SIMPL-T:

SDL Intended for Management and Planning of Tests

By

Qing Li, Robert Probert, William Skelton, Yiqun Xu

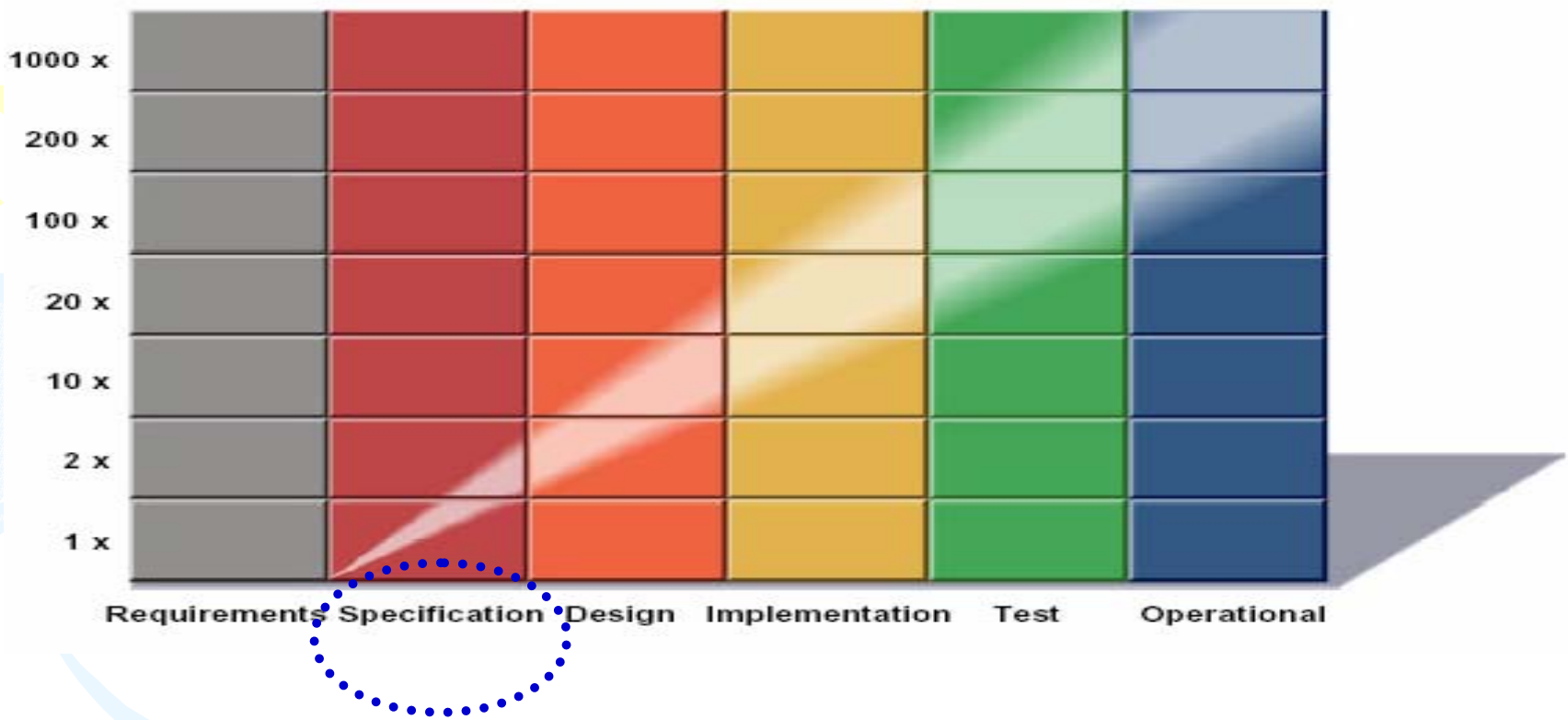
SAM'04 Workshop, Ottawa, Canada

June, 2004

Outline

- Background and Motivation
- Approach
- Assessment
- Contributions and Future Work

Relative Error Correction Cost in a Software Life Cycle



Specification must be tested!

No Existing Formal Language is Suitable for Testing SDL Specifications

- TTCN
- MSC
- UML
- URN/UCM
- LOTOS
- SDL

SDL Task Force

- The graphical representation, ensuring auto-layout is possible
- ***Test capabilities, such as SDL based test scripts***
- ASN.1(1994) support, including encoding/ decoding of PDUs
- Associated methodology issues, such as maximum integration of tool chain

Statement of Research Problem

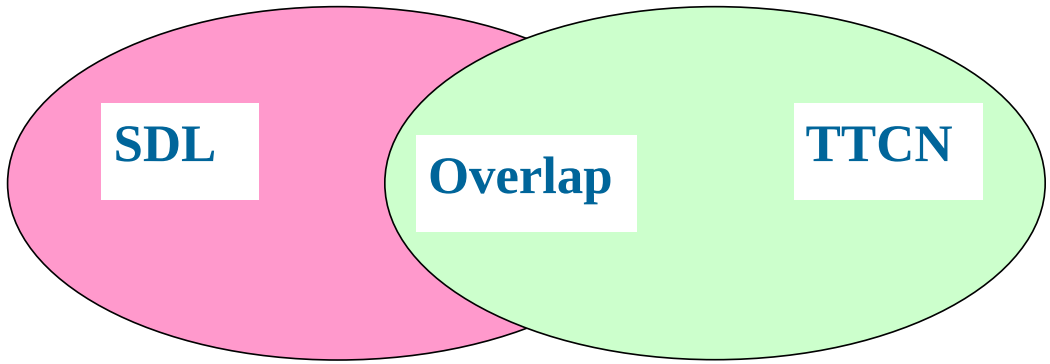
To define and investigate the applicability of a simple, useful and efficient language for describing tests of SDL specifications

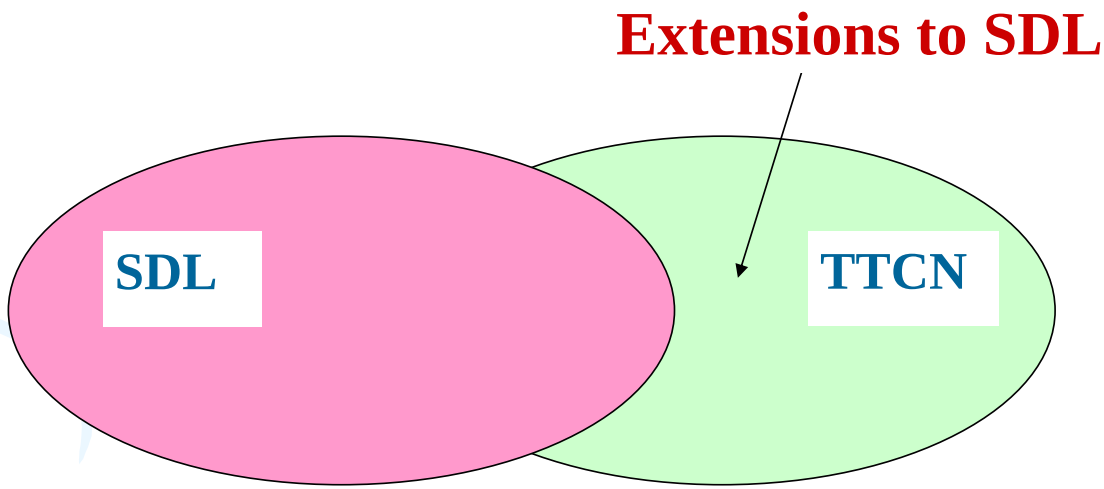
SIMPL-T
-- SDL Intended for Management and Planning of Tests

Approach

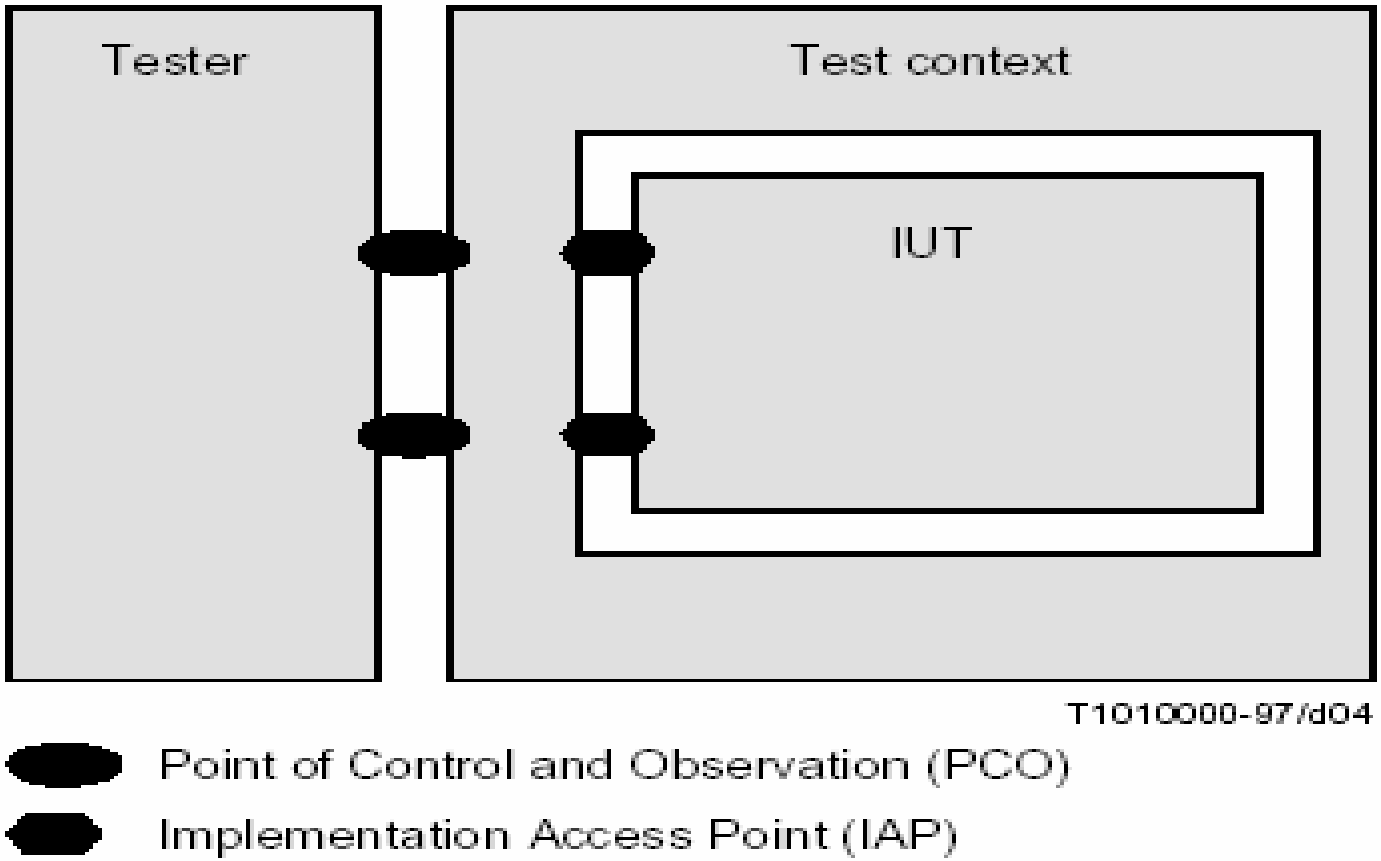
- Basic Testing Concepts
- Key Requirements
- Suitability of SDL for Test Specification
- SIMPL-T – SDL with Extensions

SDL & TTCN Overlap

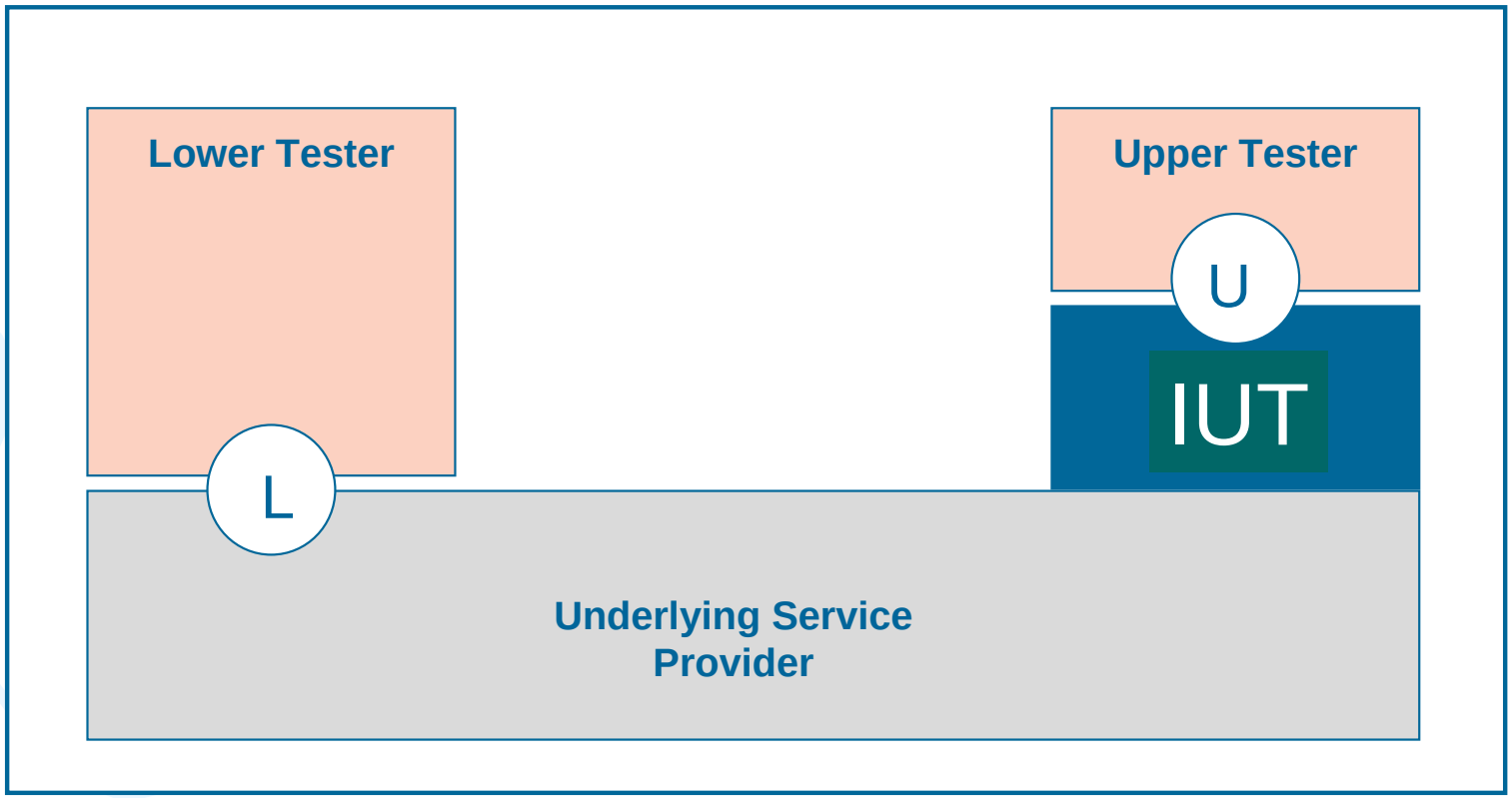




Test Architecture (ITU-T Z.500)



Test Architecture (ISO 9646)



* Test Configuration

Test Architecture

- Tester – Run test suite
- IUT
- Connection – PCOs
- Communication Channels

Test Case and Test Suite

- Test Suite
- Test Case
  - Test Purpose
  - Test Case Behaviour
    - Sending a stimulus to the IUT
    - Specifying expected response
    - Store and Transfer data
    - Take alternative actions
    - Repeated test steps or actions

Observations

- Check the responses
- Measure the timing of response
- Assign Verdict

Key Requirements

- Test Architecture – Tester and SUT
- Connection between the Tester and the SUT
- Communication between the Tester and the SUT
- Organization and Management of Tests
- Sending Stimuli to the IUT
- Receiving Response from the IUT
- Storing and Transferring Data
- Flow Control
- Test Step Repetition
- Checking Responses and Matching Mechanism
- Measuring the Timing of Responses.
- Assigning and Handling of Verdict

Suitability of SDL for Test Specification

Key Requirements	SDL Features
Test Architecture – Tester, SUT & Test Context	SDL Blocks
Test Architecture - Connecting between Tester and SUT (PCOs & IAPs)	Gate & Channel
Test Architecture - Communication between Tester and SUT	Signal Exchange
Organization and Management of Tests	**Not Supported**
Test Case Behaviour - Sending Stimuli to SUT	Output
Test Case Behaviour – Receiving Responses from SUT	Input
Test Case Behaviour - Storing and Transferring data	Variable & Data Type
Test Case Behaviour – Flow Control	Decision
Test Case Behaviour - Test Step Repetition	Procedure
Observation - Checking Responses	**Partially Supported**
Observation - Measuring the Timing of Responses	Timer
Assigning and Handling of Verdicts	**Not Supported**

Extensions

- Organization and Management of Tests
- Checking Responses
  -- “Input Via” and Matching mechanism
- Assigning and Handling of Verdicts

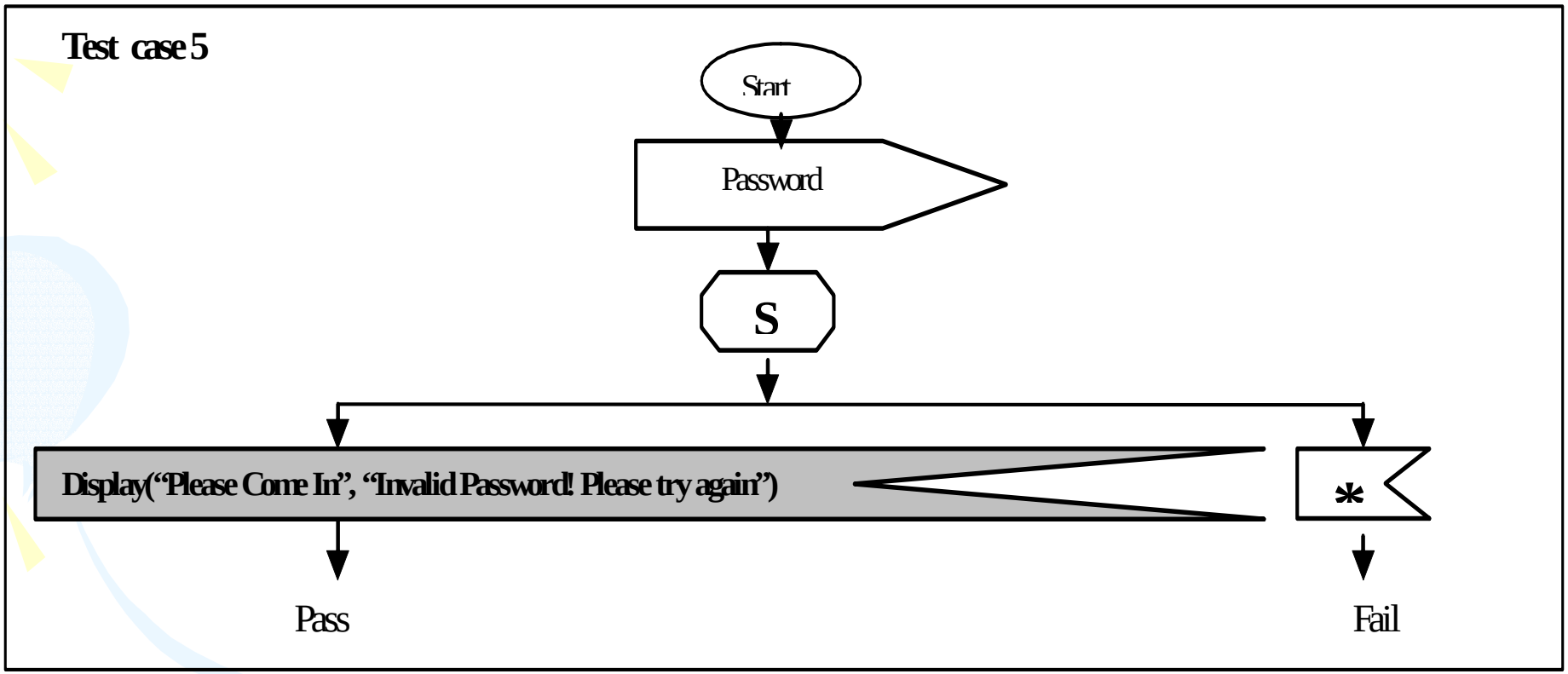
Organization and Management of Tests

```
Testsuite_Definition ::=    “TESTSUITE” TestsuiteName  “;”
                            [ Gate_Definition ]
                            [ Testsuite_Component ]
                            “ENDTESTSUITE;”
Gate_Definition ::=                 “GATE” GateName “;”
                            [ In_Signal_List ] “;”
                            [ Out_Signal_List ]”;”
In_Signal_List ::=                  Signal_Identifier
                            [ “,” In_Signal_List ]
Out_Signal_List ::=                 Signal_Identifier
                            [ “,” Out_Signal_List ]
Testsuite_Component ::= ([Signal_Definition]
                            [Signal_List_Definition]
                            [Timer_Definition]

                            ……
                            [Test_Group_Definition]
                            [Test_Case_Definition] )
                            [Testsuite_Component ]
Test_Group_Definition ::=           “TESTGROUP” TestGroupName “;”
                           Test_Case_Definition_List

                            ……

……
```

New INPUT VIA Construct

```
STATE S1;
    INPUT A VIA Gate1;
NEXTSTATE S2;
```

Specifying Expected Values of Parameters inside INPUT



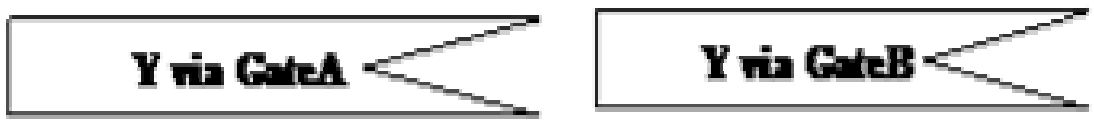
Matching Mechanism

- Unmatched Signal Handling:
  - Disregard by default
  - Explicitly use “Save” construct when necessary

Matching Mechanism

- Overlapped Signal Handling:

(1) the same signal arriving from different gates/channels;

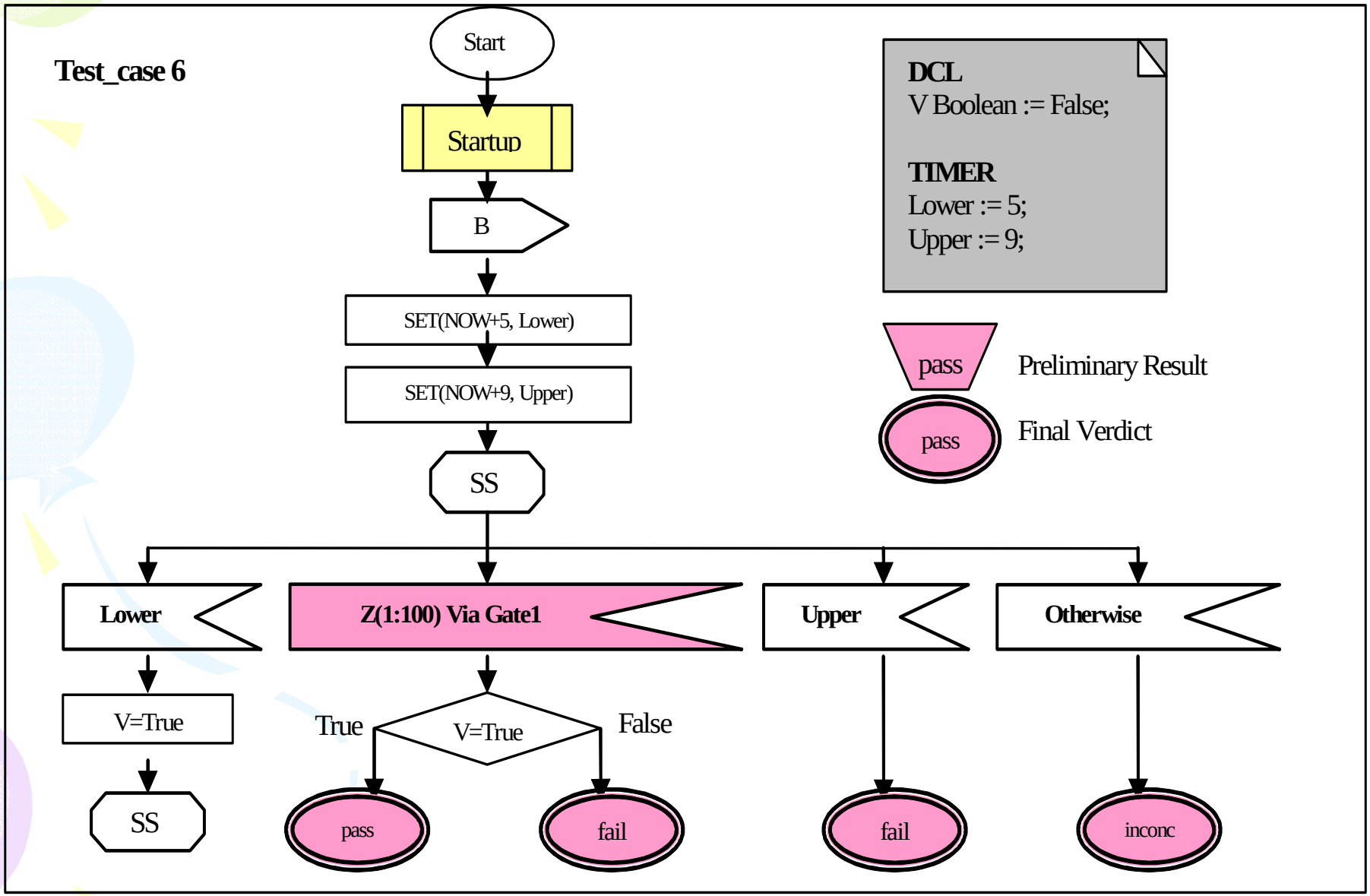


-- > They are not considered as overlap in SIMPL-T

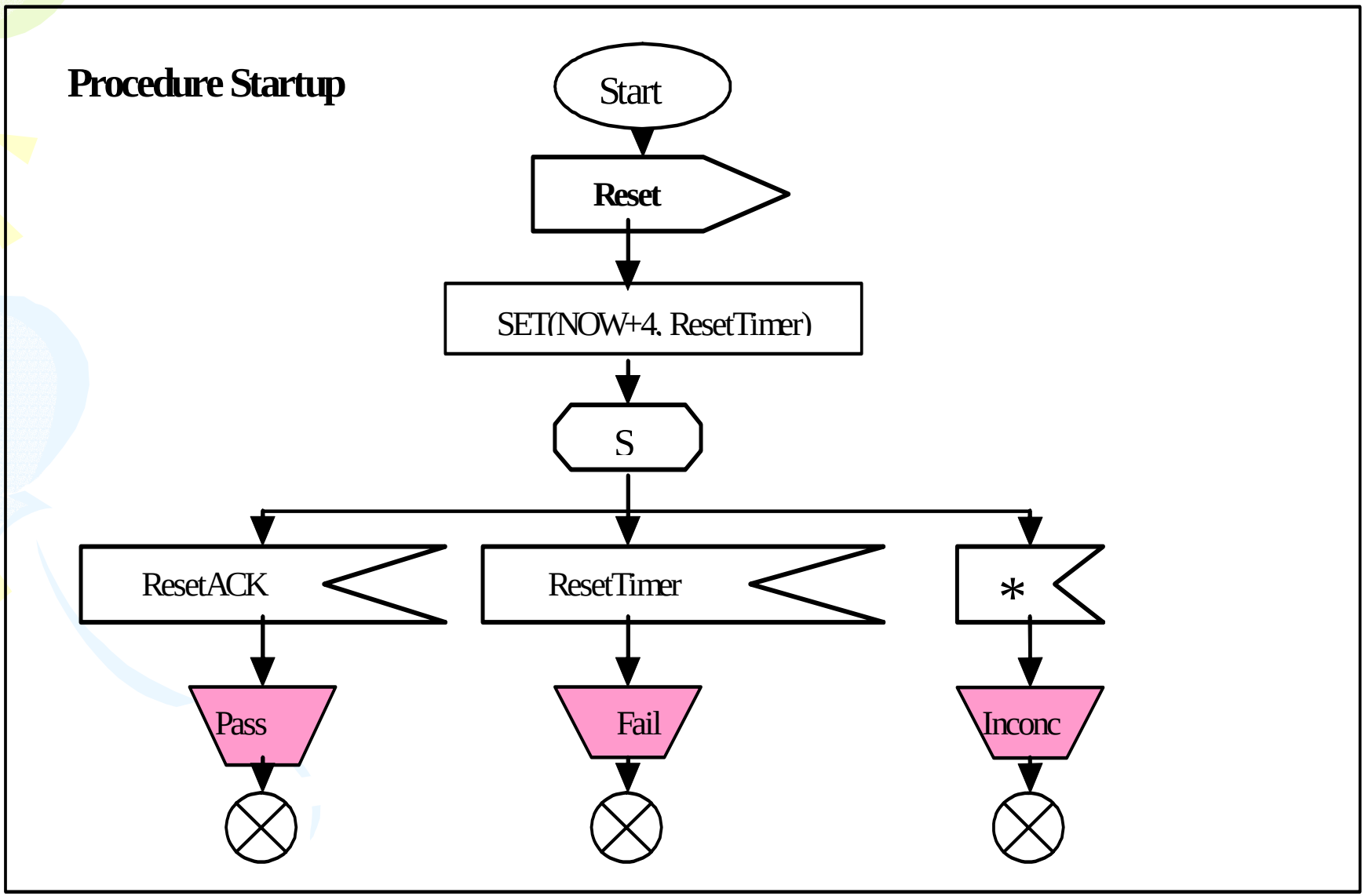
(2) the parameters carried by the same signal have different values and the values have overlap

-- > They are not allowed in SIMPL-T

An Example of a SIMPL-T Test Case



An Example of a SIMPL-T Test Case (Cont.)



The Strengths and Limitations of SIMPL-T Comparing to TTCN

\+ Strength - Weakness = Same / Not needed

Number	Description	SIMPL-T	TTCN
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			

Ordering Problem

- Two or more signals can arrive in arbitrary order
- The order is irrelevant,
- The test language does not have a mechanism to specify this situation

- SIMPL-T
  -- solve it using “save” construct

Contributions

- Submitted to the SDL Task Force
- Defined a simple, easy to learn test language
- Create a potential for lower cost tools
- Lead to more interest in SDL and testing

Future Work

- Concurrency
- Defaults
- Extensions for larger applications