#### Functional Testing

Software Engineering Andreas Zeller • Saarland University

## Testing

# 00:002 Testing

arte

## Even more Testing



## Software is



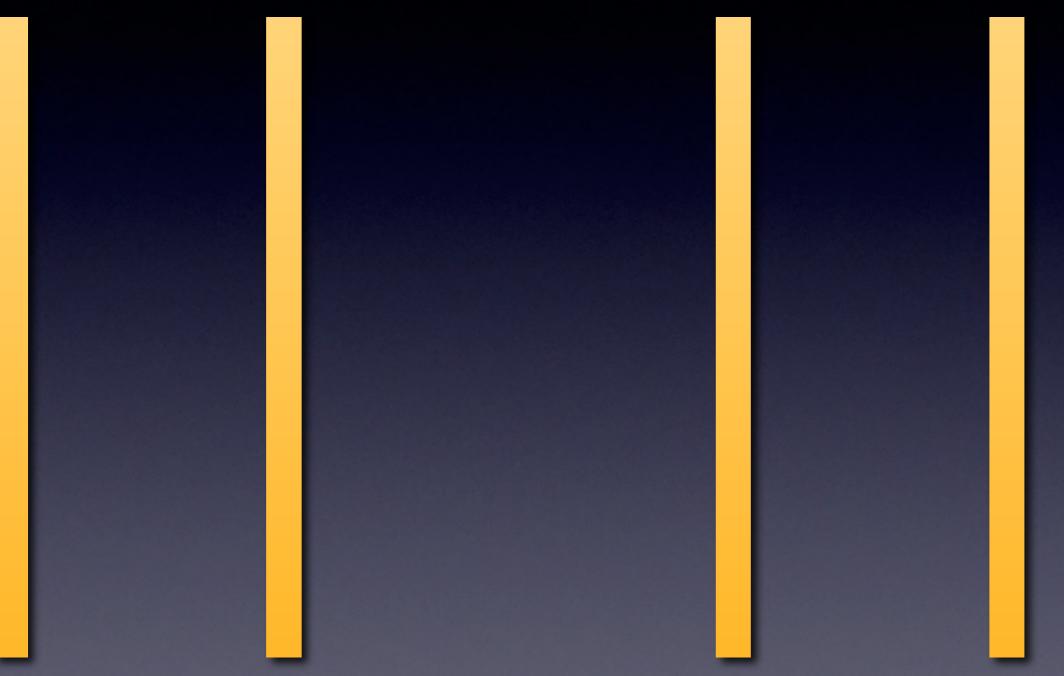
# Software is monoclassical structure is the second stru





## Testing

#### What to test?



### Dijkstra's Curse

Testing can only find the presence of errors, not their absence



Abstraction





#### Zeller's Variation on Diikstra

Verification can only find the *absence* of errors, but never their *presence* 

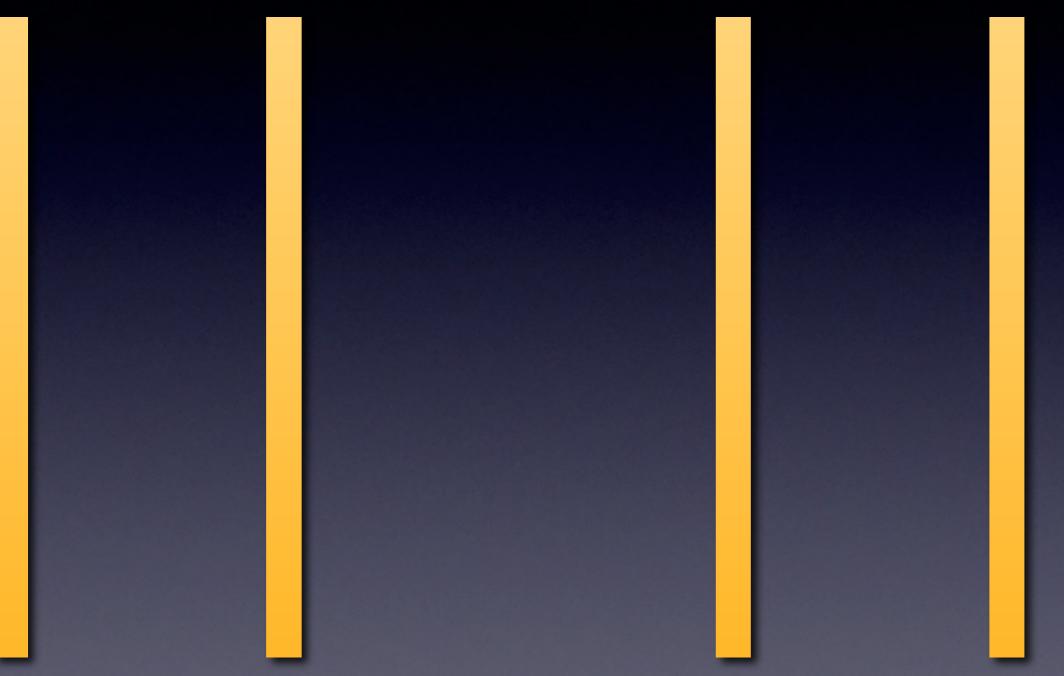


#### The Best of two Worlds

Abstraction



#### What to test?



#### Functional Testing

Software Engineering Andreas Zeller • Saarland University







### **Testing Tactics**

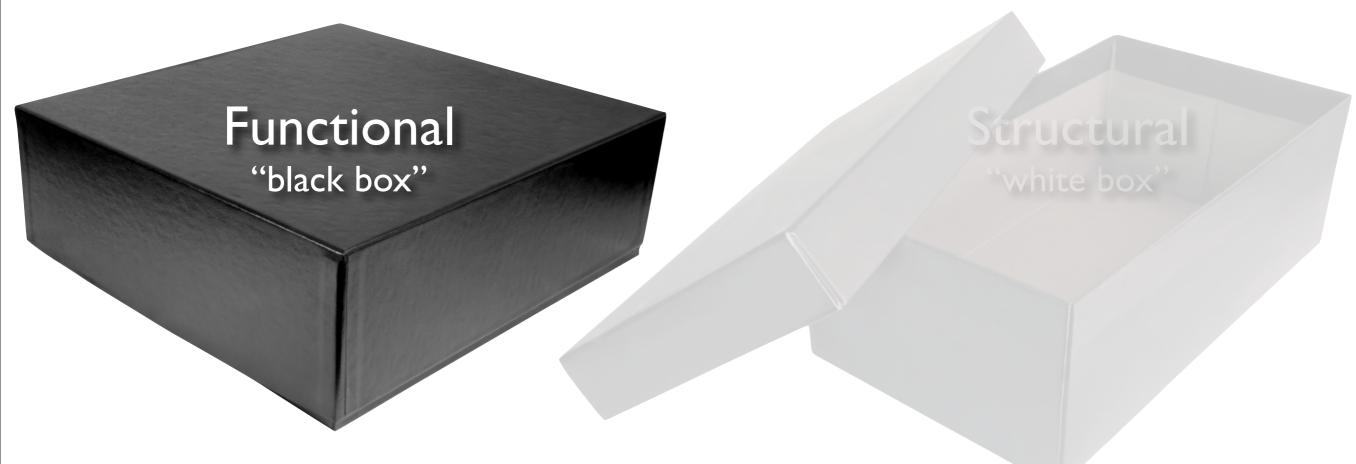
#### Functional "black box"

Structural "white box"

- Tests based on spec
- Test covers as much specified behavior as possible

- Tests based on code
- Test covers as much implemented behavior as possible

## Why Functional?



- Program code not necessary
- Early functional test design has benefits reveals spec problems • assesses testability • gives additional explanation of spec • may even serve as spec, as in XP

## Why Functional?





- Best for missing logic defects
   Common problem: Some program logic was simply forgotten Structural testing would not focus on code that is not there
- Applies at all granularity levels

   unit tests integration tests system tests regression tests

## Random Testing

- Pick possible inputs uniformly
- Avoids designer bias A real problem: The test designer can make the same logical mistakes and bad assumptions as the program designer (especially if they are the same person)
- But treats all inputs as equally valuable







### Infinite Monkey Theorem







 $2^{32} = 4.294.967.296$ different values

#### **FORCE<sup>232</sup> = 4.294.967.296** different values

#### $2^{32} = 4.294.967.296$ X $2^{32} = 4.294.967.296$ different values different values

#### 2<sup>64</sup> = 18.446.744.073.709.551.616 different runs



#### 18.446.744.073.709.551.616 Minutes

gadgets-club.com





#### 9.223.372.036.854.775.808 Minutes









#### 4.611.686.018.427.387.904 Minutes

#### 18.446.744.073.709.551.616





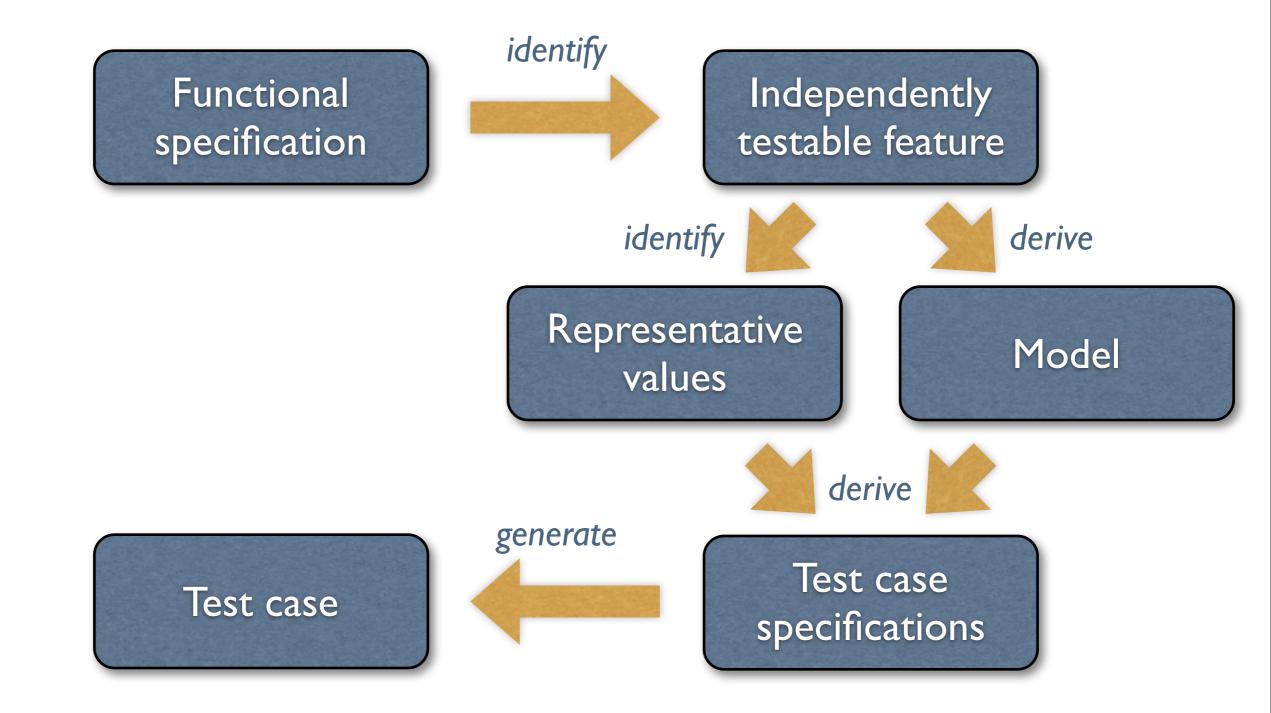
#### 1 Minute







### Systematic Functional Testing



#### **Testable Features**



- Decompose system into independently testable features (ITF)
- An ITF need not correspond to units or subsystems of the software
- For system testing, ITFs are exposed through user interfaces or APIs

#### **Testable Fatures**

```
class Roots {
   // Solve ax<sup>2</sup> + bx + c = 0
   public roots(double a, double b, double c)
   { ... }
   // Result: values for x
   double root_one, root_two;
```

}

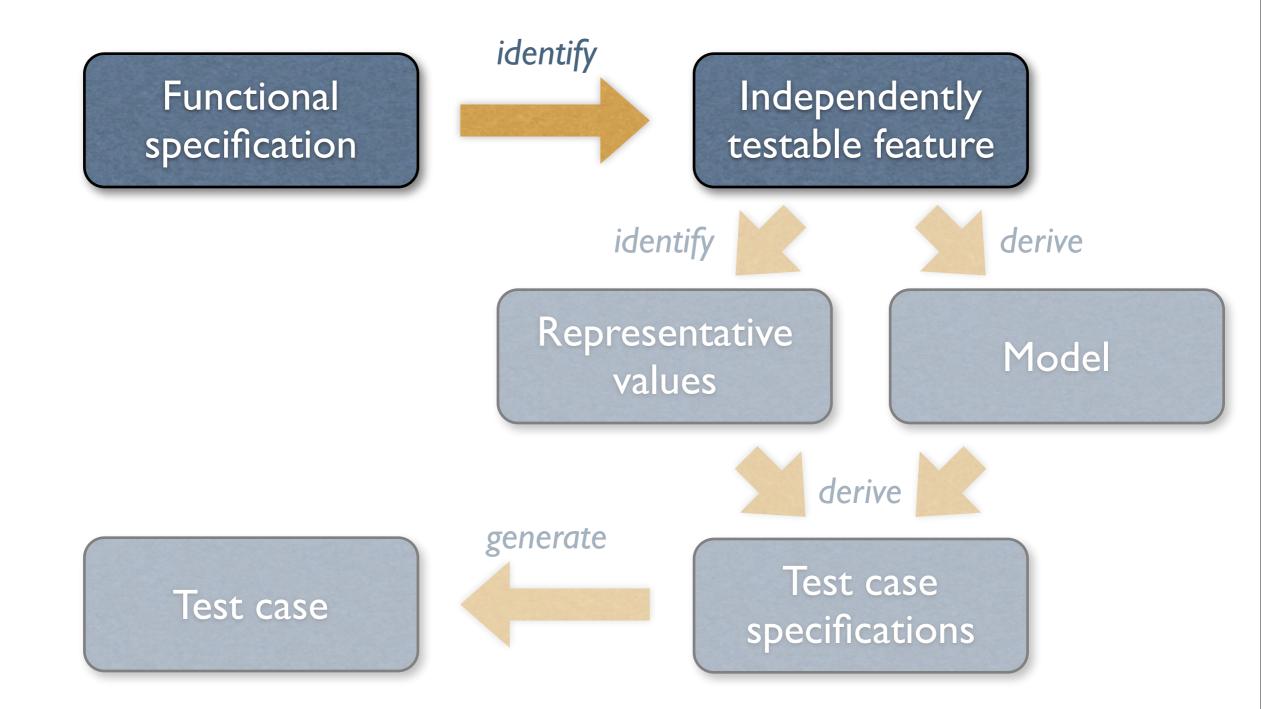
• What are the independently testable features?

### **Testable Fatures**

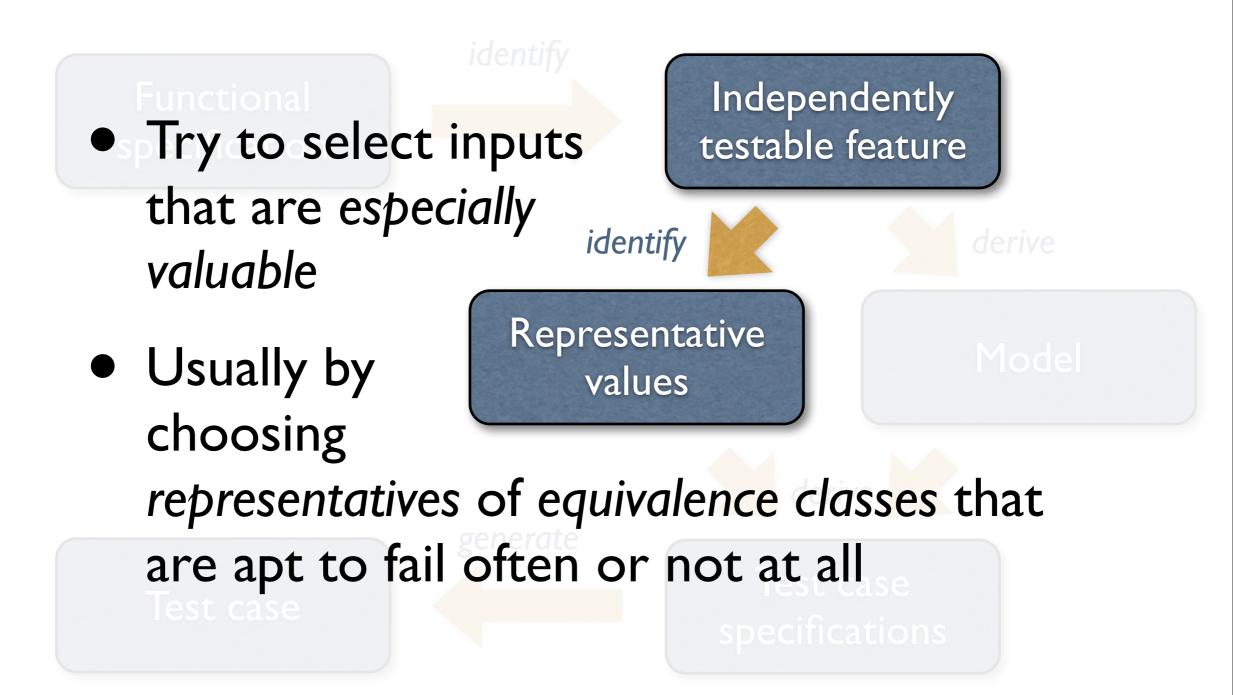
0 0	HP48-R (GX	)
	$\alpha$	
C HOME 3		
4:		
3:	2.510.58	8.971,09
- Zi		32,87
L · Waanalika	TRI LIST I HVP	28,88
	JB LEEUC LEE S MODES MEMOR	D COLE COLF
	UH UNDO PICTUI	JJUTJKUJL RE VIEW SWAP
· STO		
		7 10 <sup>×</sup> LOG e <sup>×</sup> LN
SIN <sub>s</sub> COS	$S_{T}$ TAN $\sqrt{x}$	$y_{\rm v} y_{\rm H} \frac{y_{\rm X}}{y_{\rm H}} \frac{1/x_{\rm X}}{x_{\rm X}}$
EQUATION MATE		
ENTER	, E₽	
	DLVE PLOT	SYMBOLIC () #
α	7 8	9 [÷]
5	4 5	
	I/O LIBRARY	
	1 2	3 –
CONT OFF =	→, →	π 4 {} ::
ON	0	SPC +

- Consider a multi-function calculator
- What are the independently testable features?

#### **Testable Features**



## **Representative Values**



#### Needles in a Haystack

- To find needles, look systematically
- We need to find out what makes needles special

#### Systematic Partition Testing Failures are sparse in ... but dense in some Failure (valuable test case) the space of possible parts of the space $\Box$ No failure inputs ... (the haystack) If we systematically test some Functional testing is one way of cases from each part, we will drawing orange lines to isolate include the dense parts regions with likely failures

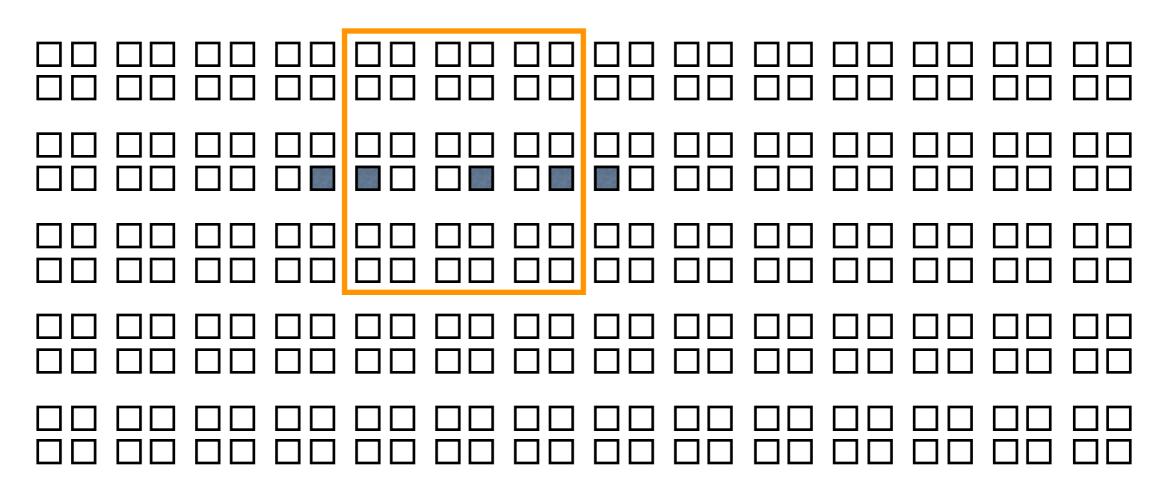
The space of possible input values

# Equivalence Partitioning

Input condition	Equivalence classes
range	one valid, two invalid (larger and smaller)
specific value	one valid, two invalid (larger and smaller)
member of a set	one valid, one invalid
boolean	one valid, one invalid

# Boundary Analysis

#### Possible test case



 Test at lower range (valid and invalid), at higher range (valid and invalid), and at center

# Example: ZIP Code

	NITED STATES POSTAL SERVICE.
Ser. S.	ZIP Code Lookup
	Search By Address 🔉 Search By City 🔉
	Find a list of cities that are in a ZIP Code.
	* Required Fields
	* ZIP Code 12345
	Submit >
Ļ	

- Input:
   5-digit ZIP code
- Output:
   list of cities
- What are representative values to test?

## Valid ZIP Codes

	POSTAL SERVICE.
Children of the second	Search By Address       Search By City >>
	Find a list of cities that are in a ZIP Code. * Required Fields * ZIP Code 12345
	Submit >

- I. with 0 cities
   as output
   (0 is boundary value)
- 2. with I city as output
- 3. with many cities as output

## Invalid ZIP Codes

	NITED STATES POSTAL SERVICE.
No.5	ZIP Code Lookup
	Search By Address 🔊 Search By City 🔊
	Find a list of cities that are in a ZIP Code.
	* Required Fields
	* ZIP Code 12345
	Submit >

- 4. empty input
- 5. I-4 characters (4 is boundary value)
- 6. 6 characters (6 is boundary value)
- 7. very long input
- 8. no digits
- 9. non-character data

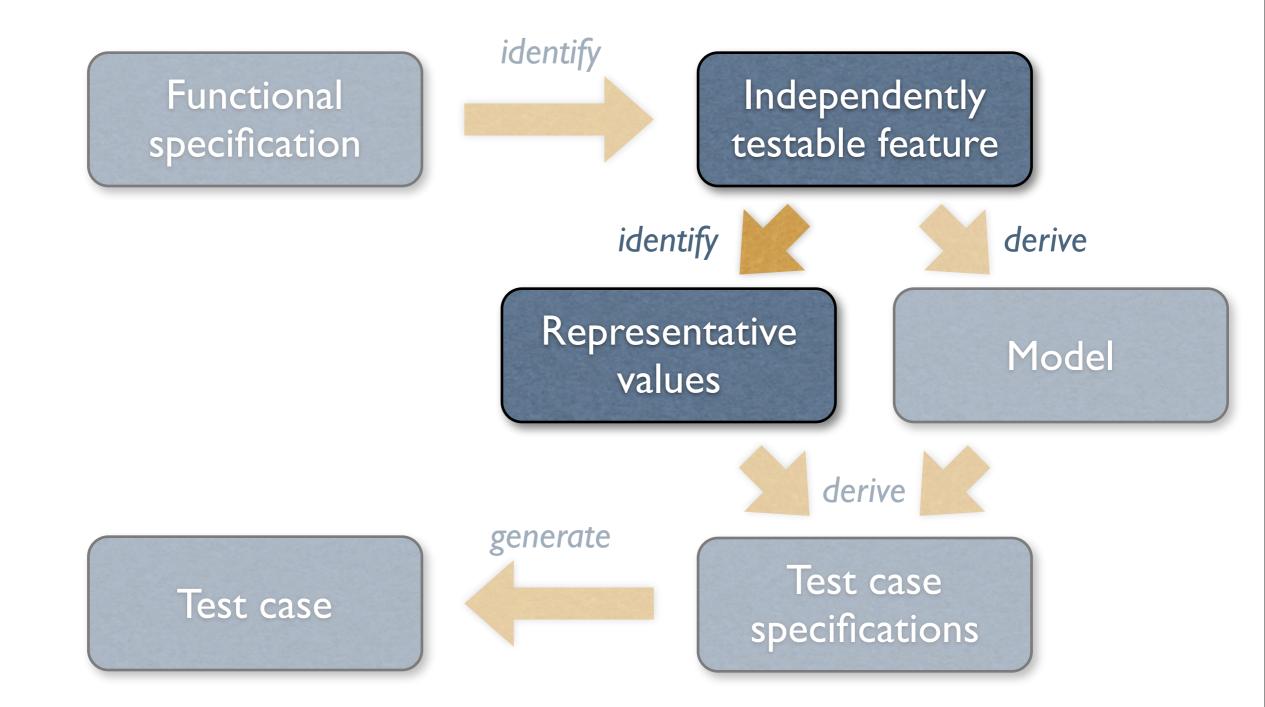
## "Special" ZIP Codes

- How about a ZIP code that reads
  - 12345'; DROP TABLE orders; SELECT
    \* FROM zipcodes WHERE 'zip' = '
- Or a ZIP code with 65536 characters...
- This is security testing

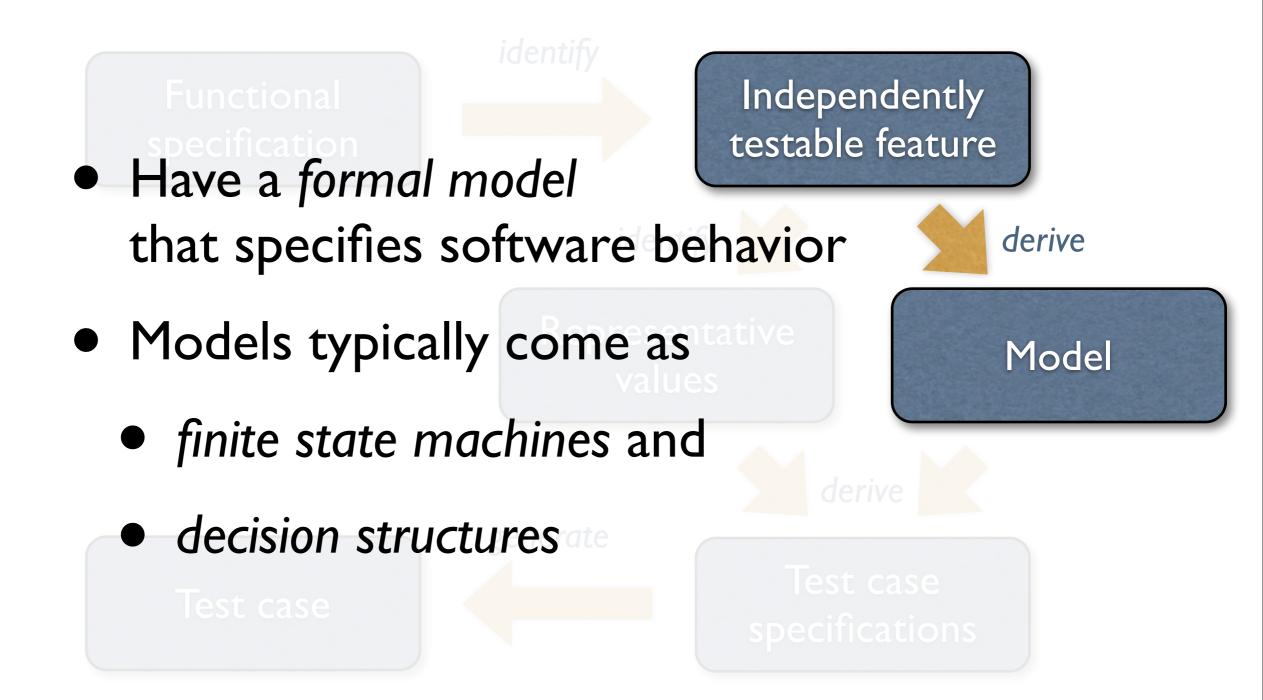
## Gutjahr's Hypothesis

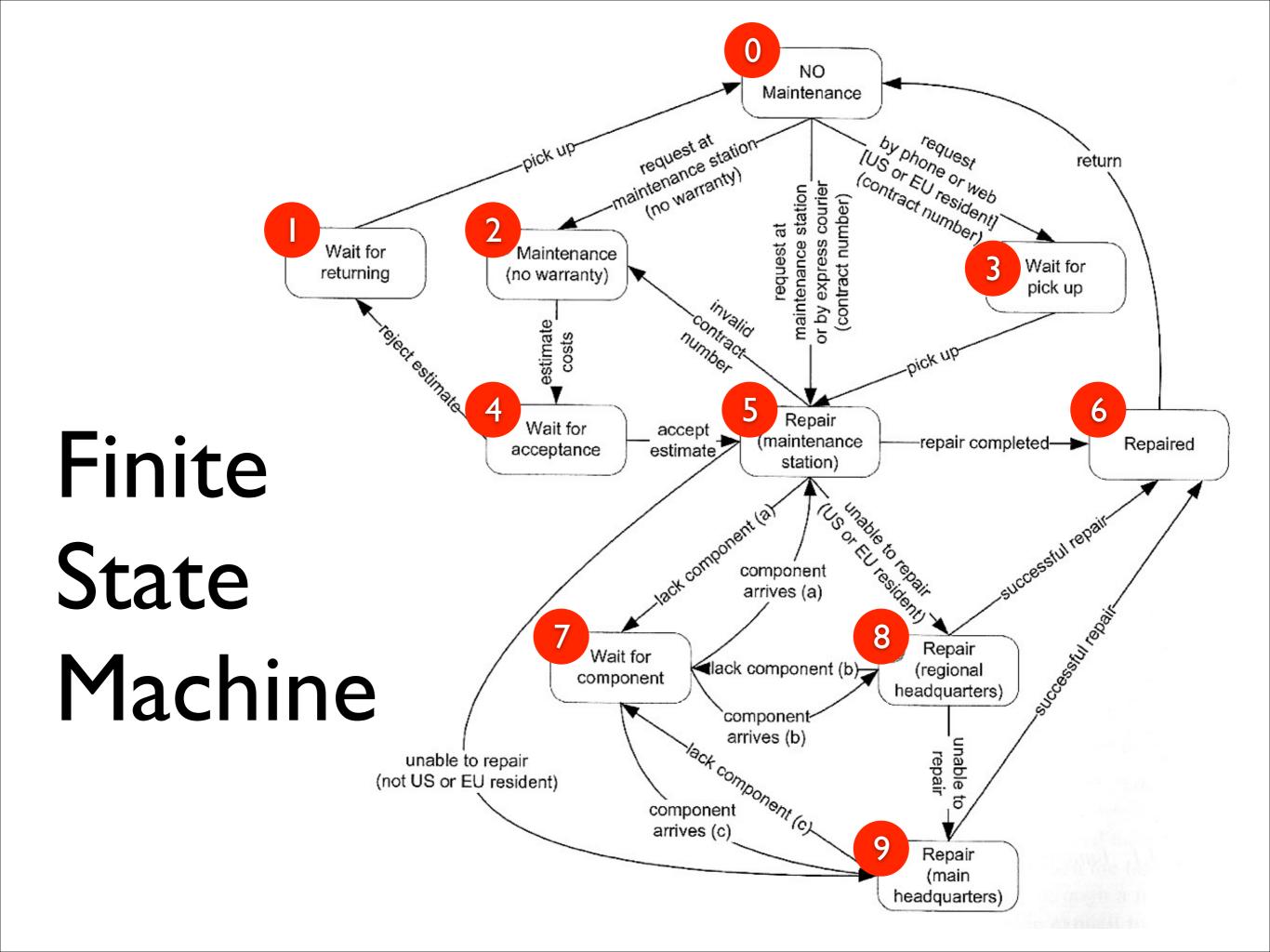
Partition testing is more effective than random testing.

#### **Representative Values**



#### Model-Based Testing





Maintenance: The Maintenance function records the history of items undergoing maintenance.

If the product is covered by warranty or maintenance contract, maintenance can be requested either by calling the maintenance toll free number, or through the Web site, or by bringing the item to a designated maintenance station.

If the maintenance is requested by phone or Web site and the customer is a US or EU resident, the item is picked up at the customer site, otherwise, the customer shall ship the item with an express courier.

If the maintenance contract number provided by the customer is not valid, the item follows the procedure for items not covered by warranty.

If the product is not covered by warranty or maintenance contract, maintenance can be requested only by bringing the item to a maintenance station. The maintenance station informs the customer of the estimated costs for repair. Maintenance starts only when the customer accepts the estimate. If the customer does not accept the estimate, the product is returned to the customer.

Small problems can be repaired directly at the maintenance station. If the maintenance station cannot solve the problem, the product is sent to the maintenance regional headquarters (if in US or EU) or to the maintenance main headquarters (otherwise).

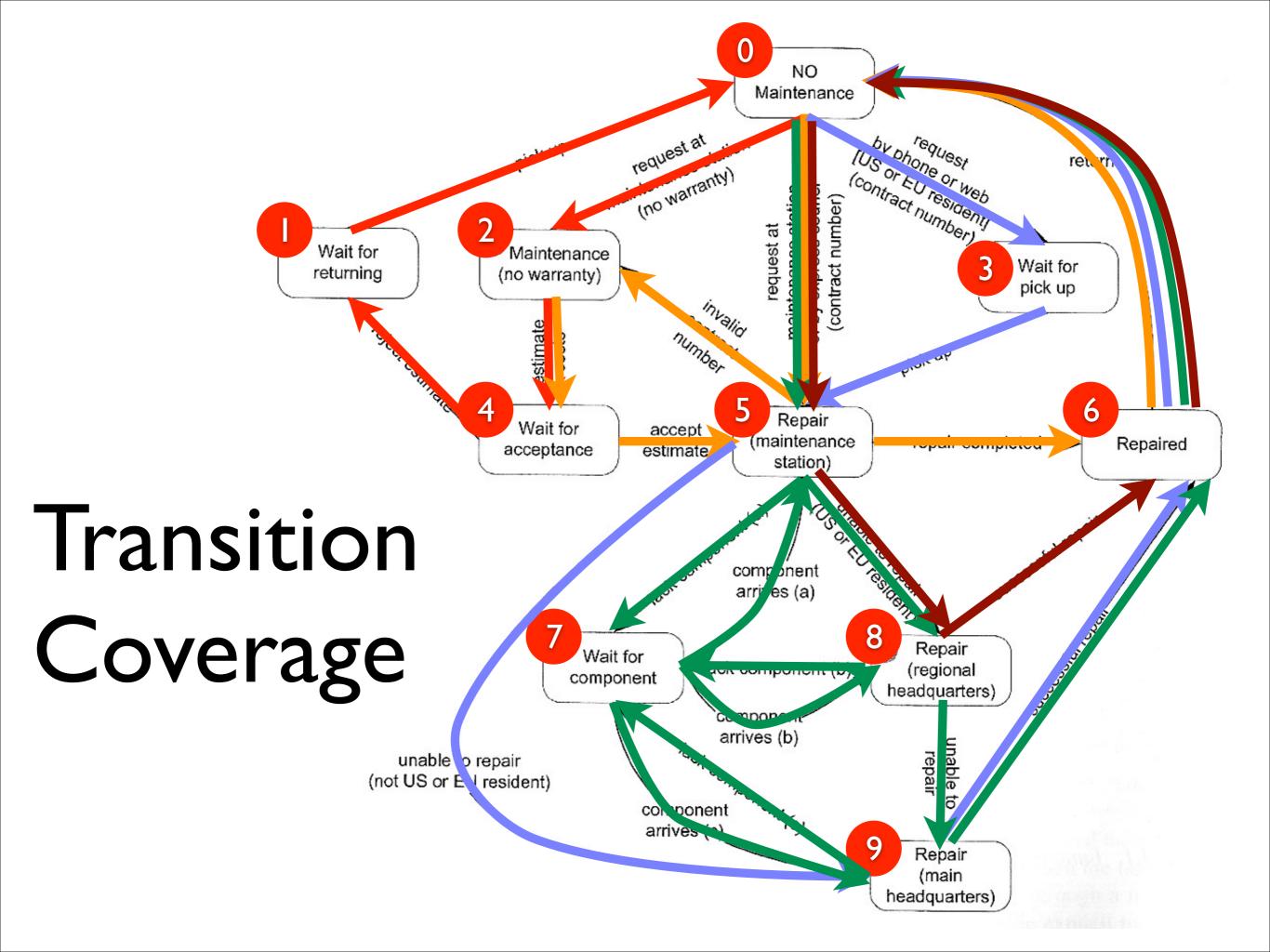
If the maintenance regional headquarters cannot solve the problem, the product is sent to the maintenance main headquarters.

Maintenance is suspended if some components are not available.

Once repaired, the product is returned to the customer.

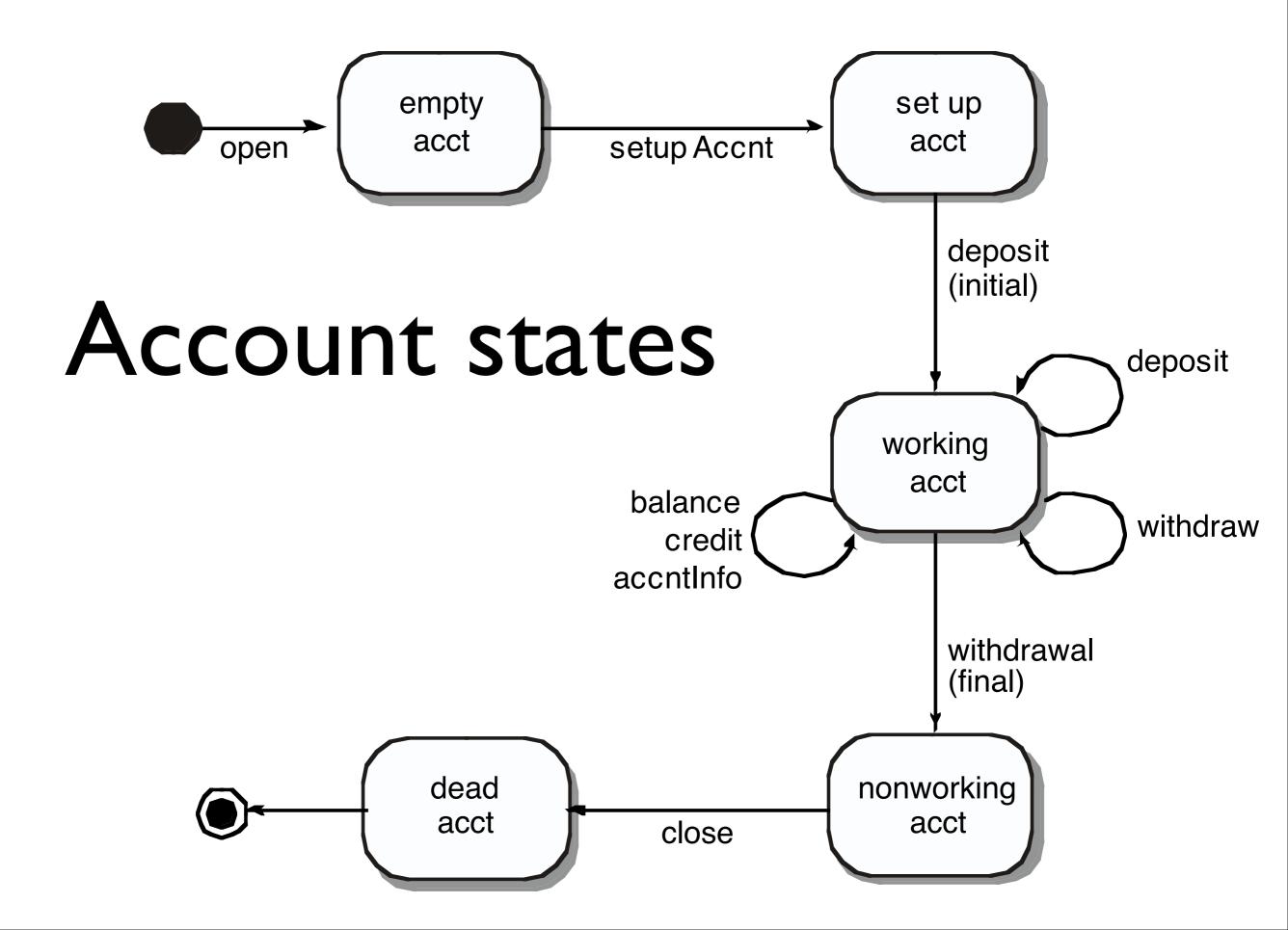
# Coverage Criteria

- Path coverage: Tests cover every path Not feasible in practice due to infinite number of paths
- State coverage: Every node is executed A minimum testing criterion
- Transition coverage: Every edge is executed Typically, a good coverage criterion to aim for



## State-based Testing

- Protocols (e.g., network communication)
- GUIs (sequences of interactions)
- Objects (methods and states)



#### **Decision Tables**

	Education Individual							
Education account	Т	Т	F	F	F	F	F	F
Current purchase > Threshold I	_	_	F	F	Т	Т	_	
Current purchase > Threshold 2	_		—		F	F	Т	Т
Special price < scheduled price	F	Т	F	Т	_		_	
Special price < Tier I					F	Т	_	_
Special price < Tier 2	_		_	_		_	F	Т
Out	Edu discount	Special price	No discount	Special price	Tier I discount	Special price	Tier 2 discount	Special Price

## Condition Coverage

- Basic criterion: Test every column "Don't care" entries (--) can take arbitrary values
- Compound criterion: Test every combination Requires 2<sup>n</sup> tests for n conditions and is unrealistic
- Modified condition decision criterion (MCDC): like basic criterion, but additionally, modify each T/F value at least once such that the outcome changes

Again, a good coverage criterion to aim for

	Educ	ation		Individual				
Education account	F	Т	F	F	F	F	F	F
Current purchase > Threshold I	_		F	F	Т	Т	_	
Current purchase > Threshold 2	—	_	_	_	F	F	Т	Т
Special price < scheduled price	F	Т	F	Т	_		_	_
Special price < Tier I	_	_	_	_	F	Т	_	_
Special price < Tier 2	_		_	_			F	Т
Out	Edu discount	Special price	No discount	Special price	Tier I discount	Special price	Tier 2 discount	Special Price

	Educ	ation	on Individual					
Education account	Т	Т	F	F	F	F	F	F
Current purchase > Threshold I	_	_	F	F	Т	Т		
Current purchase > Threshold 2	—	_	_	_	F	F	Т	Т
Special price < scheduled price	Т	Т	F	Т	_	_	_	—
Special price < Tier I	—	_	_	_	F	Т	_	—
Special price < Tier 2		—	_				F	Т
Out	Edu discount	Special price	No discount	Special price	Tier I discount	Special price	Tier 2 discount	Special Price

	Educ	ation	Individual					
Education account	Т	F	F	F	F	F	F	F
Current purchase > Threshold I	_		F	F	Т	Т	_	
Current purchase > Threshold 2	—		—	_	F	F	Т	Т
Special price < scheduled price	F	Т	F	Т	_		_	_
Special price < Tier I	—		_	_	F	Т	_	
Special price < Tier 2	_		_	_	_		F	Т
Out	Edu discount	Special price	No discount	Special price	Tier I discount	Special price	Tier 2 discount	Special Price

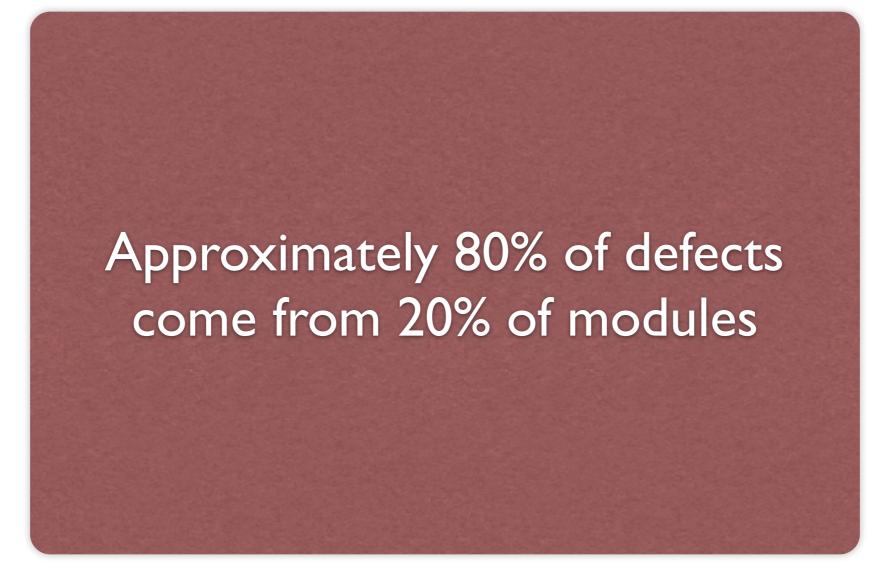
	Educ	ation	Individual					
Education account	Т	Т	F	F	F	F	F	F
Current purchase > Threshold I	_		F	F	Т	Т	_	
Current purchase > Threshold 2	_	_	_	—	F	F	Т	Т
Special price < scheduled price	F	F	F	Т	_	—	_	
Special price < Tier I	_	_	_	_	F	Т	_	_
Special price < Tier 2	_					_	F	Т
Out	Edu discount	Special price	No discount	Special price	Tier I discount	Special price	Tier 2 discount	Special Price

# Weyuker's Hypothesis

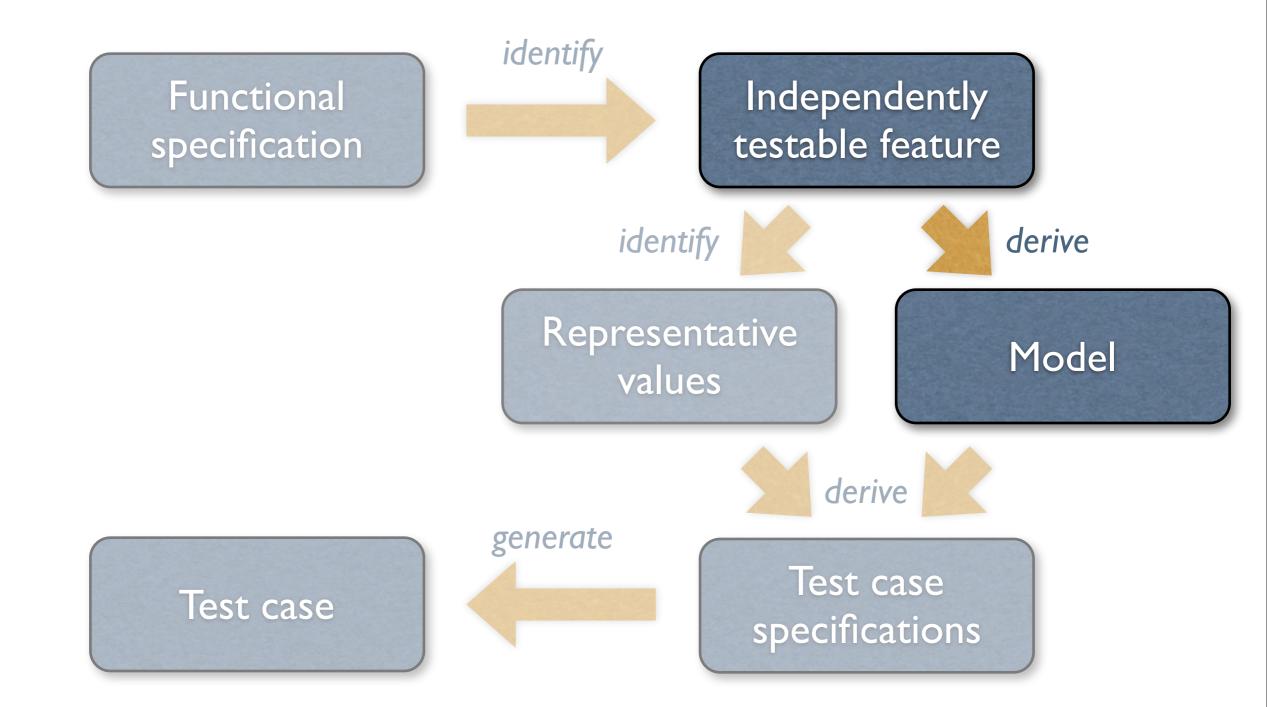
The adequacy of a coverage criterion can only be intuitively defined.

		Mozilla Vulnerabilities			
security	mailnews	content		extensions	nsprpub
nss	base imap	base xslt	t xul	canvas3d webservice python s	pellch pr li
lib	src outil src	src p src	temp doc	src soap pro xpco s	src tests t
libpkix fre bl softolen		XSIt .	inath src st		lec md Littling
pkix_pl_nss mpi el	rn lang	Tror		s ie was in in	src wi uni ma
modu pki sy	s ar n		LEE cont	- Java	
		svg	events xml	xpcom met pre in	ns typ
pkix incl ssl util certd smim	addrbook compose	html content	t src d	xforms sr	misc pthre
top uti r	src src outl src	content doc src	xtf	xmlterm aut w	v s p thr cp md p li
		src src	xbl	base line   sql coo s	io io
crmf pki pkcs de	eud		src can		
ckfw		xpcom	directory	db	ef xpinstall
builtins ca pk11wr pkcs12	mime local news exten	io glue	c-sdk	sqlite3 Com	npiler Utilitie wizard
ns pkil Jar cry	src src palm		Idap	src Code	
S Certhia	bs		libraries clie	i md	Primi zli qa setup uni GUSI
bas asn	mapi dh h	reflect string typelib	libldap exa		Primi ZII Qa
cmd t	old ma db b	xptcal x pu sr xpi x	suncsdk	xp	mac mac
zlib lib m pk si fips pk ce c		src s 🔛 🖬	c-sdk	b mork tri Runtir	m oc Pack
criu bit S	modules	hase tests	Idap	src sr Syste	Secup unit
manager jss s	oji plugin tests sr tools sam		libraries cli i		co D src
org	src test s s def	build compo			
layout		obsolete	intl	editor	toolkit xpfe
generic style xul	C Arr A C base S	c MoreFi thr pr	uconv		onents airbag compone bootstra
base	Ac pu src		ucvlat src uti uc		his s airbag sear boo app
src		widget		Com cre	
	libimg libfont libpr0n zlib png jmcge dec s src	src	ucvcn 🔤		do pa xre m hi
	png jmcge dec s src	mac gtk2	u –	text txtsv	appshel
	src src		unichar locale ct	calendar parser	tools accessible
tables mathml forms	libre libp libb		src src sr		
base base	rdf src src src src	windows os2 beos	chardet	s src src p li	h I II III atk has ht yu
Src Src	src libjar xml s li		src B	libical libic	
svg		PE April dr bu	netwerk	au test dom	le jp ms
base re prin in ht bu	gfx	gtk =====	base protoco	dom	msgsdk cck gc
	src	xlib co g g	src http ft	p base src	C expat muc boehm
js	ps xlib mac theb xlib		src	js2 base	protocol ib c
src tamarin		embedding		Src .	
xpconnect liveco core	qt phot	browser	streamco test co	o re	plugin uriloader camino ipc
src test	gtk windo be xp sh	activex gtk phot	b	ui other-license	oji extha b src ipcd
		src src	cache dns s		
Callib Internet	x11sh xpr	co co web		src	plu pl lib mston view mail
pcre code MM	cairo thebe	plu pl powerp qt	java	7zi	rdf mac src src com
	cairo glitz src		webclient plugga	b browser atk-1.	base chro
	src src	compon qa tests	src_moz wf	components ipeg	province
shell pl e	publi	printin teste mfc w		places migrat	
	libpixma publ	win fi	xpcom do pl	u src boo s	docshell config caps sto gcon mini base s src sr share
jsd	Src	web	te ja u jni		base s src sr chro

#### Pareto's Law



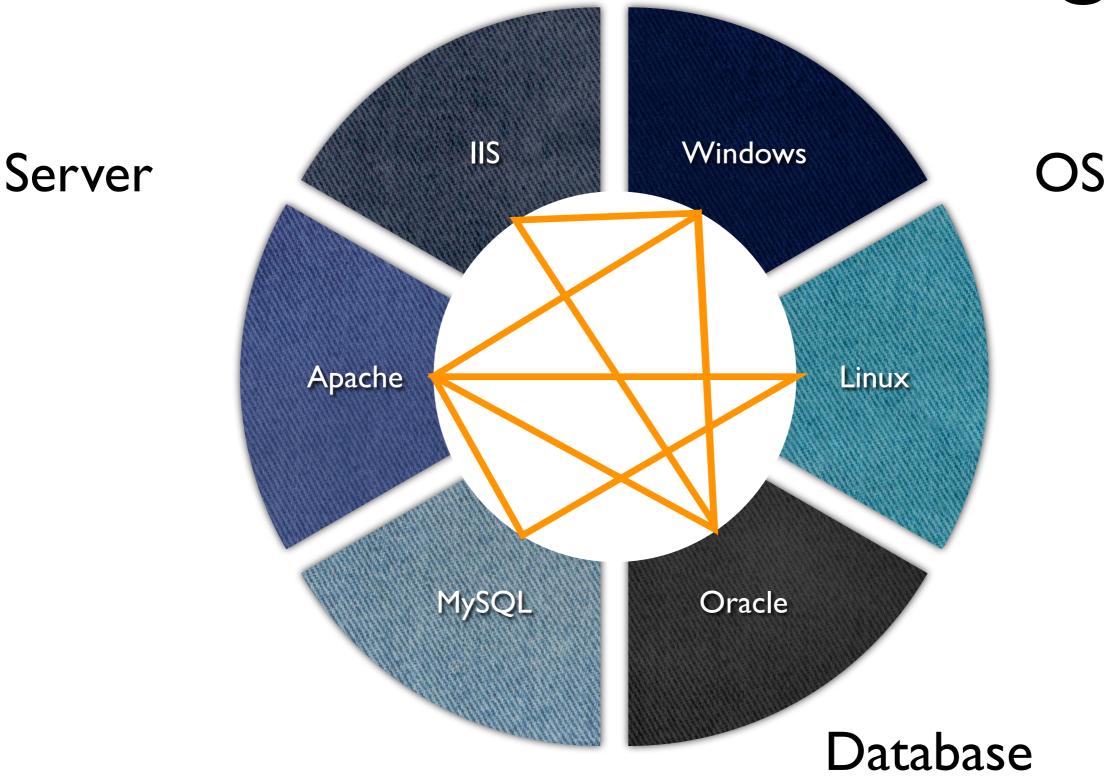
#### Model-Based Testing



# Deriving Test Case Specs

- Input values enumerated in previous step
   specification
   Now pood to take care of combinations
- Now: need to take care of combinations
- Typically, one uses models and representative values to generate
   test cases
   Test case

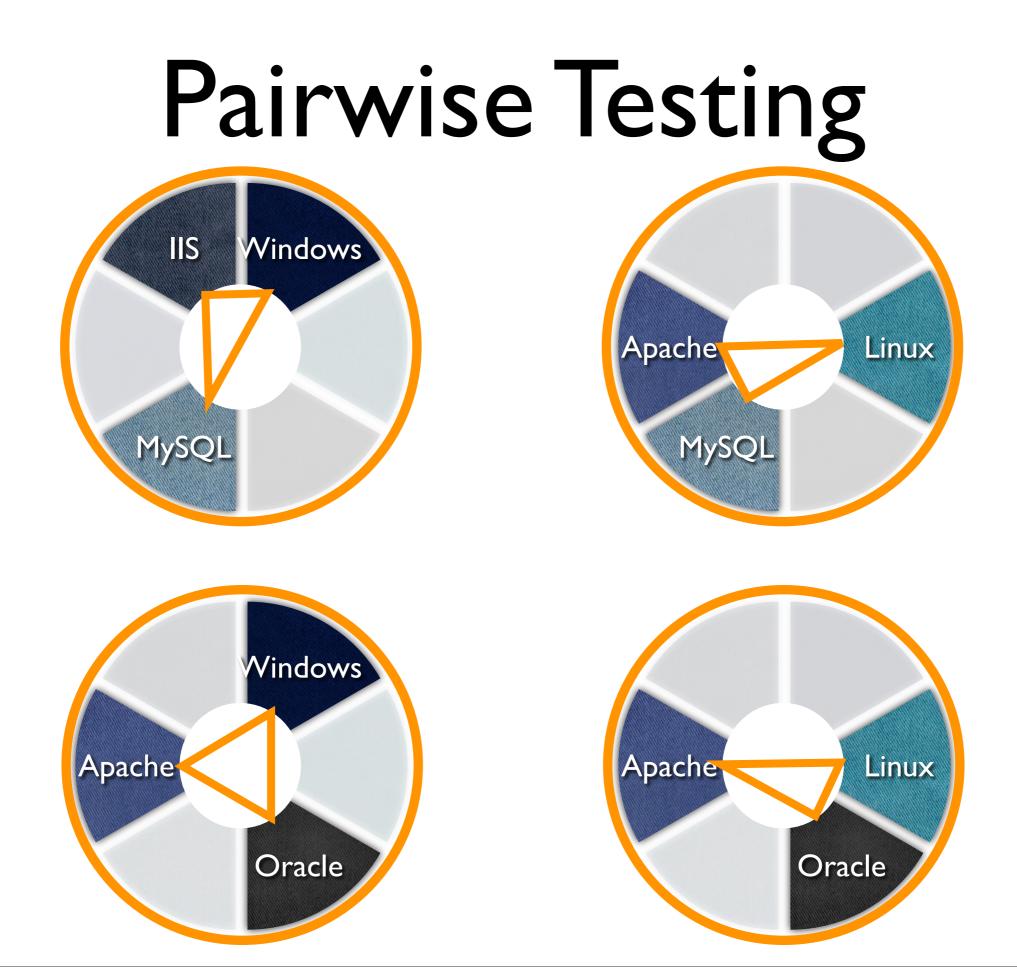
#### **Combinatorial Testing**



# **Combinatorial Testing**

- Eliminate invalid combinations IIS only runs on Windows, for example
- Cover all pairs of combinations such as MySQL on Windows and Linux
- Combinations typically generated automatically

and – hopefully – tested automatically, too



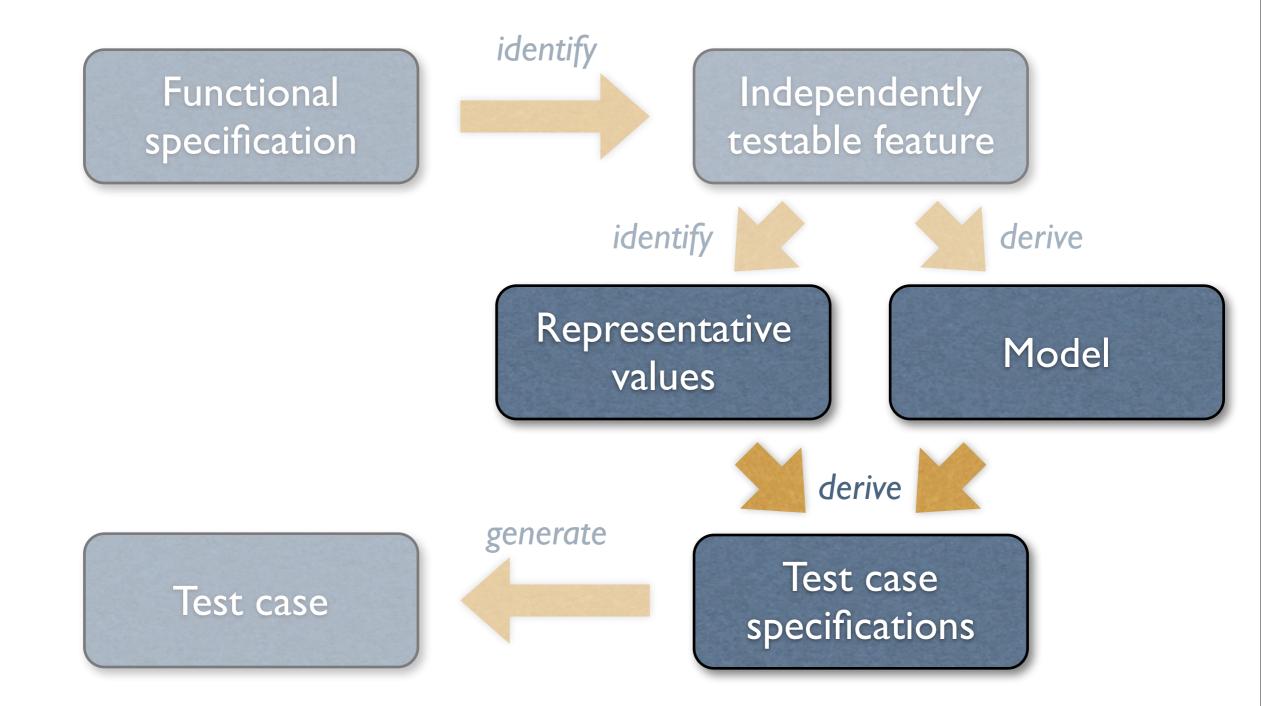
# lesting environment

#### Millions of configurations

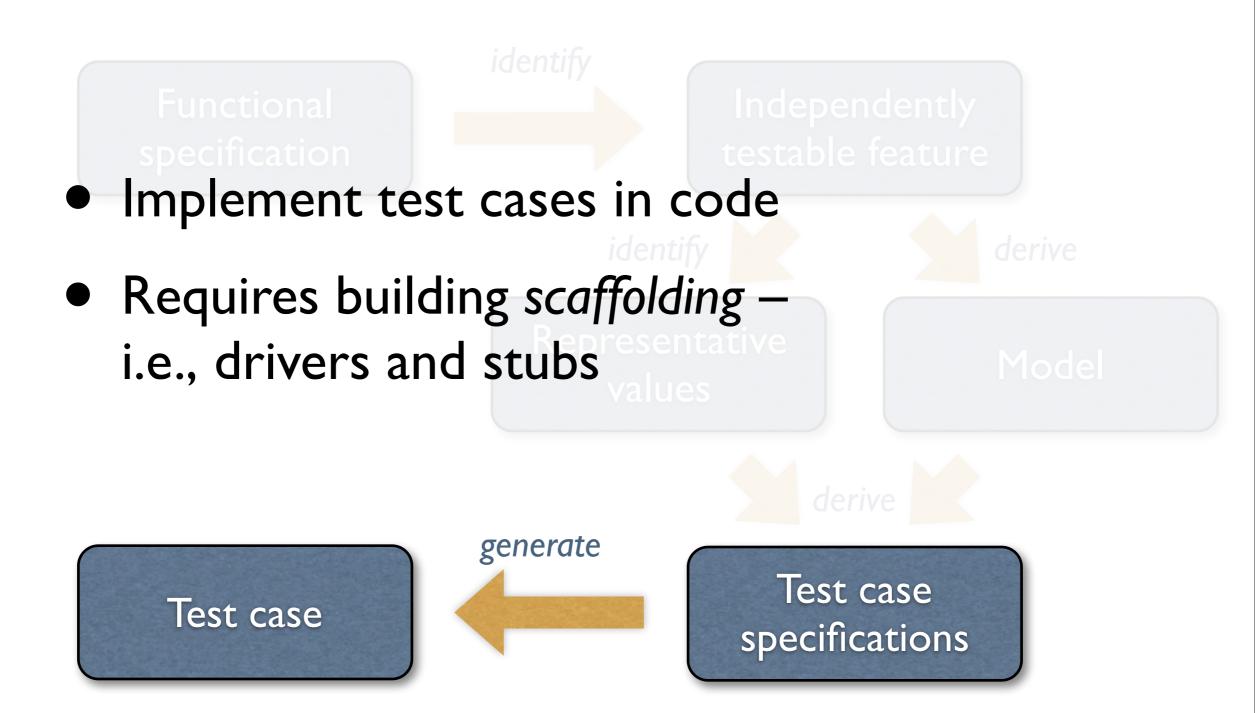
#### Testing on dozens of different machines

All needed to find & reproduce problems

### Deriving Test Case Specs



## Deriving Test Cases



#### Unit Tests

- Directly access units (= classes, modules, components...) at their programming interfaces
- Encapsulate a set of tests as a single syntactical unit
- Available for all programming languages (JUNIT for Java, CPPUNIT for C++, etc.)

# Running a Test

A test case...

- 1. sets up an environment for the test
- 2. tests the unit
- 3. tears down the environment again.

# Testing a URL Class

http://www.askigor.org/status.php?id=sample



```
import junit.framework.Test;
import junit.framework.TestCase;
import junit.framework.TestSuite;
```

```
public class URLTest extends TestCase {
    private URL askigor_url;
```

```
// Create new test
public URLTest(String name) { super(name); }
```

// Assign a name to this test case
public String toString() { return getName(); }

```
// Setup environment
protected void setUp() {
    askigor_url = new URL("http://www.askigor.org/" +
                                 "status.php?id=sample"); }
// Release environment
protected void tearDown() { askigor_url = null;}
```

```
// Test for protocol (http, ftp, etc.)
public void testProtocol() {
  assertEquals(askigor_url.getProtocol(), "http");
}
                                          This functional test
// Test for host
                                          can be used
                                          as a specification!
public void testHost() {
  int noPort = -1;
  assertEquals(askigor_url.getHost(), "www.askigor.org");
  assertEquals(askigor_url.getPort(), noPort);
}
```

```
// Test for path
public void testPath() {
   assertEquals(askigor_url.getPath(), "/status.php");
}
```

```
// Test for query part
public void testQuery() {
   assertEquals(askigor_url.getQuery(), "id=sample");
}
```

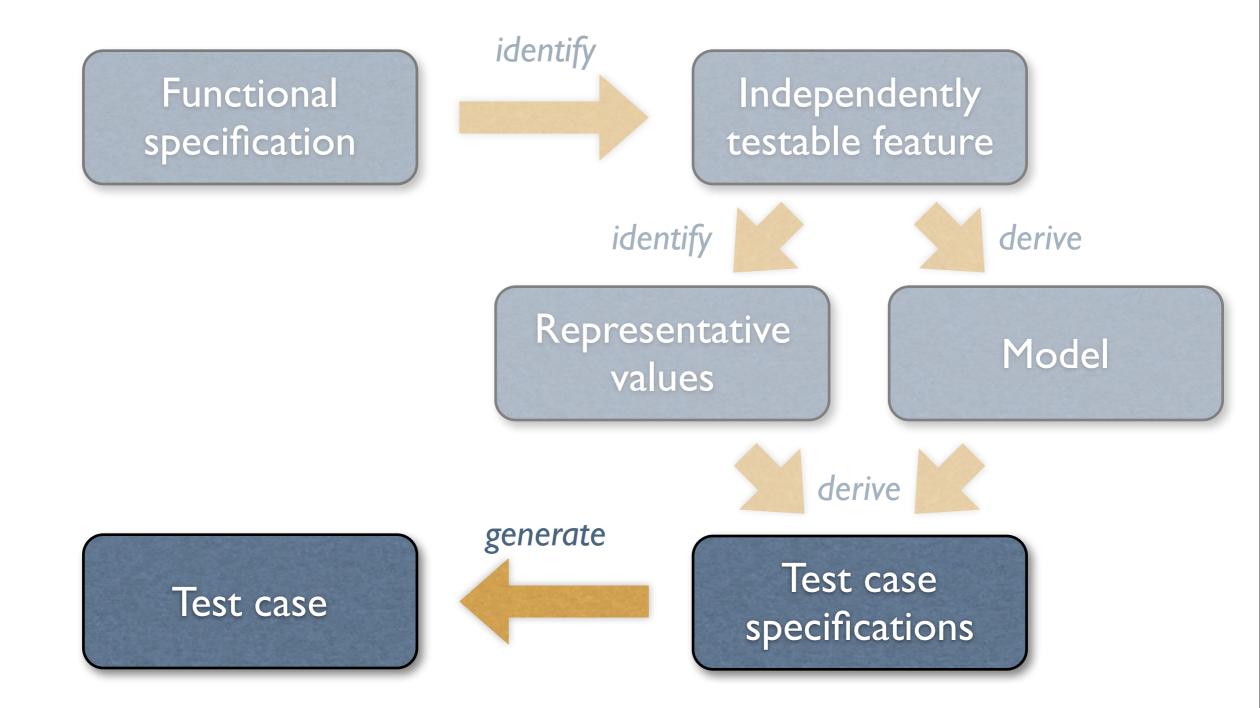
```
// Set up a suite of tests
public static Test suite() {
    TestSuite suite = new TestSuite(URLTest.class);
    return suite;
}
// Main method: Invokes GUI
public static void main(String args[]) {
    String[] testCaseName =
        { URLTest.class.getName() };
    // junit.textui.TestRunner.main(testCaseName);
    junit.swingui.TestRunner.main(testCaseName);
    // junit.awtui.TestRunner.main(testCaseName);
}
```

}

# JUnit

JUnit JUnit		
JUnit		
Test class name:		
URLTest	JUnit JUnit	<u> </u>
	JUnit	
Reload classes every run	Test class name:	
	URLTest 👻	Run
Runs: 4/4 × Errors: 0 × Failures	Reload classes every run	
Results:		_ I
URLTest		JU
L okcrest	Runs: 4/4 <sup>X</sup> Errors: 0 <sup>X</sup> Failures: 1	
— ✓ testHoto	Results:	
− 🖌 testPath	URLTest	Run
_ ✓ testQuery	L order est ↓ ↓ ✓ testProtocol	
🗙 Failures 🛛 🏄 Test Hierarchy	− 🗸 testHost	
	- X testPath	
	📃 🚽 🗸 testQuery	
	🗙 Failures 🛛 🏄 Test Hierarchy	
•	junit.framework.ComparisonFailure: expected:  but was: <>  at URLTest.testPath(URLTest.java:41)	
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)	
Finished: 0.041 seconds	at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccess	
	et sun reflect DelegatingMethodAccessorImpLinvoke/DelegatingMetho	
	Finished: 0.102 seconds	Exit
	Finished. 0.102 seconds	

### Deriving Test Cases



### Systematic Functional Testing

