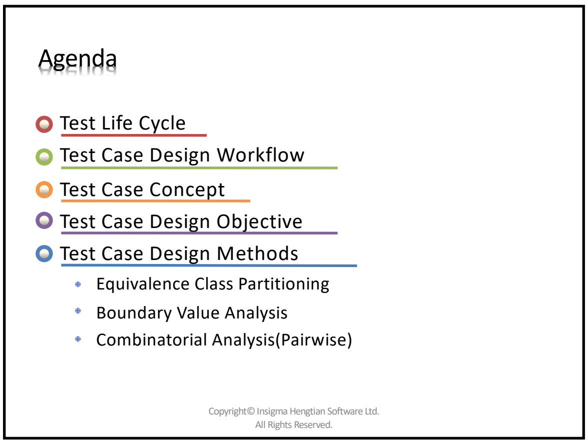
对美外包软件测试实践课程

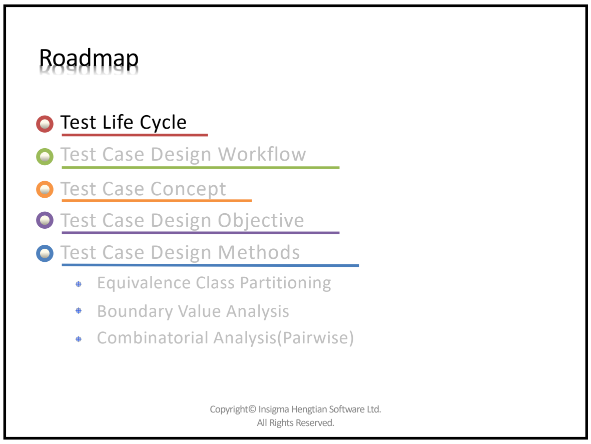
Test Case Design

Weifeng Lu
2012/05/14



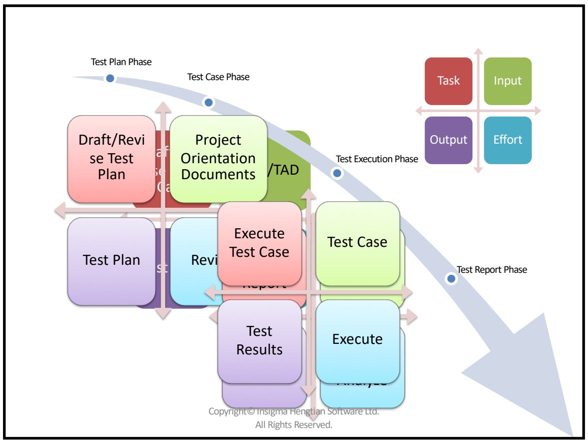
Agenda

- Test Life Cycle
- Test Case Design Workflow
- Test Case Concept
- Test Case Design Objective
- Test Case Design Methods
  - Equivalence Class Partitioning
  - Boundary Value Analysis
  - Combinatorial Analysis(Pairwise)



Roadmap

- Test Life Cycle
- Test Case Design Workflow
- Test Case Concept
- Test Case Design Objective
- Test Case Design Methods
  - Equivalence Class Partitioning
  - Boundary Value Analysis
  - Combinatorial Analysis(Pairwise)



Test Plan Phase

Test Case Phase

Test Execution Phase

Test Report Phase

Task

Input

Output

Effort

Draft/Revise Test Plan

Project Orientation Documents

/TAD

Test Plan

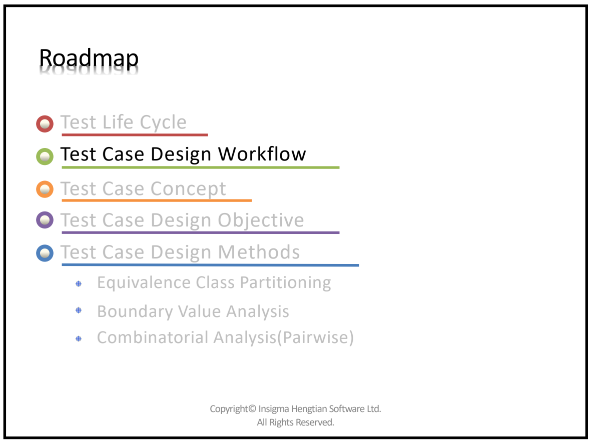
Revi

Execute Test Case

Test Case

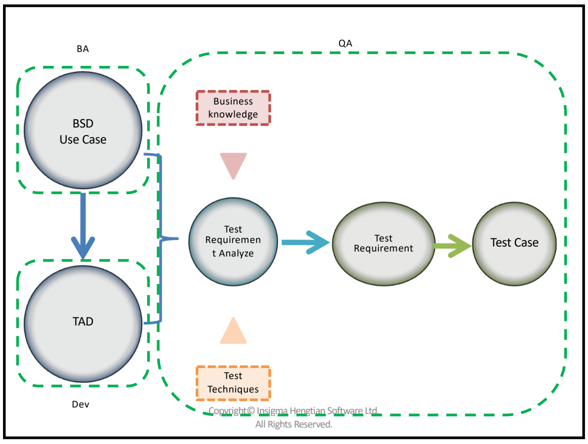
Test Results

Execute



Roadmap

- Test Life Cycle
- Test Case Design Workflow
- Test Case Concept
- Test Case Design Objective
- Test Case Design Methods
  - Equivalence Class Partitioning
  - Boundary Value Analysis
  - Combinatorial Analysis(Pairwise)



BA

QA

BSD
Use Case

TAD

Dev

Business knowledge

Test Requirement Analyze

Test Requirement

Test Case

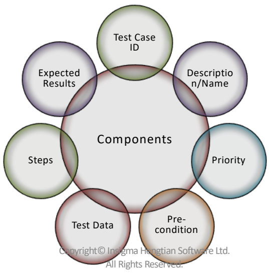
Test Techniques

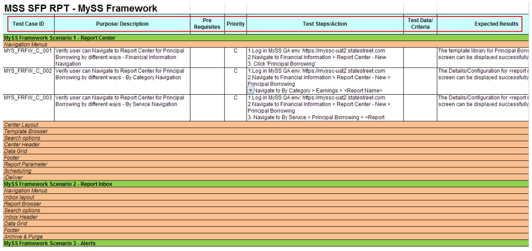
Roadmap

- Test Life Cycle
- Test Case Design Workflow
- Test Case Concept
- Test Case Design Objective
- Test Case Design Methods
    - Equivalence Class Partitioning
    - Boundary Value Analysis
    - Combinatorial Analysis(Pairwise)

A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.

IEEE Std 610-1990





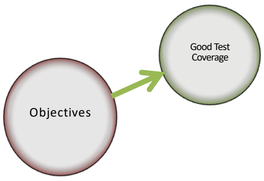
MSS SFP RPT - MySS Framework

Test Case ID	Purpose/ Description	Pre Requisites	Priority	Test Steps/Action	Test Data/ Criteria	Expected Results
MySS Framework Scenario 1 - Report Center						
Navigation Menus						
MYS_FRFW_C_001	Verify user can Navigate to Report Center for Principal Borrowing by different ways - Financial Information Navigation		C	1.Log in MySS QA env: https://myssc-uat2.statestreet.com 2.Navigate to Financial Information > Report Center - New 3. Click 'Principal Borrowing'		The template library for Principal Borro screen can be displayed successfully
MYS_FRFW_C_002	Verify user can Navigate to Report Center for Principal Borrowing by different ways - By Category Navigation		C	1.Log in MySS QA env: https://myssc-uat2.statestreet.com 2.Navigate to Financial Information > Report Center - New > Principal Borrowing Navigate to By Category > Earnings > <Report Name>		The Details/Configuration for <report r screen can be displayed successfully
MYS_FRFW_C_003	Verify user can Navigate to Report Center for Principal Borrowing by different ways - By Service Navigation		C	1.Log in MySS QA env: https://myssc-uat2.statestreet.com 2.Navigate to Financial Information > Report Center - New > Principal Borrowing 3. Navigate to By Service > Principal Borrowing > <Report		The Details/Configuration for <report r screen can be displayed successfully
Center Layout						
Template Browser						
Search options						
Center Header						
Data Grid						
Footer						
Report Parameter						
Scheduling						
iDeliver						
MySS Framework Scenario 2 - Report Inbox						
Navigation Menus						
Inbox layout						
Report Browser						
Search options						
Inbox Header						
Data Grid						
Footer						
Archive & Purge						
MySS Framework Scenario 3 - Alerts						

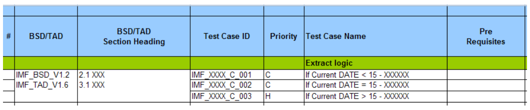
Roadmap

- Test Life Cycle
- Test Case Design Workflow
- Test Case Concept
- Test Case Design Objective
- Test Case Design Methods
    - Equivalence Class Partitioning
    - Boundary Value Analysis
    - Combinatorial Analysis(Pairwise)

Objectives



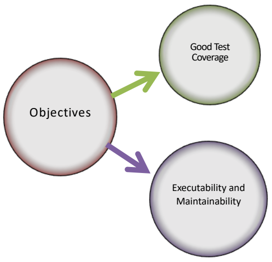
Traceability Matrix



#	BSD/TAD	BSD/TAD Section Heading	Test Case ID	Priority	Test Case Name	Pre Requisites
					Extract logic	
	IMF_BSD_V1.2 IMF_TAD_V1.6	2.1 XXX 3.1 XXX	IMF_XXXX_C_001	C	If Current DATE < 15 - XXXXXX	
			IMF_XXXX_C_002	C	If Current DATE = 15 - XXXXXX	
			IMF_XXXX_C_003	H	If Current DATE > 15 - XXXXXX	

Requirement Traceability Matrix is a two-way mapping between requirements and test cases, which provides confidence about test coverage

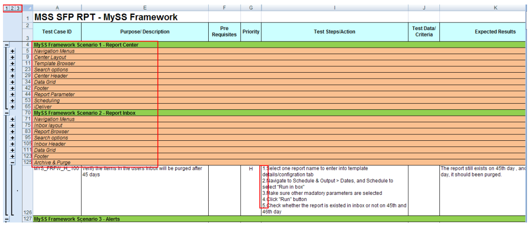
Objectives



Objectives

Good Test Coverage

Executability and Maintainability



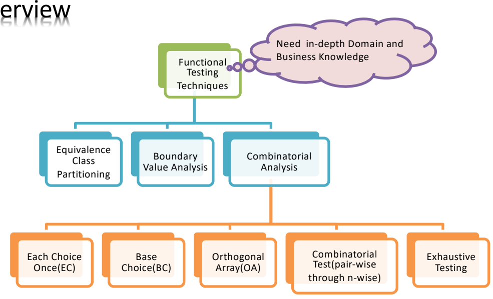
	A	E	F	G	I	J	K
1	MSS SFP RPT - MySS Framework						
2–3	Test Case ID	Purpose/ Description	Pre Requisites	Priority	Test Steps/Action	Test Data/ Criteria	Expected Results
4	MySS Framework Scenario 1 - Report Center						
5	Navigation Menus						
9	Center Layout						
11	Template Browser						
23	Search options						
29	Center Header						
34	Data Grid						
42	Footer						
44	Report Parameter						
53	Scheduling						
65	iDeliver						
70	MySS Framework Scenario 2 - Report Inbox						
71	Navigation Menus						
75	Inbox layout						
83	Report Browser						
95	Search options						
105	Inbox Header						
111	Data Grid						
123	Footer						
125	Archive & Purge						
126	MYS_FRFW_H_100	Verify the items in the users inbox will be purged after 45 days		H	1.Select one report name to enter into template details/configration tab 2.Navigate to Schedule & Output > Dates, and Schedule to select "Run in box" 3.Make sure other madatory parameters are selected 4.Click "Run" button 5.Check whether the report is existed in inbox or not on 45th and 46th day		The report still exists on 45th day , and day, it should been purged.
127	MySS Framework Scenario 3 - Alerts						

Test cases organizational structure and levels should be reasonable and the execution of the test cases should be smoothly to minimize test execution and maintainability costs.

Roadmap

- Test Life Cycle
- Test Case Design Workflow
- Test Case Concept
- Test Case Design Objective
- Test Case Design Methods
  - Equivalence Class Partitioning
  - Boundary Value Analysis
  - Combinatorial Analysis(Pairwise)

Overview



Functional Testing Techniques

Need in-depth Domain and Business Knowledge

Equivalence Class Partitioning

Boundary Value Analysis

Combinatorial Analysis

Each Choice Once(EC)

Base Choice(BC)

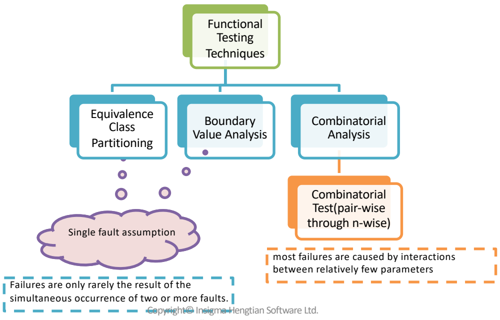
Orthogonal Array(OA)

Combinatorial Test(pair-wise through n-wise)

Exhaustive Testing

The value or effectiveness of using any test technique ultimately depends on the individual test's system and domain knowledge and ability to apply the applicable technique in the correct situation.

Theory



Functional Testing Techniques

Equivalence Class Partitioning

Boundary Value Analysis

Combinatorial Analysis

Single fault assumption

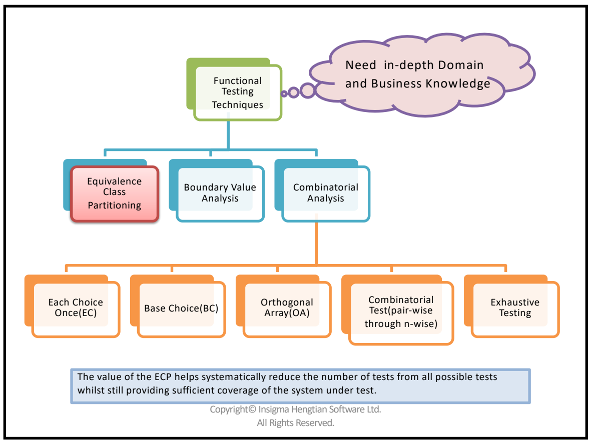
Combinatorial Test(pair-wise through n-wise)

most failures are caused by interactions between relatively few parameters

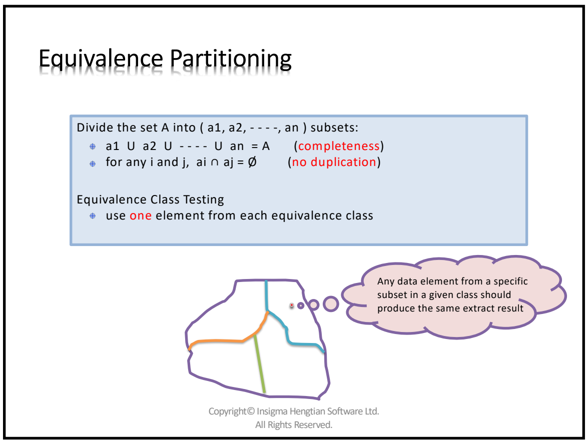
Failures are only rarely the result of the simultaneous occurrence of two or more faults.

Roadmap

- Test Life Cycle
- Test Case Design Workflow
- Test Case Concept
- Test Case Design Objective
- Test Case Design Methods
  - Equivalence Class Partitioning
  - Boundary Value Analysis
  - Combinatorial Analysis(Pairwise)


Functional Testing Techniques
Need in-depth Domain and Business Knowledge
Equivalence Class Partitioning
Boundary Value Analysis
Combinatorial Analysis
Each Choice Once(EC)
Base Choice(BC)
Orthogonal Array(OA)
Combinatorial Test(pair-wise through n-wise)
Exhaustive Testing

The value of the ECP helps systematically reduce the number of tests from all possible tests whilst still providing sufficient coverage of the system under test.



Equivalence Partitioning

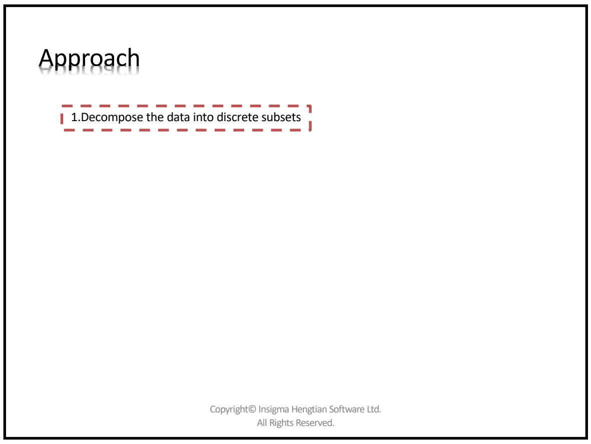
Divide the set A into (a1, a2, - - - -, an) subsets:

- a1 U a2 U - - - - U an = A (completeness)
- for any i and j, ai ∩ aj = Ø (no duplication)

Equivalence Class Testing

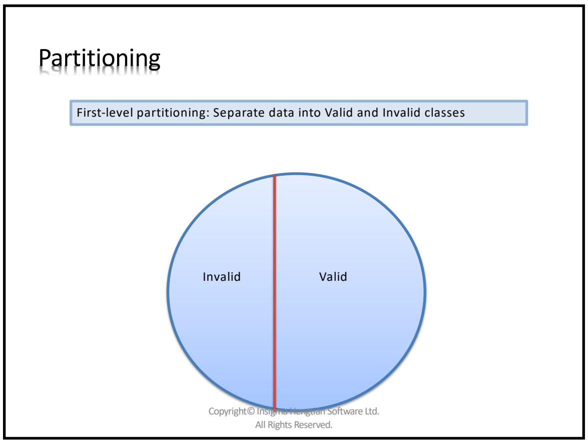
- use one element from each equivalence class

Any data element from a specific subset in a given class should produce the same extract result



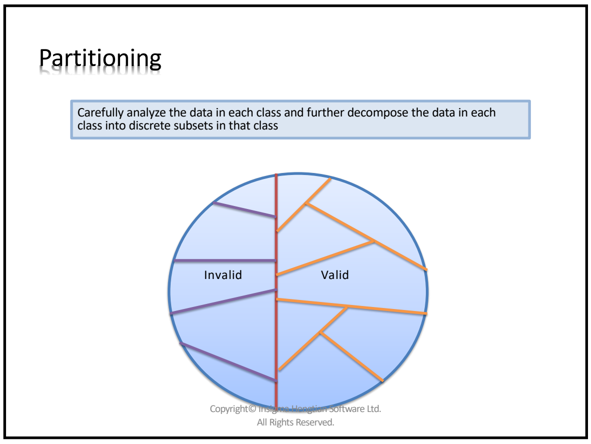
Approach

1.Decompose the data into discrete subsets



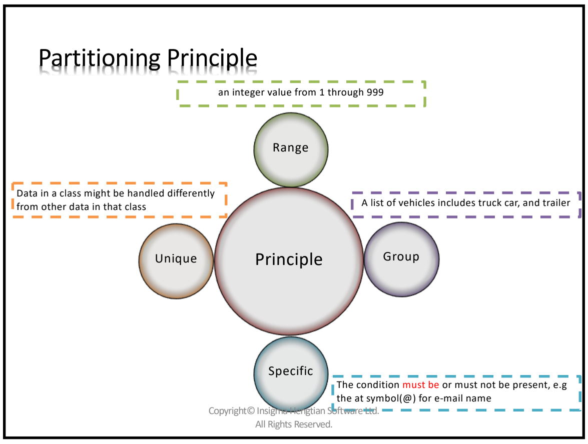
Partitioning

First-level partitioning: Separate data into Valid and Invalid classes



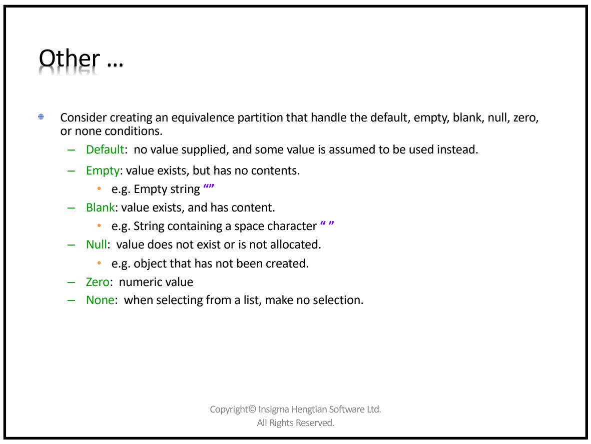
Partitioning

Carefully analyze the data in each class and further decompose the data in each class into discrete subsets in that class



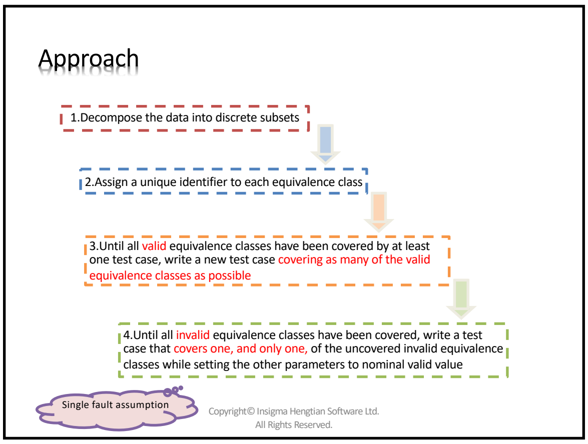
Partitioning Principle

an integer value from 1 through 999
Range
Data in a class might be handled differently from other data in that class
A list of vehicles includes truck car, and trailer
Unique
Principle
Group
Specific
The condition must be or must not be present, e.g the at symbol(@) for e-mail name



Other …

- Consider creating an equivalence partition that handle the default, empty, blank, null, zero, or none conditions.
  - Default: no value supplied, and some value is assumed to be used instead.
  - Empty: value exists, but has no contents.
    - e.g. Empty string “”
  - Blank: value exists, and has content.
    - e.g. String containing a space character “ ”
  - Null: value does not exist or is not allocated.
    - e.g. object that has not been created.
  - Zero: numeric value
  - None: when selecting from a list, make no selection.



Approach

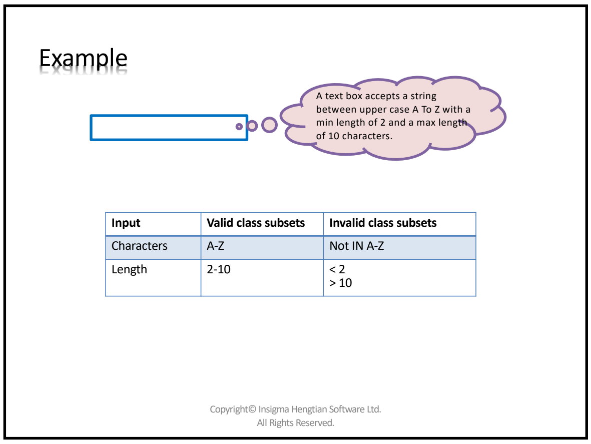
1.Decompose the data into discrete subsets

2.Assign a unique identifier to each equivalence class

3.Until all valid equivalence classes have been covered by at least one test case, write a new test case covering as many of the valid equivalence classes as possible

4.Until all invalid equivalence classes have been covered, write a test case that covers one, and only one, of the uncovered invalid equivalence classes while setting the other parameters to nominal valid value

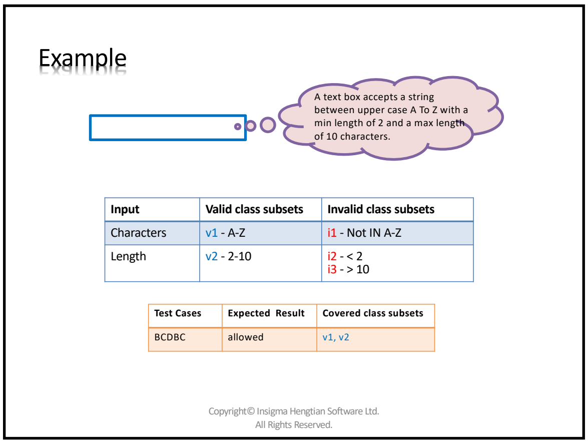
Single fault assumption



Example

A text box accepts a string between upper case A To Z with a min length of 2 and a max length of 10 characters.

Input	Valid class subsets	Invalid class subsets
Characters	A-Z	Not IN A-Z
Length	2-10	< 2 > 10

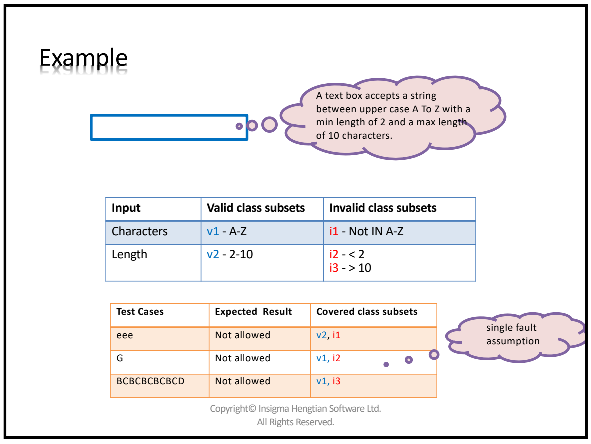


Example

A text box accepts a string between upper case A To Z with a min length of 2 and a max length of 10 characters.

Input	Valid class subsets	Invalid class subsets
Characters	v1 - A-Z	i1 - Not IN A-Z
Length	v2 - 2-10	i2 - < 2 i3 - > 10

Test Cases	Expected Result	Covered class subsets
BCDBC	allowed	v1, v2



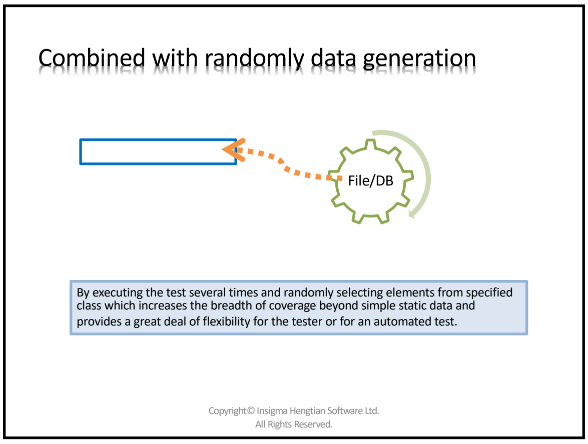
Example

A text box accepts a string between upper case A To Z with a min length of 2 and a max length of 10 characters.

Input	Valid class subsets	Invalid class subsets
Characters	v1 - A-Z	i1 - Not IN A-Z
Length	v2 - 2-10	i2 - < 2 i3 - > 10

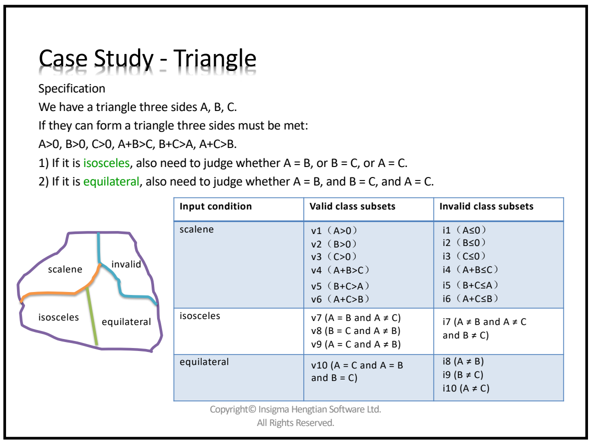
Test Cases	Expected Result	Covered class subsets
eee	Not allowed	v2, i1
G	Not allowed	v1, i2
BCBCBCBCBCD	Not allowed	v1, i3

single fault assumption



Combined with randomly data generation

By executing the test several times and randomly selecting elements from specified class which increases the breadth of coverage beyond simple static data and provides a great deal of flexibility for the tester or for an automated test.



Case Study - Triangle

Specification

We have a triangle three sides A, B, C.

If they can form a triangle three sides must be met:

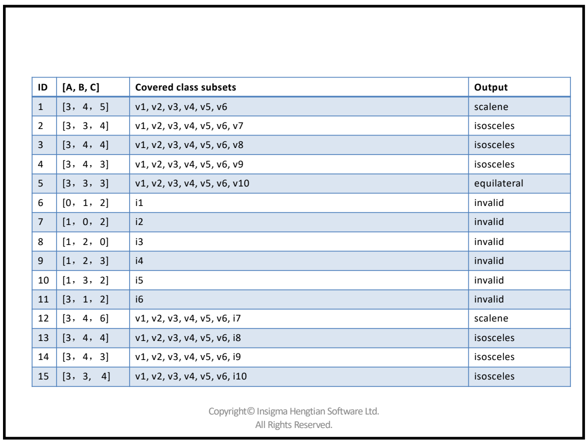
A>0, B>0, C>0, A+B>C, B+C>A, A+C>B.

1) If it is isosceles, also need to judge whether A = B, or B = C, or A = C.

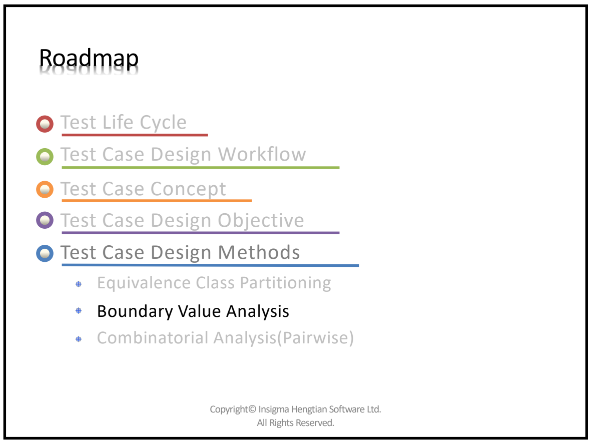
2) If it is equilateral, also need to judge whether A = B, and B = C, and A = C.

scalene
invalid
isosceles
equilateral

Input condition	Valid class subsets	Invalid class subsets
scalene	v1（A>0） v2（B>0） v3（C>0） v4（A+B>C） v5（B+C>A） v6（A+C>B）	i1（A≤0） i2（B≤0） i3（C≤0） i4（A+B≤C） i5（B+C≤A） i6（A+C≤B）
isosceles	v7 (A = B and A ≠ C) v8 (B = C and A ≠ B) v9 (A = C and A ≠ B)	i7 (A ≠ B and A ≠ C and B ≠ C)
equilateral	v10 (A = C and A = B and B = C)	i8 (A ≠ B) i9 (B ≠ C) i10 (A ≠ C)

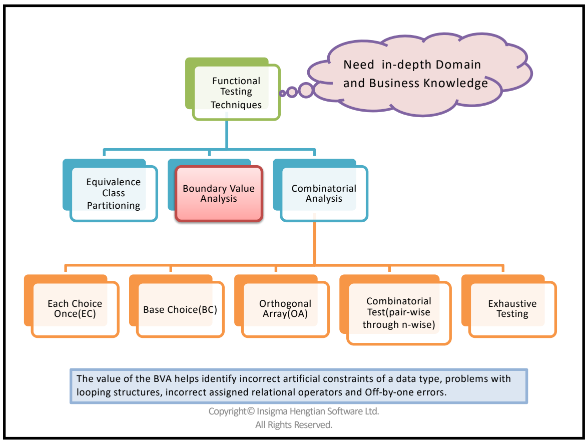


ID	[A, B, C]	Covered class subsets	Output
1	[3，4，5]	v1, v2, v3, v4, v5, v6	scalene
2	[3，3，4]	v1, v2, v3, v4, v5, v6, v7	isosceles
3	[3，4，4]	v1, v2, v3, v4, v5, v6, v8	isosceles
4	[3，4，3]	v1, v2, v3, v4, v5, v6, v9	isosceles
5	[3，3，3]	v1, v2, v3, v4, v5, v6, v10	equilateral
6	[0，1，2]	i1	invalid
7	[1，0，2]	i2	invalid
8	[1，2，0]	i3	invalid
9	[1，2，3]	i4	invalid
10	[1，3，2]	i5	invalid
11	[3，1，2]	i6	invalid
12	[3，4，6]	v1, v2, v3, v4, v5, v6, i7	scalene
13	[3，4，4]	v1, v2, v3, v4, v5, v6, i8	isosceles
14	[3，4，3]	v1, v2, v3, v4, v5, v6, i9	isosceles
15	[3，3, 4]	v1, v2, v3, v4, v5, v6, i10	isosceles



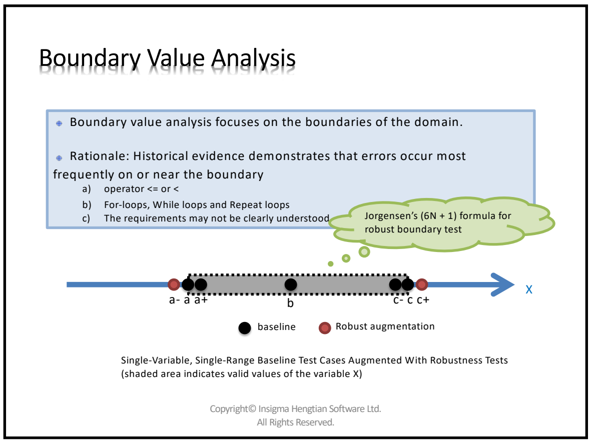
Roadmap

- Test Life Cycle
- Test Case Design Workflow
- Test Case Concept
- Test Case Design Objective
- Test Case Design Methods
  - Equivalence Class Partitioning
  - Boundary Value Analysis
  - Combinatorial Analysis(Pairwise)



Functional Testing Techniques
Need in-depth Domain and Business Knowledge
Equivalence Class Partitioning
Boundary Value Analysis
Combinatorial Analysis
Each Choice Once(EC)
Base Choice(BC)
Orthogonal Array(OA)
Combinatorial Test(pair-wise through n-wise)
Exhaustive Testing

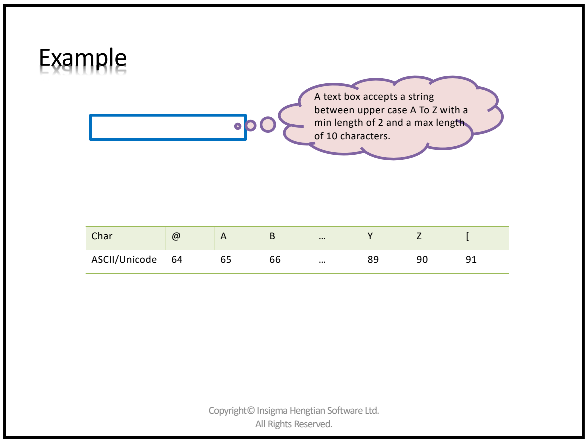
The value of the BVA helps identify incorrect artificial constraints of a data type, problems with looping structures, incorrect assigned relational operators and Off-by-one errors.



Boundary Value Analysis

- Boundary value analysis focuses on the boundaries of the domain.
- Rationale: Historical evidence demonstrates that errors occur most frequently on or near the boundary
  - a) operator <= or <
  - b) For-loops, While loops and Repeat loops
  - c) The requirements may not be clearly understood

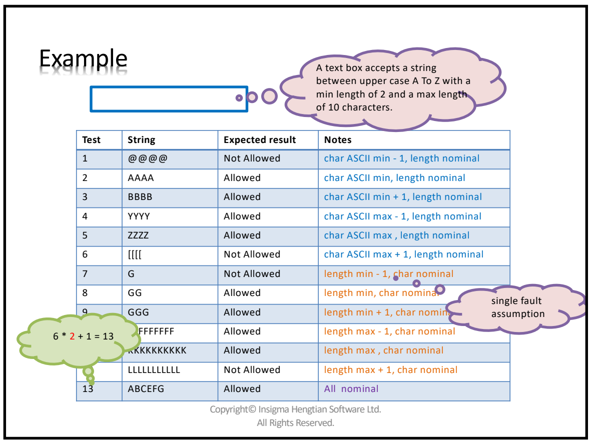
Single-Variable, Single-Range Baseline Test Cases Augmented With Robustness Tests (shaded area indicates valid values of the variable X)



Example

A text box accepts a string between upper case A To Z with a min length of 2 and a max length of 10 characters.

Char	@	A	B	…	Y	Z	[
ASCII/Unicode	64	65	66	…	89	90	91



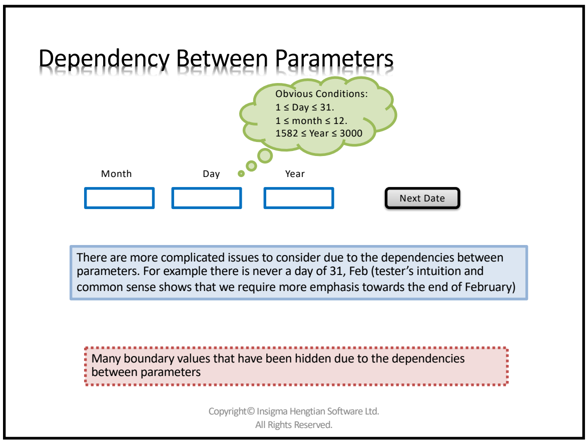
Example

A text box accepts a string between upper case A To Z with a min length of 2 and a max length of 10 characters.

Test	String	Expected result	Notes
1	@@@@	Not Allowed	char ASCII min - 1, length nominal
2	AAAA	Allowed	char ASCII min, length nominal
3	BBBB	Allowed	char ASCII min + 1, length nominal
4	YYYY	Allowed	char ASCII max - 1, length nominal
5	ZZZZ	Allowed	char ASCII max , length nominal
6	[[[[	Not Allowed	char ASCII max + 1, length nominal
7	G	Not Allowed	length min - 1, char nominal
8	GG	Allowed	length min, char nominal
9	GGG	Allowed	length min + 1, char nominal
[illegible]	FFFFFFF	Allowed	length max - 1, char nominal
[illegible]	KKKKKKKKKK	Allowed	length max , char nominal
[illegible]	LLLLLLLLLLL	Not Allowed	length max + 1, char nominal
13	ABCEFG	Allowed	All nominal

single fault assumption

6 * 2 + 1 = 13



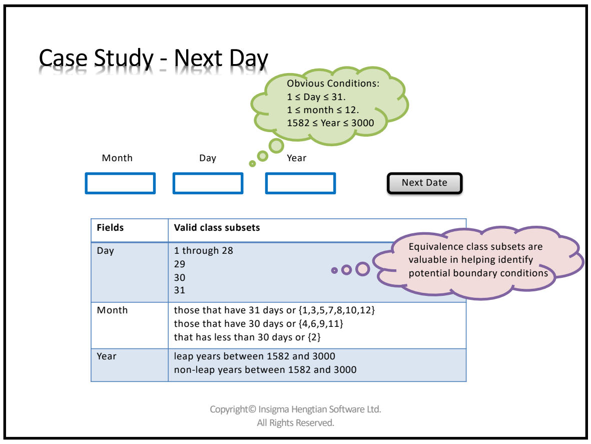
Dependency Between Parameters

Obvious Conditions:
1 ≤ Day ≤ 31.
1 ≤ month ≤ 12.
1582 ≤ Year ≤ 3000

Month
Day
Year
Next Date

There are more complicated issues to consider due to the dependencies between parameters. For example there is never a day of 31, Feb (tester's intuition and common sense shows that we require more emphasis towards the end of February)

Many boundary values that have been hidden due to the dependencies between parameters



Case Study - Next Day

Obvious Conditions:
1 ≤ Day ≤ 31.
1 ≤ month ≤ 12.
1582 ≤ Year ≤ 3000

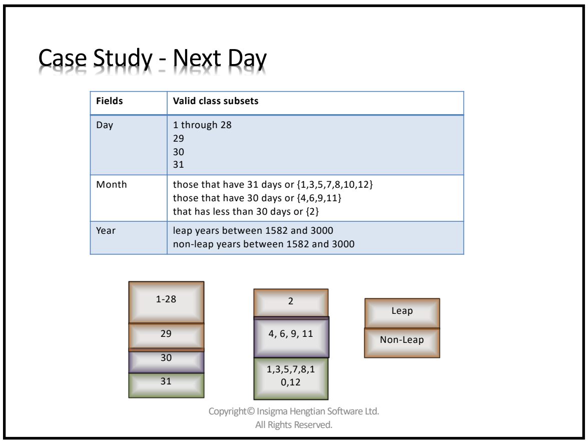
Month
Day
Year
Next Date

Fields	Valid class subsets
Day	1 through 28 29 30 31

Month	those that have 31 days or {1,3,5,7,8,10,12}
	those that have 30 days or {4,6,9,11}
	that has less than 30 days or {2}

Year	leap years between 1582 and 3000 non-leap years between 1582 and 3000

Equivalence class subsets are valuable in helping identify potential boundary conditions

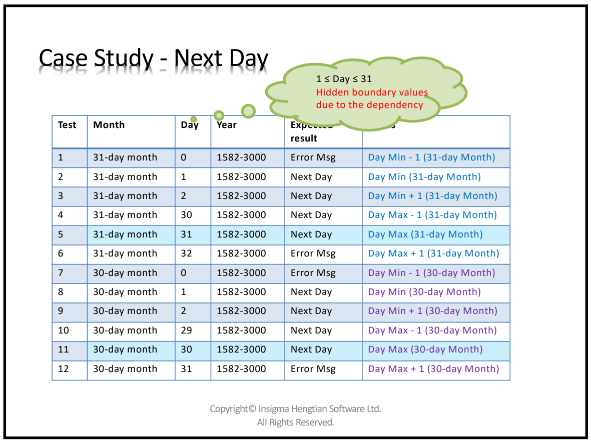


Case Study - Next Day

Fields	Valid class subsets
Day	1 through 28 29 30 31

Month	those that have 31 days or {1,3,5,7,8,10,12}
	those that have 30 days or {4,6,9,11}
	that has less than 30 days or {2}

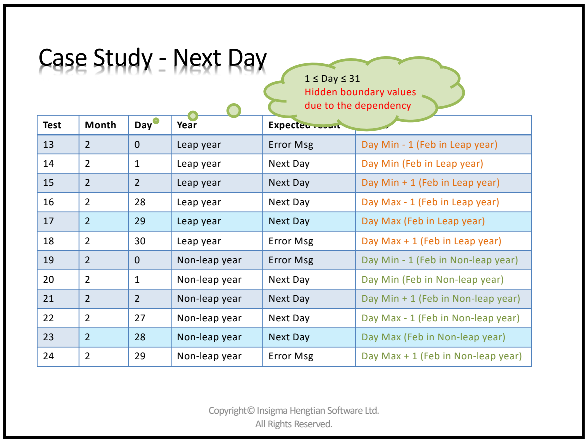
Year	leap years between 1582 and 3000 non-leap years between 1582 and 3000



Case Study - Next Day

1 ≤ Day ≤ 31
Hidden boundary values due to the dependency

Test	Month	Day	Year	Expected result	[illegible]
1	31-day month	0	1582-3000	Error Msg	Day Min - 1 (31-day Month)
2	31-day month	1	1582-3000	Next Day	Day Min (31-day Month)
3	31-day month	2	1582-3000	Next Day	Day Min + 1 (31-day Month)
4	31-day month	30	1582-3000	Next Day	Day Max - 1 (31-day Month)
5	31-day month	31	1582-3000	Next Day	Day Max (31-day Month)
6	31-day month	32	1582-3000	Error Msg	Day Max + 1 (31-day Month)
7	30-day month	0	1582-3000	Error Msg	Day Min - 1 (30-day Month)
8	30-day month	1	1582-3000	Next Day	Day Min (30-day Month)
9	30-day month	2	1582-3000	Next Day	Day Min + 1 (30-day Month)
10	30-day month	29	1582-3000	Next Day	Day Max - 1 (30-day Month)
11	30-day month	30	1582-3000	Next Day	Day Max (30-day Month)
12	30-day month	31	1582-3000	Error Msg	Day Max + 1 (30-day Month)



Case Study - Next Day

1 ≤ Day ≤ 31
Hidden boundary values due to the dependency

Test	Month	Day	Year	Expected result	[illegible]
13	2	0	Leap year	Error Msg	Day Min - 1 (Feb in Leap year)
14	2	1	Leap year	Next Day	Day Min (Feb in Leap year)
15	2	2	Leap year	Next Day	Day Min + 1 (Feb in Leap year)
16	2	28	Leap year	Next Day	Day Max - 1 (Feb in Leap year)
17	2	29	Leap year	Next Day	Day Max (Feb in Leap year)
18	2	30	Leap year	Error Msg	Day Max + 1 (Feb in Leap year)
19	2	0	Non-leap year	Error Msg	Day Min - 1 (Feb in Non-leap year)
20	2	1	Non-leap year	Next Day	Day Min (Feb in Non-leap year)
21	2	2	Non-leap year	Next Day	Day Min + 1 (Feb in Non-leap year)
22	2	27	Non-leap year	Next Day	Day Max - 1 (Feb in Non-leap year)
23	2	28	Non-leap year	Next Day	Day Max (Feb in Non-leap year)
24	2	29	Non-leap year	Error Msg	Day Max + 1 (Feb in Non-leap year)

Case Study - Next Day

1 ≤ month ≤ 12.

Test	Month	Day	Year	Expected result	Notes
25	0	1-28	1582-3000	Error Msg	Month Min - 1
26	1	1-28	1582-3000	Next Day	Month Min
27	2	1-28	1582-3000	Next Day	Month Min + 1
28	11	1-28	1582-3000	Next Day	Month Max - 1
29	12	1-28	1582-3000	Next Day	Month Max
30	13	1-28	1582-3000	Error Msg	Month Max + 1

Each month have 1-28th day

Case Study - Next Day

1582 ≤ Year ≤ 3000

Test	Month	Day	Year	Expected result	Notes
25	1-12	1-28	1581	Error Msg	Year Min - 1
26	1-12	1-28	1582	Next Day	Year Min
27	1-12	1-28	1583	Next Day	Year Min + 1
28	1-12	1-28	2999	Next Day	Year Max - 1
29	1-12	1-28	3000	Next Day	Year Max
30	1-12	1-28	3001	Error Msg	Year Max + 1

Each month have 1-28th day

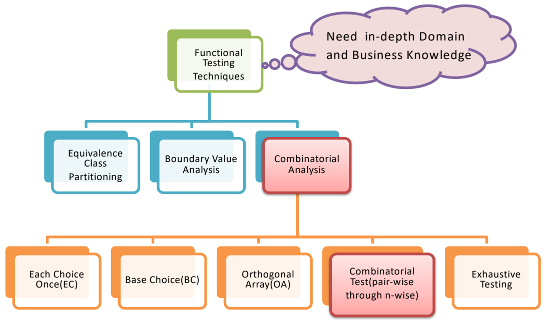
Case Study - Next Day

Output Boundary

Test	Month	Day	Year	Expected result	Notes
25	12	31	1581	Error Msg	Output Min - 1
26	1	1	1582	1/2/1582	Output Min
27	1	2	1583	1/3/1583	Output Min + 1
28	12	30	3000	12/31/3000	Output Max - 1
29	12	31	3000	1/1/3001	Output Max
30	1	1	3001	Error Msg	Output Max + 1

Roadmap

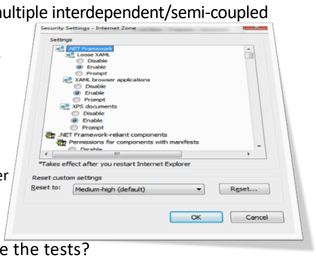
- Test Life Cycle
- Test Case Design Workflow
- Test Case Concept
- Test Case Design Objective
- Test Case Design Methods
  - Equivalence Class Partitioning
  - Boundary Value Analysis
  - Combinatorial Analysis(Pairwise)


Functional Testing Techniques
Need in-depth Domain and Business Knowledge
Equivalence Class Partitioning
Boundary Value Analysis
Combinatorial Analysis
Each Choice Once(EC)
Base Choice(BC)
Orthogonal Array(OA)
Combinatorial Test(pair-wise through n-wise)
Exhaustive Testing

The value of the CA helps identify most defects caused by variable interaction, provides greater structural coverage and reduce overall testing costs.

A common testing problem

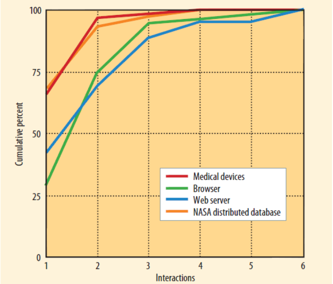
- The interactions between multiple interdependent/semi-coupled parameters with numerous
  - Variables per parameter are likely to cause bugs!
- Testing problem ...
  - 24 parameters
  - 2 - 5 variable per parameter
  - > 500,000,000,000
- So, how would you choose the tests?


Security Settings - Internet Zone
Settings
.NET Framework
Loose XAML
Disable
Enable
Prompt
XAML browser applications
Disable
Enable
Prompt
XPS documents
Disable
Enable
Prompt
.NET Framework-reliant components
Permissions for components with manifests
Disable
*Takes effect after you restart Internet Explorer
Reset custom settings
Reset to:
Medium-high (default)
Reset...
OK
Cancel

Testing challenge

- Design extremely small subset of tests from all possibilities
- Within a limited amount of time
- Provide a high degree of confidence

The basis behind combinatorial testing



Cumulative error detection rate for fault-triggering conditions. Many faults were caused by a single parameter value, a smaller proportion resulted from an interaction between two parameter values, and progressively fewer were triggered by three-, four-, five, and six -way interactions.

Understanding n-Way

Parameter	Values
OS	Windows, Linux
CPU	Intel, AMD
Protocol	IPv4, IPv6

All Pairwise	(Windows, Intel) (Windows, AMD) (Windows, IPv4) (Windows, IPv6)
	(Linux, Intel) (Linux, AMD) (Linux, IPv4) (Linux, IPv6)
	(Intel, IPv4) (Intel, IPv6)
	(AMD, IPv4) (AMD, IPv6)

Pairwise (2-way) test configurations.

Test Case	OS	CPU	Protocol
1	Windows	Intel	IPv4
2	Windows	AMD	IPv6
3	Linux	Intel	IPv6
4	Linux	AMD	IPv4

Parameter	Values
OS	Windows, Linux
Platform	x86, x64
CPU	Intel, AMD
Protocol	IPv4, IPv6

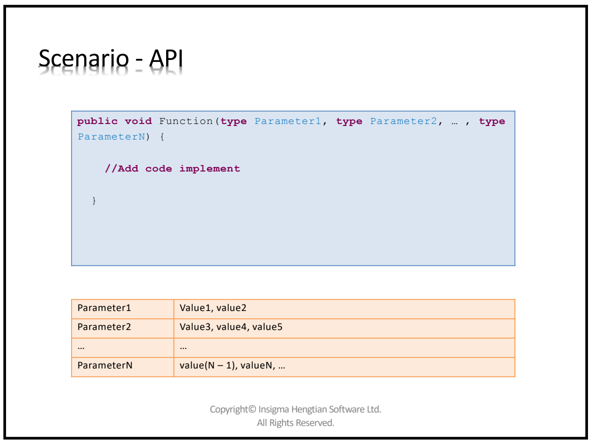
All 3-way combination	(Windows, x86, Intel) (Windows, x86, AMD) (Windows, x64, Intel) (Windows, x64, AMD) (Windows, x86, IPv4) (Windows, x86, IPv6) (Windows, x64, IPv4) (Windows, x64, IPv6) (Windows, Intel, IPv4) (Windows, Intel, IPv6) (Windows, AMD, IPv4) (Windows, AMD, IPv6)
	(Linux, x86, Intel) (Linux, x86, AMD) (Linux, x64, Intel) (Linux, x64, AMD) (Linux, x86, IPv4) (Linux, x86, IPv6) (Linux, x64, IPv4) (Linux, x64, IPv6) (Linux, Intel, IPv4) (Linux, Intel, IPv6) (Linux, AMD, IPv4) (Linux, AMD, IPv6)
	(x86, Intel, IPv4) (x86, Intel, IPv6) (x64, Intel, IPv4) (x64, Intel, IPv6)
	(x86, AMD, IPv4) (x86, AMD, IPv6) (x64, AMD, IPv4) (x64, AMD, IPv6)

3-way combination test configurations.

ID	OS	Platform	CPU	Protocol
1	Windows	x86	Intel	IPv4
2	Windows	x86	AMD	IPv6
3	Windows	x64	Intel	IPv6
4	Windows	x64	AMD	IPv4
5	Linux	x86	Intel	IPv6
6	Linux	x86	AMD	IPv4
7	Linux	x64	Intel	IPv4
8	Linux	x64	AMD	IPv6

Scenario - Configuration

Parameter	Number of Values	Values
OS	6	XP SP2, Vista SP1, 7, Server 2003 SP2, Server 2008 SP1, Server 2008 R2
Language	25	(using 3-letter language abbreviations) ARA, CHS, CHT, CSY, DAN, DEU, ELL, ENG, ESN, FIN, FRA, HEB, HUN, ITA, JPN, KOR, NLD, NOR, PLK, PSE, PTB, PTG, RUS, SVE, TRK
System Locale	2	Same as OS language, TRK
Flavor	2	Free, Checked
Platform	3	x86, x64, x64 wow
IE version	3	OS default, IE7, IE8
High DPI	2	120 DPI, 96 DPI
Theme	6	Native, Classic, Luna, Royale, Classic High Contrast, Aero
Side-by-side	9	3.5 SP1 + 4 (3.5 tests), 3.5 SP1 + 4 (4 tests), 3.5 SP1 + 4 - 4 (3.5 tests), 4 + Mock 4.5 (4 tests), 4 + Mock 5 (4 tests), 4 + 3.5 SP1 (4 tests), 4 + 3.5 SP1 (3.5 tests), 4 + 3.5 SP1 - 4 (3.5 tests), 4 (4 tests)



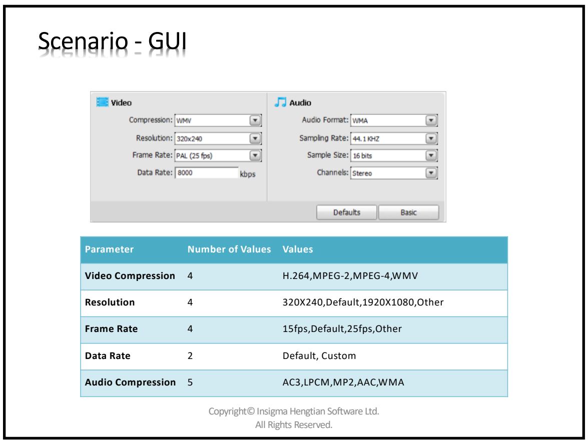
Scenario - API

```
public void Function(type Parameter1, type Parameter2, … , type
ParameterN) {

    //Add code implement

  }
```

Parameter1	Value1, value2
Parameter2	Value3, value4, value5
…	…
ParameterN	value(N – 1), valueN, …



Scenario - GUI

Compression: WMV

Resolution: 320x240

Frame Rate: PAL (25 fps)

Data Rate: 8000 kbps

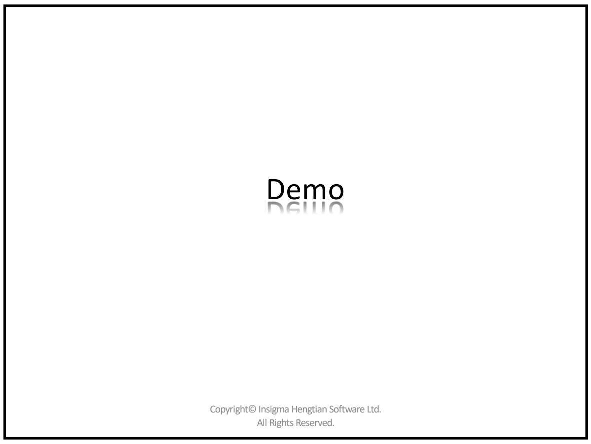
Audio Format: WMA

Sampling Rate: 44.1 KHZ

Sample Size: 16 bits

Channels: Stereo

Parameter	Number of Values	Values
Video Compression	4	H.264,MPEG-2,MPEG-4,WMV
Resolution	4	320X240,Default,1920X1080,Other
Frame Rate	4	15fps,Default,25fps,Other
Data Rate	2	Default, Custom
Audio Compression	5	AC3,LPCM,MP2,AAC,WMA



Demo

THANKS