Carving Differential Unit Test Cases from System Test Cases

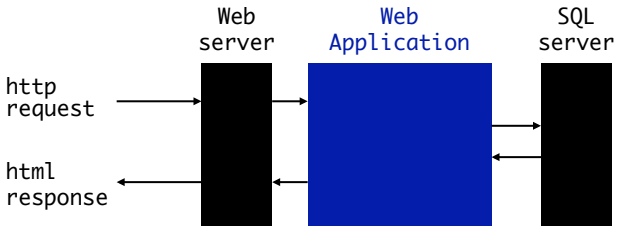
Sebastian Elbaum, Hui Nee Chin, Matthew B. Dwyer, Jonathan Dokulil

ESQuaReD

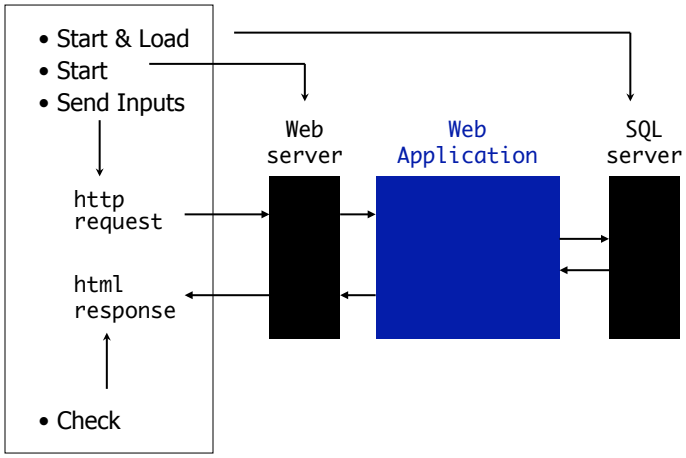
University of Nebraska - Lincoln, USA

This work was partially supported by the U.S. National Science Foundation and IBM

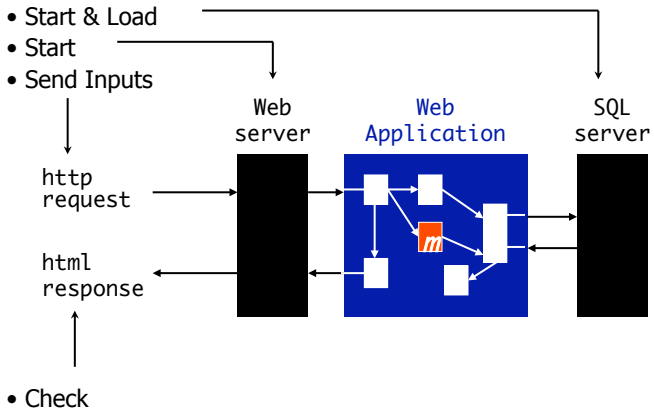
Web Application



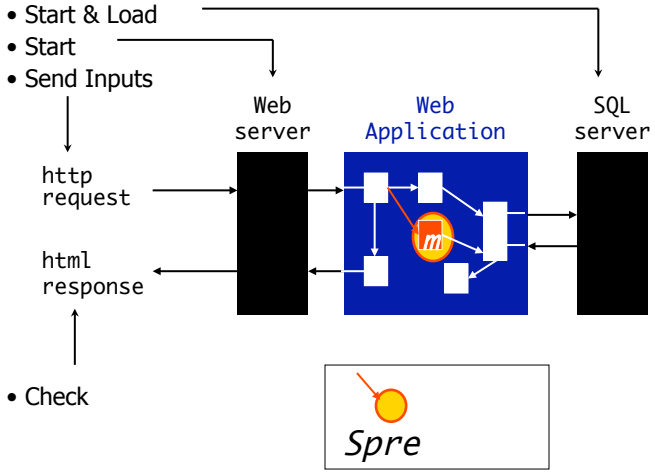
System test *St*



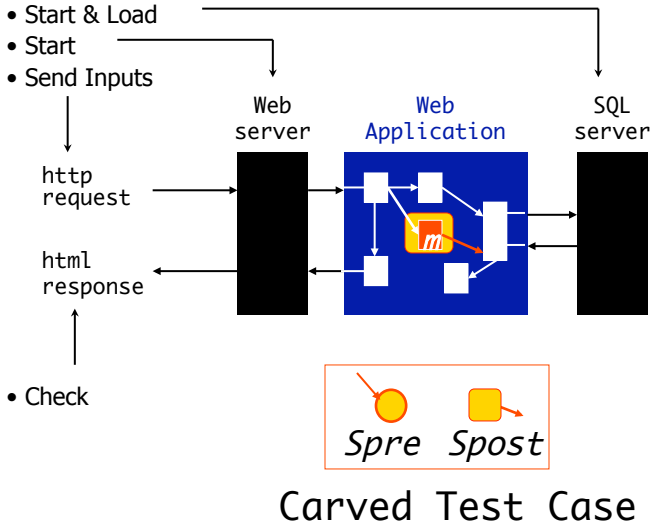
Carving for *m* as exercised by *St*



Carve program state *S* before executing *m*

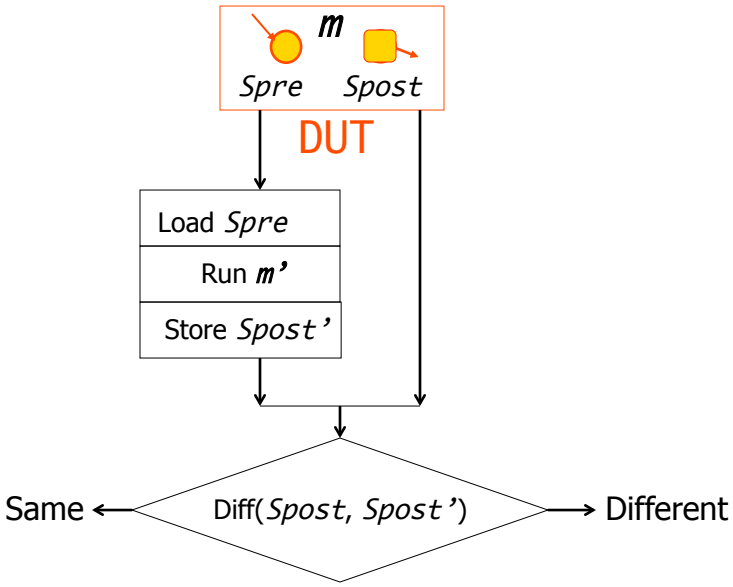


Carve program state *S* after executing *m*



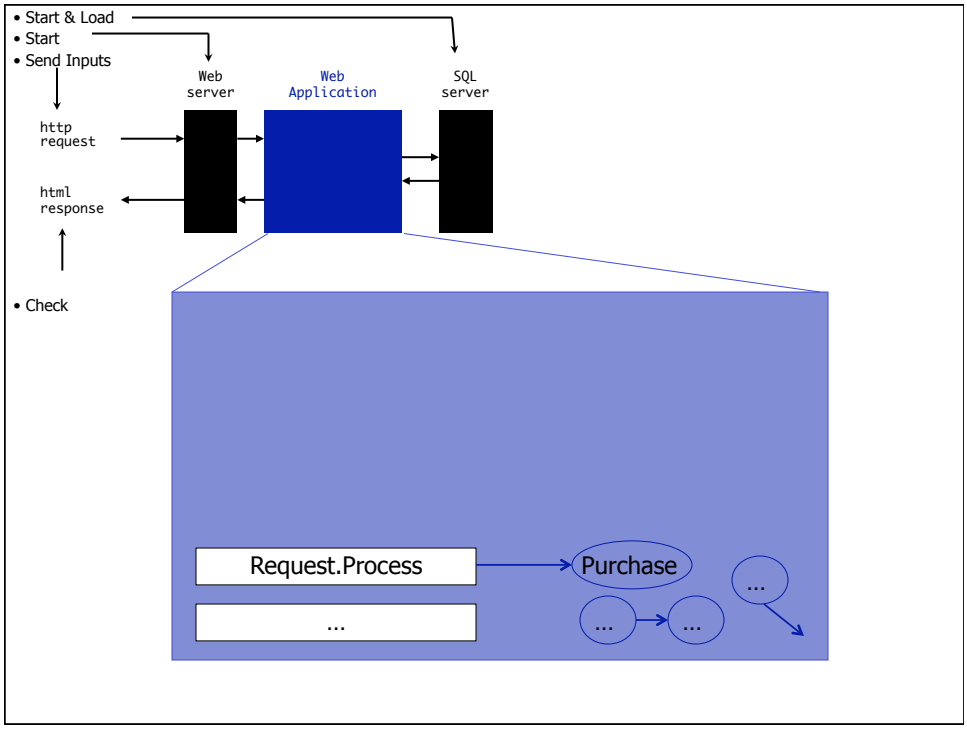
Carved Test Case

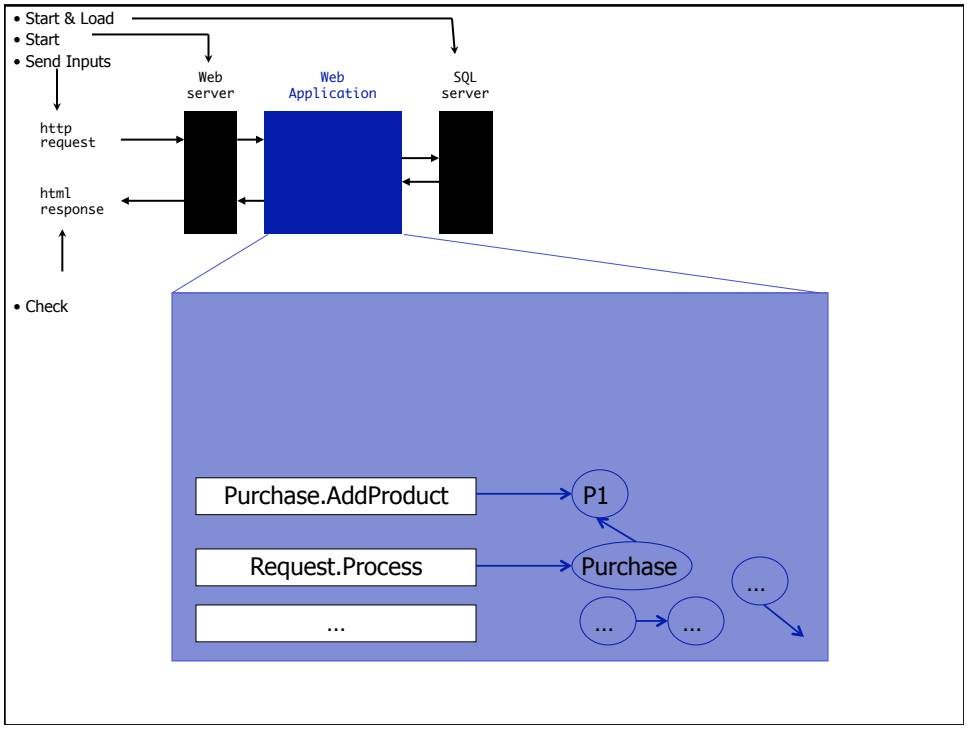
Scenario: *m* evolved to *m'*

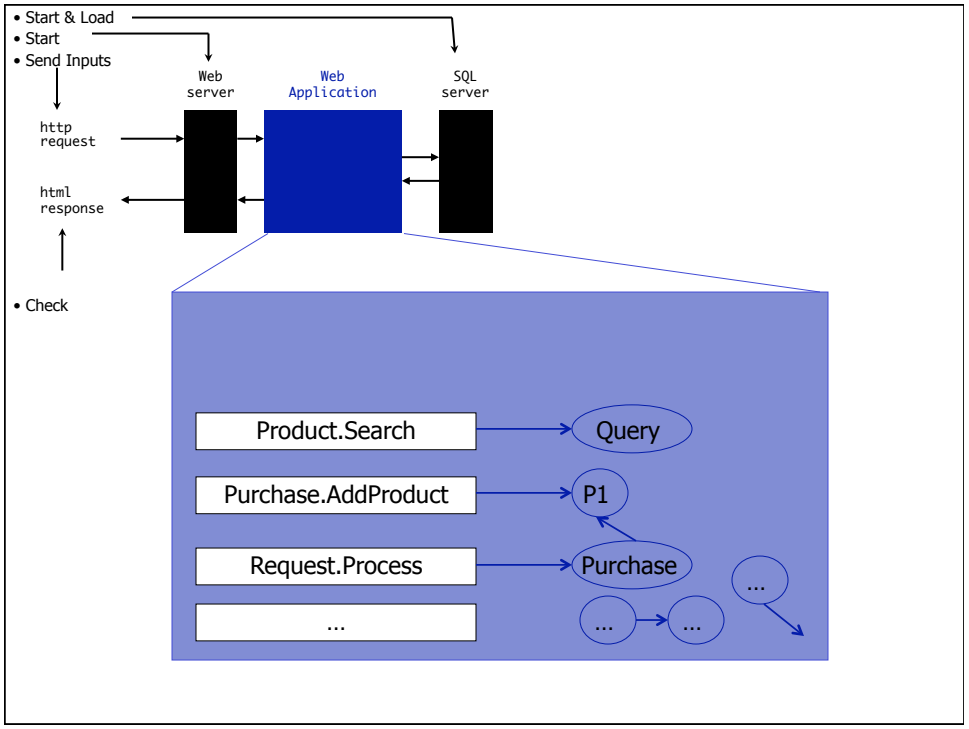


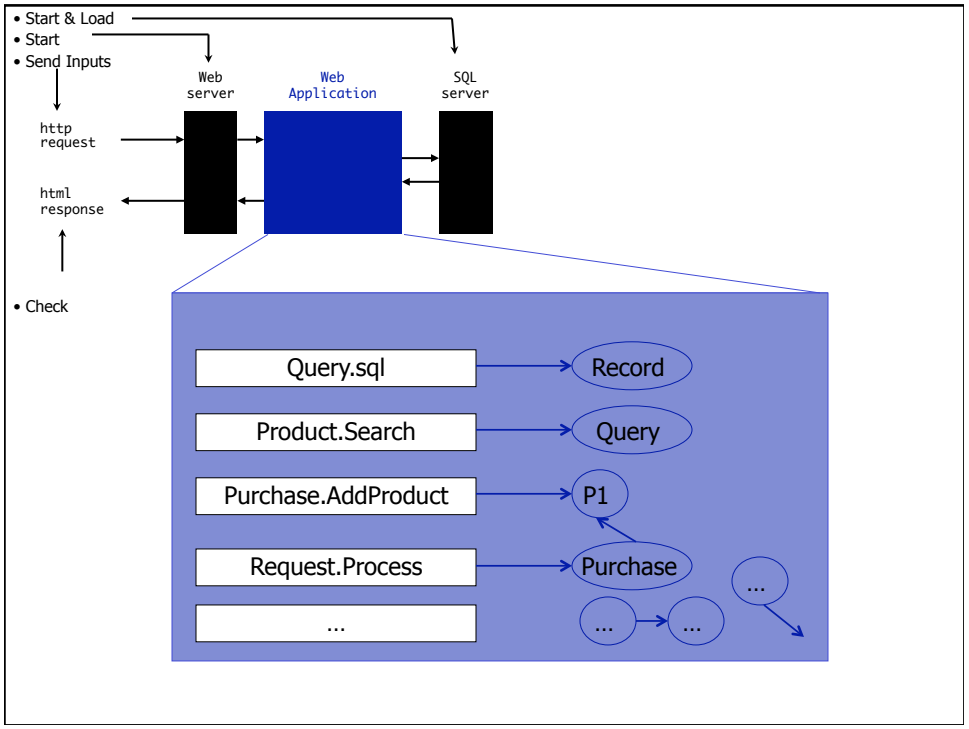
3 Reasons to Carve DUTs

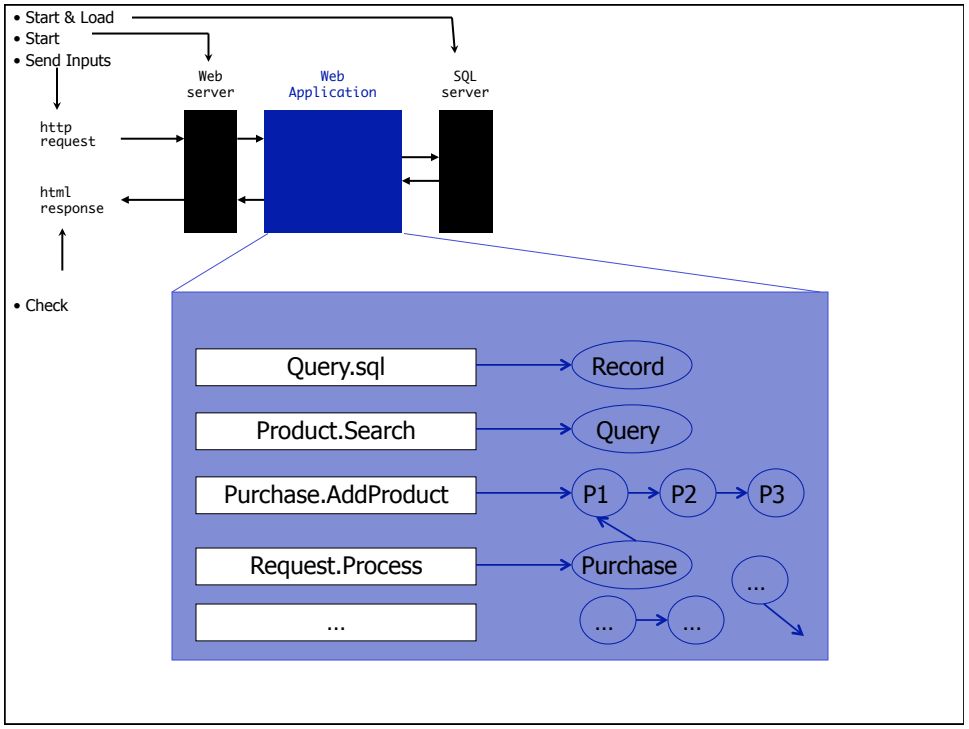
- Lack good unit test suite
- Have strong system suite
  - Evolved beyond unit test suite
  - Exercises interesting interactions
- Have inefficient system suite
  - Too cumbersome for fault isolation
  - Too coarse for regression testing
  - Too expensive to set up and execute

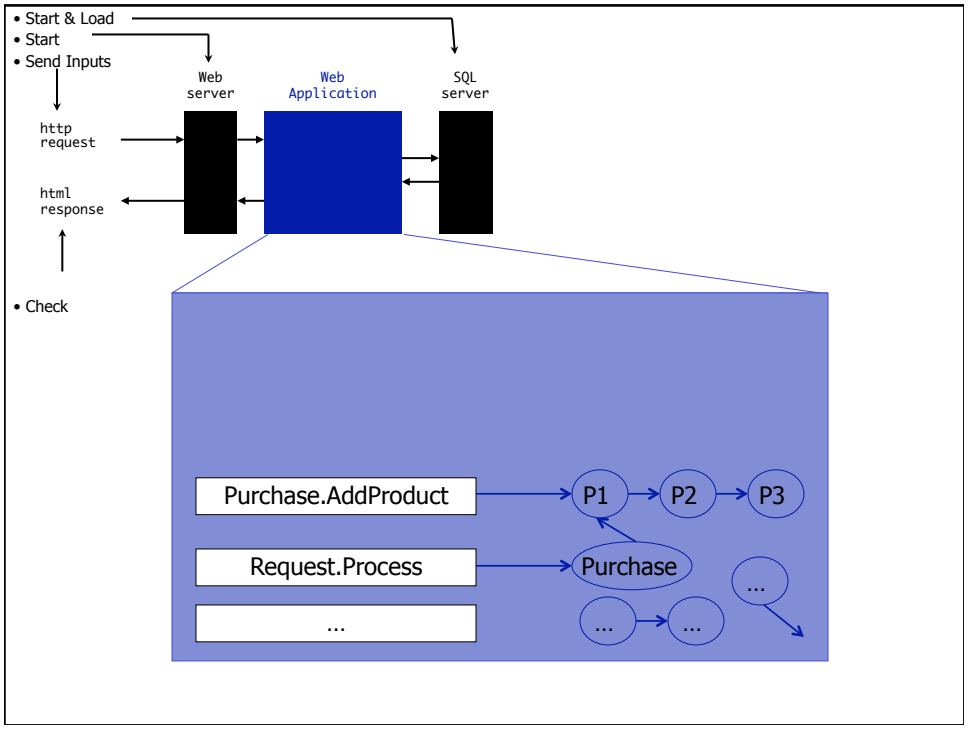

• Start & Load
• Start
• Send Inputs
Web server
Web Application
SQL server
http request
html response
• Check
Request.Process
Purchase
...
...
...
...

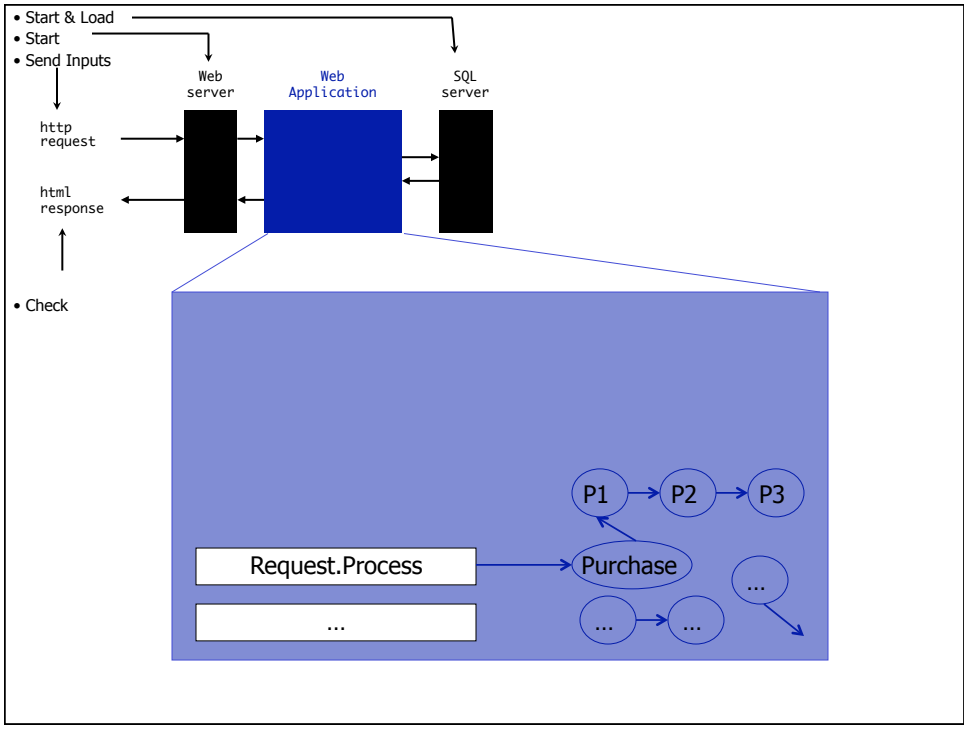

• Start & Load
• Start
• Send Inputs
Web server
Web Application
SQL server
http request
html response
• Check
Purchase.AddProduct
P1
Request.Process
Purchase
...
...
...
...

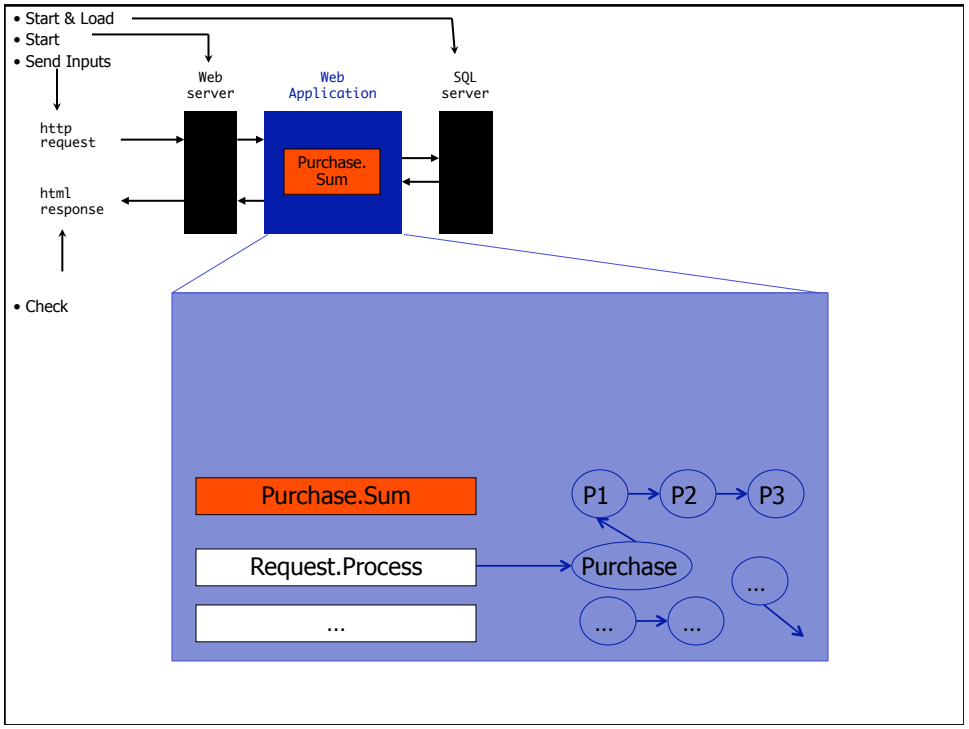

• Start & Load
• Start
• Send Inputs
Web server
Web Application
SQL server
http request
html response
• Check
Product.Search
Query
Purchase.AddProduct
P1
Request.Process
Purchase
...
...
...
...

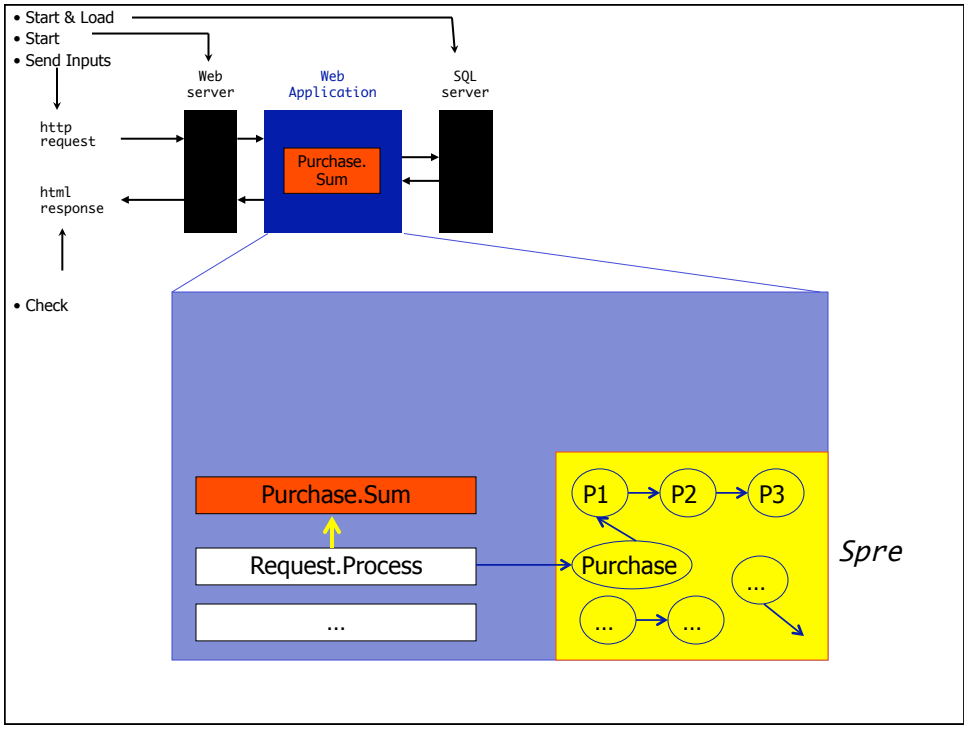

• Start & Load
• Start
• Send Inputs
Web server
Web Application
SQL server
http request
html response
• Check
Query.sql
Record
Product.Search
Query
Purchase.AddProduct
P1
Request.Process
Purchase
...
...
...
...


• Start & Load
• Start
• Send Inputs
Web server
Web Application
SQL server
http request
html response
• Check
Query.sql
Record
Product.Search
Query
Purchase.AddProduct
P1
P2
P3
Request.Process
Purchase
...


• Start & Load
• Start
• Send Inputs
Web server
Web Application
SQL server
http request
html response
• Check
Purchase.AddProduct
P1
P2
P3
Request.Process
Purchase
...


• Start & Load
• Start
• Send Inputs
Web server
Web Application
SQL server
http request
html response
• Check
P1
P2
P3
Request.Process
Purchase
...
...
...
...


• Start & Load
• Start
• Send Inputs
Web server
Web Application
SQL server
http request
html response
Purchase. Sum
• Check
Purchase.Sum
P1
P2
P3
Request.Process
Purchase
...
...
...
...

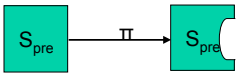


Projections

- Insight: not all parts of *Spre* are necessary!
- Projections: preserve selected parts of *Spre*
  - K-bounded reachable projections
  - Interface reachable projections
  - May-reference objects in the heap
  - Trace-based projections
  - …

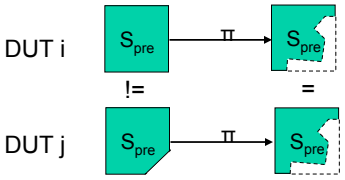
Using Projections

- Reduce size of DUTs

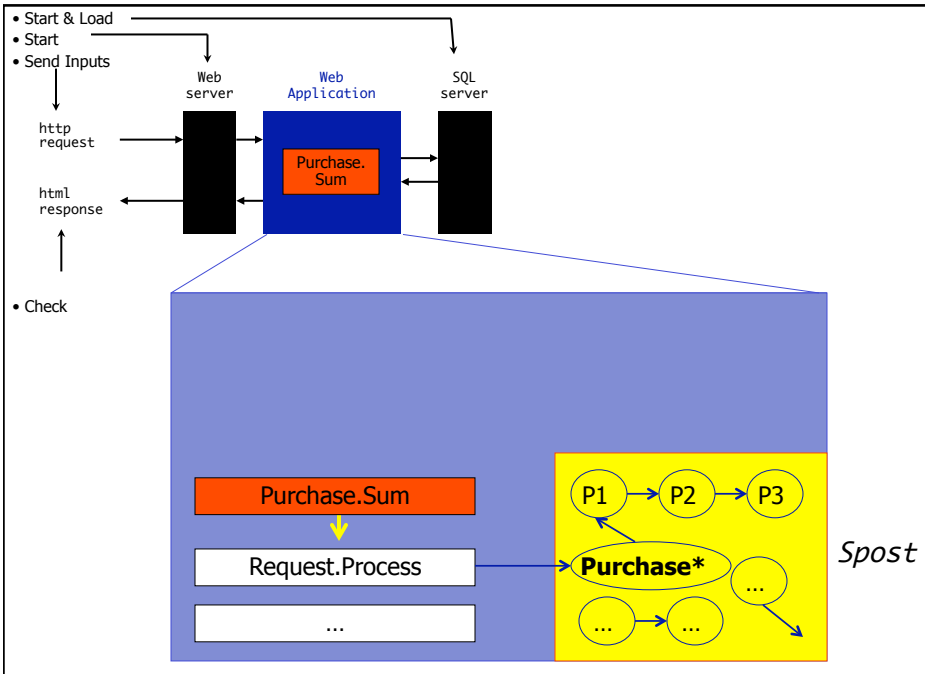


Store relevant *Spre*

- Reduce number of DUTs

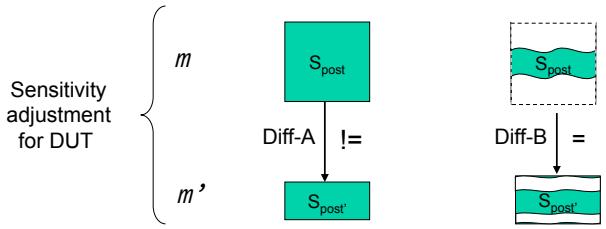


Keep i or j

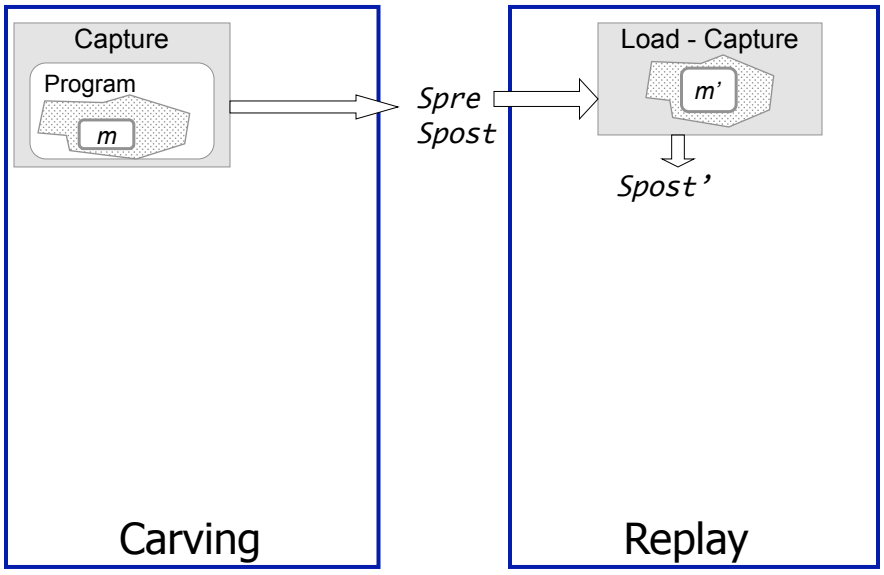


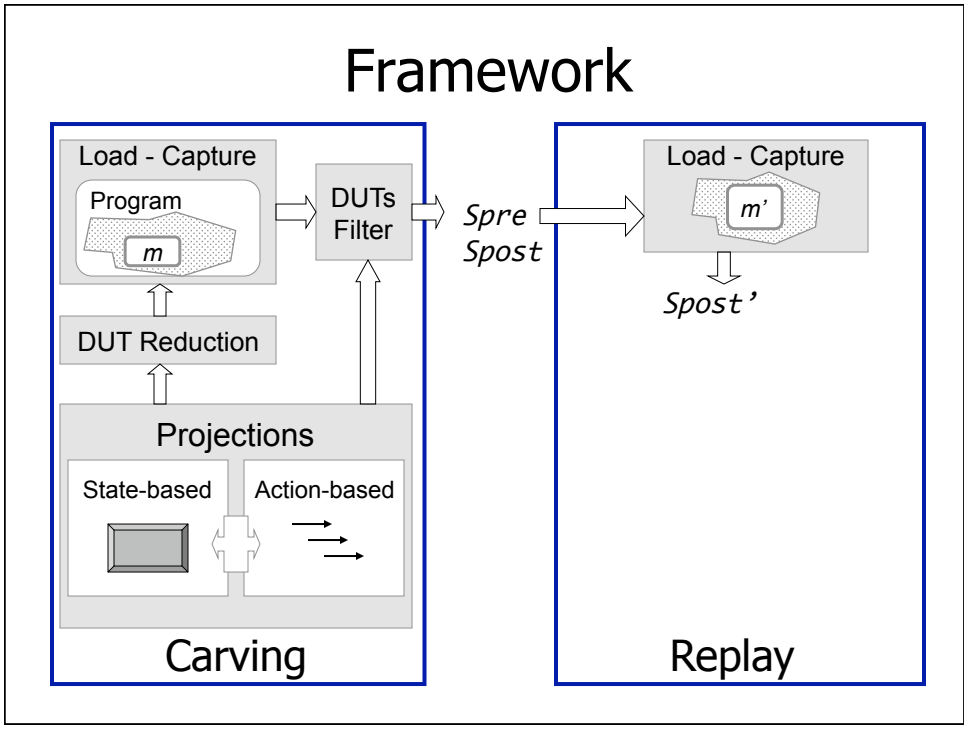
Sensitivity Adjustment through *diff* Functions

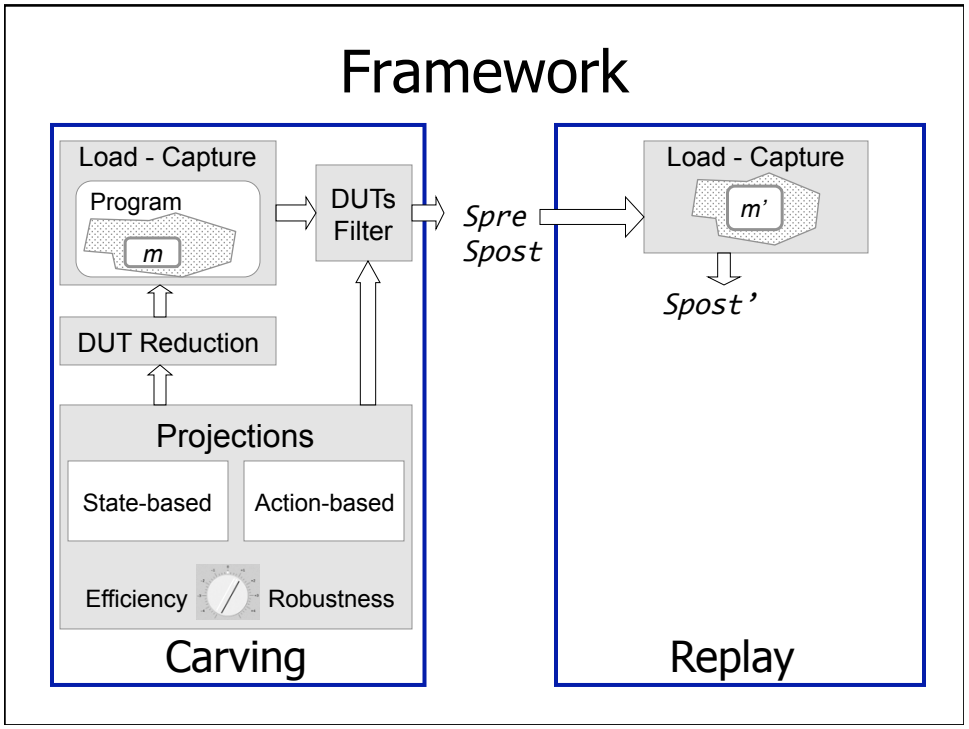
- Insight: not all *Spost* differences are relevant
- Differencing functions on *Spost* and *Spost'*
  - Return values or 'this' object instance
  - Reachable projections
  - Spectra
  - Heap shape

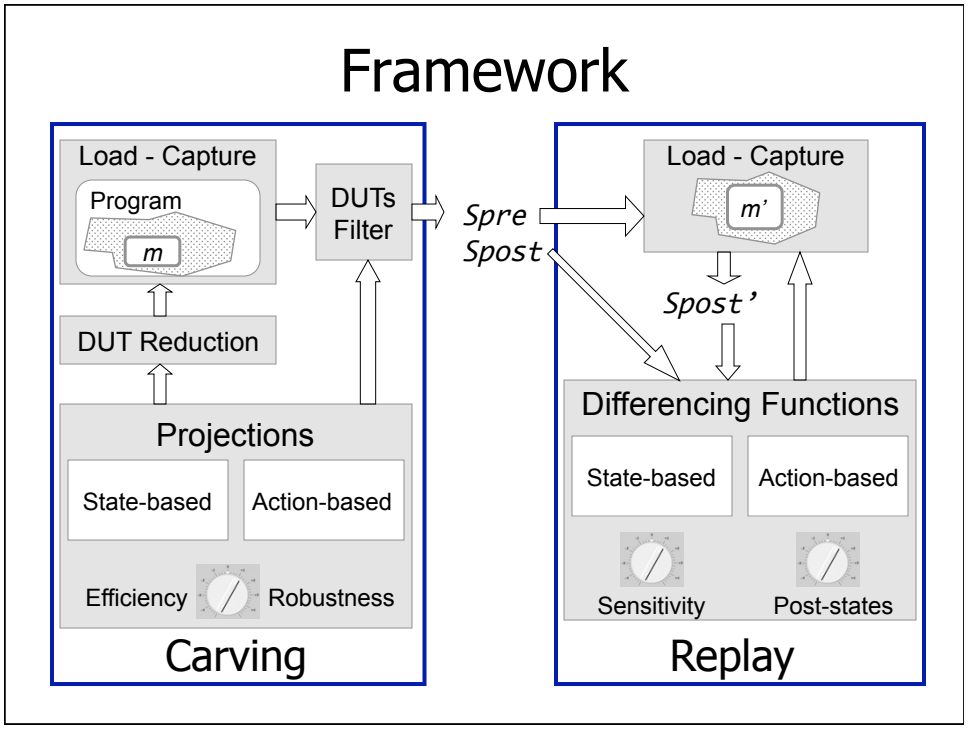


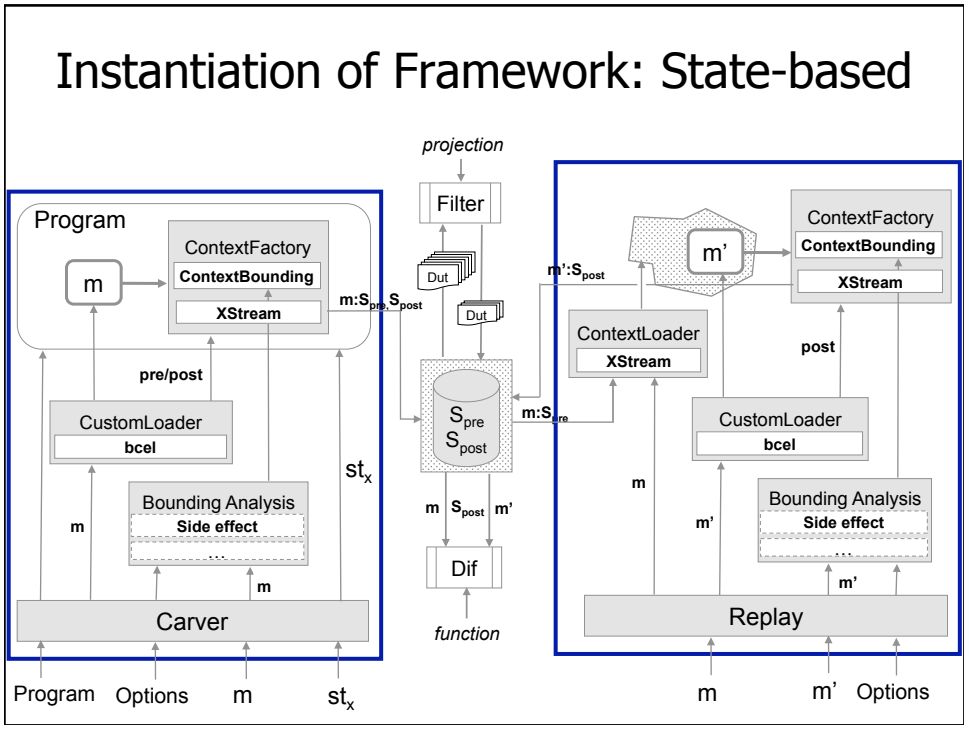
Framework



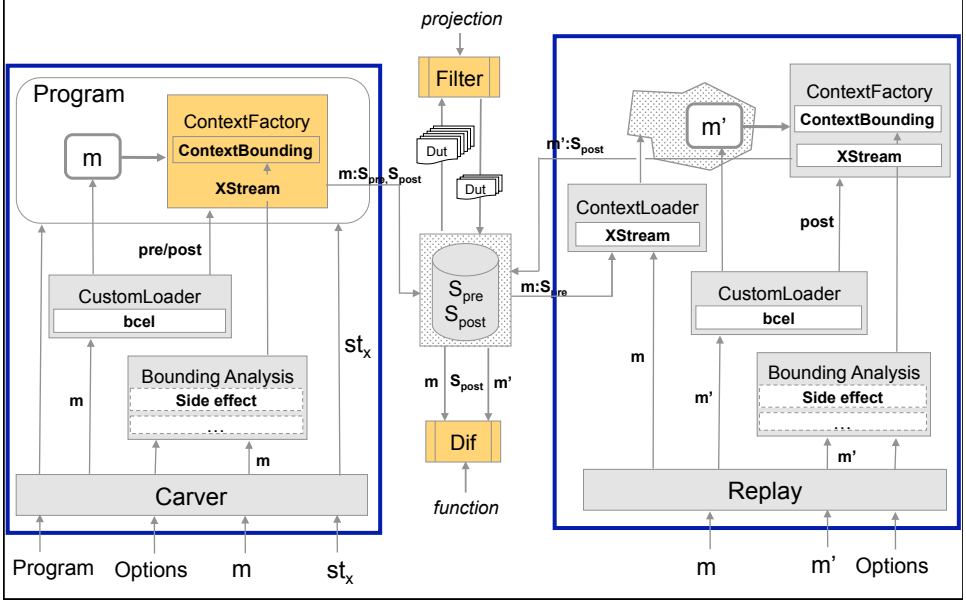

Framework
Load - Capture
Program
m
DUTs
Filter
DUT Reduction
Projections
State-based
Action-based
Carving
Spre
Spost
Load - Capture
m'
Spost'
Replay


Framework
Load - Capture
Program
m
DUTs
Filter
DUT Reduction
Projections
State-based
Action-based
Efficiency
Robustness
Carving
Spre
Spost
Load - Capture
m'
Spost'
Replay


Framework
Load - Capture
Program
m
DUTs
Filter
DUT Reduction
Projections
State-based
Action-based
Efficiency
Robustness
Carving
Spre
Spost
Load - Capture
m'
Spost'
Differencing Functions
State-based
Action-based
Sensitivity
Post-states
Replay


Instantiation of Framework: State-based
projection
Filter
Program
m
ContextFactory
ContextBounding
XStream
m:Spre,Spost
pre/post
CustomLoader
bcel
Bounding Analysis
Side effect
...
stx
m
Carver
Program
Options
m
stx
Dut
Dut
Spre
Spost
m Spost m'
Dif
function
m':Spost
m'
ContextFactory
ContextBounding
XStream
ContextLoader
XStream
post
m:Spre
CustomLoader
bcel
m
m'
Bounding Analysis
Side effect
...
m'
Replay
m
m'
Options

Instantiation of Framework: State-based



Study

- Goal: compare *St* versus DUTs
- Questions: cost, fault detection, robustness
- Context: regression testing
- Artifact: Sienna -- http://sir.unl.edu
  - Multiple versions
  - Seeded faults
  - System test suite

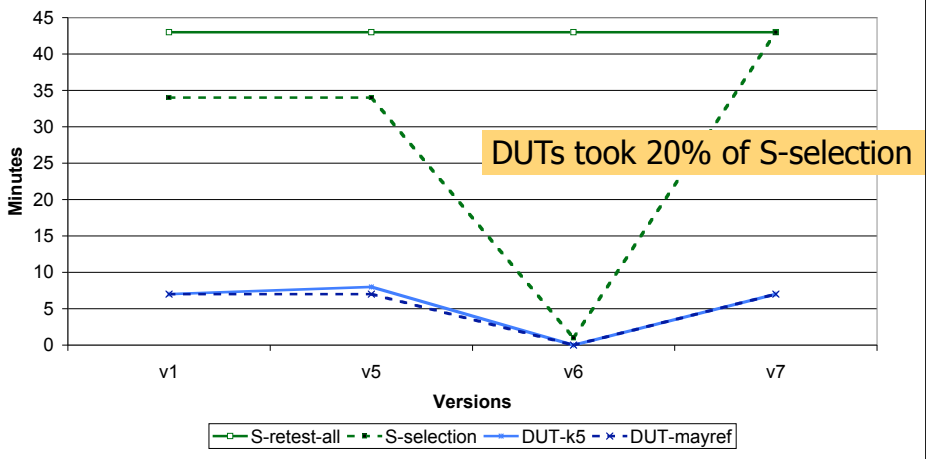
Study Setup and Design

1. Generate DUTs
   - Carved DUTs on *V0*
2. Selected tests per version
   - Selected tests exercising changes components in *Vi*
3. Run/replay test suites
   - Fault free versions of Siena -- oracle
   - Faulty versions of Siena
4. Compare outcome of test suites
   - Execution time on fault free versions
   - Fault detection

Results: Carving Efficiency

- Fully automated carving
  +100 methods, +500 system tests
  - Generated over 20K DUTs
  - Carved complete program state: ~160 minutes, 2GB
- Observations
  - Simple filters were extremely effective
  - Heap structure greatly influences effectiveness of projections to reduce *Spre*
  - Compression reduced space requirements by 2 orders of magnitude

Results: Replay Efficiency



Results: Effectiveness

- *DUT* suite detected all faults found by *Sts*
- An instance: passing *Sts* had failing *DUTs*
  - *DUTs* detected a fault not fount by corresponding *St*
  - Error did not propagate to system level
- Multiple instances: failing *Sts* without failing *DUT*
  - DUTs were not replayable
  - Multitude of DUTs compensated for limitations of individual ones

Distilled Findings

- Feasibility and Efficiency
  - Carving is fully automated - no tester participation
  - +5 times faster to execute than system tests
- Effectiveness
  - DUTs suites detected faults found by system tests
    - Exceptions: lower k-depth, structural/interface changes
  - DUTs detected differences missed by system tests
- Sensitivity
  - DUTs detected differences corresponding to legal changes
  - Differencing functions mitigated false positives

Future Work

- Synergy between DUTs/Junit/*St* tests
- Selective re-carving
- DUTs, Compound-DUTs, From-To-DUTs
- Reading and using DUTs