Regression Test

1. What is Regression test?
2. Z/Object-Z/ TCOZ model/Test Chart
3. TCOZ Technique (**TC obsolete& augmentation)**
4. TcozRts: implication
5. Usage/advantage/disadvantage

Reference:

Regression Testing of Classes based on TCOZ Specification10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'05)

Introduction of Regression Test

- **Software maintenance activity-guarantee changed parts behaving as intended**

- **How ?Reusing TS of original version.**
  All of them? Select proper subset.
  TS my be obsolete to modified version

- **Solution: techniques strictly code-based TC's**
  Procedure languages
  OO languages
  Fault on changed specification? Only detected by specification-based TC's
  New TC's involved for changed software system

  Exploit formal specs with solid mathematical bases to generate new TC's

- **Technique: Classes based on Timed Communication Object-Z (TCOZ) Specification**

Terms

- Z: The formal specification notation Z (pronounced "zed")
  - Describing computer-based systems
  - Based on Zermelo-Fraenkel set theory
  - Developed by the Programming Research Group (PRG) at the Oxford University Computing Laboratory (OUCL)
- Object-Z: an object-oriented extension of the formal specification language Z. It has been developed by a team of researchers at the Software Verification Research Centre, University of Queensland.

- TCOZ: Modelling notation based on Object-Z & Timed CSP(Communicating Sequential Process)
  Provides a timed, multi-threaded object modelling notation for the design of complex system.
  Object-Z : modelling complex data& algorithms
  Timed CSP: medelling process control& real time interaction

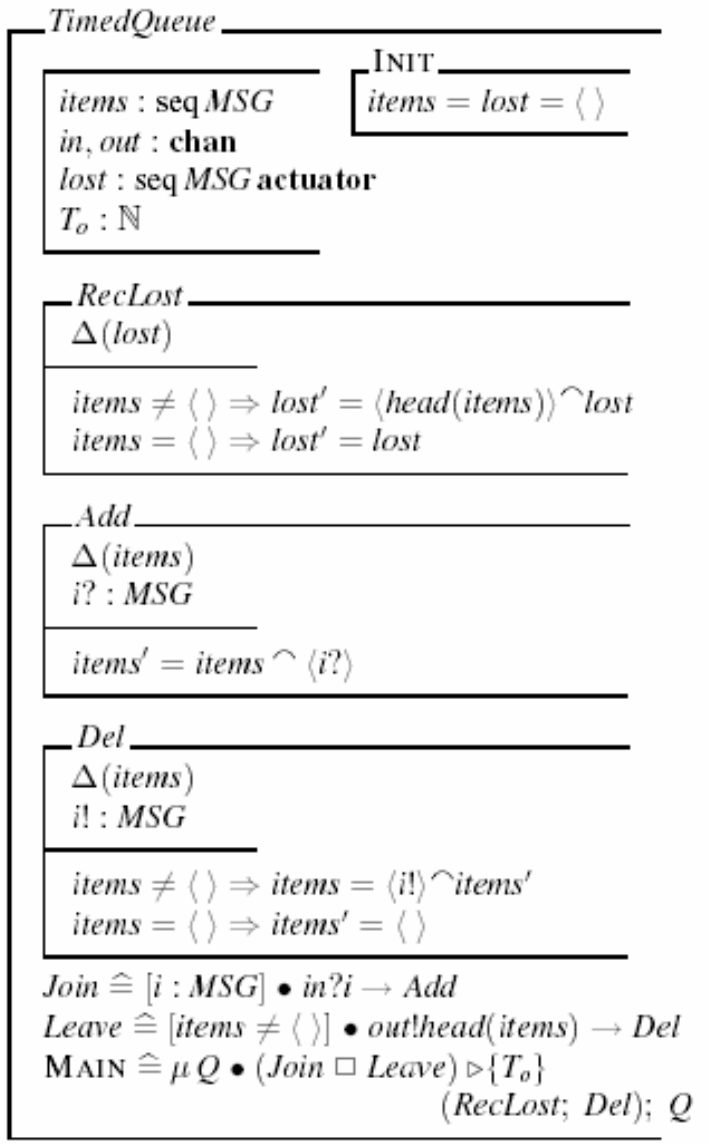
TcozRts (regression test system)

- Strength & link between TCOZ and Timed Automata: Explored by J. S. Dong, S. C. Qin, and J. Sun. Generating MSCs from an integrated formal speci.cation language. *on Software Maintenance*, pages 14–25, 1994.
- Generating System test requirement from TCOZ model and approaches by using XML/XSL to translate TCOZ into UML by J. Sun, J. S. Dong, J. Liu, and H. Wang. A XML/XSL Approach to Visualize and Animate TCOZ.
- This presentation focus on TCs selection & TSs augmentation based on TCOZ model of classes specification
  - Identify the obsolete TCs
  - Select the TCs realted to changed specification
  - Guide to generate new TCs
- TcozRts regression test system

Classes in TCOZ

- TCOZ: a blending of object-Z(operation schemas) with Timed CSP (Terminating) preserving them as a proper sub-language of the blended notation.
- Object-Z class mechanism
  - Operation schema identified with CSP (terminating) that only update state
  - Active class identified with no terminating CSP process
  - MAIN process determines the behaviors of the objects after it initialized.
- Object-Z example

  Timed Message Queue System class

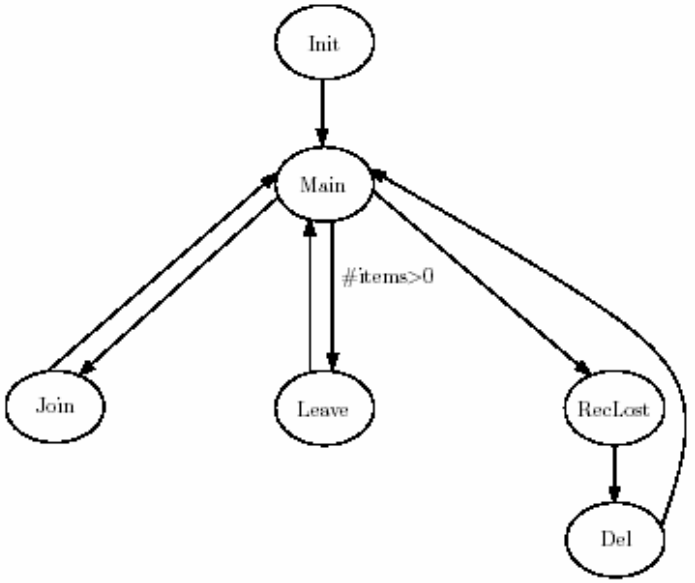


Representation Tcoz Class spec

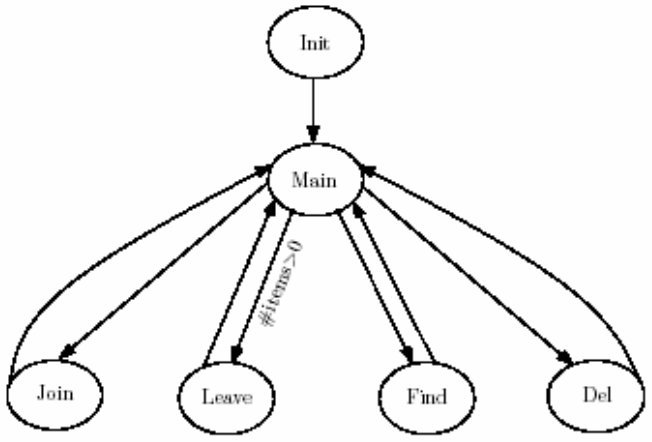
- Develop Test chart
  - Base Modification to spec of class
  - Obsolete TCs
  - Generate new TCs

3-Tuple(S,T,s0)

- Operation S finite, non empty set of vertices
- Invocations T⊆SXS directed edges
- s0 ∈S initial operation of the class



(a) Original Version



(b) Modified Version

Coverage Criteria

- Vertices-methods
- Edges- all the interactions between methods
- Using this interaction coverage criteria to guide RT TC selection and TC generation in our TCOZ technique
- A set TS satisfied the criteria iff $V_{(a,b)\in T}$ at least one test sequence $t \in TS$

TCOZ-based Regression Testing technique

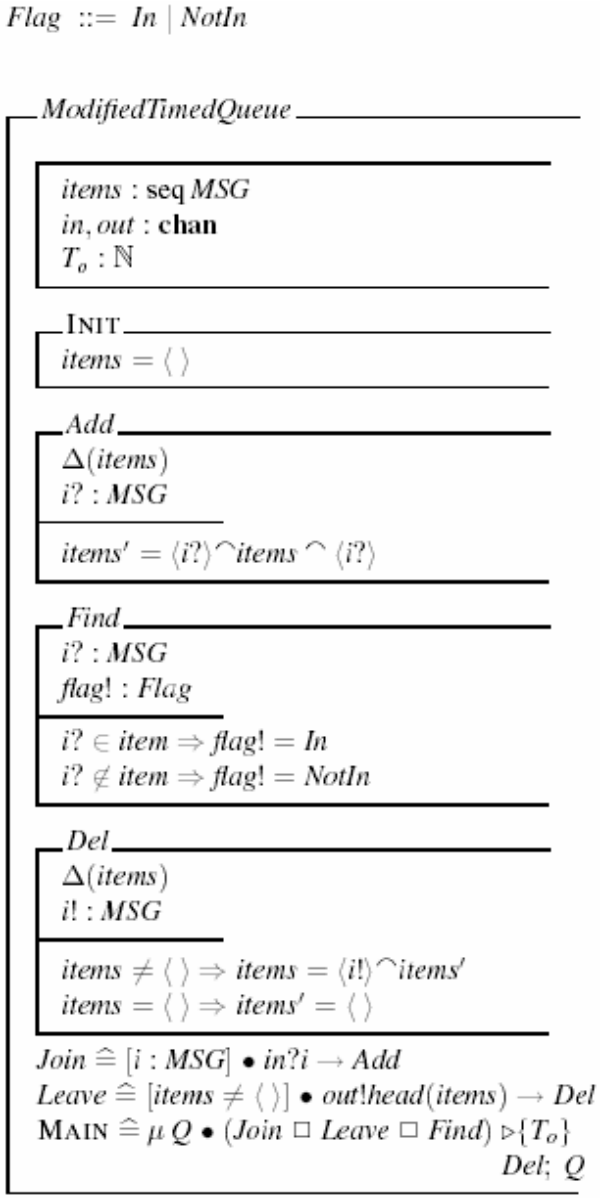
1) Parses original& new version of spec to identify modified operations
2) Constructs test charts original& new version
3) Identify Add/delete operations and interaction
4) Identify obsolete TC based on 1) and 3)
   Selects TC related to changed spec
5) Generate new TC new operations and interactions that added to the spec

Modification to the Tcoz Spec

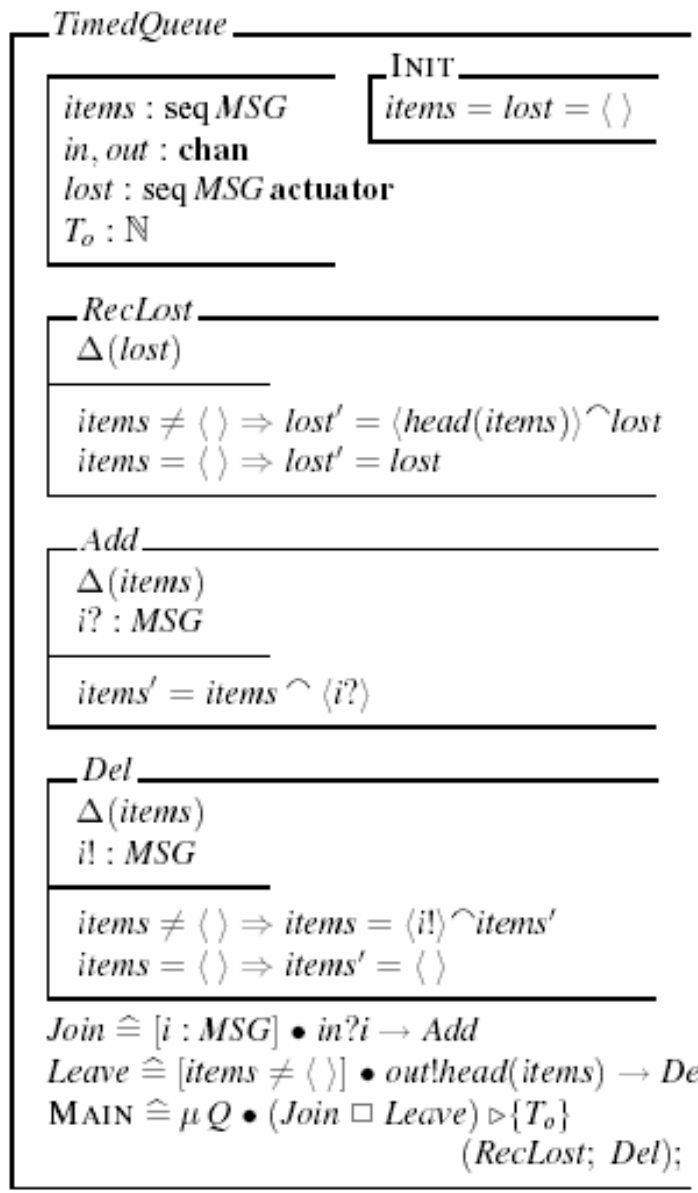
- Modified Operations-class attributes in TCOZ declared in state-schema
  - Added/delete attributes→ access
  - Modified attributes→ Scope, type, or visibility.
  - CSP→ precondition and or post conditions
- Added/Deleted operation
  - operations-added← $V(TestChart_m)$- $V(TestChart_0)$
  - operations-deleted← $V(TestChart_0)$- $V(TestChart_m)$
- Added/Deleted interaction
  - interaction-added← $E(TestChart_m)$- $E(TestChart_0)$
  - interaction-deleted← $E(TestChart_0)$- $E(TestChart_m)$

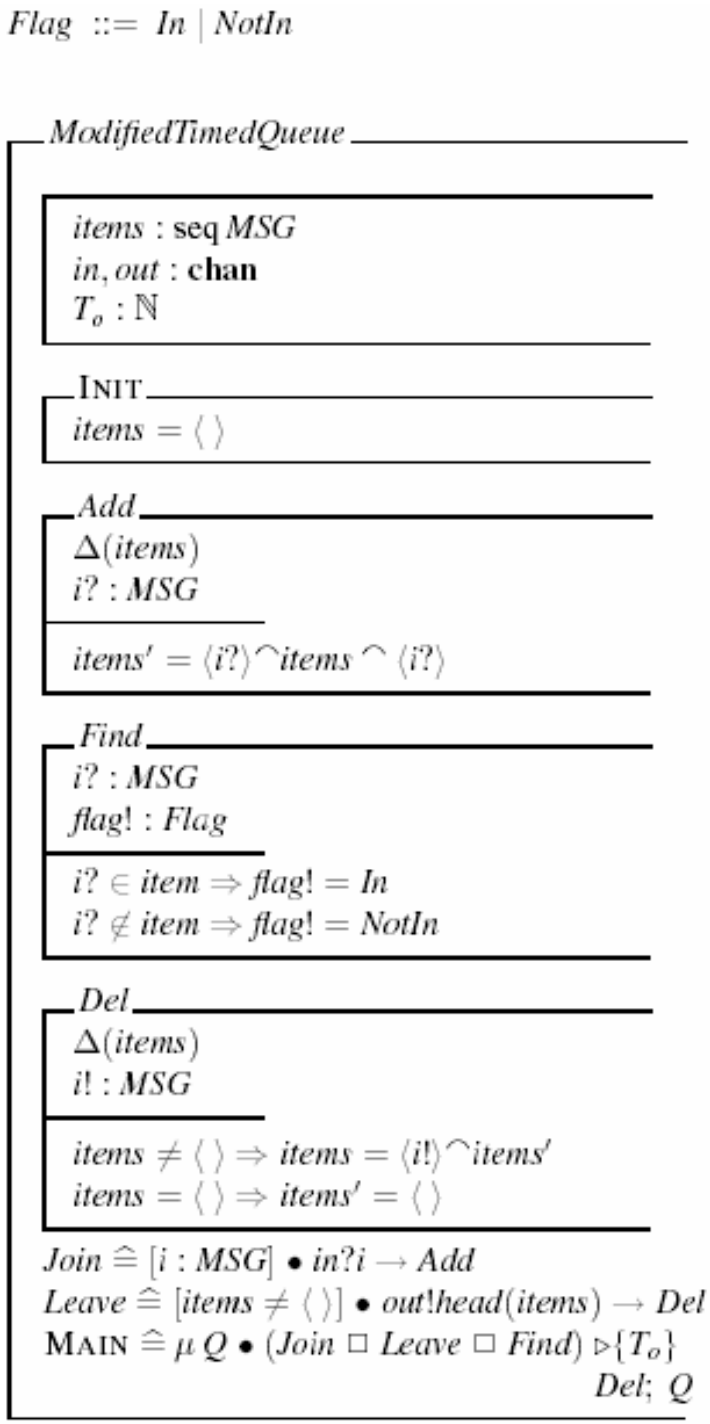
Impact of Modification on Test Case

- Obsolete: invalid sequence of operations in the modified version
- Retest able: Remains valid in the modified version
- Reusable: valid and consist of operations that has remained unchanged in the modified version.



An Example


TimedQueue
items : seq MSG
in, out : chan
lost : seq MSG actuator
$T_o : \mathbb{N}$
INIT
items = lost = ⟨ ⟩
RecLost
Δ(lost)
items ≠ ⟨ ⟩ ⇒ lost′ = ⟨head(items)⟩⌢lost
items = ⟨ ⟩ ⇒ lost′ = lost
Add
Δ(items)
i? : MSG
items′ = items ⌢ ⟨i?⟩
Del
Δ(items)
i! : MSG
items ≠ ⟨ ⟩ ⇒ items = ⟨i!⟩⌢items′
items = ⟨ ⟩ ⇒ items′ = ⟨ ⟩
Join ≙ [i : MSG] • in?i → Add
Leave ≙ [items ≠ ⟨ ⟩] • out!head(items) → Del
MAIN ≙ μ Q • (Join □ Leave) ▷{T_o}
(RecLost; Del); Q


Flag ::= In | NotIn
ModifiedTimedQueue
items : seq MSG
in, out : chan
$T_o : \mathbb{N}$
INIT
items = ⟨ ⟩
Add
Δ(items)
i? : MSG
items′ = ⟨i?⟩⌢items ⌢ ⟨i?⟩
Find
i? : MSG
flag! : Flag
i? ∈ item ⇒ flag! = In
i? ∉ item ⇒ flag! = NotIn
Del
Δ(items)
i! : MSG
items ≠ ⟨ ⟩ ⇒ items = ⟨i!⟩⌢items′
items = ⟨ ⟩ ⇒ items′ = ⟨ ⟩
Join ≙ [i : MSG] • in?i → Add
Leave ≙ [items ≠ ⟨ ⟩] • out!head(items) → Del
MAIN ≙ μ Q • (Join □ Leave □ Find) ▷{T_o}
Del; Q

An example (cont'd)

- Operation-modified={Add, Join}
- Operation-added={Find}
- Operation-deleted={RecLost}
- Interaction-add ={(Main,Find),(Find,Main),(Main,Delete)}
- Interaction-deleted= {(Main,RecLost),(RecLost, Del)}

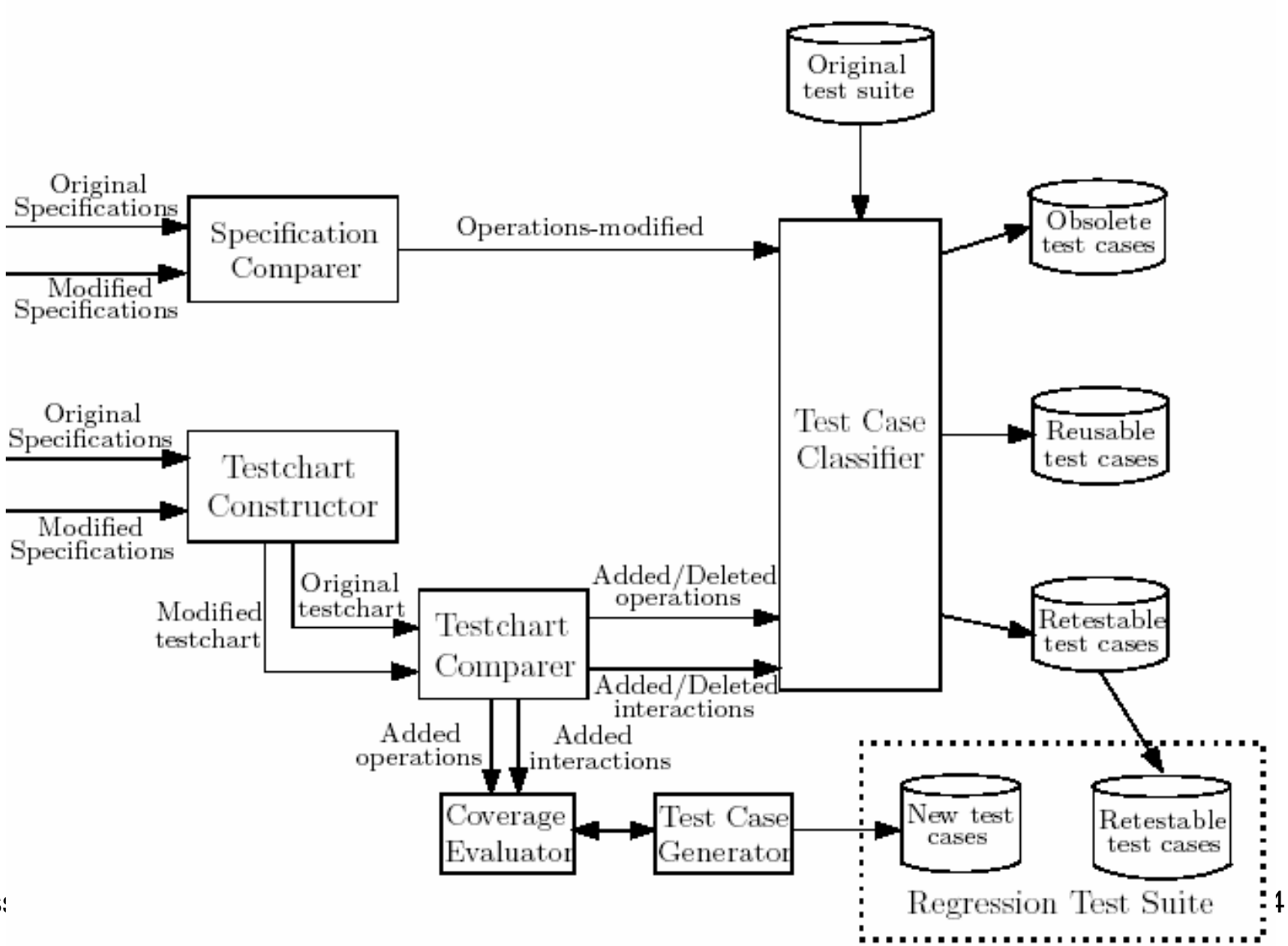
TCOZ Algorithm for Regression Test

- Identify obsolete TS & select TS from the Original TS with two bit vector-OpUsed and IntActrUsed
- Parameters

  TS(original test suite), OpMd, OpDel, IntrActDel returns RTS(sub of TS for Regression), and Obslt

```
1.   Begin
2.     Obslt = φ    RTS = φ
3.     for each (ts ∈ TS) do
4.       get bit vector Operations-Used of ts
5.       for each (O_i ∈ OpDel)do
6.         while (O_i^th bit of
                   Operations-Used == 1) do
7.           Obslt = Obslt ∪ {ts}
8.         endwhile
9.       endfor
10.      get bit vector Interactions-Used of ts
11.      for each (IntrAct_i ∈ IntrActDel) do
12.        while (IntrAct_i^th bit of
                   Interactions-Used == 1) do
13.          Obslt = Obslt ∪ {ts}
14.        endwhile
15.      endfor
16.    endfor
17.    for each (ts ∈ (TS - Obslt)) do
18.      get bit vector Operations-Used of ts
19.      for each (O_i ∈ OpMd) do
20.        while (O_i^th bit of
                   Operations-Used == 1) do
21.          RTS = RTS ∪ {ts}
22.        endwhile
23.      endfor
24.    endfor
25.  End
```

TcozRts (TCOZ based Regression Testing System)



When use?

- TCOZ can be fully automated. It is strictly specification-based;
- no complex static or dynamic code analysis is required
- Independent of the programming language which is used to implement the specification.

Q&A?


Thanks!