Software Testing Methodologies

BITS Pilani

Prashant Joshi

Module 7: Agenda

Module 7: Model Based Testing (1/2)

Topic 7.1	Model Based Testing – Introduction & Overview
Topic 7.2	Finite State Machines & Fault Model
Topic 7.3	Examples
Topic 7.4	Case Study

innovate achieve lead

BITS Pilani
Pilani Campus

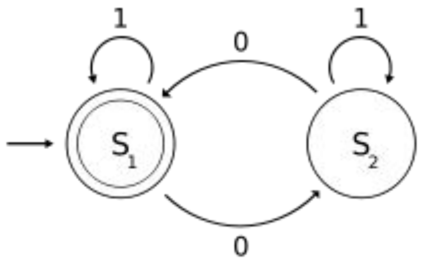
Topic 7.1: Model Based Testing – Introduction & Overview

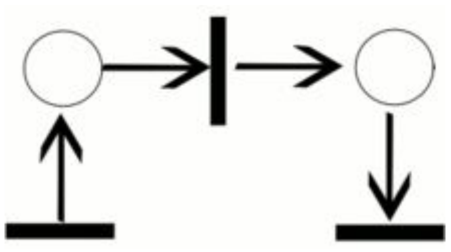
Model Based Testing

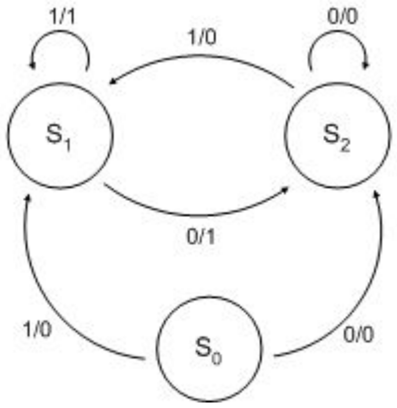
- Process of creating a **Model** results in deeper insights and understanding of the system
- Adequacy of MBT – depends on accuracy of the Model
- Sequence of steps
    - Model the system
    - Identify the threads of system behavior in the model
    - Transform threads into test cases
    - Execute the test cases (on actual system) and record the results
    - Revise the model(s) as needed and repeat the process

Executable Models

- Finite State Machines
- Petri Nets
- StateCharts







What System Is?

- The Components
- Their Functionality
- Interfaces

- All that emphasizes structure

What System Does?

- Decision tables
- State Charts
- Petri nets/EDPN
- FSM/EFSM

- All these describe System Behaviour
- Look for expressive capabilities of the system

Modelling

- What the system is
  - Emphasize structure
  - Components, their functionality and interfaces
  - DFD, Entity/Relation models, hierarchy charts, classes diagrams and class diagrams
- What the system does
  - Emphasize behavior
  - Decision Tables, FSM, State Charts & Petri Nets

Refer: Page 225 and 226 of T1

Model Based Testing Tools

- Modelling the system provides ways to generate test cases automatically
- Example: http://graphwalker.org/index

Software Testing Methodologies

BITS Pilani

Prashant Joshi

innovate achieve lead

BITS Pilani
Pilani Campus

Topic 7.2: Finite State Machine & Fault Model

Finite State Machines

- Method of expression of a design
- Simple way to model state-based behavior
- State Charts – a rich extension of FSM
- Petri nets – useful formalism to express concurrency and timing

State Based Testing

- State “Behaviour” exhibited
- Example: Stack
- Operation pop

```
s.push(5);
y=s.pop();
print(y);

Y=5
```

```
s.push(5);
s.push(7);
y=s.pop();
print(y);

Y=7
```

State Based Testing

- Testing state based components

- State-full
- State-less

- Testing only individual methods or functions for state-full components is not sufficient

State-full Component

- A set of States
- Transition between states

State-full Component

- Objects/Classes
- Control Systems
- Embedded Systems
- Communication Systems
- …

State Based Component

States	Values (of some data)
Empty State	top=0
Full State	top=10
Partial State	1<=top<=9

State Based Modeling Techniques

- State transition diagrams
- Extended Finite State Machines

State Transition Diagram

□ A State

$\longrightarrow$ A transition

$$\begin{tikzcd} \boxed{S1} \arrow[r, "event/action"] & \boxed{S2} \end{tikzcd}$$

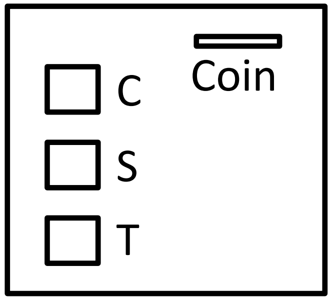
The Fault Model

- Process of Design
- Conforming of the implemented system to the Requirements
- Fault Model defines a set of small set of possible fault types that can occur
- Our focus here is FSM or EFSM (*Lets talk modelling later!*)

Fault Categories

- Operation Error
  - Error generated upon transition
  - Incorrect output function
- Transfer Error
  - Incorrect state transition
- Extra State Error
- Missing State Error

Vending Machine

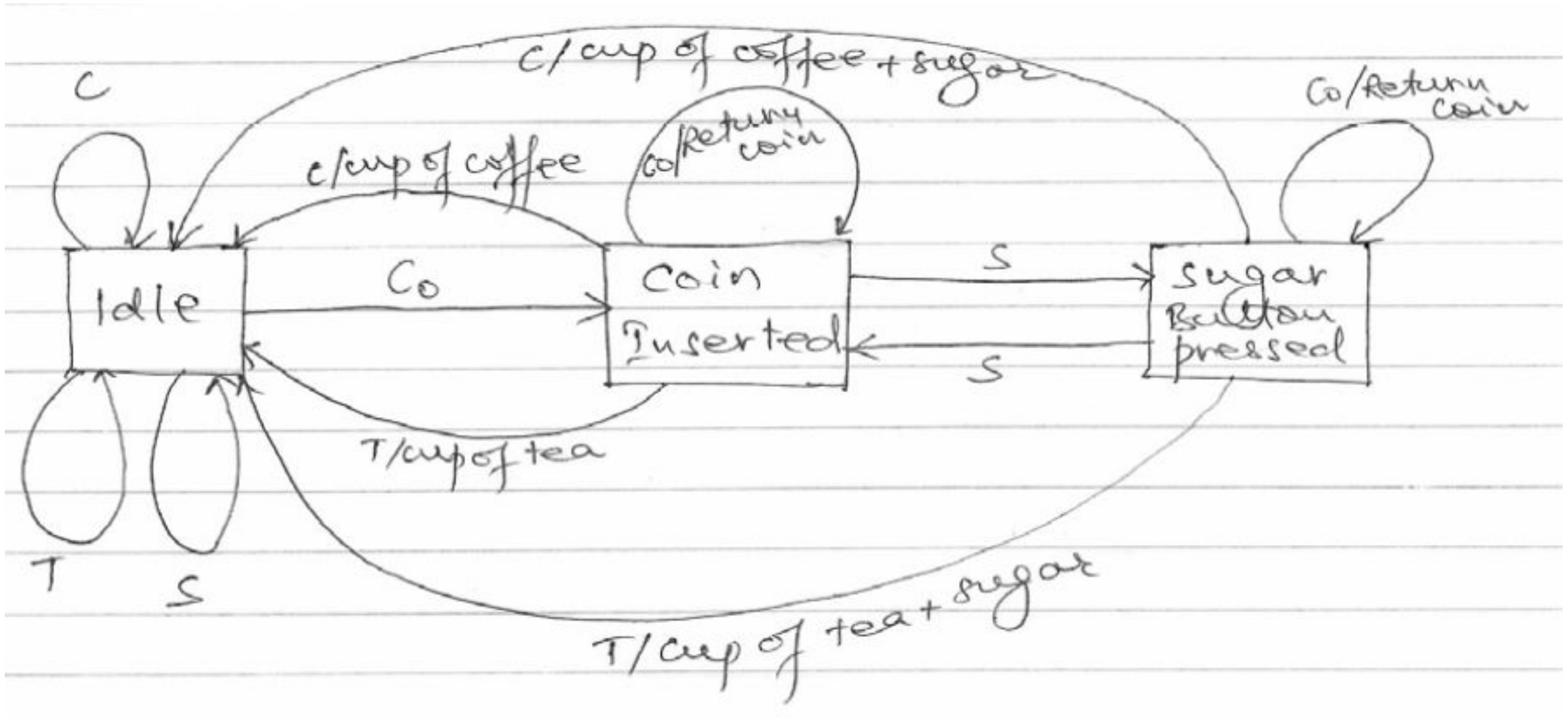


C: Coffee Button pressed

T: Tea Button pressed

S: Sugar Button pressed

Co : Coin inserted



Some examples to discuss

- Garage Door
- Building Lighting Control System
- Lift/Elevator Control System (One or Multiple)
- A MMI (Man-machine interface) Interface of an instrument

Software Testing Methodologies

BITS Pilani

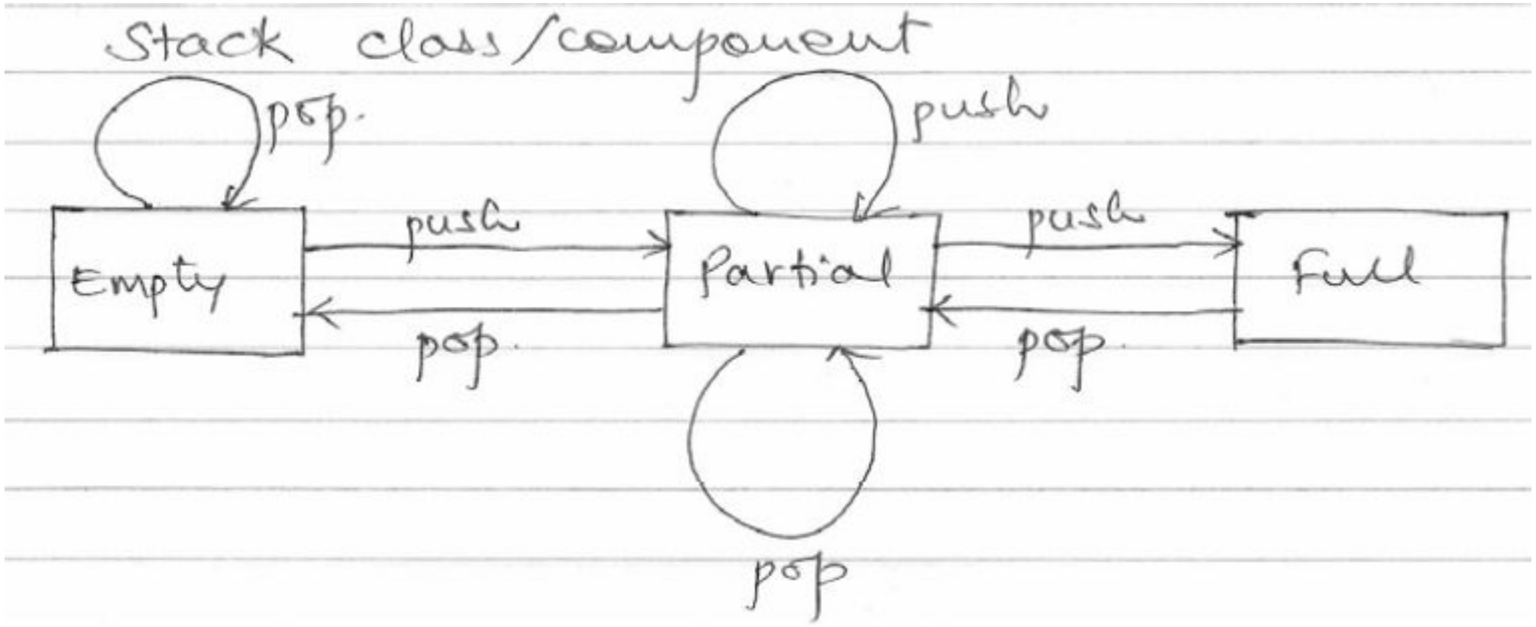
Prashant Joshi

BITS Pilani
Pilani Campus

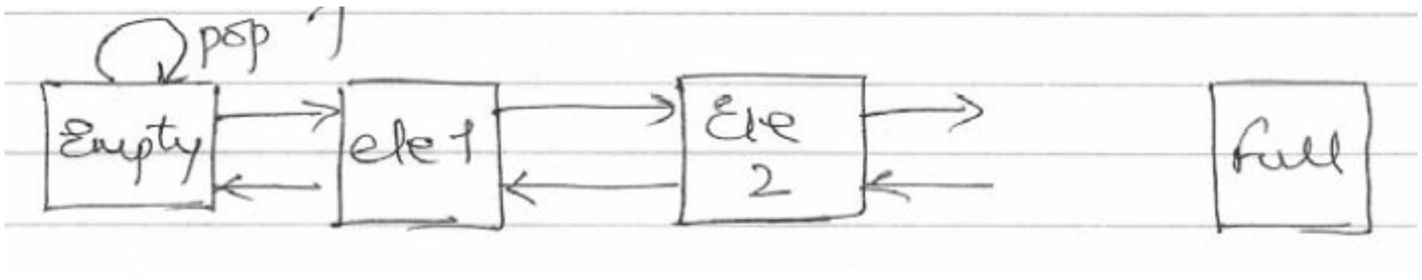
Topic 7.3: Examples

Stack

- Simple stack
- Operations (push and pop)



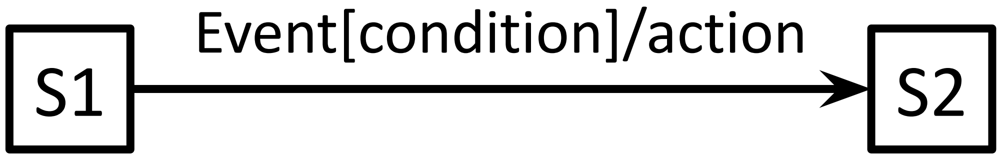
Notion of State Explosion



- Too many states
- State Explosion problem!

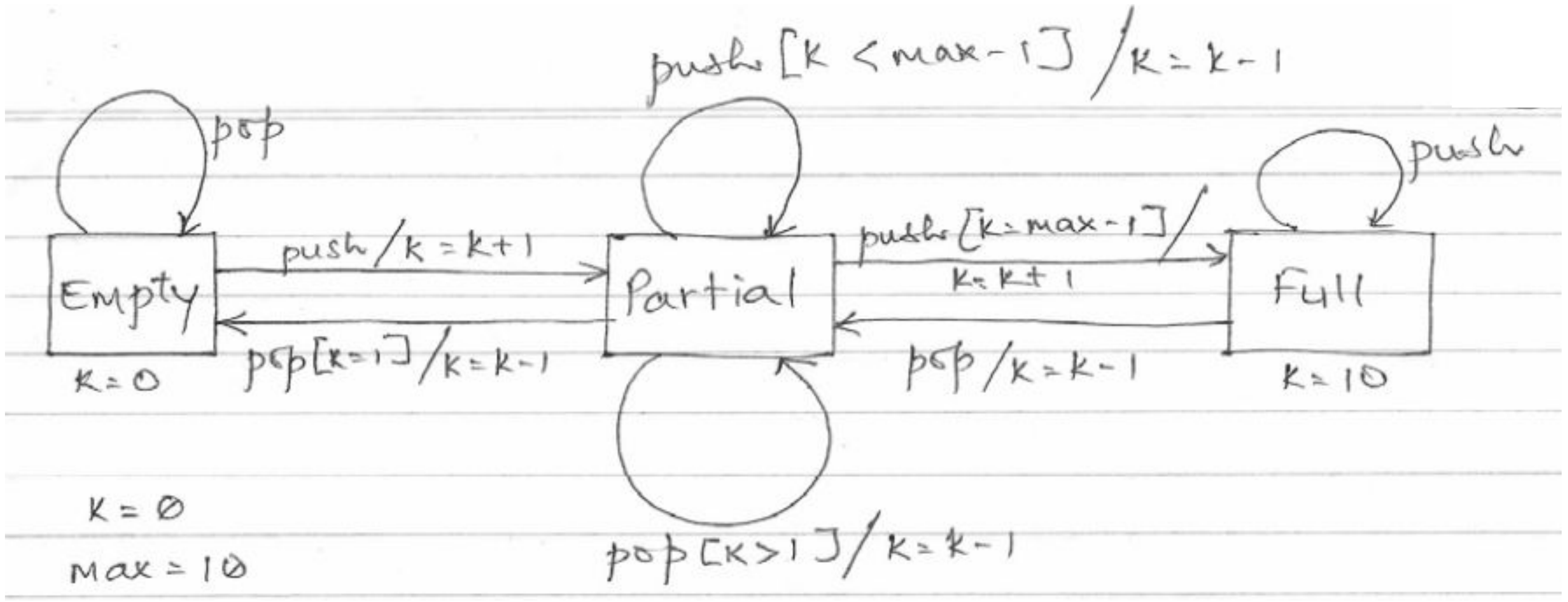
Extended FSM

- Extended Finite State Machine
- Extension of the state transition diagram by introducing
  - Variables
  - Conditions



1. The system is in S1
2. Event occurs
3. Condition evaluates to true
4. Transition from S1 to S2 takes place
5. Action is performed

Stack



push [k < max-1] / k = k-1
pop
push
Empty
push / k = k+1
Partial
push [k = max-1] / k = k+1
Full
pop [k=1] / k=k-1
pop / k = k-1
k = 0
k = 10
k = 0
Max = 10
pop [k>1] / k = k-1

Testing Stack Component

- Operations/methods
  - Push
  - Pop
- State based testing

Testing with Criteria

- State Testing
  - Every state in the model should be visited at least once
- Transition Testing
  - Every transition in the model is “traversed” at least once
- Path Testing
  - Traverse every path in the model at least once

State Coverage

```
Test #1: s.push(5)        //partial state
Test #2: s.push(5)
         s.push(7)        10 push operations

         s.push(20)       //full state
```

- State Coverage Satisfied

Transition Coverage

Test #3: y.pop()

Test #4: s.push(5)
y.pop()

Test #5:
$$\left.\begin{array}{l}\text{s.push(5)}\\ \text{s.push(7)}\\ \\ \text{s.push(20)}\\ \text{s.push(12)}\end{array}\right\}\quad \text{11th push}$$

Transition Coverage

Test #6: s.push(5)
s.push(7)
y=s.pop()

Test #7:

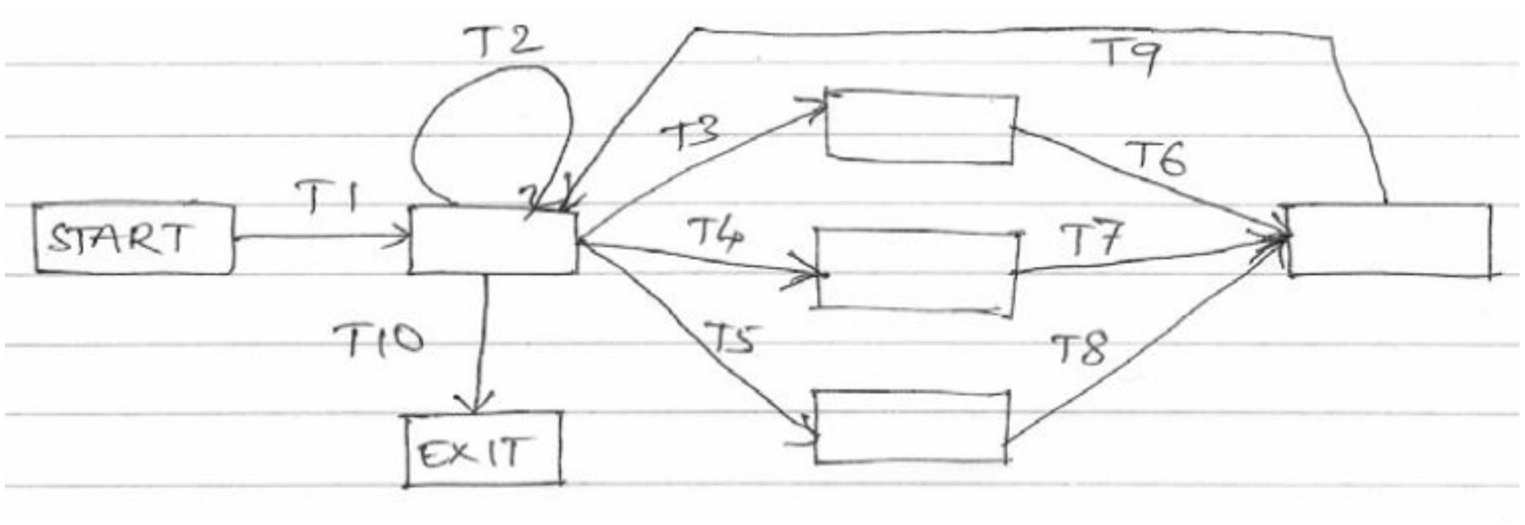
$$\left.\begin{array}{l}\texttt{s.push(5)}\\ \texttt{s.push(7)}\\ \\ \texttt{s.push(17)}\\ \texttt{y=s.pop()}\end{array}\right\}\text{10 push}$$

Constrained Path Testing

- Modified Path Testing
- Traverse every path in the model under the constraint that any transition in the path is traversed at most N times

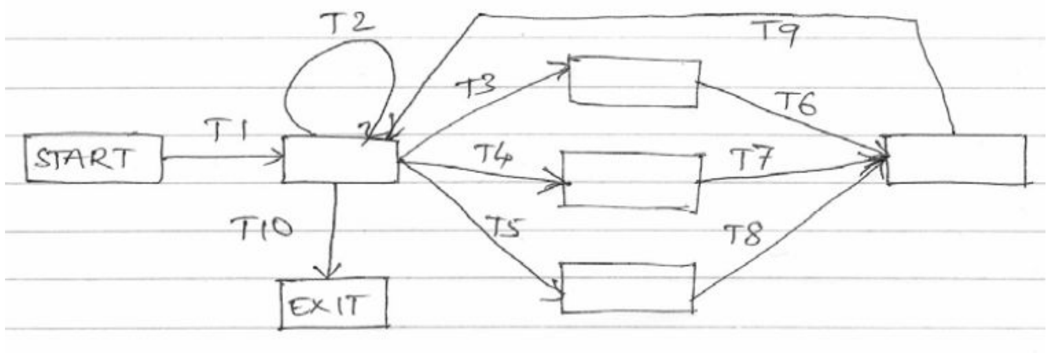
Constrained Path Testing

Use of n=1 (Say repeat only once)



Constrained Path Testing

T1: T1, T10
T2: T1, T2, T10
T3: T1,. T3, T6, T9, T10
T4: T1, T2, T3, T6, T9, T10
T5: T1, T3, T6, T9, T2, T10
T6, T1, T4, T7, T9, T10
T7, T1, T2, T4, T7, T9, T10
T8: T1, T4, T7, T9, T2, T10
T9: T1, T5, T8, T9, T10
T10: T1, T2, T5, T8, T9, T10
T11: T1, T5, T8, T9, T2, T10



State Based Testing

- We use state model to design test cases using different strategies
    - State Testing
    - Transition Testing
    - Path/Constraint path testing

- Non-executable elements e.g comments in a code.

Software Testing Methodologies

BITS Pilani

Prashant Joshi

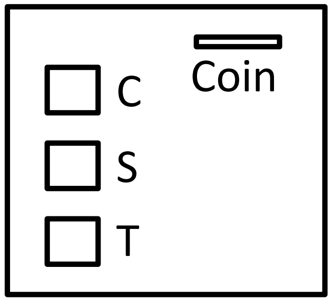
BITS Pilani
Pilani Campus

Topic 7.4: Case Study

Simple Vending Machine

- Tea/Coffee vending Machine
- Options
  - Accepts token/coin
  - Sugar

Vending Machine

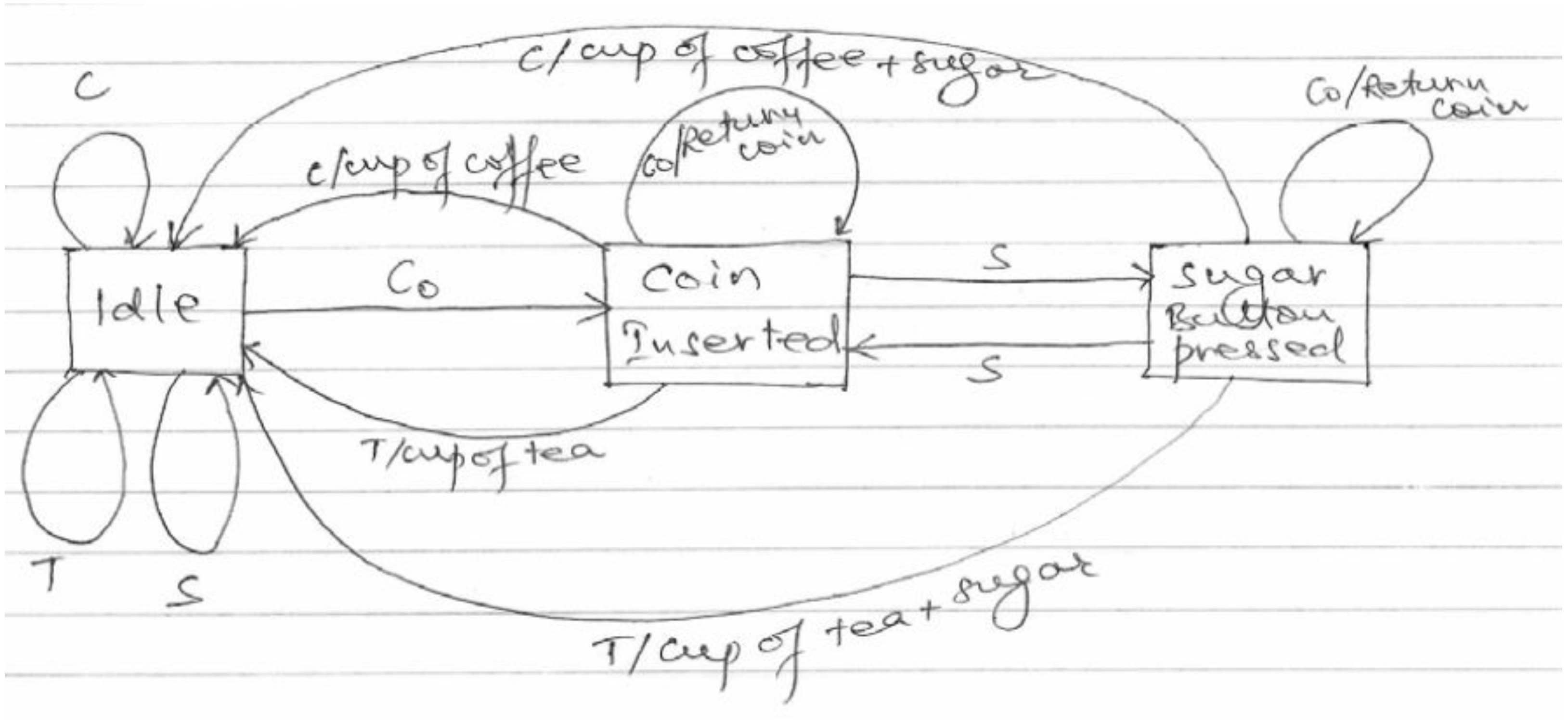


C: Coffee Button pressed

T: Tea Button pressed

S: Sugar Button pressed

Co : Coin inserted



Software Testing Methodologies

BITS Pilani

Prashant Joshi