regression test selection (RTS) techniques
  - Select a sub-set of valid test cases from an initial test suite (T) to test that the affected but unmodified parts.

  - Identification of the affected parts
  - Test case selection

- Regression test suite minimization (TSM) techniques
  - Eliminate redundant test cases such that the coverage achieved by initial test case suite.

- Regression test case prioritization (TCP) techniques
  - Higher priority (fault-detection capability) test case execution should taken earlier.

Classification of regression test

- By program paradigms
  - Procedural, object-oriented, component-based, database, aspect, and web applications.
- By model, graph
  - Procedural: data flow-based, module level firewall-based, differencing-based, control flow analysis-based
  - Object-oriented: firewall-based, program model-based, design model-based, specification-based
  - …
- By develop level
  - System, unit, integration

Class of RTS Techniques	References	Key Features	Merits	Demerits
Dataflow analysis-based techniques	[37, 43, 44, 92]	Based on dataflow and structural coverage criteria	Can analyze both intra- and inter-procedural modifications provided the modifications alter some def-use relations	Low on safety, imprecise
Slicing-based techniques	[7, 10, 2]	Based on slicing of programs or dependence graph models	Can analyze both intra- and inter-procedural modifications	Low on safety, imprecise, computationally more expensive than dataflow techniques
Module level firewall-based techniques	[56, 58]	Based on analyzing dependencies among modules	Comparatively more efficient as analysis of source code is limited to only modified modules	Low on safety, and highly imprecise
Modified code entity-based technique	[17]	Level of granularity can be adapted	Safe, and most efficient procedural RTS technique	Highly imprecise
Textual differencing-based technique	[97, 98, 30]	Based on textual differencing of C programs	Safe, and comparatively easy to implement a prototype	Imprecise, and difficult to adapt to other languages, maybe inefficient for large programs
Graph walk-based technique	[80]	Based on analysis of control flow models	Safe and most precise procedural RTS technique	Less efficient than [17, 56, 58]

Table 1: A comparison of RTS techniques for procedural programs.

Concepts related to regression testing

- *P* is a program.
- *P′* is a modified *P*.

- *G* is a CFG for *P*.
- *G′* is a CFG for *P′*.

- *t* is a test case.
- *ET(P(t))* is the execution trace of a test case *t* on a program *P*.
  - Sequence of a statements in *P* when *t* is executed.

- *n* is a node of *ET(P(t))*.
- *n′* is a node of *ET(P′(t))*.

Graph walk-based technique

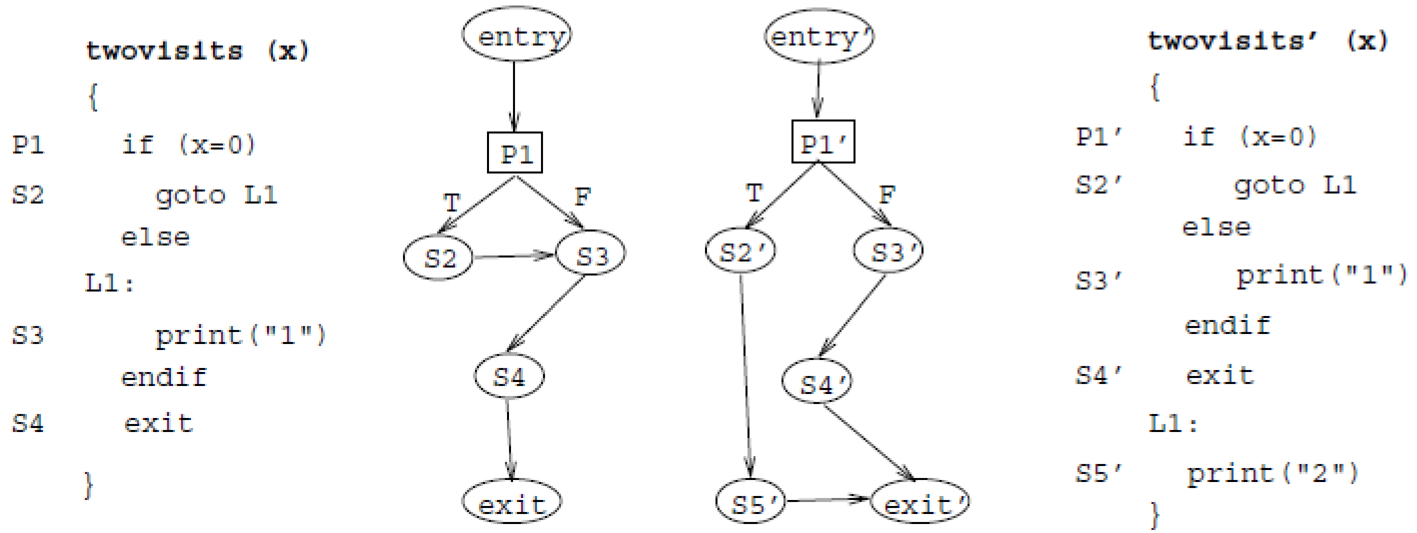


Figure 5: Procedures twovisits and twovisits', and their CFGs.

- Find states from *G* and *G'*.
- Check successor *n* and *n'* the states.
- If they are not identical, the edges that lead to the nodes are dangerous edges.

```
algorithm   SelectInterTests( P, P', P_E, P'_E, T ) : T'
input       P,P': base and modified versions of a program or subsystem
            P_E,P'_E: entry procedures to P and P'
            T: a test set used previously to test P
output      T': the subset of T selected for use in regression testing P'
data        proctable: contains fields name and status

begin
    T' = φ
    proctable = φ
    SelectTests2( P_E, P'_E )
    return T'
end


algorithm   SelectTests2( P, P' )
input       P,P': base and modified versions of a procedure

begin
    add P to proctable, setting its status to "visited"
    construct G and G', CFGs for P and P', with entry nodes E and E'
    Compare2( E, E' )
    if the exit node in G is not marked "S visited" for some node S in G'
        set the status flag for P to "selectsall"
    endif
end


procedure   Compare2( N, N' )
input       N and N': nodes in G and G'

begin
    mark N "N'-visited"
    for each successor C of N in G do
        L = the label on edge ( N, C ) or ε if the edge is unlabeled
        C' =the node in G' such that ( N', C' ) has label L
        if C is not marked "C'-visited"
            if ¬ LEquivalent( C, C' )
                T' = T' ∪ TestsOnEdge(( N, C ))
            else
                for each procedure O called in C do
                    if O ∉ proctable or status for O is not "visited" or "selectsall"
                        SelectTests2( O, O' )
                    endif
                endfor
                if any procedures called in C do not have status flag "selectsall"
                    Compare2( C, C' )
                endif
            endif
        endif
    endfor
end
```

Figure 9: Algorithm for interprocedural test selection.

DFA model-based approach

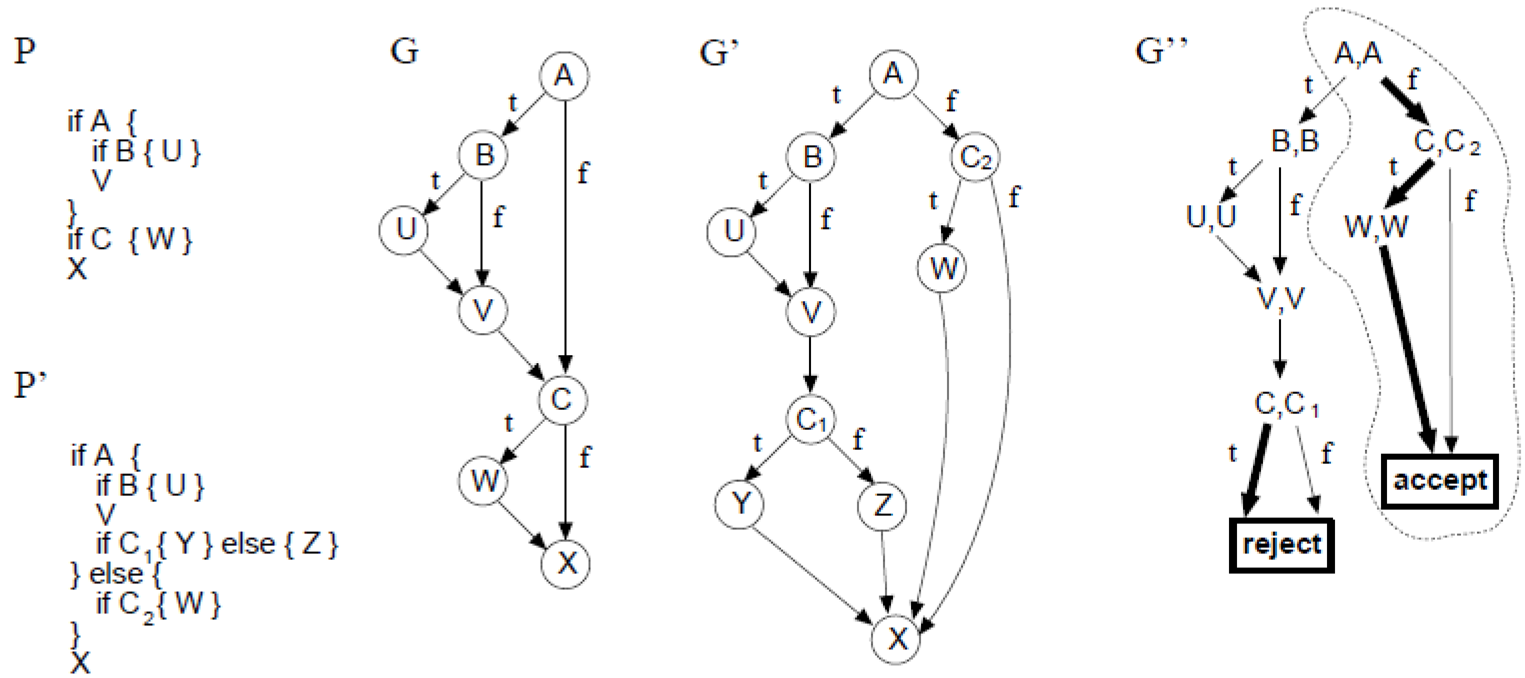


Figure 1: Example programs *P* and *P'*, their corresponding control flow graphs *G* and *G'*, and the intersection graph *G''* of *G* and *G'*.

- Modeling CFG *G* for a program *P* as a deterministic finite state automaton (DFA) *M*.