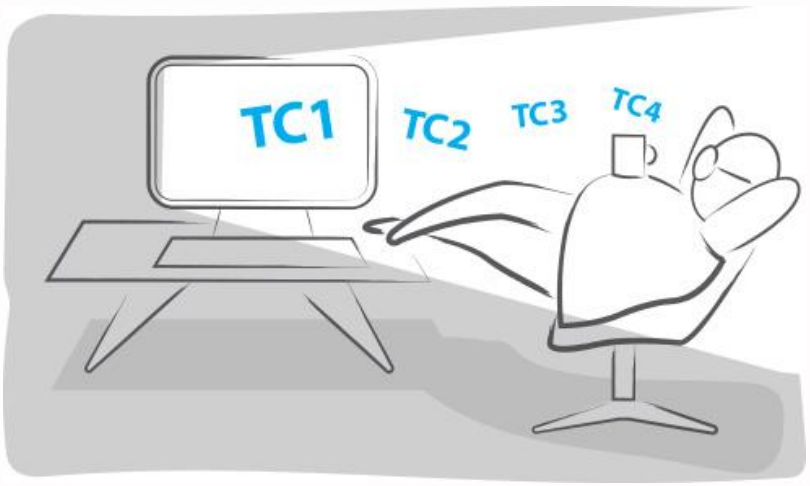
4TEST, AN AGILE MBT METHOD

Presented by István Forgács

4Test-Plus

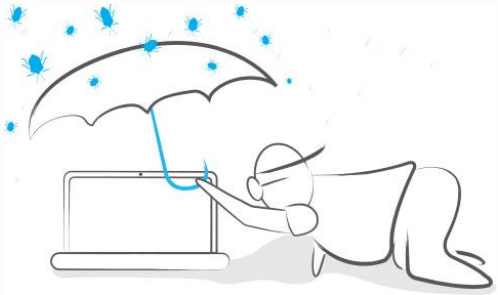
OVERVIEW

- Traditional test design
- MBT and test design techniques
- Gherkin-based modelling
- How to use 4Test



INTRODUCTION – WHAT MBT SHOULD DO

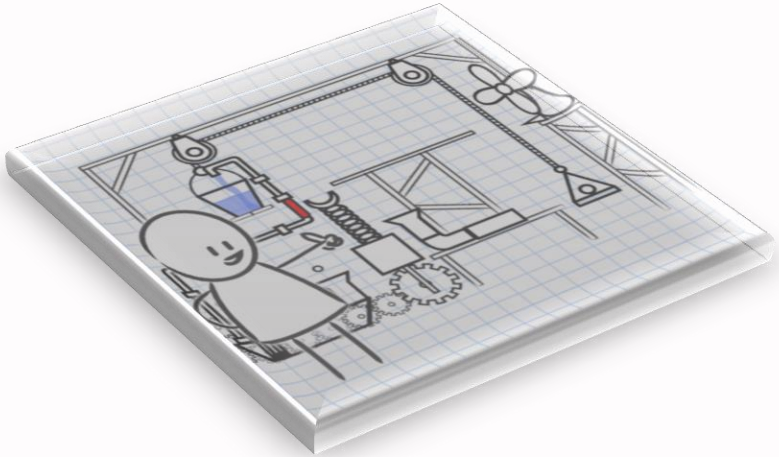
- Generate reliable test cases very fast
- Prevent design errors
- Support test first method
- Models have to be implementation independent
- Support existing test design techniques
- Should be involved in agile methodology



TRADITIONAL TEST DESIGN

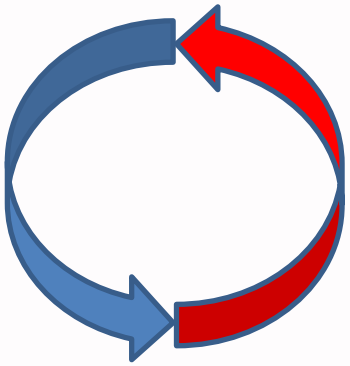
TEST DESIGN TECHNIQUES

- **EP+BVA**
- **state transition testing**
- decision table testing
- use case testing
- exploratory testing
- combinatorial testing
- etc.
- in most cases the first two are enough



IN PRACTICE MORE METHODS ARE NEEDED

- Stateless code is rare – STT is needed
- EP, BVA are always needed
- We have to combine EP, BVA, STT
- Not easy – testers with ISTQB exams had problems
- Example: Bus Ticket Vending Machine



Bus Ticket Vending Machine

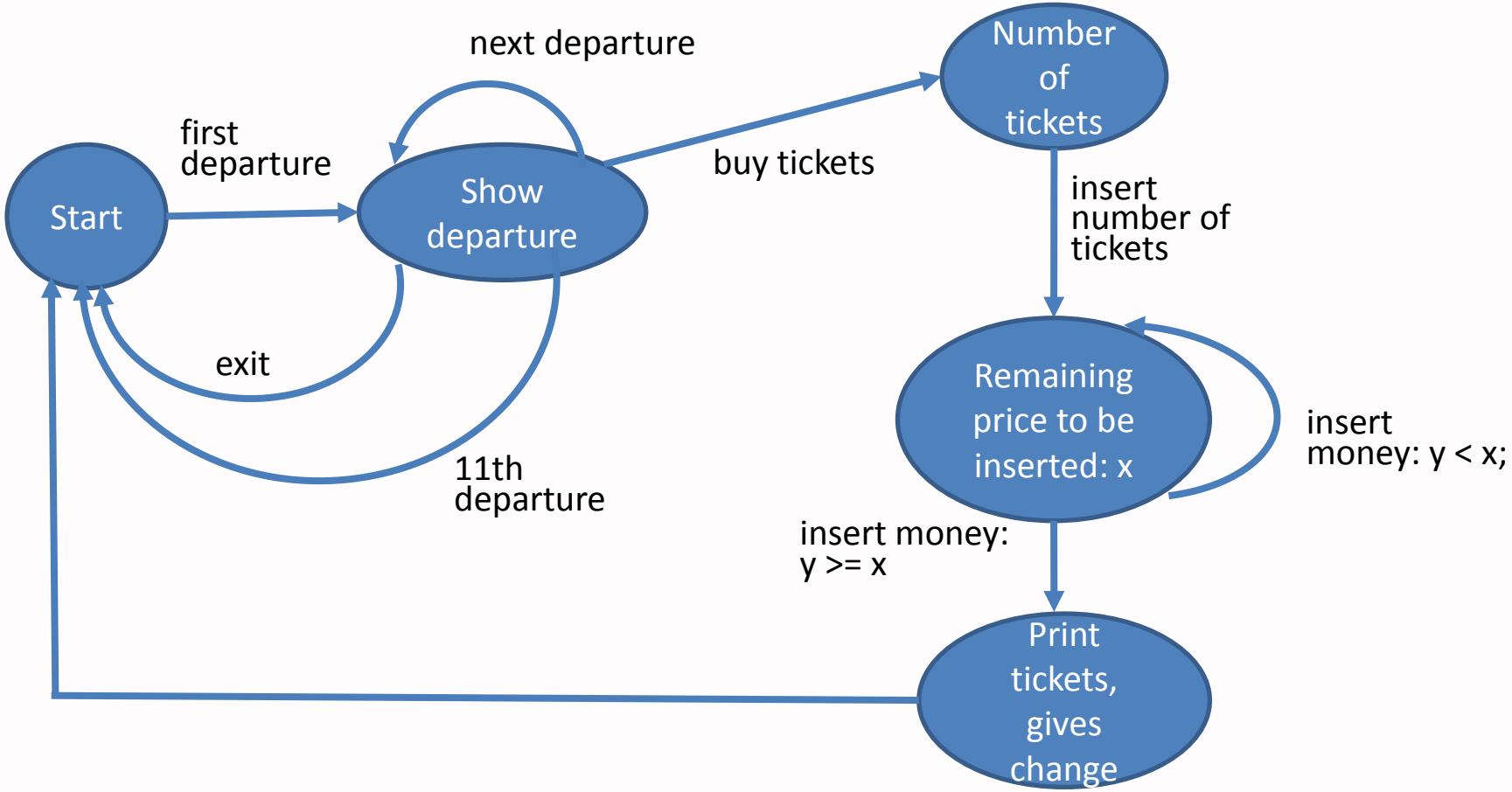
Initially, the machine offers to display the next available departure time. Selecting this only option the first departure time is displayed, and we have three possibilities:

- Buy ticket
- Go back to the initial screen
- Display the next available departure time

Selecting the last one, the subsequent departure time is displayed and we also select the three possibilities above. However, after the 10th departure time instead of showing the 11th, the machine goes back to the starting state. Selecting the first possibility, the machine asks the number of tickets to buy. We can only select a number of tickets, which are available. After we select the number of tickets, the machine shows the price of the tickets.

Then we have to insert the money. **The machine accepts coins and banknotes, i.e. €5, €10, €20, €50 and €100. The machine accepts only those banknotes for which selecting a smaller value banknote would not reach the necessary amount. €5 is always accepted. This maximum banknote is always displayed.** When we insert a coin or the acceptable banknote, the remaining tickets price to be inserted is displayed. When the inserted money reaches or exceeds the price of the tickets, they are printed, the change money is given back and the machine goes back to the initial state.

STATE TRANSITION DIAGRAM



EQUIVALENCE PARTITIONS

EP	Ticket price EUR	Acceptable banknotes
#1	<=5	€5
#2	> 5 and <=10	€5, €10
#3	> 10 and <=20	€5, €10, €20
#4	> 20 and <=50	€5, €10, €20, €50
#5	> 50	€5, €10, €20, €50, €100

BVA TEST CASES

Test case	EP	Ticket price EUR	Acceptable banknotes
T1	#1	5	€5
T2	#2	5.1	€5, €10
T3	#2	10	€5, €10
T4	#3	10.1	€5, €10, €20
T5	#3	20	€5, €10, €20
T6	#4	20.1	€5, €10, €20, €50
T7	#4	50	€5, €10, €20, €50
T8	#5	50.1	€5, €10, €20, €50, €100

SATISFYING BVAs BY ONE TEST CASE

Inserted money	Remaining amount to be inserted before	Remaining amount to be inserted after
9.9	60	50.1
0.1	50.1	50
29.9	50	20.1
0.1	20.1	20
9.9	20	10.1
0.1	10.1	10
4.9	10	5.1
0.1	5.1	5
5	5	0

TEST CASES

T1: Start:30, Show, Number:1, Remaining:30, *insert money: 30,* Print, Start, Show, Number:1, Remaining:30, Print, Start

T2**: Start:30, Show, Start, Show**, Number:1, Remaining: 30, *insert money: 30,* Print, Start

T3: Start:30, **Show, Show, Show, Show, Show, Show, Show, Show, Show, Show**, Number:1, Remaining: 30, *insert money: 30,* Print, Start

T4: Start: 30, Show, Show, Show, Show, Show, Show, Show, Show, Show, **Show**, **Start**, Show, Number:4, Remaining: 120, *insert money: 120,* Print, Start

BVA TEST CASE

T5: Start: 30, Show, Number: 2, **Remaining: 60**, **Max banknote: 100,** *insert money: 9.9,* **Remaining: 50.1**, **Max banknote: 100,** *insert money: 0.1,* **Remaining: 50**, **Max banknote: 50,** *insert money: 29.9,* **Remaining: 20.1**, **Max banknote: 50,** *insert money: 0.1,* **Remaining: 20**, **Max banknote: 20,** *insert money: 9.9,* **Remaining: 10.1**, **Max banknote: 20,** *insert money: 0.1,* **Remaining: 10**, **Max banknote: 10,** *insert money: 4.9,* **Remaining: 5.1**, **Max banknote: 10,** *insert money: 0.1,* **Remaining: 5**, **Max banknote: 5,** *insert money: 5,* Print, Start

MBT AND TEST DESIGN TECHNIQUES

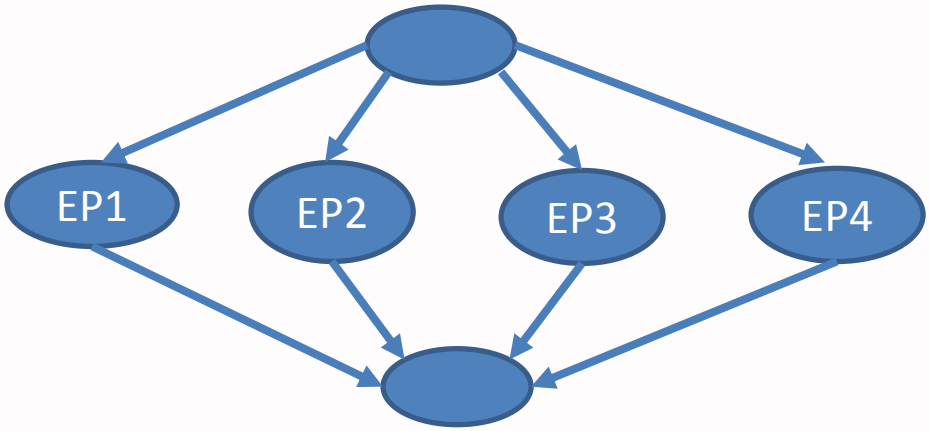
MBT MODELS – Z. MICSKEI

TOOL	INPUT FORMAT
4Test	Custom (Gherkin based)
BPM-Xchange	BPMN, UML...
Conformiq Creator	Activity Diagrams, DSL
Conformiq Designer	UML State Machines, QML
DTM	custom activity model
fMBT	Custom (AAL)
GraphWalker	FSM
JSXM	EFSM (Stream X-machines)
JTorX	LTS
MaTeLo	Markov chains
MBTsuite	UML or BPMN

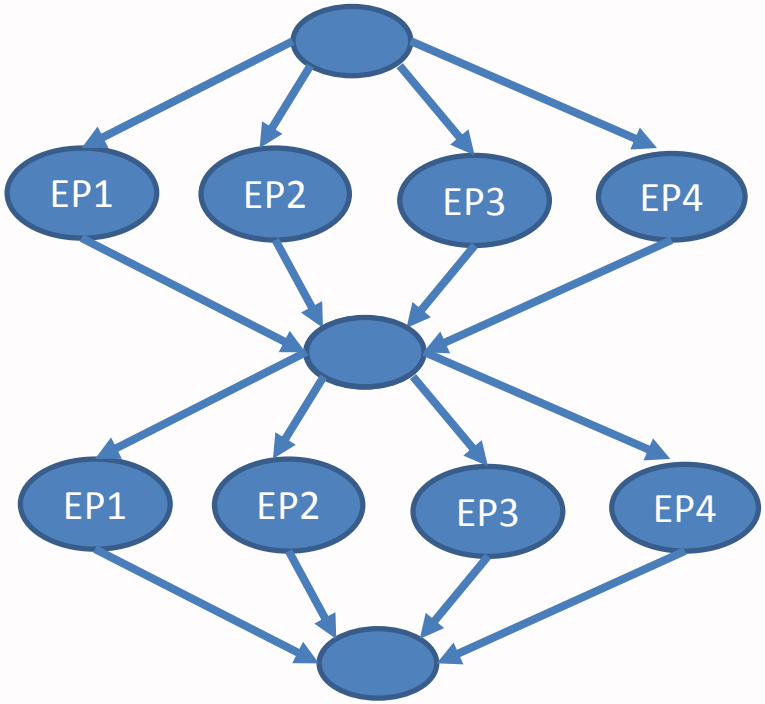
TOOL	INPUT FORMAT
MISTA	PrT net
Modbat	EFSM (Scala-based DSL)
ModelJUnit	EFSM
MoMuT::UML	UML state machines, OOAS
OSMO	model program in Java
PyModel	Python source
RT-Tester	UML/SysML, Matlab
Smartesting CertifyIt	UML + OCL
Spec Explorer	model programs in C#
Tcases	Custom
TEMPPO	Task flow model
TestCast	UML State Machines

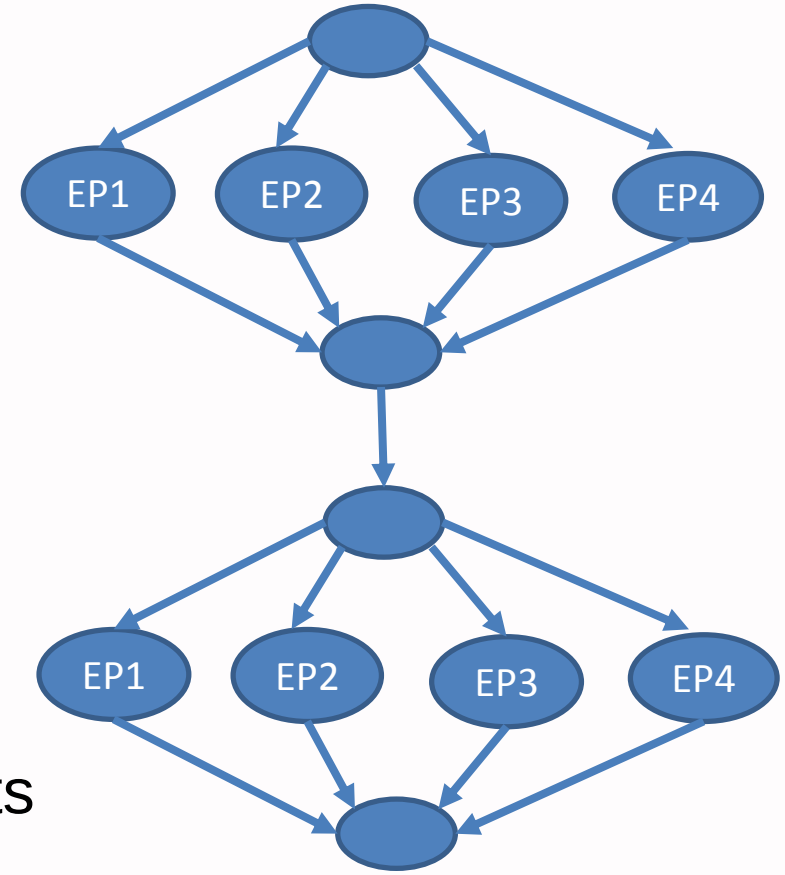
MBT WITH TEST DESIGN TECHIQUES

- Most models modelling the system and not the tests
- To include test design technology (EP, BVA) we should
  - Insert sub-models



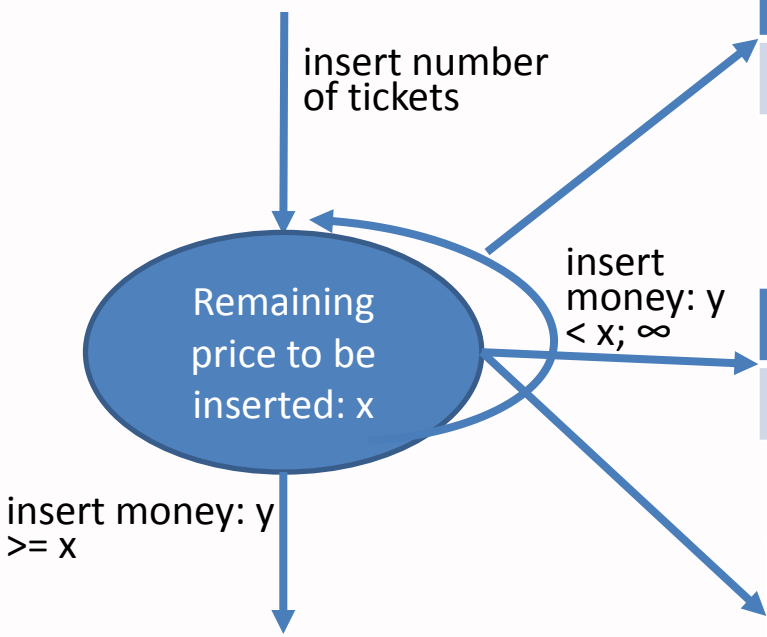
INSERTING SUBMODELS





Generates different number of tests
Test case explosion

INSERTING TABLES



30	-	-	-	-	-	-	-
9.9	0.1	29.9	0.1	9.9	0.1	4.9	0.1

50	-	-	-	-	-	-	-
100	50	50	20	20	10	10	5

50	-	-	-	-	-	-	-
50.1	50	20.1	20	10.1	10	5.1	5

- Difficult to understand
- Combination of two such cases is difficult

INSERTING CODE

- Models are not understandable for non-IT people
- Model code – graphical and code models may conflict, maintenance is difficult
  - To our knowledge: no existing solution
- Test case code – models contain only one part of the specification, less understandable, odd

GHERKIN-BASED MODELLING

PRICE REDUCTION

For online book purchasing, regular customers with cards obtain a 10% price reduction.

Similarly, any customer buying books for at least EUR 50 gets a 10% price reduction.

If somebody has a card and buys books for at least EUR 50, then the price reduction is 15%.

The total book's price appears on the screen.

Categories and choices

- card owner (I): yes (S); no
- book price (I): 49.99; 50 (S)
- price reduction (O): 10%; 15%; no reduction
- total price (O): 45; 44.99; 42.5; 49.99

GENERATED TESTS WITHOUT CONSTRAINTS

FEATURES + ADD

Price reduction

project: BuyBook UCAAT > feature: Price reduction

Search project | Exit Editor

DOCUM...✓ SAVED

B I U { }

For online book purch
price reduction.

Similarly, any custome
price reduction.

If somebody has a car
reduction is 15%.

The total book's price

CATEGORIES ✓ SAVED

```
1 card owner (I): yes (S); no
2 book price (I): 49.99; 50 (S)
3 price reduction (O): 10%; 15%; no reduction;
4 total price (O): 45; 44.99; 42.5; 49.99
5
```

CONSTRAINTS ✓ SAVED

```
1
```

TESTS | EXPORT XLS TEST REPORT | EXPORT JSON | » REGENERATE TESTS

Price reduction

T1: card owner(I) =no, book price(I) =49.99
T2: card owner(I) =yes, book price(I) =49.99
T3: card owner(I) =no, book price(I) =50

CONSTRAINTS

Card owner: WHEN card owner IS yes ~~AND book price IS 49.99~~ THEN price reduction IS 10% AND total price IS 44.99

Expensive: WHEN book price IS 50 THEN price reduction IS 10% AND total price IS 45

Both: WHEN card owner IS yes AND book price IS 50 THEN price reduction IS 15% AND total price IS 42.5

No reduction: WHEN book price IS 49.99 AND card owner IS no THEN price reduction IS no reduction AND total price IS 49.99

PRICE REDUCTION MODEL AND TESTS

FEATURES + ADD

Price reduction

project: BuyBook UCAAT > feature: Price reduction

Search project | Exit Editor

DOCUMENTATION ✓ SAVED

Normal

For online book purchasing, regular customers with cards obtain a 10% price reduction.

Similarly, any customer buying books for at least EUR 50 gets a 10% price reduction.

If somebody has a card and buys books for over EUR 50, then the price reduction is 15%.

The total book's price appears on the screen.

CATEGORIES ✓ SAVED

```
1 card owner (I): yes (S); no
2 book price (I): 49.99; 50 (S)
3 price reduction (O): 10%; 15%; no reduction;
4 total price (O): 45; 44.99; 42.5; 49.99
```

CONSTRAINTS ✓ SAVED

```
1 Card owner: WHEN card owner IS yes THEN
  price reduction IS 10% AND total price IS
  44.99
2 Expensive: WHEN book price IS 50 THEN price
  reduction IS 10% AND total price IS 45
3 Both: WHEN card owner IS yes AND book price
  IS 50 THEN price reduction IS 15% AND total
  price IS 42.5
4 No reduction: WHEN book price IS 49.99 AND
  card owner IS no THEN price reduction IS no
  reduction AND total price IS 49.99
```

TESTS | EXPORT XLS TEST REPORT | EXPORT JSON | » REGENERATE TESTS

Price reduction

Card owner: card owner(I) =yes, book price(I) =49.99, price reduction(O) =10%, total price(O) =44.99
Expensive: card owner(I) =no, book price(I) =50, price reduction(O) =10%, total price(O) =45
Both: card owner(I) =yes, book price(I) =50, price reduction(O) =15%, total price(O) =42.5
No reduction: book price(I) =49.99, card owner(I) =no, price reduction(O) =no reduction, total price(O) =49.99

GENERATED TEST CASES

- Card owner: card owner*(I)* =yes, book price*(I)* =49.99, price reduction*(O)* =10%, total price*(O)* =44.99
- Expensive: card owner*(I)* =no, book price*(I)* =50, price reduction*(O)* =10%, total price*(O)* =45
- Both: card owner*(I)* =yes, book price*(I)* =50, price reduction*(O)* =15%, total price*(O)* =42.5
- No reduction: book price*(I)* =49.99, card owner*(I)* =no, price reduction*(O)* =no reduction, total price*(O)* =49.99

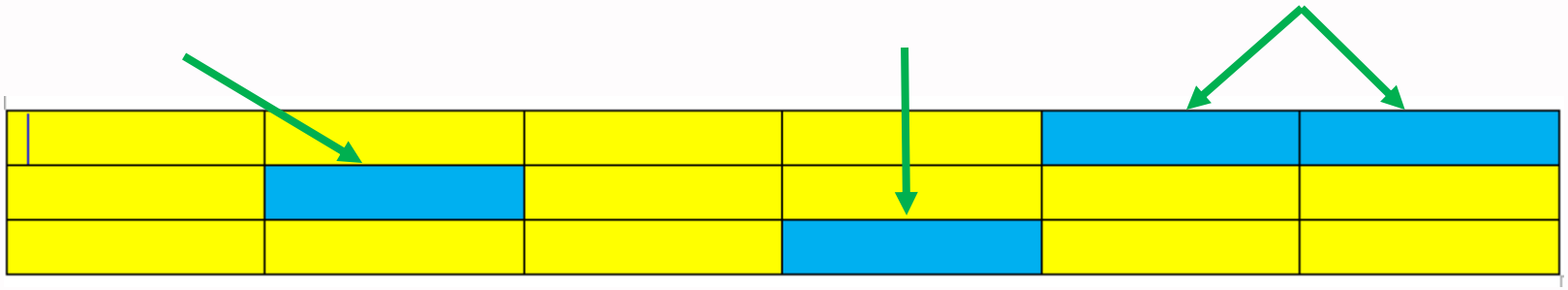
MANUAL TEST CASES

Test cases	4				
passed	3 (75%)				
failed	1 (25%)				
passed+failed	4 (100%)				
Result (passed/failed)	Test case	card owner (I)	book price (I)	price reduction (O)	total price (O)
passed	Card owner	yes	49.99	10%	44.99
failed	Expensive	no	50	10%	45
passed	Both	yes	50	15%	42.5
passed	No reduction	no	49.99	no reduction	49.99

MODEL

- Every user story/feature has a test model
- Categories are preconditions, variables, parameters
- Choices are abstract or given values of the variable
- Constraints are relationships among choices
- Models are structured – the model of feature A may contain a test from feature B
- Complex test cases are generated from understandable models
- Models are developed iteratively

TEST SELECTION CRITERION



- Every choice has to be covered by at least one test case
- Every constraint has to be covered by at least one test case
- Test selection criterion is unique

FEATURES, CATEGORIES AND CHOICES

- Each feature relates to a model
- Categories can
  - Input (I) and (A)
  - Output (O)
  - Feature (F)
- Choices can be abstract or given (value)
- MyCat(I): MyFirstChoice; MySecondChoice; MyLastChoice; 13

GHERKIN

- GIVEN describes preconditions, and can be omitted.
- WHEN contains the inputs and obligatory
- THEN contains the output and obligatory
- AND connects two GIVEN/WHEN/THEN expressions
- IS/ARE connects a category and a choice of this category, such as *MyCat IS MyChoice*
- WHEN - THEN - WHEN - THEN - WHEN - THEN sequence is possible:
  - *WHEN InsertPIN IS wrong THEN message IS wrong PIN*
    WHEN InsertPIN IS good THEN message IS select transition

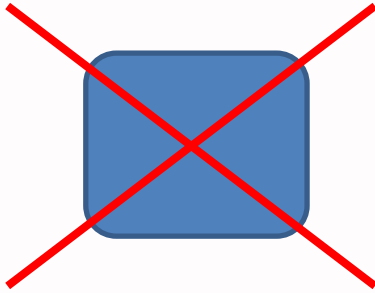
(D)

- **(D)** is the default value of the choices.
- A (D) type choice will be selected for test case generation if all the other choices have been in at least one test case

Example

money transfer price (O): 0; 100(D); 1000

IR

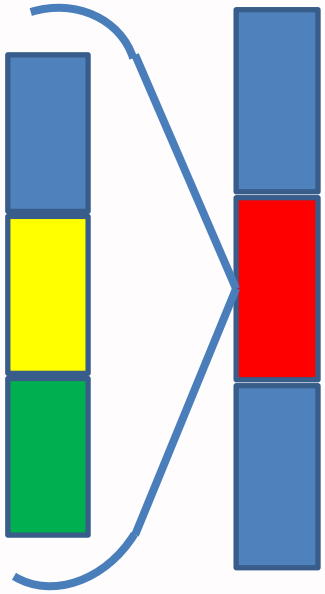


- For IR choice will not be considered as covered during the test generation
- A choice may entirely determine the test - other inputs are irrelevant
- Another test case has to be generated with the same choice for which this choice is relevant

Invalid login : WHEN login name IS invalid AND password IS IR THEN login status IS not logged in

MULTI-LAYER STRUCTURE – (F)

- **(F)** is a category type where the category name is an existing (lower level) feature
- The choices of this category can be the test case/constraint names



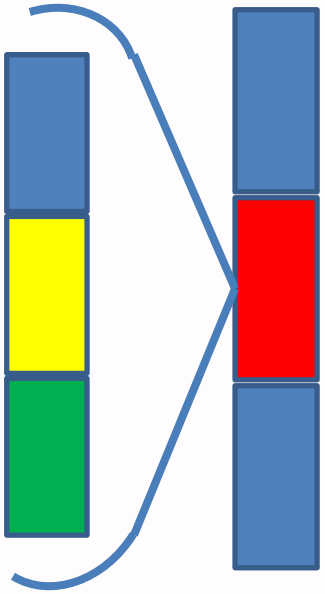
login (F): success; faulty (S)

MyTest: GIVEN **login** IS **success** WHEN total price IS 0 THEN paying IS not possible

MULTI-LAYER STRUCTURE – IMPLICIT

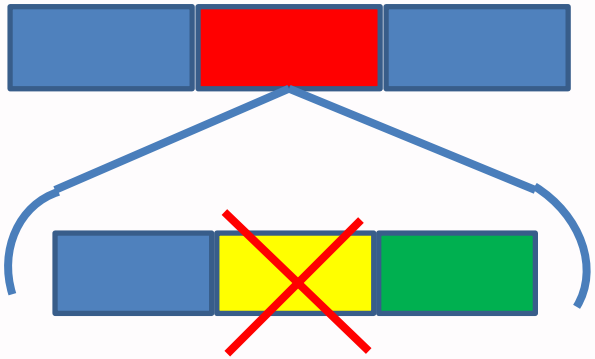
~~login (F): success; faulty (S)~~

MyTest: GIVEN **login** IS **success** WHEN total price IS 0 THEN paying IS not possible



- **Difference – indirect call will not imply login to be involved in other test cases**

MULTI-LAYER STRUCTURE – FLEXIBLE



- We can select elements from a sub-model
- We can use them as choices

TABLE: |

- Tables reducing the number of constraint if they differ in only in the choices

WHEN original price IS 20 AND reduction is 10 THEN total price IS 19

WHEN original price IS 100 AND reduction is 12 THEN total price IS 88

WHEN original price IS 20 | 100 AND reduction is 10 | 12 THEN total price IS 19 | 88

- Categories can be omitted

T1	T2	T3	T4	T5	T6

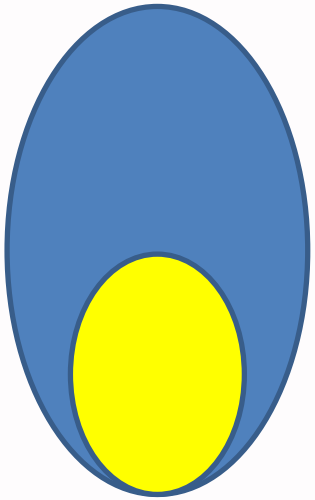
SUB-CONSTRAINT

- A constraint, which can be used in other constraints.
- Common sub-models can be used

three items to pay: WHEN number of items IS 3 AND Action IS goto pay

WHEN select food IS pizza AND **three items to pay** THEN state IS pay

WHEN select food IS fish and chips AND **three items to pay** THEN state IS pay

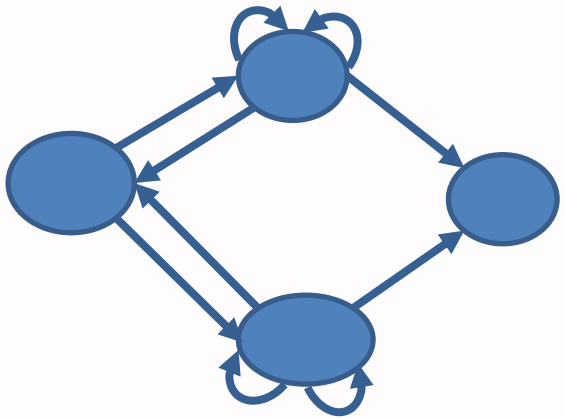


GO/GOES

- Makes it possible to model state transition diagrams.

WHEN In IS 1 THEN Out IS yes WHEN In IS 2 THEN Out IS no WHEN In IS 3 THEN Out IS N/A

WHEN In GOES 1; 2; 3 THEN Out GOES yes; no; N/A



ONLY

- **ONLY** + category names describe, which categories remain in a test case.
- For state transition testing it’s essential
- Single ONLY results in that only categories in the constraint are included in the test.
- For sub-constraints the only-s are inherited and united

ONLY EXAMPLE

sub1 (ONLY cat1; cat4): WHEN cat2 IS 1 THEN…

const (ONLY cat7; cat8): WHEN sub1 AND cat6 IS yes THEN

- the generated test case contains : cat1, cat2, cat4, cat6, cat7, cat8
- and does not contain cat3, cat5

SKIPPED

- SKIPPED + category names result in the listed categories are omitted from the test case.
- Using sub-constraints the skipped categories are inherited

sub1: WHEN cat2 IS 1 AND cat3 IS SKIPPED

const: WHEN sub1 AND cat6 IS yes AND cat5 IS SKIPPED

- The generated test case contains : cat1, cat2, cat4, cat6, cat7, cat8 and does not contain cat3, cat5

SOMEOF

- If there are more available choices, which can be selected for a test case we use SOMEOF.
- In this case the generator selects one of the choices following keyword SOMEOF.

WHEN Fruit IS SOMEOF apple; peach; orange; banana THEN Juice IS fine

- The generated test case may contain Fruit = apple or Fruit = peach or Fruit = orange or Fruit = banana.

ANYOF

- In non-deterministic applications the output is not a unique value
- More values may be equally correct
- If the output at the execution is one of them, then the test passed, else failed.

WHEN MyCat IS firstChoice THEN NonDetOutput ANYOF output1; output2; output 3

HOW TO USE 4TEST

STATE TRANSITION TESTING DEMO

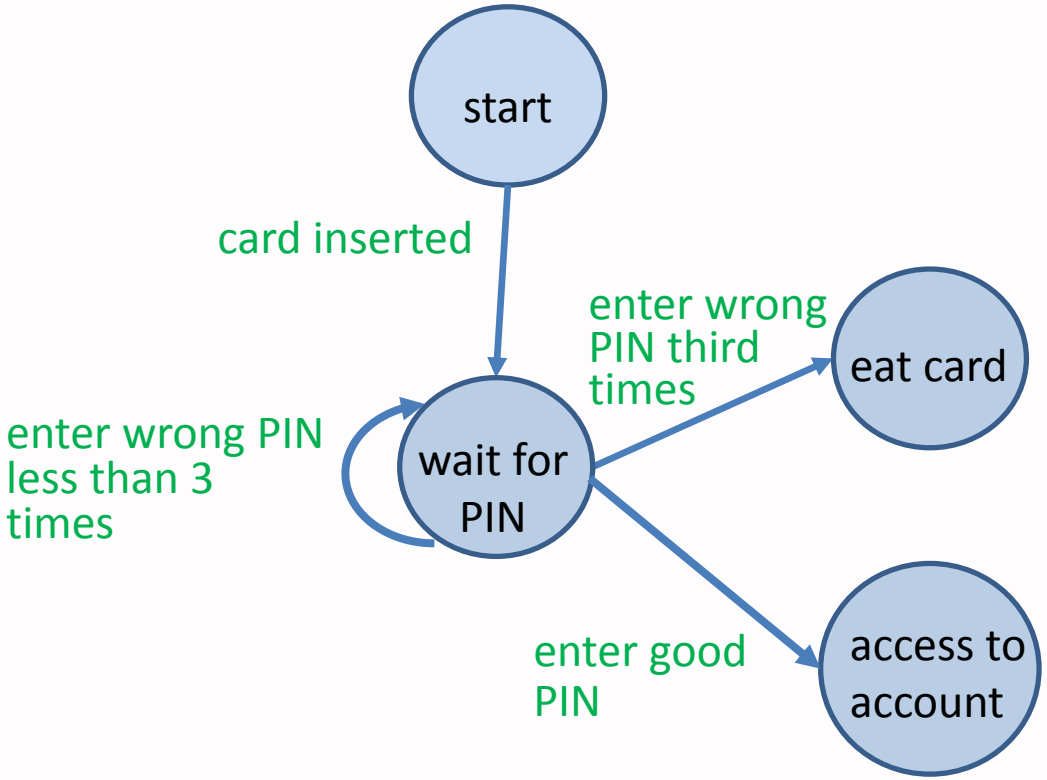
Insert PIN

After inserting a credit card, the machine asks for a PIN code.

If the PIN code is correct, then the account is accessible.

If the PIN code is wrong, then the machine asks again, but after entering the third wrong PIN code the card will be eaten by the cash machine.

STATE TRANSITION DIAGRAM



MODEL FOR INSERT PIN

actions (I): insert card; enter wrong PIN; enter good PIN
ATM states (O): wait for PIN; eat card; access to account

wrong PIN twice(ONLY): WHEN actions GOES enter wrong PIN; enter wrong PIN THEN ATM states GO wait for PIN; wait for PIN

success: WHEN actions IS insert card AND wrong PIN twice WHEN actions IS enter good PIN THEN ATM states IS access to account

fail: WHEN actions IS insert card AND wrong PIN twice WHEN actions IS enter wrong PIN THEN ATM states IS eat card

short: WHEN actions IS insert card WHEN actions IS enter good PIN THEN ATM states IS access to account

MODEL AND GENERATED TESTS FOR PIN

FEA... + ADD

insert PIN

project: Insert PIN > feature: insert PIN

Exit Editor

DOCUMENTATION ✓ SAVED

After inserting a credit card, the machine asks for a
is correct, then the account is accessible. If the PIN
machine asks again, but after entering the third wr
be eaten by the cash machine.

CATEGORIES ✓ SAVED

```
1 actions (I): insert card; enter wrong PIN; enter good PIN
2 ATM states (O): wait for PIN; eat card; access to account
3
```

CONSTRAINTS ✓ SAVED

```
1 wrong PIN twice(ONLY): WHEN  actions GOES enter wrong PIN;
  enter wrong PIN THEN ATM states GO wait for PIN; wait for
  PIN
2 success: WHEN actions IS insert card AND wrong PIN twice
  WHEN actions IS  enter good PIN THEN ATM states IS access
  to account
3 fail: WHEN actions IS insert card AND wrong PIN twice WHEN
  actions IS  enter wrong PIN THEN ATM states IS eat card
4 short: WHEN actions IS insert card WHEN actions IS enter
  good PIN THEN ATM states IS access to account
```

TESTS | EXPORT XLS TEST REPORT | EXPORT JSON | » REGENERATE TESTS

```
success:
    actions(I) =insert card,
    wrong PIN twice [ actions(I) =enter wrong PIN, ATM states(O) =wait for PIN, actions(I) =enter wrong PIN, ATM states(O) =wait for PIN ],
    actions(I) =enter good PIN, ATM states(O) =access to account
fail:
    actions(I) =insert card,
    wrong PIN twice [ actions(I) =enter wrong PIN, ATM states(O) =wait for PIN, actions(I) =enter wrong PIN, ATM states(O) =wait for PIN ],
    actions(I) =enter wrong PIN, ATM states(O) =eat card
short:    actions(I) =insert card, actions(I) =enter good PIN, ATM states(O) =access to account
```

MODEL EDITOR

CATEGORIES ✓ SAVED

```
1 Login(F): successful;
2 card owner (I): y
3 new book price (I
4 second hand book
5 price reduction f
6 total price (O): 149.99; 45; 44.99; 42.5; 142.5
```

- faulty login name
- faulty login name(S)
- faulty psw
- faulty psw(S)

TEST CASES OF INSERT PIN

success: actions*(I)* =insert card, wrong PIN twice [actions*(I)* =enter wrong PIN, ATM states*(O)* =wait for PIN, actions*(I)* =enter wrong PIN, ATM states*(O)* =wait for PIN], actions*(I)* =enter good PIN,ATM states*(O)* =access to account

fail: actions*(I)* =insert card, wrong PIN twice [actions*(I)* =enter wrong PIN, ATM states*(O)* =wait for PIN, actions*(I)* =enter wrong PIN, ATM states*(O)* =wait for PIN], actions*(I)* =enter wrong PIN,ATM states*(O)* =eat card

short: actions*(I)* =insert card, actions*(I)* =enter good PIN, ATM states*(O)* =access to account

Bus Ticket Vending Machine (BTVM)

Initially, the machine offers to display the next available departure time. Selecting this only option the first departure time is displayed, and we have three possibilities:

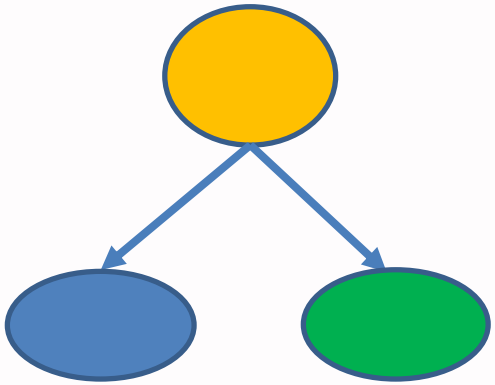
- Buy ticket
- Go back to the initial screen
- Display the next available departure time

Selecting the last one, the subsequent departure time is displayed and we also select the three possibilities above. However, after the 10th departure time instead of showing the 11th, the machine goes back to the starting state. Selecting the first possibility, the machine asks the number of tickets to buy. We can only select a number of tickets, which are available. After we select the number of tickets, the machine shows the price of the tickets.

Then we have to insert the money. **The machine accepts coins and banknotes, i.e. €5, €10, €20, €50 and €100. The machine accepts only those banknotes for which selecting a smaller value banknote would not reach the necessary amount. €5 is always accepted. This maximum banknote is always displayed.** When we insert a coin or the acceptable banknote, the remaining tickets price to be inserted is displayed. When the inserted money reaches or exceeds the price of the tickets, they are printed, the change money is given back and the machine goes back to the initial state.

HOW TO MAKE MODEL FOR BTVM

- Divide and conquer – make models as simple as possible
- Separate EP+BVA from STT
  - Separate “Insert money”
- Let the models be understandable
- We have two sub-features and models:
  - bus ticket
  - insert money



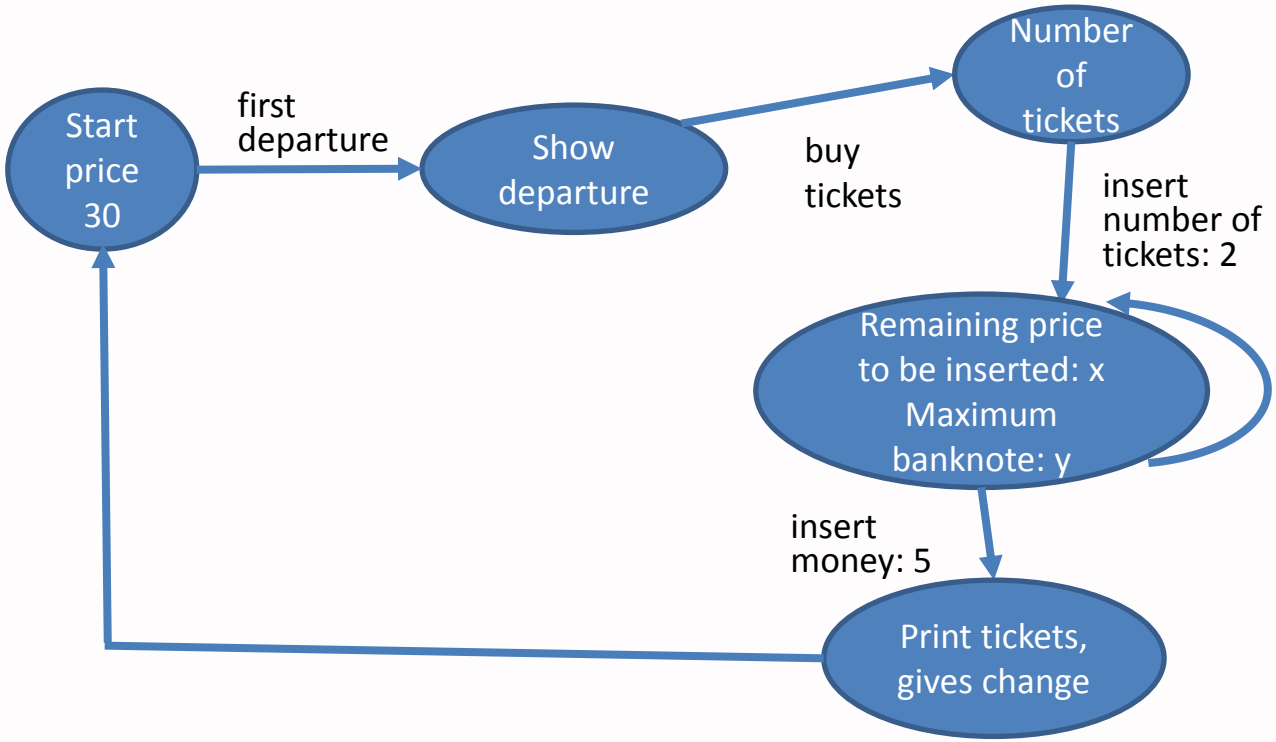
CATEGORIES AND CHOICES FOR BUS TICKETS

price of one ticket (I): 30
bus ticket actions(I): buy; go back; show next departure
number of tickets (I): 1; 2
total price of tickets (O): 30; 60
bus ticket states (O): init screen; next available departure; enter number of tickets; change money back and tickets printed and back to initial screen
maximum banknote(O): 100

SUB-CONSTRAINT FOR 10 TIMES DEPARTURE

10 times departure: WHEN bus ticket actions GO show next departure; show next departure; show next departure; show next departure; show next departure; show next departure; show next departure; show next departure; show next departure; show next departure; THEN bus ticket states GO next available departure; next available departure; next available departure; next available departure; next available departure; next available departure; next available departure; next available departure; next available departure; next available departure;

EP TEST DIAGRAM



insert money: 9.9; 0.1; 29.9; 0.1; 9.9; 0.1; 4.9; 0.1

remaining money: 50.1; 50; 20.1; 20;10.1; 10; 5.1; 5

maximum banknote: 100; 50; 50; 20; 20; 10; 10; 5

EP TEST

EP test: GIVEN 10 times departure WHEN bus ticket actions IS buy THEN bus ticket states IS enter number of tickets WHEN number of tickets IS 2 THEN total price of tickets IS 60 AND maximum banknote IS 100 WHEN insert money IS EUR 60 THEN bus ticket states IS change money back and tickets printed and back to initial screen

INSERT MONEY – CATEGORIES AND CHOICES

insert money (I): 0.1; 4.9; 5; 9.9; 29.9; 30

remaining money (O): 50.1; 50; 20.1; 20; 10.1; 10; 5.1; 5; 0

maximum banknote (O): 5; 10; 20; 50; 100

- Non-ordered lists

CONSTRAINTS

insert remaining EUR 5: WHEN insert money IS 5 THEN remaining money IS 0

EUR 60: WHEN insert money GOES 9.9; 0.1; 29.9; 0.1; 9.9; 0.1; 4.9; 0.1 THEN remaining money GOES 50.1; 50; 20.1; 20;10.1; 10; 5.1; 5 AND maximum banknote GOES 100; 50; 50; 20; 20; 10; 10; 5 WHEN insert remaining EUR 5

EUR 30: WHEN insert money IS 30 THEN remaining money IS 0

EP TEST CASE

EP test: 10 times departure [bus ticket actions*(I)* =show next departure,bus ticket actions*(I)* =show next departure,bus ticket actions*(I)* =show next departure,bus ticket actions*(I)* =show next departure,bus ticket actions*(I)* =show next departure,bus ticket actions*(I)* =show next departure,bus ticket actions*(I)* =show next departure,bus ticket actions*(I)* =show next departure,bus ticket actions*(I)* =show next departure],price of one ticket*(I)* =30,bus ticket actions*(I)* =buy,bus ticket states*(O)* =enter number of tickets,number of tickets*(I)* =2,total price of tickets*(O)* =60,maximum banknote*(O)* =100,insert money*(F)* =EUR 60 [insert money*(I)* =9.9,remaining money*(O)* =50.1,maximum banknote*(O)* =100,insert money*(I)* =0.1,remaining money*(O)* =50,maximum banknote*(O)* =50,insert money*(I)* =29.9,remaining money*(O)* =20.1,maximum banknote*(O)* =50,insert money*(I)* =0.1,remaining money*(O)* =20,maximum banknote*(O)* =20,insert money*(I)* =9.9,remaining money*(O)* =10.1,maximum banknote*(O)* =20,insert money*(I)* =0.1,remaining money*(O)* =10,maximum banknote*(O)* =10,insert money*(I)* =4.9,remaining money*(O)* =5.1,maximum banknote*(O)* =10,insert money*(I)* =0.1,remaining money*(O)* =5,maximum banknote*(O)* =5,insert remaining EUR 5 [insert money*(I)* =5,remaining money*(O)* =0]],bus ticket states*(O)* =change money back and tickets printed and back to initial screen

GENERATED MANUAL TEST CASE FOR EP+BVA

Test case	EP test	
	null (null)	10 times departure
	bus ticket actions (I)	show next departure
	bus ticket actions (I)	show next departure
	bus ticket actions (I)	show next departure
	bus ticket actions (I)	show next departure
	bus ticket actions (I)	show next departure
	bus ticket actions (I)	show next departure
	bus ticket actions (I)	show next departure
	bus ticket actions (I)	show next departure
	bus ticket actions (I)	show next departure
	bus ticket actions (I)	show next departure
	price of one ticket (I)	30
	bus ticket actions (I)	buy
	bus ticket states (O)	enter number of tickets
	number of tickets (I)	2
	total price of tickets (O)	60
	maximum banknote (O)	100

	insert money (F)	EUR 60
	insert money (I)	9.9
	remaining money (O)	50.1
	maximum banknote (O)	100
	insert money (I)	0.1
	remaining money (O)	50
	maximum banknote (O)	50
	insert money (I)	29.9
	remaining money (O)	20.1
	maximum banknote (O)	50
	insert money (I)	0.1
	remaining money (O)	20
	maximum banknote (O)	20
	insert money (I)	9.9
	remaining money (O)	10.1
	maximum banknote (O)	20
	insert money (I)	0.1
	remaining money (O)	10
	maximum banknote (O)	10
	insert money (I)	4.9
	remaining money (O)	5.1
	maximum banknote (O)	10
	insert money (I)	0.1
	remaining money (O)	5
	maximum banknote (O)	5
	null (null)	insert remaining EUR 5
	insert money (I)	5
	remaining money (O)	0
	bus ticket states (O)	change money back and tickets printed and back to initial screen

BENEFITS OF GHERKIN-BASED MODELS

- Real test model – testers can use it
- No test selection criterion has to be set
- Fewer test cases
- Tests are understandable for everybody
- Learning curve is manageable – about 20 hours
- Flexible sub-modelling
- Smooth integration of EP, BVA and STT

THANK YOU!

QUESTIONS?

MORE INFORMATION:

4test.io

forgacs@4test.hu