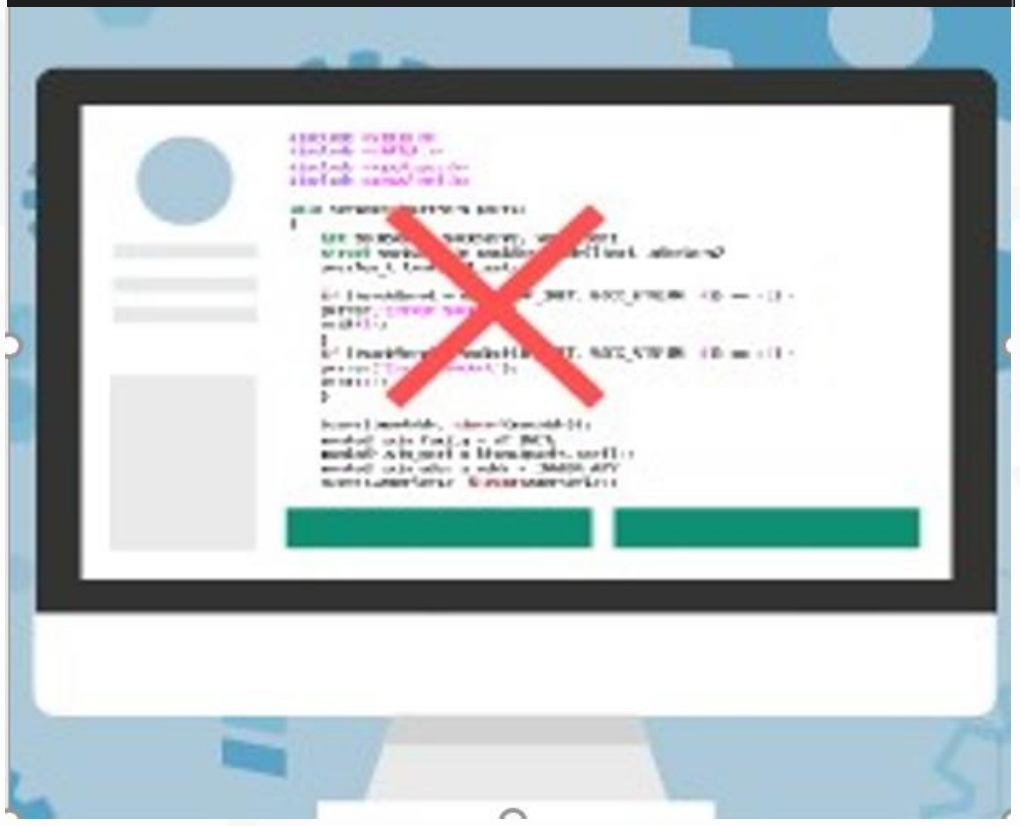
Scriptless test automation for GUI testing

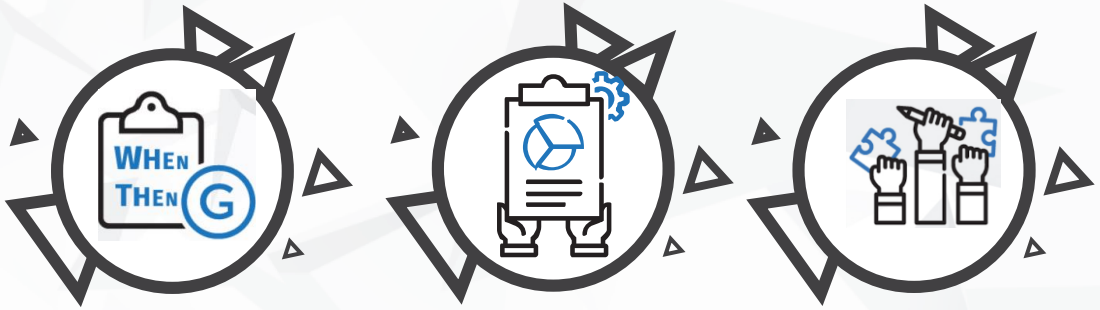
Text-based test design automation and test automation without coding, easy to use, easy to learn, transparent for all stakeholders

Scriptless test automation

```
Successful login:
GIVEN Browser IS Chrome
WHEN FourTest Login > Sign in with Google IS #pressed
  THEN Next IS #present
WHEN Email or PhoneNumber IS fourtest001@gmail.com
AND Next IS #pressed
WHEN Password IS four-test1
AND Next IS #pressed
  THEN 4Test Projects > Project List IS #present
WHEN Browser IS ChromeStop
```



- **Test first** Gherkin-based test automation
- Test Design-Driven Development (TDDD)
- One test case generation takes **<20 minutes**
- Executable acceptance criteria based on requirements/user stories
- **Web-based**, easy to start

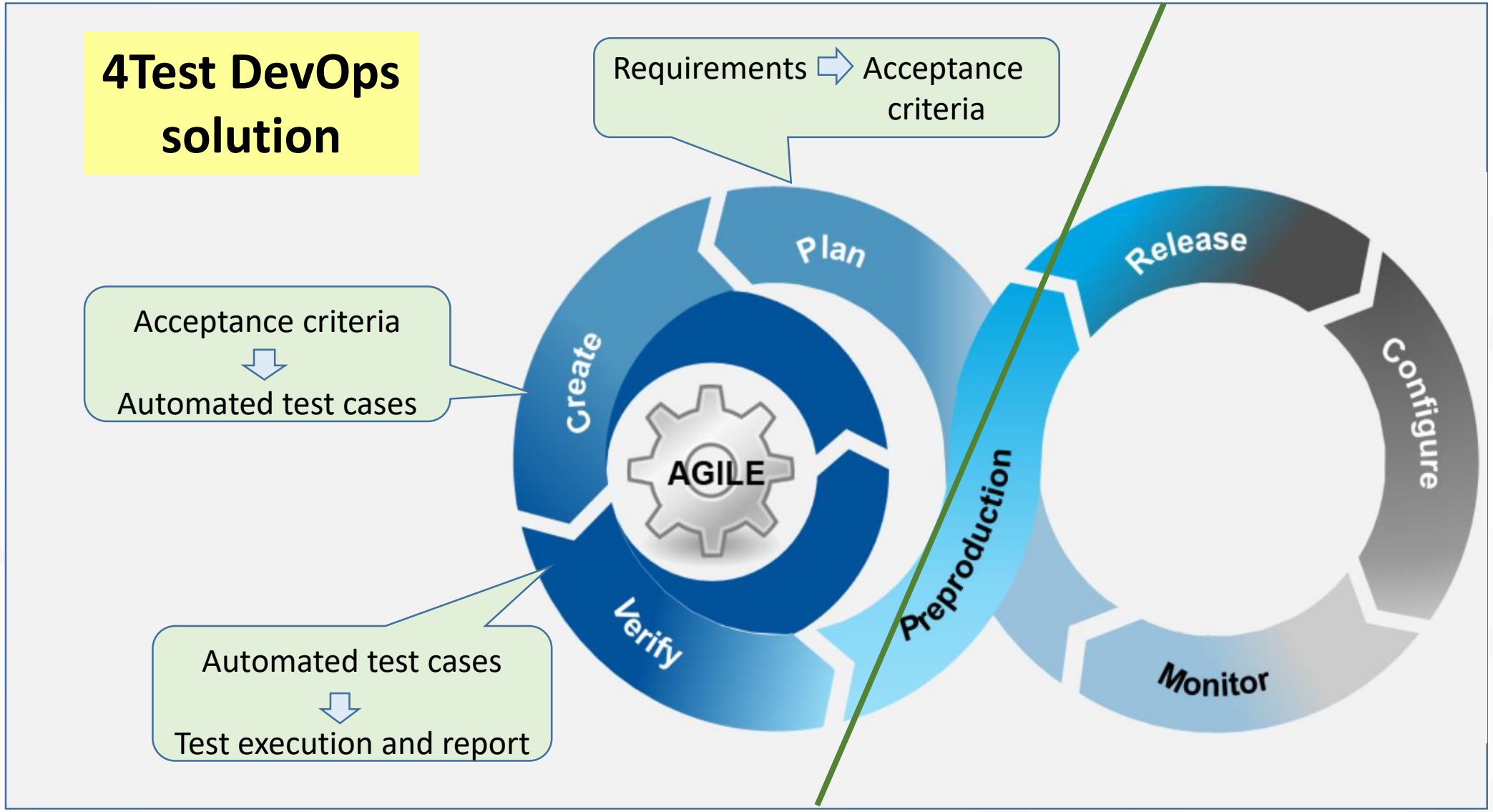


Gherkin based

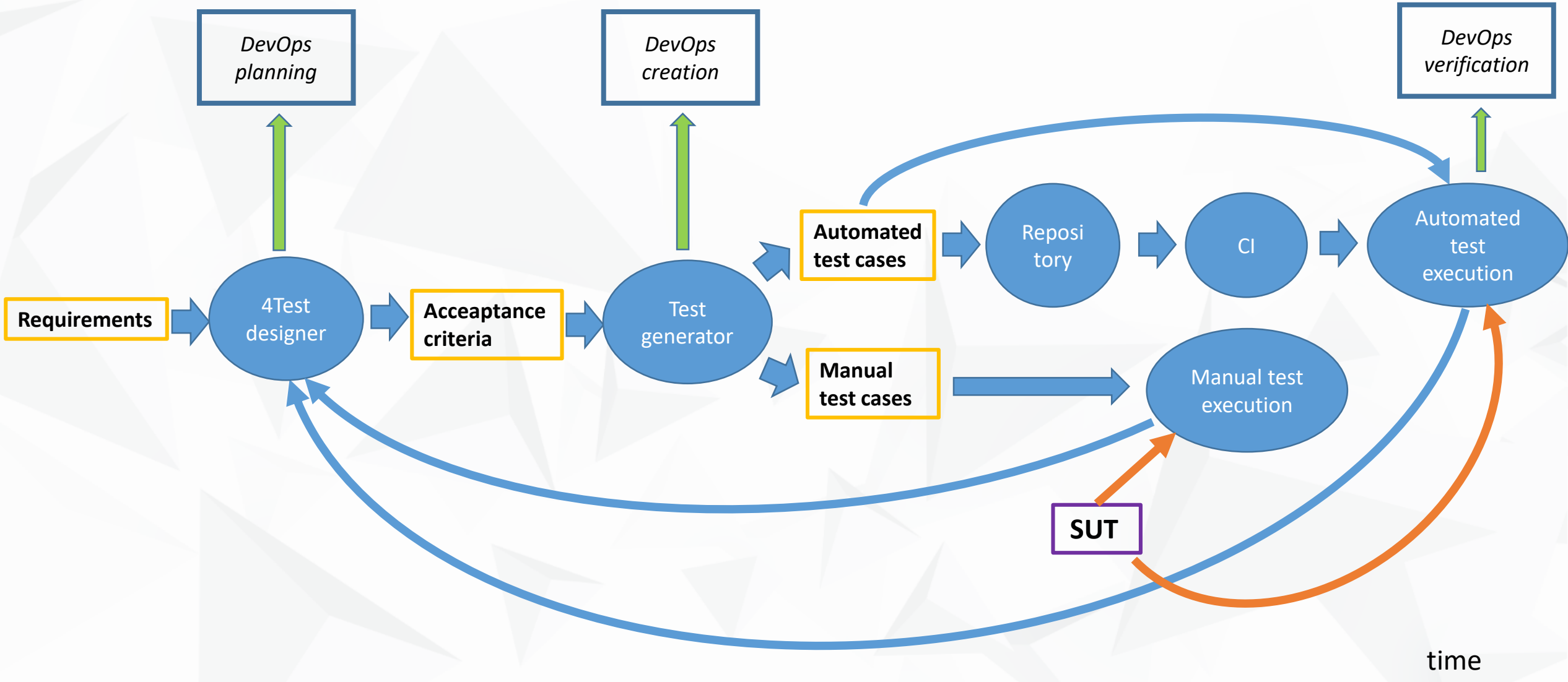
Test first scriptless automation

Ready for agile teams

4Test DevOps solution



4Test testing process



Acceptance criteria

From requirements concrete acceptance criteria are created

WHEN Book price IS 10 AND VIP IS yes
THEN reduced price IS 9

A complete test case generated from this will consist of more data, which is superfluous and reduces understandability

Acceptance criteria are validated easily against

- Correctness
- Completness (with respect to risk)

Test cases are generated from acceptance criteria

4Test description with extended Gherkin

Gherkin++ supports

- State transition testing
- Equivalence partitioning
- Boundary value analysis
- Combination of them
- Test modularisation

Gherkin

- Projects
  - Features
    - Requirements
    - Categories – choices
    - Constraints (acceptance criteria)

Feature: Price reduction

R1. For online book purchasing, regular customers with cards obtain a 10% price reduction.

R2. Customer buying books for at least EUR 50 gets a 10% price reduction.

R3. If somebody has a card and buys books for at least EUR 50, then the price reduction is 15%.

R4. The total book's price appears on the screen.

Categories and choices

Feature: Price reduction

R1.For online book purchasing, regular customers with cards obtain a 10% price reduction.
R2. Any customer buying books for at least EUR 50 gets a 10% price reduction.
R3. If somebody has a card and buys books for at least EUR 50, then the price reduction is 15%.
R4. The total book's price appears on the screen.

- 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

(I) input, (O) output, (S) single, only one test case is generated with this choice/value

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

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

Test generation

- A test contains a choice form every category
- (S) will be generated once
- (D) will be generated for outputs

Test selection criterion

- For every acceptance criterion one test case is generated
- Each choice will be in at least one test case

Basics

- GIVEN describes preconditions, and can be omitted.
- INITIALLY described initial state – an output before events
- 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

(I), (A), (O), (IA), OA), (F)

- (I): input
- (A): action such as press a button, it’s also input
- (F) submodel details later
- (O): output
- (IO), (AO): can be used for input and output

Example.

LoginName(I): Smith; Roth

Press(A): login; next; exit

Total price(O): 10; 20

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

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

In the first case no output will be generated

Constraint call without declaration

- We can call another test/constraint without declaration in the categories

~~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

PRECONDITION

- In lots of the cases when we would like to end-to-end test a feature, we have to reach the feature to be tested.
- This requires to set some preconditions, i.e. the necessary input values.

PRECONDITION Login IS successful AND Action IS open menu window

WHEN select food IS pizza AND number of items IS 3 AND Action IS goto pay THEN state IS pay

GIVEN Login IS successful AND Action IS open menu window WHEN select food IS pizza AND number of items IS 3 AND Action IS goto pay THEN state IS pay

PRECONDITION is valid till the next PRECONDITION or the end of constraints

Sub-constraint

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

SUB 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

- Subconstraints can be called in another features as well – can be used for the input part of more tests.

Code generation – basics

Gherkin – feature **Login**

```
Logged in:
WHEN Login name IS Hall AND Password IS 2@A9ih
WHEN Login button IS #pressed
THEN Message IS successful login

Feature name -> SelectWindow
WHEN Category IS Choice -> SetValue("Category", "Choice"
THEN Category IS Choice -> VerifyValue("Category", "Choice"
```

Code

```
SelectWindow( "Login" );
SetValue( "Login name", "Hall" );
SetValue( "password", "2@A9ih" );
ClickOn( "Login button");
VerifyValue( "Message", "successful login");
```

- for keyword, > for window

#pressed – ClickOn

#present – VerifyExists: YES

#non-present – VerifyExists: NO

#active – VerifyIsActive: YES

#non-active – VerifyIsActive: NO

WHEN Beer > Plus(A) IS #pressed

SelectWindow("Beer");

ClickOn("Plus");

{A; B; C} > {Plus; Delete}(I):

- WHEN A > Plus
- WHEN B > Plus
- WHEN A > Delete
- WHEN C > Delete

Test execution

From 4Test, immediately when a 4Test specification is ready

- One test case
- Test cases for a feature
- All the test cases

CI

- Test code is generated
- Test code is deployed to Travis
- Test case are executed in a scheduled way

Demo