Software Architecture-based Regression Testing

Henry Muccini
Software Engineering and Architecture Group
Università degli Studi dell'Aquila
I-67100 L'Aquila, Italy
muccini@di.univaq.it

Marcio Dias
Department of Computer Science and
e-Science Research Institute,
University of Durham, UK

Debra J. Richardson
Donald Bren School of Information
and Computer Sciences,
University of California Irvine, USA
My main research areas: **Analysis**

» SEA Group
  - Software Engineering and Architecture Group

» **Software Architecture Analysis:**
  - Model-Checking SAs (the CHARMY framework)
  - SA-based Testing
  - SA-based **Regression** Testing (the SARTE project)
  - Model-checking driven Testing (the ModTest approach)

» **Product Line:**
  - Modeling Product Line Architecture
  - Testing and Model Checking of Product Line
Our Experience on SA-based analysis

validate the SA model conformance with respect to selected functional properties

Charmy Project [www.di.univaq.it/charmy]

provide confidence on the implementation fulfillment to its architectural spec

SA-based Testing
ModTest: Model-Checking driven Testing

SA conformance to requirements through MC

- Requirements
  - drive
  - identify
  - SA model
  - functional properties
  - M.C.
  - refine SA
  - OK [test case selection]

Software Architecture

System Implementation

- Test spec
- Test Procedures
- Test Exec

Implementation conformance to SA through Testing

- Fault removal
- OK

Test spec

Test Case Implementation

Test Case Execution

System Implementation

Requirements Specification

Architectural Specification

Charmy Analysis

Critical sub-systems identification

SA Revision

Test Case Selection

Implementation Revision

System Release
Our Recent experience in SA-based Analysis

» **Industrial Experience**

- PSTDA Italy [ICSE00,ICSE01]
- Telcordia
- Marconi [FME 03]
- Siemens [ITM 04]

» **Academic Experience**

- [FASE 04][IEEE TSE04]
- [CBSE 05][COMPSAC05]
Considerations

» What happens if the (architectural) model changes?
  
- Usually, we need to remake analysis from scratch
How changes affect ModTest

SA conformance to requirements through MC

requirements

Software Architecture

M.C.

Test spec

Implementation conformance to SA through Testing

System Implementation

Test Procedures

Implementation conformance to SA through Testing

Fault removal

Test execution

Test implementation

Test Case selection

SA spec

[OK] refine SA

[OK] test case selection

[NOK]

M.C.

[OK]

functional properties

identify

drive

[OK]

SA model

© 2005 by H. Muccini, WADS 2005
SA-based Regression Testing [WADS05] [COMPSAC05]

Software Architecture SA1 (version 1)
- Component A
- Component B
- Component C

Software Architecture SA2 (version 2)
- Component A
- Component B
- Component C'
- Component Y

Code1 (version 1)
Code2 (version 2)

SA evolution
Test Reuse

Code evolution
Test Reuse

Test Reuse
Traditional Regression Testing

» Test modified software to provide a certain confidence that no new errors are introduced into previously tested code.

» Two key phases:

  i) testing the program P with respect to a specified test suite T, and

  ii) when a new version P' is released, regression testing of the modified version P' versus a test suite T'

» Selective RT:

  - Goal: selecting T' as a “relevant” subset of T

    > t1 in T is included in T' if there is the potential that it could produce different results on P' than it did on P
First phase: SA-based Code Testing

- The code conformance to the SA has been already tested

**Software Architecture “S”**

**Code “P”**

- Identify relevant SA-level Test Cases

Steps 0 - 2

© 2005 by H. Muccini, WADS 2005
SArTe - Goal 1 (code evolution)

Test Conformance of a Modified Implementation $P'$ to the initial SA

- **Context:**
  - $P$ correctly implements the SA $S$
  - $P'$ modifies $P$: some objects are modified, and/or some new objects are introduced.

- **Goal:** Test the conformance of $P'$ with respect to $S$,
  - while reusing previous test information for selective regression testing, thereby reducing the test cases that must be retested.

- To handle **Architectural Drift**
Test Conformance of an Evolved Software Architecture

- **Context:**
  - P correctly implements the SA S
  - S’ modifies S, adding or removing components
  - A modified implementation P' may have been also developed.

- **Goal:** Test the conformance of P’ with respect to S',
  - while reusing previous test information for selective RT, thereby reducing the test cases that must be retested.
Goal 1: P changes

SA-based Code Testing

Step 0
SA-spec for S

Step 1
Testing Criterion

Step 2
Extract SA-level Test Cases (ATCs)

Step 3
Map ATCs into Code-level Test Cases as a way to select T for P

Step 4
Run T over P and Evaluate Results

SA-based Regression Testing - Goal 1 -

Step A
Build the GP graph for P

Step B
Build the GP' graph for P'

Step C
Compare GP with GP'

Step D
Create a Test History

Step E
Select T' for P'
Considerations

» **Differences** with respect to traditional code-based selective RT techniques:

- code-level test cases are always selected starting from a well formalized architectural specification.

- the oracle in SA-based RT is the software architecture specification itself.

» **Advantages:**

- as in traditional RT, we reduce the size of the test suite for P', eliminating all those tests which do not need to be reapplied to P', and

- when conformance faults are detected, we can gather information on how to adjust the initial architecture.
Goal 2: SA changes

SA-based Regression Testing - Goal 2 -

- Step a: SA-spec for S''
- Step b: Testing Criterion
- Step c: Compare S and S''
- Step d: Select ATCs'' from S that need to be retested in S''
- Step e: Map ATCs'' into Code-level Test Cases as a way to select T'' for P''
- Step f: Run T'' over P'' and Evaluate Results

SA-based Code Testing

- Step 0: SA-spec for S
- Step 1: Testing Criterion
- Step 2: Extract SA-level Test Cases (ATCs)
- Step 3: Map ATCs into Code-level Test Cases as a way to select T for P
- Step 4: Run T over P and Evaluate Results
Goal 2 Idea

» Compare the two architectural specifications to identify changed/unchanged portions of the SA.

- Both structural and behavioral changes are taken into account
  
  > We compare the topology changes (if the SA structure changed)
  > We compare the behavioral changes (if the SA behavior changed)

- and, in a fashion similar to traditional code-based RT,
  
  > ATC needs to be re-run in S'' if it traverses a path modified when moving from S to S''
Experiment 1: the Elevator SA

Elevator System Version 1

- ElevatorADT1
- ElevatorPanel1
- BuildingPanel

Elevator System Version 2

- ElevatorADT1
- ElevatorPanel1
- ElevatorADT2
- ElevatorPanel2
- Scheduler
- BuildingPanel
Experiment 2: the Cargo Router System

**Cargo Router, Version 1**

- Port (P)
- Vehicle (V)
- Warehouse (W)
- Clock (C)
- NextShipment (NS)

**Cargo Router, Version 2**

- Port (P)
- Vehicle (V)
- Warehouse (W)
- Clock (C)
- Planner (P)

Note: In Version 2, the connection (*) is replaced with the connections (**)
Future Work

» To reconstruct the actual architecture when the first goal determines that the code no longer conforms to the initial SA

» Regression Testing of Component-based Software Architectures

» Regression-based Analysis of ModTest

» Apply/refine this approach into real systems (SiemensC.N.X., Terma GmbH)
Contact Information

» **Henry Muccini**
  - Dipartimento di Informatica, Universita' dell'Aquila, L'Aquila, Italy
  - mucini@di.univaq.it

» **Marcio Dias**
  - Department of Computer Science and e-Science Research Institute, University of Durham, UK
  - marcio.dias@dur.ac.uk

» **Debra J. Richardson**
  - Donald Bren School of Information and Computer Sciences, University of California Irvine, USA
  - djr@ics.uci.edu
SA diff

ATC1 in T  ATC2 in T  ATC1 in T’