Data Slice Testing

Arnav Gupta

April 5, 2024

Contents

1 Data Slices

1.1 Motivation

Argued that testing a class aims at finding the sequence of operations for which a class will be in a state or has produced output that is contradictory to its expected results.

Testing classes for all possible sequences is not usually possible, so necessary to reduce the number of sequences and still provide sufficient confidence.

Can have state-dependent faults: executing some operations is not possible in certain states where they are legal

1.2 Principles

Goal: reduce the number of method sequences to test

The concrete state of an object at a single instant of time is equivalent to the aggregated state of each of its data members at that instant.

Correctness of a class depends on:

- whether the data members are correctly representing the intended state of an object
- whether the member function are correctly manipulating the representation of the object

1.3 Approach

Data slice: portion of a class with only a single data member and a set of member functions such that each member function can manipulate the values associated with this data member.

Test one slice at a time, and for each slice, test possible sequences of the methods belonging to that slice.

Code-based approach so potential problems if code is faulty or with identifying legal sequences of methods.

1.4 Generate MaDUM

Minimal Data Member Usage Matrix: $n \times m$ matrix where n is the number of member methods, reports on the usage of data members by methods

Different usage categories are constructors, reporters (read only), transformers, others.

Can create call graph from matrix and identify redundancy where all branches of data member usage.

Matrix accounts for indirect use, through called methods.

1.5 Test Procedure

Classification of methods is used to decide the test steps to be taken:

1. test reporters
2. test constructors
3. test transformers
4. test others

1.5.1 Test Reporters

Test by:

1. Make sure the reporter method does not alter the state of the data member that it reports on
2. Create a random sequence of operations except for reporters of the data slice under test and append it with the reporter method

1.5.2 Test Constructors

Test that

- All data members are correctly initialized
- All data members are initialized in correct order

Test by:

1. Run the constructor and append the reporter methods for each data member
2. Verify if in correct initial state (state invariant)

1.5.3 Test Transformers

Test interactions between methods.

For each data slice:

1. Instantiate object under test with constructor (already tested)
2. Create sequences
3. Append the reporter methods (already tested)

For each member function, execute all control flow paths where data slice is manipulated.

1.5.4 Test Others

Do not change the state, do not report any data member, so should be tested as any other functions.

1.6 Discussion

Slicing may not be helpful for some classes. Too many sequences or impossible sequences are potential problems.

2 Testing Derived Classes

Derived classes may add facilities or modify the ones provided by the base class.

Two extreme options for testing a derived class:

- flatten the derived class and retest all slices of the base class in the new context
- only test the new/redefined slices of derived class

2.1 Bashir and Goel's Strategy

Extend the MaDUM of the base class to generate MaDUM for the derived class:

- take the MaDUM of the base class
- add a row for each newly defined or re-defined data member of the derived class
- add a column for each newly defined of re-defined member function of the derived class

2.2 Procedure

Local attributes: similar to base class testing

Retest inherited attributes:

- if they are directly or indirectly accessed by a new or re-defined method of the derived class
- check the upper right quadrant of the derived MaDUM
- check which interited attributes mandate retesting
- once these inherited attributes are identified, the test procedure is similar to slice testing in the base class but uses inherited and redefined methods

An inherited data member needs to be retested if the number of entries in its MaDUM row has increased between the base and derived MaDUMs.